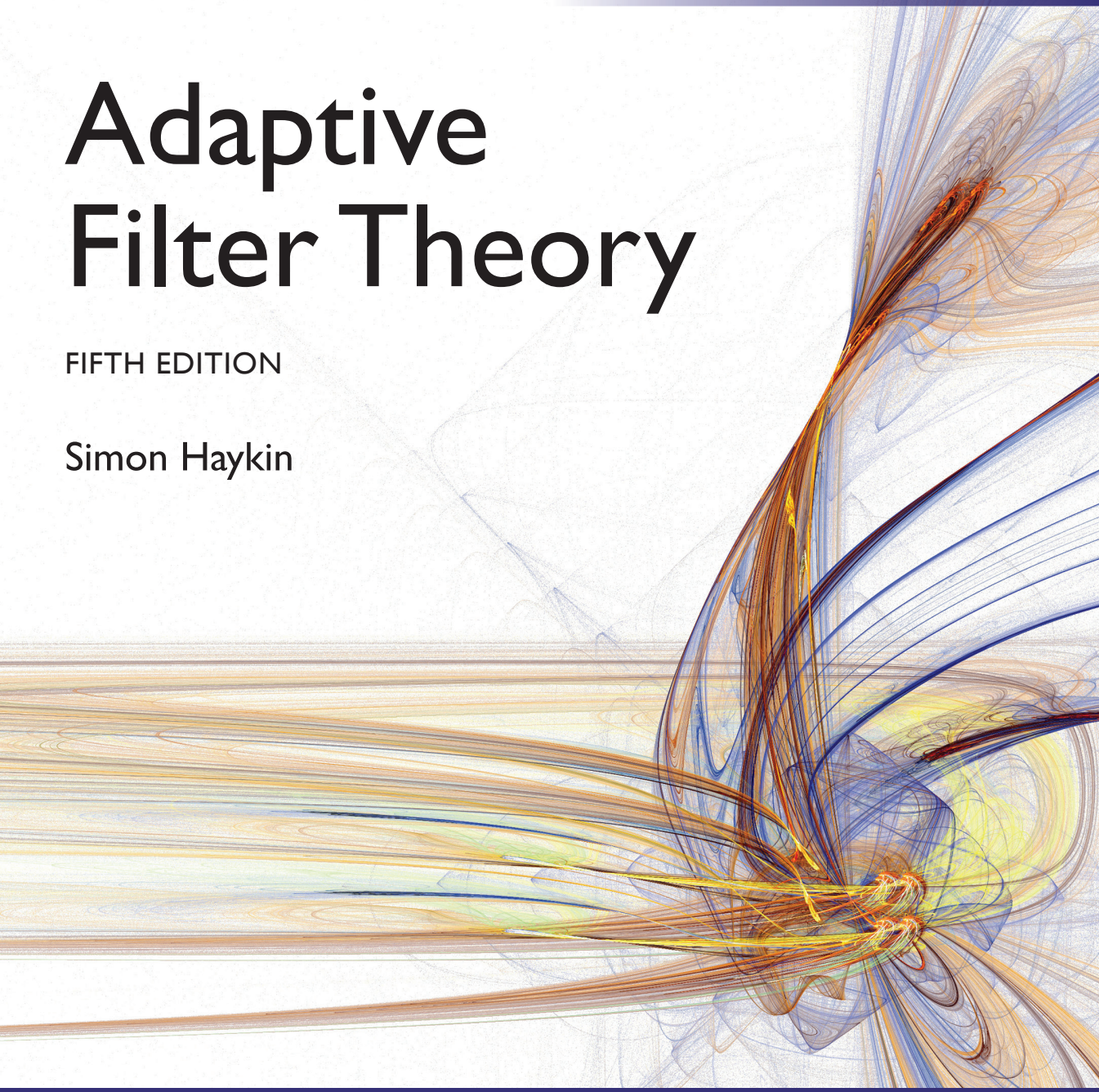
INTERNATIONAL EDITION

Adaptive Filter Theory

FIFTH EDITION

Simon Haykin



ALWAYS LEARNING

PEARSON

ADAPTIVE FILTER THEORY

Fifth Edition

Simon Haykin
Communications Research Laboratory
McMaster University
Hamilton, Ontario, Canada

International Edition contributions by

Telagarapu Prabhakar
Department of Electronics and Communication Engineering
GMR Institute of Technology
Rajam, Andhra Pradesh, India

PEARSON

Upper Saddle River Boston Columbus San Francisco New York
Indianapolis London Toronto Sydney Singapore Tokyo Montreal
Dubai Madrid Hong Kong Mexico City Munich Paris Amsterdam Cape Town

Vice President and Editorial Director, ECS: *Marcia J. Horton*
Senior Editor: *Andrew Gilfillan*
Associate Editor: *Alice Dworkin*
Editorial Assistant: *William Opaluch*
Senior Managing Editor: *Scott Disanno*
Production Liaison: *Irwin Zucker*
Production Editor: *Pavithra Jayapaul, Jouve India*
Publisher, International Edition: *Angshuman Chakraborty*
Publishing Administrator and Business Analyst, International Edition: *Shokhi Shah Khandelwal*
Associate Print and Media Editor, International Edition: *Anuprova Dey Chowdhuri*
Acquisitions Editor, International Edition: *Sandhya Ghoshal*
Publishing Administrator, International Edition: *Hema Mehta*
Project Editor, International Edition: *Daniel Luiz*
Senior Manufacturing Controller, Production, International Edition: *Trudy Kimber*
Operations Specialist: *Linda Sager*
Executive Marketing Manager: *Tim Galligan*
Art Editor: *Greg Dulles*
Art Director: *Jayne Conte*
Cover Designer: *Jodi Notowitz*
Cover Image: *Shutterstock*
Composition/Full-Service Project Management: *Jouve India*

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

Visit us on the World Wide Web at: www.pearsoninternationaleditions.com

Authorized adaptation from the United States edition, entitled Adaptive Filter Theory, 5th edition, ISBN 978-0-132-67145-3, by Simon Haykin, published by Pearson Education © 2014.

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library
10 9 8 7 6 5 4 3 2 1
14 13 12 11 10
Typeset in Times Ten LT Std by Jouve India.
Printed and bound by Courier Westford in The United States of America
The publisher's policy is to use paper manufactured from sustainable forests.

PEARSON

ISBN 10: 0-273-76408-X
ISBN 13: 978-0-273-76408-3

This book is dedicated to

- The many researchers around the world for their contributions to the ever-growing literature on adaptive filtering, and
- The many reviewers, new and old, for their useful inputs and critical comments.

Contents

Preface

Comparisons of the New to the Previous Edition of the Book

Minor and major changes as well as corrections have been made throughout the new edition of the book. Here are the major changes:

1. Chapter 5 on the Method of Stochastic Gradient Descent is new.
2. In Chapter 6 (old Chapter 5) on the Least-Mean-Square (LMS) algorithm, major changes have been made to the statistical learning theory of LMS in light of the Langevin equation and the related Brownian motion.
3. Chapter 11 on Robustness is new.
4. The second half of Chapter 13 on Adaptation in Nonstationary Environments is completely new, being devoted to the Incremental-Delta-Bar-Delta (IDBD) Algorithm and the Autostep Method.
5. Appendices B and F on the Wirtinger Calculus and the Langevin Equation, respectively, are new.
6. The Bibliography is new.
7. The chapters on Adaptive IIR and Complex Neural Networks in the old edition have been deleted.

Introductory Remarks on the New Edition

The subject of adaptive filters constitutes an important part of statistical signal processing. Whenever there is a requirement to process signals that result from operation in an environment of unknown statistics or one that is inherently nonstationary, the use of an adaptive filter offers a highly attractive solution to the problem as it provides a significant improvement in performance over the use of a fixed filter designed by conventional methods. Furthermore, the use of adaptive filters provides new signal-processing capabilities that would not be possible otherwise. We thus find that adaptive filters have been successfully applied in such diverse fields as communications, control, radar, sonar, seismology, and biomedical engineering, among others.

Aims of the Book

The primary aim of this book is to develop the mathematical theory of various realizations of *linear adaptive filters*. Adaptation is accomplished by adjusting the free parameters (coefficients) of a filter in accordance with the input data, which, in reality, makes the adaptive filter nonlinear. When we speak of an adaptive filter being "linear," we mean the following:

> The input-output map of the filter obeys the principle of superposition whenever, at any particular instant of time, the filter's parameters are all fixed.

There is no unique solution to the linear adaptive filtering problem. Rather, we have a "kit of tools" represented by a variety of recursive algorithms, each of which offers desirable features of its own. This book provides such a kit.

In terms of background, it is assumed that the reader has taken introductory undergraduate courses on probability theory and digital signal processing; undergraduate courses on communication and control systems would also be an advantage.

Organization of the Book

The book begins with an introductory chapter, where the operations and different forms of adaptive filters are discussed in general terms. The chapter ends with historical notes, which are included to provide a source of motivation for the interested reader to plough through the rich history of the subject.

The main chapters of the book, 17 in number, are organized as follows:

1. *Stochastic processes and models*, which are covered in Chapter 1. This chapter emphasizes partial characterization (i.e., second-order statistical description) of stationary stochastic processes. As such, it is basic to much of what is presented in the rest of the book.
2. *Wiener filter theory and its application to linear prediction*, which are discussed in Chapters 2 and 3. The Wiener filter, presented in Chapter 2, defines the optimum linear filter for a stationary environment and therefore provides a framework for the study of linear adaptive filters. Linear prediction theory, encompassing both of its forward and backward forms and variants thereof, is discussed in Chapter 3; the chapter finishes with the application of linear prediction to speech coding.
3. *Gradient-descent methods*, which are covered in Chapters 4 and 5. Chapter 4 presents the fundamentals of an old optimization technique known as the *method of steepest descent*, which is of a *deterministic* kind; this method provides the framework for an iterative evaluation of the Wiener filter. In direct contrast, the follow-up chapter, Chapter 5, presents the fundamentals of the *method of stochastic gradient descent*, which is well-suited for dealing with nonstationary matters; the applicability of this second method is illustrated by deriving the least-mean-square (LMS) and gradient adaptive lattice (GAL) algorithms.

4. *Family of LMS algorithms*, which occupies Chapters 6, 7, and 8:
   - Chapter 6 begins with a discussion of different applications of the LMS algorithm, followed by a detailed account of the *small step-size statistical theory*. This new theory, rooted in the Langevin equation of nonequilibrium thermodynamics, provides a fairly accurate assessment of the transient behavior of the LMS algorithm; computer simulations are presented to justify the practical validity of the theory.
   - Chapters 7 and 8 expand on the traditional LMS algorithm by presenting detailed treatments of the normalized LMS algorithm, affine projection adaptive filtering algorithms, and frequency-domain and subband adaptive LMS filtering algorithms. The affine projection algorithm may be viewed as an intermediate between the LMS and recursive least-squares (RLS) algorithms; the latter algorithm is discussed next.
5. *Method of least squares and the RLS algorithm*, which occupy Chapters 9 and 10. Chapter 9 discusses the method of least squares, which may be viewed as the deterministic counterpart of the Wiener filter rooted in stochastic processes. In the method of least squares, the input data are processed on a block-by-block basis; block methods, disregarded in the past because of their numerical complexity, are becoming increasingly attractive, thanks to continuing improvements in computer technology. Chapter 10 builds on the method of least squares to desire the *RLS algorithm*, followed by a detailed statistical theory of its transient behavior.
6. *Fundamental issues*, addressing *robustness* in Chapter 11, *finite-precision effects* in Chapter 12, and *adaptation in nonstationary environments* in Chapter 13:
   - Chapter 11 begins by introducing the H^{∞}*-theory*, which provides the mathematical basis of robustness. With this theory at hand, it is shown that the LMS algorithm is indeed robust in the H^{∞}-sense provided the chosen step-size parameter is small, whereas the RLS algorithm is less robust, when both algorithms operate in a nonstationary environment in the face of internal as well as external disturbances. This chapter also discusses the trade-off between deterministic robustness and statistical efficiency.
   - The theory of linear adaptive filtering algorithms presented in Chapters 5 through 10, is based on continuous mathematics (i.e., infinite precision). When, however, any adaptive filtering algorithm is implemented in digital form, effects due to the use of finite-precision arithmetic arise. Chapter 12 discusses these effects in the digital implementation of LMS and RLS algorithms.
   - Chapter 13 expands on the theory of LMS and RLS algorithms by evaluating and comparing their performances when they operate in a nonstationary environment, assuming a Markov model. The second part of this chapter is devoted to two new algorithms: first, the *incremental delta-bar-delta (IDBD) algorithm*, which expands on the traditional LMS algorithm by vectorizing the step-size parameter, and second, the *Autostep method*, which builds on the IDBD algorithm to experimentally formulate an adaptive procedure that bypasses the need for manual tuning of the step-size parameter.

7. *Kalman filter theory and related adaptive filtering algorithms*, which occupy Chapters 14, 15, and 16:
   - In reality, the RLS algorithm is a special case of the celebrated *Kalman filter*, which is covered in Chapter 14. A distinct feature of the Kalman filter is its emphasis on the notion of a *state*. As mentioned, it turns out that the RLS algorithm is a special case of the Kalman filter; moreover, when the environment is stationary, it also includes the Wiener filter as special case. It is therefore important that we have a good understanding of Kalman filter theory, especially given that covariance filtering and information filtering algorithms are variants of the Kalman filter.
   - Chapter 15 builds on the covariance and information filtering algorithms to derive their respective square-root versions. To be more specific, the ideas of *prearray* and *postarray* are introduced, which facilitate the formulation of a new class of adaptive filtering algorithms structured around systolic arrays whose implementations involve the use of *Givens rotations*.
   - Chapter 16 is devoted to yet another new class of *order-recursive least-squares lattice (LSL) filtering algorithms*, which again build on the covariance and information algorithmic variants of the Kalman filter. For their implementation, they exploit a numerically robust method known as *QR-decomposition*. Another attractive feature of the order-recursive LSL filtering algorithms is the fact that their computational complexity follows a linear law. However, all the nice features of these algorithms are attained at the expense of a highly elaborate framework in mathematical as well as coding terms.
8. *Unsupervised (self-organized) adaptation*, which is featured in the last chapter of the book—namely, Chapter 17 on *blind deconvolution*. The term "blind" is used herein to express the fact that the adaptive filtering procedure is performed *without* the assistance of a desired response. This hard task is achieved by exploiting the use of a model that appeals to the following notions:
   - *Subspace decomposition*, covered in the first part of the chapter, provides a clever but mathematically demanding approach for solving the blind equalization problem. To address the solution, use is made of cyclostationarity—an inherent characteristic of communication systems—for finding the second-order statistics of the channel input so as to equalize the channel in an unsupervised manner.
   - *High-order statistics*, covered in the second part of the chapter, can be of an explicit or implicit kind. It is the latter approach that this part of the chapter addresses in deriving a class of blind equalization algorithms, collectively called *Bussgang algorithms*. This second part of this chapter also includes a new blind equalization algorithm based on an information-theoretic approach that is rooted in the *maximum entropy method*.

The main part of the book concludes with an Epilogue that has two parts:

- The first part looks back on the material covered in previous chapters, with some final summarizing remarks on robustness, efficiency, and complexity, and how the

LMS and RLS algorithms feature in the context of these three fundamentally important issues of engineering.

- The second part of the Epilogue looks forward by presenting a new class of nonlinear adaptive filtering algorithms based on the use of kernels (playing the role of a hidden layer of computational units). These kernels are rooted in the *reproducing kernel Hilbert space (RKHS)*, and the motivation here is to build on material that is well developed in the machine literature. In particular, attention is focused on *kernel LMS filtering*, in which the traditional LMS algorithm plays a key role; the attributes and limitations of this relatively new way of thinking about adaptive filtering are briefly discussed.

The book also includes appendices on the following topics:

- Complex variable theory
- Wirtinger Calculus
- Method of Lagrange multipliers
- Estimation theory
- Eigenanalysis
- The Langevin equation
- Rotations and reflections
- Complex Wishart distribution

In different parts of the book, use is made of the fundamental ideas presented in these appendices.

Ancillary Material

- A Glossary is included, consisting of a list of definitions, notations and conventions, a list of abbreviations, and a list of principal symbols used in the book.
- All publications referred to in the text are compiled in the Bibliography. Each reference is identified in the text by the name(s) of the author(s) and the year of publication. A Suggested Readings section is also included with many other references that have been added for further reading.

Examples, Computer Experiments, and Problems

Many examples are included in different chapters of the book to illustrate concepts and theories under discussion.

The book also includes many computer experiments that have been developed to illustrate the underlying theory and applications of the LMS and RLS algorithms. These experiments help the reader to compare the performances of different members of these two families of linear adaptive filtering algorithms.

Each chapter of the book, except for the introductory chapter, ends with problems that are designed to do two things:

- Help the reader to develop a deeper understanding of the material covered in the chapter.
- Challenge the reader to extend some aspects of the theory discussed in the chapter.

Solutions Manual

The book has a companion solutions manual that presents detailed solutions to all the problems at the end of Chapters 1 through 17 of the book. A copy of the manual can be obtained by instructors who have adopted the book for classroom use by writing directly to the publisher.

The MATLAB codes for all the computer experiments can be accessed by going to the web site http://www.pearsoninternationaleditions.com/haykin/.

Two Noteworthy Symbols

Typically, the square-root of minus one is denoted by the italic symbol *j*, and the differential operator (used in differentiation as well as integration) is denoted by the italic symbol *d*. In reality, however, both of these terms are operators, each in its own way; it is therefore incorrect to use italic symbols for their notations. Furthermore, the italic symbol *j* and the italic symbol *d* are also frequently used as indices to represent other matters, thereby raising the potential for confusion. Accordingly, throughout the book, the *roman* symbol j and the *roman* symbol d are used to denote the square root of minus one and the differential operator, respectively.

Use of the Book

The book is written at a level suitable for use in graduate courses on adaptive signal processing. In this context, it is noteworthy that the organization of the material covered in the book offers a great deal of flexibility in the selection of a suitable list of topics for such a graduate course.

It is hoped that the book will also be useful to researchers and engineers in industry as well as government establishments, working on problems relating to the theory and applications of adaptive filters.

Simon Haykin
Ancaster, Ontario,
Canada

Acknowledgments

I would like to thank several colleagues who helped me in one form or another while writing this book:

1. Prof. Babak Hassibi, Department of Electrical Engineering, California Institute of Technology (Caltech), Pasadena, California, for reading through Chapter 11 on robustness and for numerous comments and suggestions.
2. Mr. Ashique Rupam Mahmood, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, for teaching me the merits of his newly developed procedure, the Autostep method, to train an adaptive filtering algorithm with no manual tuning and for contributing experiments as well as end-of-chapter problems in Chapter 13.
3. Prof. Jose Principe, Department of Electrical and Computer Engineering, University of Florida, Gainesville, Florida, for his many contributions to the kernel least-mean-square (LMS) algorithm in the Epilogue.
4. Prof. Phillip Regalia, Department of Electrical Engineering and Computer Science, Catholic University of America, Washington, D.C., for his continued interest in adaptive filtering and for sharpening Problem 10 in Chapter 16 on order-recursive least-squares lattice filtering.
5. Prof. Richard Sutton, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, for introducing me to his novel incremental delta-bar-delta (IDBD) algorithm.
6. Dr. Peyman Setoodeh, Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ontario, Canada, for taking care of some new computer experiments in Chapter 6 on the LMS algorithm.
7. Prof. Sergios Theodoridis, Department of Informatics and Telecommunications, University of Athens, Athens, Greece, for many constructive comments on the attributes and limitations of the kernel LMS algorithm in the Epilogue.
8. Dr. Weifeng Liu, formerly at the University of Florida and presently at the financial company Jump Trading, LLC, Chicago, Illinois, for many comments on the kernel LMS algorithm in the Epilogue and for contributing to the computer experiment therein.

I am indebted to all of them for enriching the book through their individual contributions.

I am also indebted to Wesley Morrison for copyediting the manuscript of the book, from beginning to end, in the course of which he raised numerous queries. This book is in much better shape than it would have been otherwise.

Next, turning to Pearson, I express my sincere thanks to Irwin Zucker for his limitless energy and enthusiasm as the connecting link between myself and the publisher. I also thank Alice Dworkin, with whom I have worked for the past 15 years (and possibly more), and it has been a pleasurable relationship for all that time. Thanks are due as well to the Senior Editor, Andrew Gilfillan, and Managing Editor, Scott Disanno for their kind help and continuous support.

Last but by no means least, I thank Pavithra Jayapaul for her tireless effort to publish the book in its best form.

Simon Haykin
Ancaster, Ontario,
Canada

The publishers wish to thank S. Sakthivel Murugan of SSN College of Engineering for reviewing the content of the International Edition.

Background and Preview

1. THE FILTERING PROBLEM

The term *estimator* or *filter* is commonly used to refer to a system that is designed to extract information about a prescribed quantity of interest from noisy data. With such a broad aim, estimation (filtering) theory finds applications in many diverse fields: communications, radar, sonar, navigation, seismology, biomedical engineering, and financial engineering, among others. Consider, for example, a *digital communication system*, the basic form of which consists of a transmitter, channel, and receiver connected together as shown in Fig. 1. The function of the transmitter is to convert a message signal (consisting of a sequence of symbols, 1's and 0's) generated by a digital source (e.g., a computer) into a waveform suitable for transmission over the channel. Typically, the channel suffers from two major kinds of impairments:

- *Intersymbol interference.* Ideally, the impulse response of a linear transmission medium is defined by

$$h(t) = A\delta(t - \tau), \tag{1}$$

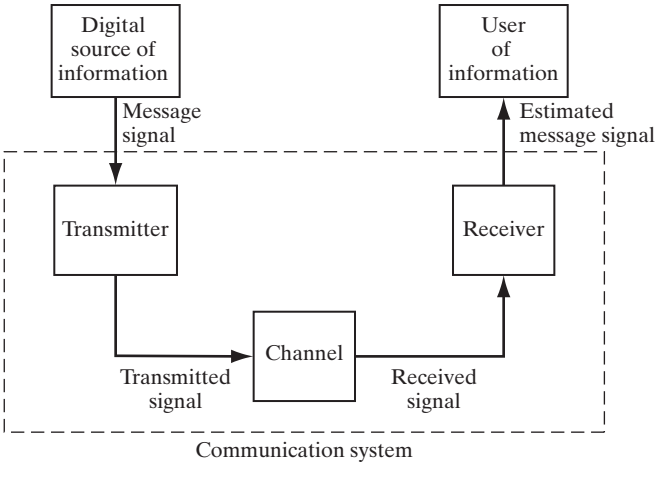


FIGURE 1 Block diagram of a communication system.

where t denotes continuous time, $h(t)$ designates the impulse response, A is an amplitude-scaling factor, $\delta(t)$ is the Dirac delta function (or unit impulse function), and τ denotes the propagation delay incurred in the course of transmitting the signal over the channel. Equation (1) is the time-domain description of an ideal transmission medium. Equivalently, we may characterize it in the frequency domain by writing

$$H(j\omega) = A \exp(-j\omega\tau), \tag{2}$$

where j is the square root of –1, ω denotes angular frequency, $H(j\omega)$ is the frequency response of the transmission medium, and exp $(\cdot)$ stands for the exponential function. In practice, it is impossible for any physical channel to satisfy the stringent requirements embodied in the idealized time-domain description given by Eq. (1) or the equivalent frequency-domain description set forth in Eq. (2): The best that we can do is to approximate Eq. (2) over a band of frequencies representing the essential spectral content of the transmitted signal, which makes the physical channel *dispersive*. In a digital communication system, this channel impairment gives rise to *intersymbol interference*—a smearing of the successive pulses (representing the transmitted sequence of 1's and 0's) into one another with the result that they are no longer distinguishable.

- *Noise.* Some form of noise is present at the output of every communication channel. The noise can be internal to the system, as in the case of thermal noise generated by an amplifier at the front end of the receiver, or external to the system due to interfering signals originating from other sources.

The net result of the two impairments is that the signal received at the channel output is a noisy and distorted version of the signal that is transmitted. The function of the receiver is to operate on the received signal and deliver a reliable *estimate of the original message signal to a user* at the output of the system.

As another example involving the use of filter theory, consider the situation depicted in Fig. 2, which shows a continuous-time dynamic system whose *state* at time t is denoted by the multidimensional vector $\mathbf{x}(t)$. The equation describing evolution of the state $\mathbf{x}(t)$ is usually subject to system errors. The filtering problem is complicated by the fact that $\mathbf{x}(t)$ is hidden and the only way it can be observed is through indirect measurements whose equation is a function of the state $\mathbf{x}(t)$ itself. Moreover, the measurement equation is subject to unavoidable noise of its own. The dynamic system depicted in Fig. 2 may be an aircraft in flight, in which case the position and velocity of the aircraft constitute the elements of the state $\mathbf{x}(t)$, and the measurement system may be a tracking radar. In any event, given the observable vector $\mathbf{y}(t)$ produced by the measuring system

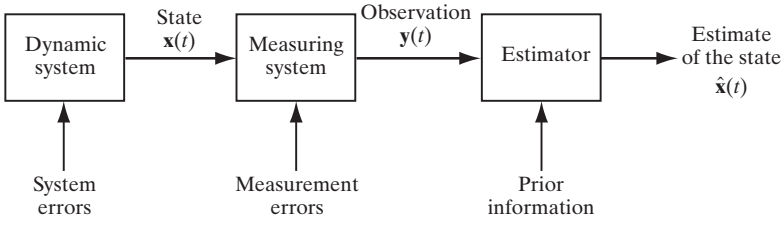


FIGURE 2 Block diagram depicting the components involved in state estimation, namely $\hat{\mathbf{x}}(t)$.

over the interval $[0, T]$, and given prior information, the requirement is to *estimate the state* $\mathbf{x}(t)$ of the dynamic system.

Estimation theory, illustrated by the two examples just described, is *statistical* in nature because of the unavoidable presence of noise or system errors contaminating the operation of the system being studied.

Three Basic Kinds of Estimation

The three basic kinds of information-processing operations are filtering, smoothing, and prediction, each of which may be performed by an estimator. The differences between these operations are illustrated in Fig. 3:

- *Filtering* is an operation that involves the extraction of information about a quantity of interest at time t by using data measured up to and including t.
- *Smoothing* is an a posteriori (i.e., after the fact) form of estimation, in that data measured after the time of interest are used in the estimation. Specifically, the smoothed estimate at time t' is obtained by using data measured over the interval $[0, t']$, where $t' < t$. There is therefore a delay of $t - t'$ involved in computing the smoothed estimate. The benefit gained by waiting for more data to accumulate is that smoothing can yield a more accurate estimate than filtering.
- *Prediction* is the forecasting side of estimation. Its aim is to derive information about what the quantity of interest will be like at some time $t + \tau$ in the future (for some $\tau > 0$) by using data measured up to and including time t.

From the figure, it is apparent that both filtering and prediction are real-time operations, whereas smoothing is not. By a *real-time operation*, we mean an operation in which the estimate of interest is computed on the basis of data available *now*.

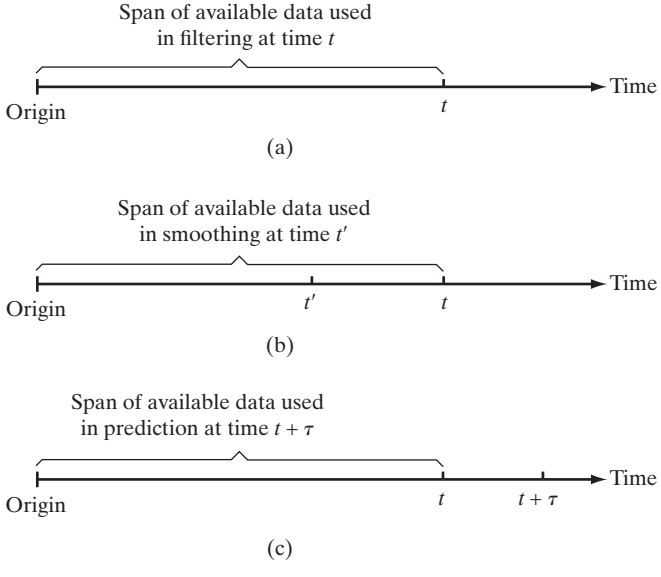


FIGURE 3 Illustrating the three basic forms of estimation: (a) filtering; (b) smoothing; (c) prediction.

2. LINEAR OPTIMUM FILTERS

We may classify filters as linear or nonlinear. A filter is said to be *linear* if the filtered, smoothed, or predicted quantity at the output of the filter is a *linear function of the observations applied to the filter input*. Otherwise, the filter is *nonlinear*.

In the statistical approach to the solution of the *linear filtering problem*, we assume the availability of certain statistical parameters (i.e., *mean and correlation functions*) of the useful signal and unwanted additive noise, and the requirement is to design a linear filter with the noisy data as input so as to minimize the effects of noise at the filter output according to some statistical criterion. A useful approach to this filter-optimization problem is to minimize the mean-square value of the *error signal* defined as the difference between some desired response and the actual filter output. For stationary inputs, the resulting solution is commonly known as the *Wiener filter*, which is said to be *optimum in the mean-square-error sense.* A plot of the mean-square value of the error signal versus the adjustable parameters of a linear filter is referred to as the *error-performance surface*. The minimum point of this surface represents the *Wiener solution.*

The Wiener filter is inadequate for dealing with situations in which *nonstationarity* of the signal and/or noise is intrinsic to the problem. In such situations, the optimum filter has to assume a *time-varying form*. A highly successful solution to this more difficult problem is found in the *Kalman filter*, which is a powerful system with a wide variety of engineering applications.

Linear filter theory, encompassing both Wiener and Kalman filters, is well developed in the literature for *continuous-time* as well as *discrete-time* signals. However, for technical reasons influenced by the wide availability of computers and the ever increasing use of digital signal-processing devices, we find in practice that the discrete-time representation is often the preferred method. Accordingly, in subsequent chapters, we only consider the discrete-time version of Wiener and Kalman filters. In this method of representation, the input and output signals, as well as the characteristics of the filters themselves, are all defined at discrete instants of time. In any case, a continuous-time signal may always be represented by a *sequence of samples* that are derived by observing the signal at uniformly spaced instants of time. No loss of information is incurred during this conversion process provided, of course, we satisfy the well-known *sampling theorem*, according to which the sampling rate has to be greater than twice the highest frequency component of the continuous-time signal. We may thus represent a continuous-time signal $u(t)$ by the sequence $u(n), n = 0, \pm 1, \pm 2, \ldots$, where for convenience we have normalized the sampling period to unity, a practice that we follow throughout the book.

3. ADAPTIVE FILTERS

The design of a Wiener filter requires a priori information about the statistics of the data to be processed. The filter is optimum only when the statistical characteristics of the input data match the a priori information on which the design of the filter is based. When this information is not known completely, however, it may not be possible to design the Wiener filter or else the design may no longer be optimum. A straightforward approach that we may use in such situations is the "estimate and plug" procedure. This

is a two-stage process whereby the filter first "estimates" the statistical parameters of the relevant signals and then "plugs" the results so obtained into a *nonrecursive* formula for computing the filter parameters. For real-time operation, this procedure has the disadvantage of requiring excessively elaborate and costly hardware. To mitigate this limitation, we may use an *adaptive filter*. By such a system we mean one that is *self-designing* in that the adaptive filter relies for its operation on a *recursive algorithm*, which makes it possible for the filter to perform satisfactorily in an environment where complete knowledge of the relevant signal characteristics is not available. The algorithm starts from some predetermined set of *initial conditions*, representing whatever we know about the environment. Yet, in a stationary environment, we find that after successive adaptation cycles of the algorithm it *converges* to the optimum Wiener solution in some statistical sense. In a nonstationary environment, the algorithm offers a *tracking* capability, in that it can track time variations in the statistics of the input data, provided that the variations are sufficiently slow.

As a direct consequence of the application of a recursive algorithm whereby the parameters of an adaptive filter are updated from one adaptation cycle to the next, the parameters become *data dependent*. This, therefore, means that an adaptive filter is in reality a *nonlinear system, in the sense that it does not obey the principle of superposition*. Notwithstanding this property, adaptive filters are commonly classified as linear or nonlinear. An adaptive filter is said to be *linear* if its input–output map obeys the principle of superposition whenever its parameters are held fixed. Otherwise, the adaptive filter is said to be *nonlinear*.

A wide variety of recursive algorithms have been developed in the literature for the operation of linear adaptive filters. In the final analysis, the choice of one algorithm over another is determined by one or more of the following factors:

- *Rate of convergence.* This is defined as the number of adaptation cycles required for the algorithm, in response to stationary inputs, to converge "close enough" to the optimum Wiener solution in the mean-square-error sense. A fast rate of convergence allows the algorithm to adapt rapidly to a stationary environment of unknown statistics.
- *Misadjustment.* For an algorithm of interest, this parameter provides a quantitative measure of the amount by which the final value of the mean-square error, averaged over an ensemble of adaptive filters, deviates from the Wiener solution.
- *Tracking.* When an adaptive filtering algorithm operates in a nonstationary environment, the algorithm is required to *track* statistical variations in the environment. The tracking performance of the algorithm, however, is influenced by two contradictory features: (1) rate of convergence and (2) steady-state fluctuation due to algorithm noise.
- *Robustness.* For an adaptive filter to be *robust*, small disturbances (i.e., disturbances with small energy) can only result in small estimation errors. The disturbances may arise from a variety of factors, internal or external to the filter.
- *Computational requirements.* Here the issues of concern include (a) the number of operations (i.e., multiplications, divisions, and additions/subtractions) required to make one complete adaptation cycle of the algorithm, (b) the size of memory

locations required to store the data and the program, and (c) the investment required to program the algorithm on a computer.

- *Structure.* This refers to the structure of information flow in the algorithm, determining the manner in which it is implemented in hardware form. For example, an algorithm whose structure exhibits high modularity, parallelism, or concurrency is well suited for implementation using very large-scale integration (VLSI).
- *Numerical properties.* When an algorithm is implemented numerically, inaccuracies are produced due to *quantization errors*, which in turn are due to analog-to-digital conversion of the input data and digital representation of internal calculations. Ordinarily, it is the latter source of quantization errors that poses a serious design problem. In particular, there are two basic issues of concern: numerical stability and numerical accuracy. *Numerical stability* is an inherent characteristic of an adaptive filtering algorithm. *Numerical accuracy*, on the other hand, is determined by the number of *bits* (i.e., *bi*nary digi*ts*) used in the numerical representation of data samples and filter coefficients. An adaptive filtering algorithm is said to be *numerically robust* when it is insensitive to variations in the wordlength used in its digital implementation.

These factors, in their own ways, also enter into the design of nonlinear adaptive filters, except for the fact that we now no longer have a well-defined frame of reference in the form of a Wiener filter. Rather, we speak of a nonlinear filtering algorithm that may converge to a local minimum or, hopefully, a global minimum on the error-performance surface.

4. LINEAR FILTER STRUCTURES

The operation of a linear adaptive filtering algorithm involves two basic processes: (1) a *filtering* process designed to produce an output in response to a sequence of input data and (2) an *adaptive* process, the purpose of which is to provide a mechanism for the *adaptive control* of an *adjustable* set of parameters used in the filtering process. These two processes work interactively with each other. Naturally, the choice of a structure for the filtering process has a profound effect on the operation of the algorithm as a whole.

The impulse response of a linear filter determines the filter's memory. On this basis, we may classify linear filters into *finite-duration impulse response* (FIR) and *infinite-duration impulse response* (IIR) filters, which are respectively characterized by *finite memory* and *infinitely long, but fading, memory*.

Linear Filters with Finite Memory

Three types of filter structures distinguish themselves in the context of an adaptive filter with finite memory:

1. FIR filter. Also referred to as a *tapped-delay line filter* or *transversal filter*, the *FIR filter* consists of three basic elements, as depicted in Fig. 4: (a) *a unit-delay element*, (b) *a multiplier*, and (c) *an adder*. The number of delay elements used in the filter determines the finite duration of its impulse response. The number of delay elements,

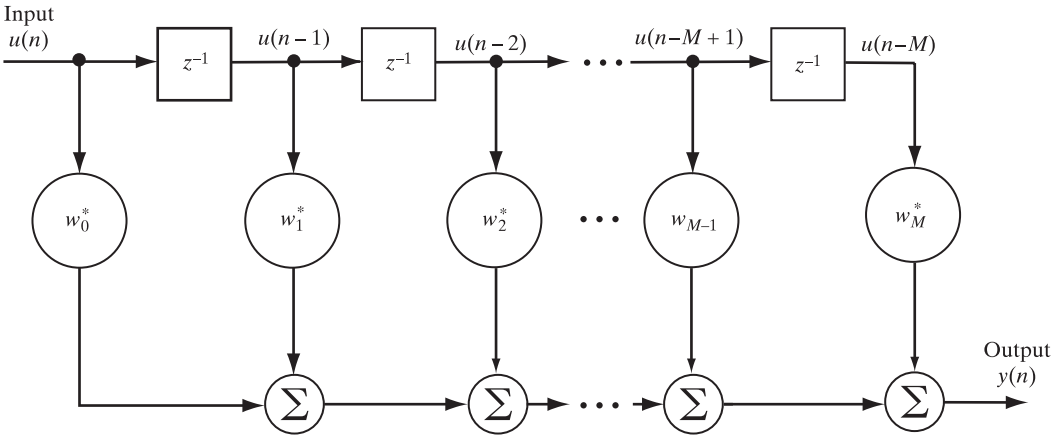


FIGURE 4 FIR filter.

shown as M in the figure, is commonly referred to as the *filter order*. In this figure, the delay elements are each identified by the *unit-delay operator* z^{-1}. In particular, when z^{-1} operates on the input $u(n)$, the resulting output is $u(n-1)$. The role of each multiplier in the filter is to multiply the *tap input* (to which it is connected) by a filter coefficient referred to as a *tap weight*. Thus, a multiplier connected to the kth tap input $u(n-k)$ produces $w_k^* u(n-k)$, where w_k is the respective tap weight and $k = 0, 1, \ldots, M$. The asterisk denotes *complex conjugation*, which assumes that the tap inputs and therefore the tap weights are all *complex valued*. The combined role of the adders in the filter is to sum the individual multiplier outputs and produce an overall response of the filter. For the FIR filter shown, the output is given by

$$y(n) = \sum_{k=0}^{M} w_k^* u(n-k). \tag{3}$$

Equation (3) is called a finite *convolution sum* in the sense that it *convolves* the finite-duration impulse response of the filter, w_n^*, with the filter input $u(n)$ to produce the filter output $y(n)$.

2. Lattice predictor. A *lattice predictor* has a modular structure, in that it consists of a number of individual stages, each of which has the appearance of a lattice—hence the name "lattice" as a structural descriptor. Figure 5 depicts a lattice predictor consisting of M stages; the number M is referred to as the *predictor order*. The mth stage of the lattice predictor shown is described by the pair of input–output relations (assuming the use of complex-valued, wide-sense stationary input data)

$$f_m(n) = f_{m-1}(n) + \kappa_m^* b_{m-1}(n-1) \tag{4}$$

and

$$b_m(n) = b_{m-1}(n-1) + \kappa_m f_{m-1}(n), \tag{5}$$

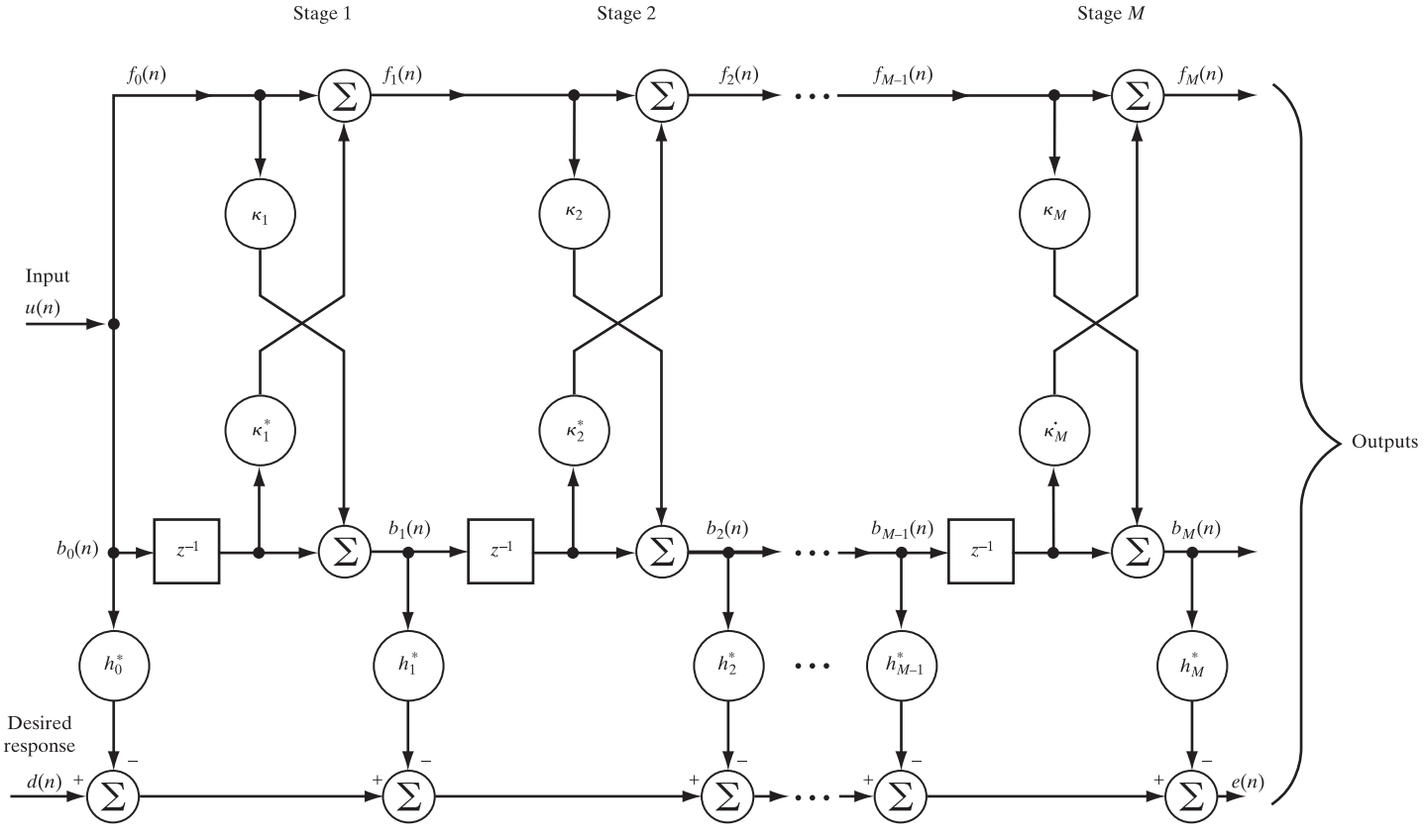


FIGURE 5 Multistage lattice filter.

where $m = 1, 2, \ldots, M$, and M is the *final* predictor order. The variable $f_m(n)$ is the mth *forward prediction error*, and $b_m(n)$ is the mth *backward prediction error*. The coefficient κ_m is called the mth *reflection coefficient*. The forward prediction error $f_m(n)$ is defined as the difference between the input $u(n)$ and its *one-step predicted* value; the latter is based on the set of m *past inputs* $u(n-1), \ldots, u(n-m)$. Correspondingly, the backward prediction error $b_m(n)$ is defined as the difference between the input $u(n-m)$ and its "backward" prediction based on the set of m "future" inputs $u(n), \ldots, u(n-m+1)$. Considering the conditions at the input of stage 1 in the figure, we have

$$f_0(n) = b_0(n) = u(n), \tag{6}$$

where $u(n)$ is the lattice predictor input at time n. Thus, starting with the *initial conditions* of Eq. (6) and given the set of reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$, we may determine the final pair of outputs $f_M(n)$ and $b_M(n)$ by moving through the lattice predictor, stage by stage.

For a *correlated* input sequence $u(n), u(n-1), \ldots, u(n-M)$ drawn from a stationary process, the backward prediction errors $b_0(n), b_1(n), \ldots, b_M(n)$ form a sequence of *uncorrelated* random variables. Moreover, there is a one-to-one correspondence between these two sequences of random variables in the sense that if we are given one of them, we may uniquely determine the other, and vice versa. Accordingly, a linear combination of the backward prediction errors $b_0(n), b_1(n), \ldots, b_M(n)$ may be used to provide an *estimate* of some desired response $d(n)$, as depicted in the lower half of Fig. 5. The difference between $d(n)$ and the estimate so produced represents the estimation error $e(n)$. The process described herein is referred to as a *joint-process estimation*. Naturally, we may use the original input sequence $u(n), u(n-1), \ldots, u(n-M)$ to produce an estimate of the desired response $d(n)$ directly. The indirect method depicted in the figure, however, has the advantage of simplifying the computation of the tap weights $h_0, h_1, \ldots, h_M$ by exploiting the uncorrelated nature of the corresponding backward prediction errors used in the estimation.

3. Systolic array. A *systolic array* represents a *parallel computing* network ideally suited for *mapping* a number of important linear algebra computations, such as *matrix multiplication, triangularization*, and *back substitution*. Two basic types of processing elements may be distinguished in a systolic array: *boundary cells* and *internal cells*. Their functions are depicted in Figs. 6(a) and 6(b), respectively. In each case, the parameter r represents a value *stored* within the cell. The function of the boundary cell is to produce an output equal to the input u divided by the number r stored in the cell. The function of the internal cell is twofold: (a) to multiply the input s (coming in from the top) by the number r stored in the cell, subtract the product rs from the second input (coming in from the left), and thereby produce the difference $u - rs$ as an output from the right-hand side of the cell and (b) to transmit the first input s downward without alteration.

Consider, for example, the 3-by-3 triangular array shown in Fig. 7. This systolic array involves a combination of boundary and internal cells. In this case, the triangular array computes an output vector $\mathbf{y}$ related to the input vector $\mathbf{u}$ by

$$\mathbf{y} = \mathbf{R}^{-\mathrm{T}}\mathbf{u}, \tag{7}$$

where $\mathbf{R}^{-\mathrm{T}}$ is the *inverse* of the transposed matrix $\mathbf{R}^{\mathrm{T}}$. The elements of $\mathbf{R}^{\mathrm{T}}$ are the contents of the respective cells of the triangular array. The zeros added to the inputs of

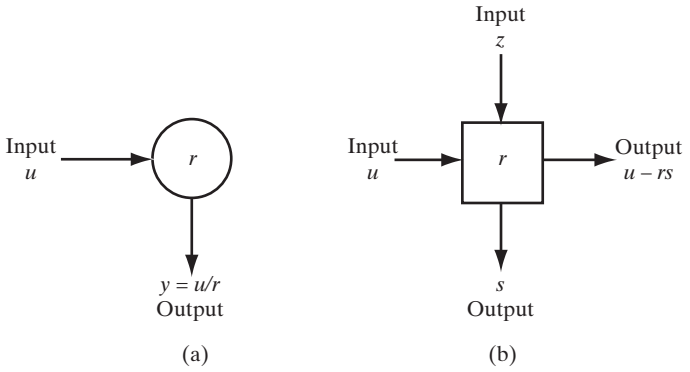


FIGURE 6 Two basic cells of a systolic array: (a) boundary cell; (b) internal cell.

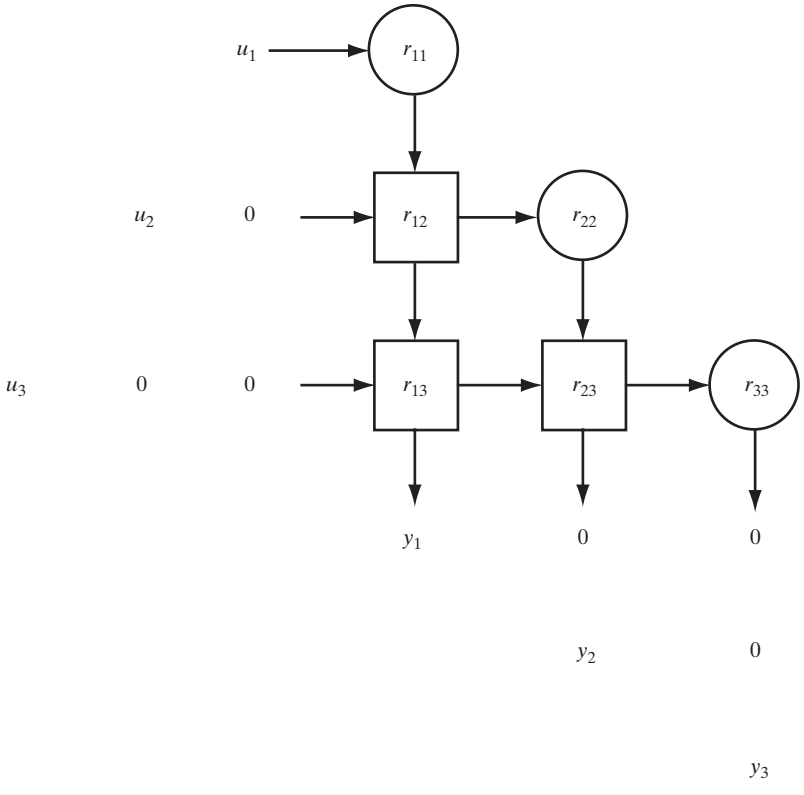


FIGURE 7 Triangular systolic array as example.

the array in the figure are intended to provide the delays necessary for pipelining the computation given by Eq. (7).

A systolic array architecture, as described herein, offers the desirable features of *modularity, local interconnections,* and highly *pipelined* and *synchronized* parallel processing; the synchronization is achieved by means of a global *clock*.

Linear Filters with Infinite Memory

We note that the structure of Fig. 4, the joint-process estimator of Fig. 5 based on a lattice predictor, and the triangular systolic array of Fig. 7 share a common property: All three of them are characterized by an impulse response of finite duration. In other words, they are examples of FIR filters whose structures contain *feedforward* paths only. On the other hand, the structure shown in Fig. 8 is an example of an IIR filter. The feature

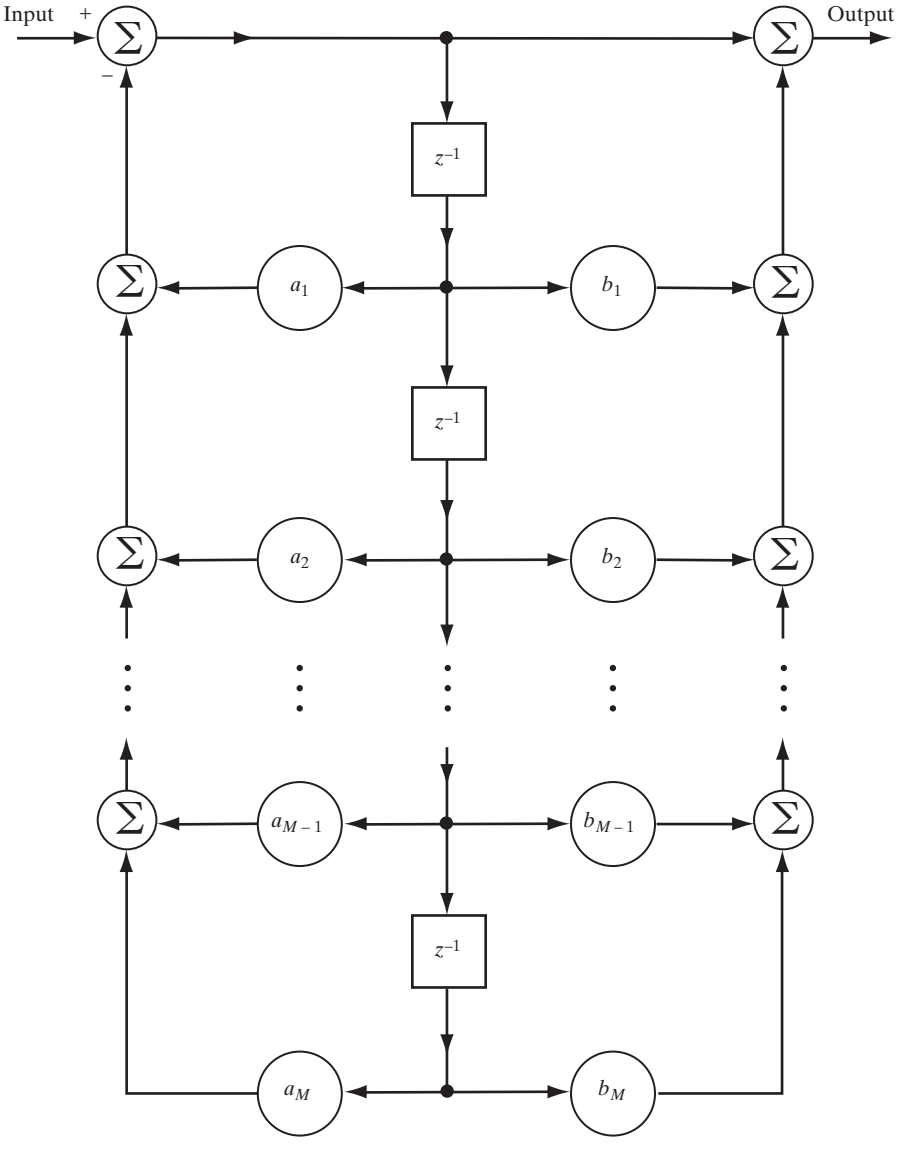


FIGURE 8 IIR filter, assuming real-valued data.

that distinguishes an IIR filter from an FIR filter is the inclusion of *feedback* paths. Indeed, it is the presence of feedback that makes the duration of the impulse response of an IIR filter infinitely long. Furthermore, the presence of feedback introduces a new problem: potential *instability*. In particular, it is possible for an IIR filter to become unstable (i.e., break into oscillation), unless special precaution is taken in the choice of feedback coefficients. By contrast, an FIR filter is inherently *stable*. This explains the popular use of FIR filters, in one form or another, as the structural basis for the design of linear adaptive filters.

5. APPROACHES TO THE DEVELOPMENT OF LINEAR ADAPTIVE FILTERS

There is no unique solution to the linear adaptive filtering problem. Rather, we have a "kit of tools" represented by a variety of recursive algorithms, each of which offers desirable features of its own. The challenge facing the user of adaptive filtering is, first, to understand the capabilities and limitations of various adaptive filtering algorithms and, second, to use this understanding in the selection of the appropriate algorithm for the application at hand.

Basically, we may identify two distinct approaches for deriving recursive algorithms for the operation of linear adaptive filters.

Method of Stochastic Gradient Descent

The stochastic gradient approach uses a tapped-delay line, or FIR filter, as the structural basis for implementing the linear adaptive filter. For the case of stationary inputs, the *cost function*, also referred to as the *index of performance*, is defined as the *mean-square error* (i.e., the mean-square value of the difference between the desired response and the FIR filter output). This cost function is precisely a second-order function of the tap weights in the FIR filter. The dependence of the mean-square error on the unknown tap weights may be viewed to be in the form of a *multidimensional paraboloid* (i.e., a "punch bowl") with a uniquely defined bottom, or *minimum point*. As mentioned previously, we refer to this paraboloid as the *error-performance surface*; the tap weights corresponding to the minimum point of the surface define the optimum Wiener solution.

To develop a recursive algorithm for updating the tap weights of the adaptive FIR filter using the stochastic gradient approach, as the name would imply it, we need to start with a *stochastic cost function*. For such a function, for example, we may use the instantaneous squared value of the error signal, defined as the difference between the externally supplied desired response and the actual response of the FIR filter to the input signal. Then, differentiating this stochastic cost function with respect to the tap-weight vector of the filter, we obtain a gradient vector that is naturally stochastic. With the desire to move towards optimality, adaptation is performed along the "negative" direction of the gradient vector. The adaptive filtering algorithm resulting from this approach may be expressed in words as follows:

$$\begin{pmatrix}\text{updated}\\ \text{tap-weight}\\ \text{vector}\end{pmatrix} = \begin{pmatrix}\text{old}\\ \text{tap-weight}\\ \text{vector}\end{pmatrix} + \begin{pmatrix}\text{learning-}\\ \text{rate}\\ \text{parameter}\end{pmatrix} \times \begin{pmatrix}\text{tap-}\\ \text{input}\\ \text{vector}\end{pmatrix} \times \begin{pmatrix}\text{error}\\ \text{signal}\end{pmatrix}.$$

The *learning-rate parameter* determines the rate at which the adaptation is performed. The recursive algorithm so described is called the *least-mean-square (LMS) algorithm*, which is simple in computational terms yet effective in performance. However, its convergence behavior is slow and difficult to study mathematically.

Method of Least Squares

The second approach to the development of linear adaptive filtering algorithms is based on the *method of least squares*. According to this method, we minimize a cost function that is defined as the *sum of weighted error squares*, where the *error* or *residual* is itself defined as the difference between some desired response and the actual filter output. Unlike the method of stochastic gradient, this minimization is achieved using algebraic matrix manipulations, resulting in an update rule that may be expressed, in words, as follows:

$$\begin{pmatrix}\text{updated}\\ \text{tap-weight}\\ \text{vector}\end{pmatrix} = \begin{pmatrix}\text{old}\\ \text{tap-weight}\\ \text{vector}\end{pmatrix} + \begin{pmatrix}\text{gain}\\ \text{vector}\end{pmatrix} \times (\text{innovation}),$$

where innovation is "new" information put into the filtering process at the updating time. The adaptive filtering algorithm so described is called the *recursive least-squares (RLS) algorithm*. A distinctive property of this second algorithm is its rapid rate of convergence, which is attained at the expense of increased computational complexity.

Two Families of Linear Adaptive Filtering Algorithms

The LMS and RLS algorithms constitute two basic algorithms, around each of which a family of algorithms is formulated. Within each family, the adaptive filtering algorithms differ from each other in the way in which the filtering structure is configured. However, regardless of the filtering structure used around which the adaptation of parameters is performed, the algorithms within each family inherit certain properties rooted in the LMS and RLS algorithms. Specifically:

- LMS-based algorithms are *model independent*, in the sense that there are no statistical assumptions made in deriving them.
- On the other hand, RLS-based algorithms are *model dependent*, in that their derivatives assume the use of a *multivariate Gaussian model*.

The differentiation we have made here has a profound impact on the rate of convergence, tracking, and robustness of the algorithm.

6. ADAPTIVE BEAMFORMING

The adaptive filtering methods and structures discussed thus far are all of a temporal kind, in that the filtering operation is performed in the time domain. Naturally, adaptive filtering may also be of a *spatial* kind, wherein an *array of independent sensors* is placed at different points in space to "listen" to a signal originating from a distant source. In effect, the sensors provide a means of *sampling* the received signal *in space*. The set of

sensor outputs collected at a particular instant of time constitutes a *snapshot* of the source. When the sensors lie uniformly on a straight line, the data snapshot in a spatial filter plays a role analogous to that of a set of consecutive tap inputs that exist in an FIR filter at a particular instant of time.

Important applications that involve the use of array signal processing include the following:

- *Radar*, in which the sensors consist of antenna elements (e.g., dipoles, horns, or slotted waveguides) that respond to incident electromagnetic waves. The requirement here is to detect the source responsible for radiating the electromagnetic waves, estimate the angle of arrival of the waves, and extract information about the source.
- *Sonar*, wherein the sensors consist of hydrophones designed to respond to incident acoustic waves and the signal-processing requirements are of a nature similar to those in radar.
- *Speech enhancement*, in which the sensors consist of microphones and the requirement is, for example, to listen to the voice of a desired speaker in the presence of background noise.

In these applications, we speak of *beamforming*, the purpose of which is to distinguish between the spatial properties of signal and noise. The system used to do the beamforming is called a *beamformer*. The term "beamformer" is derived from the fact that early antennas were designed to *form pencil beams* so as to receive a source signal radiating from a specific direction and to attenuate signals originating from other directions that were of no interest. Note that beamforming applies to the radiation (i.e., transmission) as well as reception of energy.

Figure 9 shows the block diagram of an *adaptive beamformer* that uses a linear array of identical sensors. The sensor outputs, assumed to be in *baseband* form, are individually weighted and then summed to produce the overall beamformer output. The term "baseband" refers to the original band of frequencies within which the source operates. The beamformer has to satisfy two requirements:

- *Steering* capability, whereby the target (source) signal is always protected.
- *Cancellation of interference*, so that the output signal-to-noise ratio is maximized.

One method of satisfying these requirements is to minimize the variance (i.e., average power) of the beamformer output, subject to the constraint that, during the process of adaptation, the M-by-1 weight vector $\mathbf{w}(n)$ satisfies the condition

$$\mathbf{w}^{\mathrm{H}}(n)\,\mathbf{s}(\theta) = 1 \quad \text{for all } n \text{ and } \theta = \theta_t, \tag{8}$$

where $\mathbf{s}(\theta)$ is an M-by-1 *steering vector*. The superscript H denotes Hermitian transposition (i.e., transposition combined with complex conjugation). It is assumed that the baseband data are complex valued—hence the need for complex conjugation. The value of the *electrical angle* $\theta = \theta_t$ is determined by the direction of the target. The angle θ is itself measured with, say, sensor 0 treated as the point of reference.

The dependence of the steering vector on θ is defined with the use of the relationship

$$s(\theta) = [1, e^{-\mathrm{j}\theta}, \ldots, e^{-\mathrm{j}(M-1)\theta}]^{\mathrm{T}}. \tag{9}$$

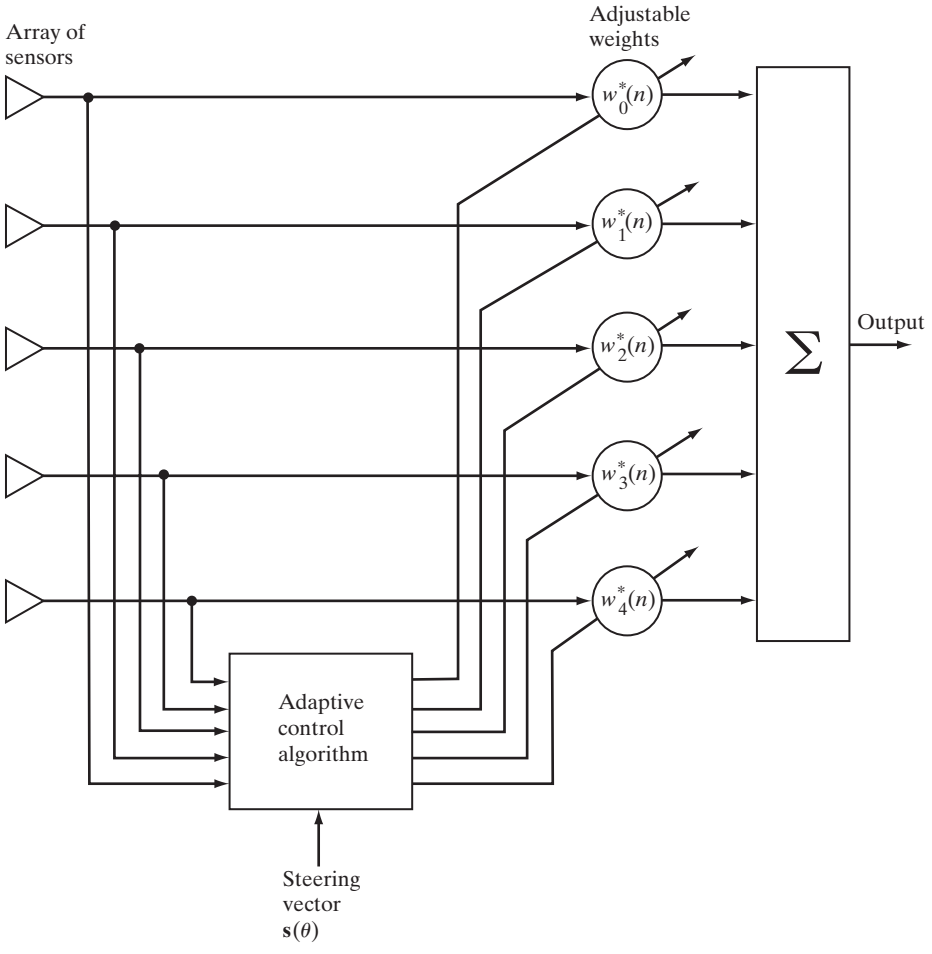


FIGURE 9 Adaptive beamformer for an array of five sensors. The sensor outputs (in baseband form) are complex valued; hence, the weights are complex valued.

Let ϕ denote the *actual angle of incidence* of a plane wave, measured with respect to the normal to the linear array. Then, from Fig. 10, we readily see that

$$\theta = \frac{2\pi d}{\lambda} \sin\phi, \quad -\pi/2 \leq \phi \leq \pi/2, \tag{10}$$

where d is the spacing between adjacent sensors of the array and λ is the wavelength of the incident wave. With ϕ restricted to lie inside the range $[-\pi/2, \pi/2]$ and the permissible values of θ lying inside range $[-\pi, \pi]$, we find, from Eq. (10), that d must be less than $\lambda/2$, so that there is one-to-one correspondence between the values of θ and ϕ without ambiguity. The requirement $d < \lambda/2$ may thus be viewed as the *spatial analog of the sampling theorem*. If this requirement is not satisfied, then the radiation (antenna)

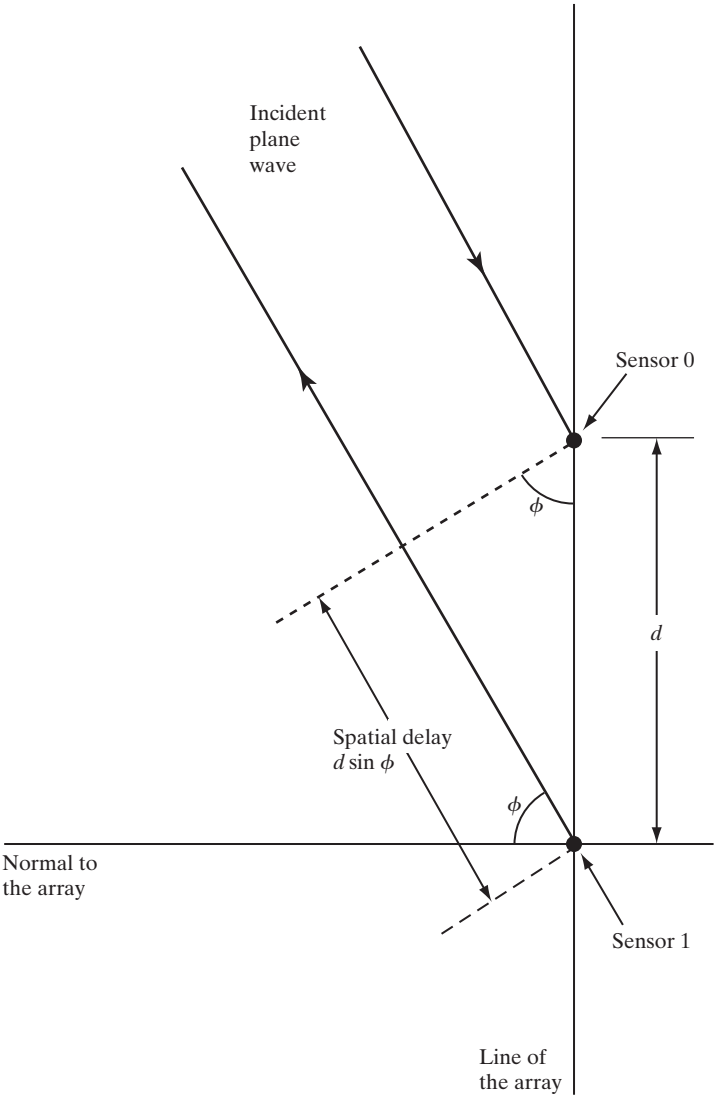


FIGURE 10 Spatial delay incurred when a plane wave impinges on a linear array.

pattern of the beamformer will exhibit *grating lobes*. The *radiation pattern* is a plot of the output power of the beamformer versus the direction along which it is measured.

The imposition of the signal-protection constraint in Eq. (8) ensures that, for the prescribed look direction $\theta = \theta_t$, the response of the array is maintained constant (equal to unity), no matter what values are assigned to the elements of the weight vector $\mathbf{w}$. An algorithm that minimizes the variance of the beamformer output, subject to

this constraint, is referred to as the *minimum-variance distortionless response (MVDR) beamforming algorithm*. Note that the imposition of the signal-protection constraint reduces the available number of *degrees of freedom* to $M - 2$, where M is the number of sensors in the array. Consequently, the number of independent nulls produced by the MVDR algorithm (i.e., the number of independent interferences that can be cancelled) is $M - 2$.

7. FOUR CLASSES OF APPLICATIONS

The ability of an adaptive filter to operate satisfactorily in an unknown environment and track time variations of input statistics makes the adaptive filter a powerful device for signal-processing and control applications. Indeed, adaptive filters have been successfully applied in such diverse fields as communications, control, radar, sonar, seismology, and biomedical engineering. Although these applications are quite different in nature, nevertheless, they have one basic feature in common: An input vector and a desired response are used to compute an estimation error, which is in turn used to control the values of a set of adjustable filter coefficients. The adjustable coefficients may take the form of tap weights, reflection coefficients, or rotation parameters, depending on the filter structure employed. However, the essential difference between the various applications of adaptive filtering arises in the manner in which the desired response is extracted. In this context, we may distinguish four basic classes of adaptive filtering applications, as depicted in Fig. 11. For convenience of presentation, the following notation is used in the figure:

u = input applied to the adaptive filter;
y = output of the adaptive filter;
d = desired response;
$e = d - y$ = estimation error.

The functions of the four basic classes of adaptive filtering applications depicted are as follows:

I. *Identification* [Fig. 11(a)]. The notion of a *mathematical model* is fundamental to science and engineering. In the class of applications dealing with identification, an adaptive filter is used to provide a linear model that represents the best fit (in some sense) to an *unknown plant.* The plant and the adaptive filter are driven by the same input. The plant output supplies the desired response for the adaptive filter. If the plant is dynamic in nature, the model will be time varying.

II. *Inverse modeling* [Fig. 11(b)]. In this second class of applications, the function of the adaptive filter is to provide an *inverse model* that represents the best fit (in some sense) to an *unknown noisy plant*. Ideally, in the case of a linear system, the inverse model has a transfer function equal to the *reciprocal (inverse)* of the plant's transfer function, such that the combination of the two constitutes an ideal transmission medium. A delayed version of the plant (system) input constitutes the desired response for the adaptive filter. In some applications, the plant input is used without delay as the desired response.

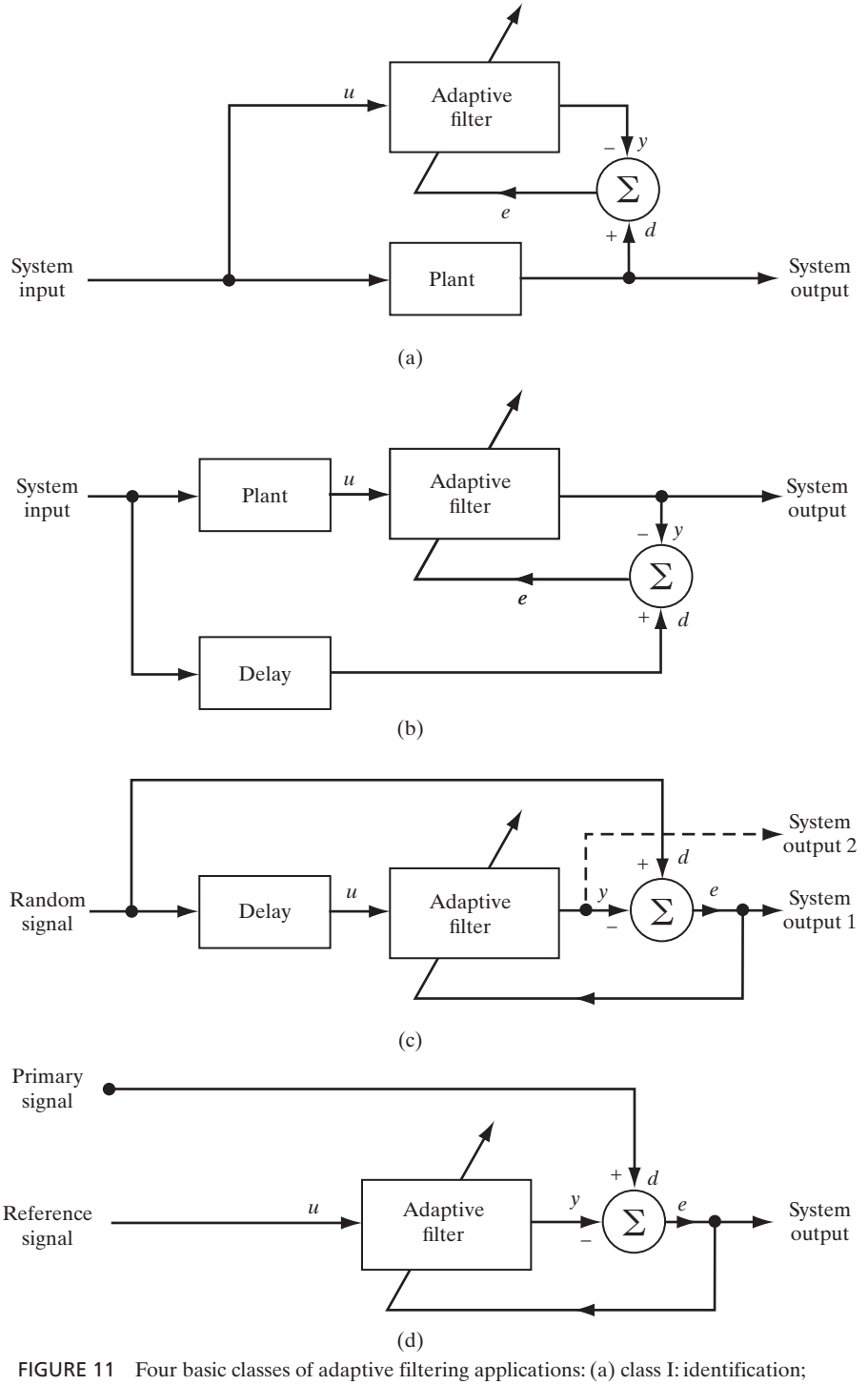


FIGURE 11 Four basic classes of adaptive filtering applications: (a) class I: identification; (b) class II: inverse modeling; (c) class III: prediction; (d) class IV: interference cancellation.

III. *Prediction* [Fig. 11(c)]. Here, the function of the adaptive filter is to provide the best *prediction* (in some sense) of the present value of a random signal. The present value of the signal thus serves the purpose of a desired response for the adaptive filter. Past values of the signal supply the input applied to the filter. Depending on the application of interest, the adaptive filter output or the estimation (prediction) error may serve as the system output. In the first case, the system operates as a *predictor*; in the latter case, it operates as a *prediction-error filter*.

IV. *Interference cancellation* [Fig. 11(d)]. In this final class of applications, the adaptive filter is used to cancel *unknown interference* contained (alongside an information-bearing signal component) in a *primary signal*, with the cancellation being optimized in some sense. The primary signal serves as the desired response for the adaptive filter. A *reference (auxiliary) signal* is employed as the input to the filter. The reference signal is derived from a sensor or a set of sensors located such that it or they supply the primary signal in such a way that the information-bearing signal component is weak or essentially undetectable.

Table 1 lists specific applications that are illustrative of the four basic classes of adaptive filtering applications. The applications listed are drawn from the fields of control systems, seismology, electrocardiography, communications, and radar. Their purposes are briefly described in the last column of the table.

TABLE 1 Applications of Adaptive Filters

Class of adaptive filtering	Application	Purpose
I. Identification	System identification	Given an unknown dynamic system, the purpose of system identification is to design an adaptive filter that provides an approximation to the system.
	Layered earth modeling	In exploration seismology, a layered model of the earth is developed to unravel the complexities of the earth's surface.
II. Inverse modeling	Equalization	Given a channel of unknown impulse response, the purpose of an adaptive equalizer is to operate on the channel output such that the cascade connection of the channel and the equalizer provides an approximation to an ideal transmission medium.
III. Prediction	Predictive coding	The adaptive prediction is used to develop a model of a signal of interest (e.g., a speech signal); rather than encode the signal directly, in predictive coding the prediction error is encoded for transmission or storage. Typically, the prediction error has a smaller variance than the original signal—hence the basis for improved encoding.
	Spectrum analysis	In this application, predictive modeling is used to estimate the power spectrum of a signal of interest.

(*continued*)

TABLE 1 Applications of Adaptive Filters (Continued)

Class of adaptive filtering	Application	Purpose
IV. Interference cancellation	Noise cancellation	The purpose of an adaptive noise canceller is to subtract noise from a received signal in an adaptively controlled manner so as to improve the signal-to-noise ratio. Echo cancellation, experienced on telephone circuits, is a special form of noise cancellation. Noise cancellation is also used in electrocardiography.
	Beamforming	A beamformer is a spatial filter that consists of an array of antenna elements with adjustable weights (coefficients). The twin purposes of an adaptive beamformer are to adaptively control the weights so as to cancel interfering signals impinging on the array from unknown directions and, at the same time, provide protection to a target signal of interest.

8. HISTORICAL NOTES

To understand a science it is necessary to know its history.
—Auguste Comte (1798–1857)

We complete this introductory chapter by presenting a brief historical review of developments in three areas that are closely related insofar as the subject matter of this book is concerned: linear estimation theory, linear adaptive filters, and adaptive signal-processing applications.

Linear Estimation Theory

The earliest stimulus for the development of linear estimation (filter) theory[1] was apparently provided by astronomical studies in which the motions of planets and comets were studied with the use of telescopic measurement data. The beginnings of a "theory" of estimation in which attempts are made to minimize various functions of errors can be attributed to Galileo Galilei in 1632. However, the origination of linear estimation theory is credited to Gauss, who, at the age of 18 in 1795, invented the *method of least squares* to study the motion of heavenly bodies (Gauss, 1809). Nevertheless, in the early 19th century, there was considerable controversy regarding the actual inventor of the method of least squares. The controversy arose because Gauss did not publish his 1795 discovery that year; rather, it was first published by Legendre in 1805, who independently invented the method (Legendre, 1810).

The first studies of minimum mean-square estimation in stochastic processes were made by Kolmogorov, Krein, and Wiener during the late 1930s and early 1940s

[1]The notes presented on linear estimation are influenced by the following review papers: Sorenson (1970), Kailath (1974), and Makhoul (1975).

(Kolmogorov, 1939; Krein, 1945; Wiener, 1949). The works of Kolmogorov and Krein were independent of Wiener's, and while there was some overlap in the results, their aims were rather different. There were many conceptual differences (as one would expect after 140 years) between the Gauss problem and the problems treated by Kolmogorov, Krein, and Wiener.

Kolmogorov, inspired by some early work of Wold on discrete-time stationary processes (Wold, 1938), developed a comprehensive treatment of the linear prediction problem for discrete-time stochastic processes. Krein noted the relationship of Kolmogorov's results to some early work by Szegö on orthogonal polynomials (Szegö, 1939; Grenander & Szegö, 1958) and extended the results to continuous time by the clever use of a bilinear transformation.

Independently, Wiener formulated the continuous-time linear prediction problem and derived an explicit formula for the optimum predictor. Wiener also considered the "filtering" problem of estimating a process corrupted by additive "noise." The explicit formula for the optimum estimate required the solution of an integral equation known as the *Wiener–Hopf equation* (Wiener & Hopf, 1931).

In 1947, Levinson formulated the Wiener filtering problem in discrete time. In the case of discrete-time signals, the Wiener–Hopf equation takes on a matrix form described by[2]

$$\mathbf{R}\mathbf{w}_o = \mathbf{p}, \tag{11}$$

where $\mathbf{w}_o$ is the tap-weight vector of the optimum Wiener filter structured in the form of an FIR filter, $\mathbf{R}$ is the correlation matrix of the tap inputs, and $\mathbf{p}$ is the cross-correlation vector between the tap inputs and the desired response. For stationary inputs, the correlation matrix $\mathbf{R}$ assumes a special structure known as *Toeplitz*, so named after the mathematician O. Toeplitz. By exploiting the properties of a Toeplitz matrix, Levinson derived an elegant recursive procedure for solving the matrix form of the Wiener–Hopf equation (Levinson, 1947). In 1960, Durbin rediscovered Levinson's recursive procedure as a scheme for the recursive fitting of autoregressive models to scalar time-series data. The problem considered by Durbin is a special case of Eq. (11) in which the column vector $\mathbf{p}$ comprises the same elements found in the correlation matrix $\mathbf{R}$. In 1963, Whittle showed that there is a close relationship between the Levinson–Durbin recursion and that for Szegö's orthogonal polynomials and also derived a multivariate generalization of the Levinson–Durbin recursion.

Wiener and Kolmogorov assumed an infinite amount of data and assumed the stochastic processes to be stationary. During the 1950s, some generalizations of Wiener–Kolmogorov filter theory were made by various authors to cover the estimation of stationary processes given only for a finite observation interval and to cover the estimation of nonstationary processes. However, some researchers were dissatisfied with the most significant of the results of this period because they were rather complicated, difficult to update with increases in the observation interval, and difficult to modify for the

[2]The Wiener–Hopf equation, originally formulated as an integral equation, specifies the optimum solution of a continuous-time linear filter subject to the contraint of causality. The equation is difficult to solve and has resulted in the development of a considerable amount of theory, including spectral factorization. (For a tutorial treatment of the subject, see Gardner, 1990.)

vector case. The last two difficulties became particularly evident in the late 1950s in the problem of determining satellite orbits. In this application, there were generally vector observations of some combinations of position and velocity, and large amounts of data were also sequentially accumulated with each pass of the satellite over a tracking station. Swerling was one of the first to tackle the problem by presenting some useful recursive algorithms (Swerling, 1958). For different reasons, Kalman independently developed a somewhat more restricted algorithm than Swerling's, but it was an algorithm that seemed particularly matched to the dynamic estimation problems that were brought by the advent of the space age (Kalman, 1960). After Kalman had published his paper and it had attained considerable fame, Swerling wrote a letter claiming priority for the Kalman filter equations (Swerling, 1963). However, Swerling's plea fell on deaf ears. It is ironic that orbit determination problems provided the stimulus for both Gauss' method of least squares and the Kalman filter and, in each case, there were squabbles concerning their inventors. Kalman's original formulation of the linear filtering problem was derived for discrete-time processes. The continuous-time filter was derived by Kalman in his subsequent collaboration with Bucy; this latter solution is sometimes referred to as the *Kalman–Bucy filter* (Kalman & Bucy, 1961).

In a series of stimulating papers, Kailath reformulated the solution to the linear filtering problem by using the *innovations* approach (Kailath, 1968, 1970; Kailath & Frost, 1968; Kailath & Geesey, 1973). In this approach, a stochastic process $u(n)$ is represented as the output of a causal and causally invertible filter driven by a white-noise process $\nu(n)$, which is called the *innovations process*, with the term "innovation" denoting newness. The reason for this terminology is that each sample of the process $\nu(n)$ provides entirely new information, in the sense that it is statistically independent of all past samples of the original process $u(n)$, assuming Gaussianity; otherwise, each sample is uncorrelated with all past samples of $u(n)$. The idea behind the innovations approach was actually introduced by Kolmogorov (1939).

Linear Adaptive Filters

Stochastic Gradient Algorithms. The earliest work on adaptive filters may be traced back to the late 1950s, during which time a number of researchers were working independently on different applications of such filters. From this early work, the *least-mean-square (LMS) algorithm* emerged as a simple, yet effective, algorithm for the operation of adaptive FIR filters. The LMS algorithm was devised by Widrow and Hoff in 1959 in their study of a pattern-recognition scheme known as the *adaptive linear* (threshold logic) *element*, commonly referred to in the literature as the *Adaline* (Widrow & Hoff, 1960; Widrow, 1970). The LMS algorithm is a stochastic gradient algorithm in that it iterates each tap weight of an FIR filter in the direction of the gradient of the squared magnitude of an error signal with respect to the tap weight. Accordingly, the LMS algorithm is closely related to the concept of *stochastic approximation* developed by Robbins and Monro (1951) in statistics for solving certain sequential parameter estimation problems. The primary difference between them is that the LMS algorithm uses a fixed step-size parameter to control the correction applied to each tap weight from one adaptation cycle to the next, whereas in stochastic approximation methods the step-size parameter is made inversely proportional to time n or to

a power of n. Another stochastic gradient algorithm, closely related to the LMS algorithm, is the *gradient adaptive lattice (GAL) algorithm* (Griffiths, 1977, 1978); the difference between them is structural in that the GAL algorithm is lattice based, whereas the LMS algorithm uses an FIR filter.

Recursive Least-Squares Algorithms. Turning next to the recursive least-squares (RLS) family of adaptive filtering algorithms, we find that the original paper on the *traditional RLS algorithm* appears to be that of Plackett (1950), although it must be said that many other investigators have derived and rederived various versions of the RLS algorithm. In 1974, Godard used Kalman filter theory to derive one variant of the algorithm that is sometimes referred to in the literature as the *Godard algorithm*. Although prior to that date several investigators had applied Kalman filter theory to solve the adaptive filtering problem, Godard's approach was widely accepted as the most successful application of Kalman filter theory for a span of two decades. Then, Sayed and Kailath (1994) published an expository paper in which the *exact* relationship between the RLS algorithm and Kalman filter theory was delineated for the first time, thereby laying the groundwork for how to exploit the vast literature on Kalman filters for solving linear adaptive filtering problems.

In 1981, Gentleman and Kung introduced a numerically robust method based on the *QR-decomposition* of matrix algebra for solving the RLS problem. The resulting adaptive filter structure, sometimes referred to as the *Gentleman–Kung (systolic) array*, was subsequently refined and extended in various ways by many other investigators.

In the 1970s and subsequently, a great deal of research effort was expended on the development of numerically stable *fast RLS algorithms*, with the aim of reducing the computational complexity of RLS algorithms to a level comparable to that of the LMS algorithm. In one form or another, the development of the fast RLS algorithms can be traced back to results derived by Morf in 1974 for solving the deterministic counterpart of the stochastic filtering problem solved efficiently by the Levinson–Durbin algorithm for stationary inputs.

Robustness in the H^{∞}-sense

Robustness, particularly in control systems, had occupied the attention of control theorists for many years in the 20th century; see Zames (1996) for a historical account of the literature on intput–output feedback stability and robustness for the period from 1959 to 1985.

A breakthrough happened in the late 1970s, when it was recognized that minimization of the sensitivity problem could admit explicit and even closed-form solutions. That idea opened the door for the formulation of a practical theory of optimizing robustness, with Zames (1979) presenting a paper at the Allerton Conference,which was followed up by a journal paper in the *IEEE Tranasctions on Automatic Control* (1981) and another in the same journal coauthored by Francis and Zames (1984). And, with these papers, what we now know as the H^{∞}-robustness theory, separated from stability theory, was born.

However, it took the signal-processing community well over a decade to come to grips with this new theory of robustness. That awareness happened with publication of a journal paper coauthored by Hassibi, Sayed, and Kailath in the *IEEE Transactions on*

Signal Processing (1996); in that paper, robustness of the LMS algorithm was described for the first time, confirming theoretically what had been known in practical applications of the LMS algorithm for a long time. Subsequently, a journal paper on bounds on least-squares estimation in the H^{∞}-sense, authored by Hassibi and Kailath (2001), was published.

Adaptive Signal-Processing Applications

Adaptive Equalization. Until the early 1960s, the equalization of telephone channels to combat the degrading effects of intersymbol interference on data transmission was performed by using either fixed equalizers (resulting in a performance loss) or equalizers whose parameters were adjusted manually (a rather cumbersome procedure). In 1965, Lucky made a major breakthrough in the equalization problem by proposing a *zero-forcing algorithm* for automatically adjusting the tap weights of an FIR equalizer (Lucky, 1965). A distinguishing feature of the work by Lucky was the use of a *minimax* type of performance criterion. In particular, he used a performance index called *peak distortion*, which is directly related to the maximum value of intersymbol interference that can occur. The tap weights in the equalizer are adjusted to minimize the peak distortion. This has the effect of *forcing* the intersymbol interference due to those adjacent pulses which are contained in the FIR equalizer to become *zero*—hence the name of the algorithm. A sufficient, but not necessary, condition for optimality of the zero-forcing algorithm is that the *initial distortion* (the distortion that exists at the equalizer input) be less than unity. In a subsequent paper published in 1966, Lucky extended the use of the zero-forcing algorithm to the tracking mode of operation. In 1965, DiToro independently used adaptive equalization to combat the effect of intersymbol interference on data transmitted over high-frequency links.

The pioneering work by Lucky inspired many other significant contributions to different aspects of the adaptive equalization problem in one way or another. Using a mean-square-error criterion, Gersho (1969) and Proakis and Miller (1969) independently reformulated the adaptive equalization problem. In 1972, using the LMS algorithm, Ungerboeck presented a detailed mathematical analysis of the convergence properties of an adaptive FIR equalizer. In 1974, as mentioned previously, Godard used Kalman filter theory to derive a powerful algorithm for adjusting the tap weights of an FIR equalizer. In 1978, Falconer and Ljung presented a modification of this algorithm that simplified its computational complexity to a level comparable to that of the simple LMS algorithm. Satorius and Alexander (1979) and Satorius and Pack (1981) demonstrated the usefulness of lattice-based algorithms for the adaptive equalization of dispersive channels.

The preceding brief historical review pertains to the use of adaptive equalizers for *linear synchronous receivers*, where, by "synchronous," we mean that the equalizer in the receiver has its taps spaced at the reciprocal of the symbol rate. Even though our interest in adaptive equalizers is largely restricted to this class of receivers, such a historical review would be incomplete without some mention of fractionally spaced equalizers and decision-feedback equalizers.

In a *fractionally spaced equalizer (FSE)*, the equalizer taps are spaced more closely than the reciprocal of the symbol rate. An FSE has the capability of compensating for

delay distortion much more effectively than a conventional synchronous equalizer does. Another advantage of the FSE is the fact that data transmission may begin with an arbitrary sampling phase. However, mathematical analysis of the FSE is much more complicated than that of a conventional synchronous equalizer. It appears that early work on the FSE was initiated by Brady (1970). Other contributions to the subject include subsequent work by Ungerboeck (1976) and Gitlin and Weinstein (1981). In the early 1990s, more light was cast on the benefits of fractional equalization by showing how it could be exploited for "blind" equalization of the channel, using only second-order statistics. Previously, the conventional wisdom had been that only minimum-phase channels could be identified or equalized in this way, since second-order statistics are indifferent to the phase information in the channel output. However, this indifference holds only for stationary second-order statistics; the oversampling used in the FSE means that the second-order statistics are *cyclostationary*, which adds a new dimension, namely, the period, to the partial description of the channel output (Franks, 1969; Gardner & Franks, 1975; Gardner, 1994a, b). Tong and his collaborators (1993, 1994a, b) showed how, under weak operating conditions, cyclostationarity could be effectively exploited for blind equalization. Their papers have led to an explosion of activity in the field, with numerous extensions, refinements, and variations subsequenty appearing. (See, for example, the tutorial paper by Tong and Perreau, 1998.)

A *decision-feedback equalizer* uses a feedforward section as well as a feedback section. The feedforward section consists of an FIR filter whose taps are spaced at the reciprocal of the symbol rate. The data sequence to be equalized is applied to the input of this section. The feedback section consists of another FIR filter whose taps are also spaced at the reciprocal of the symbol rate. The input applied to the feedback section is made up of decisions on previously detected symbols. The function of the feedback section is to subtract out that portion of intersymbol interference produced by previously detected symbols from the estimates of future symbols. This cancellation is an old idea known as the *bootstrap technique*. A decision-feedback equalizer yields good performance in the presence of severe intersymbol interference, which is experienced in fading radio channels, for example. The first report on decision-feedback equalization was published by Austin (1967), and the optimization of the decision-feedback receiver for minimum mean-square-error analysis was first accomplished by Monsen (1971).

Coding of Speech. In 1966, Saito and Itakura used a *maximum-likelihood* approach for the application of prediction to speech. A standard assumption in the application of the maximum-likelihood principle is that the input process is Gaussian. Under this condition, the exact application of the principle yields a set of nonlinear equations for the parameters of the predictor. To overcome this difficulty, Itakura and Saito utilized approximations based on the assumption that the number of available data points greatly exceeds the prediction order. The use of this assumption makes the result obtained from the maximum-likelihood principle assume an approximate form that is the same as the result of the *autocorrelation method* of linear prediction. The application of the maximum-likelihood principle is justified on the assumption that speech is a stationary Gaussian process, which seems reasonable in the case of unvoiced sounds.

In 1970, Atal presented the first use of the term "linear prediction" in speech analysis. Details of this new approach, called linear predictive coding (LPC), to speech analysis and synthesis were published by Atal and Hanauer in 1971. In LPC, the speech waveform is represented directly in terms of time-varying parameters related to the transfer function of the vocal tract and the characteristics of the excitation. The predictor coefficients are determined by minimizing the mean-square error, with the error defined as the difference between the actual and predicted values of the speech samples. In the work by Atal and Hanauer, the speech wave was sampled at 10 kHz and then analyzed by predicting the present speech sample as a linear combination of the 12 previous samples. Thus, 15 parameters [the 12 parameters of the predictor, the pitch period, a binary parameter indicating whether the speech is voiced or unvoiced, and the root-mean-square (rms) value of the speech samples] were used to describe the speech analyzer. For the speech synthesizer, an all-pole filter was used, with a sequence of quasi-periodic pulses or a white-noise source providing the excitation.

Another significant contribution to the linear prediction of speech was made in 1972 by Itakura and Saito, who used partial correlation techniques to develop a new structure, the lattice, for formulating the linear prediction problem.[3] The parameters that characterize the lattice predictor are called *reflection coefficients*. Although by 1972 the essence of the lattice structure had been considered by several other investigators, the invention of the lattice predictor is credited to Saito and Itakura. In 1973, Wakita showed that the filtering actions of the lattice predictor model and an acoustic tube model of speech are identical, with the reflection coefficients in the acoustic tube model as common factors. This discovery made possible the extraction of the reflection coefficients by the use of a lattice predictor.

A limitation of the linear predictive modeling of speech signals is the failure to recover the correct envelope from discrete spectral samples; this limitation is traceable to the choice of the minimum mean-square error as the criterion for linear predictive modeling. To mitigate the problem, Itakura and Saito (1970) developed a maximum-likelihood procedure for the spectral envelope estimation of voiced sounds; in so doing, they introduced a new criterion known as the *Itakura–Saito distance measure*. This development was followed by further contributions due to McAulay (1984) and El-Jaroudi and Makhoul (1991).

Early designs of a lattice predictor were based on a *block-processing* approach (Burg, 1967). In 1981, Makhoul and Cossell used an *adaptive* approach to design the lattice predictor for applications in speech analysis and synthesis. They showed that the convergence of the adaptive lattice predictor is fast enough for its performance to equal that of the optimal (but more expensive) adaptive autocorrelation method.

The historical review on speech coding presented up to now relates to LPC vocoders. We next present a historical review of the adaptive predictive coding of speech, starting with ordinary pulse-code modulation (PCM).

PCM was invented in 1937 by Reeves (1975). Then came the invention of differential pulse-code modulation (DPCM) by Cutler (1952). The early use of DPCM

[3]According to Markel and Gray (1976) the work of Itakura and Saito in Japan on the partial correlation (PARCOR) formulation of linear prediction had been presented in 1969.

for the predictive coding of speech signals was limited to linear predictors with *fixed* parameters (McDonald, 1966). However, because of the nonstationary nature of speech signals, a fixed predictor cannot predict the signal values efficiently at all times. In order to respond to the nonstationary characteristics of speech signals, the predictor has to be adaptive. In 1970, Atal and Schroeder described a sophisticated scheme for implementing the adaptive predictive coding of speech. The scheme recognizes that there are two main causes of redundancy in speech (Schroeder, 1966): (1) quasi-periodicity during voiced segments and (2) nonflatness of the short-time spectral envelope. Thus, the predictor is designed to remove signal redundancy in two stages: First, it removes the quasi-periodic nature of the signal, and then it removes formant information from the spectral envelope. The scheme achieves dramatic reductions in bit rate at the expense of a significant increase in complexity of the circuit. Atal and Schroeder (1970) reported that the scheme can transmit speech at 10 kb/s, which is several times slower than the bit rate required for logarithmic-PCM encoding with comparable speech quality.

Spectrum Analysis. At the turn of the 20th century, Schuster introduced the *periodogram* for analyzing the power spectrum of a time series (Schuster, 1898). The periodogram is defined as the squared amplitude of the discrete Fourier transform of the series. The periodogram was originally used by Schuster to detect and estimate the amplitude of a sine wave of known frequency that is buried in noise. Until the work of Yule in 1927, the periodogram was the only numerical method available for spectrum analysis. However, the periodogram suffers from the limitation that when it is applied to empirical time series observed in nature, the results obtained are highly erratic. This limitation led Yule to introduce a new approach based on the concept of a *finite parameter model* for a stationary stochastic process in his investigation of the periodicities in time series with special reference to Wolfer's sunspot number (Yule, 1927). In effect, Yule created a stochastic feedback model in which the present sample value of the time series is assumed to consist of a linear combination of past sample values plus an error term. This model is called an *autoregressive model*, in that a sample of the time series regresses on its own past values, and the method of spectrum analysis based on such a model is accordingly called *autoregressive spectrum analysis*. The name "autoregressive" was coined by Wold in his doctoral thesis (Wold, 1938).

Interest in the autoregressive method was reinitiated by Burg (1967, 1975), who introduced the term *maximum-entropy method* to describe an algorithmic approach for estimating the power spectrum directly from the available time series. The idea behind the maximum-entropy method is to extrapolate the autocorrelation function of the series in such a way that the *entropy* of the corresponding probability density function is maximized at each step of the extrapolation. In 1971, Van den Bos showed that the maximum-entropy method is equivalent to least-squares fitting of an autoregressive model to the known autocorrelation sequence.

Another important contribution to the literature on spectrum analysis is that by Thomson (1982). His *method of multiple windows*, based on the prolate spheroidal wave functions, is a nonparametric method for spectrum estimation that overcomes many of the limitations of the aforementioned techniques.

Adaptive Noise Cancellation. The initial work on adaptive echo cancellers started around 1965. It appears that Kelly of Bell Telephone Laboratories was the first to propose the use of an adaptive filter for echo cancellation, with the speech signal itself utilized in performing the adaptation; Kelly's contribution is recognized in a paper by Sondhi (1967). This invention and its refinement are described in patents by Kelly and Logan (1970) and Sondhi (1970).

The adaptive line enhancer was originated by Widrow and his coworkers at Stanford University. An early version of the device was built in 1965 to cancel 60-Hz interference at the output of an electrocardiographic amplifier and recorder. This work is described in a paper by Widrow et al. (1975b). The adaptive line enhancer and its application as an adaptive detector were patented by McCool et al. (1980).

The adaptive echo canceller and the adaptive linear enhancer, although intended for different applications, may be viewed as examples of the *adaptive noise canceller* discussed by Widrow et al. (1975b). This scheme operates on the outputs of two sensors: a *primary sensor* that supplies a desired signal of interest buried in noise and a *reference sensor* that supplies noise alone. It is assumed that (1) the signal and noise at the output of the primary sensor are uncorrelated and (2) the noise at the output of the reference sensor is correlated with the noise component of the primary sensor output.

The adaptive noise canceller consists of an adaptive filter that operates on the reference sensor output to produce an *estimate* of the noise, which is then subtracted from the primary sensor output. The overall output of the canceller is used to control the adjustments applied to the tap weights in the adaptive FIR filter. The adaptive canceller tends to minimize the mean-square value of the overall output, thereby causing the output to be the best estimate of the desired signal in the minimum mean-square-error sense.

Adaptive Beamforming. The development of adaptive beamforming technology may be traced back to the invention of the *intermediate frequency (IF) sidelobe canceller* by Howells in the late 1950s. In a paper published in a 1976 special issue of the *IEEE Transactions on Antennas and Propagation*, Howells describes his personal observations on early work on adaptive antennas at General Electric and Syracuse University Research Corporation. According to this historic report, by mid-1957 Howells had developed a sidelobe canceller capable of automatically nulling out the effect of a single jammer. The sidelobe canceller uses a *primary* (high-gain) antenna and a *reference omnidirectional* (low-gain) antenna to form a two-element array with one degree of freedom that makes it possible to steer a deep null anywhere in the sidelobe region of the combined antenna pattern. In particular, a null is placed in the direction of the jammer, with only a minor perturbation of the main lobe. Subsequently, Howells (1965) patented the sidelobe canceller.

The second major contribution to adaptive array antennas was made by Applebaum in 1966. In a classic report, he derived the *control law* governing the operation of an adaptive array antenna, with a control loop for each element of the array (Applebaum, 1966). The algorithm derived by Applebaum was based on maximizing the signal-to-noise ratio (SNR) at the array antenna output for any type of noise environment. Applebaum's theory included the sidelobe canceller as a special case. His 1966 classic report was reprinted in the 1976 special issue of the *IEEE Transactions on Antennas and Propagation.*

Another algorithm for the weight adjustment in adaptive array antennas was advanced independently in 1967 by Widrow and coworkers at Stanford University. They based their theory on the simple, yet effective, LMS algorithm. The 1967 paper by Widrow et al. not only was the first publication in the open literature on adaptive array antenna systems, but also is considered to be another classic of that era.

It is noteworthy that the maximum-SNR algorithm (used by Applebaum) and the LMS algorithm (used by Widrow and coworkers) for adaptive array antennas are rather similar. Both algorithms derive the control law for adaptive adjustment of the weights in the array antenna by sensing the correlation between element signals. Indeed, they both converge toward the optimum Wiener solution for stationary inputs (Gabriel, 1976).

A different method for solving the adaptive beamforming problem was proposed by Capon (1969), who realized that the poor performance of the delay-and-sum beamformer is due to the fact that its response along a direction of interest depends not only on the power of the incoming target signal, but also on undesirable contributions received from other sources of interference. To overcome this limitation, Capon proposed a new beamformer in which the weight vector $\mathbf{w}(n)$ is chosen so as to *minimize the variance* (i.e., average power) of the output, subject to the constraint $\mathbf{w}^{\mathrm{H}}(n)\mathbf{s}(\theta) = 1$ for all n, where $\mathbf{s}(\theta)$ is a prescribed *steering vector*. This constrained minimization yields an adaptive beamformer with *minimum-variance distortionless response (MVDR)*.

In 1983, McWhirter proposed a simplification of the Gentleman–Kung (systolic) array for recursive least-squares estimation. The resulting filtering structure, often referred to as the *McWhirter (systolic) array*, is particularly well suited for adaptive beamforming applications.

The historical notes presented in this introductory chapter on adaptive filter theory and applications are not claimed to be complete. Rather, they are intended to highlight many of the early significant contributions made to this important part of the ever-expanding field of adaptive signal processing. Above all, it is hoped that they provide a source of inspiration to the reader.

CHAPTER 1

Stochastic Processes and Models

The term *stochastic process*, or *random process*, is used to describe the time evolution of a statistical phenomenon according to probabilistic laws. The time evolution of the phenomenon means that the stochastic process is a function of time, defined on some observation interval. The statistical nature of the phenomenon means that, before conducting an experiment, it is not possible to define exactly the way it evolves in time. Examples of a stochastic process include speech signals, television signals, radar signals, digital computer data, the output of a communication channel, seismological data, and noise.

The form of a stochastic process that is of interest to us is one that is defined at *discrete and uniformly spaced instants of time* (Box & Jenkins, 1976; Priestley, 1981). Such a restriction may arise naturally in practice, as in the case of radar signals or digital computer data. Alternatively, the stochastic process may be defined originally for a continuous range of real values of time; however, before processing, it is *sampled uniformly* in time, with the sampling rate chosen to be greater than twice the highest frequency component of the process (Haykin, 2013).

A stochastic process is *not* just a single function of time; rather, it represents, in theory, an infinite number of *different* realizations of the process. One particular realization of a discrete-time stochastic process is called a *time series*. For convenience of notation, *we normalize time with respect to the sampling period*. For example, the sequence $u(n)$, $u(n-1), \ldots, u(n-M)$ represents a time series that consists of the *present* observation $u(n)$ made at time n and M past observations of the process made at times $n-1, \ldots, n-M$.

We say that a stochastic process is *strictly stationary* if its statistical properties are *invariant* to a time shift. Specifically, for a discrete-time stochastic process represented by the time series $u(n), u(n-1), \ldots, u(n-M)$ to be strictly stationary, the *joint probability density function* of random variables represented by the observations times n, $n-1, \ldots, n-M$ must remain the same no matter what values we assign to n for fixed M.

1.1 PARTIAL CHARACTERIZATION OF A DISCRETE-TIME STOCHASTIC PROCESS

In practice, we usually find that it is not possible to determine (by means of suitable measurements) the joint probability density function for an arbitrary set of observations made on a stochastic process. Accordingly, we must content ourselves with a partial characterization of the process by specifying its first and second moments.

Consider a discrete-time stochastic process represented by the time series $u(n)$, $u(n-1), \ldots, u(n-M)$, which may be complex valued. To simplify the terminology, we use $u(n)$ to denote such a process. This simplified terminology[1] is used throughout the book. We define the *mean-value function* of the processs

$$\mu(n) = \mathbb{E}[u(n)], \tag{1.1}$$

where $\mathbb{E}$ denotes the *statistical expectation operator*. We define the *autocorrelation function* of the process

$$r(n, n-k) = \mathbb{E}[u(n)u^*(n-k)], \quad k = 0, \pm 1, \pm 2, \ldots, \tag{1.2}$$

where the asterisk denotes *complex conjugation*. We define the *autocovariance function* of the process

$$c(n, n-k) = \mathbb{E}[(u(n) - \mu(n))(u(n-k) - \mu(n-k))^*], \; k = 0, \pm 1, \pm 2, \ldots. \tag{1.3}$$

From Eqs. (1.1) through (1.3), we see that the mean-value, autocorrelation, and autocovariance functions of the process are related by

$$c(n, n-k) = r(n, n-k) - \mu(n)\mu^*(n-k). \tag{1.4}$$

For a partial (second-order) characterization of the process, we therefore need to specify (1) the mean-value function $\mu(n)$ and (2) the autocorrelation function $r(n, n-k)$ or the autocovariance function $c(n, n-k)$ for various values of n and k that are of interest. Note that the autocorrelation and autocovariance functions have the same value when the mean $\mu(n)$ is zero for all n.

This form of partial characterization offers two important advantages:

1. It lends itself to practical measurements.
2. It is well suited to *linear* operations on stochastic processes.

For a discrete-time stochastic process that is strictly stationary, all three quantities defined in Eqs. (1.1) through (1.3) assume simpler forms. In particular, they satisfy two conditions

1. The mean-value function of the process is a constant μ (say), that is,

$$\mu(n) = \mu \qquad \text{for all } n. \tag{1.5}$$

2. The autocorrelation and autocovariance functions depend only on the *difference* between the observation times n and $n-k$; that is, they depend on the lag k, as shown by

$$r(n, n-k) = r(k) \tag{1.6}$$

and

$$c(n, n-k) = c(k). \tag{1.7}$$

[1] To be rigorous in our terminology, we should use an uppercase symbol—$U(n)$, for example—to denote a discrete-time stochastic process and the corresponding lowercase symbol—$u(n)$—to denote a sample of the process. We have not done so to simplify the exposition.

Note that when $k = 0$, corresponding to a time difference, or lag, of zero, $r(0)$ equals the *mean-square value* of $u(n)$:

$$r(0) = \mathbb{E}[\,|u(n)|^2\,]. \tag{1.8}$$

Also, $c(0)$ equals the *variance* of $u(n)$:

$$c(0) = \sigma_u^2. \tag{1.9}$$

Equations (1.5) through (1.7) are *not* sufficient to guarantee that the discrete-time stochastic process is strictly stationary. Rather, a discrete-time stochastic process that is not strictly stationary, but for which these conditions hold, is said to be *wide-sense* or *weakly stationary*. A strictly stationary process $\{u(n)\}$, or $u(n)$ for short, is stationary in the wide sense if and only if (Doob, 1953)

$$\mathbb{E}[|u(n)|^2] < \infty \qquad \text{for all } n.$$

This condition is ordinarily satisfied by stochastic processes encountered in the physical sciences and engineering.

1.2 MEAN ERGODIC THEOREM

The *expectations*, or *ensemble averages*, of a stochastic process are averages "across different realizations of the process" for a fixed instant of time. Clearly, we may also define *long-term sample averages*, or *time averages* that are averages "along the process." Indeed, time averages may be used to build a *stochastic model* of a physical process by *estimating* unknown parameters of the model. For such an approach to be rigorous, however, we have to show that time averages converge to corresponding ensemble averages of the process in some statistical sense. A popular criterion for convergence is that of the mean-square error.

In this regard, consider a discrete-time stochastic process $u(n)$ that is wide-sense stationary. Let a constant μ denote the mean of the process and $c(k)$ denote its autocovariance function for lag k. For an estimate of the mean μ, we may use the time average

$$\hat{\mu}(N) = \frac{1}{N}\sum_{n=0}^{N-1} u(n), \tag{1.10}$$

where N is the total number of samples used in the estimation. Note that the estimate $\hat{\mu}(N)$ is a random variable with a mean and variance of its own. In particular, we readily find from Eq. (1.10) that the mean (expectation) of $\hat{\mu}(N)$ is

$$\mathbb{E}[\hat{\mu}(N)] = \mu \qquad \text{for all } N. \tag{1.11}$$

It is in the sense of Eq. (1.11) that we say the time average $\hat{\mu}(N)$ is an *unbiased* estimator of the ensemble average (mean) of the process.

Moreover, we say that the process $u(n)$ is *mean ergodic in the mean-square-error sense* if the mean-square value of the error between the ensemble average μ and the time average $\hat{\mu}(N)$ approaches zero as the number of samples, N, approaches infinity; that is,

$$\lim_{N\to\infty} \mathbb{E}[|\mu - \hat{\mu}(N)|^2] = 0.$$

Using the time average formula of Eq. (1.10), we may write

$$\begin{aligned}\mathbb{E}[|\mu - \hat{\mu}(N)|^2] &= \mathbb{E}\left[\left|\mu - \frac{1}{N}\sum_{n=0}^{N-1} u(n)\right|^2\right] \\ &= \frac{1}{N^2}\mathbb{E}\left[\left|\sum_{n=0}^{N-1}(u(n) - \mu)\right|^2\right] \\ &= \frac{1}{N^2}\mathbb{E}\left[\sum_{n=0}^{N-1}\sum_{k=0}^{N-1}(u(n) - \mu)(u(k) - \mu)^*\right] \\ &= \frac{1}{N^2}\sum_{n=0}^{N-1}\sum_{k=0}^{N-1}\mathbb{E}[(u(n) - \mu)(u(k) - \mu)^*] \\ &= \frac{1}{N^2}\sum_{n=0}^{N-1}\sum_{k=0}^{N-1} c(n - k). \end{aligned} \tag{1.12}$$

Let $l = n - k$. We may then simplify the double summation in Eq. (1.12) as follows:

$$\mathbb{E}[|\mu - \hat{\mu}(N)|^2] = \frac{1}{N}\sum_{l=-N+1}^{N-1}\left(1 - \frac{|l|}{N}\right)c(l).$$

Accordingly, we may state that the necessary and sufficient condition for the process $u(n)$ to be mean ergodic in the mean-square-error sense is

$$\lim_{N\to\infty}\frac{1}{N}\sum_{l=-N+1}^{N-1}\left(1 - \frac{|l|}{N}\right)c(l) = 0. \tag{1.13}$$

In other words, if the process $u(n)$ is asymptotically uncorrelated in the sense of Eq. (1.13), then the time average $\hat{\mu}(N)$ of the process converges to the ensemble average μ in the mean-square-error sense. This is the statement of a particular form of the *mean ergodic theorem* (Gray & Davisson, 1986).

The use of the mean ergodic theorem may be extended to other time averages of the process. Consider, for example, the following time average used to estimate the autocorrelation function of a wide-sense stationary process:

$$\hat{r}(k, N) = \frac{1}{N}\sum_{n=0}^{N-1} u(n)u(n - k), \qquad 0 \le k \le N - 1. \tag{1.14}$$

The process $u(n)$ is said to be *correlation ergodic* in the mean-square-error sense if the mean-square value of the difference between the true value $r(k)$ and the estimate $\hat{r}(k, N)$ approaches zero as the number of samples, N, approaches infinity. Let $z(n, k)$ denote a new discrete-time stochastic process related to the original process $u(n)$ as follows:

$$z(n, k) = u(n)u(n - k). \tag{1.15}$$

Hence, by substituting $z(n, k)$ for $u(n)u(n - k)$, we may use the mean ergodic theorem to establish the conditions for $z(n, k)$ to be mean ergodic or, equivalently, for $u(n)$ to be correlation ergodic.

From the viewpoint of parameter estimation, the extraction of the mean (i.e., the DC component) is the most important *preprocessing* operation performed on a time series before computing the estimate of interest. This preprocessing results in a *residual* time series with zero mean, in which case the computation of the autocorrelation function for different lags is sufficient for partial characterization of the process.

Henceforth, we assume that the stochastic process $u(n)$ has zero mean, as shown by

$$\mathbb{E}[u(n)] = 0 \qquad \text{for all } n.$$

The set of autocorrelations of $u(n)$ for a finite number of lags defines the correlation matrix of the process.

1.3 CORRELATION MATRIX

Let the M-by-1 *observation vector* $\mathbf{u}(n)$ represent the elements of the zero-mean time series $u(n), u(n-1), \ldots, u(n-M+1)$. To show the composition of the vector $\mathbf{u}(n)$ explicitly, we write

$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^{\mathrm{T}}, \tag{1.16}$$

where the superscript T denotes *transposition*. We define the *correlation matrix* of a stationary discrete-time stochastic process represented by this time series as the *expectation of the outer product of the observation vector* $\mathbf{u}(n)$ *with itself*. Let $\mathbf{R}$ denote the M-by-M correlation matrix defined in this way. We thus write

$$\mathbf{R} = \mathbb{E}[\mathbf{u}(n)\,\mathbf{u}^{\mathrm{H}}(n)], \tag{1.17}$$

where the superscript H denotes *Hermitian transposition* (i.e., the operation of transposition combined with complex conjugation). Substituting Eq. (1.16) into Eq. (1.17) and using the condition of wide-sense stationarity, we express the correlation matrix in the expanded form

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r(-1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(-M+1) & r(-M+2) & \cdots & r(0) \end{bmatrix}. \tag{1.18}$$

The element $r(0)$ on the main diagonal is always real valued. For complex-valued data, the remaining elements of $\mathbf{R}$ assume complex values.

Properties of the Correlation Matrix

The correlation matrix $\mathbf{R}$ plays a key role in the statistical analysis and design of discrete-time filters. It is therefore important that we understand its various properties and their implications. In particular, using the definition given in Eq. (1.17), we find that the correlation matrix of a stationary discrete-time stochastic process has the properties that follow.

Property 1. *The correlation matrix of a stationary discrete-time stochastic process is Hermitian.*

We say that a *complex-valued* matrix is *Hermitian* if it is equal to its *conjugate transpose*. We may thus express the Hermitian property of the correlation matrix $\mathbf{R}$ by writing

$$\mathbf{R}^{\mathrm{H}} = \mathbf{R}. \tag{1.19}$$

This property follows directly from the definition of Eq. (1.17).

Another way of stating the Hermitian property of the correlation matrix $\mathbf{R}$ is to write

$$r(-k) = r^*(k), \tag{1.20}$$

where $r(k)$ is the autocorrelation function of the stochastic process $u(n)$ for a lag of k. Accordingly, for a wide-sense stationary process, we only need M values of the autocorrelation function $r(k)$ for $k = 0, 1, \ldots, M-1$ in order to completely define the correlation matrix $\mathbf{R}$. We may thus rewrite Eq. (1.18) as

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r^*(1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r^*(M-1) & r^*(M-2) & \cdots & r(0) \end{bmatrix}. \tag{1.21}$$

From here on, we will use this representation for the expanded-matrix form of the correlation matrix of a wide-sense stationary discrete-time stochastic process. Note that for the special case of *real-valued data*, the autocorrelation function $r(k)$ is real for all k and the correlation matrix $\mathbf{R}$ is *symmetric*.

Property 2. *The correlation matrix of a stationary discrete-time stochastic process is Toeplitz.*

We say that a square matrix is *Toeplitz* if all the elements on its main diagonal are equal and if the elements on any other diagonal parallel to the main diagonal are also equal. From the expanded form of the correlation matrix $\mathbf{R}$ given in Eq. (1.21), we see that all the elements on the main diagonal are equal to $r(0)$, all the elements on the first diagonal above the main diagonal are equal to $r(1)$, all the elements along the first diagonal below the main diagonal are equal to $r^*(1)$, and so on for the other diagonals. We conclude, therefore, that the correlation matrix $\mathbf{R}$ is Toeplitz.

It is important to recognize, however, that the Toeplitz property of the correlation matrix $\mathbf{R}$ is a direct consequence of the assumption that the discrete-time stochastic process represented by the observation vector $\mathbf{u}(n)$ is wide-sense stationary. Indeed, we may state that if the discrete-time stochastic process is wide-sense stationary, then its correlation matrix $\mathbf{R}$ must be Toeplitz; and, conversely, if the correlation matrix $\mathbf{R}$ is Toeplitz, then the discrete-time stochastic process must be wide-sense stationary.

Property 3. *The correlation matrix of a discrete-time stochastic process is always nonnegative definite and almost always positive definite.*

Let $\mathbf{a}$ be an arbitrary (nonzero) M-by-1 complex-valued vector. Define the scalar random variable y as the *inner product* of $\mathbf{a}$ and the observation vector $\mathbf{u}(n)$, as shown by

$$y = \mathbf{a}^{\mathrm{H}}\mathbf{u}(n).$$

Taking the Hermitian transpose of both sides and recognizing that y is a scalar, we get

$$y^* = \mathbf{u}^{\mathrm{H}}(n)\mathbf{a},$$

where the asterisk denotes *complex conjugation*. The mean-square value of the random variable y is

$$\begin{aligned}\mathbb{E}[|y|^2] &= \mathbb{E}[yy^*] \\ &= \mathbb{E}[\mathbf{a}^{\mathrm{H}}\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{a}] \\ &= \mathbf{a}^{\mathrm{H}}\mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]\mathbf{a} \\ &= \mathbf{a}^{\mathrm{H}}\mathbf{R}\mathbf{a},\end{aligned}$$

where $\mathbf{R}$ is the correlation matrix defined in Eq. (1.17). The expression $\mathbf{a}^{\mathrm{H}}\mathbf{R}\mathbf{a}$ is called a *Hermitian form*. Since

$$\mathbb{E}[|y|^2] \geq 0,$$

it follows that

$$\mathbf{a}^{\mathrm{H}}\mathbf{R}\mathbf{a} \geq 0. \tag{1.22}$$

A Hermitian form that satisfies this condition for every nonzero $\mathbf{a}$ is said to be *nonnegative definite* or *positive semidefinite*. Accordingly, we may state that the correlation matrix of a wide-sense stationary process is always nonnegative definite.

If the Hermitian form satisfies the condition

$$\mathbf{a}^{\mathrm{H}}\mathbf{R}\mathbf{a} > 0$$

for every nonzero $\mathbf{a}$, we say that the correlation matrix $\mathbf{R}$ is *positive definite*. This condition is satisfied for a wide-sense stationary process, unless there are linear dependencies between the random variables that constitute the M elements of the observation vector $\mathbf{u}(n)$. Such a situation arises essentially only when the process $u(n)$ consists of the sum of K sinusoids with $K \leq M$. (See Section 1.4 for more details.) In practice, we find that this idealized situation occurs so rarely that the correlation matrix $\mathbf{R}$ is almost always positive definite.

Property 4. *The correlation matrix of a wide-sense stationary process is nonsingular due to the unavoidable presence of additive noise.*

The matrix $\mathbf{R}$ is said to be nonsingular if its *determinant*, denoted by $\det(\mathbf{R})$, is nonzero. Typically, each element of the observation $\mathbf{u}(n)$ includes an additive noise component. Accordingly,

$$|r(l)| < r(0) \qquad \text{for all } l \neq 0,$$

in which case $\det(\mathbf{R}) \neq 0$ and the correlation matrix is nonsingular.

Property 4 has an important computational implication: Nonsingularity of the matrix $\mathbf{R}$ means that its inverse, denoted by $\mathbf{R}^{-1}$, exists. The inverse of $\mathbf{R}$ is defined by

$$\mathbf{R}^{-1} = \frac{\mathrm{adj}(\mathbf{R})}{\det(\mathbf{R})}, \tag{1.23}$$

where adj($\mathbf{R}$) is the *adjoint* of $\mathbf{R}$. By definition, adj($\mathbf{R}$) is the transpose of a matrix whose entries are the cofactors of the elements in det($\mathbf{R}$). Equation (1.23) shows that when det($\mathbf{R}$) $\neq 0$ (i.e., the matrix $\mathbf{R}$ is nonsingular), the inverse matrix $\mathbf{R}^{-1}$ exists and is therefore computable.

Property 5. *When the elements that constitute the observation vector of a stationary discrete-time stochastic process are rearranged backward, the effect is equivalent to the transposition of the correlation matrix of the process.*

Let $\mathbf{u}^{\mathrm{B}}(n)$ denote the M-by-1 vector obtained by rearranging the elements that constitute the observation vector $\mathbf{u}(n)$ *backward*. We illustrate this operation by writing

$$\mathbf{u}^{\mathrm{BT}}(n) = [u(n - M + 1), u(n - M + 2), \ldots, u(n)], \tag{1.24}$$

where the superscript B denotes the backward rearrangement of a vector. The correlation matrix of the vector $\mathbf{u}^{\mathrm{B}}(n)$ is, by definition,

$$\mathbb{E}[\mathbf{u}^{\mathrm{B}}(n)\mathbf{u}^{\mathrm{BH}}(n)] = \begin{bmatrix} r(0) & r^*(1) & \cdots & r^*(M-1) \\ r(1) & r(0) & \cdots & r^*(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(M-1) & r(M-2) & \cdots & r(0) \end{bmatrix}. \tag{1.25}$$

Hence, comparing the expanded correlation matrix of Eq. (1.25) with that of Eq. (1.21), we see that

$$\mathbb{E}[\mathbf{u}^{\mathrm{B}}(n)\mathbf{u}^{\mathrm{BH}}(n)] = \mathbf{R}^{\mathrm{T}},$$

which is the desired result.

Property 6. *The correlation matrices $\mathbf{R}_M$ and $\mathbf{R}_{M+1}$ of a stationary discrete-time stochastic process, pertaining to M and $M+1$ observations of the process, respectively, are related by*

$$\mathbf{R}_{M+1} = \left[\begin{array}{c|c} r(0) & \mathbf{r}^{\mathrm{H}} \\ \hline \mathbf{r} & \mathbf{R}_M \end{array}\right] \tag{1.26}$$

or, equivalently,

$$\mathbf{R}_{M+1} = \left[\begin{array}{c|c} \mathbf{R}_M & \mathbf{r}^{\mathrm{B}*} \\ \hline \mathbf{r}^{\mathrm{BT}} & \mathrm{r}(0) \end{array}\right], \tag{1.27}$$

where $r(0)$ is the autocorrelation of the process for a lag of zero,

$$\mathbf{r}^{\mathrm{H}} = [r(1), r(2), \ldots, r(M)], \tag{1.28}$$

and

$$\mathbf{r}^{\mathrm{BT}} = [r(-M), r(-M+1), \ldots, r(-1)]. \tag{1.29}$$

Note that in describing Property 6 we have added a subscript, M or $M+1$, to the symbol for the correlation matrix in order to display dependence on the number of observations used to define that matrix. We follow such a practice (in the context of the

correlation matrix and other vector quantities) *only* when the issue at hand involves dependence on the number of observations or dimensions of the matrix.

To derive Eq. (1.26), we express the correlation matrix $\mathbf{R}_{M+1}$ in its expanded form, partitioned as follows:

$$\mathbf{R}_{M+1} = \left[\begin{array}{c|ccc} r(0) & r(1) & r(2) & \cdots & r(M) \\ \hline r^*(1) & r(0) & r(1) & \cdots & r(M-1) \\ r^*(2) & r^*(1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r^*(M) & r^*(M-1) & r^*(M-2) & \cdots & r(0) \end{array}\right]. \tag{1.30}$$

Using Eqs. (1.18), (1.20), and (1.28) in Eq. (1.30), we obtain Eq. (1.26). Note that, according to this relation, the observation vector $\mathbf{u}_{M+1}(n)$ is *partitioned* in the form

$$\begin{aligned} \mathbf{u}_{M+1}(n) &= \left[\begin{array}{c} u(n) \\ \hline u(n-1) \\ u(n-2) \\ \vdots \\ u(n-M) \end{array}\right] \\ &= \left[\begin{array}{c} u(n) \\ \hline \mathbf{u}_M(n-1) \end{array}\right], \end{aligned} \tag{1.31}$$

where the subscript $M+1$ is intended to denote the fact that the vector $\mathbf{u}_{M+1}(n)$ has $M+1$ elements, and likewise for $\mathbf{u}_M(n)$.

To prove the relation of Eq. (1.27), we express the correlation matrix $\mathbf{R}_{M+1}$ in its expanded form, partitioned in the alternative form

$$\mathbf{R}_{M+1} = \left[\begin{array}{cccc|c} r(0) & r(1) & \cdots & r(M-1) & r(M) \\ r^*(1) & r(0) & \cdots & r(M-2) & r(M-1) \\ \vdots & \vdots & \ddots & \vdots & \\ r^*(M-1) & r^*(M-2) & \cdots & r(0) & r(1) \\ \hline r^*(M) & r^*(M-1) & \cdots & r^*(1) & r(0) \end{array}\right]. \tag{1.32}$$

Here again, using Eqs. (1.18), (1.20), and (1.29) in Eq. (1.32), we get the result given in Eq. (1.27). Note that according to this second relation the observation vector $\mathbf{u}_{M+1}(n)$ is partitioned in the alternative form

$$\begin{aligned} \mathbf{u}_{M+1}(n) &= \left[\begin{array}{c} u(n) \\ u(n-1) \\ \vdots \\ u(n-M+1) \\ \hline u(n-M) \end{array}\right] \\ &= \left[\begin{array}{c} \mathbf{u}_M(n) \\ \hline u(n-M) \end{array}\right]. \end{aligned} \tag{1.33}$$

1.4 CORRELATION MATRIX OF SINE WAVE PLUS NOISE

A time series of special interest is one that consists of a *complex sinusoid corrupted by additive noise*. Such a time series is representative of several important signal-processing applications. In the *temporal context*, for example, this type of series represents the composite signal at the input of a receiver, with the complex sinusoid representing a *target signal* and the noise representing thermal noise generated at the front end of the receiver. In the *spatial context*, it represents the received signal in a linear array of sensors, with the complex sinusoid representing a *plane wave* produced by a remote source (emitter) and the noise representing *sensor noise*.

Let α denote the amplitude of the complex sinusoid and ω denote its angular frequency. Let $\nu(n)$ denote a sample of the noise, assumed to have zero mean. We may then express a sample of the time series that consists of the complex sinusoid plus noise as

$$u(n) = \alpha \exp(j\omega n) + \nu(n), \qquad n = 0, 1, \ldots, N-1. \tag{1.34}$$

The sources of the complex sinusoid and the noise are naturally independent of each other. Since, by assumption, the noise component $\nu(n)$ has zero mean, we see from Eq. (1.34) that the mean of $u(n)$ is equal to $\alpha \exp(j\omega n)$.

To calculate the autocorrelation function of the process $u(n)$, we clearly need to know the autocorrelation function of the noise $\nu(n)$. To proceed, then, we assume a special form of noise characterized by the autocorrelation function

$$\mathbb{E}[\nu(n)\nu^*(n-k)] = \begin{cases} \sigma_\nu^2, & k = 0 \\ 0, & k \neq 0 \end{cases}. \tag{1.35}$$

Such a form of noise is commonly referred to as *white noise*; more will be said about it in Section 1.14. Since the sources responsible for the generation of the complex sinusoid and the noise are independent and, therefore, uncorrelated, it follows that the autocorrelation function of the process $u(n)$ equals the sum of the autocorrelation functions of its two individual components. Accordingly, using Eqs. (1.34) and (1.35), we find that the autocorrelation function of the process $u(n)$ for a lag k is given by

$$\begin{aligned} r(k) &= \mathbb{E}[u(n)u^*(n-k)] \\ &= \begin{cases} |\alpha|^2 + \sigma_\nu^2, & k = 0 \\ |\alpha|^2 \exp(j\omega k), & k \neq 0 \end{cases}, \end{aligned} \tag{1.36}$$

where $|\alpha|$ is the magnitude of the complex amplitude α. Note that for a lag $k \neq 0$, the autocorrelation function $r(k)$ varies with k in the same sinusoidal fashion as the sample $u(n)$ varies with n, except for a change in amplitude. Given the series of samples $u(n)$, $u(n-1), \ldots, u(n-M+1)$, we may thus express the correlation matrix of $u(n)$ as

$$\mathbf{R} = |\alpha|^2 \begin{bmatrix} 1 + \frac{1}{\rho} & \exp(j\omega) & \cdots & \exp\big(j\omega(M-1)\big) \\ \exp(-j\omega) & 1 + \frac{1}{\rho} & \cdots & \exp\big(j\omega(M-2)\big) \\ \vdots & \vdots & \ddots & \vdots \\ \exp\big(-j\omega(M-1)\big) & \exp\big(-j\omega(M-2)\big) & \cdots & 1 + \frac{1}{\rho} \end{bmatrix}, \tag{1.37}$$

where

$$\rho = \frac{|\alpha|^2}{\sigma_\nu^2} \tag{1.38}$$

is the *signal-to-noise ratio*. The correlation matrix $\mathbf{R}$ of Eq. (1.37) has all of the properties described in Section 1.3; the reader is invited to verify them.

Equation (1.36) provides the mathematical basis of a two-step practical procedure for estimating the parameters of a complex sinusoid in the presence of additive noise:

1. Measure the mean-square value $r(0)$ of the process $u(n)$. Hence, given the noise variance σ_ν^2, determine the magnitude $|\alpha|$.
2. Measure the autocorrelation function $r(k)$ of the process $u(n)$ for a lag $k \neq 0$. Hence, given $|\alpha|^2$ from step 1, determine the angular frequency ω.

Note that this estimation procedure is *invariant to the phase of* α, a property that is a direct consequence of the definition of the autocorrelation function $r(k)$.

EXAMPLE 1

Consider the idealized case of a noiseless sinusoid of angular frequency ω. For the purpose of illustration, we assume that the time series of interest consists of three uniformly spaced samples drawn from this sinusoid. Hence, setting the signal-to-noise ratio $\rho = \infty$ and the number of samples $M = 3$, we find from Eq. (1.37) that the correlation matrix of the time series so obtained is

$$\mathbf{R} = |\alpha|^2 \begin{bmatrix} 1 & \exp(\mathrm{j}\omega) & \exp(\mathrm{j}2\omega) \\ \exp(-\mathrm{j}\omega) & 1 & \exp(\mathrm{j}\omega) \\ \exp(-\mathrm{j}2\omega) & \exp(-\mathrm{j}\omega) & 1 \end{bmatrix}.$$

From this expression, we readily see that the determinant of $\mathbf{R}$ and all principal minors are identically zero. Hence, this correlation matrix is singular.

We may generalize the result we have just obtained by stating that when a process $u(n)$ consists of M samples drawn from the sum of K sinusoids with $K < M$ and there is *no* additive noise, then the correlation matrix of that process is singular.

1.5 STOCHASTIC MODELS

The term *model* is used for any hypothesis that may be applied to explain or describe the hidden laws that are supposed to govern or constrain the generation of physical data of interest. The representation of a stochastic process by a model dates back to an idea by Yule (1927). The idea is that a time series $u(n)$ consisting of highly correlated observations may be generated by applying a series of statistically independent "shocks" to a linear filter, as in Fig. 1.1. The shocks are random variables drawn from a fixed distribution that is usually assumed to be *Gaussian* with zero mean and constant variance. Such a series of random variables constitutes a purely random process, commonly referred to as *white Gaussian noise*. Specifically, we may describe the input $\nu(n)$ in the figure in statistical terms as

$$\mathbb{E}[\nu(n)] = 0 \qquad \text{for all } n \tag{1.39}$$

FIGURE 1.1 Stochastic model.

and

$$\mathbb{E}[\nu(n)\nu^*(k)] = \begin{cases} \sigma_\nu^2, & k = n \\ 0, & \text{otherwise} \end{cases}, \tag{1.40}$$

where σ_ν^2 is the noise variance. Equation (1.39) follows from the zero-mean assumption, and Eq. (1.40) follows from the fact that white noise has a flat power spectrum, as in white light. The implication of the Gaussian assumption is discussed in Section 1.11.

In general, the time-domain description of the input–output relation for the stochastic model of Fig. 1.1 may be described as follows:

$$\begin{pmatrix} \text{present value} \\ \text{of model output} \end{pmatrix} + \begin{pmatrix} \text{linear combination} \\ \text{of past values} \\ \text{of model output} \end{pmatrix} = \begin{pmatrix} \text{linear combination of} \\ \text{present and past values} \\ \text{of model input} \end{pmatrix}. \tag{1.41}$$

A stochastic process so described is referred to as a *linear process*.

The structure of the linear filter in Fig. 1.1 is determined by the manner in which the two linear combinations indicated in Eq. (1.41) are formulated. We may thus identify three popular types of linear stochastic models:

1. Autoregressive models, in which no past values of the model input are used.
2. Moving-average models, in which no past values of the model output are used.
3. Mixed autoregressive–moving-average models, in which the description of Eq. (1.41) applies in its entire form. Hence, this class of stochastic models includes autoregressive and moving-average models as special cases.

Autoregressive Models

We say that the time series $u(n), u(n-1), \ldots, u(n-M)$ represents the realization of an *autoregressive (AR) process of order M* if it satisfies the difference equation

$$u(n) + a_1^* u(n-1) + \cdots + a_M^* u(n-M) = \nu(n), \tag{1.42}$$

where $a_1, a_2, \ldots, a_M$ are constants called the AR *parameters* and $\nu(n)$ is white noise. The term $a_k^* u(n-k)$ is the scalar version of the *inner product* of a_k and $u(n-k)$, where $k = 1, \ldots, M$.

To explain the reason for the term "autoregressive," we rewrite Eq. (1.42) in the form

$$u(n) = w_1^* u(n-1) + w_2^* u(n-2) + \cdots + w_M^* u(n-M) + \nu(n), \tag{1.43}$$

where $w_k = -a_k$. We thus see that the present value of the process—that is, $u(n)$—equals a *finite linear combination of past values* of the process, $u(n-1), \ldots, u(n-M)$ plus an *error term* $\nu(n)$. We now see the reason for the term "autoregressive": A linear model

$$y = \sum_{k=1}^{M} w_k^* x_k + \nu$$

relating a *dependent* variable y to a set of *independent* variables $x_1, x_2, \ldots, x_M$ plus an error term ν is often referred to as a *regression model*, and y is said to be "regressed" on $x_1, x_2, \ldots, x_M$. In Eq. (1.43), the variable $u(n)$ is *regressed* on previous values of *itself*—hence the term "autoregressive."

The left-hand side of Eq. (1.42) represents the *convolution* of the input sequence $\{u(n)\}$ and the sequence of parameters $\{a_n^*\}$. To highlight this point, we rewrite the equation in the form of a convolution sum; that is,

$$\sum_{k=0}^{M} a_k^* u(n-k) = \nu(n), \tag{1.44}$$

where $a_0 = 1$. By taking the *z-transform* of both sides of Eq. (1.44), we transform the convolution sum on the left-hand side of the equation into a multiplication of the z-transforms of the two sequences $\{u(n)\}$ and $\{a_n^*\}$. Let

$$H_A(z) = \sum_{n=0}^{M} a_n^* z^{-n} \tag{1.45}$$

denote the z-transform of the sequence $\{a_n^*\}$, and let

$$U(z) = \sum_{n=0}^{\infty} u(n) z^{-n}, \tag{1.46}$$

where z is a *complex variable*, denote the z-transform of the input sequence $\{u(n)\}$. We may thus transform the difference Eq. (1.42) into the equivalent form

$$H_A(z)U(z) = V(z), \tag{1.47}$$

where

$$V(z) = \sum_{n=0}^{\infty} \nu(n) z^{-n}. \tag{1.48}$$

The z-transform of Eq. (1.47) offers two interpretations, depending on whether the AR process $u(n)$ is viewed as the input or output of interest:

1. Given the AR process $u(n)$, we may use the filter shown in Fig. 1.2(a) to produce the white noise $\nu(n)$ as output. The parameters of this filter bear a one-to-one correspondence with those of $u(n)$. Accordingly, such a filter represents a *process analyzer* with discrete transfer function $H_A(z) = V(z)/U(z)$. The impulse response of the AR process analyzer [i.e., the inverse z-transform of $H_A(z)$] has *finite duration*.

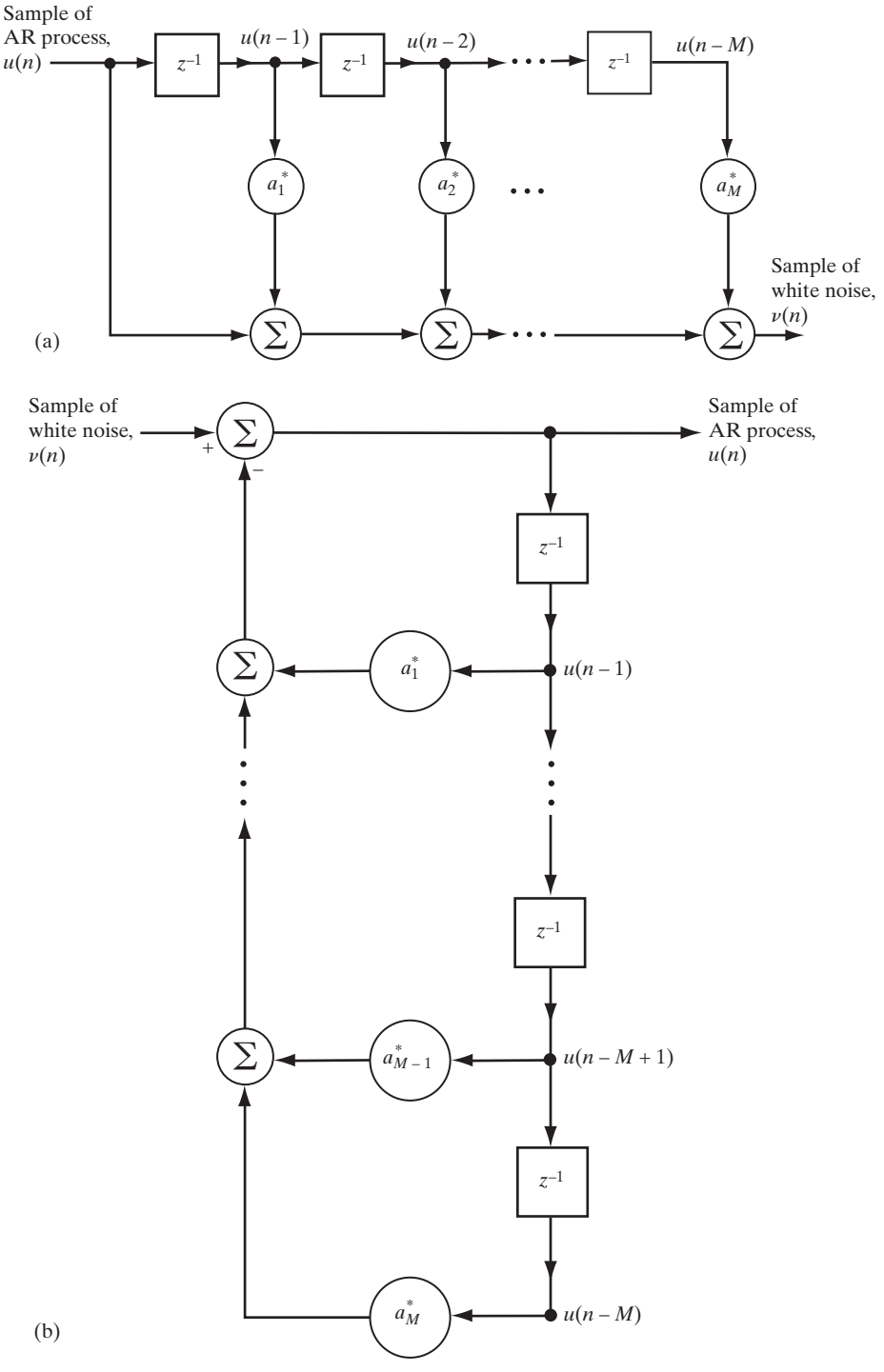


FIGURE 1.2 (a) AR process analyzer; (b) AR process generator.

2. With the white noise $\nu(n)$ acting as input, we may use the filter shown in Fig. 1.2(b) to produce the AR process $u(n)$ as output. Accordingly, this second filter represents a *process generator*, whose transfer function equals

$$\begin{aligned} H_G(z) &= \frac{U(z)}{V(z)} \\ &= \frac{1}{H_A(z)} \\ &= \frac{1}{\sum_{n=0}^{M} a_n^* z^{-n}}. \end{aligned} \tag{1.49}$$

The impulse response of the AR process generator [i.e., the inverse z-transform of $H_G(z)$] has *infinite duration*.

The AR process analyzer of Fig. 1.2(a) is an *all-zero filter*. It is so called because its transfer function $H_A(z)$ is completely defined by specifying the locations of its *zeros*. This filter is inherently stable.

The AR process generator of Fig. 1.2(b) is an *all-pole filter*. It is so called because its transfer function $H_G(z)$ is completely defined by specifying the locations of its *poles*, as shown by

$$H_G(z) = \frac{1}{(1 - p_1 z^{-1})(1 - p_2 z^{-1}) \ \ldots \ (1 - p_M z^{-1})}. \tag{1.50}$$

The parameters $p_1, p_2, \ldots, p_M$ are *poles* of $H_G(z)$; they are defined by the roots of the *characteristic equation*

$$1 + a_1^* z^{-1} + a_2^* z^{-2} + \cdots + a_M^* z^{-M} = 0. \tag{1.51}$$

For the all-pole AR process generator of Fig. 1.2(b) to be stable, the roots of the characteristic Eq. (1.51) must all lie inside the unit circle in the z-plane. This is also a necessary and sufficient condition for wide-sense stationarity of the AR process produced by the model of Fig. 1.2(b). We shall have more to say on the issue of stationarity in Section 1.7.

Moving-Average Models

In a *moving-average (MA) model*, the discrete-time linear filter of Fig. 1.1 consists of an *all-zero filter* driven by white noise. The resulting process $u(n)$, produced at the filter output, is described by the difference equation

$$u(n) = \nu(n) + b_1^* \nu(n - 1) + \cdots + b_K^* \nu(n - K), \tag{1.52}$$

where $b_1, \ldots, b_K$ are constants called the *MA parameters* and $\nu(n)$ is white noise of zero mean and variance σ_ν^2. Except for $\nu(n)$, each term on the right-hand side of Eq. (1.52) represents the scalar version of an inner product. The *order* of the MA process equals K. The term "moving average" is a rather quaint one; nevertheless, its use is firmly established in the literature. Its usage arose in the following way: If we are given a complete temporal realization of the white noise $\nu(n)$, we may compute $u(n)$ by constructing a *weighted average* of the sample values $\nu(n), \nu(n-1), \ldots, \nu(n-K)$.

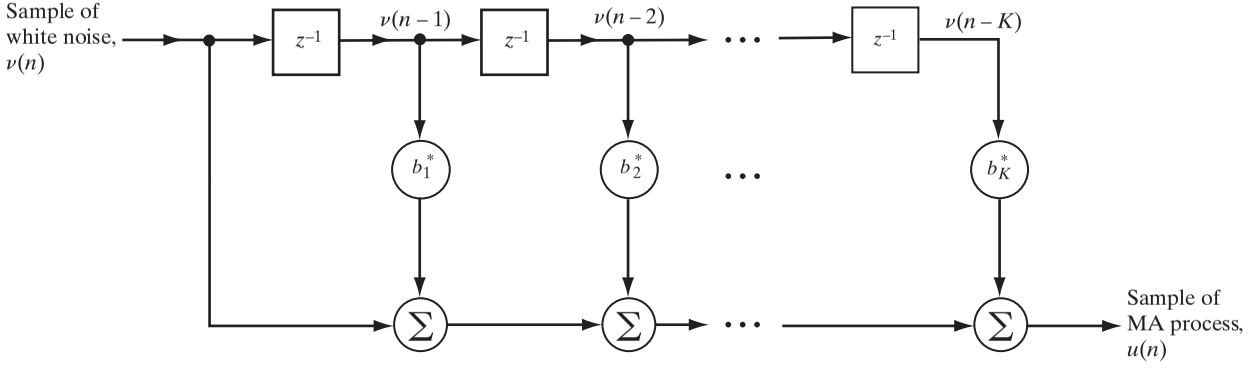


FIGURE 1.3 MA model (process generator).

From Eq. (1.52), we readily obtain the MA model (i.e., process generator) depicted in Fig. 1.3. Specifically, we start with white noise $\nu(n)$ at the model input and generate an MA process $u(n)$ of order K at the model output. To proceed in the reverse manner [i.e., to produce the white noise $\nu(n)$, given the MA process $u(n)$], we require the use of an *all-pole filter*. In other words, the filters used in the generation and analysis of an MA process are the *opposite* of those used in the case of an AR process.

Autoregressive–Moving-Average Models

To generate a mixed *autoregressive–moving-average (ARMA) process* $u(n)$, we use the discrete-time linear filter of Fig. 1.1 with a transfer function that contains *both poles and zeros*. Accordingly, given the white noise $\nu(n)$ as the filter input, the ARMA process $u(n)$ produced at the filter output is described by the difference equation

$$u(n) + a_1^* u(n-1) + \cdots + a_M^* u(n-M) = \nu(n) + b_1^* \nu(n-1) + \cdots + b_K^* \nu(n-K), \tag{1.53}$$

where $a_1, \ldots, a_M$ and $b_1, \ldots, b_K$ are called the *ARMA parameters*. Except for $u(n)$ on the left-hand side of Eq. (1.53) and $\nu(n)$ on the right-hand side, all of the terms represent scalar versions of inner products. The *order* of the ARMA process is expressed by (M, K).

From Eq. (1.53), we readily deduce the ARMA model (i.e., the process generator) depicted in Fig. 1.4. Comparing this figure with Fig. 1.2(b) and Fig. 1.3, we clearly see that AR and MA models are indeed special cases of an ARMA model, as indicated previously.

The transfer function of the ARMA process generator in Fig. 1.4 has both poles and zeros. Similarly, the ARMA analyzer used to generate white noise $\nu(n)$, given an ARMA process $u(n)$, is characterized by a transfer function containing both poles and zeros.

From a computational viewpoint, the AR model has an advantage over the MA and ARMA models: The computation of the AR coefficients in the model of Fig. 1.2(a) involves a system of *linear equations* known as the Yule–Walker equations, details of which are given in Section 1.8. On the other hand, the computation of the MA coefficients in the model of Fig. 1.3 and the computation of the ARMA coefficients in the model of Fig. 1.4 are much more complicated. Both of these computations require solving systems of *nonlinear equations*. It is for this reason that, in practice, we find that AR models are more popular than MA and ARMA models. The wide application of AR models may also be justified by virtue of a fundamental theorem of time-series analysis, which is discussed next.

1.6 WOLD DECOMPOSITION

Wold (1938) proved a fundamental theorem which states that any stationary discrete-time stochastic process may be decomposed into the sum of a *general linear process* and a *predictable process*, with these two processes being uncorrelated with each other. More precisely, Wold proved the following result: Any stationary discrete-time stochastic process $x(n)$ may be expressed in the form

$$x(n) = u(n) + s(n), \tag{1.54}$$

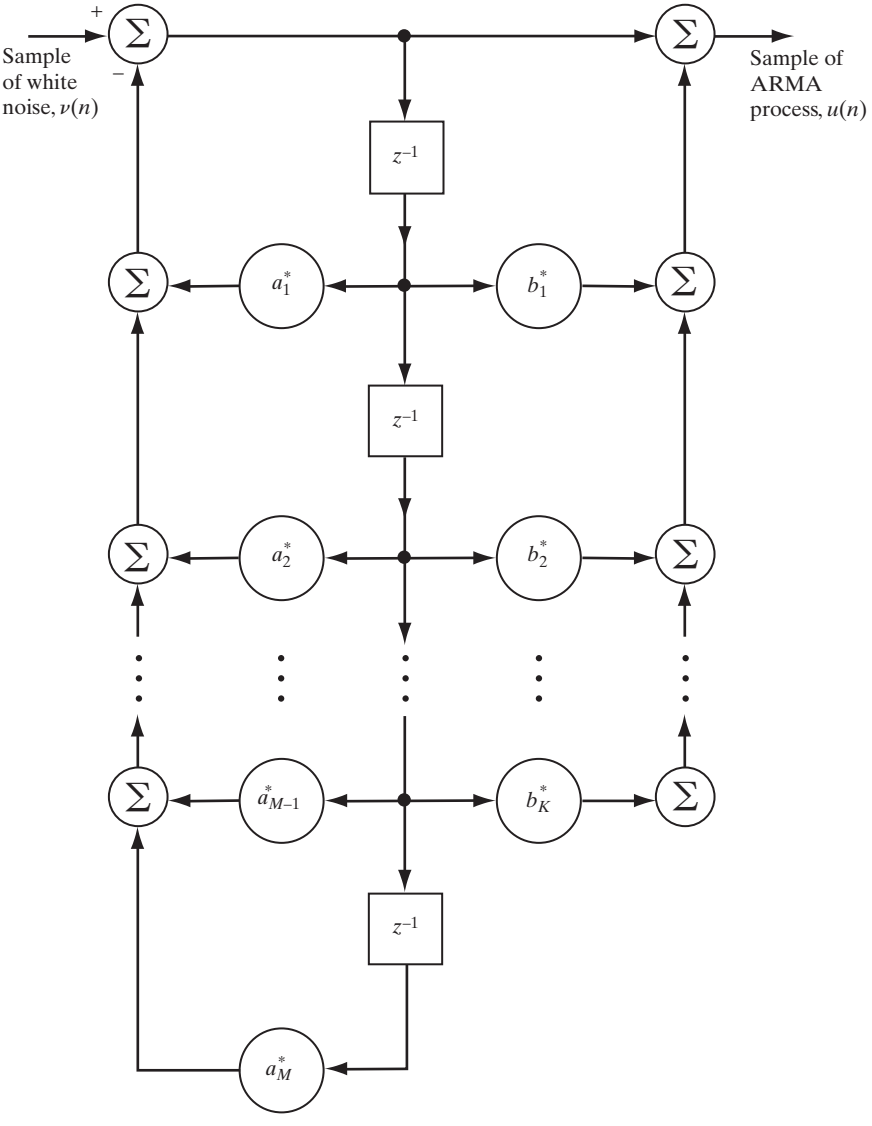


FIGURE 1.4 ARMA model (process generator) of order (M, K), assuming that $M > K$.

where

1. $u(n)$ and $s(n)$ are uncorrelated processes;
2. $u(n)$ is a general linear process represented by the MA model, viz.,

$$u(n) = \sum_{k=0}^{\infty} b_k^* \nu(n - k) \tag{1.55}$$

with $b_0 = 1$, and

$$\sum_{k=0}^{\infty} |b_k|^2 < \infty,$$

and where $\nu(n)$ is white noise uncorrelated with $s(n)$—that is,

$$\mathbb{E}[\nu(n)s^*(k)] = 0 \qquad \text{for all } (n, k);$$

3. $s(n)$ is a predictable process; that is, the process can be predicted from its own past with zero prediction variance.

This result is known as *Wold's decomposition theorem*. A proof of the theorem is given in Priestley (1981).

According to Eq. (1.55), the general linear process $u(n)$ may be generated by feeding an *all-zero filter* with white noise $\nu(n)$, as in Fig. 1.5(a). The zeros of the transfer function of this filter equal the roots of the equation

$$B(z) = \sum_{n=0}^{\infty} b_n^* z^{-n} = 0.$$

A solution of particular interest is an all-zero filter that is *minimum phase*, which means that all the zeros of the polynomial $B(z)$ lie inside the unit circle. In such a case, we may replace the all-zero filter with an *equivalent* all-pole filter that has the same impulse response $h_n = b_n$, as in Fig. 1.5(b). This means that, except for a predictable component, a stationary discrete-time stochastic process may also be represented as an AR process of the appropriate order, subject to the just-mentioned restriction on $B(z)$. The basic difference between the MA and AR models is that $B(z)$ operates on the input $\nu(n)$ in the MA model, whereas the inverse $B^{-1}(z)$ operates on the output $u(n)$ in the AR model.

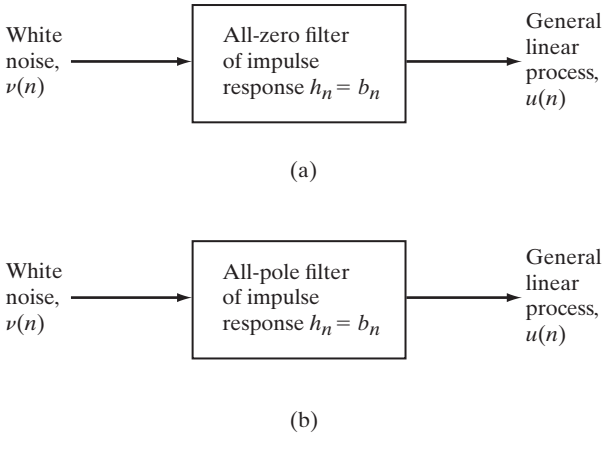


FIGURE 1.5 (a) Model, based on all-zero filter, for generating the linear process $u(n)$; (b) model, based on all-pole filter, for generating the general linear process $u(n)$. Both filters have exactly the same impulse response.

1.7 ASYMPTOTIC STATIONARITY OF AN AUTOREGRESSIVE PROCESS

Equation (1.42) is a *linear, constant-coefficient difference equation of order M* in which $\nu(n)$ plays the role of *input* or *driving function* and $u(n)$ that of *output* or *solution*. By using the *classical method*[2] for solving such an equation, we may formally express the solution $u(n)$ as the sum of a *complementary function* $u_c(n)$ and a *particular solution* $u_p(n)$:

$$u(n) = u_c(n) + u_p(n). \tag{1.56}$$

The evaluation of the solution $u(n)$ may thus proceed in two stages:

1. The complementary function $u_c(n)$ is the solution of the *homogeneous equation*

$$u(n) + a_1^* u(n-1) + a_2^* u(n-2) + \cdots + a_M^* u(n-M) = 0.$$

In general, $u_c(n)$ is therefore of the form

$$u_c(n) = B_1 p_1^n + B_2 p_2^n + \cdots + B_M p_M^n, \tag{1.57}$$

where $B_1, B_2, \ldots, B_M$ are arbitrary constants and $p_1, p_2, \ldots, p_M$ are roots of the characteristic Eq. (1.51).

2. The particular solution is defined by

$$u_p(n) = H_G(D)\,[\nu(n)], \tag{1.58}$$

where D is the *unit-delay operator* and the operator $H_G(D)$ is obtained by substituting D for z^{-1} in the discrete-transfer function of Eq. (1.49). The unit-delay operator D has the property that

$$D^k[u(n)] = u(n-k), \qquad k = 0, 1, 2, \ldots. \tag{1.59}$$

The constants $B_1, B_2, \ldots, B_M$ are determined by the choice of *initial conditions*, which equal M in number. It is customary to set

$$\begin{aligned} u(0) &= 0, \\ u(-1) &= 0, \\ &\vdots \\ u(-M+1) &= 0. \end{aligned} \tag{1.60}$$

This is equivalent to setting the output of the model in Fig. 1.2(b), as well as the succeeding $(M-1)$ tap inputs, equal to zero at time $n = 0$. Thus, by substituting these initial conditions into Eqs. (1.56) through (1.58), we obtain a set of M simultaneous equations that can be solved for the constants $B_1, B_2, \ldots, B_M$.

The result of imposing the initial conditions of Eq. (1.60) on the solution $u(n)$ is to make the discrete-time stochastic process represented by that solution nonstationary. On reflection, it is clear that this must be so, since we have given a "special status" to

[2]We may also use the z-transform method to solve the difference Eq. (1.42). However, for the discussion presented here, we find it more informative to use the classical method.

the time point $n = 0$ and the property of *invariance under a shift of time origin* cannot hold, even for second-order moments. If, however, $u(n)$ is able to "forget" its initial conditions, the resulting process is asymptotically stationary in the sense that it settles down to a stationary behavior as n approaches infinity (Priestley, 1981). This requirement may be achieved by choosing the parameters of the AR model in Fig. 1.2(b) such that the complementary function $u_c(n)$ decays to zero as n approaches infinity. From Eq. (1.57), we see that, for arbitrary constants in the equation, this requirement can be met if and only if

$$|p_k| < 1 \qquad \text{for all } k.$$

Hence, *for asymptotic stationarity of the discrete-time stochastic process represented by the solution u(n), we require that all the poles of the filter in the AR model lie inside the unit circle in the z-plane*. This requirement is intuitively satisfying.

Correlation Function of an Asymptotically Stationary AR Process

Assuming that the condition for asymptotic stationarity is satisfied, we may derive an important recursive relation for the autocorrelation function of the resulting AR process. We begin by multiplying both sides of Eq. (1.42) by $u^*(n - l)$, and then we apply the expectation operator, thereby obtaining

$$\mathbb{E}\left[\sum_{k=0}^{M} a_k^* u(n - k)u^*(n - l)\right] = \mathbb{E}[\nu(n)u^*(n - l)]. \tag{1.61}$$

Next, we simplify the left-hand side of Eq. (1.61) by interchanging the expectation and summation and by recognizing that the expectation $\mathbb{E}[u(n-k)u^*(n-l)]$ equals the autocorrelation function of the AR process for a lag of $l-k$. We simplify the right-hand side by observing that the expectation $\mathbb{E}[\nu(n)u^*(n-l)]$ is zero for $l>0$, since $u(n-l)$ involves only samples of white noise at the filter input in Fig. 1.2(b) up to time $n - l$, which are uncorrelated with the white-noise sample $\nu(n)$. Accordingly, we simplify Eq. (1.61) to

$$\sum_{k=0}^{M} a_k^* r(l - k) = 0, \quad l > 0, \tag{1.62}$$

where $a_0 = 1$. We thus see that the autocorrelation function of the AR process satisfies the difference equation

$$r(l) = w_1^* r(l - 1) + w_2^* r(l - 2) + \cdots + w_M^* r(l - M), \quad l > 0, \tag{1.63}$$

where $w_k = -a_k, k = 1, 2, \ldots, M$. Note that Eq. (1.63) is analogous to the difference equation satisfied by the AR process $u(n)$ itself.

We may express the general solution of Eq. (1.63) as

$$r(m) = \sum_{k=1}^{M} C_k p_k^m, \tag{1.64}$$

where $C_1, C_2, \ldots, C_M$ are constants and $p_1, p_2, \ldots, p_M$ are roots of the characteristic Eq. (1.51). Note that when the AR model of Fig. 1.2(b) satisfies the condition for

asymptotic stationarity, $|p_k| < 1$ for all k, in which case the autocorrelation function $r(m)$ approaches zero as the lag m approaches infinity.

The exact form of the contribution made by a pole p_k in Eq. (1.64) depends on whether the pole is real or complex. When p_k is real, the corresponding contribution decays geometrically to zero as the lag m increases. We refer to such a contribution as a *damped exponential*. On the other hand, complex poles occur in conjugate pairs, and the contribution of a complex-conjugate pair of poles is in the form of a *damped sine wave*. We thus find that, in general, the autocorrelation function of an asymptotically stationary AR process consists of a mixture of damped exponentials and damped sine waves.

1.8 YULE–WALKER EQUATIONS

In order to uniquely define the AR model of order M, depicted in Fig. 1.2(b), we need to specify two sets of model parameters:

1. The AR coefficients $a_1, a_2, \ldots, a_M$.
2. The variance σ_ν^2 of the white noise $\nu(n)$ used as excitation.

We now address these two issues in turn.

First, writing Eq. (1.63) for $l = 1, 2, \ldots, M$, we get a set of M simultaneous equations with the values $r(0), r(1), \ldots, r(M)$ of the autocorrelation function of the AR process as the known quantities and the AR parameters $a_1, a_2, \ldots, a_M$ as the unknowns. This set of equations may be expressed in the expanded matrix form

$$\begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r^*(1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r^*(M-1) & r^*(M-2) & \cdots & r(0) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} = \begin{bmatrix} r^*(1) \\ r^*(2) \\ \vdots \\ r^*(M) \end{bmatrix}, \tag{1.65}$$

where $w_k = -a_k$. The set of Eqs. (1.65) is called the *Yule–Walker equations* (Yule, 1927; Walker, 1931).

We may express the Yule–Walker equations in the compact matrix form

$$\mathbf{R}\mathbf{w} = \mathbf{r}. \tag{1.66}$$

Assuming that the correlation matrix $\mathbf{R}$ is nonsingular (i.e., the inverse matrix $\mathbf{R}^{-1}$ exists), we obtain the solution of Eq. (1.66) as

$$\mathbf{w} = \mathbf{R}^{-1}\mathbf{r}, \tag{1.67}$$

where

$$\mathbf{w} = [w_1, w_2, \ \ldots \ , w_M]^{\mathrm{T}}.$$

The correlation matrix $\mathbf{R}$ is defined by Eq. (1.21), and the vector $\mathbf{r}$ is defined by Eq. (1.28). From these two equations, we see that we may uniquely determine both the matrix $\mathbf{R}$ and the vector $\mathbf{r}$, given the autocorrelation sequence $r(0), r(1), \ldots, r(M)$. Hence, using Eq. (1.67), we may compute the coefficient vector $\mathbf{w}$ and, therefore, the AR coefficients $a_k = -w_k$, $k = 1, 2, \ldots, M$. In other words, there is a unique relationship

between the coefficients $a_1, a_2, \ldots, a_M$ of the AR model and the *normalized* correlation coefficients $\rho_1, \rho_2, \ldots, \rho_M$ of the AR process $u(n)$, as shown by

$$\{a_1, a_2, \ldots, a_M\} \rightleftharpoons \{\rho_1, \rho_2, \ldots, \rho_M\}, \tag{1.68}$$

where the kth *correlation coefficient* is defined by

$$\rho_k = \frac{r(k)}{r(0)}, \qquad k = 1, 2, \ldots, M. \tag{1.69}$$

Variance of the White Noise

For $l = 0$, we find that the expectation on the right-hand side of Eq. (1.61), in light of Eq. (1.42), assumes the special form

$$\begin{aligned} \mathbb{E}[\nu(n)u^*(n)] &= \mathbb{E}[\nu(n)\nu^*(n)] \\ &= \sigma_\nu^2, \end{aligned} \tag{1.70}$$

where σ_ν^2 is the variance of the zero-mean white noise $\nu(n)$. Accordingly, setting $l = 0$ in Eq. (1.61) and performing a complex conjugation on both sides, we get the formula

$$\sigma_\nu^2 = \sum_{k=0}^{M} a_k r(k), \tag{1.71}$$

with $a_0 = 1$, for the variance of the white noise. Hence, given the autocorrelations $r(0)$, $r(1), \ldots, r(M)$, we may determine the white-noise variance σ_ν^2.

1.9 COMPUTER EXPERIMENT: AUTOREGRESSIVE PROCESS OF ORDER TWO

To illustrate the theory just developed for the modeling of an AR process, we consider the example of a second-order AR process that is real valued.[3] Figure 1.6 shows the block diagram of the model used to generate this process. The time-domain description of the process is governed by the second-order difference equation

$$u(n) + a_1 u(n-1) + a_2 u(n-2) = \nu(n), \tag{1.72}$$

where $\nu(n)$ is drawn from a white-noise process of zero mean and variance σ_ν^2. Figure 1.7(a) shows one realization of this white-noise process. The variance σ_ν^2 is chosen to make the variance of $u(n)$ equal unity.

Conditions for Asymptotic Stationarity

The second-order AR process $u(n)$ has the characteristic equation

$$1 + a_1 z^{-1} + a_2 z^{-2} = 0. \tag{1.73}$$

[3]In this example, we follow the approach described by Box and Jenkins (1976).

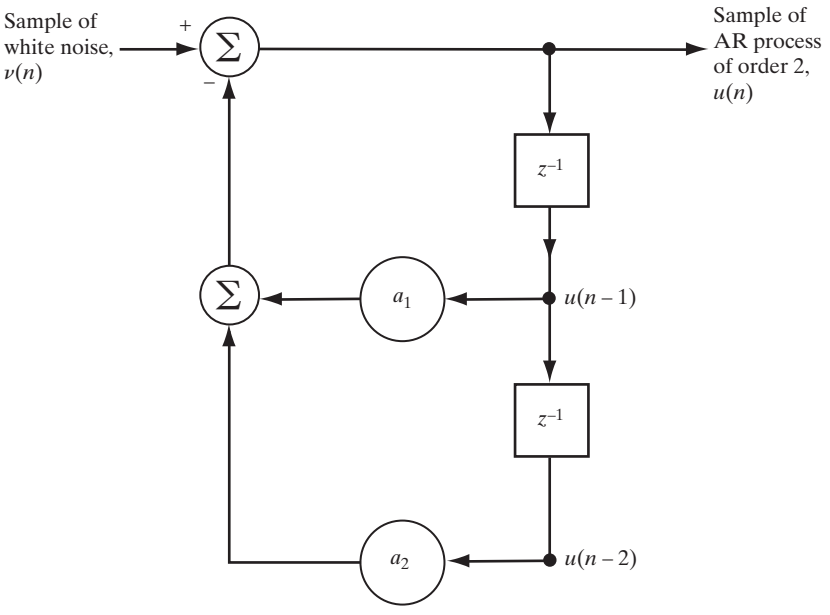


FIGURE 1.6 Model of (real-valued) AR process of order 2.

Let p_1 and p_2 denote the two roots of this equation:

$$p_1, p_2 = \frac{1}{2}\left(-a_1 \pm \sqrt{a_1^2 - 4a_2}\right). \tag{1.74}$$

To ensure the asymptotic stationarity of $u(n)$, we require that these two roots lie inside the unit circle in the z-plane. That is, both p_1 and p_2 must have a magnitude less than unity. This restriction, in turn, requires that the AR parameters a_1 and a_2 lie in the triangular region defined by

$$\begin{aligned} -1 &< a_2 + a_1, \\ -1 &< a_2 - a_1, \end{aligned} \tag{1.75}$$

and

$$-1 < a_2 < 1,$$

as shown in Fig. 1.8.

Autocorrelation Function

The autocorrelation function $r(m)$ of an asymptotically stationary AR process for lag m satisfies the difference Eq. (1.63). Hence, using this equation, we obtain the following second-order difference equation for the autocorrelation function of a second-order AR process:

$$r(m) + a_1 r(m-1) + a_2 r(m-2) = 0, \quad m > 0. \tag{1.76}$$

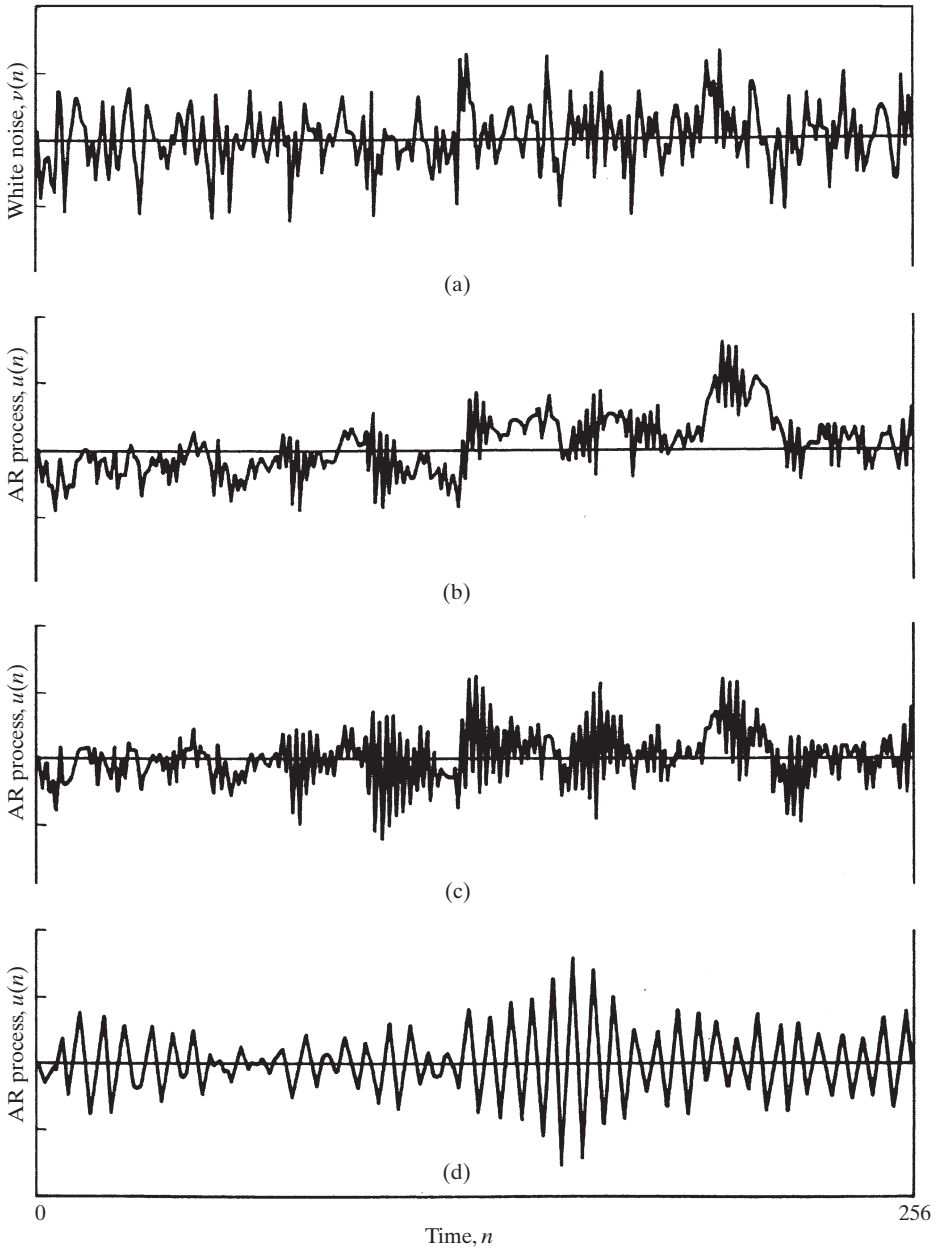


FIGURE 1.7 (a) One realization of white-noise input; (b), (c), (d) corresponding outputs of AR model of order 2 for parameters of Eqs. (1.79), (1.80), and (1.81), respectively.

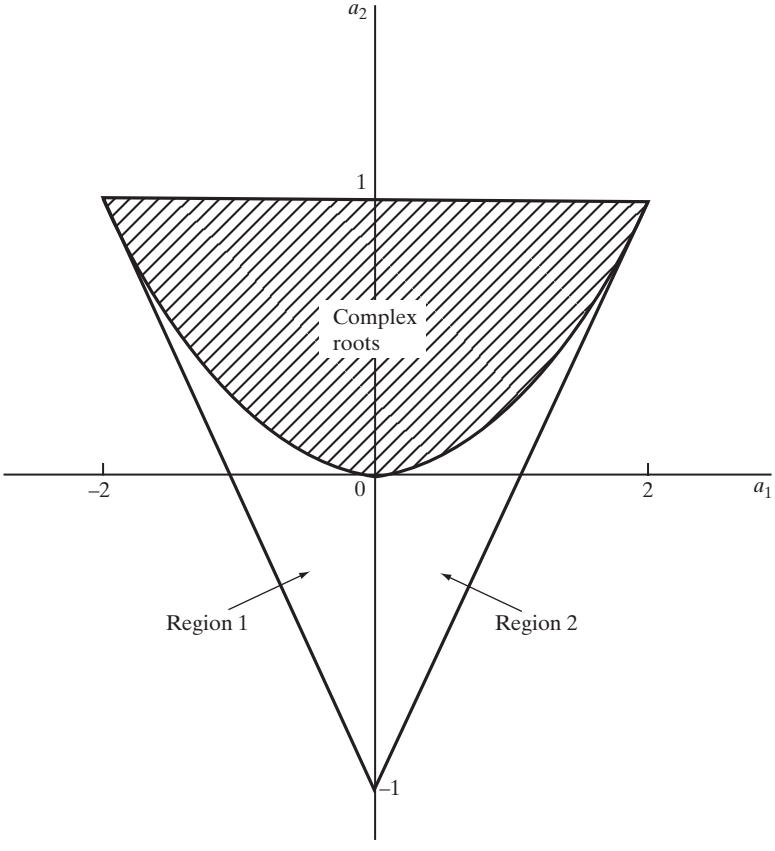


FIGURE 1.8 Permissible region for the AR parameters a_1 and a_2.

For the initial values, we have (as will be explained later)

$$r(0) = \sigma_u^2 \tag{1.77}$$

and

$$r(1) = \frac{-a_1}{1 + a_2}\sigma_u^2.$$

Thus, solving Eq. (1.76) for $r(m)$, we get (for $m > 0$)

$$r(m) = \sigma_u^2\left[\frac{p_1(p_2^2 - 1)}{(p_2 - p_1)(p_1p_2 + 1)}p_1^m - \frac{p_2(p_1^2 - 1)}{(p_2 - p_1)(p_1p_2 + 1)}p_2^m\right], \tag{1.78}$$

where p_1 and p_2 are defined by Eq. (1.74).

There are two specific cases to be considered, depending on whether the roots p_1 and p_2 are real or complex.

Case 1: *Real Roots*. This case occurs when

$$a_1^2 - 4a_2 > 0,$$

which corresponds to regions 1 and 2 below the parabolic boundary in Fig. 1.8. In region 1, the autocorrelation function remains positive as it damps out, corresponding to a positive dominant root. This situation is illustrated in Fig. 1.9(a) for the AR parameters

$$a_1 = -0.10$$

and (1.79)

$$a_2 = -0.8.$$

In Fig. 1.7(b), we show the time variation of the output of the model in Fig. 1.6 [with a_1 and a_2 assigned the values given in Eq. (1.79)]. This output is produced by the white-noise input shown in Fig. 1.7(a).

In region 2 of Fig. 1.8, the autocorrelation function alternates in sign as it damps out, corresponding to a negative dominant root. This situation is illustrated in Fig. 1.9(b) for the AR parameters

$$a_1 = 0.1$$

and (1.80)

$$a_2 = -0.8.$$

In Fig. 1.7(c), we show the time variation of the output of the model in Fig. 1.6 [with a_1 and a_2 assigned the values given in Eq. (1.80)]. This output is also produced by the white-noise input shown in Fig. 1.7(a).

Case 2: *Complex-Conjugate Roots.* This second case occurs when

$$a_1^2 - 4a_2 < 0,$$

which corresponds to the shaded region shown in Fig. 1.8 above the parabolic boundary. Here, the autocorrelation function displays a pseudoperiodic behavior, as illustrated in Fig. 1.9(c) for the AR parameters

$$a_1 = -0.975$$

and (1.81)

$$a_2 = 0.95.$$

In Fig. 1.7(d), we show the time variation of the output of the model in Fig. 1.6 [with a_1 and a_2 assigned the values given in Eq. (1.81)]. This output, too, is produced by the white-noise input shown in Fig. 1.7(a).

Yule–Walker Equations

Substituting the value $M = 2$ for the order of the AR model of Eq. (1.65), we get the following Yule–Walker equations for the second-order AR process:

$$\begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \end{bmatrix}. \tag{1.82}$$

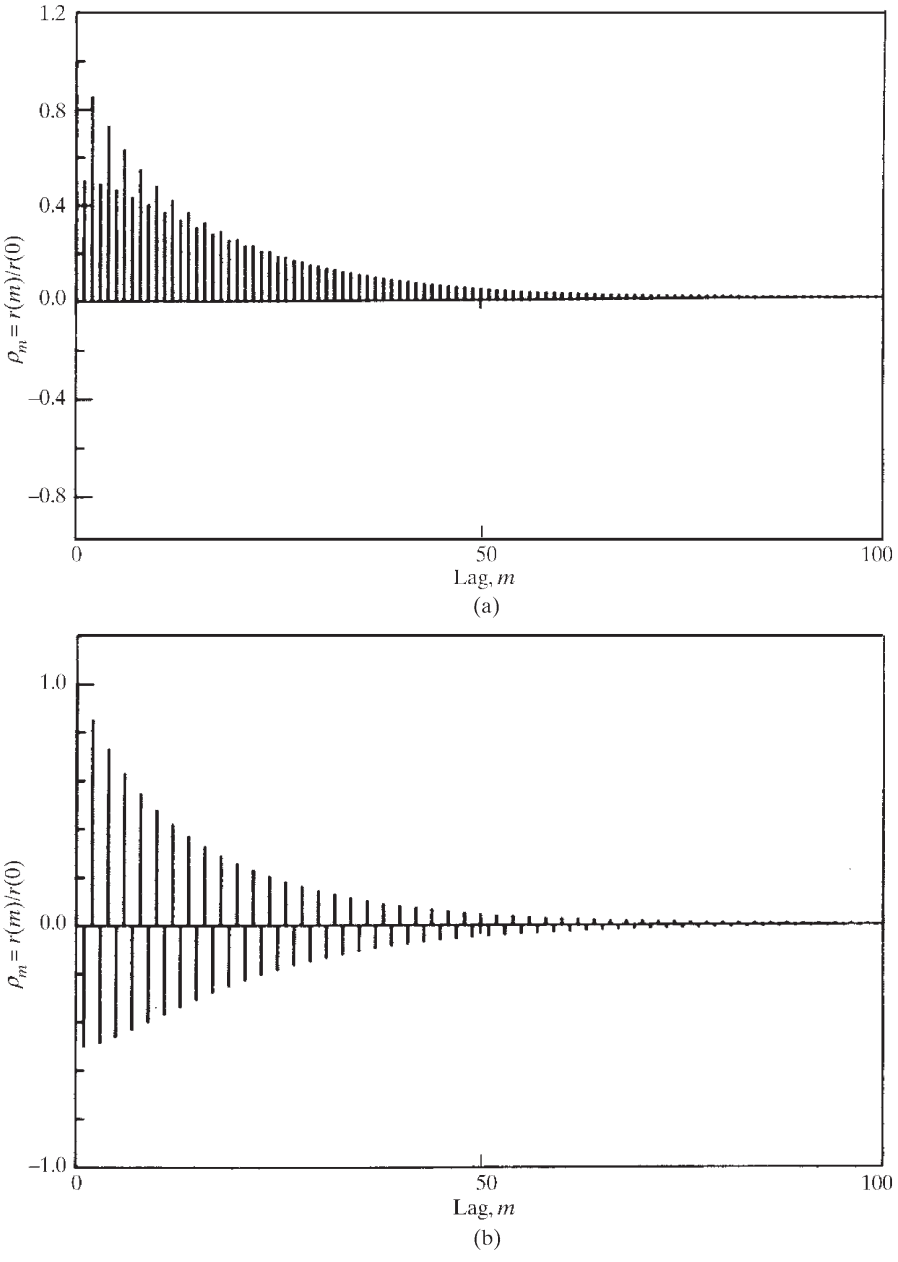
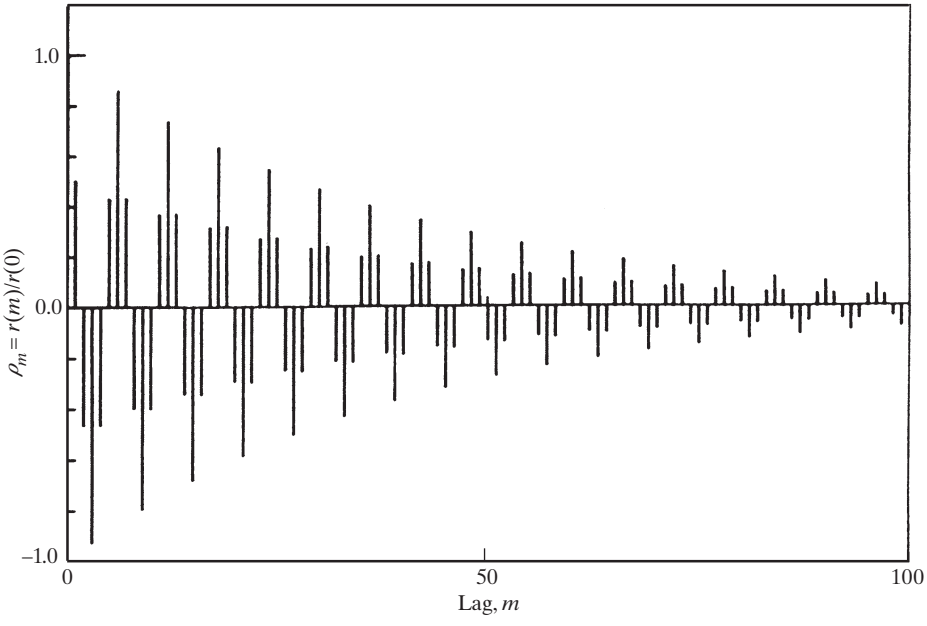


FIGURE 1.9 Plots of normalized autocorrelation function of real-valued second-order process; (a) $r(1) > 0$; (b) $r(1) < 0$; (c) conjugate roots, see next page.



(c)

FIGURE 1.9 *(continued)*

Here, we have used the fact that $r(-1) = r(1)$ for a real-valued process. Solving Eq. (1.82) for w_1 and w_2, we get

$$w_1 = -a_1 = \frac{r(1)[r(0) - r(2)]}{r^2(0) - r^2(1)}$$

and (1.83)

$$w_2 = -a_2 = \frac{r(0)r(2) - r^2(1)}{r^2(0) - r^2(1)}.$$

We may also use Eq. (1.82) to express $r(1)$ and $r(2)$ in terms of the AR parameters a_1 and a_2; that is,

$$r(1) = \left(\frac{-a_1}{1 + a_2}\right)\sigma_u^2$$

and (1.84)

$$r(2) = \left(-a_2 + \frac{a_1^2}{1 + a_2}\right)\sigma_u^2,$$

where $\sigma_u^2 = r(0)$. This solution explains the initial values for $r(0)$ and $r(1)$ that were quoted in Eq. (1.77).

The conditions for asymptotic stationarity of the second-order AR process are given in terms of the AR parameters a_1 and a_2 in Eq. (1.75). Using the expressions for $r(1)$ and $r(2)$ in terms of a_1 and a_2 given in Eq. (1.84), we may reformulate the conditions for asymptotic stationarity as

$$\begin{aligned} -1 < \rho_1 < 1, \\ -1 < \rho_2 < 1, \end{aligned} \tag{1.85}$$

and

$$\rho_1^2 < \frac{1}{2}(1 + \rho_2),$$

where

$$\rho_1 = \frac{r(1)}{r(0)}$$

and (1.86)

$$\rho_2 = \frac{r(2)}{r(0)}$$

are the normalized correlation coefficients. Fig. 1.10 shows the admissible region for ρ_1 and ρ_2.

Variance of the White Noise

Putting $M = 2$ in Eq. (1.71), we may express the variance of the white noise $\nu(n)$ as

$$\sigma_\nu^2 = r(0) + a_1 r(1) + a_2 r(2). \tag{1.87}$$

Next, substituting Eq. (1.84) into Eq. (1.87), and solving for $\sigma_u^2 = r(0)$, we get

$$\sigma_u^2 = \left(\frac{1 + a_2}{1 - a_2}\right) \frac{\sigma_\nu^2}{[(1 + a_2)^2 - a_1^2]}. \tag{1.88}$$

For the three sets of AR parameters considered previously, we thus find that the variance of the white noise $\nu(n)$ has the values given in Table 1.1, assuming that $\sigma_u^2 = 1$.

TABLE 1.1 AR Parameters and Noise Variance

a_1	a_2	σ_ν^2
−0.10	−0.8	0.27
0.1	−0.8	0.27
−0.975	0.95	0.0731

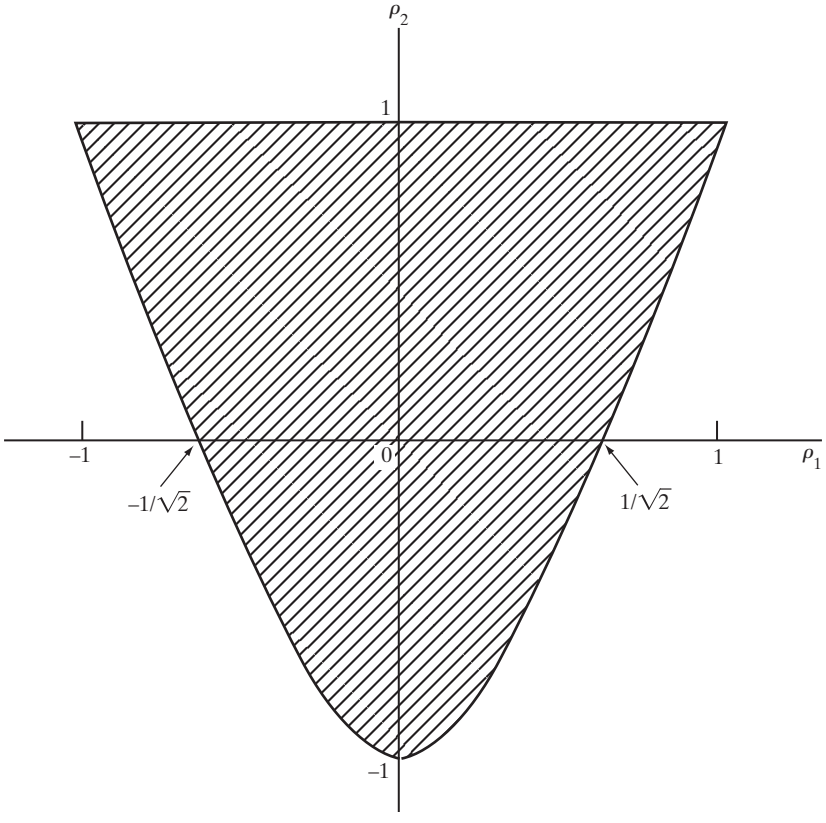


FIGURE 1.10 Permissible region for parameters of second-order AR process in terms of the normalized correlation coefficients ρ_1 and ρ_2.

1.10 SELECTING THE MODEL ORDER

The representation of a stochastic process by a linear model may be used for synthesis or analysis. In *synthesis*, we generate a desired time series by assigning a prescribed set of values to the parameters of the model and feeding it with white noise of zero mean and prescribed variance. In *analysis*, on the other hand, we *estimate* the parameters of the model by processing a given time series of finite length. Insofar as the estimation is statistical, we need an appropriate measure of the fit between the model and the observed data. This implies that, unless we have some prior information, the estimation procedure should include a criterion for selecting the *model order* (i.e., the degrees of freedom in the model). In the case of an AR process defined by Eq. (1.42), the model order equals M. In the case of an MA process defined by Eq. (1.52), the model order equals K. In the case of an ARMA process defined by Eq. (1.53), the model order equals (M, K). Various criteria for selecting the model order are described in the literature (Priestley, 1981; Kay, 1988). In this section, we describe two such important criteria, one of which was pioneered by Akaike (1973, 1974) and the other by Rissanen (1978) and

Schwartz (1978); both criteria result from the use of information-theoretic arguments, but in entirely different ways.

An Information-Theoretic Criterion

Let $u_i = u(i)$, $i = 1, 2, \ldots, N$, denote the data obtained by N independent observations of a stationary discrete-time stochastic process and $g(u_i)$ denote the probability density function of u_i. Let $f_U(u_i|\,\hat{\boldsymbol{\theta}}_m)$ denote the conditional probability density function of u_i, given $\hat{\boldsymbol{\theta}}_m$, the *estimated* vector of parameters that model the process. Let m be the model order, so that we may write

$$\hat{\boldsymbol{\theta}}_m = [\hat{\theta}_{1m}, \hat{\theta}_{2m}, \ldots, \hat{\theta}_{mm}]^{\mathrm{T}}. \tag{1.89}$$

We thus have several models that compete with each other to represent the process of interest. *An information-theoretic criterion* (AIC) proposed by Akaike selects the model for which the quantity

$$\mathrm{AIC}(m) = -2L(\hat{\boldsymbol{\theta}}_m) + 2m \tag{1.90}$$

is a minimum. The function

$$L(\hat{\boldsymbol{\theta}}_m) = \max \sum_{i=1}^{N} \ln f_U(u_i|\,\hat{\boldsymbol{\theta}}_m), \tag{1.91}$$

where ln denotes the natural logarithm. The criterion expressed in Eq. (1.91) is derived by minimizing the *Kullback–Leibler divergence*,[4] which is used to provide a measure of the divergence between the "unknown" true probability density function $g(u)$ and the conditional probability density function $f_U(u_i|\,\hat{\boldsymbol{\theta}}_m)$ given by the model in light of the observed data.

The function $L(\hat{\boldsymbol{\theta}}_m)$, constituting the first term on the right-hand side of Eq. (1.90), except for a scalar, is recognized as a *logarithm* of the *maximum-likelihood estimates* of the parameters in the model. (The method of maximum likelihood is briefly reviewed in Appendix D.) The second term, $2m$, represents a *model complexity penalty* that makes AIC(m) an estimate of the Kullback–Leibler divergence.

The first term in the equation tends to decrease rapidly with the model order m. On the other hand, the second term increases linearly with m. The result is that if we plot AIC(m) versus m, the graph will, in general, show a definite minimum value, and the *optimum order* of the model is determined by that value of m at which AIC(m) attains its minimum value, which, incidentally, is called MAIC (minimum AIC).

[4] In Akaike (1973, 1974, 1977) and in Ulrych and Ooe (1983), Eq. (1.90) is derived from the principle of minimizing the expectation

$$D_{g\|f}(\hat{\boldsymbol{\theta}}_m) = \int_{-\infty}^{\infty} g(u) \ln g(u) \mathrm{d}u - \int_{-\infty}^{\infty} g(u) \ln f_U(u|\,\hat{\boldsymbol{\theta}}_m) \mathrm{d}u.$$

We refer to $D_{g\|f}(\hat{\boldsymbol{\theta}}_m)$ as the *Kullback–Leibler* divergence for discrimination between the two probability density functions $g(u)$ and $f_U(u|\,\hat{\boldsymbol{\theta}}_m)$ (Kullback & Leibler, 1951). The idea is to minimize the information added to the time series by modeling it as an AR, MA, or ARMA process of finite order, since any information added is virtually false information in a real-world situation. Because $g(u)$ is fixed and unknown, the problem of maximizing the second term that makes up $D_{g\|f}(\hat{\boldsymbol{\theta}}_m)$ reduces to one.

Minimum Description Length Criterion

Rissanen (1978, 1989) used an entirely different approach to solve the statistical model identification problem. Specifically, he started with the notion that a model may be viewed as a device for describing the regular features of a set of observed data, with the objective being that of searching for a model that best captures the regular features or constraints that give the data their special structure. Recognizing that the presence of constraints reduces uncertainty about the data, we note that the objective may equally be that of *encoding* the data in the shortest or least redundant manner. The term "encoding," as used here, refers to an exact description of the observed data. Accordingly, the number of binary digits needed to encode both the observed data, when advantage is taken of the constraints offered by a model, and the model itself may be used as a criterion for *measuring the amount of the same constraints* and therefore the goodness of the model.

We may thus formally state Rissanen's *minimum description length (MDL) criterion*[5] as follows:

> Given a data set of interest and a family of competing statistical models, the best model is the one that provides the shortest description length for the data.

In mathematical terms, this model is defined by[6] (Rissanen, 1978, 1989; Wax, 1995)

$$\text{MDL}(m) = -L(\hat{\boldsymbol{\theta}}_m) + \frac{1}{2} m \ln N, \tag{1.92}$$

where m is the number of independently adjusted parameters in the model and N is the sample size (i.e., the number of observations). As with Akaike's information-theoretic criterion, $L(\hat{\boldsymbol{\theta}}_m)$ is the logarithm of the maximum-likelihood estimates of the model parameters. In comparing Eqs. (1.90) and (1.92), we see that the principal difference between the AIC and MDL criteria lies in the structure-dependent term.

According to Rissanen (1989), the MDL criterion offers the following attributes:

- The model permits the shortest encoding of the observed data and captures all of the *learnable* properties of the observed data in the best possible manner.
- The MDL criterion is a *consistent* model-order estimator in the sense that it converges to the true model order as the sample size increases.
- The model is optimal in the context of linear regression problems as well as ARMA models.

[5]The idea of a *minimum description length* of individual recursively definable objects may be traced to Kolmogorov (1968).

[6]Schwartz (1989) has derived a similar result, using a Bayesian approach. In particular, he considers the asymptotic behavior of Bayes estimators under a special class of priors. These priors put positive probability on the subspaces that correspond to the competing models. The decision is made by selecting the model that yields the maximum a posteriori probability.

It turns out that, in the large sample limit, the two approaches taken by Schwartz and Rissanen yield essentially the same result. However, Rissanen's approach is much more general, whereas Schwartz's approach is restricted to the case that the observations are independent and come from an exponential distribution.

Perhaps the most significant point to note is that, in nearly all of the applications involving the MDL criterion, no anomalous result or model with undesirable properties has been reported in the literature.

1.11 COMPLEX GAUSSIAN PROCESSES

Gaussian stochastic processes, or simply *Gaussian processes*, are frequently encountered in both theoretic and applied analysis. In this section, we present a summary of some important properties of Gaussian processes that are *complex valued*.[7]

Let $u(n)$ denote a complex Gaussian process consisting of N samples. For the first- and second-order statistics of this process, we assume the following:

1. A mean of zero, as shown by

$$\mu = \mathbb{E}[u(n)] = 0 \qquad \text{for } 1, 2, \ldots, N. \tag{1.93}$$

2. An autocorrelation function denoted by

$$r(k) = \mathbb{E}[u(n)u^*(n-k)], \quad k = 0, 1, \ldots, N-1. \tag{1.94}$$

The set of autocorrelation functions $\{r(k), k = 0, 1, \ldots, N-1\}$ defines the correlation matrix $\mathbf{R}$ of the Gaussian process $u(n)$.

The shorthand notation $\mathcal{N}(\mathbf{0}, \mathbf{R})$ is commonly used to refer to a Gaussian process with a mean vector of zero and correlation matrix $\mathbf{R}$.

Equations (1.93) and (1.94) imply wide-sense stationarity of the process. Knowledge of the mean μ and the autocorrelation function $r(k)$ for varying values of lag k is indeed sufficient for a complete characterization of the complex Gaussian process $u(n)$. In particular, the *joint probability density function* of N samples of the process so described is defined by (Kelly et al., 1960)

$$\mathbf{f}_U(\mathbf{u}) = \frac{1}{(2\pi)^N \det(\mathbf{\Lambda})} \exp\left(-\frac{1}{2}\mathbf{u}^{\mathrm{H}}\mathbf{\Lambda}^{-1}\mathbf{u}\right), \tag{1.95}$$

where

$$\mathbf{u} = [u(1), u(2), \ldots, u(N)]^{\mathrm{T}}$$

is an N-by-1 data vector and Λ is the N-by-N Hermitian-symmetric *moment matrix* of the process, defined in terms of the correlation matrix $\mathbf{R} = \{r(k)\}$ as

$$\begin{aligned} \mathbf{\Lambda} &= \tfrac{1}{2}\mathbb{E}[\mathbf{u}\mathbf{u}^{\mathrm{H}}] \\ &= \tfrac{1}{2}\mathbf{R}. \end{aligned} \tag{1.96}$$

Note that the joint probability density function $\mathrm{f}_U(\mathbf{u})$ is $2N$-dimensional, where the factor 2 accounts for the fact that each of the N samples of the process has a real and an

[7]For a detailed treatment of complex Gaussian processes, see the book by Miller (1974). Properties of complex Gaussian processes are also discussed in Kelly et al. (1960), Reed (1962), and McGee (1971).

imaginary part. Note also that the probability density function of a single sample of the process, which is a special case of Eq. (1.95), is given by

$$f_U(u) = \frac{1}{\pi\sigma^2}\exp\left(-\frac{|u|^2}{\sigma^2}\right), \tag{1.97}$$

where $|u|$ is the magnitude of the sample $u(n)$ and σ^2 is its variance.

Based on the representation described herein, we may now summarize some important properties of a *zero-mean complex Gaussian process* $u(n)$ *that is wide-sense stationary* as follows:

1. The process $u(n)$ is *stationary in the strict sense*.
2. The process $u(n)$ is *circularly complex*, in the sense that any two different samples $u(n)$ and $u(k)$ of the process satisfy the condition

$$\mathbb{E}[u(n)u(k)] = 0 \qquad \text{for } n \neq k.$$

   It is for this reason that the process $u(n)$ is often referred to as a *circularly complex Gaussian process*.
3. Let $u_n = u(n), n = 1, 2, \ldots, N$, denote samples picked from a zero-mean, complex Gaussian process. According to Reed (1962)

   (a) If $k \neq l$, then

$$\mathbb{E}[u_{s_1}^* u_{s_2}^* \ldots\ u_{s_k}^* u_{t_1} u_{t_2} \ldots\ u_{t_l}] = 0, \tag{1.98}$$

   where s_i and t_j are integers selected from the available set $\{1, 2, \ldots, N\}$.

   (b) If $k = l$, then

$$\mathbb{E}[u_{s_1}^* u_{s_2}^* \ldots\ u_{s_l}^* u_{t_1} u_{t_2} \ldots\ u_{t_l}] = \pi\mathbb{E}[u_{s_{\pi(1)}}^* u_{t_1}]\mathbb{E}[u_{s_{\pi(2)}}^* u_{t_2}] \ldots\ \mathbb{E}[u_{s_{\pi(l)}}^* u_{t_1}], \tag{1.99}$$

   where π is a permutation of the set of integers $\{1, 2, \ldots, l\}$ and $\pi(j)$ is the jth element of that permutation. For the set of integers $\{1, 2, \ldots, l\}$, we have a total of $l!$ possible permutations. This means that the right-hand side of Eq. (1.99) consists of the sum of $l!$ expectation product terms. Equation (1.99) is called the *Gaussian moment-factoring theorem*.

EXAMPLE 2

Consider first the odd case of $N = 3$, for which the complex Gaussian process $u(n)$ consists of the three samples u_1, u_2, and u_3. Applying Eq. (1.98) yields the null result:

$$\mathbb{E}[u_1^* u_2^* u_3] = 0. \tag{1.100}$$

Consider next the even case of $N = 4$, for which the complex Gaussian process $u(n)$ consists of the four samples u_1, u_2, u_3, and u_4. The use of the Gaussian moment-factoring theorem given in Eq. (1.99) yields the identity:

$$\mathbb{E}[u_1^* u_2^* u_3 u_4] = \mathbb{E}[u_1^* u_3]\mathbb{E}[u_2^* u_4] + \mathbb{E}[u_2^* u_3]\mathbb{E}[u_1^* u_4]. \tag{1.101}$$

(For other useful identities derived from the Gaussian moment-factoring theorem, see Problem 13.)

1.12 POWER SPECTRAL DENSITY

The autocorrelation function, defined in Section 1.1 and used thereafter, is a *time-domain* description of the second-order statistics of a stochastic process. The *frequency-domain* counterpart of this statistical parameter is the *power spectral density*, which is also referred to as the *power spectrum* or, simply, *spectrum*. Indeed, the power spectral density of a stochastic process is firmly established as the most useful description of the time series commonly encountered in engineering and physical sciences.

To proceed with the definition of the power spectral density, consider again a wide-sense stationary discrete-time stochastic process whose mean is zero and whose autocorrelation function is denoted by $r(l)$ for lag $l = 0, \pm 1, \pm 2, \ldots$. Let the infinitely long time series $u(n), n = 0, \pm 1, \pm 2, \ldots$, denote a single realization of the process. Initially, we focus attention on a *windowed* portion of this time series by writing

$$u_N(n) = \begin{cases} u(n), & n = 0, \pm 1, \ldots, \pm N. \\ 0, & |n| > N \end{cases} \tag{1.102}$$

Then we permit the length $2N+1$ to approach infinity. By definition, the *discrete-time Fourier transform* of the windowed time series $u_N(n)$ is given by

$$U_N(\omega) = \sum_{n=-N}^{N} u_N(n) e^{-j\omega n}, \tag{1.103}$$

where ω is the *angular frequency*, lying in the interval $(-\pi, \pi]$. In general, $U_N(\omega)$ is complex valued; specifically, its *complex conjugate* is given by

$$U_N^*(\omega) = \sum_{k=-N}^{N} u_N^*(k) e^{j\omega k}, \tag{1.104}$$

where the asterisk denotes complex conjugation. In Eq. (1.104) we have used the variable k to denote discrete time for reasons that will become apparent immediately. In particular, we multiply Eq. (1.103) by (1.104) to express the squared magnitude of $U_N(n)$ as follows:

$$|U_N(\omega)|^2 = \sum_{n=-N}^{N} \sum_{k=-N}^{N} u_N(n) u_N^*(k) e^{-j\omega(n-k)}. \tag{1.105}$$

Each realization $U_N(n)$ produces such a result. The *expected* result is obtained by taking the statistical expectation of both sides of Eq. (1.105) and interchanging the order of expectation and double summation:

$$\mathbb{E}[|U_N(\omega)|^2] = \sum_{n=-N}^{N} \sum_{k=-N}^{N} \mathbb{E}[u_N(n) u_N^*(k)] e^{-j\omega(n-k)}. \tag{1.106}$$

We are permitted to do this interchange as we are dealing with linear operations. We now recognize that, for the wide-sense stationary discrete-time stochastic process under discussion, the autocorrelation function of $u_N(n)$ for lag $n - k$ is

$$r_N(n - k) = \mathbb{E}[u_N(n) u_N^*(k)], \tag{1.107}$$

which may be rewritten as follows, in light of the defining Eq. (1.102):

$$r_N(n-k) = \begin{cases} \mathbb{E}[u(n)u^*(k)] = r(n-k) & \text{for } -N \le (n,k) \le N \\ 0 & \text{otherwise} \end{cases}. \tag{1.108}$$

Accordingly, Eq. (1.106) takes on the form

$$\mathbb{E}[|U_N(\omega)|^2] = \sum_{n=-N}^{N} \sum_{k=-N}^{N} r(n-k)e^{-j\omega(n-k)}. \tag{1.109}$$

Let $l = n - k$, and so rewrite Eq. (1.109) as

$$\frac{1}{N}\mathbb{E}[|U_N(\omega)|^2] = \sum_{l=-N}^{N}\left(1 - \frac{|l|}{N}\right)r(l)e^{-j\omega l}. \tag{1.110}$$

Equation (1.110) may be interpreted as the discrete-time Fourier transform of the product of two time functions: the autocorrelation function $r_N(l)$ for lag l, and a triangular window known as the *Bartlett window*. The latter function is defined by

$$w_B(l) = \begin{cases} 1 - \dfrac{|l|}{N}, & |l| \le N \\ 0, & |l| \ge N \end{cases}. \tag{1.111}$$

As N approaches infinity, $w_B(l)$ approaches unity for all l. Correspondingly, we may write

$$\lim_{N\to\infty} \frac{1}{N}\mathbb{E}[|U_N(\omega)|^2] = \sum_{l=-\infty}^{\infty} r(l)e^{-j\omega l}, \tag{1.112}$$

where $r(l)$ is the autocorrelation function of the original time series $u(n)$. To be precise, Eq. (1.112) holds under the condition

$$\lim_{N\to\infty} \frac{1}{2N+1} \sum_{l=-N+1}^{N-1} |l| r(l)e^{-j\omega l} = 0.$$

Equation (1.112) leads us to define the quantity

$$S(\omega) = \lim_{N\to\infty} \frac{1}{N}\mathbb{E}[|U_N(\omega)|^2], \tag{1.113}$$

where $|U_N(\omega)|^2/N$ is called the *periodogram* of the windowed time series $u_N(n)$. Note that the order of expectation and limiting operations indicated in Eq. (1.113) cannot be changed. Note also that the periodogram converges to $S(\omega)$ only in the mean value, *not* in the mean square or any other meaningful way.

When the limit in Eq. (1.113) exists, the quantity $S(\omega)$ has the following interpretation (Priestley, 1981):

$S(\omega)d\omega$ = average of the contribution to the total power from components of a wide-sense stationary stochastic process with angular frequencies located between w and $w + dw$; the average is taken over all possible realizations of the process. (1.114)

Accordingly, the quantity $S(\omega)$ is the "spectral density of expected power," which is abbreviated as the *power spectral density* of the process. Thus, equipped with the definition of power spectral density given in Eq. (1.113), we may now rewrite Eq. (1.112) as

$$S(\omega) = \sum_{l=-\infty}^{\infty} r(l)e^{-j\omega l}. \tag{1.115}$$

In sum, Eq. (1.113) gives a basic definition of the power spectral density of a wide-sense stationary stochastic process, and Eq. (1.115) defines the mathematical relationship between the autocorrelation function and the power spectral density of such a process.

1.13 PROPERTIES OF POWER SPECTRAL DENSITY

Property 1. *The autocorrelation function and power spectral density of a wide-sense stationary stochastic process form a Fourier transform pair.*

Consider a wide-sense stationary stochastic process represented by the time series $u(n)$, assumed to be of infinite length. Let $r(l)$ denote the autocorrelation function of such a process for lag l, and let $S(\omega)$ denote the power spectral density of the process. According to Property 1, these two quantities are related by the pair of relations

$$S(\omega) = \sum_{l=-\infty}^{\infty} r(l)e^{-j\omega l}, \qquad -\pi < \omega \leq \pi \tag{1.116}$$

and

$$r(l) = \frac{1}{2\pi}\int_{-\pi}^{\pi} S(\omega)e^{j\omega l}\,d\omega, \qquad l = 0, \pm 1, \pm 2, \ldots . \tag{1.117}$$

Equation (1.116) states that *the power spectral density is the discrete-time Fourier transform of the autocorrelation function*. On the other hand, Eq. (1.117) states that *the autocorrelation function is the inverse discrete-time Fourier transform of the power spectral density*. This fundamental pair of equations constitutes the *Einstein–Wiener–Khintchine relations*.

In a way, we already have a proof of this property. Specifically, Eq. (1.116) is merely a restatement of Eq. (1.115), previously established in Section 1.12. Equation (1.117) follows directly from this result by invoking the formula for the inverse discrete-time Fourier transform.

Property 2. *The frequency support of the power spectral density $S(\omega)$ is the Nyquist interval $-\pi < \omega \leq \pi$.*

Outside this interval, $S(\omega)$ is periodic, as shown by the relationship

$$S(\omega + 2k\pi) = S(\omega) \qquad \text{for integer } k. \tag{1.118}$$

Property 3. *The power spectral density of a stationary discrete-time stochastic process is real.*

To derive this property, we rewrite Eq. (1.116) as

$$S(\omega) = r(0) + \sum_{k=1}^{\infty} r(k)e^{-j\omega k} + \sum_{k=-\infty}^{-1} r(k)e^{-j\omega k}.$$

Replacing k with $-k$ in the third term on the right-hand side of this equation, and recognizing that $r(-k) = r^*(k)$, we get

$$\begin{aligned} S(\omega) &= r(0) + \sum_{k=1}^{\infty} [r(k)e^{-j\omega k} + r^*(k)e^{j\omega k}] \\ &= r(0) + 2\sum_{k=1}^{\infty} \mathrm{Re}[r(k)e^{-j\omega k}], \end{aligned} \tag{1.119}$$

where Re denotes the *real-part operator*. Equation (1.119) shows that the power spectral density $S(\omega)$ is a real-valued function of ω. It is because of this property that we have used the notation $S(\omega)$ rather than $S(e^{j\omega})$ for the power spectral density.

Property 4. *The power spectral density of a real-valued stationary discrete-time stochastic process is even (i.e., symmetric); if the process is complex valued, its power spectral density is not necessarily even.*

For a real-valued stochastic process, we find that $S(-\omega) = S(\omega)$, indicating that $S(\omega)$ is an even function of ω; that is, it is symmetric about the origin. If, however, the process is complex valued, then $r(-k) = r^*(k)$, in which case we find that $S(-\omega) \neq S(\omega)$, and $S(\omega)$ is *not* an even function of ω.

Property 5. *The mean-square value of a stationary discrete-time stochastic process equals, except for the scaling factor* $1/2\pi$, *the area under the power spectral density curve for* $-\pi < \omega \leq \pi$.

This property follows directly from Eq. (1.117), evaluated for $l = 0$. For this condition, we may thus write

$$r(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} S(\omega)\, d\omega. \tag{1.120}$$

Since $r(0)$ equals the mean-square value of the process, we see that Eq. (1.120) is a mathematical description of Property 5. The mean-square value of a process is equal to the *expected power* of the process developed across a load resistor of 1 Ω. On this basis, the terms "expected power" and "mean-square value" are used interchangeably in what follows.

Property 6. *The power spectral density of a stationary discrete-time stochastic process is nonnegative.*

That is,

$$S(\omega) \geq 0 \qquad \text{for all } \omega. \tag{1.121}$$

This property follows directly from the basic formula of Eq. (1.113), reproduced here for convenience of presentation:

$$S(\omega) = \lim_{N\to\infty} \frac{1}{N} \mathbb{E}[|U_N(\omega)|^2].$$

We first note that $|U_N(\omega)|^2$, representing the squared magnitude of the discrete-time Fourier transform of a windowed portion of the time series $u(n)$, is nonnegative for all ω. The expectation $\mathbb{E}[|U_N(\omega)|^2]$ is also nonnegative for all ω. Thus, using the basic definition of $S(\omega)$ in terms of $U_N(\omega)$, we see that the property described by Eq. (1.121) follows immediately.

1.14 TRANSMISSION OF A STATIONARY PROCESS THROUGH A LINEAR FILTER

Consider a discrete-time filter that is *linear*, *time invariant*, and *stable*. Let the filter be characterized by the *discrete transfer function* $H(z)$, defined as the *ratio of the z-transform of the filter output to the z-transform of the filter input*. Suppose that we feed the filter with a stationary discrete-time stochastic process with power spectral density $S(\omega)$, as in Fig. 1.11. Let $S_o(\omega)$ denote the power spectral density of the filter output. We may then write

$$S_o(\omega) = |H(e^{j\omega})|^2 S(\omega), \tag{1.122}$$

where $H(e^{j\omega})$ is the *frequency response* of the filter. The frequency response $H(e^{j\omega})$ equals the discrete transfer function $H(z)$ evaluated on the unit circle in the z-plane. The important feature of this result is that the value of the output spectral density at angular frequency ω depends purely on the squared *amplitude response* of the filter and the input power spectral density at the same angular frequency ω.

Equation (1.122) is a fundamental relation in stochastic process theory. To derive it, we may proceed as follows: Let $y(n)$ denote the filter output in Fig. 1.11 produced in response to $u(n)$ applied to the filter input. Assuming that $u(n)$ represents a wide-sense stationary discrete-time stochastic process, we find that $y(n)$ also represents a wide-sense stationary discrete-time stochastic process modified by the filtering operation. Thus, given the autocorrelation function of the filter input $u(n)$, written as

$$r_u(l) = \mathbb{E}[u(n)u^*(n - l)],$$

we may express the autocorrelation function of the filter output $y(n)$ in a corresponding way as

$$r_y(l) = \mathbb{E}[y(n)y^*(n - l)], \tag{1.123}$$

where $y(n)$ is related to $u(n)$ by the convolution sum

$$y(n) = \sum_{i=-\infty}^{\infty} h(i)u(n - i). \tag{1.124}$$

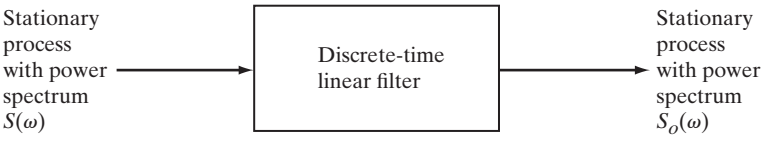


FIGURE 1.11 Transmission of stationary process through a discrete-time linear filter.

Similarly, we may write

$$y^*(n - l) = \sum_{k=-\infty}^{\infty} h^*(k)u^*(n - l - k). \tag{1.125}$$

Substituting Eqs. (1.124) and (1.125) into Eq. (1.123) and interchanging the orders of expectation and summation, we find that the autocorrelation functions $r_y(l)$ and $r_u(l)$, for lag l, are related by

$$r_y(l) = \sum_{i=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} h(i)h^*(k)r_u(k - i + l). \tag{1.126}$$

Finally, taking the discrete-time Fourier transforms of both sides of Eq. (1.126), and invoking Property 1 of the power spectral density and the fact that the transfer function of a linear filter is equal to the Fourier transform of its impulse response, we get the result described in Eq. (1.122).

Power Spectrum Analyzer

Suppose that the discrete-time linear filter in Fig. 1.11 is designed to have a bandpass characteristic. That is, the amplitude response of the filter is defined by

$$|H(e^{j\omega})| = \begin{cases} 1, & |\omega - \omega_c| \le \Delta\omega \\ 0, & \text{remainder of the interval } -\pi < \omega \le \pi. \end{cases} \tag{1.127}$$

This amplitude response is depicted in Fig. 1.12. We assume that the *angular bandwidth* of the filter, $2\Delta\omega$, is small enough for the spectrum inside this bandwidth to be essentially constant. Then, using Eq. (1.122), we may write

$$S_o(\omega) = \begin{cases} S(\omega_c), & |\omega - \omega_c| \le \Delta\omega \\ 0, & \text{remainder of the interval } -\pi < \omega \le \pi. \end{cases} \tag{1.128}$$

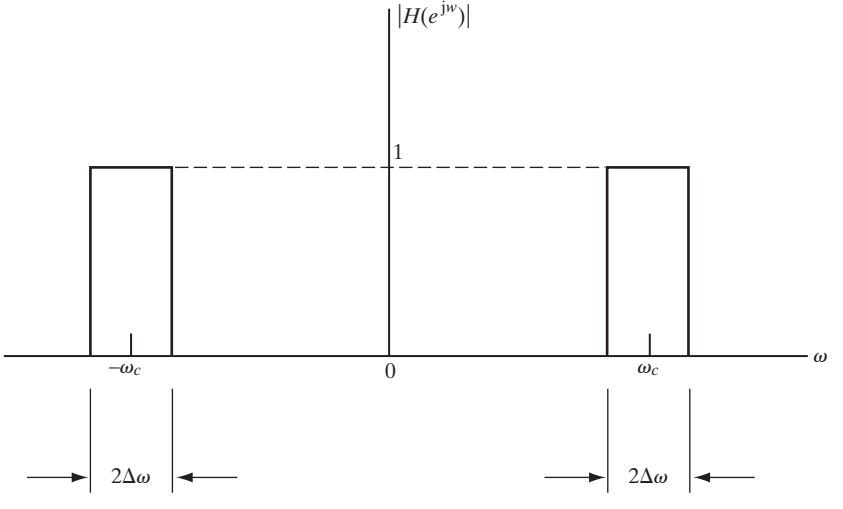


FIGURE 1.12 Ideal bandpass characteristic.

Next, using Properties 4 and 5 of the power spectral density, we may express the mean-square value of the filter output resulting from a real-valued stochastic input as

$$\begin{aligned} P_o &= \frac{1}{2\pi}\int_{-\pi}^{\pi} S_o(\omega)\, d\omega \\ &= \frac{2\Delta\omega}{2\pi} S(\omega_c) + \frac{2\Delta\omega}{2\pi} S(-\omega_c) \\ &= 2\frac{\Delta\omega}{\pi} S(\omega_c) \qquad \text{(for real data)}. \end{aligned}$$

Equivalently, we may write

$$S(\omega_c) = \frac{\pi P_o}{2\Delta\omega}, \tag{1.129}$$

where $\Delta\omega/\pi$ is that fraction of the Nyquist interval that corresponds to the passband of the filter. Equation (1.129) states that the value of the power spectral density of the filter input $u(n)$, measured at the center frequency ω_c, of the filter, is equal to the mean-square value P_o of the filter output, scaled by a constant factor. We may thus use Eq. (1.129) as the mathematical basis for building a *power spectrum analyzer*, as depicted in Fig. 1.13. Ideally, the discrete-time bandpass filter employed here should satisfy two requirements: It should have a *fixed bandwidth* and an *adjustable center frequency*. Clearly, in a practical filter design, we can only approximate these two ideal requirements. Also, note that the reading of the *average power meter* at the output end of the figure approximates (for finite averaging time) the expected power of an ergodic process $y(n)$.

FIGURE 1.13 Power spectrum analyzer.

EXAMPLE 3 White Noise

A stochastic process of zero mean is said to be *white* if its power spectral density $S(\omega)$ is constant for all frequencies, as shown by

$$S(\omega) = \sigma^2 \quad \text{for } -\pi < \omega \le \pi,$$

where σ^2 is the variance of a sample taken from the process. Suppose that this process is passed through a discrete-time bandpass filter characterized as in Fig. 1.12. Then, from Eq. (1.129), we find that the mean-square value of the filter output is

$$P_o = \frac{2\sigma^2\Delta\omega}{\pi}.$$

White noise has the property that any two of its samples are uncorrelated, as shown by the autocorrelation function

$$r(\tau) = \sigma^2 \delta_{\tau, 0},$$

where $\delta_{\tau, 0}$ is the Kronecker delta:

$$\delta_{\tau, 0} = \begin{cases} 1, & \tau = 0 \\ 0, & \text{otherwise} \end{cases}.$$

If the white noise is Gaussian, then any two samples of the process are *statistically independent*. In a sense, white Gaussian noise represents the ultimate in randomness.

1.15 CRAMÉR SPECTRAL REPRESENTATION FOR A STATIONARY PROCESS

Equation (1.113) provides one way of defining the power spectral density of a wide-sense stationary process. Another way of defining the power spectral density is to use the *Cramér spectral representation for a stationary process*. According to this representation, a discrete-time stochastic process $u(n)$ is described by the inverse Fourier transform (Thomson, 1982):

$$u(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega n} \, dZ(\omega). \tag{1.130}$$

If the process $u(n)$ is wide-sense stationary with no periodic components, then the *increment process* $dZ(\omega)$ has two basic properties:

1. The mean of $dZ(\omega)$ is zero; that is,

$$\mathbb{E}[dZ(\omega)] = 0 \qquad \text{for all } \omega. \tag{1.131}$$

2. The *generalized spectral density*, in terms of the increment process, is given by

$$\mathbb{E}[dZ(\omega)\, dZ^*(\nu)] = S(\omega)\, \delta(\omega - \nu) d\omega \, d\nu. \tag{1.132}$$

The $\delta(\omega)$ is the *delta function* in the ω-domain; for a continuous function $G(\omega)$, it satisfies the *sifting property*

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} G(\nu) \delta(\omega - \nu) \, d\nu = G(\omega). \tag{1.133}$$

In other words, for a wide-sense stationary discrete-time stochastic process $u(n)$, the increment process $dZ(\omega)$ defined by Eq. (1.130) is a *zero-mean orthogonal process*. More precisely, $dZ(\omega)$ may be viewed as a "white process" described in the frequency domain in a manner similar to the way ordinary white noise is described in the time domain.

Equation (1.133), in conjunction with Eq. (1.132), provides another basic definition for the power spectral density $S(\omega)$. Taking the complex conjugate of both sides of Eq. (1.130) and using ν in place of ω, we get

$$u^*(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-j\nu n} \, dZ^*(\nu). \tag{1.134}$$

Hence, multiplying Eq. (1.130) by Eq. (1.134), we may express the squared magnitude of $u(n)$ as

$$|u(n)|^2 = \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} e^{jn(\omega-\nu)}\, dZ(\omega)\, dZ^*(\nu). \tag{1.135}$$

Next, taking the statistical expectation of Eq. (1.135) and interchanging the order of expectation and double integration, we get

$$\mathbb{E}[|u(n)|]^2 = \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} e^{jn(\omega-\nu)} \mathbb{E}[dZ(\omega)\, dZ^*(\nu)]. \tag{1.136}$$

If we now use the basic property of the increment process $dZ(\omega)$ described by Eqs. (1.132) and (1.133), we may simplify Eq. (1.136) to the form

$$\mathbb{E}[|u(n)|^2] = \frac{1}{2\pi} \int_{-\pi}^{\pi} S(\omega)\, d\omega. \tag{1.137}$$

The expectation $\mathbb{E}[|u(n)|^2]$ on the left-hand side of Eq. (1.137) is recognized as the mean-square value of $u(n)$. The right-hand side of this equation equals the total area under the curve of the power spectral density $S(\omega)$, scaled by the factor $1/2\pi$. Accordingly, Eq. (1.137) is merely a restatement of Property 5 of the power spectral density $S(\omega)$, described by Eq. (1.120).

The Fundamental Equation

Using the dummy variable ν in place of ω in the Cramér spectral representation given in Eq. (1.130) and then substituting the result into Eq. (1.103), we get

$$U_N(\omega) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \sum_{n=-N}^{N} (e^{-j(\omega-\nu)n})\, dZ(\nu), \tag{1.138}$$

where we have interchanged the order of summation and integration. Next, we define

$$K_N(\omega) = \sum_{n=-N}^{N} e^{-j\omega n}, \tag{1.139}$$

which is known as the *Dirichlet kernel*. The kernel $K_N(\omega)$ represents a geometric series with a first term equal to $e^{j\omega N}$, a common ratio of $e^{-j\omega}$, and a total number of terms equal to $2N + 1$. Summing this series, we may redefine the kernel as

$$\begin{aligned} K_N(\omega) &= \frac{e^{j\omega N}(1 - e^{-j\omega(2N+1)})}{1 - e^{-j\omega}} \\ &= \frac{\sin((2N + 1)\omega/2)}{\sin(\omega/2)}. \end{aligned} \tag{1.140}$$

Note that $K_N(0) = 2N + 1$. Returning to Eq. (1.138), we may use the definition of the Dirichlet kernel $K_N(\omega)$ given in Eq. (1.139) to write

$$U_N(\omega) = \frac{1}{2\pi} \int_{-\pi}^{\pi} K_N(\omega - \nu)\, dZ(\nu). \tag{1.141}$$

The integral Eq. (1.141) is a *linear* relation referred to as the *fundamental equation* of power spectrum analysis.

An integral equation involves an *unknown* function under the integral sign. In the context of power spectrum analysis as described by Eq. (1.141), the increment variable $dZ(\omega)$ is the unknown function, and $U_N(\omega)$ is known. Accordingly, that equation may be viewed as an example of a *Fredholm integral equation of the first kind* (Morse & Feshbach, 1953; Whittaker & Watson, 1965).

Note that $U_N(\omega)$ may be inverse Fourier transformed to recover the original data. It follows, therefore, that $U_N(\omega)$ is a *sufficient statistic* for determining the power spectral density. This property makes the use of Eq. (1.141) for spectrum analysis all the more important.

1.16 POWER SPECTRUM ESTIMATION

An issue of practical importance is how to *estimate* the power spectral density of a wide-sense stationary process. Unfortunately, this issue is complicated by the fact that there is a bewildering array of power spectrum estimation procedures, with each procedure purported to have or to show some optimum property. The situation is made worse by the fact that, unless care is taken in the selection of the right method, we may end up with misleading conclusions.

Two philosophically different families of power spectrum estimation methods can be identified in the literature: *parametric methods* and *nonparametric methods*. The basic ideas behind these methods are discussed next.

Parametric Methods

In parametric methods of spectrum estimation, we begin by postulating a *stochastic model* for the situation at hand. Depending on the specific model adopted, we may identify three different parametric approaches for spectrum estimation:

1. *Model-identification procedures.* In this class of parametric methods, a rational function or a polynomial in $e^{-j\omega}$ is assumed for the transfer function of the model, and a white-noise source is used to drive the model, as depicted in Fig. 1.14. The power spectrum of the resulting model output provides the desired spectrum estimate. Depending on the application of interest, we may adopt one of the following models (Kay & Marple, 1981; Marple, 1987; Kay, 1988):
   - **(i)** An *autoregressive (AR) model* with an all-pole transfer function.
   - **(ii)** A *moving-average (MA) model* with an all-zero transfer function.

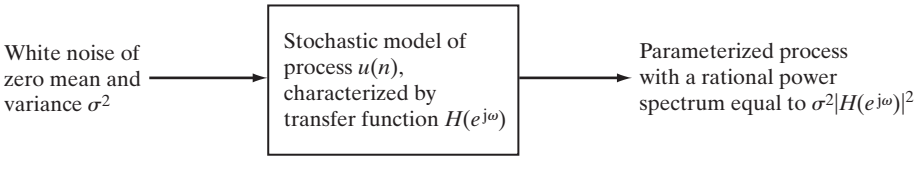


FIGURE 1.14 Rationale for model-identification procedure for power spectrum estimation.

(iii) An *autoregressive–moving-average (ARMA) model* with a pole–zero transfer function.

The resulting power spectra measured at the outputs of these models are referred to as AR, MA, and ARMA spectra, respectively. With reference to the input–output relation of Eq. (1.122), set the power spectrum $S(\omega)$ of the model input equal to the white noise variance σ^2. We then find that the power spectrum $S_o(\omega)$ of the model output is equal to the squared amplitude response $|H(e^{j\omega})|^2$ of the model, multiplied by σ^2. The problem thus becomes one of estimating the model parameters [i.e., parameterizing the transfer function $H(e^{j\omega})$] such that the process produced at the model output provides an acceptable representation (in some statistical sense) of the stochastic process under study. Such an approach to power spectrum estimation may indeed be viewed as a problem in *model (system) identification*.

Among the model-dependent spectra defined herein, the AR spectrum is by far the most popular. The reason for this popularity is twofold: (1) the *linear* form of the system of simultaneous equations involving the unknown AR model parameters and (2) the availability of efficient algorithms for computing the solution.

2. *Minimum-variance distortionless response method.* To describe this second parametric approach for power spectrum estimation, consider the situation depicted in Fig. 1.15. The process $u(n)$ is applied to a finite-duration impulse response (FIR) filter (i.e., a discrete-time filter with an all-zero transfer function). In the *minimum-variance distortionless response (MVDR) method*, the filter coefficients are chosen so as to minimize the variance (which is the same as the expected power for a zero-mean process) of the filter output, subject to the constraint that the frequency response of the filter is equal to unity at some angular frequency ω_0. Under this constraint, the process $u(n)$ is passed through the filter with *no distortion* at the angular frequency ω_0. Moreover, signals at angular frequencies other than ω_0 tend to be attenuated.

3. *Eigendecomposition-based methods.* In this final class of parametric spectrum estimation methods, the eigendecomposition of the ensemble-average correlation matrix $\mathbf{R}$ of the process $u(n)$ is used to define two disjoint subspaces: *signal subspace* and *noise subspace*. This form of partitioning is then exploited to derive an appropriate algorithm for estimating the power spectrum (Schmidt, 1979, 1981). (Eigenanalysis and the notion of subspace decomposition are discussed in Appendix E.)

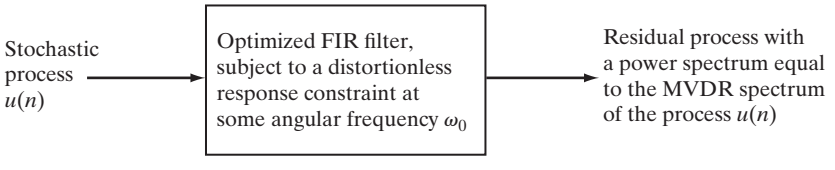


FIGURE 1.15 Diagram illustrating the MVDR procedure for power spectrum estimation.

Nonparametric Methods

In nonparametric methods of power spectrum estimation, no assumptions are made with respect to the stochastic process under study. The starting point in the discussion is the fundamental Eq. (1.141). Depending on the way in which this equation is interpreted, we may distinguish two different nonparametric approaches:

1. *Periodogram-based methods.* Traditionally, the fundamental Eq. (1.141) is treated as a *convolution* of two frequency functions. One, $U(\omega)$, represents the discrete-time Fourier transform of an *infinitely long* time series, $\{u(n)\}$; this function arises from the definition of the increment variable $\mathrm{d}Z(\omega)$ as the product of $U(\omega)$ and the frequency increment $\mathrm{d}\omega$. The other frequency function is the kernel $K_N(\omega)$, defined by Eq. (1.140). This approach leads us to consider Eq. (1.113) as the basic definition of the power spectral density $S(\omega)$ and therefore the *periodogram* $|U_N(\omega)|^2/N$ as the starting point for the data analysis. However, the periodogram suffers from a serious limitation in the sense that *it is not a sufficient statistic for the power spectral density*. This implies that the phase information ignored in the use of the periodogram is essential. Consequently, the statistical insufficiency of the periodogram is inherited by any estimate that is based on or equivalent to the periodogram.
2. *Multiple-window method* also called the *multitaper method.* A more constructive nonparametric approach is to treat the fundamental Eq. (1.141) as a *Fredholm integral equation of the first kind* for the increment variable $\mathrm{d}Z(\omega)$; the goal here is to obtain an *approximate solution* of the equation with statistical properties that are close to those of $\mathrm{d}Z(\omega)$ in some sense (Thomson, 1982). The key to the attainment of this important goal is the use of windows defined by a set of special sequences known as *Slepian sequences*,[8] or *discrete prolate spheroidal sequences*, which are fundamental to the study of time- and frequency-limited systems. The remarkable property of this family of windows is that the energy distributions of the windows add up in a very special way that collectively defines an ideal (in the sense of the total in-bin versus out-of-bin energy concentration) rectangular frequency bin. This property, in turn, allows us to trade spectral resolution for improved spectral properties (e.g., reduced variance of the spectral estimate).

In general, a discrete-time stochastic process $u(n)$ has a *mixed spectrum* in that its power spectrum contains two components: a deterministic one and a continuous one. The *deterministic component* represents the *first moment* of the increment process $\mathrm{d}Z(\omega)$ and is explicitly given by

$$\mathbb{E}[\mathrm{d}Z(\omega)] = \sum_k a_k \delta(\omega - \omega_k)\, \mathrm{d}\omega, \tag{1.142}$$

[8]Detailed information on Slepian sequences is given in Slepian (1978). A method for computing such sequences, for large data length, is given in the appendix of the paper by Thomson (1982). [For additional information, see the references listed in Thomson's paper; Mullis and Scharf (1991) also present an informative discussion of the role of Slepian sequences in spectrum analysis.]

where $\delta(\omega)$ is the *Dirac delta function* defined in the frequency domain. The ω_k are the angular frequencies of *periodic* or *line components* contained in the process $u(n)$, and the a_k are their amplitudes. The continuous component, on the other hand, represents the *second central moment* of the increment process, namely,

$$\mathbb{E}[|\mathrm{d}Z(\omega) - \mathbb{E}[\mathrm{d}Z(\omega)]|^2]. \tag{1.143}$$

It is important that the distinction between the first and second moments be carefully noted.

Spectra computed using the parametric methods tend to have sharper peaks and higher resolution than those obtained from the nonparametric (classical) methods. The application of these parametric methods is therefore well suited for estimating the deterministic component and, in particular, for locating the frequencies of periodic components in additive white noise when the signal-to-noise ratio is high. Another well-proven technique for estimating the deterministic component is the classical method of maximum likelihood. (As mentioned previously, Appendix D briefly reviews maximum-likelihood estimation.) Of course, if the physical laws governing the generation of a process match a stochastic model (e.g., the AR model) in an exact manner or approximately in some statistical sense, then the parametric method corresponding to that model may be used to estimate the power spectrum of the process. If, however, the stochastic process of interest has a purely continuous power spectrum and the underlying physical mechanism responsible for the generation of the process is unknown, then the recommended procedure is the nonparametric method of multiple windows.

In this book, we confine our attention to classes 1 and 2 of parametric methods of spectrum estimation, as their theory fits naturally under the umbrella of adaptive filters.[9]

1.17 OTHER STATISTICAL CHARACTERISTICS OF A STOCHASTIC PROCESS

In the material presented up to this point in the discussion, we have focused our attention on a partial characterization of a discrete-time stochastic process. According to this particular characterization, we only need to specify the mean as the first moment of the process and its autocorrelation function as the second moment. Since the autocorrelation function and power spectral density form a Fourier-transform pair, we may equally well specify the power spectral density in place of the autocorrelation function. The use of second-order statistics as described in Sections 1.1 through 1.14 is adequate for the study of linear adaptive filters operating under the supervision of a teacher. However, when we move on later in the book to consider difficult applications (e.g., blind deconvolution), we have to resort to the use of other statistical properties of a stochastic process.

[9]For a comprehensive discussion of the other methods of spectrum analysis, see Gardner (1987), Marple (1987), Kay (1988), Thomson (1982), and Mullis and Scharf (1991).

Two particular statistical properties that bring in additional information about a stochastic process, which can prove useful in the study of blind deconvolution, are as follows:

1. *High-order statistics.* An obvious way of expanding the characterization of a stationary stochastic process is to include *high-order statistics* (HOS) of the process. This is done by invoking the use of *cumulants* and their Fourier transforms, known as *polyspectra*. Indeed, cumulants and polyspectra of a zero-mean stochastic process may be viewed as generalizations of the autocorrelation function and power spectral density, respectively. It is important to note that HOS are meaningful only in the context of *non-Gaussian processes*. Furthermore, to exploit them, we need to use some form of nonlinear filtering.
2. *Cyclostationarity.* In an important class of stochastic processes commonly encountered in practice, the mean and autocorrelation function of the process exhibit periodicity, as in

$$\mu(t_1 + T) = \mu(t_1) \tag{1.144}$$

and

$$r(t_1 + T, t_2 + T) = r(t_1, t_2) \tag{1.145}$$

for all t_1 and t_2. Both t_1 and t_2 represent values of the continuous-time variable t, and T denotes the period. A stochastic process satisfying Eqs. (1.144) and (1.145) is said to be *cyclostationary* in the wide sense (Franks, 1969; Gardner & Franks, 1975; Gardner, 1994a, b). Modeling a stochastic process as cyclostationary adds a new dimension, namely, the period T, to the partial description of the process. Examples of cyclostationary processes include a modulated process obtained by varying the amplitude, phase, or frequency of a sinusoidal carrier.

In Sections 1.18 and 1.19, we discuss these two specific aspects of stochastic processes as they pertain to polyspectra and spectral-correlation density. As already mentioned, polyspectra provide a frequency-domain description of the HOS of a stationary stochastic process. By the same token, spectral-correlation density provides a frequency-domain description of a cyclostationary stochastic process.

1.18 POLYSPECTRA

Consider a stationary stochastic process with zero mean. Let $u(n)$, $u(n + \tau_1), \cdots, u(n + \tau_{k-1})$ denote the random variables obtained by observing this stochastic process at times $n, n + \tau_1, \ldots, n + \tau_{k-1}$, respectively. These random variables form the k-by-1 vector

$$\mathbf{u} = [u(n), u(n + \tau_1), \ \ldots \ , u(n + \tau_{k-1})]^{\mathrm{T}}.$$

Correspondingly, define a k-by-1 vector

$$\mathbf{z} = [z_1, z_2, \ \ldots \ , z_k]^{\mathrm{T}}.$$

We may then define the kth-*order cumulant* of the stochastic process $u(n)$, denoted by $c_k(\tau_1, \tau_2, \ldots, \tau_{k-1})$, as the coefficient of the vector $\mathbf{z}$ in the Taylor expansion of the *cumulant-generating function* (Priestley, 1981; Swami & Mendel, 1990; Gardner, 1994a, b):

$$K(\mathbf{z}) = \ln \mathbb{E}[\exp(\mathbf{z}^{\mathrm{T}}\mathbf{u})]. \tag{1.146}$$

The kth-order cumulant of the process $u(n)$ is thus defined in terms of its joint moments of orders up to k; to simplify the presentation in this section, we assume that $u(n)$ is real valued. Specifically, the second-, third-, and fourth-order cumulants are given, respectively, by

$$c_2(\tau) = \mathbb{E}[u(n)u(n+\tau)], \tag{1.147}$$

$$c_3(\tau_1, \tau_2) = \mathbb{E}[u(n)u(n+\tau_1)u(n+\tau_2)], \tag{1.148}$$

and

$$\begin{aligned} c_4(\tau_1, \tau_2, \tau_3) = {} & \mathbb{E}[u(n)u(n+\tau_1)u(n+\tau_2)u(n+\tau_3)] \\ & - \mathbb{E}[u(n)u(n+\tau_1)]\mathbb{E}[u(n+\tau_2)u(n+\tau_3)] \\ & - \mathbb{E}[u(n)u(n+\tau_2)]\mathbb{E}[u(n+\tau_3)u(n+\tau_1)] \\ & - \mathbb{E}[u(n)u(n+\tau_3)]\mathbb{E}[u(n+\tau_1)u(n+\tau_2)]. \end{aligned} \tag{1.149}$$

From the definitions given in Eqs. (1.147) through (1.149), we note the following:

1. The second-order cumulant $c_2(\tau)$ is the same as the autocorrelation function $r(\tau)$.
2. The third-order cumulant $c_3(\tau_1, \tau_2)$ is the same as the third-order moment $\mathbb{E}[u(n)u(n+\tau_1)u(n+\tau_2)]$.
3. The fourth-order cumulant $c_4(\tau_1, \tau_2, \tau_3)$ is *different* from the fourth-order moment $\mathbb{E}[u(n)u(n+\tau_1)u(n+\tau_2)u(n+\tau_3)]$. In order to generate the fourth-order cumulant, we need to know the fourth-order moment and six different values of the autocorrelation function.

Note that the kth-order cumulant $c(\tau_1, \tau_2, \ldots, \tau_{k-1})$ does not depend on time n. For this to be valid, however, the process $u(n)$ has to be stationary up to order k. A process $u(n)$ is said to be *stationary up to order k* if, for any admissible set of time instants $\{n_1, n_2, \ldots, n_p\}$, all the joint moments up to order k of $\{u(n_1), u(n_2), \ldots, u(n_p)\}$ exist and equal the corresponding joint moments up to order k of $\{u(n_1+\tau), u(n_2+\tau), \ldots, u(n_p+\tau)\}$, where $\{n_1+\tau, n_2+\tau, \ldots, n_p+\tau\}$ is an admissible set, too (Priestley, 1981).

Consider next a linear time-invariant system characterized by the impulse response h_n. Let the system be excited by a process $x(n)$ consisting of independent and identically distributed (i.i.d.) samples. Let $u(n)$ denote the resulting system output. The kth-order cumulant of $u(n)$ is given by

$$c_k(\tau_1, \tau_2, \ldots, \tau_{k-1}) = \gamma_k \sum_{i=-\infty}^{\infty} h_i h_{i+\tau_1} \cdots h_{i+\tau_{k-1}}, \tag{1.150}$$

where γ_k is the kth-order cumulant of the input process $x(n)$. Note that the summation term on the right-hand side of Eq. (1.150) has a form similar to that of a kth-order moment, except that the expectation operator has been replaced by a summation.

The kth-*order polyspectrum* (or kth-*order cumulant spectrum*) is defined by (Priestley, 1981; Nikias & Raghuveer, 1987)

$$C_k(\omega_1, \omega_2, \ldots, \omega_{k-1}) = \sum_{\tau_1=-\infty}^{\infty} \cdots \sum_{\tau_{k-1}=-\infty}^{\infty} c_k(\tau_1, \tau_2, \ldots, \tau_{k-1}) \times \exp[-\mathrm{j}(\omega_1\tau_1 + \omega_2\tau_2 + \ldots + \omega_{k-1}\tau_{k-1})]. \tag{1.151}$$

A sufficient condition for the existence of the polyspectrum $C_k(\omega_1, \omega_2, \ldots, \omega_{k-1})$ is that the associated kth-order cumulant $c_k(\tau_1, \tau_2, \ldots, \tau_{k-1})$ be absolutely summable, as shown by

$$\sum_{\tau_1=-\infty}^{\infty} \cdots \sum_{\tau_{k-1}=-\infty}^{\infty} |c_k(\tau_1, \tau_2, \ldots, \tau_{k-1})| < \infty. \tag{1.152}$$

The *power spectrum*, *bispectrum*, and *trispectrum* are special cases of the kth-order polyspectrum defined in Eq. (1.151). Specifically, we may state the following:

1. For $k = 2$, we have the ordinary power spectrum,

$$C_2(\omega_1) = \sum_{\tau_1=-\infty}^{\infty} c_2(\tau_1) \exp(-\mathrm{j}\omega_1\tau_1), \tag{1.153}$$

which is a restatement of the Einstein–Wiener–Khintchine relation given in Eq. (1.116).

2. For $k = 3$, we have the *bispectrum*, defined by

$$C_3(\omega_1, \omega_2) = \sum_{\tau_1=-\infty}^{\infty} \sum_{\tau_2=-\infty}^{\infty} c_3(\tau_1, \tau_2) \exp[-\mathrm{j}(\omega_1\tau_1 + \omega_2\tau_2)]. \tag{1.154}$$

3. For $k = 4$, we have the *trispectrum*, defined by

$$C_4(\omega_1, \omega_2, \omega_3) = \sum_{\tau_1=-\infty}^{\infty} \sum_{\tau_2=-\infty}^{\infty} \sum_{\tau_3=-\infty}^{\infty} c_4(\tau_1, \tau_2, \tau_3) \exp[-\mathrm{j}(\omega_1\tau_1 + \omega_2\tau_2 + \omega_3\tau_3)]. \tag{1.155}$$

An outstanding property of the polyspectrum is that all polyspectra of order higher than the second vanish when the process $u(n)$ is Gaussian. This property is a direct consequence of the fact that all joint cumulants of order higher than the second are zero for multivariate Gaussian distributions. Accordingly, the bispectrum, trispectrum, and all higher-order polyspectra are identically zero if the process $u(n)$ is Gaussian. Thus, higher-order spectra provide measures of the *departure of a stochastic process from Gaussianity*.

The kth-order cumulant $c_k(\tau_1, \tau_2, \ldots, \tau_{k-1})$ and the kth-order polyspectrum $C_k(\omega_1, \omega_2, \ldots, \omega_{k-1})$ form a pair of multidimensional Fourier transforms. Specifically, the polyspectrum $C_k(\omega_1, \omega_2, \ldots, \omega_{k-1})$ is the *multidimensional discrete-time Fourier transform* of $c_k(\tau_1, \tau_2, \ldots, \tau_{k-1})$, and $c_k(\tau_1, \tau_2, \ldots, \tau_{k-1})$ is the *inverse multidimensional discrete-time Fourier transform* of $C_k(\omega_1, \omega_2, \ldots, \omega_{k-1})$. For example, given the bispectrum $C_3(\omega_1, \omega_2)$, we may determine the third-order cumulant $c_3(\tau_1, \tau_2)$ by using the inverse two-dimensional discrete-time Fourier transform:

$$c_3(\tau_1, \tau_2) = \left(\frac{1}{2\pi}\right)^2 \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} C_3(\omega_1, \omega_2) \exp[\mathrm{j}(\omega_1\tau_1 + \omega_2\tau_2)] \, \mathrm{d}\omega_1 \, \mathrm{d}\omega_2. \tag{1.156}$$

We may use this relation to develop an alternative definition of the bispectrum as follows: According to the *Cramér spectral representation*, we have

$$u(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{j\omega n}\, dZ(\omega) \qquad \text{for all } n. \tag{1.157}$$

Hence, using Eq. (1.157) in Eq. (1.148), we get

$$\begin{aligned} c_3(\tau_1, \tau_2) = \left(\frac{1}{2\pi}\right)^3 \int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} &\exp\left[jn(\omega_1 + \omega_2 + \omega_3)\right] \\ &\times \exp\left[j(\omega_1\tau_1 + \omega_2\tau_2)\right]\mathbb{E}[dZ(\omega_1)dZ(\omega_2)dZ(\omega_3)]. \end{aligned} \tag{1.158}$$

Comparing the right-hand sides of Eqs. (1.156) and (1.158), we deduce the following result:

$$\mathbb{E}[dZ(\omega_1)\, dZ(\omega_2)\, dZ(\omega_3)] = \begin{cases} C_3(\omega_1, \omega_2)\, d\omega_1\, d\omega_2, & \omega_1 + \omega_2 + \omega_3 = 0 \\ 0, & \text{otherwise} \end{cases}. \tag{1.159}$$

It is apparent from Eq. (1.159) that the bispectrum $C_3(\omega_1, \omega_2)$ represents the contribution to the mean product of three Fourier components whose *individual frequencies add up to zero*. This property is an extension of the interpretation developed for the ordinary power spectrum in Section 1.14. In a similar manner, we may develop an interpretation of the trispectrum.

In general, the polyspectrum $C_k(\omega_1, \omega_2, \ldots, \omega_{k-1})$ is *complex for order k higher than two*, as shown by

$$C_k(\omega_1, \omega_2, \ldots, \omega_{k-1}) = |C_k(\omega_1, \omega_2, \ldots, \omega_{k-1})| \exp[j\phi_k(\omega_1, \omega_2, \ldots, \omega_{k-1})], \tag{1.160}$$

where we note that $|C_k(\omega_1, \omega_2, \ldots, \omega_{k-1})|$ is the *magnitude* of the polyspectrum and $\phi_k(\omega_1, \omega_2, \ldots, \omega_{k-1})$ is the *phase*. Moreover, the polyspectrum is a *periodic* function with period 2π; that is,

$$C_k(\omega_1, \omega_2, \ldots, \omega_{k-1}) = C_k(\omega_1 + 2\pi, \omega_2 + 2\pi, \ldots, \omega_{k-1} + 2\pi). \tag{1.161}$$

Whereas the power spectral density of a stationary stochastic process is *phase blind*, the polyspectra of the process are *phase sensitive*. More specifically, the power spectral density is real valued: Referring to the input–output relation of Eq. (1.122), we clearly see that, in passing a stationary stochastic process through a linear system, information about the phase response of the system is completely destroyed in the power spectrum of the output. In contrast, the polyspectrum is complex valued, with the result that in a similar situation the polyspectrum of the output signal preserves information about the phase response of the system. It is for this reason that polyspectra provide a useful tool for the "blind" identification of an unknown system, where we have access only to the output signal and some additional information in the form of a probabilistic model of the input signal.

1.19 SPECTRAL-CORRELATION DENSITY

Polyspectra preserve phase information about a stochastic process by invoking higher-order statistics of the process, which is feasible only if the process is non-Gaussian. The preservation of phase information is also possible if the process is *cyclostationary* in the

wide sense, as defined in Eqs. (1.144) and (1.145). The latter approach has two important advantages over the higher-order statistics approach:

1. The phase information is contained in second-order cyclostationary statistics of the process; hence, the phase information can be exploited in a computationally efficient manner that avoids the use of higher-order statistics.
2. Preservation of the phase information holds, irrespective of Gaussianity.

Consider, then, a discrete-time stochastic process $u(n)$ that is cyclostationary in the wide sense. Without loss of generality, the process is assumed to have zero mean. The ensemble-average autocorrelation function of $u(n)$ is defined in the usual way by Eq. (1.2):

$$r(n, n - k) = \mathbb{E}[u(n)u^*(n - k)].$$

Under the condition of cyclostationarity, the autocorrelation function $r(n, n-k)$ is periodic in n for every k. Keeping in mind the discrete-time nature of $u(n)$, we may expand the autocorrelation function $r(n, n-k)$ into the complex Fourier series (Gardner, 1994a, b)

$$r(n, n - k) = \sum_{\{\alpha\}} r^{\alpha}(k) e^{\mathrm{j}2\pi\alpha n - \mathrm{j}\pi\alpha k}, \tag{1.162}$$

where both n and k take on only integer values and the set $\{\alpha\}$ includes all values of α for which the corresponding Fourier coefficient $r^{\alpha}(k)$ is not zero. The Fourier coefficient $r^{\alpha}(k)$ is itself defined by

$$r^{\alpha}(k) = \frac{1}{N} \sum_{n=0}^{N-1} r(n, n - k) e^{-\mathrm{j}2\pi\alpha n + \mathrm{j}\pi\alpha k}, \tag{1.163}$$

where the number of samples, N, denotes the period. Equivalently, in light of Eq. (1.6), we may define

$$r^{\alpha}(k) = \frac{1}{N} \left\{ \sum_{n=0}^{N-1} \mathbb{E}[u(n)u^*(n - k)e^{-\mathrm{j}2\pi\alpha n}] \right\} e^{\mathrm{j}\pi\alpha k}. \tag{1.164}$$

The quantity $r^{\alpha}(k)$ is called the *cyclic autocorrelation function*, which has the following properties:

1. The cyclic autocorrelation function $r^{\alpha}(k)$ is periodic in α with period two.
2. For any α, we have, from Eq. (1.164),

$$r^{\alpha+1}(k) = (-1)^k r^{\alpha}(k). \tag{1.165}$$

3. For the special case of $\alpha = 0$, Eq. (1.164) reduces to

$$r^{0}(k) = r(k), \tag{1.166}$$

where $r(k)$ is the ordinary autocorrelation function of a stationary process.

According to the Einstein–Wiener–Khintchine relations given in Eqs. (1.116) and (1.117), the ordinary versions of the autocorrelation function and power spectral density of a wide-sense stationary stochastic process form a Fourier-transform pair. In a

corresponding way, we may define the discrete-time Fourier transform of the cyclic autocorrelation function $r^{\alpha}(k)$ as follows (Gardner, 1994a, b):

$$S^{\alpha}(\omega) = \sum_{k=-\infty}^{\infty} r^{\alpha}(k)e^{-j\omega k}, \qquad -\pi < \omega \leq \pi. \tag{1.167}$$

The new quantity $S^{\alpha}(\omega)$ is called the *spectral-correlation density*, which is complex valued for $\alpha \neq 0$. Note that, for the special case of $\alpha = 0$, Eq. (1.167) reduces to

$$S^{0}(\omega) = S(\omega),$$

where $S(\omega)$ is the ordinary power spectral density.

In light of the defining Eqs. (1.164) and (1.167), we may set up the block diagram of Fig. 1.16 for measuring the spectral-correlation density $S^{\alpha}(\omega)$. For this measurement, it is assumed that the process $u(n)$ is *cycloergodic* (Gardner, 1994a, b), which means that time averages may be substituted for ensemble averages, with samples taken once per period. According to the instrumentation described in Fig. 1.16, $S^{\alpha}(\omega)$ is the bandwidth-normalized version of the cross-correlation narrowband spectral components contained in the time series $u(n)$ at the angular frequencies $\omega + \alpha\pi$ and $\omega - \alpha\pi$, in the limit as the bandwidth of these spectral components is permitted to approach zero (Gardner, 1994a, b). Note that the two narrowband filters in the figure are identical, both having a midband (angular) frequency ω and a bandwidth $\Delta\omega$ that is small compared with ω, but large compared with the reciprocal of the averaging time used in the cross-correlator at the output end in the diagram. In one channel of this scheme the input $u(n)$ is multiplied by $\exp(-j\pi\alpha n)$, and in the other channel it is multiplied by $\exp(j\pi\alpha n)$; the resulting filtered signals are then applied to the cross-correlator. It is these two multiplications (prior to correlation) that provide the spectral-correlation density $S^{\alpha}(\omega)$ with a phase-preserving property for nonzero values of α.

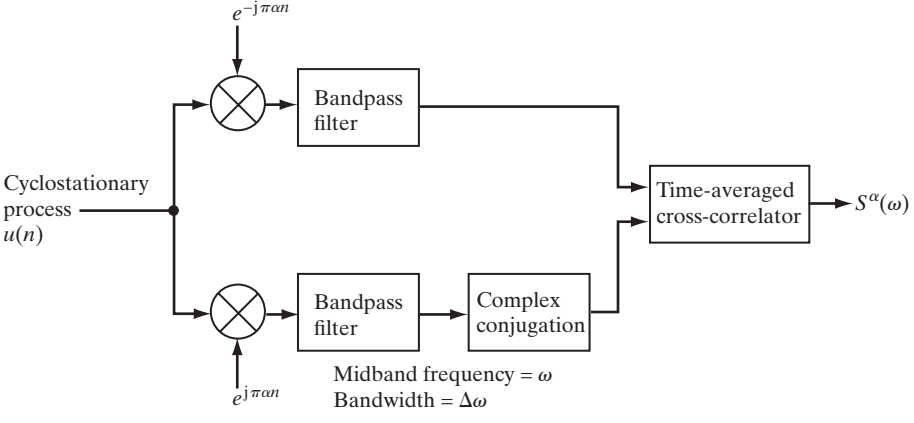


FIGURE 1.16 Scheme for measuring the spectral-correlation density of a cyclostationary process.

1.20 SUMMARY AND DISCUSSION

In this chapter we studied the partial characterization of a stationary discrete-time stochastic process, which, in the time domain, is uniquely described in terms of two statistical parameters:

1. The mean, which is a constant;
2. The autocorrelation function, which depends only on the time difference between any two samples of the process.

The mean of the process may naturally be zero, or it can always be subtracted from the process to yield a new process with zero mean. For that reason, in much of the discussion in subsequent chapters of this book, the mean of the process is assumed to be zero. Thus, given an M-by-1 observation vector $\mathbf{u}(n)$ known to belong to a complex, stationary, discrete-time stochastic process with zero mean, we may partially describe the process by defining an M-by-M correlation matrix $\mathbf{R}$ as the statistical expectation of the outer product of $\mathbf{u}(n)$ with itself, that is, $\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)$. The matrix $\mathbf{R}$ is Hermitian, Toeplitz, and almost always positive definite.

Another topic we discussed in the chapter is the notion of a stochastic model, the need for which arises when we are given a set of experimental data known to be of a statistical nature and the requirement is to analyze the data. In this context, there are two general requirements for a suitable model:

1. An *adequate number of adjustable parameters* for the model to capture the essential information content of the input data.
2. *Mathematical tractability* of the model.

The first requirement, in effect, means that the complexity of the model should closely match the complexity of the underlying physical mechanism responsible for generating the input data; when this is the case, problems associated with underfitting or overfitting the input data are avoided. The second requirement is usually satisfied by the choice of a linear model.

Within the family of linear stochastic models, the autoregressive (AR) model is often preferred over the moving-average (MA) model and the autoregressive–moving-average (ARMA) model for an important reason: Unlike the situation in an MA or ARMA model, computation of the AR coefficients is governed by a system of linear equations, namely, the Yule–Walker equations. Moreover, except for a predictable component, we may approximate a stationary, discrete-time stochastic process by an AR model of sufficiently high order, subject to certain restrictions. To select a suitable value for the model order, we may use an information-theoretic criterion according to Akaike or the minimum description length (MDL) criterion according to Rissanen. A useful feature of the MDL criterion is that it is a consistent model-order estimator.

Another important way of characterizing a wide-sense stationary stochastic process is in terms of the power spectral density or power spectrum. In the latter part of this chapter, we identified three distinct spectral parameters that depend on the statistical characterization of the process:

1. The *power spectral density* $S(\omega)$, defined as the discrete-time Fourier transform of the ordinary autocorrelation function of a wide-sense stationary process. For such

a process, the autocorrelation function is Hermitian, always making $S(\omega)$ a real-valued quantity. Accordingly, $S(\omega)$ destroys phase information about the process. Despite this limitation, the power spectral density is commonly accepted as a useful parameter for displaying the correlation properties of a wide-sense stationary process.

2. *Polyspectra* $C_k(\omega_1, \omega_2, \ldots, \omega_{k-1})$, defined as the multidimensional Fourier transform of the cumulants of a stationary process. For second-order statistics, $k = 2$, $C_2(\omega_1)$ reduces to the ordinary power spectral density $S(\omega)$. For higher-order statistics, $k > 2$, the polyspectra $C_k(\omega_1, \omega_1, \ldots, \omega_{k-1})$ take on complex forms. It is this property of polyspectra that makes them a useful tool for dealing with situations in which knowledge of the phase is a requirement. However, for polyspectra to be meaningful, the process has to be non-Gaussian, and the exploitation of phase information contained in polyspectra requires the use of nonlinear filtering.
3. The *spectral-correlation density* $S^{\alpha}(\omega)$, defined as the discrete-time Fourier transform of the cyclic autocorrelation function of a process that is cyclostationary in the wide sense. For $\alpha \neq 0$, $S^{\alpha}(\omega)$ is complex valued; for $\alpha = 0$, it reduces to $S(\omega)$. The useful feature of $S^{\alpha}(\omega)$ is that it preserves phase information, which can be exploited by means of linear filtering, irrespective of whether the process is or is not Gaussian.

The different properties of the ordinary power spectral density, polyspectra, and the spectral-correlation density give these statistical parameters their own individual areas of application in adaptive filtering.

One last comment is in order: The theories of second-order cyclostationary processes and conventional polyspectra have been brought together under the umbrella of *cyclic polyspectra*. Simply stated, cyclic polyspectra are spectral cumulants in which the individual frequencies involved can add up to any cycle frequency α, whereas they must add up to zero for polyspectra.

PROBLEMS

1. The sequences $y(n)$ and $u(n)$ are related by the difference equation

$$y(n) = u(n + a) + u(n - a),$$

where a is a constant. Evaluate the autocorrelation function of $y(n)$ in terms of $u(n)$.

2. Consider a correlation matrix $\mathbf{R}$ for which the inverse matrix $\mathbf{R}^{-1}$ exists. Show that $\mathbf{R}^{-1}$ is Hermitian.

3. The received signal of a digital communication system is given by

$$u(n) = s(n) + \nu(n)$$

where $s(n)$ is a distorted version of the transmitted signal and $\nu(n)$ denotes additive white Gaussian noise. The correlation matrices of $s(n)$ and $\nu(n)$ are denoted by $\mathbf{R}_s$ and $\mathbf{R}_\nu$, respectively. Assume that the elements of the matrix $\mathbf{R}_\nu$ are defined by

$$r_\nu(l) = \begin{cases} \sigma^2 & \text{for } l = 0 \\ 0 & \text{for } l \neq 0 \end{cases}.$$

Determine the condition that the noise variance σ^2 must satisfy for the correlation matrix of $u(n)$ to be nonsingular. You may illustrate this derivation by considering the case of a two-by-two correlation matrix.

4. In theory, a square matrix can be nonnegative definite and yet singular. Demonstrate the truth of this statement with the example of a 2-by-2 matrix given by

$$\mathbf{R} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

5. **(a)** Equation (1.26) relates the $(M+1)$-by-$(M+1)$ correlation matrix $\mathbf{R}_{M+1}$, pertaining to the observation vector $\mathbf{u}_{M+1}(n)$ taken from a stationary stochastic process, to the M-by-M correlation matrix $\mathbf{R}_M$ of the observation vector $\mathbf{u}_M(n)$ taken from the same process. Evaluate the inverse of the correlation matrix $\mathbf{R}_{M+1}$ in terms of the inverse of the correlation matrix $\mathbf{R}_M$.

(b) Repeat your evaluation using Eq. (1.27).

6. A first-order real-valued AR process $u(n)$ satisfies the real-valued difference equation

$$u(n) + a_1 u(n-1) = \nu(n),$$

where a_1 is a constant and $\nu(n)$ is a white-noise process with variance σ_ν^2.

(a) Show that if $\nu(n)$ has a nonzero mean, the AR process $u(n)$ is nonstationary.

(b) For the case when $\nu(n)$ has zero mean and the constant a_1 satisfies the condition $|a_1| < 1$, show that the variance of $u(n)$ is given by

$$\text{var}[u(n)] = \frac{\sigma_\nu^2}{1 - a_1^2}.$$

(c) For the conditions specified in part (b), find the autocorrelation function of the AR process $u(n)$. Sketch this autocorrelation function for the two cases $0 < a_1 < 1$ and $-1 < a_1 < 0$.

7. Consider an autoregressive process $u(n)$ of order two described by the difference equation

$$u(n) = u(n-1) + 0.5u(n-2) + \nu(n)$$

where $\nu(n)$ is white noise with zero mean and variance 0.5.

(a) Write the Yule–Walker equations for the process.

(b) Find the variance of $u(n)$.

8. Consider a wide-sense stationary process that is modeled as an AR process $u(n)$ of order M. The set of parameters made up of the average power P_0 and the AR coefficients $a_1, a_2, \ldots, a_M$ bear a one-to-one correspondence with the autocorrelation sequence $r(0), r(1), r(2), \ldots, r(M)$, as shown by

$$\{r(0), r(1), r(2), \ldots, r(M)\} \rightleftharpoons \{P_0, a_1, a_2, \ldots, a_M\}.$$

Show that this statement is true.

9. Evaluate the transfer functions of the following two stochastic models:

(a) The MA model of Fig. 1.3.

(b) The ARMA model of Fig. 1.4.

(c) Specify the conditions for which the transfer function of the ARMA model of Fig. 1.4 reduces (1) to that of an AR model and (2) to that of an MA model.

10. Consider an MA process $x(n)$ of order two described by the difference equation

$$x(n) = \nu(n) + 0.75\,\nu(n-1) + 0.75\,\nu(n-2),$$

where $\nu(n)$ is a zero-mean white-noise process of unit variance. The requirement is to approximate this process by an AR process $u(n)$ of order M. Do this approximation for the following orders:
(a) $M = 2$. **(b)** $M = 5$. **(c)** $M = 10$.
Comment on your results. How big would the order M of the AR process $u(n)$ have to be for it to be equivalent to the MA process $x(n)$ exactly?

11. A time series $u(n)$ obtained from a wide-sense stationary stochastic process of zero mean and correlation matrix $\mathbf{R}$ is applied to an FIR filter with impulse response w_n, which defines the coefficient vector $\mathbf{w}$.
(a) Show that the average power of the filter output is equal to $\mathbf{w}^{\mathrm{H}}\mathbf{R}\mathbf{w}$.
(b) How is the result in part (a) modified if the stochastic process at the filter input is a white noise with variance σ^2?

12. A general linear complex-valued process is described by

$$u(n) = \sum_{k=0}^{\infty} b_k^* \nu(n - k),$$

where $\nu(n)$ is white noise and b_k is a complex coefficient. Justify the following statements:
(a) If $\nu(n)$ is Gaussian, then $u(n)$ is also Gaussian.
(b) Conversely, if $u(n)$ is Gaussian, then $\nu(n)$ must be Gaussian.

13. Consider a complex Gaussian process $u(n)$. Let $u(n) = u_n$. Using the Gaussian moment-factoring theorem, demonstrate the following identities:
(a) $\mathbb{E}[(u_1^* u_2)^k] = k!(\mathbb{E}[u_1^* u_2])^k$. **(b)** $\mathbb{E}[|u|^{2k}] = k!(\mathbb{E}[|u|^2])^k$.

14. Consider the definition of the power spectral density given in Eq. (1.115). Is it permissible to interchange the operation of taking the limit and that of the expectation in this equation? Justify your answer.

15. In deriving Eq. (1.126), we invoked the notion that if a wide-sense stationary process is applied to a linear, time-invariant, stable filter, the stochastic process produced at the filter output is wide-sense stationary, too. Show that, in general,

$$r_y(n, m) = \sum_{i=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} h(i) h^*(k) r_u(n - i, m - k),$$

which includes the result of Eq. (1.126) as a special case.

16. How can you estimate the parameters of a complex sinusoid in the presence of additive noise in the correlation matrix?

17. Consider a correlation matrix $\mathbf{R}$ for which the inverse matrix $\mathbf{R}^{-1}$ exists. Under what condition will $\mathbf{R}^{-1}$ be Hermitian and Toeplitz?

18. A real-valued, stationary stochastic process $u(n)$ is said to be *periodic* if its autocorrelation function is periodic, as shown by

$$\begin{aligned} r(l) &= \mathbb{E}[u(n)u(n - l)] \\ &= r(l + N), \end{aligned}$$

where N is the period. Expanding $r(l)$ into a Fourier series, we write

$$r(l) = \frac{1}{N} \sum_{k=0}^{N-1} S_k \exp(\mathrm{j} l \omega_k), \qquad l = 0, 1, \ldots, N - 1,$$

where

$$\omega_k = \frac{2\pi k}{N}$$

and

$$S_k = \sum_{k=0}^{N-1} r(l) \exp(-\mathrm{j}l\omega_k), \qquad k = 0, 1, \ldots, N-1.$$

The parameter

$$S_k = \mathbb{E}[|U_k|^2]$$

specifies the discrete power spectrum; the U_k are complex random variables defined by

$$U_k = \sum_{n=0}^{N-1} u(n) \exp(-\mathrm{j}n\omega_k), \qquad k = 0, 1, \ldots, N-1$$

and

$$u(n) = \frac{1}{N} \sum_{k=0}^{N-1} U_k \exp(\mathrm{j}n\omega_k), \qquad n = 0, 1, \ldots, N-1.$$

(a) Show that the spectral samples $U_0, U_1, \ldots, U_{N-1}$ are uncorrelated; that is, show that

$$\mathbb{E}[U_k U_j^*] = \begin{cases} S_k & \text{for } j = k \\ 0 & \text{otherwise} \end{cases}.$$

(b) Assuming that $u(n)$, and therefore U_k, is Gaussian distributed, show that the joint probability density function of $U_0, U_1, \ldots, U_{N-1}$ is given by

$$f_{\mathrm{U}}(U_0, U_1, \ldots, U_{N-1}) = \pi^{-N} \exp\left(- \sum_{k=0}^{N-1} \frac{|U_k|^2}{S_k} - \ln \mathrm{S}_k \right).$$

19. Determine the correlation matrix **R** for $M = 2$ no. of samples. [*Hint*: signal-to-noise $\rho = \infty$.]

20. Show that the integral of a stochastic process $X(t)$ is a random variable.

21. Evaluate the transfer functions of the stochastic MA model of Fig 1.3.

22. Consider a stochastic process whose mean is zero. The process is *white*, which means that its power spectral density $\mathrm{S}(\omega)$ is constant for all frequencies, as shown by

$$S(\omega) = \sigma^2 \text{ for } -2\pi < \omega < -2\pi$$

where σ^2 is the variance of a sample taken from the process. Suppose that this process is passed through a discrete-time band pass filter as characterized in Fig. 1.13, find the mean-square of the filter output.

23. Determine the complex Gaussian process for the odd case of $N = 5$ with $u(n)$ consisting of the five samples u_1, u_2, u_3, u_4, and u_5.

24. By using the Gaussian moment-factoring theorem, determine the complex Gaussian process for the even case of $N = 2$. $U(n)$ consists of the samples u_1 and u_2.

25. Prove that a random process {X (t)} is mean square continuous if its autocorrelation function is continuous.

CHAPTER 2

Wiener Filters

With the material of Chapter 1 on the statistical characterization of stationary stochastic processes at hand, we are ready to develop a framework for assessing the performance of linear adaptive filters. In particular, in this chapter we study a class of *linear optimum discrete-time filters* known collectively as *Wiener filters*. Wiener filter theory is formulated for the general case of a complex-valued stochastic process with the filter specified in terms of its impulse response. The reason for using complex-valued time series is that in many practical situations (e.g., communications, radar, sonar) the observables are measured in baseband form; as mentioned in Chapter 1, the term "baseband" is used to designate a band of frequencies representing the original signal, as delivered by the source of information. The case of real-valued time series may, of course, be considered a special case of this theory. We begin the chapter by outlining the linear optimum filtering problem and setting the stage for examining the rest of the theory of the Wiener filter and its variants.

2.1 LINEAR OPTIMUM FILTERING: STATEMENT OF THE PROBLEM

Consider the block diagram of Fig. 2.1 built around a *linear discrete-time filter*. The filter *input* consists of a *time series* $u(0), u(1), u(2), \ldots$, and the filter is itself characterized by the *impulse response* represented by the sequence $w_0, w_1, w_2, \ldots$. At some *discrete time* n, the filter produces an *output* denoted by $y(n)$. This output is used to provide an *estimate* of a *desired response* designated by $d(n)$. With the filter input and the desired response representing single realizations of respective stochastic processes, the estimation is ordinarily accompanied by an error with statistical characteristics of its own. In particular, the *estimation error*, denoted by $e(n)$, is defined as the difference between the desired response $d(n)$ and the filter output $y(n)$. The requirement is to make the estimation error $e(n)$ "as small as possible" in some statistical sense.

Two restrictions have so far been placed on the filter:

1. The filter is *linear*, which makes the mathematical analysis easy to handle.
2. The filter operates in *discrete time*, which makes it possible for the filter to be implemented using digital hardware or software.

The final details of the filter specification, however, depend on two other choices that have to be made:

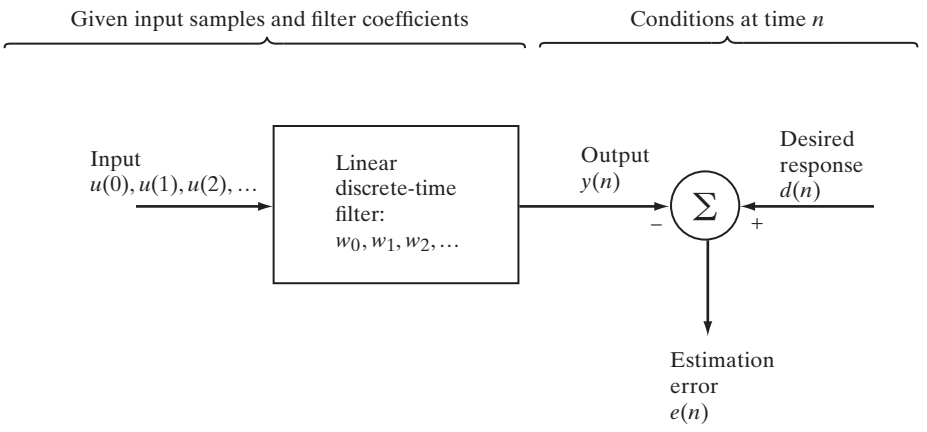


FIGURE 2.1 Block diagram representation of the statistical filtering problem.

1. Whether the impulse response of the filter has *finite* or *infinite* duration.
2. The type of *statistical criterion* used for the optimization.

The choice of a *finite-duration impulse response* (FIR) or an *infinite-duration impulse response* (IIR) for the filter is dictated by *practical considerations*. The choice of a statistical criterion for optimizing the filter design is influenced by *mathematical tractability*. We next consider these two issues in turn.

For the initial development of Wiener filter theory, we will assume an IIR filter; so developed, the theory includes FIR filters as a special case. However, for much of the material presented in this chapter, and also in the rest of the book, we will confine our attention to the use of FIR filters. We do so because an FIR filter is *inherently stable*, since its structure involves the use of *forward paths only*. In other words, the only mechanism for input–output interaction in the filter is via forward paths from the filter input to its output. Indeed, it is this form of signal transmission through the filter that limits its impulse response to a finite duration. On the other hand, an IIR filter involves *both feedforward and feedback*. The presence of feedback means that portions of the filter output and possibly other *internal* variables in the filter are fed back to the input. Consequently, unless the filter is properly designed, feedback can indeed make it *unstable*, with the result that the filter *oscillates*; this kind of operation is clearly unacceptable when the requirement is that stability is a "must." By itself, the stability problem in IIR filters is manageable in both theoretic and practical terms. However, when the filter is required to be *adaptive*, bringing with it stability problems of its own, the inclusion of adaptivity combined with feedback that is inherently present in an IIR filter makes an already difficult problem that much more difficult to handle. It is for this reason that we find that in the majority of applications requiring the use of adaptivity, the use of an FIR filter is preferred over an IIR filter even though the latter is less demanding in computational requirements.

Turning next to the issue of what criterion to choose for statistical optimization, we find that there are indeed several criteria that suggest themselves. Specifically, we

may consider optimizing the filter design by *minimizing a cost function*, or *index of performance*, selected from the following short list of possibilities:

1. Mean-square value of the estimation error.
2. Expectation of the absolute value of the estimation error.
3. Expectation of third or higher powers of the absolute value of the estimation error.

Option 1 has a clear advantage over the other two, because it leads to tractable mathematics. In particular, the choice of *the mean-square-error criterion results in a second-order dependence for the cost function on the unknown coefficients in the impulse response of the filter. Moreover, the cost function has a distinct minimum that uniquely defines the optimum statistical design of the filter*. Accordingly, henceforth we confine attention to the mean-square-error criterion.

We now summarize the essence of the filtering problem with the following statement:

> Design a linear discrete-time filter whose output $y(n)$ provides an estimate of a desired response $d(n)$, given a set of input samples $u(0), u(1), u(2), \ldots$, such that the mean-square value of the estimation error $e(n)$, defined as the difference between the desired response $d(n)$ and the actual response $y(n)$, is minimized.

We develop the mathematical solution to this statistical optimization problem by following two entirely different approaches that are complementary. One approach leads to the development of an important theorem commonly known as the principle of orthogonality. The other approach highlights the error-performance surface that describes the second-order dependence of the cost function on the filter coefficients. We will proceed by deriving the principle of orthogonality first, because the derivation is relatively simple and the principle is highly insightful.

2.2 PRINCIPLE OF ORTHOGONALITY

Consider again the statistical filtering problem described in Fig. 2.1. The filter input is denoted by the time series $u(0), u(1), u(2), \ldots$, and the impulse response of the filter is denoted by $w_0, w_1, w_2, \ldots$, both of which are assumed to have *complex values* and *infinite duration*. The filter output at a discrete time n is defined by the *linear convolution sum*

$$y(n) = \sum_{k=0}^{\infty} w_k^* u(n-k), \qquad n = 0, 1, 2, \ldots, \tag{2.1}$$

where the asterisk denotes *complex conjugation*. Note that, in complex terminology, the term $w_k^* u(n-k)$ represents the scalar version of an *inner product of the filter coefficient w_k and the filter input $u(n-k)$*. Figure 2.2 illustrates the steps involved in computing the linear discrete-time form of convolution described in Eq. (2.1) for real data.

The purpose of the filter in Fig. 2.1 is to produce an estimate of the desired response $d(n)$. We assume that the filter input and the desired response are single realizations of *jointly wide-sense stationary stochastic processes*, both with zero mean. If the means are nonzero, we simply subtract them from the input $u(n)$ and desired response $d(n)$ before filtering, in accordance with the remarks on preprocessing made in Section 1.2. The estimation of $d(n)$ is naturally accompanied by an error, which is defined by the difference

$$e(n) = d(n) - y(n). \tag{2.2}$$

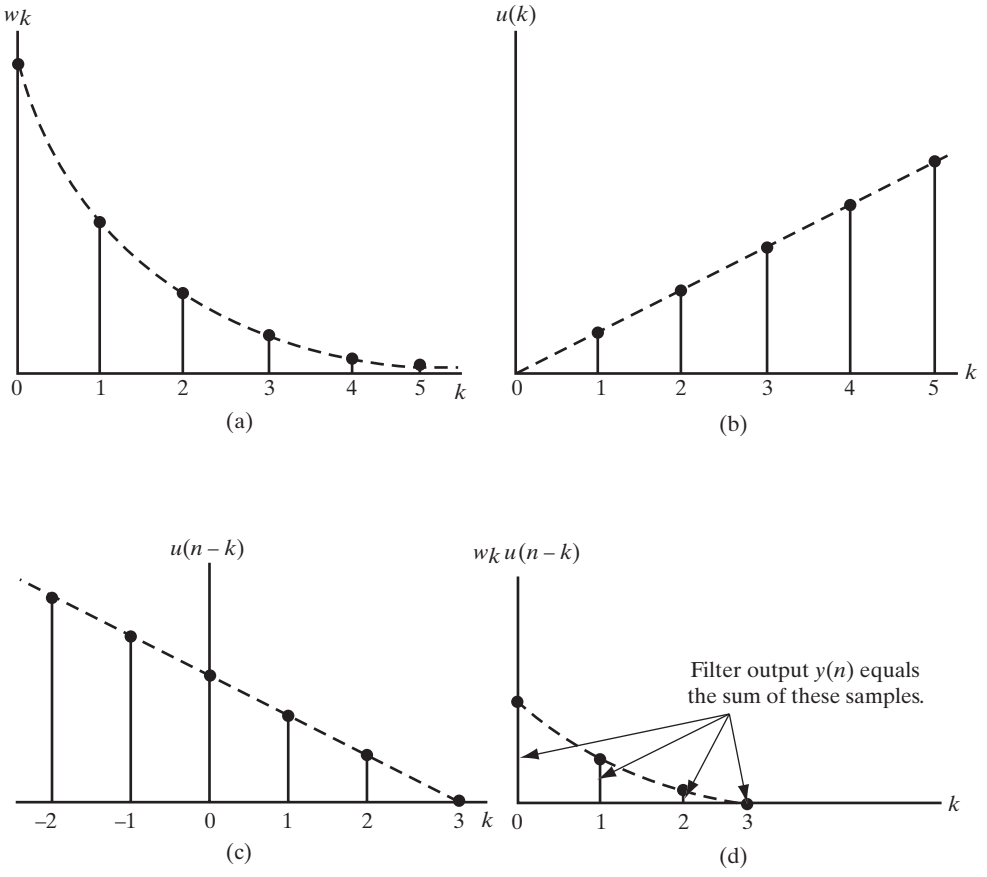


FIGURE 2.2 Linear convolution: (a) impulse response; (b) filter input; (c) time-reversed and shifted version of filter input; (d) calculation of filter output at time $n = 3$.

The estimation error $e(n)$ is the sample value of a random variable. *To optimize the filter design, we choose to minimize the mean-square value of* $e(n)$. We thus define the cost function as the *mean-square error*

$$\begin{aligned} J &= \mathbb{E}[e(n)e^*(n)] \\ &= \mathbb{E}[|e(n)|^2], \end{aligned} \tag{2.3}$$

where $\mathbb{E}$ denotes the *statistical expectation operator*. The requirement is therefore to determine the operating conditions under which J attains its minimum value.

For complex input data, the filter coefficients are, in general, complex, too. Let the kth filter coefficient w_k be denoted in terms of its real and imaginary parts as

$$w_k = a_k + \mathrm{j}b_k, \qquad k = 0, 1, 2, \ldots . \tag{2.4}$$

Correspondingly, we may define a *gradient operator*, the kth element of which is written in terms of first-order partial derivatives with respect to the real part a_k and the imaginary part b_k, as

$$\nabla_k = \frac{\partial}{\partial a_k} + \mathrm{j}\frac{\partial}{\partial b_k}, \qquad k = 0, 1, 2, \ldots . \tag{2.5}$$

Thus, for the situation at hand, applying the operator ∇ to the cost function J, we obtain a multidimensional complex *gradient vector* ∇J, the kth element of which is

$$\nabla_k J = \frac{\partial J}{\partial a_k} + \mathrm{j}\frac{\partial J}{\partial b_k}, \qquad k = 0, 1, 2, \ldots . \tag{2.6}$$

Equation (2.6) represents a natural extension of the customary definition of the gradient for a function of real coefficients to the more general case of a function of complex coefficients.[1] Note that for the definition of the complex gradient given in Eq. (2.6) to be valid, it is essential that J be *real*. The gradient operator is always used in the context of finding the *stationary points* of a function of interest. This means that a complex constraint must be converted to a pair of *real* constraints. In Eq. (2.6), the pair of real constraints is obtained by setting both the real and imaginary parts of $\nabla_k J$ equal to zero.

For the cost function J to attain its minimum value, all the elements of the gradient vector ∇J must be simultaneously equal to zero; that is,

$$\nabla_k J = 0, \qquad k = 0, 1, 2, \ldots . \tag{2.7}$$

Under this set of conditions, the filter is said to be *optimum in the mean-square-error sense*.[2]

According to Eq. (2.3), the cost function J is a scalar that is independent of time n. Hence, substituting the first term of that equation into Eq. (2.6), we get

$$\nabla_k J = \mathbb{E}\left[\frac{\partial e(n)}{\partial a_k}e^*(n) + \frac{\partial e^*(n)}{\partial a_k}e(n) + \frac{\partial e(n)}{\partial b_k}\mathrm{j}e^*(n) + \frac{\partial e^*(n)}{\partial b_k}\mathrm{j}e(n)\right]. \tag{2.8}$$

Using Eqs. (2.2) and (2.4), we obtain the four partial derivatives

$$\begin{aligned}
\frac{\partial e(n)}{\partial a_k} &= -u(n-k),\\
\frac{\partial e(n)}{\partial b_k} &= \mathrm{j}u(n-k),\\
\frac{\partial e^*(n)}{\partial a_k} &= -u^*(n-k),\\
\frac{\partial e^*(n)}{\partial b_k} &= -\mathrm{j}u^*(n-k).
\end{aligned} \tag{2.9}$$

[1]For the general case of complex data, the cost function J is not complex–differentiable for reasons discussed in Appendix B; see Problem 1 for an illustrative example.

In this chapter we have gotten around this difficulty by defining the complex gradient vector of the cost function with respect to a set of filter coefficients in the manner described in Eq. (2.6). Specifically, the kth partial derivative of the complex gradient vector consists of two parts, one dealing with the real part of the kth filter coefficient and the other dealing with its imaginary part. From an algebraic perspective, this procedure makes intuitive sense.

In Appendix B, we describe another procedure based on Wirtinger calculus, which is simple and straightforward to apply. Mathematically, however, it is more sophisticated than the procedure described in this chapter.

[2]Note that in Eq. (2.7), we have presumed optimality at a stationary point. In the linear filtering problem, finding a stationary point assures global optimization of the filter by virtue of the quadratic nature of the error-performance surface. (See Section 2.5.)

Thus, substituting these partial derivatives into Eq. (2.8) and then cancelling common terms finally yields

$$\nabla_k J = -2\mathbb{E}[u(n-k)e^*(n)]. \tag{2.10}$$

We are now ready to specify the operating conditions required for minimizing the cost function J. Let e_o denote *the special value of the estimation error that results when the filter operates in its optimum condition.* We then find that the conditions specified in Eq. (2.7) are indeed equivalent to

$$\mathbb{E}[u(n-k)e_o^*(n)] = 0, \qquad k = 0, 1, 2, \ldots. \tag{2.11}$$

In words, Eq. (2.11) states the following:

> The necessary and sufficient condition for the cost function J to attain its minimum value is for the corresponding value of the estimation error $e_o(n)$ to be orthogonal to each input sample that enters into the estimation of the desired response at time n.

Indeed, this statement constitutes the *principle of orthogonality*; it represents one of the most elegant theorems in the subject of linear optimum filtering. It also provides the mathematical basis of a procedure for testing whether the linear filter is operating in its optimum condition.

Corollary to the Principle of Orthogonality

There is a corollary to the principle of orthogonality that we may derive by examining the correlation between the filter output $y(n)$ and the estimation error $e(n)$. Using Eq. (2.1), we may express this correlation as

$$\begin{aligned}\mathbb{E}[y(n)e^*(n)] &= \mathbb{E}\left[\sum_{k=0}^{\infty} w_k^* u(n-k)e^*(n)\right] \\ &= \sum_{k=0}^{\infty} w_k^* \mathbb{E}[u(n-k)e^*(n)].\end{aligned} \tag{2.12}$$

Let $y_o(n)$ denote the output produced by the filter optimized in the mean-square-error sense, with $e_o(n)$ *denoting the corresponding estimation error.* Hence, using the principle of orthogonality described by Eq. (2.11), we get the desired result:

$$\mathbb{E}[y_o(n)e_o^*(n)] = 0. \tag{2.13}$$

We may thus state the corollary to the principle of orthogonality as follows:

> When the filter operates in its optimum condition, the estimate of the desired response defined by the filter output $y_o(n)$ and the corresponding estimation error $e_o(n)$ are orthogonal to each other.

Let $\hat{d}(n|\mathcal{U}_n)$ denote *the estimate of the desired response that is optimized in the mean-square-error sense, given the input data that span the space* $\mathcal{U}_n$ *up to and including time* n.[3]

We may then write

$$\hat{d}(n|\mathcal{U}_n) = y_o(n). \tag{2.14}$$

Note that the estimate $\hat{d}(n|\mathcal{U}_n)$ has zero mean, because the tap inputs are assumed to have zero mean. This condition matches the assumed zero mean of the desired response $d(n)$.

Geometric Interpretation of the Corollary to the Principle of Orthogonality

Equation (2.13) offers an interesting geometric interpretation of the conditions that exist at the output of the optimum filter, as illustrated in Fig. 2.3. In this figure, the desired response, the filter output, and the corresponding estimation error are represented by vectors labeled $\mathbf{d}$, $\mathbf{y}_o$, and $\mathbf{e}_o$, respectively; the subscript o in $\mathbf{y}_o$ and $\mathbf{e}_o$ refers to the optimum condition. We see that, for the optimum filter, the vector representing the estimation error is *normal* (i.e., perpendicular) to the vector representing the filter output. It should, however, be emphasized that the situation depicted in the Fig. 2.3 is merely an *analogy* wherein random variables and expectations are replaced with vectors and vector inner products, respectively. Also, for obvious reasons, the geometry depicted in the figure may be viewed as a *statistician's Pythagorean theorem*.

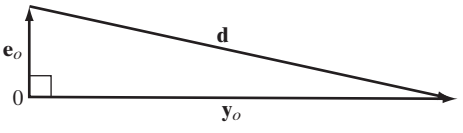


FIGURE 2.3 Geometric interpretation of the relationship between the desired response, the estimate at the filter output, and the estimation error.

2.3 MINIMUM MEAN-SQUARE ERROR

When the linear discrete-time filter in Fig. 2.1 operates in its optimum condition, Eq. (2.2) takes on the special form

$$\begin{aligned} e_o(n) &= d(n) - y_o(n) \\ &= d(n) - \hat{d}(n|\mathcal{U}_n), \end{aligned} \tag{2.15}$$

[3]If a space $\mathcal{U}_n$ consists of all linear combinations of random variables, $u_1, u_2, \ldots, u_n$, then these random variables are said to *span* that particular space. In other words, every random variable in $\mathcal{U}_n$ can be expressed as some combination of the u's, as shown by

$$u = w_1^* u_1 + \ldots + w_n^* u_n$$

for some coefficients $w_1, \ldots, w_n$. This assumes that the space $\mathcal{U}_n$ has a finite dimension.

where, in the second line, we have made use of Eq. (2.14). Rearranging terms in Eq. (2.15), we have

$$d(n) = \hat{d}(n|\mathcal{U}_n) + e_o(n). \tag{2.16}$$

Let

$$J_{\min} = \mathbb{E}[\,|e_o(n)|^2\,] \tag{2.17}$$

denote the mininum mean-square error. Then, evaluating the mean-square values of both sides of Eq. (2.16) and applying to it the corollary to the principle of orthogonality described by Eqs. (2.13) and (2.14), we get

$$\sigma_d^2 = \sigma_{\hat{d}}^2 + J_{\min}, \tag{2.18}$$

where σ_d^2 is the variance of the desired response and $\sigma_{\hat{d}}^2$ is the variance of the estimate $\hat{d}(n|\mathcal{U}_n)$; both of these random variables are assumed to be of zero mean. Solving Eq. (2.18) for the minimum mean-square error, we get

$$J_{\min} = \sigma_d^2 - \sigma_{\hat{d}}^2. \tag{2.19}$$

This relation shows that, for the optimum filter, the minimum mean-square error equals the difference between the variance of the desired response and the variance of the estimate that the filter produces at its output.

It is convenient to normalize the expression in Eq. (2.19) in such a way that the minimum value of the mean-square error always lies between zero and unity. We may do this by dividing both sides of Eq. (2.19) by σ_d^2, obtaining

$$\frac{J_{\min}}{\sigma_d^2} = 1 - \frac{\sigma_{\hat{d}}^2}{\sigma_d^2}. \tag{2.20}$$

Clearly, this is possible because σ_d^2 is never zero, except in the trivial case of a desired response $d(n)$ that is zero for all n. Now, let

$$\varepsilon = \frac{J_{\min}}{\sigma_d^2}. \tag{2.21}$$

The quantity ε is called the *normalized mean-square error*, in terms of which we may rewrite Eq. (2.20) in the form

$$\varepsilon = 1 - \frac{\sigma_{\hat{d}}^2}{\sigma_d^2}. \tag{2.22}$$

We note that (1) the ratio ε can never be negative and (2) the ratio $\sigma_{\hat{d}}^2/\sigma_d^2$ is always positive. We therefore have

$$0 \le \varepsilon \le 1. \tag{2.23}$$

If ε is zero, the optimum filter operates perfectly, in the sense that there is complete agreement between the estimate $\hat{d}(n|\mathcal{U}_n)$ at the filter output and the desired response $d(n)$. On the other hand, *if ε is unity, there is no agreement whatsoever between these two quantities; this corresponds to the worst possible situation.*

2.4 WIENER–HOPF EQUATIONS

The principle of orthogonality, described in Eq. (2.11), specifies the necessary and sufficient condition for the optimum operation of the filter. We may reformulate the necessary and sufficient condition for optimality by substituting Eqs. (2.1) and (2.2) into Eq. (2.11). In particular, we write

$$\mathbb{E}\left[u(n-k)\left(d^*(n) - \sum_{i=0}^{\infty} w_{oi}u^*(n-i)\right)\right] = 0, \qquad k = 0, 1, 2, \ldots,$$

where w_{oi} is the ith coefficient in the impulse response of the optimum filter. Expanding this equation and rearranging terms, we get

$$\sum_{i=0}^{\infty} w_{oi}\mathbb{E}[u(n-k)u^*(n-i)] = \mathbb{E}[u(n-k)d^*(n)], \qquad k = 0, 1, 2\ldots. \quad (2.24)$$

The two expectations in Eq. (2.24) may be interpreted as follows:

1. The expectation $\mathbb{E}[u(n-k)u^*(n-i)]$ is equal to the *autocorrelation function of the filter input* for a lag of $i-k$. We may thus express this expectation as
$$r(i-k) = \mathbb{E}[u(n-k)u^*(n-i)]. \quad (2.25)$$
2. The expectation $\mathbb{E}[u(n-k)d^*(n)]$ is equal to the cross-correlation between the filter input $u(n-k)$ and the desired response $d(n)$ for a lag of $-k$. We may thus express this second expectation as
$$p(-k) = \mathbb{E}[u(n-k)d^*(n)]. \quad (2.26)$$

Accordingly, using the definitions of Eqs. (2.25) and (2.26) in Eq. (2.24), we get an infinitely large system of equations as the necessary and sufficient condition for optimality of the filter:

$$\sum_{i=0}^{\infty} w_{oi}r(i-k) = p(-k), \qquad k = 0, 1, 2, \ldots. \quad (2.27)$$

The system of equations (2.27) defines the optimum filter coefficients, in the most general setting, in terms of two correlation functions: the autocorrelation function of the filter input and the cross-correlation between the filter input and the desired response. These equations are called the *Wiener–Hopf equations*.[4]

Solution of the Wiener–Hopf Equations for FIR Filters

The solution of the set of Wiener–Hopf equations is greatly simplified for the special case when an FIR filter, also known as a *linear transversal filter*, is used to obtain the estimation of the desired response $d(n)$ in Fig. 2.1. Consider, then, the structure shown in Fig. 2.4. The

[4]In order to solve the Wiener–Hopf equations (2.27) for the optimum filter coefficients, we need to use a special technique known as *spectral factorization*. For a description of this technique and its use in solving the Wiener–Hopf equations (2.27), the interested reader is referred to Haykin (1989a).

It should also be noted that the defining equation for a linear optimim filter was formulated originally by Wiener and Hopf (1931) for the case of a continuous-time filter, whereas, of course, the system of equations (2.27) is formulated for a discrete-time filter; the latter is simpler than the former.

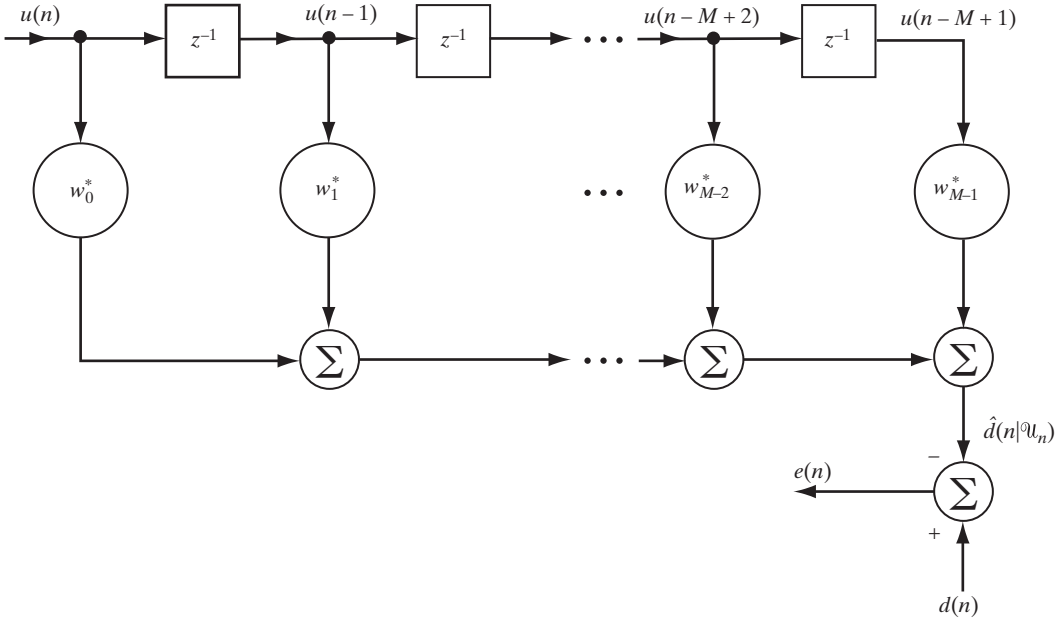


FIGURE 2.4 FIR filter for studying the Wiener filter.

FIR filter involves a combination of three basic operations: *storage, multiplication,* and *addition*, as described here:

1. The storage is represented by a cascade of $M - 1$ *one-sample delays*, with the block for each such unit labeled z^{-1}. We refer to the various points at which the one-sample delays are accessed as *tap points*. The tap inputs are denoted by $u(n), u(n-1), \ldots, u(n-M+1)$. Thus, with $u(n)$ viewed as the *current* value of the filter input, the remaining $M - 1$ tap inputs, $u(n-1), \ldots, u(n-M+1)$, represent *past values of the input*.
2. The scalar *inner products* of tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$ and *tap weights* $w_0, w_1, \ldots, w_{M-1}$ are respectively formed by using a corresponding set of multipliers. In particular, the multiplication involved in forming the scalar inner product of $u(n)$ and w_0 is represented by a block labeled w_0^*, and so on for the other inner products.
3. The function of the adders is to sum the multiplier outputs to produce an overall output for the filter.

The impulse response of the FIR filter in Fig. 2.4 is defined by the finite set of tap weights $w_0, w_1, \ldots, w_{M-1}$. Accordingly, the Wiener–Hopf equations (2.27) reduce to the *system of M simultaneous equations*.

$$\sum_{i=0}^{M-1} w_{oi} r(i-k) = p(-k), \qquad k = 0, 1, \ldots, M-1, \tag{2.28}$$

where $w_{o0}, w_{o1}, \ldots, w_{o,M-1}$ are the optimum values of the tap weights of the filter.

Matrix Formulation of the Wiener–Hopf Equations

Let $\mathbf{R}$ denote the M-by-M *correlation matrix* of the tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$ in the FIR filter of Fig. 2.4; that is,

$$\mathbf{R} = \mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)], \tag{2.29}$$

where

$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^{\mathrm{T}} \tag{2.30}$$

is the M-by-1 tap-input vector. The superscript T in Eq. (2.30) denotes *transposition*. In expanded form, we have

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r^*(1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r^*(M-1) & r^*(M-2) & \cdots & r(0) \end{bmatrix}. \tag{2.31}$$

Correspondingly, let $\mathbf{p}$ denote the M-by-1 *cross-correlation vector* between the tap inputs of the filter and the desired response $d(n)$:

$$\mathbf{p} = \mathbb{E}[\mathbf{u}(n)d^*(n)]. \tag{2.32}$$

In expanded form, we have

$$\mathbf{p} = [p(0), p(-1), \ldots, p(1-M)]^{\mathrm{T}}. \tag{2.33}$$

Note that the lags used in the definition of $\mathbf{p}$ are either zero or negative. We may thus rewrite the Wiener–Hopf equations (2.28) in the compact matrix form

$$\mathbf{R}\mathbf{w}_o = \mathbf{p}, \tag{2.34}$$

where $\mathbf{w}_o$ denotes the M-by-1 *optimum tap-weight vector* of the FIR filter (optimum in the mean-square-error sense); that is,

$$\mathbf{w}_o = [\mathbf{w}_{o0}, \mathbf{w}_{o1}, \ \ldots\ , \mathbf{w}_{o,M-1}]^{\mathrm{T}}. \tag{2.35}$$

To solve the Wiener–Hopf equations (2.34) for $\mathbf{w}_o$, we assume that the correlation matrix $\mathbf{R}$ is nonsingular. We may then premultiply both sides of Eq. (2.34) by $\mathbf{R}^{-1}$, the *inverse* of the correlation matrix, obtaining

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p}. \tag{2.36}$$

The computation of the optimum tap-weight vector $\mathbf{w}_o$ requires knowledge of two quantities: (1) the correlation matrix $\mathbf{R}$ of the tap-input vector $\mathbf{u}(n)$ and (2) the cross-correlation vector $\mathbf{p}$ between the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$.

2.5 ERROR-PERFORMANCE SURFACE

The Wiener–Hopf equations (2.34), as derived in the previous section, are traceable to the principle of orthogonality, which itself was derived in Section 2.2. We may also derive the Wiener–Hopf equations by examining the dependence of the cost function J on the tap weights of the FIR filter in Fig. 2.4. First, we write the estimation error $e(n)$ as

$$e(n) = d(n) - \sum_{k=0}^{M-1} w_k^* u(n-k), \tag{2.37}$$

where $d(n)$ is the desired response; $w_0, w_1, \ldots, w_{M-1}$ are the tap weights of the filter; and $u(n), u(n-1), \ldots, u(n-M+1)$ are the corresponding tap inputs. Accordingly, we may define the cost function for the FIR filter structure of Fig. 2.4 as

$$\begin{aligned} J &= \mathbb{E}[e(n)e^*(n)] \\ &= \mathbb{E}[|d(n)|^2] - \sum_{k=0}^{M-1} w_k^* \mathbb{E}[u(n-k)d^*(n)] - \sum_{k=0}^{M-1} w_k \mathbb{E}[u^*(n-k)d(n)] \\ &\quad + \sum_{k=0}^{M-1}\sum_{i=0}^{M-1} w_k^* w_i \mathbb{E}[u(n-k)u^*(n-i)]. \end{aligned} \tag{2.38}$$

We now recognize the four expectations on the right-hand side of the second line in Eq. (2.38):

- For the first expectation, we have

$$\sigma_d^2 = \mathbb{E}[|d(n)|^2], \tag{2.39}$$

  where σ_d^2 is the *variance* of the desired response $d(n)$, assumed to be of zero mean.
- For the second and third expectations, we have, respectively,

$$p(-k) = \mathbb{E}[u(n-k)d^*(n)] \tag{2.40}$$

  and

$$p^*(-k) = \mathbb{E}[u^*(n-k)d(n)], \tag{2.41}$$

  where $p(-k)$ is the cross-correlation between the tap input $u(n-k)$ and the desired response $d(n)$.
- Finally, for the fourth expectation, we have

$$r(i-k) = \mathbb{E}[u(n-k)u^*(n-i)], \tag{2.42}$$

  where $r(i-k)$ is the autocorrelation function of the tap inputs for lag $i-k$.

We may thus rewrite Eq. (2.38) in the form

$$J = \sigma_d^2 - \sum_{k=0}^{M-1} w_k^* p(-k) - \sum_{k=0}^{M-1} w_k p^*(-k) + \sum_{k=0}^{M-1}\sum_{i=0}^{M-1} w_k^* w_i r(i-k). \tag{2.43}$$

Equation (2.43) states that, for the case when the tap inputs of the FIR filter and the desired response are jointly stationary, the cost function, or mean-square error, J is precisely a *second-order function of the tap weights in the filter*. Consequently, we may visualize the dependence of J on the tap weights $w_0, w_1, \ldots, w_{M-1}$ as a *bowl-shaped* $(M+1)$*-dimensional surface* with M *degrees of freedom represented by the tap weights of the filter*; it is $(M+1)$-dimensional because we have the variance σ_d^2 plus the M tap weights which are all needed to fully describe the cost function J. This surface is characterized by a unique minimum. For obvious reasons, we refer to the surface so described as the *error-performance surface* of the FIR filter in Fig. 2.4.

At the *bottom*, or *minimum point*, of the error-performance surface, the cost function J attains its *minimum value*, denoted by $J_{\min}$. At this point, the *gradient vector* ∇J is identically zero. In other words,

$$\nabla_k J = 0, \qquad k = 0, 1, \ldots, M-1, \tag{2.44}$$

where $\nabla_k J$ is the kth element of the gradient vector. As before, we write the kth tap weight as

$$w_k = a_k + \mathrm{j} b_k.$$

Hence, using Eq. (2.43), we may express $\nabla_k J$ as

$$\begin{aligned} \nabla_k J &= \frac{\partial J}{\partial a_k} + \mathrm{j}\,\frac{\partial J}{\partial b_k} \\ &= -2p(-k) + 2\sum_{i=0}^{M-1} w_i r(i-k). \end{aligned} \tag{2.45}$$

Applying the necessary and sufficient condition of Eq. (2.44) for optimality to Eq. (2.45), we find that the optimum tap weights $w_{o0}, w_{o1}, \ldots, w_{o,M-1}$ for the FIR filter in Fig. 2.4 are defined by the system of equations

$$\sum_{i=0}^{M-1} w_{oi} r(i-k) = p(-k), \qquad k = 0, 1, \ldots, M-1.$$

This system of equations is identical to the Wiener–Hopf equations (2.28) derived in Section 2.4.

Minimum Mean-Square Error

Let $\hat{d}(n|\mathcal{U}_n)$ denote the estimate of the desired response $d(n)$, produced at the output of the FIR filter in Fig. 2.4 that is optimized in the mean-square-error sense, given the tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$ that span the space $\mathcal{U}_n$. Then, from that figure, we deduce

$$\begin{aligned} \hat{d}(n|\mathcal{U}_n) &= \sum_{k=0}^{M-1} w_{ok}^{*} u(n-k) \\ &= \mathbf{w}_o^{\mathrm{H}} \mathbf{u}(n), \end{aligned} \tag{2.46}$$

where $\mathbf{w}_o$ is the tap-weight vector of the optimum filter with elements $w_{o0}, w_{o1}, \ldots, w_{o,M-1}$, and $\mathbf{u}(n)$ is the tap-input vector defined in Eq. (2.30). Note that $\mathbf{w}_o^{\mathrm{H}}\mathbf{u}(n)$ denotes an inner product of the optimum tap-weight vector $\mathbf{w}_o$ and the tap-input vector $\mathbf{u}(n)$. We assume that $\mathbf{u}(n)$ has zero mean, which makes the estimate $\hat{d}(n|\mathcal{U}_n)$ have zero mean, too. Hence, we may use Eq. (2.46) to evaluate the variance of $\hat{d}(n|\mathcal{U}_n)$, obtaining

$$\begin{aligned} \sigma_{\hat{d}}^2 &= \mathbb{E}[\mathbf{w}_o^{\mathrm{H}}\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{w}_o] \\ &= \mathbf{w}_o^{\mathrm{H}}\mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]\mathbf{w}_o \\ &= \mathbf{w}_o^{\mathrm{H}}\mathbf{R}\mathbf{w}_o, \end{aligned} \tag{2.47}$$

where $\mathbf{R}$ is the correlation matrix of the tap-weight vector $\mathbf{u}(n)$, as defined in Eq. (2.29). We may eliminate the dependence of the variance $\sigma_{\hat{d}}^2$ on the optimum tap-weight vector $\mathbf{w}_o$ by using Eq. (2.34). In particular, we may rewrite Eq. (2.47) as

$$\begin{aligned}\sigma_{\hat{d}}^2 &= \mathbf{p}^{\mathrm{H}}\mathbf{w}_o \\ &= \mathbf{p}^{\mathrm{H}}\mathbf{R}^{-1}\mathbf{p}.\end{aligned} \tag{2.48}$$

To evaluate the minimum mean-square error produced by the FIR filter in Fig. 2.4, we may use Eq. (2.47) or Eq. (2.48) in Eq. (2.19), obtaining

$$\begin{aligned}J_{\min} &= \sigma_d^2 - \mathbf{w}_o^{\mathrm{H}}\mathbf{R}\mathbf{w}_o \\ &= \sigma_d^2 - \mathbf{p}^{\mathrm{H}}\mathbf{w}_o \\ &= \sigma_d^2 - \mathbf{p}^{\mathrm{H}}\mathbf{R}^{-1}\mathbf{p},\end{aligned} \tag{2.49}$$

which is the desired result.

Canonical Form of the Error-Performance Surface

Equation (2.43) defines the expanded form of the mean-square error J produced by the FIR filter in Fig. 2.4. We may rewrite this equation in matrix form by using the definitions of the correlation matrix $\mathbf{R}$ and the cross-correlation vector $\mathbf{p}$ given in Eqs. (2.31) and (2.33), respectively, as shown by

$$J(\mathbf{w}) = \sigma_d^2 - \mathbf{w}^{\mathrm{H}}\mathbf{p} - \mathbf{p}^{\mathrm{H}}\mathbf{w} + \mathbf{w}^{\mathrm{H}}\mathbf{R}\mathbf{w}, \tag{2.50}$$

where the mean-square error is written as $J(\mathbf{w})$ to emphasize its dependence on the tap-weight vector $\mathbf{w}$. As pointed out in Chapter 1, the correlation matrix $\mathbf{R}$ is almost always nonsingular, in which case the inverse matrix $\mathbf{R}^{-1}$ exists. Accordingly, expressing $J(\mathbf{w})$ as a "perfect square" in $\mathbf{w}$, we may rewrite Eq. (2.50) in the equivalent form

$$J(\mathbf{w}) = \sigma_d^2 - \mathbf{p}^{\mathrm{H}}\mathbf{R}^{-1}\mathbf{p} + (\mathbf{w} - \mathbf{R}^{-1}\mathbf{p})^{\mathrm{H}}\mathbf{R}(\mathbf{w} - \mathbf{R}^{-1}\mathbf{p}). \tag{2.51}$$

From Eq. (2.51), we now immediately see that

$$\min_{\mathbf{w}} J(\mathbf{w}) = \sigma_d^2 - \mathbf{p}^{\mathrm{H}}\mathbf{R}^{-1}\mathbf{p}$$

for

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p}.$$

In effect, starting from Eq. (2.50), we have rederived the Wiener filter in a rather simple way. Moreover, we may use the defining equations for the Wiener filter to write

$$J(\mathbf{w}) = J_{\min} + (\mathbf{w} - \mathbf{w}_o)^{\mathrm{H}}\mathbf{R}(\mathbf{w} - \mathbf{w}_o). \tag{2.52}$$

This equation shows explicitly the unique optimality of the minimizing tap-weight vector $\mathbf{w}_o$, since we immediately see that $J(\mathbf{w}_o) = J_{\min}$.

Although the quadratic form on the right-hand side of Eq. (2.52) is quite informative, nevertheless it is desirable to change the basis on which it is defined so that the representation of the error-performance surface is considerably simplified. To this end,

we use *eigendecomposition* to express the correlation matrix $\mathbf{R}$ of the tap-input vector in terms of its *eigenvalues* and associated *eigenvectors* (see Appendix E)

$$\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{H}}, \tag{2.53}$$

where Λ is a diagonal matrix consisting of the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$ of the correlation matrix and the matrix $\mathbf{Q}$ has for its columns the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ associated with these eigenvalues, respectively. Hence, substituting Eq. (2.53) into Eq. (2.52), we get

$$J = J_{\min} + (\mathbf{w} - \mathbf{w}_o)^{\mathrm{H}}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{H}}(\mathbf{w} - \mathbf{w}_o). \tag{2.54}$$

If we next define a *transformed* version of the difference between the optimum solution $\mathbf{w}_o$ and the tap-weight vector $\mathbf{w}$ as

$$\mathbf{v} = \mathbf{Q}^{\mathrm{H}}(\mathbf{w}_o - \mathbf{w}), \tag{2.55}$$

then we may put the quadratic form of Eq. (2.54) into its *canonical form* defined by

$$J = J_{\min} + \mathbf{v}^{\mathrm{H}}\mathbf{\Lambda}\mathbf{v}. \tag{2.56}$$

This new formulation of the mean-square error contains no cross-product terms, as shown by

$$\begin{aligned} J &= J_{\min} + \sum_{k=1}^{M} \lambda_k v_k v_k^* \\ &= J_{\min} + \sum_{k=1}^{M} \lambda_k |v_k|^2, \end{aligned} \tag{2.57}$$

where v_k is the kth component of the vector $\mathbf{v}$. The feature that makes the canonical form of Eq. (2.57) a rather useful representation of the error-performance surface is the fact that the components of the transformed coefficient vector $\mathbf{v}$ constitute the *principal axes* of the surface. The practical significance of this result will become apparent in later chapters.

2.6 MULTIPLE LINEAR REGRESSION MODEL

The minimum mean-square error of the Wiener filter, denoted by $J_{\min}$ and defined in Eq. (2.49), applies to an FIR filter configuration of prescribed length (i.e., prescribed number of tap weights). Given the possibility of adjusting the length of the FIR filter as a design parameter, how can we decrease $J_{\min}$ down to an irreducible level? Clearly, the answer to this fundamental question depends on the model describing the relationship between the desired response $d(n)$ and the input vector $\mathbf{u}(n)$.

In what follows, we make three reasonable assumptions:

1. The model is linear.
2. The observable (measurable) data are *noisy*.
3. The noise is additive and white.

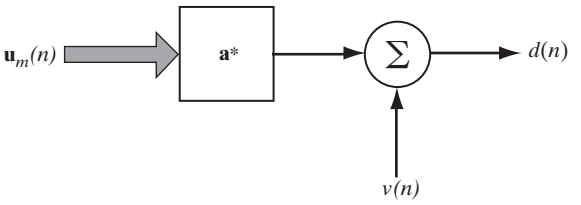


FIGURE 2.5 Multiple linear regression model.

Figure 2.5 shows a model that satisfies these assumptions. This model, called the *multiple linear regression model*, is described by the formula (Weisberg, 1980)

$$d(n) = \mathbf{a}^{\mathrm{H}}\mathbf{u}_m(n) + v(n), \tag{2.58}$$

where $\mathbf{a}$ denotes an *unknown* parameter vector of the model, $\mathbf{u}_m(n)$ denotes the input vector, or regressor, and $v(n)$ accounts for the additive white noise that is *statistically independent* of $\mathbf{u}_m(n)$; the parameter vector $\mathbf{a}$ is also referred to as the *regression vector*. So long as the stochastic environment under study is linear and the *model order m* is large enough, the underlying mechanism responsible for generating the observable data $d(n)$ in response to the input $\mathbf{u}_m(n)$ can be closely approximated by the multiple regression model of Eq. (2.58). The vectors $\mathbf{a}$ and $\mathbf{u}_m(n)$ are both of dimension m. Let σ_v^2 denote the variance of the noise $v(n)$. Then the variance of the observable data $d(n)$ supplying the desired response is given by

$$\begin{aligned}\sigma_d^2 &= \mathbb{E}[d(n)d^*(n)] \\ &= \sigma_v^2 + \mathbf{a}^{\mathrm{H}}\mathbf{R}_m\mathbf{a},\end{aligned} \tag{2.59}$$

where

$$\mathbf{R}_m = \mathbb{E}[\mathbf{u}_m(n)\mathbf{u}_m^{\mathrm{H}}(n)] \tag{2.60}$$

is the m-by-m correlation matrix of the input vector.

Consider now a Wiener filter that operates on an input vector $\mathbf{u}(n)$ and desired response $d(n)$ to produce a minimum mean-square error $J_{\min}(M)$, which is adjustable by varying the filter length M. The input vector $\mathbf{u}(n)$ and corresponding tap-weight vector $\mathbf{w}_o$ are both M-by-1 vectors. Substituting Eq. (2.59) into the first line of Eq. (2.49), we get

$$J_{\min}(M) = \sigma_v^2 + (\mathbf{a}^{\mathrm{H}}\mathbf{R}_m\mathbf{a} - \mathbf{w}_o^{\mathrm{H}}\mathbf{R}\mathbf{w}_o). \tag{2.61}$$

The only adjustable term in this expression is $\mathbf{w}_o^{\mathrm{H}}\mathbf{R}\mathbf{w}_o$, which is quadratic in M. We may identify three regimes of model selection:

1. *Underfitted model: $M < m$*

In this regime, the Wiener filter exhibits improved performance with increasing M for a prescribed m. In particular, the minimum mean-square error decreases quadratically with the filter length M, starting from the worst possible condition; that is,

$$J_{\min}(0) = \sigma_v^2 + \mathbf{a}^{\mathrm{H}}\mathbf{R}_m\mathbf{a}. \tag{2.62}$$

2. *Critically fitted model: $M = m$*

At the critical point $M = m$, the Wiener filter is perfectly matched to the regression model, in that $\mathbf{w}_o = \mathbf{a}$ and $\mathbf{R} = \mathbf{R}_m$. Correspondingly, the minimum mean-square error of the Wiener filter attains the *irreducible* value,[5] which is defined by

$$J_{\min}(0) = \sigma_v^2. \tag{2.63}$$

3. *Overfitted model: $M > m$*

When the length of the Wiener filter is greater than the model order m, the tail end of the tap-weight vector of the Wiener filter is zero, as shown by the formula

$$\mathbf{w}_o = \begin{bmatrix} \mathbf{a} \\ \mathbf{0} \end{bmatrix}. \tag{2.64}$$

Correspondingly, the tap-input vector of the Wiener filter takes the form

$$\mathbf{u}(n) = \begin{bmatrix} \mathbf{u}_m(n) \\ \mathbf{u}_{M-m}(n) \end{bmatrix}, \tag{2.65}$$

where $\mathbf{u}_{M-m}(n)$ is an $(M - m)$-by-1 vector made up of past data samples immediately preceding the m-by-1 vector $\mathbf{u}_m(n)$. The net result, in theory, is the same minimum mean-square-error performance as the critically fitted case, but with a longer filter length.

From this discussion, it is obvious that the preferred design strategy is to match the length M of the Wiener filter to the order m of the regression model. In this critical case, *the estimation error $e_o(n)$ produced by the Wiener filter is white with variance σ_v^2*, inheriting the statistical characterization of the additive noise $v(n)$ in the regression model of Eq. (2.58).

2.7 EXAMPLE

To illustrate the optimum filtering theory developed in the preceding sections, consider a regression model of order $m = 3$ with its parameter vector denoted by

$$\mathbf{a} = [a_o, a_1, a_2]^{\mathrm{T}}.$$

The statistical characterization of the model, assumed to be real valued, is as follows:

(a) The correlation matrix of the input vector $\mathbf{u}(n)$ is

$$\mathbf{R}_4 = \begin{bmatrix} 1.1 & 0.5 & 0.1 & -0.05 \\ 0.5 & 1.1 & 0.5 & 0.1 \\ 0.1 & 0.5 & 1.1 & 0.5 \\ -0.05 & 0.1 & 1.5 & 1.1 \end{bmatrix},$$

where the dashed lines are included to identify the submatrices that correspond to varying filter lengths.

[5]The term "irreducible error," as used here, is different from similar terminology used in the literature on optimum filtering. When we say "irreducible error," we mean the optimum estimation error produced by a realizable discrete-time Wiener filter whose length matches the order of the multiple regression model described in Eq. (2.58). On the other hand, in the literature (e.g., Thomas, 1969), the term "irreducible" is used to refer to an unrealizable Wiener filter, which is noncausal (i.e., its impulse response is nonzero for negative time).

(b) The cross-correlation vector between the input vector $\mathbf{u}(n)$ and observable data $d(n)$ is

$$\mathbf{p} = [0.5272, -0.4458, -0.1003, -0.0126]^{\mathrm{T}},$$

where the value of the fourth entry ensures that the model parameter a_3 is zero (i.e., the model order m is 3, as prescribed; see Problem 9).

(c) The variance of the observable data is

$$\sigma_d^2 = 0.9486.$$

(d) The variance of the additive white noise is

$$\sigma_v^2 = 0.1066.$$

The requirement is to do three things:

1. Investigate the variation of the minimum mean-square error $J_{\min}$ produced by a Wiener filter of varying length $M = 1, 2, 3, 4$.
2. Display the error-performance surface of a Wiener filter with length $M = 2$.
3. Compute the canonical form of the error-performance surface.

Variation of $J_{\min}$ with filter length M With model order $M = 3$, the real-valued regression model is described by

$$d(n) = a_o u(n) + a_1 u(n-1) + a_2 u(n-2) + v(n), \tag{2.66}$$

where $a_k = 0$ for all $k \geq 3$. Table 2.1 summarizes the computations of the M-by-1 optimum tap-weight vector and minimum mean-square error $J_{\min}(M)$ produced by the Wiener filter for $M = 1, 2, 3, 4$. The table also includes the pertinent values of the correlation matrix $\mathbf{R}$ and cross-correlation vector $\mathbf{p}$ that are used in Eqs. (2.36) and (2.49) to perform the computations.

Figure 2.6 displays the variation of the minimum mean-square error $J_{\min}(M)$ with the Wiener filter length M. The figure also includes the point corresponding to the worst

TABLE 2.1 Summary of Wiener Filter Computations for Varying Filter Length M

Filter length M	Correlation matrix $\mathbf{R}$	Cross-correlation vector $\mathbf{p}$	Optimum tap-weight vector $\mathbf{w}_o$	Minimum mean-square error $J_{\min}(M)$
1	[1.1]	[0.5272]	[0.4793]	0.6959
2	$\begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}$	$\begin{bmatrix} 0.5272 \\ -0.4458 \end{bmatrix}$	$\begin{bmatrix} 0.8360 \\ -0.7853 \end{bmatrix}$	0.1576
3	$\begin{bmatrix} 1.1 & 0.5 & 0.1 \\ 0.5 & 1.1 & 0.5 \\ 0.1 & 0.5 & 1.1 \end{bmatrix}$	$\begin{bmatrix} 0.5272 \\ -0.4458 \\ -0.1003 \end{bmatrix}$	$\begin{bmatrix} 0.8719 \\ -0.9127 \\ 0.2444 \end{bmatrix}$	0.1066
4	$\begin{bmatrix} 1.1 & 0.5 & 0.1 & -0.05 \\ 0.5 & 1.1 & 0.5 & 0.1 \\ 0.1 & 0.5 & 1.1 & 0.5 \\ -0.05 & 0.1 & 0.5 & 1.1 \end{bmatrix}$	$\begin{bmatrix} 0.5272 \\ -0.4458 \\ -0.1003 \\ -0.0126 \end{bmatrix}$	$\begin{bmatrix} 0.8719 \\ -0.9129 \\ 0.2444 \\ 0 \end{bmatrix}$	0.1066

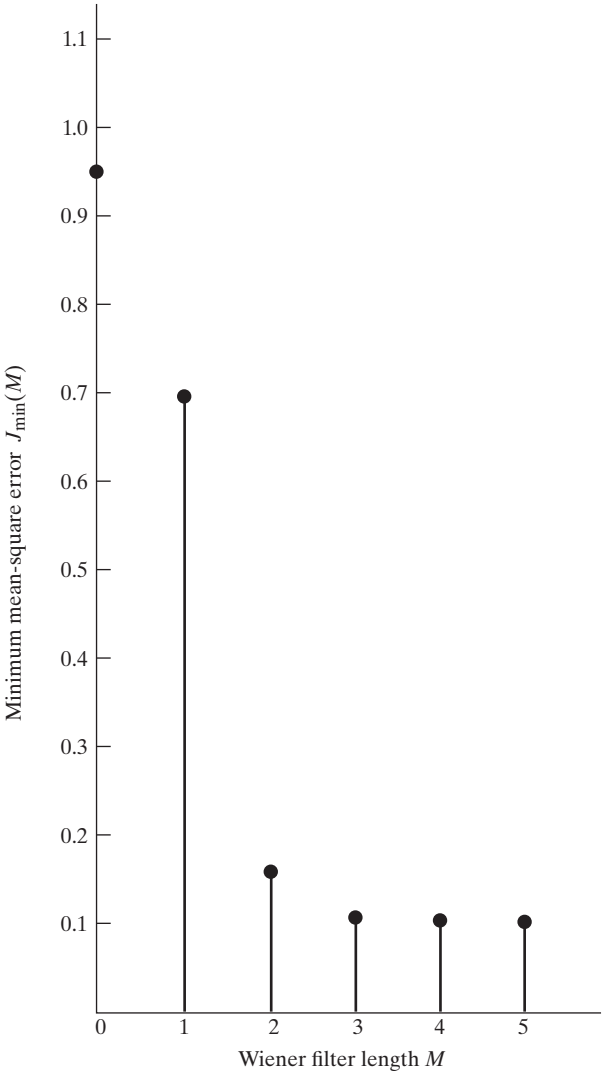


FIGURE 2.6 Variation of $J_{\min}(M)$ with Wiener filter length M for the example of Section 2.7.

possible condition $M = 0$ for which $J_{\min}(0) = \sigma_d^2$. Notice the steep drop in the minimum mean-square error when the filter length M is increased from one to two.

Error-performance surface For filter length $M = 2$, the dependence of the mean-square error on the 2-by-1 tap-weight vector $\mathbf{w}$ is defined [in accordance with Eq. (2.50)] as

$$\begin{aligned} J(w_0, w_1) &= 0.9486 - 2[0.5272, -0.4458]\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} + [w_0, w_1]\begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \\ &= 0.9486 - 1.0544\, w_0 + 0.8961\, w_1 + w_0 w_1 + 1.1(w_0^2 + w_1^2). \end{aligned} \tag{2.67}$$

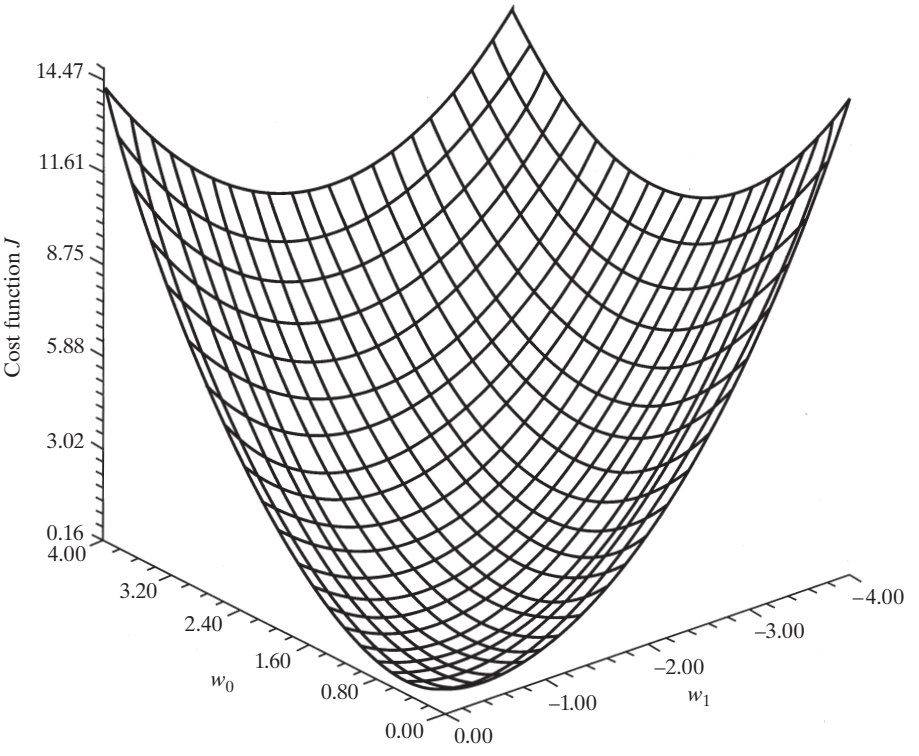


FIGURE 2.7 Error-performance surface of the two-tap FIR filter described in the example of Section 2.7.

Figure 2.7 shows a three-dimensional computer plot of the mean-square error $J(w_0, w_1)$ versus the tap weights w_0 and w_1.

Figure 2.8 shows contour plots of the tap weight w_1 versus w_0 for varying values of the mean-square error J. We see that the locus of w_1 versus w_0 for a fixed J is in the form of an ellipse. The elliptical locus shrinks in size as J approaches the minimum value $J_{\min}$. For $J = J_{\min}$, the locus reduces to a point with coordinates $w_{o0} = 0.8360$ and $w_{o1} = -0.7853$; that is, the optimum tap-weight vector is

$$\mathbf{w}_o = \begin{bmatrix} 0.8360 \\ -0.7853 \end{bmatrix}. \tag{2.68}$$

The minimum mean-square error is, in accordance with Eq. (2.49),

$$\begin{aligned} J_{\min} &= 0.9486 - [0.5272 \; -0.4458] \begin{bmatrix} 0.8360 \\ -0.7853 \end{bmatrix} \\ &= 0.1579. \end{aligned} \tag{2.69}$$

The point represented jointly by the optimum tap-weight vector $\mathbf{w}_o = [0.8360, -0.7853]^{\mathrm{T}}$ and the minimum mean-square error $J_{\min} = 0.1579$ defines the bottom of the error-performance surface in Fig. 2.7 or the center of the contour plots in Fig. 2.8.

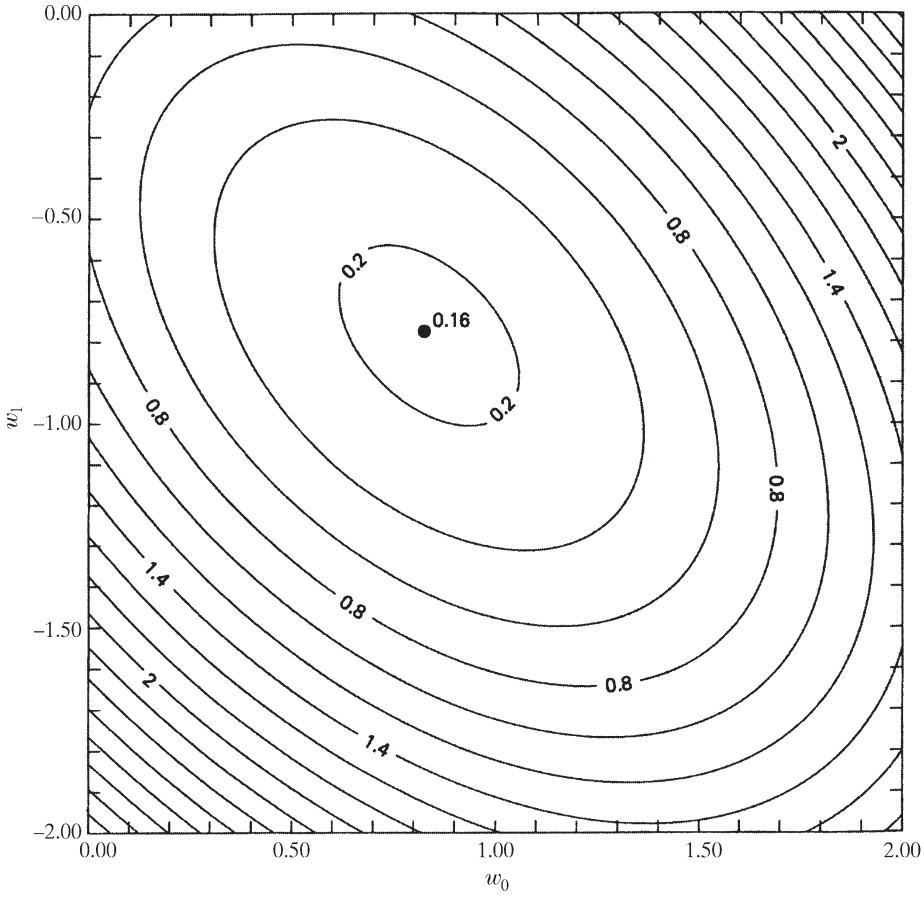


FIGURE 2.8 Contour plots of the error-performance surface depicted in Fig. 2.7.

Computation of the Canonical Form of the Error-Performance Surface

For eigenvalue analysis, we formulate the characteristic equation of the 2-by-2 correlation matrix

$$\mathbf{R} = \begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix},$$

which is given by the determinant of the matrix

$$\begin{bmatrix} 1.1 - \lambda & 0.5 \\ 0.5 & 1.1 - \lambda \end{bmatrix};$$

that is,

$$(1.1 - \lambda)^2 - (0.5)^2 = 0.$$

The two eigenvalues of the correlation matrix $\mathbf{R}$ are therefore

$$\lambda_1 = 1.6$$

and

$$\lambda_2 = 0.6$$

The canonical error-performance surface is defined [in accordance with Eq. (2.57)] as

$$J(v_1, v_2) = J_{\min} + 1.6\, v_1^2 + 0.6\, v_2^2. \tag{2.70}$$

The locus of v_2 versus v_1 traces an ellipse for a fixed value of $J - J_{\min}$. In particular, the ellipse has a minor axis of $[(J - J_{\min})/\lambda_1]^{1/2}$ along the v_1-coordinate and a major axis of $[(J - J_{\min})/\lambda_2]^{1/2}$ along the v_2-coordinate, where it is assumed that $\lambda_1 > \lambda_2$, which is in fact how they are related.

2.8 LINEARLY CONSTRAINED MINIMUM-VARIANCE FILTER

The essence of a Wiener filter is that it minimizes the mean-square value of an estimation error, defined as the difference between a desired response and the actual filter output. In solving this optimization (minimization) problem, *no* constraints are imposed on the solution. In some filtering applications, however, it may be desirable (or even mandatory) to design a filter that minimizes a mean-square criterion subject to a specific *constraint*. For example, the requirement may be that of minimizing the average output power of a linear filter while the response of the filter measured at some specific frequency of interest is constrained to remain constant. In this section, we examine one such solution for two different scenarios:

Scenario 1: Signal processing in the time domain.

Consider an FIR filter, as in Fig. 2.9. The filter output, in response to the tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$, is given by

$$y(n) = \sum_{k=0}^{M-1} w_k^* u(n-k). \tag{2.71}$$

For the special case of a *sinusoidal excitation*

$$u(n) = e^{j\omega n}, \tag{2.72}$$

we may rewrite Eq. (2.71) as

$$y(n) = e^{j\omega n} \sum_{k=0}^{M-1} w_k^* e^{-j\omega k}, \tag{2.73}$$

where ω is the angular frequency of the excitation, which is normalized with respect to the sampling rate; the summation is the *frequency response* of the filter.

The *constrained optimization problem* that we wish to solve may now be stated as follows:

> Find the optimum set of filter coefficients $w_{o0}, w_{o1}, \ldots, w_{o,M-1}$ that minimizes the mean-square value of the filter output $y(n)$, subject to the linear constraint
>
> $$\sum_{k=0}^{M-1} w_k^* e^{-j\omega_0 k} = g, \tag{2.74}$$
>
> where ω_0 is a prescribed value of the normalized angular frequency ω, lying inside the range $-\pi < \omega \leq \pi$, and g is a complex-valued gain.

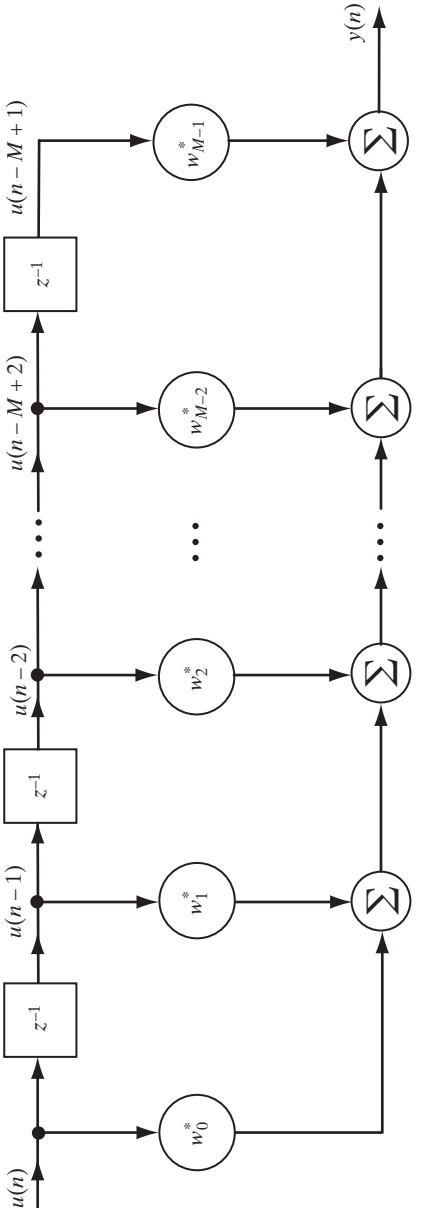


FIGURE 2.9 FIR filter.

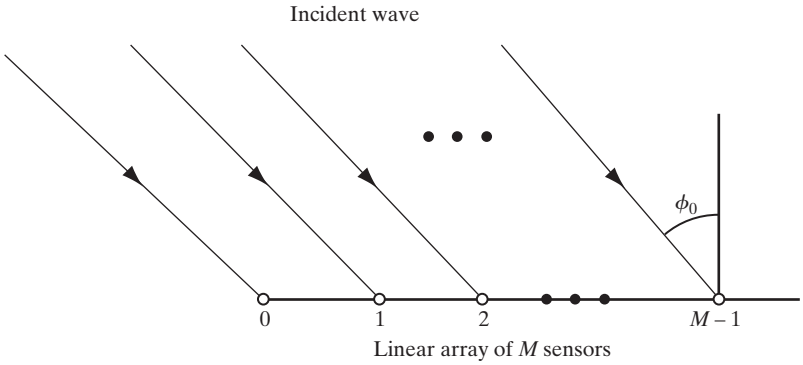


FIGURE 2.10 Plane wave incident on a linear-array antenna.

Scenario 2: Signal processing in the spatial domain.

The constrained optimization filtering problem described by Eqs. (2.71) and (2.74) is *temporal* in nature. We may formulate the *spatial* version of the problem by considering the *beamformer* depicted in Fig. 2.10, which consists of a linear array of uniformly spaced antenna elements with adjustable weights (not shown in the figure). The array is illuminated by an isotropic source located in the *far field* such that, at time n, a *plane wave* impinges on the array along a direction specified by the angle ϕ_0 with respect to the perpendicular to the array. It is also assumed that the interelement spacing of the array is less than $\lambda/2$, where λ is the wavelength of the transmitted signal, so as to satisfy the spatial analog of the sampling theorem. (See the discussion presented in Section 6 of the chapter on Background and Preview.) The resulting beamformer output is given by

$$y(n) = u_0(n) \sum_{k=0}^{M-1} w_k^* e^{-\mathrm{j}k\theta_0}, \tag{2.75}$$

where the *direction of arrival*, signified by the electrical angle θ_0, is related to the actual angle of incidence ϕ_0 ; $u_0(n)$ is the electrical signal picked up by the antenna element labeled 0 in Fig. 2.10 that is treated as the point of reference; and the w_k denotes the weights of the beamformer. The spatial version of the constrained optimization problem may thus be stated as follows:

> Find the optimum set of elemental weights $w_{o0}, w_{o1}, \ldots, w_{o,M-1}$ that minimizes the mean-square value of the beamformer output, subject to the linear constraint
>
> $$\sum_{k=0}^{M-1} w_k^* e^{-\mathrm{j}k\theta_0} = g, \tag{2.76}$$
>
> where θ_0 is a prescribed value of the electrical angle θ, lying inside the range $-\pi < \theta \leq \pi$, and g is a complex-valued gain. The beamformer is narrowband in the sense that its response needs to be constrained only at a single frequency.

Comparing the temporal structure of Fig. 2.9 and the spatial picture of Fig. 2.10, we see that, although they address entirely different physical situations, their formulations

are equivalent in mathematical terms. Indeed, in both cases we have exactly the same constrained optimization problem on our hands.

To solve the constrained optimization problem, we use the *method of Lagrange multipliers*, which is discussed in Appendix C. We begin by defining a *real-valued* cost function J that combines the two parts of the constrained optimization problem. Specifically, we write

$$J = \underbrace{\sum_{k=0}^{M-1}\sum_{i=0}^{M-1} w_k^* w_i r(i-k)}_{\text{output power}} + \underbrace{\text{Re}\left[\lambda^*\left(\sum_{k=0}^{M-1} w_k^* e^{-j\theta_0 k} - g\right)\right]}_{\text{linear constraint}}, \tag{2.77}$$

where λ is a *complex Lagrange multiplier*. Note that there is no desired response in the definition of J; rather, it includes a linear constraint that has to be satisfied for the prescribed electrical angle θ_0 in the context of beamforming, or, equivalently, the angular frequency ω_0 in FIR filtering. In any event, imposition of the linear constraint preserves the signal of interest, and minimization of the cost function J attenuates interference or noise, which can be troublesome if left unchecked.

Optimal Solution of the Beamforming Problem.

We wish to solve for the optimum values of the elemental weights of the beamformer that minimize J defined in Eq. (2.77). To do so, we may determine the gradient vector ∇J and then set it equal to zero. Thus, proceeding in a manner similar to that described in Section 2.2, we find that the kth element of the gradient vector ∇J is

$$\nabla_k J = 2\sum_{i=0}^{M-1} w_i r(i-k) + \lambda^* e^{-j\theta_0 k}. \tag{2.78}$$

Let w_{oi} be the ith element of the optimum weight vector $\mathbf{w}_o$. Then the condition for optimality of the beamformer is described by

$$\sum_{i=0}^{M-1} w_{oi} r(i-k) = -\frac{\lambda^*}{2} e^{-j\theta_0 k}, \qquad k = 0, 1, \ldots, M-1. \tag{2.79}$$

This system of M simultaneous equations defines the optimum values of the beamformer's elemental weights. It has a form somewhat similar to that of the Wiener–Hopf equations (2.28).

At this point in the analysis, we find it convenient to switch to matrix notation. In particular, we may rewrite the system of M simultaneous equations given in Eq. (2.79) simply as

$$\mathbf{R}\mathbf{w}_o = -\frac{\lambda^*}{2}\mathbf{s}(\theta_0), \tag{2.80}$$

where $\mathbf{R}$ is the M-by-M correlation matrix and $\mathbf{w}_o$ is the M-by-1 optimum weight vector of the constrained beamformer. The M-by-1 *steering* vector is defined by

$$\mathbf{s}(\theta_0) = [1, e^{-j\theta_0}, \ldots, e^{-j(M-1)\theta_0}]^T. \tag{2.81}$$

Solving Eq. (2.80) for $\mathbf{w}_o$, we thus have

$$\mathbf{w}_o = -\frac{\lambda^*}{2}\mathbf{R}^{-1}\mathbf{s}(\theta_0), \tag{2.82}$$

where $\mathbf{R}^{-1}$ is the inverse of the correlation matrix $\mathbf{R}$, assuming that $\mathbf{R}$ is nonsingular. This assumption is perfectly justified in practice by virtue of the fact that, in the context of a beamformer, the received signal at the output of each antenna element of the systems may include a white (thermal) noise component representing sensor noise.

The solution for the optimum weight vector $\mathbf{w}_o$ given in Eq. (2.82) is not quite complete, as it involves the unknown Lagrange multiplier λ (or its complex conjugate, to be precise). To eliminate λ^* from this expression, we first use the linear constraint of Eq. (2.76) to write

$$\mathbf{w}_o^{\mathrm{H}}\mathbf{s}(\theta_0) = g, \tag{2.83}$$

where the superscript H denotes *Hermitian transposition* (i.e., the operation of transposition combined with complex conjugation). Hence, taking the Hermitian transpose of both sides of Eq. (2.82), postmultiplying by $\mathbf{s}(\theta_0)$, and then using the linear constraint of Eq. (2.83), we get

$$\lambda = -\frac{2g}{\mathbf{s}^{\mathrm{H}}(\theta_0)\mathbf{R}^{-1}\mathbf{s}(\theta_0)}, \tag{2.84}$$

where we have used the fact that $\mathbf{R}^{-\mathrm{H}} = \mathbf{R}^{-1}$. The quadratic form $\mathbf{s}^{\mathrm{H}}(\theta_0)\mathbf{R}^{-1}\mathbf{s}(\theta_0)$ is real valued. Hence, substituting Eq. (2.84) into Eq. (2.82), we get the desired formula for the optimum weight vector:

$$\mathbf{w}_o = \frac{g^*\mathbf{R}^{-1}\mathbf{s}(\theta_0)}{\mathbf{s}^{\mathrm{H}}(\theta_0)\mathbf{R}^{-1}\mathbf{s}(\theta_0)}. \tag{2.85}$$

Note that by minimizing the output power subject to the linear constraint of Eq. (2.83), signals incident on the array along directions different from the prescribed value θ_0 tend to be attenuated.

For obvious reasons, a beamformer characterized by the weight vector $\mathbf{w}_o$ is referred to as a *linearly constrained minimum-variance* (LCMV) *beamformer*. For a zero-mean input and therefore zero-mean output, "minimum variance" and "minimum mean-square value" are indeed synonymous. Also, in light of what we said previously, the solution defined by Eq. (2.85) with ω_0 substituted for θ_0 may be referred to as an *LCMV filter*. Although the LCMV beamformer and LCMV filter are quite different in physical terms, their optimality is one and the same in mathematical terms.

Minimum-Variance Distortionless Response Beamformer

The complex constant g defines the response of an LCMV beamformer at the electrical angle θ_0. For the special case of $g = 1$, the optimum solution given in Eq. (2.85) reduces to

$$\mathbf{w}_o = \frac{\mathbf{R}^{-1}\mathbf{s}(\theta_0)}{\mathbf{s}^{\mathrm{H}}(\theta_0)\mathbf{R}^{-1}\mathbf{s}(\theta_0)}. \tag{2.86}$$

The response of the beamformer defined by Eq. (2.86) is constrained to equal unity at the electrical angle θ_0. In other words, this beamformer is constrained to produce a distortionless response along the look direction corresponding to θ_0.

Now, the minimum mean-square value (average power) of the optimum beamformer output may be expressed as the quadratic form

$$J_{\min} = \mathbf{w}_o^{\mathrm{H}}\mathbf{R}\mathbf{w}_o. \tag{2.87}$$

Hence, substituting Eq. (2.86) into Eq. (2.87) and simplifying terms, we get the result

$$J_{\min} = \frac{1}{\mathbf{s}^{\mathrm{H}}(\theta_0)\mathbf{R}^{-1}\mathbf{s}(\theta_0)}. \tag{2.88}$$

The optimum beamformer is constrained to pass the target signal with unit response, while at the same time minimizing the total output variance. This variance minimization process attenuates interference and noise not originating at the electrical angle θ_0. Hence, $J_{\min}$ represents an estimate of the variance of the signal impinging on the array along the direction corresponding to θ_0. We may generalize this result and obtain an estimate of variance as a function of direction by formulating $J_{\min}$ as a function of θ. In so doing, we obtain the MVDR (spatial) power spectrum defined as

$$S_{\mathrm{MVDR}}(\theta) = \frac{1}{\mathbf{s}^{\mathrm{H}}(\theta)\mathbf{R}^{-1}\mathbf{s}(\theta)}, \tag{2.89}$$

where

$$\mathbf{s}(\theta) = [1, e^{-\mathrm{j}\theta}, \ldots, e^{-\mathrm{j}\theta(M-1)}]^{\mathrm{T}}. \tag{2.90}$$

The M-by-1 vector $\mathbf{s}(\theta)$ is called a *spatial scanning vector* in the context of the beamforming environment of Fig. 2.10 and a *frequency scanning vector* with ω in place of θ for the FIR filter of Fig. 2.9. By definition, $S_{\mathrm{MVDR}}(\theta)$ or $S_{\mathrm{MVDR}}(\omega)$ has the dimension of power. Its dependence on the electrical angle θ at the beamformer input or the angular frequency ω at the FIR filter input therefore justifies referring to it as a power spectrum estimate. Indeed, it is commonly referred to as the *minimum-variance distortionless response* (MVDR) *spectrum*.[6] Note that at any ω in the temporal context, power due to other angular frequencies is minimized. Accordingly, the MVDR spectrum tends to have sharper peaks and higher resolution, compared with nonparametric (classical) methods based on the definition of the power spectrum that were discussed in Chapter 1.

2.9 GENERALIZED SIDELOBE CANCELLERS

Continuing with the discussion of the LCMV narrowband beamformer defined by the linear constraint of Eq. (2.76), we note that this constraint represents the inner product

$$\mathbf{w}^{\mathrm{H}}\mathbf{s}(\theta_0) = g,$$

in which $\mathbf{w}$ is the weight vector and $\mathbf{s}(\theta_0)$ is the steering vector pointing along the electrical angle θ_0. The steering vector is an M-by-1 vector, where M is the number of antenna elements in the beamformer. We may generalize the notion of a linear constraint by introducing *multiple linear constraints* defined by

$$\mathbf{C}^{\mathrm{H}}\mathbf{w} = \mathbf{g}. \tag{2.91}$$

[6]The formula given in Eq. (2.89) is credited to Capon (1969). It is also referred to in the literature as the *maximum-likelihood method (MLM)*. In reality, however, this formula has no bearing on the classical principle of maximum likelihood. The use of the terminology *MLM* for the formula is therefore not recommended.

The matrix $\mathbf{C}$ is termed the *constraint matrix*, and the vector $\mathbf{g}$, termed the *gain vector*, has constant elements. Assuming that there are L linear constraints, $\mathbf{C}$ is an M-by-L matrix and $\mathbf{g}$ is an L-by-1 vector; each column of the matrix $\mathbf{C}$ represents a single linear constraint. Furthermore, it is assumed that the constraint matrix $\mathbf{C}$ has linearly independent columns. For example, with

$$[\,s(\theta_0), s(\theta_1)\,]^{\mathrm{H}}\mathbf{w} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

the narrowband beamformer is constrained to preserve a signal of interest impinging on the array along the electrical angle θ_0 and, at the same time, to suppress an interference known to originate along the electrical angle θ_1.

Let the columns of an M-by-$(M - L)$ matrix $\mathbf{C}_a$ be defined as a basis for the *orthogonal complement* of the space spanned by the columns of matrix $\mathbf{C}$. Using the definition of an orthogonal complement, we may thus write

$$\mathbf{C}^{\mathrm{H}}\mathbf{C}_a = \mathbf{0}, \tag{2.92}$$

or, just as well,

$$\mathbf{C}_a^{\mathrm{H}}\mathbf{C} = \mathbf{0}. \tag{2.93}$$

The null matrix $\mathbf{0}$ in Eq. (2.92) is L by $(M - L)$, whereas in Eq. (2.93) it is $(M - L)$ by L; we naturally have $M > L$. We now define the M-by-M partitioned matrix

$$\mathbf{U} = [\,\mathbf{C} \;\vdots\; \mathbf{C}_a\,] \tag{2.94}$$

whose columns span the entire M-dimensional signal space. The inverse matrix $\mathbf{U}^{-1}$ exists by virtue of the fact that the determinant of matrix $\mathbf{U}$ is nonzero.

Next, let the M-by-1 weight vector of the beamformer be written in terms of the matrix $\mathbf{U}$ as

$$\mathbf{w} = \mathbf{U}\mathbf{q}. \tag{2.95}$$

Equivalently, the M-by-1 vector $\mathbf{q}$ is defined by

$$\mathbf{q} = \mathbf{U}^{-1}\mathbf{w}. \tag{2.96}$$

Let $\mathbf{q}$ be partitioned in a manner compatible with that in Eq. (2.94), as shown by

$$\mathbf{q} = \begin{bmatrix} \mathbf{v} \\ \cdots\cdots \\ -\mathbf{w}_a \end{bmatrix}, \tag{2.97}$$

where $\mathbf{v}$ is an L-by-1 vector and the $(M - L)$-by-1 vector $\mathbf{w}_a$ is that portion of the weight vector $\mathbf{w}$ that is not affected by the constraints. We may then use the definitions of Eqs. (2.94) and (2.97) in Eq. (2.95) to write

$$\begin{aligned} \mathbf{w} &= [\,\mathbf{C} \;\vdots\; \mathbf{C}_a\,] \begin{bmatrix} \mathbf{v} \\ \cdots\cdots \\ -\mathbf{w}_a \end{bmatrix} \\ &= \mathbf{C}\mathbf{v} - \mathbf{C}_a\mathbf{w}_a. \end{aligned} \tag{2.98}$$

We may now apply the multiple linear constraints of Eq. (2.91), obtaining

$$\mathbf{C}^{\mathrm{H}}\mathbf{C}\mathbf{v} - \mathbf{C}^{\mathrm{H}}\mathbf{C}_a\mathbf{w}_a = \mathbf{g}. \tag{2.99}$$

But, from Eq. (2.92), we know that $\mathbf{C}^{\mathrm{H}}\mathbf{C}_a$ is zero; hence, Eq. (2.99) reduces to

$$\mathbf{C}^{\mathrm{H}}\mathbf{C}\mathbf{v} = \mathbf{g}. \tag{2.100}$$

Solving for the vector $\mathbf{v}$, we thus get

$$\mathbf{v} = (\mathbf{C}^{\mathrm{H}}\mathbf{C})^{-1}\mathbf{g}, \tag{2.101}$$

which shows that the multiple linear constraints do not affect $\mathbf{w}_a$.

Next, we define a fixed beamformer component represented by

$$\mathbf{w}_q = \mathbf{C}\mathbf{v} = \mathbf{C}(\mathbf{C}^{\mathrm{H}}\mathbf{C})^{-1}\mathbf{g}, \tag{2.102}$$

which is orthogonal to the columns of matrix $\mathbf{C}_a$ by virtue of the property described in Eq. (2.93); the rationale for using the subscript q in $\mathbf{w}_q$ will become apparent later. From this definition, we may use Eq. (2.98) to express the overall weight vector of the beamformer as

$$\mathbf{w} = \mathbf{w}_q - \mathbf{C}_a\mathbf{w}_a. \tag{2.103}$$

Substituting Eq. (2.103) into Eq. (2.91) yields

$$\mathbf{C}^{\mathrm{H}}\mathbf{w}_q - \mathbf{C}^{\mathrm{H}}\mathbf{C}_a\mathbf{w}_a = \mathbf{g},$$

which, by virtue of Eq. (2.92), reduces to

$$\mathbf{C}^{\mathrm{H}}\mathbf{w}_q = \mathbf{g}. \tag{2.104}$$

Equation (2.104) shows that the weight vector $\mathbf{w}_q$ is that part of the weight vector $\mathbf{w}$ which satisfies the constraints. In contrast, the vector $\mathbf{w}_a$ is unaffected by the constraints; it therefore provides the degrees of freedom built into the design of the beamformer. Thus, in light of Eq. (2.103), the beamformer may be represented by the block diagram shown in Fig. 2.11(a). The beamformer described herein is referred to as a *generalized sidelobe canceller* (GSC).[7]

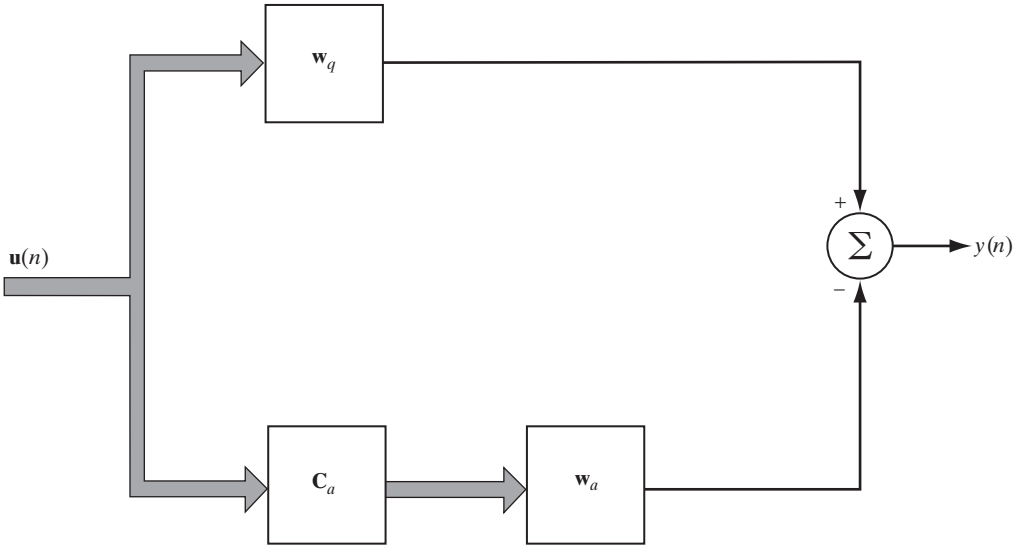
In light of Eq. (2.102), we may now perform an unconstrained minimization of the mean-square value of the beamformer output $y(n)$ with respect to the adjustable weight vector $\mathbf{w}_a$. According to Eq. (2.75), the beamformer output is defined by the inner product

$$y(n) = \mathbf{w}^{\mathrm{H}}\mathbf{u}(n), \tag{2.105}$$

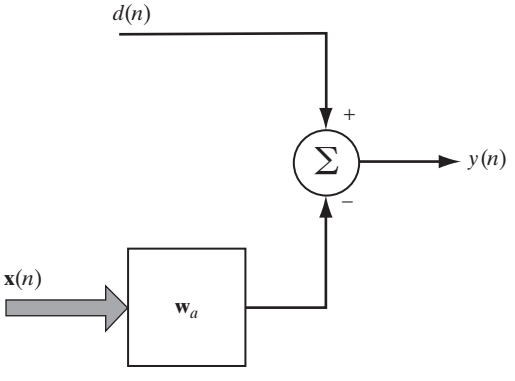
where

$$\mathbf{u}(n) = u_0(n)\,[1, e^{-\mathrm{j}\theta_0}, \ldots, e^{-\mathrm{j}(M-1)\theta_0}]^{\mathrm{T}} \tag{2.106}$$

[7]The essence of the GSC may be traced back to a method for solving linearly constrained quadratic minimization problems originally proposed by Hanson and Lawson (1969). The term "generalized sidelobe canceller" was coined by Griffiths and Jim (1982). For a discussion of this canceller, see Van Veen and Buckley (1988) and Van Veen (1992).



(a)



(b)

FIGURE 2.11 (a) Block diagram of generalized sidelobe canceller. (b) Reformulation of the generalized sidelobe cancelling problem as a standard optimum filtering problem.

is the input signal vector, in which the electrical angle θ_0 is defined by the direction of arrival of the incoming plane wave and $u_0(n)$ is the electrical signal picked up by antenna element 0 of the linear array in Fig. 2.10 at time n. Hence, substituting Eq. (2.103) into Eq. (2.105) yields

$$y(n) = \mathbf{w}_q^{\mathrm{H}}\mathbf{u}(n) - \mathbf{w}_a^{\mathrm{H}}\mathbf{C}_a^{\mathrm{H}}\mathbf{u}(n). \tag{2.107}$$

If we now define

$$\mathbf{w}_q^{\mathrm{H}}\mathbf{u}(n) = d(n) \tag{2.108}$$

and

$$\mathbf{C}_a^{\mathrm{H}}\mathbf{u}(n) = \mathbf{x}(n), \tag{2.109}$$

we may rewrite Eq. (2.107) in a form that resembles the standard Wiener filter exactly, as shown by

$$y(n) = d(n) - \mathbf{w}_a^{\mathrm{H}}\mathbf{x}(n), \tag{2.110}$$

where $d(n)$ plays the role of a "desired response" for the GSC and $\mathbf{x}(n)$ plays the role of input vector, as depicted in Fig. 2.11(b). We thus see that the combined use of vector $\mathbf{w}_q$ and matrix $\mathbf{C}_a$ has converted the linearly constrained optimization problem into a standard optimum filtering problem. In particular, we now have an unconstrained optimization problem involving the adjustable portion $\mathbf{w}_a$ of the weight vector, which may be formally written as

$$\min_{\mathbf{w}_a} \mathbb{E}[\,|y(n)|^2\,] = \min_{\mathbf{w}_a}(\sigma_d^2 - \mathbf{w}_a^{\mathrm{H}}\mathbf{p}_x - \mathbf{p}_x^{\mathrm{H}}\mathbf{w}_a + \mathbf{w}_a^{\mathrm{H}}\mathbf{R}_x\mathbf{w}_a), \tag{2.111}$$

where the $(M - L)$-by-1 cross-correlation vector

$$\mathbf{p}_x = \mathbb{E}[\mathbf{x}(n)d^*(n)\,] \tag{2.112}$$

and the $(M - L)$-by-$(M - L)$ correlation matrix

$$\mathbf{R}_x = \mathbb{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{H}}(n)\,]. \tag{2.113}$$

The cost function of Eq. (2.111) is quadratic in the unknown vector $\mathbf{w}_a$, which, as previously stated, embodies the available degrees of freedom in the GSC. Most importantly, this cost function has exactly the same mathematical form as that of the standard Wiener filter defined in Eq. (2.50). Accordingly, we may readily use our previous results to obtain the optimum value of $\mathbf{w}_a$ as

$$\mathbf{w}_{ao} = \mathbf{R}_x^{-1}\mathbf{p}_x. \tag{2.114}$$

Using the definitions of Eqs. (2.108) and (2.109) in Eq. (2.112), we may express the vector $\mathbf{p}_x$ as

$$\begin{aligned}\mathbf{p}_x &= \mathbb{E}[\mathbf{C}_a^{\mathrm{H}}\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{w}_q] \\ &= \mathbf{C}_a^{\mathrm{H}}\mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\,]\mathbf{w}_q \\ &= \mathbf{C}_a^{\mathrm{H}}\mathbf{R}\mathbf{w}_q,\end{aligned} \tag{2.115}$$

where $\mathbf{R}$ is the correlation matrix of the incoming data vector $\mathbf{u}(n)$. Similarly, using the definition of Eq. (2.109) in Eq. (2.113), we may express the matrix $\mathbf{R}_x$ as

$$\begin{aligned}\mathbf{R}_x &= \mathbb{E}[\mathbf{C}_a^{\mathrm{H}}\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{C}_a] \\ &= \mathbf{C}_a^{\mathrm{H}}\mathbf{R}\mathbf{C}_a.\end{aligned} \tag{2.116}$$

The matrix $\mathbf{C}_a$ has full rank, and the correlation matrix $\mathbf{R}$ is positive definite, since the incoming data always contain some form of additive sensor noise, with the result that $\mathbf{R}_x$ is nonsingular. Accordingly, we may rewrite the optimum solution of Eq. (2.114) as

$$\mathbf{w}_{ao} = (\mathbf{C}_a^{\mathrm{H}}\mathbf{R}\mathbf{C}_a)^{-1}\mathbf{C}_a^{\mathrm{H}}\mathbf{R}\mathbf{w}_q. \tag{2.117}$$

Let P_o denote the minimum output power of the GSC attained by using the optimum solution $\mathbf{w}_{ao}$. Then, adapting the previous result derived in Eq. (2.49) for the standard Wiener filter and proceeding in a manner similar to that just described, we may express P_o as

$$\begin{aligned} P_o &= \sigma_d^2 - \mathbf{p}_x^{\mathrm{H}}\mathbf{R}_x^{-1}\mathbf{p}_x \\ &= \mathbf{w}_q^{\mathrm{H}}\mathbf{R}\mathbf{w}_q - \mathbf{w}_q^{\mathrm{H}}\mathbf{R}\mathbf{C}_a(\mathbf{C}_a^{\mathrm{H}}\mathbf{R}\mathbf{C}_a)^{-1}\mathbf{C}_a^{\mathrm{H}}\mathbf{R}\mathbf{w}_q. \end{aligned} \tag{2.118}$$

Now consider the special case of a *quiet environment*, for which the received signal consists of white noise acting alone. Let the corresponding value of the correlation matrix $\mathbf{R}$ be written as

$$\mathbf{R} = \sigma^2\mathbf{I}, \tag{2.119}$$

where $\mathbf{I}$ is the M-by-M identity matrix and σ^2 is the noise variance. Under this condition, we readily find, from Eq. (2.117), that

$$\mathbf{w}_{ao} = (\mathbf{C}_a^{\mathrm{H}}\mathbf{C}_a)^{-1}\mathbf{C}_a^{\mathrm{H}}\mathbf{w}_q.$$

By definition, the weight vector $\mathbf{w}_q$ is orthogonal to the columns of matrix $\mathbf{C}_a$. It follows, therefore, that the optimum weight vector $\mathbf{w}_{ao}$ is identically zero for the quiet environment described by Eq. (2.119). Thus, with $\mathbf{w}_{ao}$ equal to zero, we find from Eq. (2.103) that $\mathbf{w} = \mathbf{w}_q$. It is for this reason that $\mathbf{w}_q$ is often referred to as the *quiescent weight vector*—hence the use of subscript q to denote it.

Filtering Interpretations of $\mathbf{w}_q$ and $\mathbf{C}_a$

The quiescent weight vector $\mathbf{w}_q$ and matrix $\mathbf{C}_a$ play critical roles of their own in the operation of the GSC. To develop physical interpretations of them, consider an MVDR spectrum estimator (formulated in temporal terms) for which we have

$$\begin{aligned} \mathbf{C} &= \mathbf{s}(\omega_0) \\ &= [1, e^{-\mathrm{j}\omega_0}, \ldots, e^{-\mathrm{j}(M-1)\omega_0}]^{\mathrm{T}} \end{aligned} \tag{2.120}$$

and

$$\mathbf{g} = 1.$$

Hence, the use of these values in Eq. (2.102) yields the corresponding value of the quiescent weight vector, viz.,

$$\begin{aligned} \mathbf{w}_q &= \mathbf{C}(\mathbf{C}^{\mathrm{H}}\mathbf{C})^{-1}\mathbf{g} \\ &= \frac{1}{M}[1, e^{-\mathrm{j}\omega_0}, \ldots, e^{-\mathrm{j}(M-1)\omega_0}]^{\mathrm{T}}, \end{aligned} \tag{2.121}$$

which represents an FIR filter of length M. The frequency response of this filter is given by

$$\begin{aligned}\mathbf{w}_q^{\mathrm{H}}\mathbf{s}(\omega) &= \frac{1}{M}\sum_{k=0}^{M-1} e^{\mathrm{j}k(\omega_0-\omega)} \\ &= \frac{1-e^{\mathrm{j}M(\omega_0-\omega)}}{1-e^{\mathrm{j}(\omega_0-\omega)}} \\ &= \left(\frac{\sin\left(\frac{M}{2}(\omega_0-\omega)\right)}{\sin\left(\frac{1}{2}(\omega_0-\omega)\right)}\right)\exp\left(\frac{\mathrm{j}(M-1)}{2}(\omega_0-\omega)\right).\end{aligned} \tag{2.122}$$

Figure 2.12(a) shows the amplitude response of this filter for $M = 4$ and $\omega_0 = 1$. From this figure, we clearly see that the FIR filter representing the quiescent weight vector $\mathbf{w}_q$ acts like a bandpass filter tuned to the angular frequency ω_0, for which the MVDR spectrum estimator is constrained to produce a distortionless response.

Consider next a physical interpretation of the matrix $\mathbf{C}_a$. The use of Eq. (2.120) in Eq. (2.92) yields

$$\mathbf{s}^{\mathrm{H}}(\omega_0)\mathbf{C}_a = \mathbf{0}. \tag{2.123}$$

According to Eq. (2.123), each of the $(M-L)$ columns of matrix $\mathbf{C}_a$ represents an FIR filter with an amplitude response that is zero at ω_0, as illustrated in Fig. 2.12(b) for $\omega_0 = 1$, $M = 4$, $L = 1$, and

$$\mathbf{C}_a = \begin{bmatrix} -1 & -1 & -1 \\ e^{-\mathrm{j}\omega_0} & 0 & 0 \\ 0 & e^{-\mathrm{j}2\omega_0} & 0 \\ 0 & 0 & e^{-\mathrm{j}3\omega_0} \end{bmatrix}.$$

In other words, the matrix $\mathbf{C}_a$ is represented by a bank of band-rejection filters, each of which is tuned to ω_0. Thus, $\mathbf{C}_a$ is referred to as a *signal-blocking matrix*, since it blocks (rejects) the received signal at the angular frequency ω_0. The function of the matrix $\mathbf{C}_a$ is to cancel interference that leaks through the sidelobes of the bandpass filter representing the quiescent weight vector $\mathbf{w}_q$.

2.10 SUMMARY AND DISCUSSION

The discrete-time version of Wiener filter theory, as described in this chapter, has evolved from the pioneering work of Norbert Wiener on linear optimum filters for continuous-time signals. The importance of the Wiener filter lies in the fact that it provides a frame of reference for the linear filtering of stochastic signals, assuming wide-sense stationarity.

The Wiener filter has two important properties from a practical perspective:

1. *Principle of orthogonality*: The error signal (estimation error) produced by the Wiener filter is orthogonal to its tap inputs.
2. *Statistical characterization of the error signal as white noise*: This condition is attained when the filter length matches the order of the multiple regression model describing the generation of the observable data (i.e., the desired response).

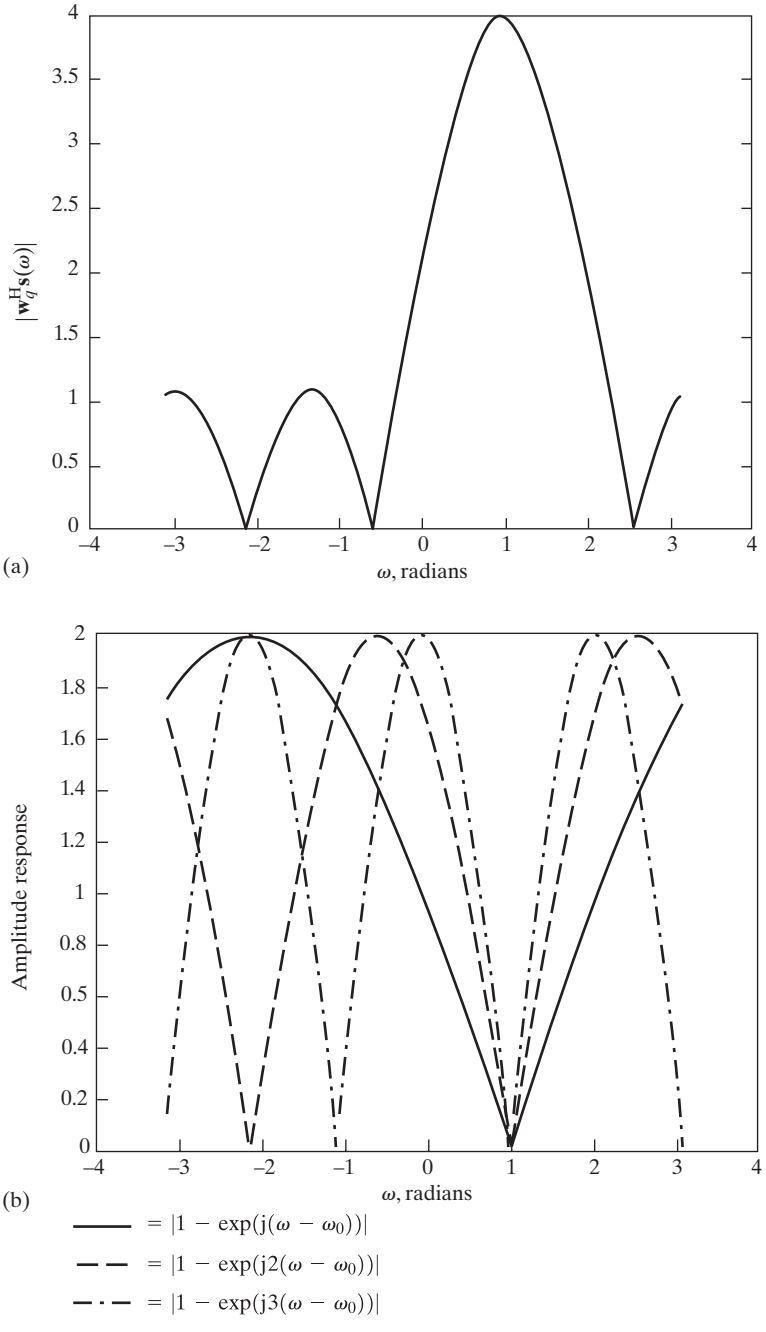


FIGURE 2.12 (a) Interpretation of $\mathbf{w}_q^H\mathbf{s}(\omega)$ as the response of an FIR filter. (b) Interpretation of each column of matrix $\mathbf{C}_a$ as a band-rejection filter. In both parts of the figure it is assumed that $\omega_0 = 1$.

The filtering structures that fall under the umbrella of Wiener filter theory are of two different physical types:

- FIR filters, which are characterized by an impulse response of finite duration.
- Narrowband beamformers, which consist of a finite set of uniformly spaced antenna elements with adjustable weights.

These two structures share a common feature: They are both examples of a linear system whose output is defined by the inner product of its weight vector and the input vector. The optimum filter involving such a structure is embodied in the Wiener–Hopf equations, the solution of which involves two ensemble-average parameters:

- The correlation matrix of the input vector.
- The cross-correlation vector between the input vector and desired response.

The standard formulation of Wiener filtering requires the availability of a desired response. There are, however, applications in which it is not feasible to provide such a response. For these applications, we may use a class of linear optimum filters known as linearly constrained minimum variance (LCMV) filters or LCMV beamformers, depending on whether the application is temporal or spatial in nature. The essence of the LCMV approach is that it minimizes the average output power, subject to a set of linear constraints on the weight vector. The constraints are imposed so as to prevent the weight vector from cancelling the signal of interest. To satisfy the requirement of multiple constraints, we may use the generalized sidelobe canceller (GSC) whose weight vector is separated into two components:

- A quiescent weight vector, which satisfies the prescribed constraints.
- An unconstrained weight vector, the optimization of which, in accordance with Wiener filter theory, minimizes the effects of receiver noise and interfering signals.

PROBLEMS

1. The correlation matrix **R** of the tap-input vector $u(n)$ of a Wiener filter is

$$R = \begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}$$

The cross-correlation vector between the tap-input vector $u(n)$ and the desired response $d(n)$ is

$$p = [0.5, 0.2]^T$$

Evaluate the tap weights of the filter.

2. The correlation matrix **R** of the tap-input vector $u(n)$ of a Wiener filter is

$$R = \begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}$$

The cross-correlation vector between the tap-input vector $u(n)$ and the desired response $d(n)$ is

$$p = [0.5, 0.2]^T$$

Evaluate the minimum mean-square error.

3. Repeat steps (a), (b), and (c) of Problem 2, given the following values for the correlation matrix $\mathbf{R}$ and cross-correlation vector $\mathbf{p}$:

$$\mathbf{R} = \begin{bmatrix} 1 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 1 \end{bmatrix};$$
$$\mathbf{p} = [0.5, 0.25, 0.125]^{\mathrm{T}}.$$

Note that these values require the use of a Wiener filter with three tap inputs.

4. Suppose you are given the two time series $u(0), u(1), \ldots, u(N)$ and $d(0), d(1), \ldots, d(N)$, both of which are realizations of two jointly wide-sense stationary processes. The series are used to supply the tap inputs of an FIR filter of length M and the desired response, respectively. Assuming that both of these processes are jointly ergodic, derive an estimate for the tap-weight vector of the Wiener filter by using time averages.

5. The tap-weight vector of an FIR filter is defined by

$$\mathbf{u}(n) = \alpha(n)\mathbf{s}(n) + \nu(n),$$

where

$$\mathbf{s}(\omega) = [1, e^{-\mathrm{j}\omega}, \ldots, e^{-\mathrm{j}\omega(M-1)}]^{\mathrm{T}}$$

and

$$\nu(n) = [\nu(n), \nu(n-1), \ldots, \nu(n-M+1)]^{\mathrm{T}}.$$

The complex amplitude of the sinusoidal vector $\mathbf{s}(\omega)$ is a random variable with zero mean and variance $\sigma_\alpha^2 = \mathbb{E}[|\alpha(n)|^2]$.

(a) Determine the correlation matrix of the tap-input vector $\mathbf{u}(n)$.

(b) Suppose that the desired response $d(n)$ is uncorrelated with $\mathbf{u}(n)$. What is the value of the tap-weight vector of the corresponding Wiener filter?

(c) Suppose that the variance σ_α^2 is zero and the desired response is defined by

$$d(n) = \nu(n-k),$$

where $0 \le k \le M-1$. What is the new value of the tap-weight vector of the Wiener filter?

(d) Determine the tap-weight vector of the Wiener filter for a desired response defined by

$$d(n) = \alpha(n)e^{-\mathrm{j}\omega\tau},$$

where τ is a prescribed delay.

6. Show that the Wiener–Hopf equations (2.34), defining the tap-weight vector $\mathbf{w}_o$ of the Wiener filter, and Eq. (2.49), defining the minimum mean-square error $J_{\min}$, may be combined into the single matrix relation

$$\mathbf{A}\begin{bmatrix} 1 \\ -\mathbf{w}_o \end{bmatrix} = \begin{bmatrix} J_{\min} \\ \mathbf{0} \end{bmatrix}.$$

The matrix $\mathbf{A}$ is the correlation matrix of the augmented vector

$$\begin{bmatrix} d(n) \\ \mathbf{u}(n) \end{bmatrix},$$

where $d(n)$ is the desired response and $\mathbf{u}(n)$ is the tap-input vector of the Wiener filter.

7. The minimum mean-square error is defined by [see Eq. (2.49)]

$$J_{\min} = \sigma_d^2 - \mathbf{p}^{\mathrm{H}}\mathbf{R}^{-1}\mathbf{p},$$

where σ_d^2 is the variance of the desired response $d(n)$, $\mathbf{R}$ is the correlation matrix of the tap-input vector $\mathbf{u}(n)$, and $\mathbf{p}$ is the cross-correlation vector between $\mathbf{u}(n)$ and $d(n)$. By applying eigendecompositions to the inverse of the correlation matrix, (i.e., $\mathbf{R}^{-1}$), show that

$$J_{\min} = \sigma_d^2 - \sum_{k=1}^{M} \frac{|\mathbf{q}_k^{\mathrm{H}}\mathbf{p}|^2}{\lambda_k},$$

where λ_k is the kth eigenvalue of the correlation matrix $\mathbf{R}$ and $\mathbf{q}_k$ is the corresponding eigenvector. Note that $\mathbf{q}_k^{\mathrm{H}}\mathbf{p}$ is a scalar.

8. Compute the canonical form of the error-performance surface for

$$R = \begin{bmatrix} 1. & 0.5 \\ 0.5 & 1 \end{bmatrix}.$$

9. Consider a linear regression model whose input–output behavior is defined by

$$d(n) = \mathbf{a}_M^{\mathrm{H}}\mathbf{u}_M(n) + \nu(n),$$

where

$$\mathbf{a}_M = [\,\mathbf{a}_m \,\vdots\, \mathbf{0}_{M-m}\,]^{\mathrm{T}}$$

and

$$\mathbf{u}_M(n) = [\,\mathbf{u}_m(n)\ \mathbf{u}_{M-m}(n-m)\,]^{\mathrm{T}}.$$

Assume that M is greater than the model order m. The correlation matrix of the input vector $\mathbf{u}_M(n)$ is partitioned as follows:

$$\mathbf{R}_M = \left[\begin{array}{c:c} \mathbf{R}_m & \mathbf{r}_{M-m} \\ \hdashline \mathbf{r}_{M-m}^{\mathrm{H}} & \mathbf{R}_{M-m,\ M-m} \end{array}\right].$$

The cross-correlation vector between the observable data $d(n)$ and the input vector $\mathbf{u}(n)$ is correspondingly partitioned as follows:

$$\mathbf{p}_M = \begin{bmatrix} \mathbf{p}_m \\ \mathbf{p}_{M-m} \end{bmatrix}.$$

(a) Find the condition that the $(M-m)$-by-1 vector $\mathbf{p}_{M-m}$ must satisfy for $\mathbf{a}_m$, $\mathbf{R}_m$, and $\mathbf{p}_m$ to satisfy the Wiener–Hopf equation.

(b) Applying the result derived in part (a) to the Example in Section 2.7, show that for $M = 4$, the last entry in the cross-correlation vector $\mathbf{p}$ has the value

$$p(3) = -0.0126.$$

10. The statistical characterization of a multiple linear regression model of order four is as follows:

- The correlation matrix of the input vector $\mathbf{u}(n)$ is

$$\mathbf{R}_4 = \begin{bmatrix} 1.1 & 0.5 & 0.1 & -0.1 \\ 0.5 & 1.1 & 0.5 & 0.1 \\ 0.1 & 0.5 & 1.1 & 0.5 \\ -0.1 & 0.1 & 0.5 & 1.1 \end{bmatrix}.$$

- The cross-correlation vector between the observable data and the input vector is

$$\mathbf{p}_4 = [0.5, -0.4, -0.2, -0.1]^{\mathrm{T}}.$$

- The variance of the observable data $d(n)$ is

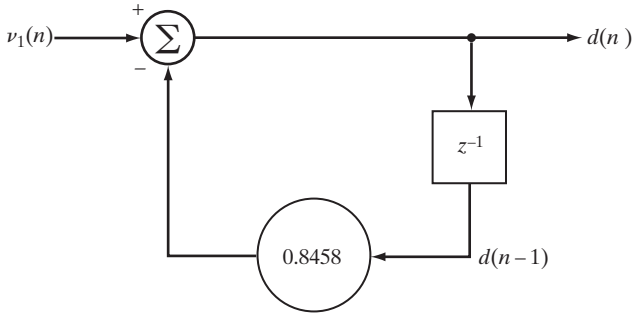
$$\sigma_d^2 = 1.0.$$

- The variance of the additive white noise is

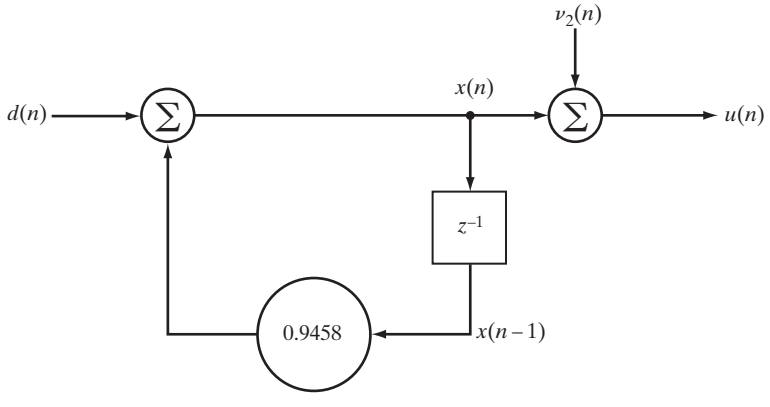
$$\sigma_v^2 = 0.1.$$

A Wiener filter of varying length M operates on the input vector $\mathbf{u}(n)$ as input and on the observable data $d(n)$ as the desired response. Compute and plot the mean-square error produced by the Wiener filter for $M = 0, 1, 2, 3, 4$.

11. Parts (a) and (b) of Fig. P2.1 show the autoregressive model of desired response $d(n)$ and the model of a noisy communication channel, respectively. In Fig. P2.1(a), $\nu_1(n)$ is a white-noise process of zero mean and variance $\sigma_1^2 = 0.27$. In Fig. P2.1(b), $\nu_2(n)$ is another white-noise source of zero mean and variance $\sigma_2^2 = 0.1$. The two noise sources $\nu_1(n)$ and $\nu_2(n)$ are statistically independent.



(a)



(b)

FIGURE P2.1

(a) Show that the channel output is

$$u(n) = x(n) + \nu_2(n),$$

where

$$x(n) = 0.1x(n-1) + 0.8x(n-2) + \nu_1(n).$$

(b) Assuming the use of a Wiener filter of length two, determine the correlation matrix $\mathbf{R}$ of the 2-by-1 tap-input vector and the cross-correlation vector between the tap-input vector and the desired response of the filter.

(c) Using the results of parts (a) and (b), determine the optimum weight vector of the Wiener filter and minimum mean-square error produced by the Wiener filter.

12. In this problem, we explore the extent of the improvement that may result from using a more complex Wiener filter for the environment described in Problem 11. To be specific, the new formulation of the Wiener filter has three taps.

(a) Find the 3-by-3 correlation matrix of the tap inputs of this filter and the 3-by-1 cross-correlation vector between the desired response and the tap inputs.

(b) Compute the 3-by-1 tap-weight vector of the Wiener filter, and also compute the new value for the minimum mean-square error.

13. In this problem, we explore an application of Wiener filtering to *radar*. The sampled form of the transmitted radar signal is $A_0 e^{j\omega_0 n}$ where ω_0 is the transmitted angular frequency and A_0 is the transmitted complex amplitude. The received signal is

$$u(n) = A_1 e^{-j\omega_1 n} + \nu(n),$$

where $|A_1| < |A_0|$ and ω_1 differs from ω_0 by virtue of the *Doppler* shift produced by the motion of a target of interest and $\nu(n)$ is a sample of white noise.

(a) Show that the correlation matrix of the time series $u(n)$, made up of M elements, may be written as

$$\mathbf{R} = \sigma_\nu^2 \mathbf{I} + \sigma_1^2 \mathbf{s}(\omega_1)\mathbf{s}^H(\omega_1),$$

where σ_ν^2 is the variance of the zero-mean white noise $\nu(n)$,

$$\sigma_1^2 = \mathbb{E}[|A_1|^2],$$

and

$$\mathbf{s}(\omega_1) = [1, e^{-j\omega_1}, \ldots, e^{-j\omega_1(M-1)}]^T.$$

(b) The time series $u(n)$ is applied to an M-tap Wiener filter, with the cross-correlation vector between $u(n)$ and a desired response $d(n)$ preset to

$$\mathbf{p} = \sigma_0^2 \mathbf{s}(\omega_0),$$

where

$$\sigma_0^2 = \mathbb{E}[|A_0|^2]$$

and

$$\mathbf{s}(\omega_0) = [1, e^{-j\omega_0}, \ldots, e^{-j\omega_0(M-1)}]^T.$$

Derive an expression for the tap-weight vector of the Wiener filter.

14. An array processor consists of a primary sensor and a reference sensor interconnected with each other. The output of the reference sensor is weighted by w and then subtracted from the output of the primary sensor. Show that the mean-square value of the output of the array processor is minimized when the weight w attains the optimum value

$$w_o = \frac{\mathbb{E}[u_1(n)u_2^*(n)]}{\mathbb{E}[|u_2(n)|^2]},$$

where $u_1(n)$ and $u_2(n)$ are, respectively, the primary- and reference-sensor outputs at time n.

15. Consider a discrete-time stochastic process $u(n)$ that consists of K (uncorrelated) complex sinusoids, plus additive white noise of zero mean and variance σ^2. That is,

$$u(n) = \sum_{k=1}^{K} A_k e^{j\omega_k n} + \nu(n),$$

where the terms $A_k \exp(j\omega_k n)$ and $\nu(n)$ refer to the kth sinusoid and noise, respectively. The process $u(n)$ is applied to an FIR filter with M taps, producing the output

$$e(n) = \mathbf{w}^H \mathbf{u}(n).$$

Assume that $M > K$. The requirement is to choose the tap-weight vector $\mathbf{w}$ so as to minimize the mean-square value of $e(n)$, subject to the multiple signal-protection constraint

$$\mathbf{S}^H \mathbf{w} = \mathbf{D}^{1/2}\mathbf{1},$$

where $\mathbf{S}$ is the M-by-K signal matrix whose kth column has $1, \exp(j\omega_k), \ldots, \exp[j\omega_k(M-1)]$ for its elements, $\mathbf{D}$ is the K-by-K diagonal matrix whose nonzero elements equal the average powers of the individual sinusoids, and the K-by-1 vector $\mathbf{1}$ has 1's for all its K elements. Using the method of Lagrange multipliers, show that the value of the optimum weight vector that results from this constrained optimization is

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{S}(\mathbf{S}^H\mathbf{R}^{-1}\mathbf{S})^{-1}\mathbf{D}^{1/2}\mathbf{1},$$

where $\mathbf{R}$ is the correlation matrix of the M-by-1 tap-input vector $\mathbf{u}(n)$. This formula represents a temporal generalization of the MVDR formula.

16. The weight vector $\mathbf{w}_o$ of the LCMV beamformer is defined by Eq. (2.85). In general, the LCMV beamformer so defined does not maximize the output signal-to-noise ratio. To be specific, let the input vector be written as

$$\mathbf{u}(n) = \mathbf{s}(n) + \boldsymbol{\nu}(n),$$

where the vector $\mathbf{s}(n)$ represents the signal component and the vector $\boldsymbol{\nu}(n)$ represents the additive noise component. Show that the weight vector $\mathbf{w}$ does *not* satisfy the condition

$$\max_{\mathbf{w}} \frac{\mathbf{w}^H\mathbf{R}_s\mathbf{w}}{\mathbf{w}^H\mathbf{R}_\nu\mathbf{w}},$$

where $\mathbf{R}_s$ is the correlation matrix of $\mathbf{s}(n)$ and $\mathbf{R}_\nu$ is the correlation matrix of $\boldsymbol{\nu}(n)$.

17. In this problem, we explore the design of constraints for a beamformer using a *nonuniformly spaced array* of antenna elements. Let t_i denote the propagation delay to the ith element for a plane wave impinging on the array from the actual look direction ϕ; the delay t_i is measured with respect to the zero-time reference.

(a) Find the response of the beamformer with elemental weight vector $\mathbf{w}$ to a signal of angular frequency ω that originates from the actual look direction ϕ.

(b) Hence, specify the linear constraint imposed on the array to produce a response equal to g along the direction ϕ.

18. Consider the problem of detecting a known signal in the presence of additive noise. The noise is assumed to be Gaussian, to be independent of the signal, and to have zero mean and a positive definite correlation matrix $\mathbf{R}_\nu$. The aim of the problem is to show that, under these conditions, the three criteria, namely, minimum mean-square error, maximum signal-to-noise ratio, and likelihood ratio test, yield identical designs for the FIR filter.

Let $u(n), n = 1, 2, \ldots, M$, denote a set of M complex-valued data samples. Let $\nu(n), n = 1, 2, \ldots, M$, denote a set of samples taken from a Gaussian noise process of zero mean. Finally, let $s(n), n = 1, 2, \ldots, M$, denote samples of the signal. The detection problem is to determine whether the input consists of the signal plus noise or noise alone. That is, the two hypotheses to be tested for are as follows:

$$\begin{aligned} &\text{hypothesis } \mathrm{H}_1\text{:} \quad u(n) = s(n) + \nu(n), \quad n = 1, 2, \ldots, M; \\ &\text{hypothesis } \mathrm{H}_0\text{:} \quad u(n) = \nu(n), \qquad\quad n = 1, 2, \ldots, M. \end{aligned}$$

(a) The *Wiener filter* minimizes the mean-square error. Show that this criterion yields an optimum tap-weight vector for estimating s_k, the kth component of signal vector $\mathbf{s}$, which equals

$$\mathbf{w}_o = \frac{s_k}{1 + \mathbf{s}^{\mathrm{H}}\mathbf{R}_\nu^{-1}\mathbf{s}}\,\mathbf{R}_\nu^{-1}\mathbf{s}.$$

(*Hint*: To evaluate the inverse of the correlation matrix of $\mathbf{u}(n)$ under hypothesis H_1, you may use the matrix inversion lemma.) Let

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^{\mathrm{H}},$$

where $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{D}$ are positive-definite matrices. Then

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^{\mathrm{H}}\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^{\mathrm{H}}\mathbf{B}.$$

(b) The *maximum signal-to-noise-ratio filter* maximizes the ratio

$$\begin{aligned} \rho &= \frac{\text{average power of filter output due to signal}}{\text{average power of filter output due to noise}} \\ &= \frac{\mathbb{E}[(\mathbf{w}^{\mathrm{H}}\mathbf{s})^2]}{\mathbb{E}[(\mathbf{w}^{\mathrm{H}}\nu)^2]}. \end{aligned}$$

Show that the tap-weight vector for which the output signal-to-noise ratio ρ is a maximum equals

$$\mathbf{w}_{SN} = \mathbf{R}_\nu^{-1}\mathbf{s}.$$

(*Hint*: Since $\mathbf{R}_\nu$ is positive definite, you may use $\mathbf{R}_\nu = \mathbf{R}_\nu^{1/2}\mathbf{R}_\nu^{1/2}$.)

(c) The *likelihood ratio processor* computes the log-likelihood ratio and compares it with a threshold. If the threshold is exceeded, it decides in favor of hypothesis H_1; otherwise, it decides in favor of the null hypothesis H_0. The likehood ratio is defined by

$$\Lambda = \frac{\mathbf{f}_{\mathbf{U}}(\mathbf{u} \mid \mathrm{H}_1)}{\mathbf{f}_{\mathbf{U}}(\mathbf{u} \mid \mathrm{H}_0)},$$

where $\mathbf{f}_{\mathbf{U}}(\mathbf{u}|\mathrm{H}_i)$ is the conditional joint probability density function of the observation vector $\mathbf{u}$, given that hypothesis H_i is true, where $i = 0, 1$. Show that the likelihood ratio test is equivalent to the test

$$\mathbf{w}_{ml}^{\mathrm{H}}\mathbf{u} \underset{\mathrm{H}_0}{\overset{\mathrm{H}_1}{\lessgtr}} \eta,$$

where η is the threshold and

$$\mathbf{w}_{ml} = \mathbf{R}_{\nu}^{-1}\mathbf{s}.$$

(*Hint*: Refer to Section 1.11 for the joint probability function of the M-by-1 Gaussian noise vector ν with zero mean and correlation matrix $\mathbf{R}_{\nu}$.)

19. The formulation of the Wiener–Hopf equations presented in Section 2.4 applies to a filter of infinitely long impulse response. However, this formulation is subject to the *causality constraint*, namely, that the impulse response of the filter is zero for negative time. In this problem, we extend the theory presented therein by removing the constraint; that is, we permit the filter to be *noncausal*.

(a) Let

$$S(z) = \sum_{k=-\infty}^{\infty} r(k)z^{-k}$$

denote the two-sided z-transform of the autocorrelation sequence of the tap inputs. Likewise, let

$$P(z) = \sum_{k=-\infty}^{\infty} p(k)z^{-k}$$

denote the two-sided z-transform of the cross-correlation between the tap inputs and desired response. (Note that for $z = e^{j\omega}$, these two formulas reduce to the definitions of power spectral densities.) Starting with Eq. (2.27) with the lower limit replaced by $i = -\infty$, show that the transfer function of the *unrealizable Wiener filter* is defined by

$$H_u(z) = \frac{P(1/z)}{S(z)},$$

where

$$H_u(z) = \sum_{k=-\infty}^{\infty} w_{u,k}z^{-k}.$$

(b) Suppose you are given

$$P(z) = \frac{0.36}{(1 - 0.2z^{-1})(1 - 0.2z)}$$

and

$$S(z) = \frac{1.37(1 - 0.146z^{-1})(1 - 0.146z)}{(1 - 0.2z^{-1})(1 - 0.2z)}.$$

Determine the transfer function of the unrealizable Wiener filter, and plot the impulse response of the filter.

(c) Assuming the use of a delay in the impulse response of the filter, suggest a suitable value for this delay for which the filter is essentially realizable.

CHAPTER 3

Linear Prediction

One of the most celebrated problems in time-series analysis is that of *predicting* the future value of a stationary discrete-time stochastic process, given a set of past samples of the process. To be specific, consider the time series $u(n), u(n-1), \ldots, u(n-M)$, representing $(M+1)$ samples of such a process up to and including time n. The operation of prediction may, for example, involve using the past samples $u(n-1), u(n-2), \ldots, u(n-M)$ to make an estimate of $u(n)$. Let $\mathcal{U}_{n-1}$ denote the M-dimensional space spanned by the samples $u(n-1), u(n-2), \ldots, u(n-M)$, and use $\hat{u}(n|\mathcal{U}_{n-1})$ to denote the *predicted value* of $u(n)$, given this set of samples. In *linear prediction*, we express this predicted value as a linear combination of the samples $u(n-1), u(n-2), \ldots, u(n-M)$. The operation corresponds to one-step prediction of the future, measured with respect to time $n-1$. Accordingly, we refer to this form of prediction as *one-step linear prediction in the forward direction* or, simply, *forward linear prediction.* In another form of prediction, we use the samples $u(n), u(n-1), \ldots, u(n-M+1)$ to make a prediction of the past sample $(n-M)$. We refer to this second form of prediction as *backward linear prediction*.[1]

In this chapter, we study forward linear prediction (FLP) as well as backward linear prediction (BLP). In particular, we use the Wiener filter theory of Chapter 2 to optimize the design of a forward or backward *predictor* in the mean-square-error sense for the case of a wide-sense stationary discrete-time stochastic process. As explained in that chapter, the correlation matrix of such a process has a Toeplitz structure. We will put this structure to good use in developing algorithms that are computationally efficient.

3.1 FORWARD LINEAR PREDICTION

Figure 3.1(a) shows a *forward predictor* that consists of a finite-duration impulse response (FIR) filter with M tap weights $w_{f,1}, w_{f,2}, \ldots, w_{f,M}$ and tap inputs $u(n-1), \ldots, u(n-M)$, respectively. We assume that these tap inputs are drawn from a wide-sense stationary stochastic process of zero mean. We further assume that the tap weights

[1]The term "backward prediction" is somewhat of a misnomer. A more appropriate description for this operation is "hindsight." Correspondingly, the use of "forward" in the associated operation of forward prediction is superfluous. Nevertheless, the terms "forward prediction" and "backward prediction" have become deeply embedded in the literature on linear prediction.

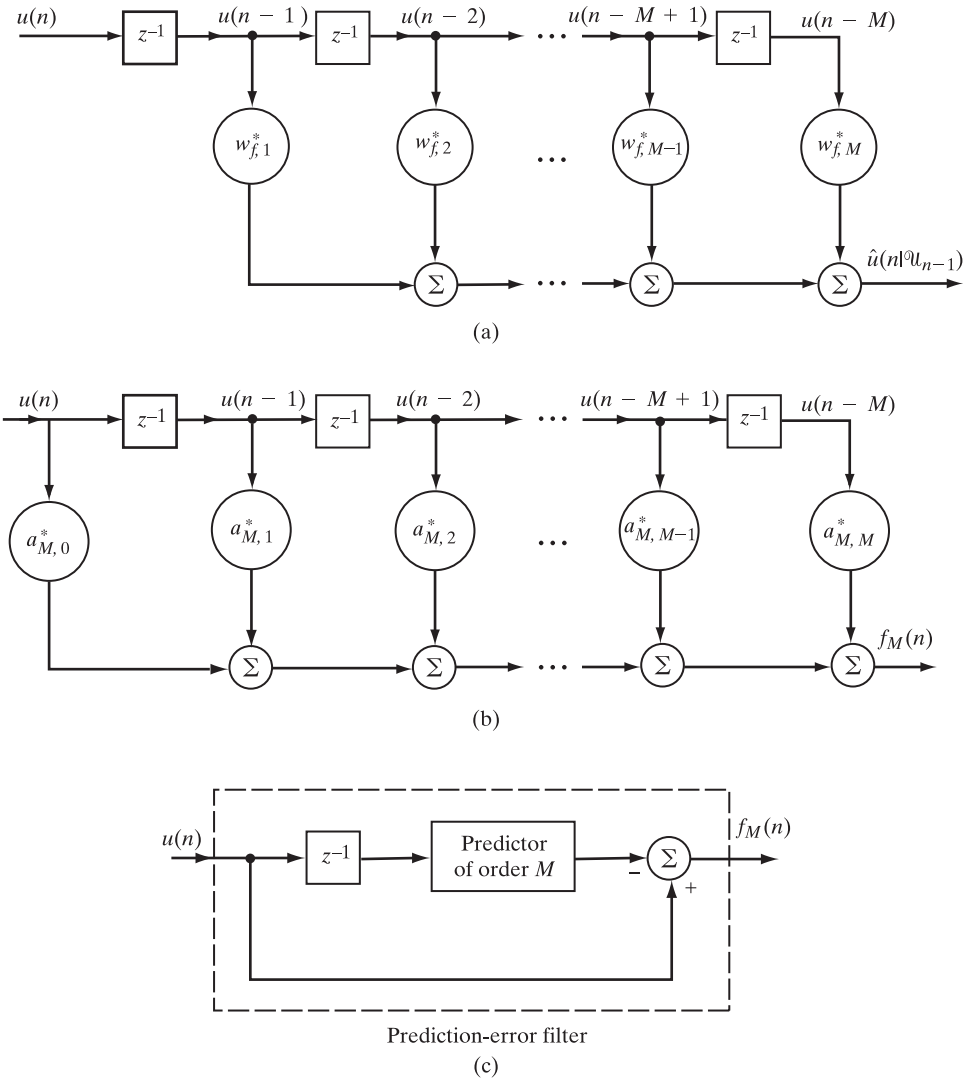


FIGURE 3.1 (a) One-step predictor; (b) prediction-error filter; (c) relationship between the predictor and the prediction-error filter.

are optimized in the mean-square-error sense in accordance with Wiener filter theory. The predicted value is

$$\hat{u}(n|\mathfrak{U}_{n-1}) = \sum_{k=1}^{M} w_{f,k}^{*} u(n-k), \tag{3.1}$$

where the asterisk indicates *complex conjugation.* For the situation described herein, the desired response $d(n)$ equals $u(n)$, representing the actual sample of the input process at time n. We may thus write

$$d(n) = u(n). \tag{3.2}$$

The *forward prediction error* equals the difference between the input sample $u(n)$ and its predicted value $\hat{u}(n|\mathcal{U}_{n-1})$. We denote the forward prediction error by $f_M(n)$ and thus write

$$f_M(n) = u(n) - \hat{u}(n|\mathcal{U}_{n-1}). \tag{3.3}$$

The subscript M in the symbol for the forward prediction error signifies the *order* of the predictor, defined as *the number of unit-delay elements needed to store the given set of samples used to make the prediction*. The reason for using the subscript will become apparent later in the chapter.

Let

$$P_M = \mathbb{E}[|f_M(n)|^2], \quad \text{for all } n, \tag{3.4}$$

denote the *minimum mean-square prediction error*. With the tap inputs assumed to have zero mean, the forward prediction error $f_M(n)$ will likewise have zero mean. Under this condition, P_M will also equal the variance of the forward prediction error. Yet another interpretation for P_M is that it may be viewed as the ensemble-average *forward prediction error power*, assuming that $f_M(n)$ is developed across a 1-Ω load. We shall use the latter description to refer to P_M.

Let $\mathbf{w}_f$ denote the M-by-1 optimum tap-weight vector of the forward predictor in Fig. 3.1(a). In expanded form,

$$\mathbf{w}_f = [w_{f,1}, w_{f,2}, \ldots, w_{f,M}]^{\mathrm{T}}, \tag{3.5}$$

where the superscript T denotes *transposition*. To solve the Wiener–Hopf equations for the weight vector $\mathbf{w}_f$, we require knowledge of two quantities: (1) the M-by-M correlation matrix of the tap inputs $u(n-1), u(n-2), \ldots, u(n-M)$ and (2) the M-by-1 cross-correlation vector between these tap inputs and the desired response $u(n)$. To evaluate P_M, we require a third quantity: the variance of $u(n)$. We now consider these three quantities, one by one:

1. The tap inputs $u(n-1), u(n-2), \ldots, u(n-M)$ define the M-by-1 tap-input vector

$$\mathbf{u}(n-1) = [u(n-1), u(n-2), \ldots, u(n-M)]^{\mathrm{T}}. \tag{3.6}$$

Hence, the correlation matrix of the tap inputs equals

$$\begin{aligned}\mathbf{R} &= \mathbb{E}[\mathbf{u}(n-1)\mathbf{u}^{\mathrm{H}}(n-1)] \\ &= \begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r^*(1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r^*(M-1) & r^*(M-2) & \cdots & r(0) \end{bmatrix},\end{aligned} \tag{3.7}$$

where $r(k)$ is the autocorrelation function of the input process for lag k, where lag $k = 0, 1, \ldots, M-1$ and the superscript H denotes *Hermitian transposition* (i.e., transposition combined with complex conjugation). Note that the symbol

used for the correlation matrix of the tap inputs in Fig. 3.1(a) is the same as that for the correlation matrix of the tap inputs in the Wiener filter. We are justified in using this symbol here, since the input process in both cases is assumed to be wide-sense stationary, so the correlation matrix of the process is invariant to a time shift.

2. The cross-correlation vector between the tap inputs $u(n-1), \ldots, u(n-M)$ and the desired response $u(n)$ is

$$\begin{aligned}\mathbf{r} &= \mathbb{E}[\mathbf{u}(n-1)u^*(n)] \\ &= \begin{bmatrix} r^*(1) \\ r^*(2) \\ \vdots \\ r^*(M) \end{bmatrix} = \begin{bmatrix} r(-1) \\ r(-2) \\ \vdots \\ r(-M) \end{bmatrix}.\end{aligned} \tag{3.8}$$

3. The variance of $u(n)$ equals $r(0)$, since $u(n)$ has zero mean.

In Table 3.1, we summarize the various quantities pertaining to the Wiener filter and the corresponding quantities pertaining to the forward predictor of Fig. 3.1(a). The last column of this table pertains to the backward predictor, about which more will be said later.

Thus, using the correspondences shown in the table, we may adapt the Wiener–Hopf equations (2.45) to solve the forward linear prediction (FLP) problem for stationary inputs and so write

$$\mathbf{R}\mathbf{w}_f = \mathbf{r}. \tag{3.9}$$

Similarly, the use of Eq. (2.49), together with Eq. (3.8), yields the following expression for the forward prediction-error power:

$$P_M = r(0) - \mathbf{r}^{\mathrm{H}}\mathbf{w}_f. \tag{3.10}$$

TABLE 3.1 Summary of Wiener Filter Variables

Quantity	Wiener filter	Forward predictor of Fig. 3.1(a)	Backward predictor of Fig. 3.2(a)
Tap-input vector	$\mathbf{u}(n)$	$\mathbf{u}(n-1)$	$\mathbf{u}(n)$
Desired response	$d(n)$	$u(n)$	$u(n-M)$
Tap-weight vector	$\mathbf{w}_o$	$\mathbf{w}_f$	$\mathbf{w}_b$
Estimation error	$e(n)$	$f_M(n)$	$b_M(n)$
Correlation matrix of tap inputs	$\mathbf{R}$	$\mathbf{R}$	$\mathbf{R}$
Cross-correlation vector between tap inputs and desired response	$\mathbf{p}$	$\mathbf{r}$	$\mathbf{r}^{\mathbf{B}*}$
Minimum mean-square error	$J_{\min}$	P_M	P_M

From Eqs. (3.8) and (3.9), we see that the M-by-1 tap-weight vector of the forward predictor and the forward prediction-error power are determined solely by the set of $(m + 1)$ autocorrelation function values of the input process for lags $0, 1, \ldots, M$.

Relation between Linear Prediction and Autoregressive Modeling

It is highly informative to compare the Wiener–Hopf equations (3.9) for linear prediction with the Yule–Walker equations (1.66) for an autoregressive (AR) model. We see that these two systems of simultaneous equations are of exactly the same mathematical form. Furthermore, Eq. (3.10) defining the average power (i.e., variance) of the forward prediction error is also of the same mathematical form as Eq. (1.71) defining the variance of the white-noise process used to excite the autoregressive model. For the case of an AR process for which we know the model order M, we may thus state that when a forward predictor is optimized in the mean-square-error sense, in theory, its tap weights take on the same values as the corresponding parameters of the process. This relationship should not be surprising, since the equation defining the forward prediction error and the difference equation defining the autoregressive model have the same mathematical form. When the process is not autoregressive, however, the use of a predictor provides an approximation to the process.

Forward Prediction-Error Filter

The forward predictor of Fig. 3.1(a) consists of M unit-delay elements and M tap weights $w_{f,1}, w_{f,2}, \ldots, w_{f,M}$ that are fed with the respective samples $u(n-1)$, $u(n-2), \ldots, u(n-M)$ as inputs. The resulting output is the predicted value of $u(n)$, which is defined by Eq. (3.1). Hence, substituting Eq. (3.1) into Eq. (3.3), we may express the forward prediction error as

$$f_M(n) = u(n) - \sum_{k=1}^{M} w_{f,k}^{*}\, u(n-k). \tag{3.11}$$

Let $a_{M,k}$, $k = 0, 1, \ldots, M$, denote the tap weights of a new FIR filter, which are related to the tap weights of the forward predictor as follows:

$$a_{M,k} = \begin{cases} 1, & k = 0 \\ -w_{f,k}, & k = 1, 2, \ldots, M. \end{cases} \tag{3.12}$$

Then we may combine the two terms on the right-hand side of Eq. (3.11) into the single summation

$$f_M(n) = \sum_{k=0}^{M} a_{M,k}^{*} u(n-k). \tag{3.13}$$

This input–output relation is represented by the FIR filter shown in Fig. 3.1(b). A filter that operates on the set of samples $u(n), u(n-1), u(n-2), \ldots, u(n-M)$ to produce the forward prediction error $f_M(n)$ at its output is called a *forward prediction-error filter*.

The relationship between the forward prediction-error filter and the forward predictor is illustrated in block diagram form in Fig. 3.1(c). Note that the length of the prediction-error filter exceeds the length of the one-step prediction filter by 1. However, both filters have the same order, M, as they both involve the same number of delay elements for the storage of past data.

Augmented Wiener–Hopf Equations for Forward Prediction

The Wiener–Hopf equations (3.9) define the tap-weight vector of the forward predictor, while Eq. (3.10) defines the resulting forward prediction-error power P_M. We may combine these two equations into the single matrix relation

$$\begin{bmatrix} r(0) & \mathbf{r}^{\mathrm{H}} \\ \mathbf{r} & \mathbf{R} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_f \end{bmatrix} = \begin{bmatrix} P_M \\ \mathbf{0} \end{bmatrix}, \tag{3.14}$$

where $\mathbf{0}$ is the M-by-1 null vector. The M-by-M correlation matrix $\mathbf{R}$ is defined in Eq. (3.7), and the M-by-1 correlation vector $\mathbf{r}$ is defined in Eq. (3.8). The partitioning of the $(M + 1)$-by-$(M + 1)$ correlation matrix on the left-hand side of Eq. (3.14) into the form shown therein was discussed in Section 1.3. Note that this $(M + 1)$-by-$(M + 1)$ matrix equals the correlation matrix of the tap inputs $u(n), u(n - 1), \ldots, u(n - M)$ in the prediction-error filter of Fig. 3.1(b). Moreover, the $(M + 1)$-by-1 coefficient vector on the left-hand side of Eq. (3.14) equals the *forward prediction-error filter vector*; that is,

$$\mathbf{a}_M = \begin{bmatrix} 1 \\ -\mathbf{w}_f \end{bmatrix}. \tag{3.15}$$

We may also express the matrix relation of Eq. (3.14) as the following system of $(M + 1)$ simultaneous equations:

$$\sum_{l=0}^{M} a_{M,l}\, r(l - i) = \begin{cases} P_M, & i = 0 \\ 0 & i = 1, 2, \ldots, M. \end{cases} \tag{3.16}$$

We refer to Eq. (3.14) or Eq. (3.16) as the *augmented Wiener–Hopf equations* of a forward prediction-error filter of order M.

EXAMPLE 1

For the case of a prediction-error filter of order $M = 1$, Eq. (3.14) yields a pair of simultaneous equations described by

$$\begin{bmatrix} r(0) & r(1) \\ r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} a_{1,0} \\ a_{1,1} \end{bmatrix} = \begin{bmatrix} P_1 \\ 0 \end{bmatrix}.$$

Solving for $a_{1,0}$ and $a_{1,1}$, we get

$$a_{1,0} = \frac{P_1}{\Delta_r}\, r(0)$$

and

$$a_{1,1} = -\frac{P_1}{\Delta_r}\, r^*(1),$$

where

$$\begin{aligned}\Delta_r &= \begin{vmatrix} r(0) & r(1) \\ r^*(1) & r(0) \end{vmatrix} \\ &= r^2(0) - |r(1)|^2\end{aligned}$$

is the determinant of the correlation matrix. But $a_{1,0}$ equals 1; hence,

$$P_1 = \frac{\Delta_r}{r(0)}$$

and

$$a_{1,1} = -\frac{r^*(1)}{r(0)}.$$

Consider next the case of a prediction-error filter of order $M = 2$. Equation (3.14) yields a system of three simultaneous equations:

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r^*(1) & r(0) & r(1) \\ r^*(2) & r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} a_{2,0} \\ a_{2,1} \\ a_{2,2} \end{bmatrix} = \begin{bmatrix} P_2 \\ 0 \\ 0 \end{bmatrix}.$$

Solving for $a_{2,0}$, $a_{2,1}$, and $a_{2,2}$, we get

$$\begin{aligned}a_{2,0} &= \frac{P_2}{\Delta_r}[r^2(0) - |r(1)|^2], \\ a_{2,1} &= -\frac{P_2}{\Delta_r}[r^*(1)r(0) - r(1)r^*(2)],\end{aligned}$$

and

$$a_{2,2} = -\frac{P_2}{\Delta_r}[(r^*(1))^2 - r(0)r^*(2)],$$

where

$$\Delta_r = \begin{vmatrix} r(0) & r(1) & r(2) \\ r^*(1) & r(0) & r(1) \\ r^*(2) & r^*(1) & r(0) \end{vmatrix}$$

is the determinant of the correlation matrix. The coefficient $a_{2,0}$ equals 1; accordingly, we may express the prediction-error power as

$$P_2 = \frac{\Delta_r}{r^2(0) - |r(1)|^2}$$

and the prediction-error filter coefficients as

$$a_{2,1} = \frac{r^*(1)r(0) - r(1)r^*(2)}{r^2(0) - |r(1)|^2}$$

and

$$a_{2,2} = \frac{(r^*(1))^2 - r(0)r^*(2)}{r^2(0) - |r(1)|^2}.$$

3.2 BACKWARD LINEAR PREDICTION

The form of linear prediction considered in Section 3.1 is said to be in the *forward* direction. That is, given the time series $u(n), u(n-1), \ldots, u(n-M)$, we use the subset of M past samples $u(n-1), u(n-2), \ldots, u(n-M)$ to make a prediction of the current sample $u(n)$. This operation corresponds to *one-step linear prediction into the future*, measured with respect to time $n-1$. Naturally, we may also operate on the same time series in the *backward* direction; that is, we may use the subset of M samples $u(n), u(n-1), \ldots, u(n-M+1)$ to make a prediction of the sample $u(n-M)$. This second operation corresponds to *backward linear prediction by one step*, measured with respect to time $n-M+1$.

Let $\mathcal{U}_n$ denote the M-dimensional space spanned by $u(n), u(n-1), \ldots, u(n-M+1)$, which are used in making the backward prediction. Then, using this set of samples as tap inputs, we make a linear prediction of the sample $u(n-M)$, as shown by

$$\hat{u}(n-M|\mathcal{U}_n) = \sum_{k=1}^{M} w_{b,k}^{*} u(n-k+1), \tag{3.17}$$

where $w_{b,1}, w_{b,2}, \ldots, w_{b,M}$ are the tap weights. Figure 3.2(a) shows a representation of the backward predictor as described by Eq. (3.17). We assume that these tap weights are optimized in the mean-square-error sense in accordance with Wiener filter theory.

In the case of backward prediction, the desired response is

$$d(n) = u(n-M). \tag{3.18}$$

The *backward prediction error* equals the difference between the actual sample value $u(n-M)$ and its predicted value $\hat{u}(n-M|\mathcal{U}_n)$. We denote the backward prediction error by $b_M(n)$ and thus write

$$b_M(n) = u(n-M) - \hat{u}(n-M|\mathcal{U}_n). \tag{3.19}$$

Here, again, the subscript M in the symbol for the backward prediction error $b_M(n)$ signifies the number of unit-delay elements needed to store the given set of samples used to make the prediction; that is, M is the order of the predictor.

Let

$$P_M = \mathbb{E}[\,|b_M(n)|^2], \qquad \text{for all } n, \tag{3.20}$$

denote the *minimum mean-square prediction error*. We may also view P_M as the ensemble-average *backward prediction-error power*, assuming that $b_M(n)$ is developed across a 1-Ω load.

Let $\mathbf{w}_b$ denote the M-by-1 optimum tap-weight vector of the backward predictor in Fig. 3.2(a). In expanded form,

$$\mathbf{w}_b = [w_{b,1}, w_{b,2}, \ldots, w_{b,M}]^{\mathrm{T}}. \tag{3.21}$$

To solve the Wiener–Hopf equations for the weight vector $\mathbf{w}_b$, we require knowledge of two quantities: (1) the M-by-M correlation matrix of the tap inputs $u(n)$, $u(n-1), \ldots, u(n-M+1)$ and (2) the M-by-1 cross-correlation vector between the

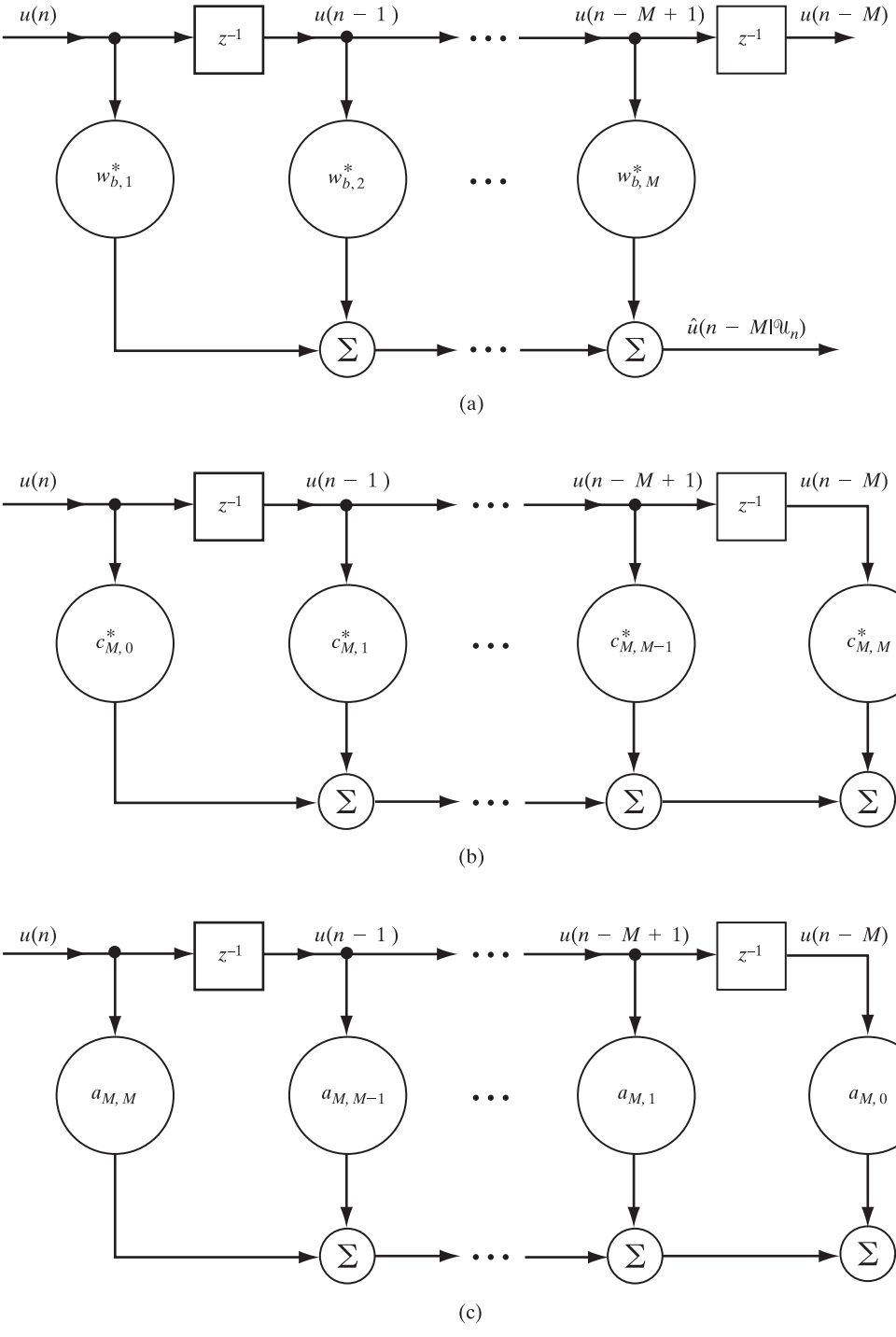


FIGURE 3.2 (a) Backward one-step predictor; (b) backward prediction-error filter; (c) backward prediction-error filter defined in terms of the tap weights of the corresponding forward prediction-error filter.

desired response $u(n - M)$ and these tap inputs. To evaluate P_M, we need a third quantity: the variance of $u(n - M)$. We consider these three quantities in turn:

1. Let $\mathbf{u}(n)$ denote the M-by-1 tap-input vector in the backward predictor of Fig. 3.2(a). In expanded form,

$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^{\mathrm{T}}. \tag{3.22}$$

   The M-by-M correlation matrix of the tap inputs in Fig. 3.2(a) is thus

$$\mathbf{R} = \mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)].$$

   The expanded form of the correlation matrix $\mathbf{R}$ is given in Eq. (3.7).

2. The M-by-1 cross-correlation vector between the tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$ and the desired response $u(n-M)$ is

$$\begin{aligned}\mathbf{r}^{\mathrm{B}*} &= \mathbb{E}[\mathbf{u}(n)u^*(n-M)] \\ &= \begin{bmatrix} r(M) \\ r(M-1) \\ \vdots \\ r(1) \end{bmatrix}.\end{aligned} \tag{3.23}$$

   The expanded form of the correlation vector $\mathbf{r}$ is given in Eq. (3.8). As usual, the superscript B denotes the backward arrangement.

3. The variance of the desired response $u(n - M)$ equals $r(0)$.

The last column of Table 3.1 summarizes the various quantities pertaining to the backward predictor of Fig. 3.2(a).

Accordingly, using the correspondences shown in the table, we may adapt the Wiener–Hopf equations (2.34) to solve the backward linear prediction (BLP) problem for stationary inputs and so write

$$\mathbf{R}\mathbf{w}_b = \mathbf{r}^{\mathrm{B}*}. \tag{3.24}$$

Similarly, the use of Eq. (2.49), together with Eq. (3.24), yields the following expression for the backward prediction-error power:

$$P_M = r(0) - \mathbf{r}^{\mathrm{BT}}\mathbf{w}_b. \tag{3.25}$$

Here, again, we see that the M-by-1 tap-weight vector $\mathbf{w}_b$ of a backward predictor and the backward prediction-error power P_M are uniquely defined by knowledge of the set of autocorrelation function values of the process for lags $0, 1, \ldots, M$.

Relations between Backward and Forward Predictors

In comparing the two sets of Wiener–Hopf equations (3.9) and (3.24) pertaining to forward prediction and backward prediction, respectively, we see that the vector on the right-hand side of Eq. (3.24) differs from that of Eq. (3.9) in two respects: (1) Its

elements are arranged backward, and (2) they are complex conjugated. To correct for the first difference, we reverse the order in which the elements of the vector on the right-hand side of Eq. (3.24) are arranged. This operation has the effect of replacing the left-hand side of Eq. (3.24) by $\mathbf{R}^{\mathrm{T}}\mathbf{w}_b^{\mathrm{B}}$, where $\mathbf{R}^{\mathrm{T}}$ is the transpose of the correlation matrix $\mathbf{R}$ and $\mathbf{w}_b^{\mathrm{B}}$ is the backward version of the tap-weight vector $\mathbf{w}_b$. (See Problem 3.) We may thus write

$$\mathbf{R}^{\mathrm{T}}\mathbf{w}_b^{\mathrm{B}} = \mathbf{r}^*. \tag{3.26}$$

To correct for the remaining difference, we complex-conjugate both sides of Eq. (3.26), obtaining

$$\mathbf{R}^{\mathrm{H}}\mathbf{w}_b^{\mathrm{B}*} = \mathbf{r}.$$

Since the correlation matrix $\mathbf{R}$ is Hermitian (i.e., $\mathbf{R}^{\mathrm{H}} = \mathbf{R}$), we may reformulate the Wiener–Hopf equations for backward prediction as

$$\mathbf{R}\mathbf{w}_b^{\mathrm{B}*} = \mathbf{r}. \tag{3.27}$$

Now we compare Eq. (3.27) with Eq. (3.9) and thus deduce the following fundamental relationship between the tap-weight vectors of a backward predictor and the corresponding forward predictor:

$$\mathbf{w}_b^{\mathrm{B}*} = \mathbf{w}_f. \tag{3.28}$$

Equation (3.28) states that *we may modify a backward predictor into a forward predictor by reversing the sequence in which its tap weights are positioned and also taking the complex conjugates of them.*

Next, we wish to show that the ensemble-average error powers for backward prediction and forward prediction have exactly the same value. To do this, we first observe that the product $\mathbf{r}^{\mathrm{BT}}\mathbf{w}_b$ equals $\mathbf{r}^{\mathrm{T}}\mathbf{w}_b^{\mathrm{B}}$, so we may rewrite Eq. (3.25) as

$$P_M = r(0) - \mathbf{r}^{\mathrm{T}}\mathbf{w}_b^{\mathrm{B}}. \tag{3.29}$$

Taking the complex conjugate of both sides of Eq. (3.29) and recognizing that both P_M and $r(0)$ are unaffected by this operation, since they are real-valued scalars, we get

$$P_M = r(0) - \mathbf{r}^{\mathrm{H}}\mathbf{w}_b^{\mathrm{B}*}. \tag{3.30}$$

Comparing this result with Eq. (3.10) and using the equivalence of Eq. (3.28), we find that the backward prediction-error power has exactly the same value as the forward prediction-error power. Indeed, it is in anticipation of this equality that we have used the same symbol P_M to denote both quantities. Note, however, that this equality holds only for linear prediction applied to a wide-sense stationary process.

Backward Prediction-Error Filter

The backward prediction error $b_M(n)$ equals the difference between the desired response $u(n - M)$ and the linear prediction of that response, given the samples $u(n), u(n-1), \ldots, u(n - M + 1)$. This prediction is defined by Eq. (3.17). Therefore, substituting that equation into Eq. (3.19), we get

$$b_M(n) = u(n - M) - \sum_{k=1}^{M} w_{bk}^* u(n - k + 1). \tag{3.31}$$

Now we define the tap weights of the backward prediction-error filter in terms of the corresponding backward predictor as follows:

$$c_{M,k} = \begin{cases} -w_{b,k+1}, & k = 0, 1, \ldots, M-1. \\ 1, & k = M \end{cases} \tag{3.32}$$

Hence, we may rewrite Eq. (3.31) as [see Fig. 3.2(b)]

$$b_M(n) = \sum_{k=0}^{M} c^*_{M,k} u(n-k). \tag{3.33}$$

Equation (3.28) defines the tap-weight vector of the backward predictor in terms of that of the forward predictor. We may express the scalar version of this relation as

$$w^*_{b,M-k+1} = w_{f,k}, \qquad k = 1, 2, \ldots, M,$$

or, equivalently,

$$w_{b,k} = w^*_{f,M-k+1}, \qquad k = 1, 2, \ldots, M. \tag{3.34}$$

Hence, substituting Eq. (3.34) into Eq. (3.32), we get

$$c_{M,k} = \begin{cases} -w^*_{f,M-k}, & k = 0, 1, \ldots, M-1 \\ 1, & k = 0 \end{cases}. \tag{3.35}$$

Thus, using the relationship between the tap weights of the forward prediction-error filter and those of the forward predictor as given in Eq. (3.12), we may write

$$c_{M,k} = a^*_{M,M-k}, \qquad k = 0, 1, \ldots, M. \tag{3.36}$$

Accordingly, we may express the input–output relation of the backward prediction-error filter in the equivalent form

$$b_M(n) = \sum_{k=0}^{M} a_{M,M-k} u(n-k). \tag{3.37}$$

The input–output relation of Eq. (3.37) is depicted in Fig. 3.2(c). A comparison of this representation for a backward prediction-error filter with that of Fig. 3.1(b) for the corresponding forward prediction-error filter reveals that these two forms of a prediction-error filter for *stationary inputs* are uniquely related to each other. In particular, *we may change a forward prediction-error filter into the corresponding backward prediction-error filter by reversing the sequence in which the tap weights are positioned and taking the complex conjugate of them.* Note that in both figures the respective tap inputs have the same values.

Augmented Wiener–Hopf Equations for Backward Prediction

The set of Wiener–Hopf equations for backward prediction is defined by Eq. (3.24), and the resultant backward prediction-error power is defined by Eq. (3.25). We may combine these two equations into the single relation

$$\begin{bmatrix} \mathbf{R} & \mathbf{r}^{B*} \\ \mathbf{r}^{BT} & r(0) \end{bmatrix} \begin{bmatrix} -\mathbf{w}_b \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ P_M \end{bmatrix}, \tag{3.38}$$

where $\mathbf{0}$ is the M-by-1 null vector. The M-by-M matrix $\mathbf{R}$ is the correlation matrix of the M-by-1 tap-input vector $\mathbf{u}(n)$; it has the expanded form shown in the second line of Eq. (3.7) by virtue of the assumed wide-sense stationarity of the input process. The M-by-1 vector $\mathbf{r}^{B*}$ is the cross-correlation vector between the input vector $\mathbf{u}(n)$ and the desired response $u(n - M)$; here, again, the assumed wide-sense stationarity of the input process means that the vector $\mathbf{r}$ has the expanded form shown in the second line of Eq. (3.8). The $(M + 1)$-by-$(M + 1)$ matrix on the left-hand side of Eq. (3.38) equals the correlation matrix of the tap inputs in the backward prediction-error filter of Fig. 3.2(c). The partitioning of this $(M + 1)$-by-$(M + 1)$ matrix into the form shown in Eq. (3.38) was discussed in Section 1.3.

We may also express the matrix relation of Eq. (3.38) as a system of $(M + 1)$ simultaneous equations:

$$\sum_{l=0}^{M} a_{M,M-l}^{*} r(l - i) = \begin{cases} 0, & i = 0, \ldots, M - 1 \\ P_M, & i = M \end{cases}. \tag{3.39}$$

We refer to Eq. (3.38) or (3.39) as the *augmented Wiener–Hopf equations* of a backward prediction-error filter of order M.

Note that in the matrix form of the augmented Wiener–Hopf equations for backward prediction defined by Eq. (3.38), the correlation matrix of the tap inputs is equivalent to that in the corresponding Eq. (3.14). This is merely a restatement of the fact that the tap inputs in the backward prediction-error filter of Fig. 3.2(c) are exactly the same as those in the forward prediction-error filter of Fig. 3.1(b).

3.3 LEVINSON–DURBIN ALGORITHM

We now describe a direct method for computing the prediction-error filter coefficients and prediction-error power by solving the augmented Wiener–Hopf equations. The method is recursive in nature and makes particular use of the Toeplitz structure of the correlation matrix of the tap inputs of the filter. It is known as the *Levinson–Durbin algorithm*, so named in recognition of its use first by Levinson (1947) and then its independent reformulation at a later date by Durbin (1960). Basically, the procedure utilizes the solution of the augmented Wiener–Hopf equations for a prediction-error filter of order $m - 1$ to compute the corresponding solution for a prediction-error filter of order m (i.e., one order higher). The order $m = 1, 2, \ldots, M$, where M is the *final* order of the filter. The important virtue of the Levinson–Durbin algorithm is its computational efficiency, in that its use results in a big saving in the number of operations (multiplications or divisions) and storage locations compared with standard methods such as the Gauss elimination method (Makhoul, 1975). To derive the Levinson–Durbin recursive procedure, we will use the matrix formulation of both forward and backward predictions in an elegant way (Burg, 1968, 1975).

Let the $(m + 1)$-by-1 vector $\mathbf{a}_m$ denote the tap-weight vector of a forward prediction-error filter of order m. The $(m + 1)$-by-1 tap-weight vector of the corresponding backward prediction-error filter is obtained by backward rearrangement of the elements of vector $\mathbf{a}_m$ and their complex conjugation. We denote the combined effect of these two operations by $\mathbf{a}_m^{B*}$. Let the m-by-1 vectors $\mathbf{a}_{m-1}$ and $\mathbf{a}_{m-1}^{B*}$ denote the tap-weight

vectors of the corresponding forward and backward prediction-error filters of order $m-1$, respectively. The Levinson–Durbin recursion may be stated in one of two equivalent ways:

1. The tap-weight vector of a *forward* prediction-error filter may be order-updated by the equation

$$\mathbf{a}_m = \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} + \kappa_m \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix}, \tag{3.40}$$

where κ_m is a constant. The scalar version of this *order update* is

$$a_{m,l} = a_{m-1,l} + \kappa_m a_{m-1,m-l}^*, \qquad l = 0, 1, \ldots, m, \tag{3.41}$$

where $a_{m,l}$ is the lth tap weight of a backward prediction-error filter of order m, and likewise for $a_{m-1,l}^*$. The element $a_{m-1,m-l}^*$ is the lth tap weight of a backward prediction-error filter of order $m-1$. In Eq. (3.41), note that $a_{m-1,0} = 1$ and $a_{m-1,m} = 0$.

2. The tap-weight vector of a *backward* prediction-error filter may be order-updated by the equation

$$\mathbf{a}_m^{B*} = \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} + \kappa_m^* \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix}. \tag{3.42}$$

The scalar version of this order update is

$$a_{m,m-l}^* = a_{m-1,m-l}^* + \kappa_m^* a_{m-1,l}, \qquad l = 0, 1, \ldots, m, \tag{3.43}$$

where $a_{m,m-l}^*$ is the lth tap weight of the backward prediction-error filter of order m and the other elements are as defined previously.

The Levinson–Durbin recursion is usually formulated in the context of forward prediction, in vector form as in Eq. (3.40) or scalar form as in Eq. (3.41). The formulation of the recursion in the context of backward prediction, in vector form as in Eq. (3.42) or scalar form as in Eq. (3.43), follows directly from that of Eqs. (3.40) and (3.41), respectively, through a combination of backward rearrangement and complex conjugation. (See Problem 8.)

To establish the condition that the constant κ_m has to satisfy in order to justify the validity of the Levinson–Durbin algorithm, we proceed in four stages:

1. We premultiply both sides of Eq. (3.40) by $\mathbf{R}_{m+1}$, the $(m+1)$-by-$(m+1)$ correlation matrix of the tap inputs $u(n), u(n-1), \ldots, u(n-M)$ in the forward prediction-error filter of order m. For the left-hand side of Eq. (3.40), we thus get [see Eq. (3.14) for comparison]

$$\mathbf{R}_{m+1}\mathbf{a}_m = \begin{bmatrix} P_m \\ \mathbf{0}_m \end{bmatrix}, \tag{3.44}$$

where P_m is the forward prediction-error power and $\mathbf{0}_m$ is the m-by-1 null vector. The subscripts in the matrix $\mathbf{R}_{m+1}$ and the vector $\mathbf{0}_m$ refer to their dimensions, whereas the subscripts in the vector $\mathbf{a}_m$ and the scalar P_m refer to the prediction order.

2. For the first term on the right-hand side of Eq. (3.40), we use the following partitioned form of the correlation matrix $\mathbf{R}_{m+1}$ [see Eq. (1.32) for comparison]:

$$\mathbf{R}_{m+1} = \begin{bmatrix} \mathbf{R}_m & \mathbf{r}_m^{\mathrm{B}*} \\ \mathbf{r}_m^{\mathrm{BT}} & r(0) \end{bmatrix}.$$

Here, $\mathbf{R}_m$ is the m-by-m correlation matrix of the tap inputs $u(n), u(n-1), \ldots, u(n-m+1)$, and $\mathbf{r}_m^{\mathrm{B}*}$ is the cross-correlation vector between these tap inputs and $u(n-m)$. We may thus write

$$\begin{aligned} \mathbf{R}_{m+1}\begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} &= \begin{bmatrix} \mathbf{R}_m & \mathbf{r}_m^{\mathrm{B}*} \\ \mathbf{r}_m^{\mathrm{BT}} & r(0) \end{bmatrix} \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_m \mathbf{a}_{m-1} \\ \mathbf{r}_m^{\mathrm{BT}} \mathbf{a}_{m-1} \end{bmatrix}. \end{aligned} \tag{3.45}$$

The set of augmented Wiener–Hopf equations for the forward prediction-error filter of order $m-1$ is

$$\mathbf{R}_m \mathbf{a}_{m-1} = \begin{bmatrix} P_{m-1} \\ \mathbf{0}_{m-1} \end{bmatrix}, \tag{3.46}$$

where P_{m-1} is the prediction-error power for this filter and $\mathbf{0}_{m-1}$ is the $(m-1)$-by-1 null vector. We next define the scalar

$$\begin{aligned} \Delta_{m-1} &= \mathbf{r}_m^{\mathrm{BT}} \mathbf{a}_{m-1} \\ &= \sum_{l=0}^{m-1} r(l-m)\, a_{m-1,l}. \end{aligned} \tag{3.47}$$

Substituting Eqs. (3.46) and (3.47) into Eq. (3.45), we may therefore write

$$\mathbf{R}_{m+1}\begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} = \begin{bmatrix} P_{m-1} \\ \mathbf{0}_{m-1} \\ \Delta_{m-1} \end{bmatrix}. \tag{3.48}$$

3. For the second term on the right-hand side of Eq. (3.40), we use the following partitioned form of the correlation matrix $\mathbf{R}_{m+1}$:

$$\mathbf{R}_{m+1} = \begin{bmatrix} r(0) & \mathbf{r}_m^{\mathrm{H}} \\ \mathbf{r}_m & \mathbf{R}_m \end{bmatrix}.$$

In this equation, $\mathbf{R}_m$ is the m-by-m correlation matrix of the tap inputs $u(n-1)$, $u(n-2), \ldots, u(n-m)$, and $\mathbf{r}_m$ is the m-by-1 cross-correlation vector between these tap inputs and $u(n)$. We may thus write

$$\begin{aligned} \mathbf{R}_{m+1}\begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{\mathrm{B}*} \end{bmatrix} &= \begin{bmatrix} r(0) & \mathbf{r}_m^{\mathrm{H}} \\ \mathbf{r}_m & \mathbf{R}_m \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{\mathrm{B}*} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{r}_m^{\mathrm{H}} \mathbf{a}_{m-1}^{\mathrm{B}*} \\ \mathbf{R}_m \mathbf{a}_{m-1}^{\mathrm{B}*} \end{bmatrix}. \end{aligned} \tag{3.49}$$

The scalar

$$\begin{aligned}\mathbf{r}_m^{\mathrm{H}}\mathbf{a}_{m-1}^{\mathrm{B}*} &= \sum_{k=1}^{m} r^*(-k)a_{m-1,\,m-k}^* \\ &= \sum_{l=0}^{m-1} r^*(l-m)a_{m-1,\,l}^* \\ &= \Delta_{m-1}^*.\end{aligned} \tag{3.50}$$

Also, the set of augmented Wiener–Hopf equations for the backward prediction-error filter of order $m-1$ is

$$\mathbf{R}_m\mathbf{a}_{m-1}^{\mathrm{B}*} = \begin{bmatrix}\mathbf{0}_{m-1}\\ P_{m-1}\end{bmatrix}. \tag{3.51}$$

Substituting Eqs. (3.50) and (3.51) into Eq. (3.49), we may therefore write

$$\mathbf{R}_{m+1}\begin{bmatrix}0\\ \mathbf{a}_{m-1}^{\mathrm{B}*}\end{bmatrix} = \begin{bmatrix}\Delta_{m-1}^*\\ \mathbf{0}_{m-1}\\ P_{m-1}\end{bmatrix}. \tag{3.52}$$

4. Summarizing the results obtained in stages 1, 2, and 3 and, in particular, using Eqs. (3.44), (3.48), and (3.52), we now see that the premultiplication of both sides of Eq. (3.40) by the correlation matrix $\mathbf{R}_{m+1}$ yields

$$\begin{bmatrix}P_m\\ \mathbf{0}_m\end{bmatrix} = \begin{bmatrix}P_{m-1}\\ \mathbf{0}_{m-1}\\ \Delta_{m-1}\end{bmatrix} + \kappa_m\begin{bmatrix}\Delta_{m-1}^*\\ \mathbf{0}_{m-1}\\ P_{m-1}\end{bmatrix}. \tag{3.53}$$

We conclude, therefore, that if the order-update recursion of Eq. (3.40) holds, the results described by Eq. (3.53) are direct consequences of that recursion. Conversely, if the conditions described by Eq. (3.53) apply, the tap-weight vector of a forward prediction-error filter may be order-updated as in Eq. (3.40).

From Eq. (3.53), we may make two important deductions:

1. By considering the first elements of the vectors on the left- and right-hand sides of Eq. (3.53), we have

$$P_m = P_{m-1} + \kappa_m\Delta_{m-1}^*. \tag{3.54}$$

2. By considering the last elements of the vectors on the left- and right-hand sides of Eq. (3.53), we have

$$0 = \Delta_{m-1} + \kappa_m P_{m-1}. \tag{3.55}$$

From Eq. (3.55), we see that the constant

$$\kappa_m = -\frac{\Delta_{m-1}}{P_{m-1}}, \tag{3.56}$$

where Δ_{m-1} is itself defined by Eq. (3.47). Furthermore, eliminating Δ_{m-1} between Eqs. (3.54) and (3.55), we get the following relation for the order update of the prediction-error power:

$$P_m = P_{m-1}(1 - |\kappa_m|^2). \tag{3.57}$$

As the order m of the prediction-error filter increases, the corresponding value of the prediction-error power P_m normally decreases or else remains the same. Of course, P_m can never be negative; hence, we always have

$$0 \leq P_m \leq P_{m-1}, \qquad m \geq 1. \tag{3.58}$$

For the elementary case of a prediction-error filter of order zero, we naturally have

$$P_0 = r(0),$$

where $r(0)$ is the autocorrelation function of the input process for zero lag.

Starting with $m = 0$ and increasing the filter order by one at a time, we find that, through the repeated application of Eq. (3.57), the prediction-error power for a prediction-error filter of *final* order M equals

$$P_M = P_0 \prod_{m=1}^{M} (1 - |\kappa_m|^2). \tag{3.59}$$

Interpretations of the Parameters κ_m and Δ_{m-1}

The parameters $\kappa_m, 1 \leq m \leq M$, resulting from the application of the Levinson–Durbin recursion to a prediction-error filter of final order M are called *reflection coefficients*. The use of this term comes from the analogy of Eq. (3.57) with transmission-line theory, where (in the latter context) κ_m may be considered as the reflection coefficient at the boundary between two sections with different characteristic impedances. Note that the condition on the reflection coefficient corresponding to that of Eq. (3.58) is

$$|\kappa_m| \leq 1, \quad \text{for all } m.$$

From Eq. (3.41), we see that for a prediction-error filter of order m, the reflection coefficient κ_m equals the *last* tap-weight $a_{m,m}$ of the filter. That is,

$$\kappa_m = a_{m,m}.$$

As for the parameter Δ_{m-1}, it may be interpreted as a cross-correlation between the forward prediction error $f_{m-1}(n)$ and the delayed backward prediction error $b_{m-1}(n-1)$. Specifically, we may write (see Problem 9)

$$\Delta_{m-1} = \mathbb{E}[b_{m-1}(n-1) f^*_{m-1}(n)], \tag{3.60}$$

where $f_{m-1}(n)$ is produced at the output of a forward prediction-error filter of order $m-1$ in response to the tap inputs $u(n), u(n-1), \ldots, u(n-m+1)$ and $b_{m-1}(n-1)$ is produced at the output of a backward prediction-error filter of order $m-1$ in response to the tap inputs $u(n-1), u(n-2), \ldots, u(n-m)$.

Note that

$$f_0(n) = b_0(n) = u(n),$$

where $u(n)$ is the prediction-error filter input at time n. Accordingly, from Eq. (3.60), we find that this cross-correlation parameter has the *zero-order value*

$$\begin{aligned}\Delta_0 &= \mathbb{E}[b_0(n-1)f_0^*(n)] \\ &= \mathbb{E}[u(n-1)u^*(n)] \\ &= r^*(1),\end{aligned}$$

where $r(1)$ is the autocorrelation function of the input $u(n)$ for unity lag.

Relationship between Reflection Coefficients and Partial Correlation Coefficients

The *partial correlation (PARCOR) coefficient* between the forward prediction error $f_{m-1}(n)$ and the delayed backward prediction error $b_{m-1}(n-1)$ is defined as (Makhoul, 1977)

$$\rho_m = \frac{\mathbb{E}[b_{m-1}(n-1)f_{m-1}^*(n)]}{(\mathbb{E}[|b_{m-1}(n-1)|^2]\mathbb{E}[|f_{m-1}(n)|^2])^{1/2}}. \tag{3.61}$$

According to this definition, we always have

$$|\rho_m| \leq 1, \qquad \text{for all } m.$$

This upper bound on the magnitude of ρ_m follows readily from the *Cauchy–Schwarz inequality*, which, for the problem at hand, is[2]

$$|\mathbb{E}[b_{m-1}(n-1)f_{m-1}^*(n)]|^2 \leq \mathbb{E}[|[b_{m-1}(n-1)]|^2]\mathbb{E}[|f_{m-1}(n)|^2].$$

Using Eqs. (3.56) and (3.60), we may express the mth reflection coefficient as

$$\kappa_m = -\frac{\mathbb{E}[b_{m-1}(n-1)f_{m-1}^*(n)]}{P_{m-1}}. \tag{3.62}$$

In light of Eqs. (3.4) and (3.20), we may write

$$P_{m-1} = \mathbb{E}[|f_{m-1}(n)|^2] = \mathbb{E}[|b_{m-1}(n-1)|^2].$$

Hence, comparing Eqs. (3.61) and (3.62), we may now state that the reflection coefficient κ_m is the negative of the PARCOR coefficient ρ_m. However, this relationship holds just under the assumption of wide-sense stationarity, for it is only then that we may equate the forward prediction-error power to the backward prediction-error power for a prescribed prediction error.

[2]Consider the two sets of complex-valued numbers $\{a_n\}_{n=1}^N$ and $\{b_n\}_{n=1}^N$. According to the Cauchy–Schwarz inequality,

$$\left|\sum_{n=1}^N a_n b_n^*\right|^2 \leq \sum_{n=1}^N |a_n|^2 \sum_{n=1}^N |b_n|^2.$$

Substituting the expectation operation for summation, we may also express the Cauchy–Schwarz inequality as

$$|\mathbb{E}[a_n b_n^*]|^2 \leq \mathbb{E}[|a_n|^2][\mathbb{E}|b_n|^2].$$

Putting $a_n = f_{m-1}(n)$ and $b_n = b_{m-1}(n-1)$ in Eq. (3.61) and using the Cauchy–Schwarz inequality, we readily see that $|\rho_m| \leq 1$.

Application of the Levinson–Durbin Algorithm

There are two possible ways of applying the Levinson–Durbin algorithm to compute the prediction-error filter coefficients $a_{M,k}$, $k = 0, 1, \ldots, M$, and the prediction-error power P_M for a final prediction order M:

1. Suppose we have explicit knowledge of the autocorrelation function of the input process; in particular, suppose we have $r(0), r(1), \ldots, r(M)$, denoting the values of the autocorrelation function for lags $0, 1, \ldots, M$, respectively. For example, we may compute *biased* estimates of these parameters by means of the *time-average formula*

$$\hat{r}(k) = \frac{1}{N} \sum_{n=1+k}^{N} u(n)u^*(n-k), \qquad k = 0, 1, \ldots, M, \tag{3.63}$$

   where $N \gg M$ is the total length of the input time series. There are, of course, other estimators that we may use.[3] In any event, given $r(0), r(1), \ldots, r(M)$, the computation proceeds by using Eq. (3.47) for Δ_{m-1} and Eq. (3.57) for P_m. The recursion is initiated with $m = 0$, for which we have $P_0 = r(0)$ and $\Delta_0 = r^*(1)$. Note also that $a_{m,0}$ is unity for all m and $a_{m,k}$ is zero for all $k > m$. The computation is terminated when $m = M$. The resulting estimates of the prediction-error filter coefficients and prediction-error power obtained by using this procedure are known as the *Yule–Walker estimates*.

2. Suppose next we have explicit knowledge of the reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$ and the autocorrelation function $r(0)$ for a lag of zero. In this second application of the Levinson–Durbin recursion, we only need the pair of relations

$$a_{m,k} = a_{m-1,k} + \kappa_m a^*_{m-1,m-k}, \qquad k = 0, 1, \ldots, m,$$

   and

$$P_m = P_{m-1}(1 - |\kappa_m|^2).$$

   Here, again, the recursion is initiated with $m = 0$ and stopped when the order m reaches the final value M.

EXAMPLE 2

To illustrate the second method for the application of the Levinson–Durbin recursion, suppose we are given the reflection coefficients κ_1, κ_2, and κ_3 and the average power P_0. The problem we wish to solve is to use these parameters to determine the corresponding tap weights $a_{3,1}, a_{3,2}$, and $a_{3,3}$ and the prediction-error power P_3 for a prediction-error filter of order three. Application of the

[3]In practice, the biased estimate of Eq. (3.63) is preferred over an unbiased estimate because it yields a much lower variance for the estimate $\hat{r}(k)$ for values of the lag k close to the data length N. (For more details, see Box & Jenkins, 1976.) For a more refined estimate of the autocorrelation function $r(k)$, we may use the multiple-window method described in McWhorter and Scharf (1995). This method uses a multiplicity of special windows, resulting in the most general Hermitian, nonnegative-definite, and modulation-invariant estimate. The Hermitian and nonnegative-definite properties were described in Chapter 1. To define the modulation-invariant property, let $\hat{\mathbf{R}}$ denote an estimate of the correlation matrix, given the input vector $\mathbf{u}$. The estimate is said to be modulation invariant if $\mathbf{D}(e^{j\phi})\mathbf{u}$ has a correlation matrix equal to $\mathbf{D}(e^{j\phi})\hat{\mathbf{R}}\mathbf{D}(e^{-j\phi})$, where $\mathbf{D}(e^{j\phi}) = \text{diag}(\psi(e^{j\phi}))$ is a modulation matrix and $\psi(e^{j\phi}) = [1, e^{j\phi}, \ldots, e^{j\phi(M-1)}]^T$.

Levinson–Durbin recursion, described by Eqs. (3.41) and (3.57), yields the following results for $m = 1, 2, 3$:

1. For a prediction-error filter of order $m = 1$,

$$\begin{aligned} a_{1,0} &= 1, \\ a_{1,1} &= \kappa_1, \end{aligned}$$

and

$$P_1 = P_0(1 - |\kappa_1|^2).$$

2. Next, for a prediction-error filter of order $m = 2$,

$$\begin{aligned} a_{2,0} &= 1, \\ a_{2,1} &= \kappa_1 + \kappa_2\kappa_1^*, \\ a_{2,2} &= \kappa_2, \end{aligned}$$

and

$$P_2 = P_1(1 - |\kappa_2|^2),$$

where P_1 is as defined for $m = 1$.

3. Finally, for a prediction-error filter of order $m = 3$,

$$\begin{aligned} a_{3,0} &= 1, \\ a_{3,1} &= a_{2,1} + \kappa_3\kappa_2^*, \\ a_{3,2} &= \kappa_2 + \kappa_3 a_{2,1}^*, \\ a_{3,3} &= \kappa_3, \end{aligned}$$

and

$$P_3 = P_2(1 - |\kappa_3|^2),$$

where $a_{2,1}$ and P_2 are as defined for $m = 2$.

The interesting point to observe from this example is that the Levinson–Durbin recursion yields not only the values of the tap weights and prediction-error power for the prediction-error filter of final order M, but also the corresponding values of these parameters for the prediction-error filters of intermediate orders $M - 1, \ldots, 1$.

Inverse Levinson–Durbin Algorithm

In the normal application of the Levinson–Durbin recursion, as illustrated in Example 2, we are given the set of reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$, and the requirement is to compute the corresponding set of tap weights $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$ for a prediction-error filter of final order M. Of course, the remaining coefficient of the filter, $a_{M,0} = 1$. Frequently, however, the need arises to solve the following *inverse* problem: Given the set of tap weights $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$, solve for the corresponding set of reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$. We may solve this problem by applying the inverse form of the Levinson–Durbin recursion, which we refer to simply as the *inverse recursion*.

To derive the inverse recursion, we first combine Eqs. (3.41) and (3.43), representing the scalar versions of the Levinson–Durbin recursion for forward and backward prediction-error filters, respectively, in matrix form as

$$\begin{bmatrix} a_{m,k} \\ a^*_{m,m-k} \end{bmatrix} = \begin{bmatrix} 1 & \kappa_m \\ \kappa^*_m & 1 \end{bmatrix} \begin{bmatrix} a_{m-1,k} \\ a^*_{m-1,m-k} \end{bmatrix}, \qquad k = 0, 1, \ldots, m, \tag{3.64}$$

where the order $m = 1, 2, \ldots, M$. Then, assuming that $|\kappa_m| < 1$ and solving Eq. (3.64) for the tap weight $a_{m-1,k}$, we get

$$a_{m-1,k} = \frac{a_{m,k} - a_{m,m}a^*_{m,m-k}}{1 - |a_{m,m}|^2}, \qquad k = 0, 1, \ldots, m, \tag{3.65}$$

where we have used the fact that $\kappa_m = a_{m,m}$. We may now describe the procedure:

1. Starting with the set of tap weights $\{a_{M,k}\}$ for which the prediction-error filter order equals M, use the inverse recursion, Eq. (3.65), with decreasing filter order $m = M, M-1, \ldots, 2$ to compute the tap weights of the corresponding prediction-error filters of order $M-1, M-2, \ldots, 1$, respectively.
2. Finally, knowing the tap weights of all the prediction-error filters of interest (whose order ranges all the way from M down to 1), use the fact that

$$\kappa_m = a_{m,m}, \quad m = M, M-1, \ldots, 1,$$

to determine the desired set of reflection coefficients $\kappa_M, \kappa_{M-1}, \ldots, \kappa_1$.

Example 3 illustrates the application of the inverse recursion.

EXAMPLE 3

Suppose we are given the tap weights $a_{3,1}, a_{3,2}, a_{3,3}$ of a prediction-error filter of order three and the requirement is to determine the corresponding reflection coefficients $\kappa_1, \kappa_2, \kappa_3$. Application of the inverse recursion, described by Eq. (3.65), for filter order $m = 3, 2$ yields the following set of tap weights:

1. For a prediction-error filter of order two [corresponding to $m = 3$ in Eq. (3.65)],

$$a_{2,1} = \frac{a_{3,1} - a_{3,3}a^*_{3,2}}{1 - |a_{3,3}|^2}$$

and

$$a_{2,2} = \frac{a_{3,2} - a_{3,3}a^*_{3,1}}{1 - |a_{3,3}|^2}.$$

2. For a prediction-error filter of order one [corresponding to $m = 2$ in Eq. (3.65)],

$$a_{1,1} = \frac{a_{2,1} - a_{2,2}a^*_{2,1}}{1 - |a_{2,2}|^2},$$

where $a_{2,1}$ and $a_{2,2}$ are as defined for a filter of order two. Thus, the required reflection coefficients are given by

$$\kappa_3 = a_{3,3},$$
$$\kappa_2 = a_{2,2},$$

and

$$\kappa_1 = a_{1,1},$$

where $a_{3,3}$ is given and $a_{2,2}$ and $a_{1,1}$ are as just computed.

3.4 PROPERTIES OF PREDICTION-ERROR FILTERS

Property 1. ***Relations between the autocorrelation function and the reflection coefficients*** It is customary to represent the second-order statistics of a stationary time series in terms of its autocorrelation function or, equivalently, the power spectrum. The autocorrelation function and power spectrum form a discrete-time Fourier-transform pair. (See Chapter 1.) Another way of describing the second-order statistics of a stationary time series is to use the set of numbers $P_0, \kappa_1, \kappa_2, \ldots, \kappa_M$, where $P_0 = r(0)$ is the value of the autocorrelation function of the process for a lag of zero and $\kappa_1, \kappa_2, \ldots, \kappa_M$, are the reflection coefficients for a prediction-error filter of final order M. This is a consequence of the fact that the set of numbers $P_0, \kappa_1, \kappa_2, \ldots, \kappa_M$ uniquely determines the corresponding set of autocorrelation function values $r(0), r(1), \ldots, r(M)$ and vice versa.

To prove this property, we first eliminate Δ_{m-1} between Eqs. (3.47) and (3.55), obtaining

$$\sum_{k=0}^{m-1} a_{m-1,k} r(k-m) = -\kappa_m P_{m-1}. \tag{3.66}$$

Solving Eq. (3.66) for $r(m) = r^*(-m)$ and recognizing that $a_{m-1,0} = 1$, we get

$$r(m) = -\kappa_m^* P_{m-1} - \sum_{k=1}^{m-1} a_{m-1,k}^* r(m-k). \tag{3.67}$$

This is the desired recursive relation for order-updating of the autocorrelation function of a wide-shape stationary process. If we are given the set of numbers $r(0)$, $\kappa_1, \kappa_2, \ldots, \kappa_M$, then, by using Eq. (3.67), together with the Levinson–Durbin recursive equations (3.41) and (3.57), we may recursively generate the corresponding set of numbers $r(0), r(1), \ldots, r(M)$.

For $|k_m| \leq 1$, we find from Eq. (3.67) that the permissible region for $r(m)$, the value of the autocorrelation function of the input signal for a lag of m, is the interior (including the circumference) of a circle of radius P_{m-1} and center at the complex-valued quantity

$$-\sum_{k=1}^{m-1} a_{m-1,k}^* r(m-k).$$

This region is illustrated in Fig. 3.3.

Suppose now that we are given the set of autocorrelation function values $r(1), \ldots, r(M)$. Then we may recursively generate the corresponding set of numbers $\kappa_1, \kappa_2, \ldots, \kappa_M$ by using the relation

$$\kappa_m = -\frac{1}{P_{m-1}} \sum_{k=0}^{m-1} a_{m-1,k} r(k-m), \tag{3.68}$$

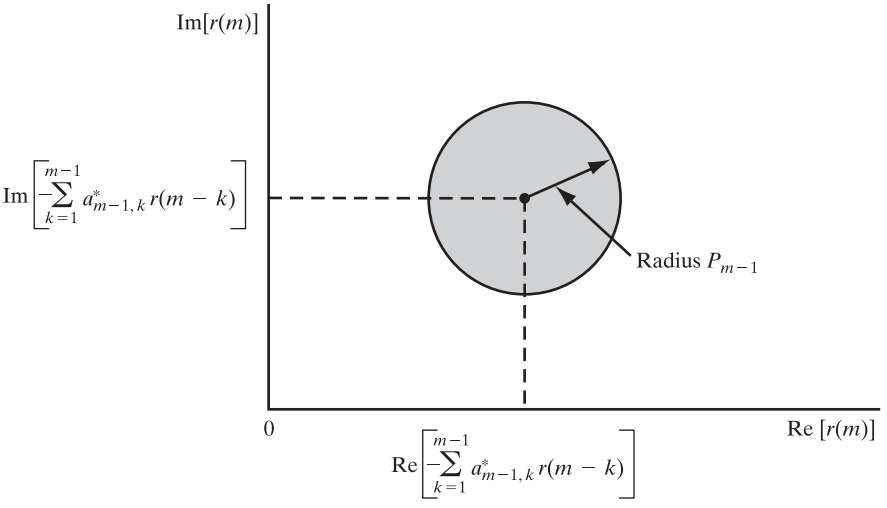


FIGURE 3.3 Permissible region for $r(m)$, shown shaded, for the case when $|\kappa_m| \leq 1$.

which is obtained by solving Eq. (3.66) for K_m. In Eq. (3.68), it is assumed that P_{m-1} is nonzero. If P_{m-1} is zero, this would have been the result for $|k_{m-1}| = 1$, and the sequence of reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_{m-1}$ would terminate.

We may therefore make the following statement:

There is a one-to-one correspondence between the two sets of quantities $\{P_0, \kappa_1, \kappa_2, \ldots, \kappa_M\}$ and $\{r(0), r(1), r(2), \ldots, r(M)\}$ *in that if we are given the one, we may uniquely determine the other in a recursive manner.*

EXAMPLE 4

Suppose that we are given P_0, κ_1, κ_2, and κ_3 and the requirement is to compute $r(0), r(1), r(2)$, and $r(3)$. We start with $m = 1$, for which Eq. (3.67) yields

$$r(1) = -P_0 \kappa_1^*,$$

where

$$P_0 = r(0).$$

Next, for $m = 2$, Eq. (3.67) yields

$$r(2) = -P_1 \kappa_2^* - r(1)\kappa_1^*,$$

where

$$P_1 = P_0(1 - |\kappa_1|^2).$$

Finally, for $m = 3$, the use of Eq. (3.67) yields

$$r(3) = -P_2 \kappa_3^* - [a_{2,1}^* r(2) + \kappa_2^* r(1)],$$

where

$$P_2 = P_1(1 - |\kappa_2|^2)$$

and

$$a_{2,1} = \kappa_1 + \kappa_2 \kappa_1^*.$$

Property 2. ***Transfer function of a forward prediction-error filter*** Let $H_{f,m}(z)$ denote the *transfer function* of a forward prediction-error filter of order m and whose impulse response is defined by the sequence of numbers $a^*_{m,k}$, $k = 0, 1, \ldots, m$, as illustrated in Fig. 3.1(b) for $m = M$. From discrete-time (digital) signal processing, we know that the transfer function of a discrete-time filter equals the z-transform of its impulse response. We may therefore write

$$H_{f,m}(z) = \sum_{k=0}^{m} a^*_{m,k} z^{-k}, \tag{3.69}$$

where z is a complex variable. Based on the Levinson–Durbin recursion—in particular, Eq. (3.41)—we may relate the coefficients of this filter of order m to those of a corresponding prediction-error filter of order $m - 1$ (i.e., one order smaller). In particular, substituting Eq. (3.41) into Eq. (3.69), we get

$$\begin{aligned} H_{f,m}(z) &= \sum_{k=0}^{m} a^*_{m-1,k} z^{-k} + \kappa^*_m \sum_{k=0}^{m} a_{m-1,m-k} z^{-k} \\ &= \sum_{k=0}^{m-1} a^*_{m-1,k} z^{-k} + \kappa^*_m z^{-1} \sum_{k=0}^{m-1} a_{m-1,m-1-k} z^{-k}, \end{aligned} \tag{3.70}$$

where, in the second line, we have used the fact that $a_{m-1,m} = 0$. The sequence of numbers $a^*_{m-1,k}$, $k = 0, 1, \ldots, m-1$, defines the impulse response of a forward prediction-error filter of order $m - 1$. Hence, we may write

$$H_{f,m-1}(z) = \sum_{k=0}^{m-1} a^*_{m-1,k} z^{-k}. \tag{3.71}$$

The sequence of numbers $a_{m-1,m-1-k}$, $k = 0, 1, \ldots, m-1$, defines the impulse response of a backward prediction-error filter of order $m - 1$; this is illustrated in Fig. 3.2(c) for the case of prediction order $m = M$. Thus, the second summation on the right-hand side of Eq. (3.70) represents the transfer function of this backward prediction-error filter. Let $H_{b,m-1}(z)$ denote that transfer function, as shown by

$$H_{b,m-1}(z) = \sum_{k=0}^{m-1} a_{m-1,m-1-k} z^{-k}. \tag{3.72}$$

Then, substituting Eqs. (3.71) and (3.72) into Eq. (3.70), we may write

$$H_{f,m}(z) = H_{f,m-1}(z) + \kappa^*_m z^{-1} H_{b,m-1}(z). \tag{3.73}$$

On the basis of the order-update recursion of Eq. (3.73), we may now state the following:

> *Given the reflection coefficient κ_m and the transfer functions of the forward and backward prediction-error filters of order $m - 1$,* the transfer function of the corresponding forward prediction-error filter of order *m is uniquely determined.*

Property 3. ***A forward prediction-error filter is minimum phase*** On the unit circle in the z-plane (i.e., for $|z| = 1$), we find that

$$|H_{f,m-1}(z)| = |H_{b,m-1}(z)|, \qquad |z| = 1.$$

This equality is readily proved by substituting $z = \exp(j\omega)$, $-\pi < \omega \leq \pi$, in Eqs. (3.71) and (3.72). Suppose that the reflection coefficient κ_m satisfies the requirement $|\kappa_m| < 1$ for all m. Then we find that on the unit circle in the z-plane, the magnitude of the second term in the right-hand side of Eq. (3.73) satisfies the condition

$$|\kappa_m^* z^{-1} H_{b,\,m-1}(z)| < |H_{b,\,m-1}(z)| = |H_{f,\,m-1}(z)|, \qquad |z| = 1. \tag{3.74}$$

At this stage in our discussion, it is useful to recall *Rouché's theorem* from the theory of complex variables. Rouché's theorem states the following:

> If a function $F(z)$ is analytic upon a contour $\mathscr{C}$ in the z-plane and within the region enclosed by that contour, and if a second function $G(z)$, in addition to satisfying the same analyticity conditions, also fulfills the condition $|G(z)| < F(z)$ on the contour $\mathscr{C}$, then the function $F(z) + G(z)$ has the same number of zeros within the region enclosed by the contour $\mathscr{C}$ as does the function $F(z)$.

(For a review of complex-variable theory, including Rouché's theorem, see the material presented in Appendix A.)

Ordinarily, the enclosed contour $\mathscr{C}$ is transversed in the *counterclockwise* direction, and the region enclosed by the contour lies to the *left* of it, as illustrated in Fig. 3.4. We say that a function is *analytic upon the contour* $\mathscr{C}$ *and within the region enclosed by* $\mathscr{C}$ if the function has a continuous derivative everywhere upon the contour $\mathscr{C}$ and within the region enclosed by $\mathscr{C}$. For this requirement to be satisfied, the function must have no poles upon the contour $\mathscr{C}$ or inside the region enclosed by the contour.

Let the contour $\mathscr{C}$ be the unit circle in the z-plane, which is traversed in the *clockwise* direction, as in Fig. 3.5. According to the convention just described, this requirement

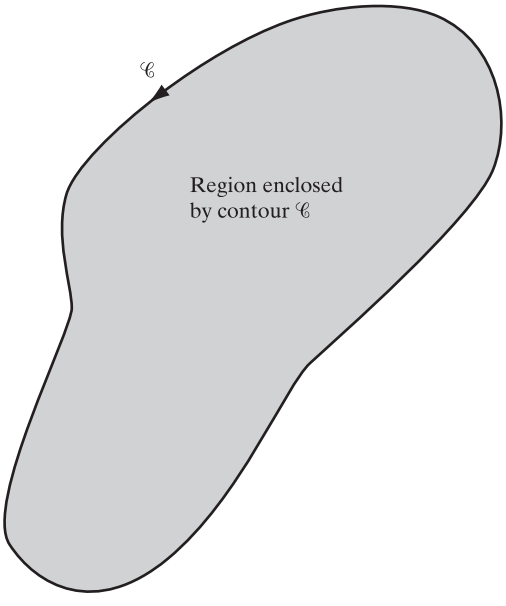


FIGURE 3.4 Contour $\mathscr{C}$ (traversed in the counterclockwise direction) in the z-plane and the region enclosed by it.

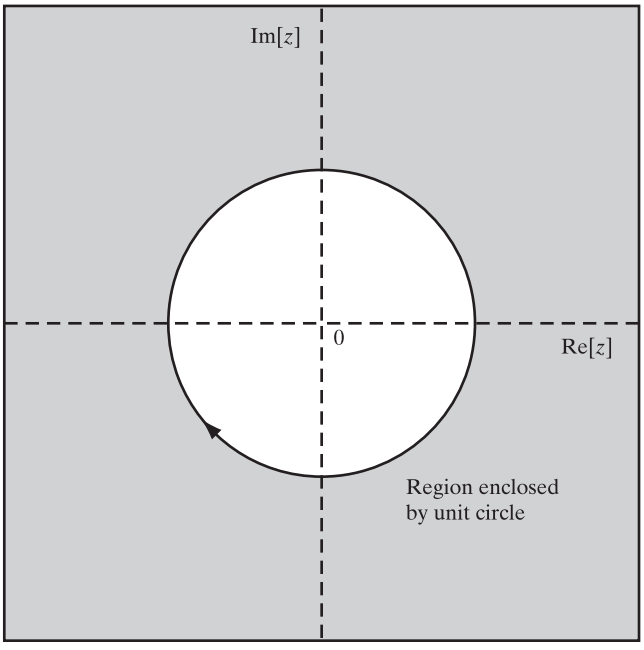


FIGURE 3.5 Unit circle (traversed in the clockwise direction) used as contour $\mathscr{C}$.

implies that the region enclosed by the contour $\mathscr{C}$ is represented by the entire part of the z-plane that lies *outside* the unit circle.

Let the functions $F(z)$ and $G(z)$ be identified, in order, with the two terms on the right-hand side of Eq. (3.73); that is,

$$F(z) = H_{f,m-1}(z) \tag{3.75}$$

and

$$G(z) = \kappa_m^* z^{-1} H_{b,m-1}(z). \tag{3.76}$$

We observe that:

1. The functions $F(z)$ and $G(z)$ have no poles inside the contour $\mathscr{C}$ defined in Fig. 3.5. Indeed, their derivatives are continuous throughout the region enclosed by this contour. Therefore, $F(z)$ and $G(z)$ are both analytic everywhere upon the unit circle and the region outside it.
2. In view of Eq. (3.74) and the fact that $|\kappa_m| < 1$, we have $|G(z)| < |F(z)|$ on the unit circle.

Accordingly, the functions $F(z)$ and $G(z)$ defined by Eqs. (3.75) and (3.76), respectively, satisfy all the conditions required by Rouché's theorem with respect to the contour $\mathscr{C}$ defined as the unit circle in Fig. 3.5.

Suppose now that $H_{f,m-1}(z)$ and therefore $F(z)$ are known to have no zeros outside the unit circle in the z-plane. Then, by applying Rouché's theorem, we find that $F(z) + G(z)$, or, equivalently, $H_{f,m}(z)$ also has no zeros on or outside the unit circle in the z-plane.

In particular, for $m = 0$, the transfer function $H_{f,0}(z)$ is a constant equal to unity; therefore, it has no zeros at all. Using the result just derived, we may state that, since $H_{f,0}(z)$ has no zeros outside the unit circle, $H_{f,1}(z)$ will also have no zeros in this region of the z-plane, provided that $|\kappa_1| < 1$. Indeed, we can easily prove the latter result by noting that

$$\begin{aligned} H_{f,1}(z) &= a^*_{1,0} + a^*_{1,1}z^{-1} \\ &= 1 + \kappa^*_1 z^{-1}. \end{aligned}$$

Hence, $H_{f,1}(z)$ has a single zero located at $z = -\kappa^*_1$ and a pole at $z = 0$. With the reflection coefficient κ_1 constrained by the condition $|\kappa_1| < 1$, it follows that this zero must lie inside the unit circle. In other words, $H_{f,1}(z)$ has no zeros on or outside the unit circle. In that case, $H_{f,2}(z)$ will also have no zeros on or outside the unit circle, provided that $|\kappa_2| < 1$ and so on.

We may thus state that, for all values of m, the transfer function $H_{f,m}(z)$ of a forward prediction-error filter of order m has no zeros on or outside the unit circle in the z-plane if and only if the reflection coefficients satisfy the condition $|\kappa_m| < 1$ for all m. Such a filter is said to be *minimum phase* in the sense that, for a specified amplitude response, it has the minimum phase response possible for all values of z on the unit circle. Moreover, the amplitude response and phase response of the filter are uniquely related to each other. On the basis of these findings, we may now make the following statement:

> *For all values of m, the transfer function $H_{f,m}(z)$ of a forward prediction-error filter of order m has no zeros on or outside the unit circle in the z-plane if and only if the reflection coefficients satisfy the condition $|\kappa_m| < 1$. In other words, a forward prediction-error filter is minimum phase in the sense that, for a specified amplitude response, it has the minimum phase response possible for all values of z on the unit circle.*

Property 4. ***A backward prediction-error filter is maximum phase*** The transfer functions of backward and forward prediction-error filters of the same order are related in that if we are given one, we may uniquely determine the other. To find this relationship, we first evaluate $H^*_{f,m}(z)$, the complex conjugate of the transfer function of a forward prediction-error filter of order m, and so write [see Eq. (3.69)]

$$H^*_{f,m}(z) = \sum_{k=0}^{m} a_{m,k}(z^*)^{-k}. \tag{3.77}$$

Replacing z by the reciprocal of its complex conjugate z^*, we may rewrite Eq. (3.77) as

$$H^*_{f,m}\left(\frac{1}{z^*}\right) = \sum_{k=0}^{m} a_{m,k} z^k.$$

Next, replacing k by $m - k$, we get

$$H^*_{f,m}\left(\frac{1}{z^*}\right) = z^m \sum_{k=0}^{m} a_{m,m-k} z^{-k}. \tag{3.78}$$

The summation on the right-hand side of Eq. (3.78) constitutes the transfer function of a backward prediction-error filter of order m, as shown by

$$H_{b,m}(z) = \sum_{k=0}^{m} a_{m,m-k} z^{-k}. \tag{3.79}$$

We thus find that $H_{b,m}(z)$ and $H_{f,m}(z)$ are related by the formula

$$H_{b,m}(z) = z^{-m}H_{f,m}^{*}\left(\frac{1}{z^{*}}\right), \tag{3.80}$$

where $H_{f,m}^{*}(1/z^{*})$ is obtained by taking the complex conjugate of $H_{f,m}(z)$, the transfer function of a forward prediction-error filter of order m, and replacing z by the reciprocal of z^{*}. Equation (3.80) states that multiplication of the new function obtained in this way by z^{-m} yields $H_{b,m}(z)$, the transfer function of the corresponding backward prediction-error filter.

Let the transfer function $H_{f,m}(z)$ be expressed in its factored form as

$$H_{f,m}(z) = \prod_{i=1}^{m}(1 - z_i z^{-1}), \tag{3.81}$$

where the $z_i, i = 1, 2, \ldots, m$, denote the zeros of the forward prediction-error filter. Hence, substituting Eq. (3.81) into Eq. (3.80), we may express the transfer function of the corresponding backward prediction-error filter in the factored form

$$\begin{aligned} H_{b,m}(z) &= z^{-m}\prod_{i=1}^{m}(1 - z_i^{*}z) \\ &= \prod_{i=1}^{m}(z^{-1} - z_i^{*}). \end{aligned} \tag{3.82}$$

The zeros of this transfer function are located at $1/z_i^{*}$, $i = 1, 2, \ldots m$. That is, the zeros of the backward and forward prediction-error filters are the *inverse* of each other with respect to the unit circle in the z-plane. The geometric nature of this relationship is illustrated for $m = 1$ in Fig. 3.6. The forward prediction-error filter has a zero at $z = -\kappa_1^{*}$, as in Fig. 3.6(a), and the backward prediction-error filter has a zero at $z = -1/\kappa_1$, as in Fig. 3.6(b). In both parts of the figure, it is assumed that the reflection coefficient κ_1 has a complex value. Consequently, a backward prediction-error filter has all of its zeros located outside the unit circle in the z-plane, because $|\kappa_m| < 1$ for all m.

We may therefore formally make the following statement:

A backward prediction-error filter is maximum phase in the sense that, for a specified amplitude response, it has the maximum phase response possible for all values of z on the unit circle.

Property 5. ***A forward prediction-error filter is a whitening filter*** By definition, a *white-noise* process consists of a sequence of uncorrelated random variables. Thus, assuming that such a process, denoted by $\nu(n)$, has zero mean and variance σ_ν^2, we may write (see Section 1.5)

$$\mathbb{E}[\nu(k)\nu^{*}(n)] = \begin{cases} \sigma_\nu^2, & k = n \\ 0, & k \neq n \end{cases}. \tag{3.83}$$

Accordingly, we say that white noise is unpredictable in the sense that the value of the process at time n is uncorrelated with all past values of the process up to and including time $n - 1$ (and, indeed, with all future values of the process, too).

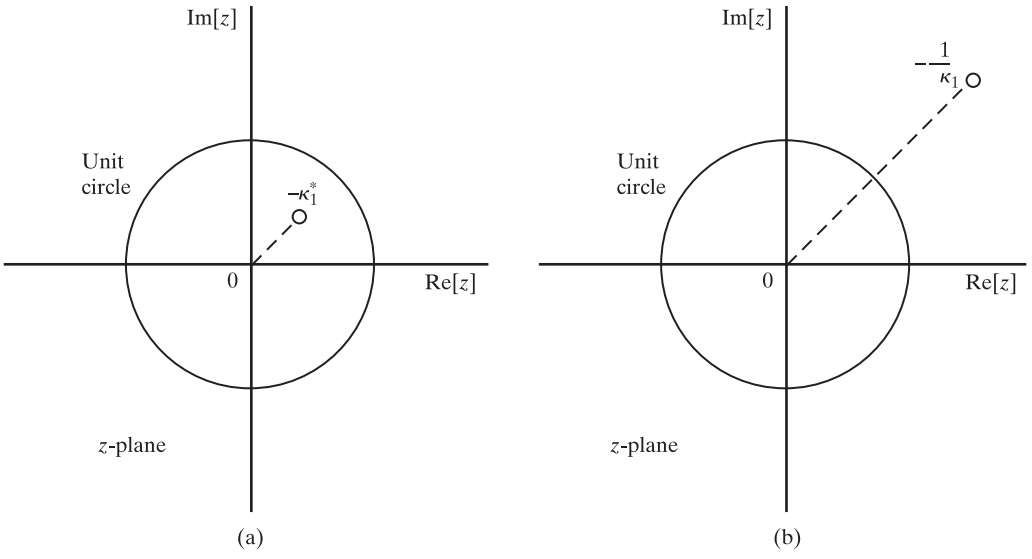


FIGURE 3.6 (a) Zero of forward prediction-error filter at $z = -\kappa_1^*$; (b) corresponding zero of backward prediction-error filter at $z = -1/\kappa_1$.

We may now state another important property of prediction-error filters:

A prediction-error filter is capable of whitening a stationary discrete-time stochastic process applied to its input, provided that the order of the filter is high enough.

Basically, prediction relies on the presence of correlation between adjacent samples of the input process. The implication of this correlation is that, as we increase the order of the prediction-error filter, we successively reduce the correlation between adjacent samples of the input process, until ultimately we reach a point at which the filter has a high enough order to produce an output process that consists of a sequence of uncorrelated samples. The whitening of the original process applied to the filter input will have thereby been accomplished.

Property 6. ***Eigenvector representation of forward prediction-error filters*** The representation of a forward prediction-error filter is naturally related to eigenvalues (and associated eigenvectors) of the correlation matrix of the tap inputs in the filter. To develop such a representation, we first rewrite the augmented Wiener–Hopf equations (3.14), pertaining to a forward prediction-error filter of order M, in the compact matrix form

$$\mathbf{R}_{M+1}\mathbf{a}_M = P_M\mathbf{i}_{M+1}, \tag{3.84}$$

where $\mathbf{R}_{M+1}$ is the $(M + 1)$-by-$(M + 1)$ correlation matrix of the tap inputs $u(n)$, $u(n - 1), \ldots, u(n - M)$ in the filter of Fig. 3.1(b), $\mathbf{a}_M$ is the $(M + 1)$-by-1 tap-weight vector of the filter, and the scalar P_M is the prediction-error power. The $(M + 1)$-by-1 vector $\mathbf{i}_{M+1}$, called the *first coordinate vector*, has unity for its first element and zero for all the others. We define this vector as

$$\mathbf{i}_{M+1} = [\,1, 0, \ldots, 0\,]^{\mathrm{T}}. \tag{3.85}$$

Solving Eq. (3.84) for $\mathbf{a}_M$, we get

$$\mathbf{a}_M = P_M \mathbf{R}_{M+1}^{-1} \mathbf{i}_{M+1}, \tag{3.86}$$

where $\mathbf{R}_{M+1}^{-1}$ is the inverse of the correlation matrix $\mathbf{R}_{M+1}$. (The matrix $\mathbf{R}_{M+1}$ is assumed to be nonsingular, so that its inverse exists.) Using the eigenvalue–eigenvector representation of a correlation matrix, we may express the inverse matrix $\mathbf{R}_{M+1}^{-1}$ as (see Appendix E):

$$\mathbf{R}_{M+1}^{-1} = \mathbf{Q}\mathbf{\Lambda}^{-1}\mathbf{Q}^{\mathrm{H}}, \tag{3.87}$$

where Λ is an $(M + 1)$-by-$(M + 1)$ diagonal matrix consisting of the eigenvalues of the correlation matrix $\mathbf{R}_{M+1}$ and $\mathbf{Q}$ is an $(M + 1)$-by-$(M + 1)$ matrix whose columns are the associated eigenvectors. That is,

$$\mathbf{\Lambda} = \mathrm{diag}[\,\lambda_0, \lambda_1, \ldots, \lambda_M\,] \tag{3.88}$$

and

$$\mathbf{Q} = [\,\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_M], \tag{3.89}$$

where $\lambda_0, \lambda_1, \ldots, \lambda_M$ are the real-valued eigenvalues of the correlation matrix $\mathbf{R}_{M+1}$ and $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_M$ are the respective eigenvectors. Thus, substituting Eqs. (3.87), (3.88), and (3.89) into Eq. (3.86), we get

$$\begin{aligned}
\mathbf{a}_M &= P_M \mathbf{Q}\Lambda^{-1}\mathbf{Q}^{\mathrm{H}}\mathbf{i}_{M+1} \\
&= P_M[\,\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_M\,]\ \mathrm{diag}[\,\lambda_0^{-1}, \lambda_1^{-1}, \ldots, \lambda_M^{-1}\,] \begin{bmatrix} \mathbf{q}_0^{\mathrm{H}} \\ \mathbf{q}_1^{\mathrm{H}} \\ \vdots \\ \mathbf{q}_M^{\mathrm{H}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\
&= P_M \sum_{k=0}^{M} \left(\frac{q_{k0}^*}{\lambda_k} \right) \mathbf{q}_k,
\end{aligned} \tag{3.90}$$

where q_{k0}^* is the first element of the kth eigenvector of the correlation matrix $\mathbf{R}_{M+1}$. We now note that the first element of the forward prediction-error filter vector $\mathbf{a}_M$ is unity; therefore, solving Eq. (3.90), for the prediction-error power, we get

$$P_M = \frac{1}{\displaystyle\sum_{k=0}^{M} |\,q_{k0}\,|^2 \lambda_k^{-1}}. \tag{3.91}$$

Thus, on the basis of Eqs. (3.90) and (3.91), we may make the following statement:

> *The tap-weight vector of a forward prediction-error filter of order M and the resultant prediction-error power are uniquely defined by specifying the* $(M + 1)$ *eigenvalues and the corresponding* $(M + 1)$ *eigenvectors of the correlation matrix of the tap inputs of the filter.*

3.5 SCHUR–COHN TEST

The test described under Property 3 in Section 3.4 for the minimum-phase condition of a forward prediction-error of order M is relatively simple to apply if we know the associated set of reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$. For the filter to be minimum phase [i.e., for all the zeros of the transfer function $H_{f,m}(z)$ of the filter to lie inside the unit circle], we simply require that $|k_m| < 1$ for all m. Suppose, however, that instead of these reflection coefficients we are given the tap weights of the filter, $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$. In that case, first we apply the inverse recursion [described by Eq. (3.65)] to compute the corresponding set of reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$, and then, as before, we check whether $|k_m| < 1$ for all m.

The method just described for determining whether $H_{f,m}(z)$ has zeros inside the unit circle, given the coefficients $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$, is essentially the same as the *Schur–Cohn test*.[4]

To formulate the Schur–Cohn test, let

$$x(z) = a_{M,M}z^M + a_{M,M-1}z^{M-1} + \cdots + a_{M,0}, \tag{3.92}$$

which is a polynomial in z, with $x(0) = a_{M,0} = 1$. Define

$$\begin{aligned} x'(z) &= z^M x^*(1/z^*) \\ &= a^*_{M,M} + a^*_{M,M-1}z + \cdots + a^*_{M,0}z^M, \end{aligned} \tag{3.93}$$

which is the *reciprocal polynomial* associated with $x(z)$. The polynomial $x'(z)$ is so called because its zeros are the reciprocals of the zeros of $x(z)$. For $z = 0$, we have $x'(0) = a^*_{M,M}$. Next, define the linear combination

$$T[x(z)] = a^*_{M,0}x(z) - a_{M,M}x'(z) \tag{3.94}$$

so that, in particular, the value

$$\begin{aligned} T[x(0)] &= a^*_{M,0}x(0) - a_{M,M}x'(0) \\ &= 1 - |a_{M,M}|^2 \end{aligned} \tag{3.95}$$

is real, as it should be. Note also that $T[x(z)]$ has no term in z^M. Repeat this operation as far as possible, so that if we define

$$T^i[x(z)] = T\{T^{i-1}[x(z)]\}, \tag{3.96}$$

we generate a finite sequence of polynomials in z of *decreasing* order. The coefficient $a_{M,0}$ is equal to unity. Let it also be assumed that

1. the polynomial $x(z)$ has no zeros on the unit circle and
2. the integer m is the smallest for which

$$T^m[x(z)] = 0, \qquad \text{where } m \leq M + 1.$$

[4]The classical Schur–Cohn test is discussed in Marden (1949) and Tretter (1976). The origin of the test can be traced back to Schur (1917) and Cohn (1922)—hence the name. The test is also referred to as the Lehmer–Schur method (Ralston, 1965), in recognition of the application of Schur's theorem by Lehmer (1961).

Then, we may state the Schur–Cohn theorem as follows (Lehmer, 1961):

> *If, for some i such that* $1 \leq i \leq m$, *we have* $T^i[x(0)] < 0$, *then* $x(z)$ *has at least one zero inside the unit circle. If, on the other hand,* $T^i[x(0)] > 0$ *for* $1 \leq i < m$ *and* $T^{m-1}[x(z)]$ *is a constant, then no zero of* $x(z)$ *lies inside the unit circle.*

To apply this theorem to determine whether the polynomial $x(z)$ of Eq. (3.92), with $a_{M,0} \neq 0$, has a zero inside the unit circle, we proceed as follows (Ralston, 1965):

1. Calculate $T[x(z)]$. Is $T[x(0)]$ negative? If so, there is a zero inside the unit circle; if not, proceed to step 2.
2. Calculate $T^i[x(z)]$, $i = 1, 2, \ldots,$ until $T^i[x(0)] < 0$ for $i < m$ or until $T^i[x(0)] > 0$ for $i < m$. If the former occurs, there is a zero inside the unit circle. If the latter occurs and if $T^{m-1}[x(z)]$ is a constant, then there is no zero inside the unit circle.

Note that when the polynomial $x(z)$ has zeros inside the unit circle, this algorithm does not tell us how many there are; rather, it only confirms their existence.

The connection between the Schur–Cohn method and inverse recursion is readily established by observing that (see Problem 10)

1. The polynomial $x(z)$ is related to the transfer function of a backward prediction-error filter of order M by the formula

$$x(z) = z^M H_{b,M}(z). \tag{3.97}$$

   Accordingly, if the Schur–Cohn test indicates that $x(z)$ has one or more zeros inside the unit circle, we may conclude that the transfer function $H_{b,M}(z)$ is *not* maximum phase.
2. The reciprocal polynomial $x'(z)$ is related to the transfer function of the corresponding forward prediction-error filter of order M by the formula

$$x'(z) = z^M H_{f,M}(z). \tag{3.98}$$

   Accordingly, if the Schur–Cohn test indicates that the original polynomial $x(z)$ with which $x'(z)$ is associated has no zeros inside the unit circle, we may then conclude that the transfer function $H_{f,M}(z)$ is *not* minimum phase.
3. In general,

$$T^i[x(0)] = \prod_{j=0}^{i-1} (1 - |a_{M-j,M-j}|^2), \qquad 1 \leq i \leq M, \tag{3.99}$$

   and

$$H_{b,M-i}(z) = \frac{z^{i-M} T^i[x(z)]}{T^i[x(0)]}, \tag{3.100}$$

where $H_{b,M-i}(z)$ is the transfer function of the backward prediction-error filter of order $M - i$.

3.6 AUTOREGRESSIVE MODELING OF A STATIONARY STOCHASTIC PROCESS

The whitening property of a forward prediction-error filter operating on a stationary discrete-time stochastic process is intimately related to the *autoregressive modeling* of the process. Indeed, we may view these two operations as *complementary*, as illustrated in Fig. 3.7. Part (a) of the figure depicts a forward prediction-error filter of order M, whereas part (b) depicts the corresponding autoregressive model. Based on these two parts of the figure, we may make the following observations:

1. We may view the operation of prediction-error filtering applied to a stationary process $u(n)$ as one of *analysis*. In particular, we may use such an operation to whiten the process $u(n)$ by choosing the prediction-error filter order M sufficiently large, in which case the prediction error process $f_M(n)$ at the filter output consists of uncorrelated samples. When this unique condition has been established, the original stochastic process $u(n)$ is represented by the set of tap weights of the filter, $\{a_{M,k}\}$, and the prediction error power P_M.
2. We may view the autoregressive (AR) modeling of the stationary process $u(n)$ as one of *synthesis*. In particular, we may generate the AR process $u(n)$ by applying a white-noise process $\nu(n)$ of zero mean and variance σ_ν^2 to the input of an *inverse* filter whose parameters are set equal to the AR parameters w_{ok}, $k = 1, 2, \ldots, M$. The resulting output of the model, denoted by $u(n)$, is regressed on M past values of the input, namely, $u(n-1), \ldots, u(n-M)$—hence the name of the model.

The two filter structures of Fig. 3.7 constitute a matched pair, with their parameters related by

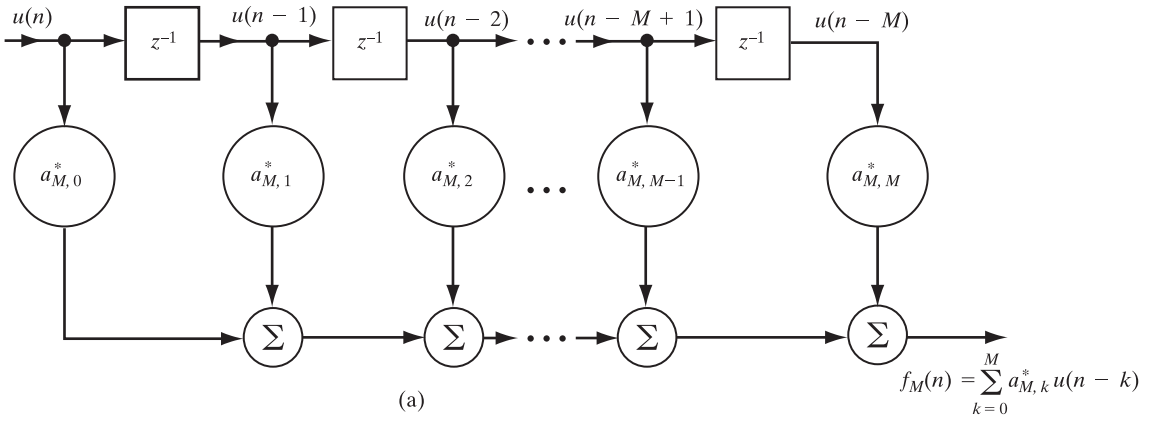
$$a_{M,k} = -w_{ok}, \quad k = 1, 2, \ldots, M,$$

and

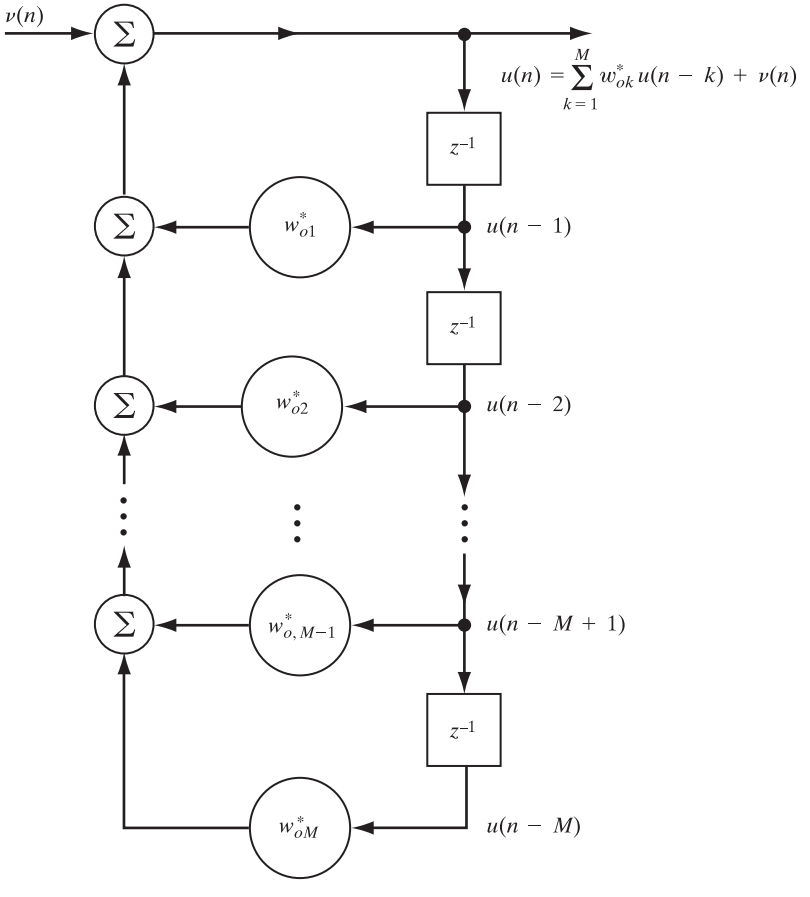
$$P_M = \sigma_\nu^2.$$

The prediction-error filter of Fig. 3.7(a) is an *all-zero filter with an impulse response of finite duration*. On the other hand, the inverse filter in the AR model of Fig. 3.7(b) is an *all-pole filter with an impulse response of infinite duration*. The prediction-error filter is minimum phase, with the zeros of its transfer function located at exactly the same positions (inside the unit circle in the z-plane) as the poles of the transfer function of the inverse filter in part (b) of the figure. This assures the stability of the inverse filter in the bounded-input–bounded-output sense or, equivalently, the asymptotic stationarity of the AR process generated at the output of the filter.

The mathematical equivalence between forward prediction-error filtering and autoregressive modeling can be put to practical use in the following way: Suppose we have an autoregressive process of order M, but the regression coefficients of the process are unknown. In such a situation, we may use an adaptive form of linear prediction to estimate the regression coefficients. Adaptive algorithms for the design of linear predictors or prediction-error filters are discussed in subsequent chapters.



(a)



(b)

FIGURE 3.7 (a) Prediction-error (all-zero) filter. (b) Autoregressive (all-pole) model with $w_{ok} = -a_{M,k}$, where $k = 1, 2, \ldots, M$; the input $\nu(n)$ is white noise.

Equivalence of the Autoregressive and Maximum-Entropy Power Spectra

Another way in which the mathematical equivalence between forward prediction-error filter modeling and autoregressive modeling manifests itself is in *parametric power spectrum estimation*. To explore this issue, consider the AR model of Fig. 3.7(b). The process $\nu(n)$ applied to the model input consists of a white noise of zero mean and variance σ_ν^2. To find the power spectral density of the AR process $u(n)$ produced at the model output, we multiply the power spectral density of the model input $\nu(n)$ by the squared amplitude response of the model. (See Chapter 1.) Let $S_{\text{AR}}(\omega)$ denote the power spectral density of $u(n)$. We may then write

$$S_{\text{AR}}(\omega) = \frac{\sigma_\nu^2}{\left|1 - \sum_{k=1}^{M} w_{ok}^* e^{-\text{j}\omega k}\right|^2}. \tag{3.101}$$

Equation (3.101) is called the *autoregressive power spectrum* or, simply, the *AR spectrum.*

A power spectrum of closely related interest is obtained using the *maximum-entropy method (MEM)*. Suppose that we are given $2M + 1$ values of the autocorrelation function of a wide-sense stationary process $u(n)$. The essence of the maximum-entropy method is to determine the particular power spectrum of the process that corresponds to the most random time series whose autocorrelation function is consistent with the set of $2M + 1$ known values (Burg, 1968, 1975). The result so obtained is referred to as the *maximum-entropy power spectrum* or, simply, the *MEM spectrum*. Let $S_{\text{MEM}}(\omega)$ denote this spectrum. The determination of $S_{\text{MEM}}(\omega)$ is linked with the characterization of a prediction-error filter of order M, as shown by

$$S_{\text{MEM}}(\omega) = \frac{P_M}{\left|1 + \sum_{k=1}^{M} a_{M,k}^* e^{-\text{j}k\omega}\right|^2}, \tag{3.102}$$

where the $a_{M,k}$ are the prediction-error filter coefficients and P_M is the prediction-error power, all of which correspond to a prediction order M.

In view of the one-to-one correspondence that exists between the prediction-error filter of Fig. 3.7(a) and the AR model of Fig. 3.7(b), we have

$$a_{M,k} = -w_{ok}, \quad k = 1, 2, \ldots, M \tag{3.103}$$

and

$$P_M = \sigma_\nu^2. \tag{3.104}$$

Accordingly, the formulas given in Eqs. (3.101) and (3.102) are one and the same. In other words, for the case of a wide-sense stationary process, the AR spectrum (for model order M) and the MEM spectrum (for prediction order M) are indeed *equivalent* (Van den Bos, 1971).

3.7 CHOLESKY FACTORIZATION

Consider next a stack of backward prediction-error filters of orders 0 through M, which are connected in parallel as in Fig. 3.8. The filters are all excited at time n by the same input, denoted by $u(n)$. Note that for the case of zero prediction order, we simply have a direct connection, as shown at the top end of Fig. 3.8. Let $b_0(n), b_1(n), \ldots, b_M(n)$ denote the sequence of backward prediction errors produced by these filters. Building on Fig. 3.2(c), we may express these errors in terms of the respective filter inputs and filter coefficients as follows [see Fig. 3.2(c)]:

$$\begin{aligned}
b_0(n) &= u(n),\\
b_1(n) &= a_{1,1}u(n) + a_{1,0}u(n-1),\\
b_2(n) &= a_{2,2}u(n) + a_{2,1}u(n-1) + a_{2,0}u(n-2),\\
&\vdots\\
b_M(n) &= a_{M,M}u(n) + a_{M,M-1}u(n-1) + \cdots + a_{M,0}u(n-M).
\end{aligned}$$

Combining this system of $(M+1)$ simultaneous linear equations into a compact matrix form, we have

$$\mathbf{b}(n) = \mathbf{L}\mathbf{u}(n), \tag{3.105}$$

where

$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M)]^{\mathrm{T}}$$

is the $(M+1)$-by-1 input vector;

$$\mathbf{b}(n) = [b_0(n), b_1(n), \ldots, b_M(n)]^{\mathrm{T}}$$

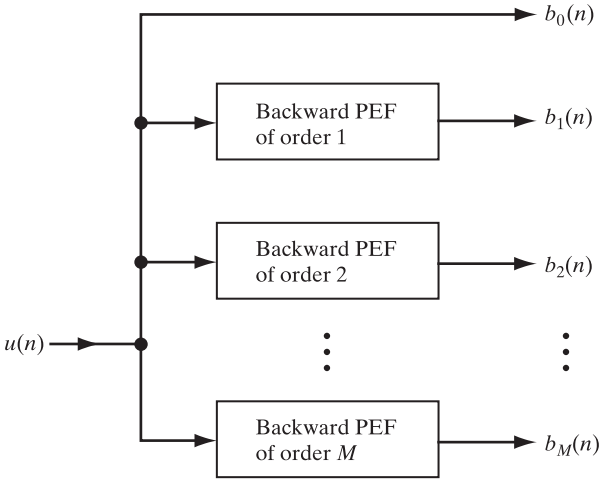


FIGURE 3.8 Parallel connection of stack of backward prediction-error filters (PEF) of orders 0 through M. (*Note*: The direct connection represents a PEF with a single coefficient equal to unity.)

is the $(M + 1)$-by-1 output vector of backward prediction errors. The $(M + 1)$-by-$(M + 1)$ coefficient matrix, $\mathbf{L}$, on the right-hand side of Eq. (3.105) is defined in terms of the backward prediction-error filter coefficients of orders 0 to M as

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ a_{1,1} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,M} & a_{M,M-1} & \cdots & 1 \end{bmatrix}, \tag{3.106}$$

which has three useful properties:

1. The matrix $\mathbf{L}$ is a *lower triangular matrix* with 1's along its main diagonal; all of its elements above the main diagonal are zero.
2. The determinant of matrix $\mathbf{L}$ is unity; hence, it is nonsingular (i.e., its inverse exists).
3. The nonzero elements of each row of the matrix $\mathbf{L}$, except for complex conjugation, equal the weights of a backward prediction-error filter whose order corresponds to the position of that row in the matrix.

Equation (3.105) is known as the *Gram–Schmidt orthogonalization algorithm*.[5] According to this algorithm, there is a *one-to-one correspondence* between the input vector $\mathbf{u}(n)$ and the backward prediction-error vector $\mathbf{b}(n)$. In particular, given $\mathbf{u}(n)$, we may obtain $\mathbf{b}(n)$ by using Eq. (3.105). Conversely, given $\mathbf{b}(n)$, we may obtain the corresponding vector $\mathbf{u}(n)$ by using the inverse of Eq. (3.105), as shown by

$$\mathbf{u}(n) = \mathbf{L}^{-1}\mathbf{b}(n), \tag{3.107}$$

where $\mathbf{L}^{-1}$ is the inverse of the matrix $\mathbf{L}$.

Orthogonality of Backward Prediction Errors

The backward prediction errors $b_0(n), b_1(n), \ldots, b_M(n)$ constituting the elements of the vector $\mathbf{b}(n)$ have an important property: *They are all orthogonal to each other,* as shown by

$$\mathbb{E}[b_m(n)b_i^*(n)] = \begin{cases} P_m, & i = m \\ 0, & i \neq m \end{cases}. \tag{3.108}$$

To derive this property, we may proceed as follows: First of all, without loss of generality, we may assume that $m \geq i$. Then we express the backward prediction error $b_i(n)$ in terms of the input $u(n)$ as the linear convolution sum

$$b_i(n) = \sum_{k=0}^{i} a_{i,i-k}u(n-k), \tag{3.109}$$

which is nothing more than Eq. (3.37) with prediction order i used in place of M. Hence, using this relation to evaluate the expectation of $b_m(n)b_i^*(n)$, we get

[5]For a full discussion of the Gram–Schmidt algorithm and the various methods for its implementation, see Haykin (2013).

$$\mathbb{E}[b_m(n)b_i^*(n)] = \mathbb{E}\left[b_m(n)\sum_{k=0}^{i} a_{i,i-k}^* u^*(n-k) \right]. \tag{3.110}$$

From the principle of orthogonality, we have

$$\mathbb{E}[b_m(n)u^*(n-k)] = 0, \quad 0 \le k \le m-1. \tag{3.111}$$

For $m > i$ and $0 \le k \le i$, we therefore find that all the expectation terms inside the summation on the right-hand side of Eq. (3.110) are zero. Correspondingly,

$$\mathbb{E}[b_m(n)b_i^*(n)] = 0, \qquad m \neq i.$$

When $m = i$, Eq. (3.110) reduces to

$$\begin{aligned}\mathbb{E}[b_m(n)b_i^*(n)] &= \mathbb{E}[b_m(n)b_m^*(n)] \\ &= P_m, \qquad m = 1.\end{aligned}$$

Hence, Eq. (3.108) is obtained. It is important, however, to note that it does so only for wide-sense stationary input data.

We thus see that the Gram–Schmidt orthogonalization algorithm given by Eq. (3.105) transforms the input vector $\mathbf{u}(n)$ consisting of correlated samples into an equivalent vector $\mathbf{b}(n)$ of uncorrelated backward prediction errors.[6]

Factorization of the Inverse of Correlation Matrix R

Having equipped ourselves with the knowledge that backward prediction errors are indeed orthogonal to each other, we may return to the transformation described by the Gram–Schmidt algorithm in Eq. (3.105). Specifically, using this transformation, we may express the correlation matrix of the backward prediction-error vector $\mathbf{b}(n)$ in terms of the correlation matrix of the input vector $\mathbf{u}(n)$ as follows:

$$\begin{aligned}\mathbb{E}[\mathbf{b}(n)\mathbf{b}^{\mathrm{H}}(n)] &= \mathbb{E}[\mathbf{L}\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{L}^{\mathrm{H}}] \\ &= \mathbf{L}\mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]\mathbf{L}^{\mathrm{H}}.\end{aligned} \tag{3.112}$$

Let

$$\mathbf{D} = \mathbb{E}[\mathbf{b}(n)\mathbf{b}^{\mathrm{H}}(n)] \tag{3.113}$$

[6]Two random variables X and Y are said to be *orthogonal* to each other if

$$\mathbb{E}[XY^*] = 0$$

and are said to be *uncorrelated* with each other if

$$\mathbb{E}[X - \mathbb{E}[X])(Y - \mathbb{E}[Y])^*] = 0.$$

If one or both of X and Y have zero mean, then these two conditions become one and the same. For the discussion presented herein, the input data, and therefore the backward predicted errors, are assumed to have zero mean. Under this assumption, we may interchange orthogonality with uncorrelatedness when we refer to backward prediction errors.

denote the correlation matrix of the backward prediction-error vector $\mathbf{b}(n)$. As before, the correlation matrix of the input vector $\mathbf{u}(n)$ is denoted by $\mathbf{R}$. We may therefore rewrite Eq. (3.112) as

$$\mathbf{D} = \mathbf{L}\mathbf{R}\mathbf{L}^{\mathrm{H}}. \tag{3.114}$$

We now make two observations:

1. When the correlation matrix $\mathbf{R}$ of the input vector $\mathbf{u}(n)$ is positive definite and its inverse exists, the correlation matrix $\mathbf{D}$ of the backward prediction-error vector $\mathbf{b}(n)$ is also positive definite and, likewise, its inverse exists.
2. The correlation matrix $\mathbf{D}$ is a diagonal matrix, because $\mathbf{b}(n)$ consists of elements that are all orthogonal to each other. In particular, we may express $\mathbf{D}$ in the expanded form

$$\mathbf{D} = \mathrm{diag}(P_0, P_1, \ldots, P_M), \tag{3.115}$$

where P_i is the average power of the ith backward prediction error $b_i(n)$; that is,

$$P_i = \mathbb{E}[|b_i(n)|^2], \qquad i = 0, 1, \ldots, M. \tag{3.116}$$

The inverse of matrix $\mathbf{D}$ is also a diagonal matrix; that is,

$$\mathbf{D}^{-1} = \mathrm{diag}(P_0^{-1}, P_1^{-1}, \ldots, P_M^{-1}). \tag{3.117}$$

Accordingly, we may use Eq. (3.114) to express the inverse of the correlation matrix $\mathbf{R}$ as

$$\begin{aligned}\mathbf{R}^{-1} &= \mathbf{L}^{\mathrm{H}}\mathbf{D}^{-1}\mathbf{L} \\ &= (\mathbf{D}^{-1/2}\mathbf{L})^{\mathrm{H}}(\mathbf{D}^{-1/2}\mathbf{L}),\end{aligned} \tag{3.118}$$

which is the desired result.

The inverse matrix $\mathbf{D}^{-1}$, in the first line of Eq. (3.118), is a diagonal matrix defined by Eq. (3.117). The matrix $\mathbf{D}^{-1/2}$, the *square root* of $\mathbf{D}^{-1}$, in the second line of Eq. (3.118), is also a diagonal matrix, defined by

$$\mathbf{D}^{-1/2} = \mathrm{diag}(P_0^{-1/2}, P_1^{-1/2}, \ldots, P_M^{-1/2}).$$

The transformation expressed in Eq. (3.118) is called the *Cholesky factorization of the inverse matrix* $\mathbf{R}^{-1}$ (Stewart, 1973). Note that the matrix $\mathbf{D}^{-1/2}\mathbf{L}$ is a lower triangular matrix; however, it differs from the lower triangular matrix $\mathbf{L}$ of Eq. (3.106) in that its diagonal elements are different from unity. Note also that the Hermitian-transposed matrix product $(\mathbf{D}^{-1/2}\mathbf{L})^{\mathrm{H}}$ is an upper triangular matrix whose diagonal elements are different from unity. Thus:

> *According to the Cholesky factorization, the inverse correlation matrix* $\mathbf{R}^{-1}$ *may be factored into the product of an upper triangular matrix and a lower triangular matrix that are the Hermitian transpose of each other.*

3.8 LATTICE PREDICTORS

To implement the Gram–Schmidt algorithm of Eq. (3.105) for transforming an input vector $\mathbf{u}(n)$ consisting of correlated samples into an equivalent vector $\mathbf{b}(n)$ consisting of uncorrelated backward prediction errors, we may use the parallel connection of a direct

path and an appropriate number of backward prediction-error filters, as illustrated in Fig. 3.8. The vectors $\mathbf{b}(n)$ and $\mathbf{u}(n)$ are said to be "equivalent" in the sense that they contain the same amount of information. (See Problem 21.) A much more efficient method of implementing the Gram–Schmidt orthogonalization algorithm, however, is to use an order-recursive structure in the form of a ladder, known as a *lattice predictor*. This system combines several forward and backward prediction-error filtering operations into a single structure. Specifically, a lattice predictor consists of a cascade connection of elementary units (stages), all of which have a structure similar to that of a lattice—hence the name. The number of stages in a lattice predictor equals the prediction order. Thus, for a prediction-error filter of order m, there are m stages in the lattice realization of the filter.

Order-Update Recursions for the Prediction Errors

The input–output relations that characterize a lattice predictor may be derived in various ways, depending on the particular form in which the Levinson–Durbin algorithm is utilized. For the derivation presented here, we start with the matrix formulations of this algorithm given by Eqs. (3.40) and (3.42) that pertain, respectively, to the forward and backward operations of a prediction-error filter. For convenience of presentation, we reproduce these two relations here:

$$\mathbf{a}_m = \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} + \kappa_m \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix}; \tag{3.119}$$

$$\mathbf{a}_m^{B*} = \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} + \kappa_m^* \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix}. \tag{3.120}$$

The $(m + 1)$-by-1 vector $\mathbf{a}_m$ and the m-by-1 vector $\mathbf{a}_{m-1}$ refer to forward prediction-error filters of order m and $m - 1$, respectively. The $(m + 1)$-by-1 vector $\mathbf{a}_m^{B*}$ and the m-by-1 vector $\mathbf{a}_{m-1}^{B*}$ refer to the corresponding backward prediction-error filters of order m and $m - 1$, respectively. The scalar κ_m is the associated reflection coefficient.

Consider first the forward prediction-error filter of order m, with its tap inputs denoted by $u(n), u(n-1), \ldots, u(n-m)$. We may partition $\mathbf{u}_{m+1}(n)$, the $(m + 1)$-by-1 tap-input vector of this filter, in the form

$$\mathbf{u}_{m+1}(n) = \begin{bmatrix} \mathbf{u}_m(n) \\ \hdashline u(n-m) \end{bmatrix} \tag{3.121}$$

or in the equivalent form

$$\mathbf{u}_{m+1}(n) = \begin{bmatrix} u(n) \\ \hdashline \mathbf{u}_m(n-1) \end{bmatrix}. \tag{3.122}$$

Next, we form the inner product of the $(m + 1)$-by-1 vectors $\mathbf{a}_m$ and $\mathbf{u}_{m+1}(n)$ by premultiplying $\mathbf{u}_{m+1}(n)$ by the Hermitian transpose of $\mathbf{a}_m$. Thus, using Eq. (3.119) for $\mathbf{a}_m$, we may treat the terms resulting from this multiplication as follows:

1. For the left-hand side of Eq. (3.119), premultiplication of $\mathbf{u}_{m+1}(n)$ by $\mathbf{a}_m^H$ yields

$$f_m(n) = \mathbf{a}_m^H \mathbf{u}_{m+1}(n), \tag{3.123}$$

where $f_m(n)$ is the forward prediction error produced at the output of the forward prediction-error filter of order m.

2. For the first term on the right-hand side of Eq. (3.119), we use the partitioned form of $\mathbf{u}_{m+1}(n)$ given in Eq. (3.121). We write

$$\begin{aligned}\left[\mathbf{a}_{m-1}^{\mathrm{H}} \,\vdots\, 0\right]\mathbf{u}_{m+1}(n) &= \left[\mathbf{a}_{m-1}^{\mathrm{H}} \,\vdots\, 0\right]\begin{bmatrix}\mathbf{u}_m(n)\\ \cdots\\ u(n-m)\end{bmatrix}\\ &= \mathbf{a}_{m-1}^{\mathrm{H}}\mathbf{u}_m(n)\\ &= f_{m-1}(n),\end{aligned} \tag{3.124}$$

where $f_{m-1}(n)$ is the forward prediction error produced at the output of the forward prediction-error filter of order $m-1$.

3. For the second matrix term on the right-hand side of Eq. (3.119), we use the partitioned form of $\mathbf{u}_{m+1}(n)$ given in Eq. (3.122). We write

$$\begin{aligned}\left[0 \,\vdots\, \mathbf{a}_{m-1}^{\mathrm{BT}}\right]\mathbf{u}_{m+1}(n) &= \left[0 \,\vdots\, \mathbf{a}_{m-1}^{\mathrm{BT}}\right]\begin{bmatrix}u(n)\\ \cdots\\ \mathbf{u}_m(n-1)\end{bmatrix}\\ &= \mathbf{a}_{m-1}^{\mathrm{BT}}\mathbf{u}_m(n-1)\\ &= b_{m-1}(n-1),\end{aligned} \tag{3.125}$$

where $b_{m-1}(n-1)$ is the *delayed* backward prediction error produced at the output of the backward prediction-error filter of order $m-1$.

Combining the results of the multiplications, described by Eqs. (3.123), (3.124), and (3.125), we may thus write

$$f_m(n) = f_{m-1}(n) + \kappa_m^* b_{m-1}(n-1). \tag{3.126}$$

Consider next the backward prediction-error filter of order m, with its tap inputs denoted by $u(n), u(n-1), \ldots, u(n-m)$. Here, again, we may express $\mathbf{u}_{m+1}(n)$, the $(m+1)$-by-1 tap-input vector of this filter, in the partitioned form of Eq. (3.121) or that of Eq. (3.122). In this case, the terms resulting from the formation of the inner product of the vectors $\mathbf{a}_m^{\mathrm{B}*}$ and $\mathbf{u}_{m+1}(n)$ are treated as follows:

1. For the left-hand side of Eq. (3.120), premultiplication of $\mathbf{u}_{m+1}(n)$ by the Hermitian transpose of $\mathbf{a}_m^{\mathrm{B}*}$ yields

$$b_m(n) = \mathbf{a}_m^{\mathrm{BT}}\mathbf{u}_{m+1}(n), \tag{3.127}$$

where $b_m(n)$ is the backward prediction error produced at the output of the backward prediction-error filter of order m.

2. For the first term on the right-hand side of Eq. (3.120), we use the partitioned form of the tap-input vector $\mathbf{u}_{m+1}(n)$ given in Eq. (3.122). Multiplying the Hermitian transpose of this first term by $\mathbf{u}_{m+1}(n)$, we get

$$\begin{aligned}\left[0 \,\vdots\, \mathbf{a}_{m-1}^{\mathrm{BT}}\right]\mathbf{u}_{m+1}(n) &= \left[0 \,\vdots\, \mathbf{a}_{m-1}^{\mathrm{BT}}\right]\begin{bmatrix}u(n)\\ \cdots\\ \mathbf{u}_m(n-1)\end{bmatrix}\\ &= \mathbf{a}_{m-1}^{\mathrm{BT}}\mathbf{u}_m(n-1)\\ &= b_{m-1}(n-1).\end{aligned} \tag{3.128}$$

3. For the second matrix term on the right-hand side of Eq. (3.120), we use the partitioned form of the tap-input vector $\mathbf{u}_{m+1}(n)$ given in Eq. (3.121). Multiplying the Hermitian transpose of this second term by $\mathbf{u}_{m+1}(n)$, we get

$$\begin{aligned}\left[\mathbf{a}_{m-1}^{\mathrm{H}} \;\vdots\; 0\right]\mathbf{u}_{m+1}(n) &= \left[\mathbf{a}_{m-1}^{\mathrm{H}} \;\vdots\; 0\right]\left[\begin{array}{c}\mathbf{u}_m(n)\\ \hdashline u(n-m)\end{array}\right]\\ &= \mathbf{a}_{m-1}^{\mathrm{H}}\mathbf{u}_m(n)\\ &= f_{m-1}(n).\end{aligned} \tag{3.129}$$

Combining the results of Eqs. (3.127), (3.128), and (3.129), we find that the inner product of $\mathbf{a}_m^{\mathrm{B}*}$ and $\mathbf{u}_{m+1}(n)$ yields

$$b_m(n) = b_{m-1}(n-1) + \kappa_m f_{m-1}(n). \tag{3.130}$$

Equations (3.126) and (3.130) are the sought-after pair of *order-update recursions* that characterize stage m of the lattice predictor. They are reproduced here in matrix form as

$$\begin{bmatrix} f_m(n) \\ b_m(n) \end{bmatrix} = \begin{bmatrix} 1 & \kappa_m^* \\ \kappa_m & 1 \end{bmatrix} \begin{bmatrix} f_{m-1}(n) \\ b_{m-1}(n-1) \end{bmatrix}, \qquad m = 1, 2, \ldots, M. \tag{3.131}$$

We may view $b_{m-1}(n-1)$ as the result of applying the unit-delay operator z^{-1} to the backward prediction error $b_{m-1}(n)$; that is,

$$b_{m-1}(n-1) = z^{-1}[b_{m-1}(n)]. \tag{3.132}$$

Thus, using Eqs. (3.131) and (3.132), we may represent stage m of the lattice predictor by the signal-flow graph shown in Fig. 3.9(a). Except for the branch pertaining to the block labeled z^{-1}, this signal-flow graph has the appearance of a lattice—hence the name "lattice predictor."[7] Note that the parameterization of stage m of the lattice predictor is uniquely defined by the reflection coefficient κ_m.

For the elementary case of $m = 0$, we get the *initial conditions*

$$f_0(n) = b_0(n) = u(n), \tag{3.133}$$

where $u(n)$ is the input signal at time n. Therefore, starting with $m = 0$ and progressively increasing the order of the filter by 1, we obtain the *lattice equivalent model*

[7]The first application of lattice filters in on-line adaptive signal processing was apparently made by Itakura and Saito (1971) in the field of speech analysis. Equivalent lattice-filter models, however, were familiar in geophysical signal processing as "layered earth models" (Robinson, 1957; Burg, 1968). It is also of interest to note that such lattice filters have been well studied in network theory, especially in the cascade synthesis of multiport networks (Dewilde, 1969).

Actually, there is another structure for building a linear predictor that is based on the *Schur algorithm* (Schur, 1917). Like the Levinson–Durbin algorithm, the Schur algorithm provides a procedure for computing the reflection coefficients from a known autocorrelation sequence. The Schur algorithm, however, lends itself to parallel implemetation, with the result that it achieves a throughput rate higher than that obtained using the Levinson–Durbin algorithm. For a discussion of the Schur algorithm, including mathematical details and considerations revolving about its implementation, see Haykin (1989a).

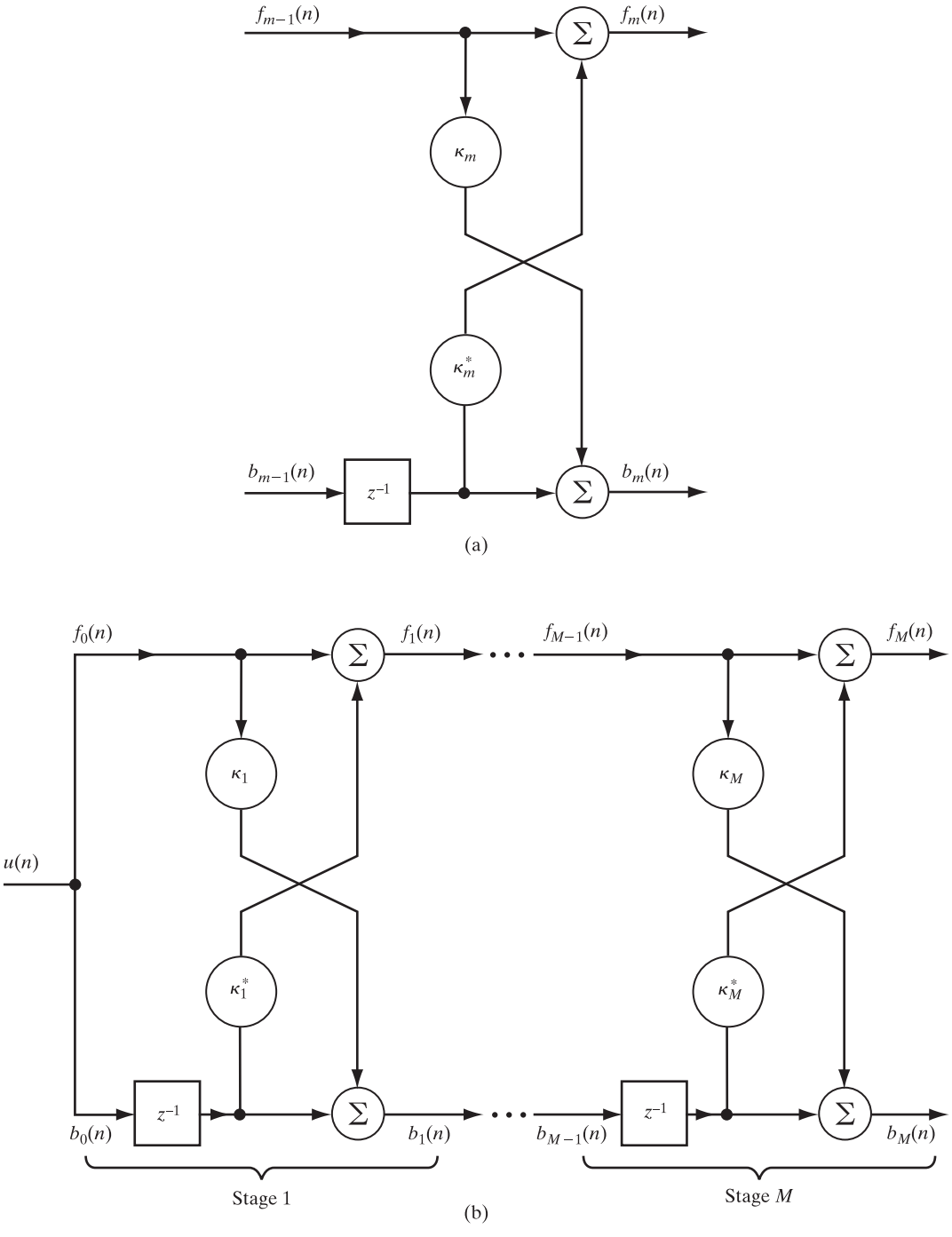


FIGURE 3.9 (a) Signal-flow graph for stage m of a lattice predictor; (b) lattice equivalent model of prediction-error (all-zero) filter of order M.

shown in Fig. 3.9(b) for a prediction-error filter of final order M. In this figure, we merely require knowledge of the complete set of reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$, one for each stage of the filter.

The lattice filter, depicted in Fig. 3.9(b), offers the following attractive features:

1. A lattice filter is a highly efficient structure for generating the sequence of forward prediction errors and the corresponding sequence of backward prediction errors simultaneously.
2. The various stages of a lattice predictor are "decoupled" from each other. This decoupling property was indeed derived in Section 3.7, where it was shown that the backward prediction errors produced by the various stages of a lattice predictor are "orthogonal" to each other for wide-sense stationary input data.
3. The lattice filter is *modular* in structure; hence, if we are required to increase the order of the predictor, we simply add one or more stages (as desired) without affecting earlier computations.

3.9 ALL-POLE, ALL-PASS LATTICE FILTER

The multistage lattice predictor of Fig. 3.9(b) combines two *all-zero* prediction-error filters into a single structure. More specifically, invoking Properties 3 and 4 of prediction-error filters presented in Section 3.4, we may respectively make the following statements:

- The path from the common input $u(n)$ to the forward prediction error $f_M(n)$ is a minimum-phase all-zero filter.
- The alternative path from the common input $u(n)$ to the backward prediction error $b_M(n)$ is a maximum-phase all-zero filter.

The multistage lattice predictor of Fig. 3.9(b) may be rewired to operate as a combined *all-pole, all-pass* lattice filter. To do this rewiring, we first rearrange terms in Eq. (3.126) to obtain

$$f_{m-1}(n) = f_m(n) - \kappa_m^* b_{m-1}(n-1), \tag{3.134}$$

where the forward prediction error $f_m(n)$ is now treated as an input variable for the mth stage of the rewired lattice filter. However, Eq. (3.130) is left intact and reproduced here for convenience of presentation:

$$b_m(n) = b_{m-1}(n-1) + \kappa_m f_{m-1}(n). \tag{3.135}$$

Equations (3.134) and (3.135) define the input–output relations of the mth stage in the rewired lattice filter. Thus, starting with the initial conditions of Eq. (3.133) corresponding to order $m = 0$ and progressively increasing the order of the filter, we obtain the rewired multistage lattice filter of Fig. 3.10. Note, however, that the signal $u(n)$ produced at the common terminal in Fig. 3.10 is the output, whereas $u(n)$ acts as the input signal in Fig. 3.9(b). By the same token, $f_m(n)$ acts as the input signal in Fig. 3.10, whereas it is an output signal in Fig. 3.9(b).

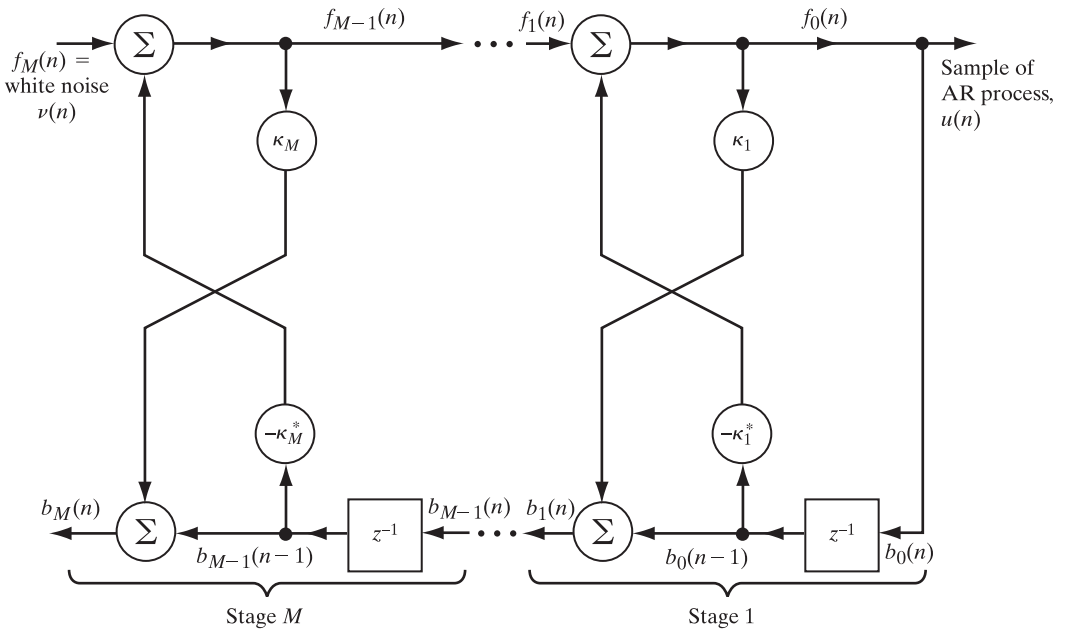


FIGURE 3.10 All-pole, all-pass lattice filter of order M.

To excite the multistage lattice filter of Fig. 3.10, a sequence of samples taken from white noise $\nu(n)$ is used as the input signal $f_M(n)$. The path from the input $f_m(n)$ to the output $u(n)$ in Fig. 3.10 is an *all-pole filter* of order M. Hence, in light of the discussion presented in Section 3.6 on the autoregressive modeling of a stationary process, we may view this all-pole filter as a *synthesizer* and the corresponding all-zero lattice predictor of Fig. 3.9(b) as an *analyzer*.

The backward prediction error $b_M(n)$ constitutes the second output of the filter in Fig. 3.10. The path from the input $f_M(n)$ to the output $b_M(n)$ is an all-pass filter of order M. The poles and zeros of this filter are the inverse of each other with respect to the unit circle in the z-plane. (See Problem 18.) Thus, the rewired multistage lattice filter of Fig. 3.10 combines an all-pole filter and an all-pass filter in a single structure.

EXAMPLE 5

Consider the two-stage all-pole lattice filter of Fig. 3.11. There are four possible paths that can contribute to the makeup of the output $u(n)$, as illustrated in Fig. 3.12. In particular, we have

$$\begin{aligned} u(n) &= \nu(n) - \kappa_1^* u(n-1) - \kappa_1\kappa_2^* u(n-1) - \kappa_2^* u(n-2) \\ &= \nu(n) - (\kappa_1^* + \kappa_1\kappa_2^*)u(n-1) - \kappa_2^* u(n-2). \end{aligned}$$

From Example 2, we recall that

$$a_{2,1} = \kappa_1 + \kappa_1^*\kappa_2$$

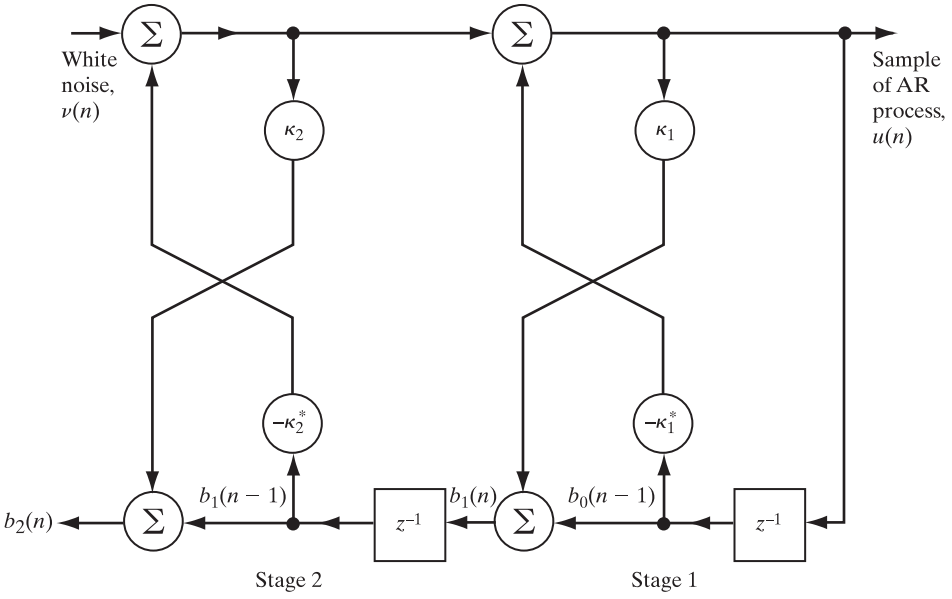


FIGURE 3.11 All-pole lattice filter of order 2.

and

$$a_{2,2} = \kappa_2.$$

We may therefore express the mechanism governing the generation of process $u(n)$ as

$$u(n) + a_{2,1}^* u(n-1) + a_{2,2}^* u(n-2) = v(n),$$

which is recognized as the difference equation of a second-order AR process.

3.10 JOINT-PROCESS ESTIMATION

In this section, we use the lattice predictor as a subsystem to solve a *joint-process estimation problem* that is optimal in the mean-square-error sense (Griffiths, 1978; Makhoul, 1978). In particular, we consider the minimum mean-square-error estimation of a process $d(n)$, termed the *desired response*, by using a set of *observables* derived from a related process $u(n)$. We assume that the processes $d(n)$ and $u(n)$ are jointly stationary. This estimation problem is similar to that considered in Chapter 2, with one basic difference: There we directly used samples of the process $u(n)$ as the observables, whereas our approach here is different in that, for the observables, we use the set of backward prediction errors obtained by feeding the input of a multistage lattice predictor with samples of the process $u(n)$. The fact that the backward prediction errors are orthogonal to each other simplifies the solution to the problem significantly.

The structure of the *joint-process estimator* is shown in Fig. 3.13. This system performs two optimum estimations jointly:

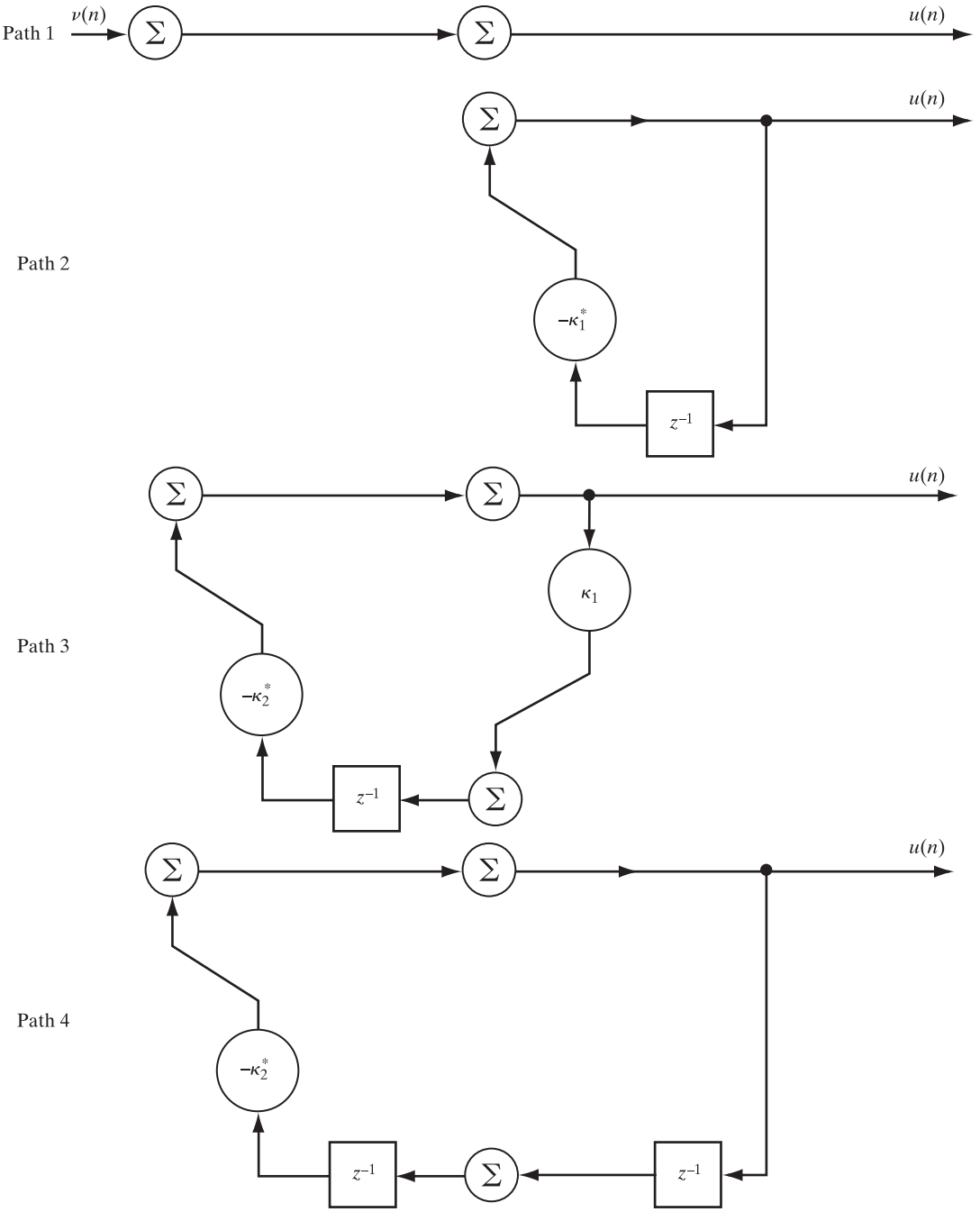


FIGURE 3.12 The four possible paths that contribute to the makeup of the output $u(n)$ in the all-pole lattice inverse filter of Fig. 3.11.

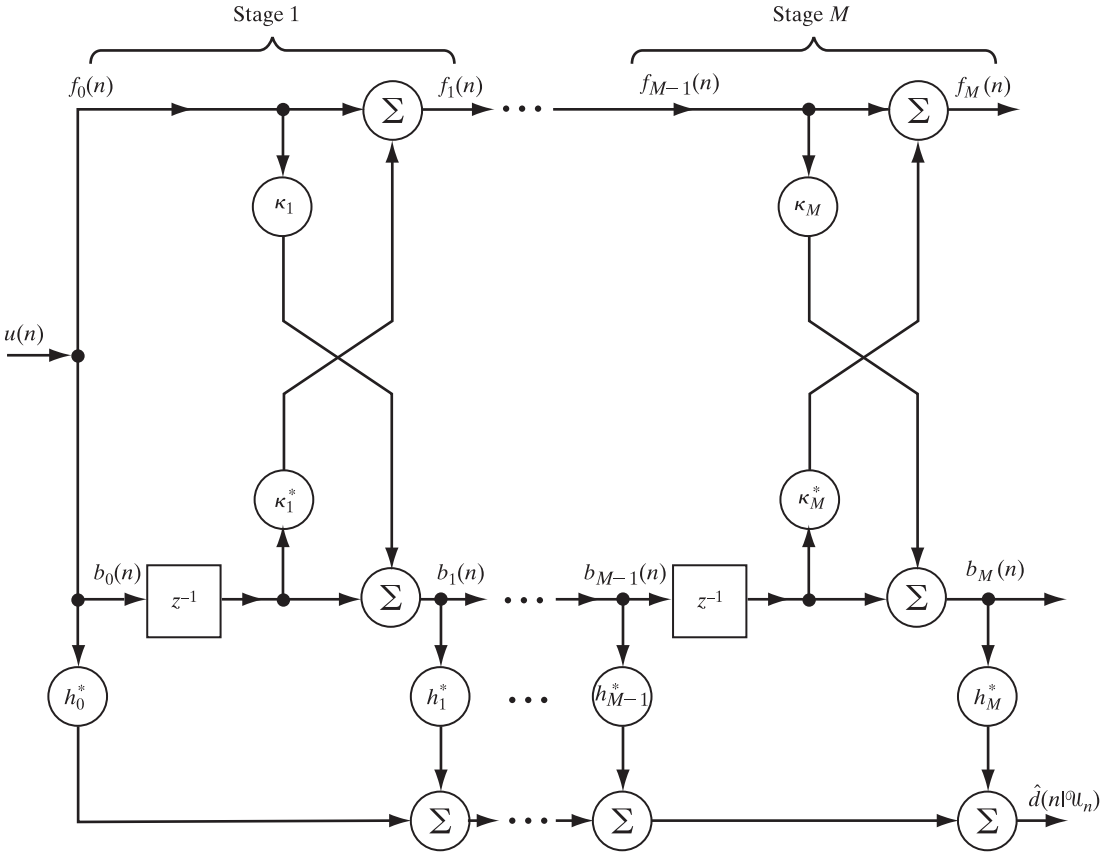


FIGURE 3.13 Lattice-based structure for joint-process estimation.

1. The *lattice predictor*, consisting of a cascade of M stages characterized individually by the reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$, performs predictions (of varying orders) on the input. In particular, it transforms the sequence of (correlated) input samples $u(n), u(n-1), \ldots, u(n-M)$ into a corresponding sequence of (uncorrelated) backward prediction errors $b_0(n), b_1(n), \ldots, b_M(n)$.

2. The multiple regression filter, characterized by the set of weights $h_0, h_1, \ldots, h_M$, operates on the sequence of backward prediction errors $b_0(n), b_1(n), \ldots, b_M(n)$ as inputs, respectively, to produce an estimate of the desired response $d(n)$. The resulting estimate is defined as the sum of the respective scalar inner products of these two sets of quantities; that is,

$$\hat{d}(n|\mathcal{U}_n) = \sum_{i=0}^{M} h_i^* b_i(n), \tag{3.136}$$

where $\mathcal{U}_n$ is the space spanned by the inputs $u(n), u(n-1), \ldots, u(n-M)$. We may rewrite Eq. (3.136) in matrix form as

$$\hat{d}(n|\mathcal{U}_n) = \mathbf{h}^{\mathrm{H}}\mathbf{b}(n), \tag{3.137}$$

where

$$\mathbf{h} = [h_0, h_1, \ldots, h_M]^{\mathrm{T}} \tag{3.138}$$

is an $(M+1)$-by-1 vector. We refer to $h_0, h_1, \ldots, h_M$ as the *regression coefficients* of the estimator and to $\mathbf{h}$ as the *regression-coefficient vector*.

Let $\mathbf{D}$ denote the $(M+1)$-by-$(M+1)$ correlation matrix of $\mathbf{b}(n)$, the $(M+1)$-by-1 vector of backward prediction errors $b_0(n), b_1(n), \ldots, b_M(n)$. Let $\mathbf{z}$ denote the $(M+1)$-by-1 cross-correlation vector between the backward prediction errors and the desired response. Then

$$\mathbf{z} = \mathbb{E}[\mathbf{b}(n)d^*(n)]. \tag{3.139}$$

Therefore, applying the Wiener–Hopf equations to our present situation, we find that the optimum regression-coefficient vector $\mathbf{h}_o$ is defined by

$$\mathbf{D}\mathbf{h}_o = \mathbf{z}. \tag{3.140}$$

Solving for $\mathbf{h}_o$, we get

$$\mathbf{h}_o = \mathbf{D}^{-1}\mathbf{z}, \tag{3.141}$$

where the inverse matrix $\mathbf{D}^{-1}$ is a diagonal matrix defined in terms of various prediction-error powers, as in Eq. (3.117). Note that, unlike the ordinary FIR filter realization of the Wiener filter, the computation of $\mathbf{h}_o$ in the joint-process estimator of Fig. 3.12 is relatively simple to accomplish.

Relationship between the Optimum Regression-Coefficient Vector and the Wiener Solution

From the Cholesky factorization given in Eq. (3.118), we deduce that

$$\mathbf{D}^{-1} = \mathbf{L}^{-\mathrm{H}}\mathbf{R}^{-1}\mathbf{L}^{-1}. \tag{3.142}$$

Hence, substituting Eq. (3.142) into Eq. (3.141) yields

$$\mathbf{h}_o = \mathbf{L}^{-\mathrm{H}}\mathbf{R}^{-1}\mathbf{L}^{-1}\mathbf{z}. \tag{3.143}$$

Moreover, from Eq. (3.105), we note that

$$\mathbf{b}(n) = \mathbf{L}\mathbf{u}(n). \tag{3.144}$$

Therefore, substituting Eq. (3.144) into Eq. (3.139) yields

$$\begin{aligned}\mathbf{z} &= \mathbf{L}\mathbb{E}[\mathbf{u}(n)d^*(n)] \\ &= \mathbf{L}\mathbf{p},\end{aligned} \tag{3.145}$$

where $\mathbf{p}$ is the cross-correlation vector between the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$. Thus, using Eq. (3.145) in Eq. (3.143), we finally obtain

$$\begin{aligned}\mathbf{h}_o &= \mathbf{L}^{-\mathrm{H}}\mathbf{R}^{-1}\mathbf{L}^{-1}\mathbf{L}\mathbf{p} \\ &= \mathbf{L}^{-\mathrm{H}}\mathbf{R}^{-1}\mathbf{p} \\ &= \mathbf{L}^{-\mathrm{H}}\mathbf{w}_o, \end{aligned} \tag{3.146}$$

where $\mathbf{L}$ is a lower triangular matrix defined in terms of the equivalent forward prediction-error filter coefficients, as in Eq. (3.106). Equation (3.146) is the sought-after relationship between the optimum regression-coefficient vector $\mathbf{h}_o$ and the Wiener solution $\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p}$.

3.11 PREDICTIVE MODELING OF SPEECH

The linear prediction theory developed in this chapter finds application in the linear predictive coding of speech and video signals. In this section, we confine the discussion to the coding of speech.

Linear predictive coding of speech exploits the special properties of a classical model of the speech-production process. Figure 3.14 shows a simplified block diagram of this model. The model assumes that the sound-generating mechanism (i.e., the source of excitation) is linearly separable from the intelligence-modulating vocal-tract filter. The precise form of the excitation depends on whether the speech sound is voiced or unvoiced:

- A *voiced speech* sound (such as /*i*/ in *e*ve) is generated from a quasi-periodic excitation of the vocal tract. (The symbol /·/ is used to denote the *phoneme* as a basic

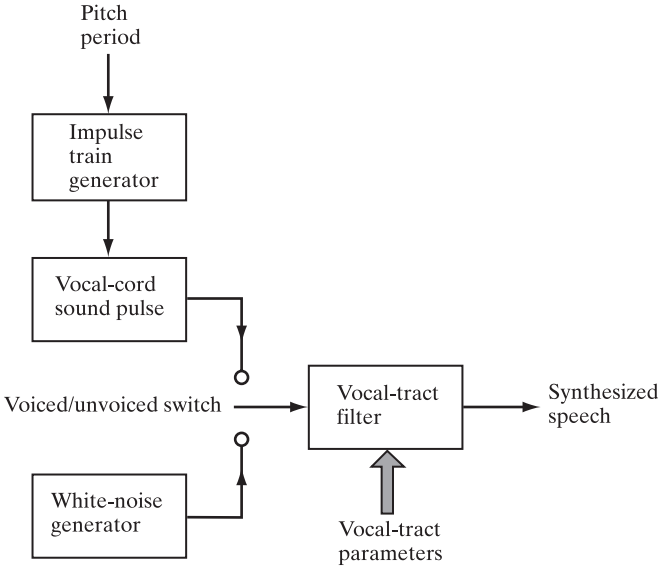


FIGURE 3.14 Block diagram of simplified model for the speech-production process.

linguistic unit.) In the model of Fig. 3.14, the impulse-train generator produces a sequence of impulses (i.e., very short pulses) that are spaced by a fundamental period equal to the pitch period. This signal, in turn, excites a linear filter whose frequency response determines the identity of the sound.

- An *unvoiced speech* sound (such as /*f*/ in *f*ish) is generated from random sounds produced by turbulent airflow through a constriction along the vocal tract. In this second scenario, the excitation consists simply of a white-noise source (i.e., a source with a broad spectrum). The probability distribution of the noise samples does not seem to be of critical importance.

The short-time spectrum[8] of the speech signal is obtained by multiplying the spectrum of the source, be it voiced or unvoiced, by the frequency response of the vocal-tract filter. Whether the source is a periodic sequence of pulses or white noise, its spectral envelope is flat. Therefore, the short-time spectral envelope of the speech signal is determined by the frequency response of the vocal-tract filter.

The method of *linear predictive coding* (LPC) is a model-dependent form of source coding that is widely used for the digital representation of speech at low bit rates (2.4 kb/s or less). LPC also provides a procedure for producing accurate estimates of basic speech parameters. The development of LPC relies on the model of Fig. 3.14, in which the vocal-tract filter is represented by the all-pole transfer function

$$H(z) = \frac{G}{1 + \sum_{k=1}^{M} a_k z^{-k}}, \tag{3.147}$$

where G is a gain factor. The form of excitation applied to this filter is changed by switching between voiced and unvoiced sounds. Thus, the filter with transfer function $H(z)$ is excited by a sequence of impulses to generate voiced sounds or a white-noise sequence to generate unvoiced sounds. In this application, the data are real valued; hence, the filter coefficients a_k are likewise real valued.

[8]The *short-time spectrum*, or more precisely, the *short-time Fourier transform*, of a continuous-time signal $x(t)$ is obtained by first multiplying the signal by a *window function* $h(t)$, centered at time t, to produce the modified signal

$$x_t(\tau) = x(\tau)h(\tau - t)$$

which is a function of two variables:

- The *fixed time* t that we are interested in.
- The *running time* denoted by τ.

The window function $h(t)$ is chosen in such a way that the signal $x(t)$ is left essentially unchanged around the time t, but the signal is suppressed for times distant from the time of interest (Cohen, 1995). The short-time Fourier transform of the signal $x(t)$ is thus defined as

$$\begin{aligned} x_t(\omega) &= \int_{-\infty}^{\infty} x_t(\tau)\exp(-\mathrm{j}\omega\tau)\mathrm{d}\tau \\ &= \int_{-\infty}^{\infty} x(\tau)h(t - \tau)\exp(-\mathrm{j}\omega\tau)\mathrm{d}\tau \end{aligned}$$

which emphasizes the distribution of frequency around time t.

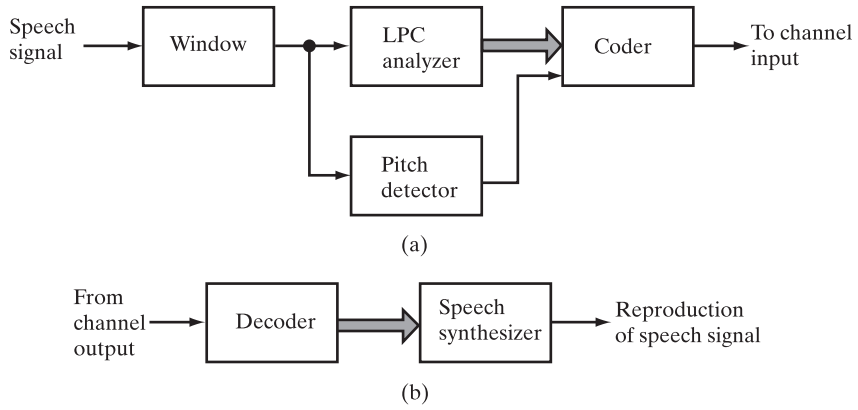


FIGURE 3.15 Block diagram of LPC vocoder: (a) transmitter; (b) receiver.

Figure 3.15 shows the block diagram of an LPC system for the digital transmission and reception of speech signals over a communication channel. The transmitter first applies a *window* (typically, 10 to 30 ms long) to the input speech signal, thereby identifying a block of speech samples for processing. This window is short enough for the vocal-tract shape to be viewed as quasi-stationary, so that the parameters of the speech-production model of Fig. 3.14 may be treated as essentially constant for the duration of the window. The transmitter then analyzes the input speech signal, block by block, by performing two operations: linear prediction and pitch detection. Finally, the following parameters are encoded for transmission over the communication channel:

- The set of coefficients computed by the LPC analyzer.
- The pitch period.
- The gain parameter.
- The voiced–unvoiced parameter.

The receiver performs the inverse operations on the channel output by first decoding the incoming parameters and then using those parameters to synthesize a speech signal that sounds like the original speech signal by utilizing the model of Fig. 3.14.

Itakura–Saito Distance Measure

In the use of an all-pole model for speech, there are two separate issues that need to be considered:

1. The appropriateness of the all-pole filter to model the spectral envelope of a speech signal.
2. The method that is used to estimate the coefficients of the all-pole filter.

If the "true" spectral envelope has only poles, then the all-pole model is the right model. If, however, the spectral envelope has both poles and zeros, then the all-pole model is *not* the right model. Nevertheless, the all-pole model does an adequate job for many applications.

Suppose now that the all-pole model is a good model for whatever envelope we are trying to estimate. If the sound is unvoiced, then the usual (linear prediction) method of computing the all-pole filter's coefficients leads to a good estimate. If, however, the sound is voiced, then the usual method of computing the all-pole filter's coefficients produces a "biased estimate," with the estimate worsening as the pitch frequency increases. El-Jaroudi and Makhoul (1991) overcame this problem by using the Itakura–Saito distance measure, which is discussed below. In particular, if the spectral envelope of the original speech signal is all-pole and the sound is voiced, El-Jaroudi and Makhoul showed that it is possible to recover the true envelope, which cannot be accomplished by the usual method based on linear prediction.

The example of Fig. 3.16 illustrates the limitations of standard all-pole modeling of periodic waveforms. The specifications of this example are as follows:

1. All-pole filter order, $M = 12$.
2. Input signal: periodic pulse sequence with $N = 32$ points in each period.

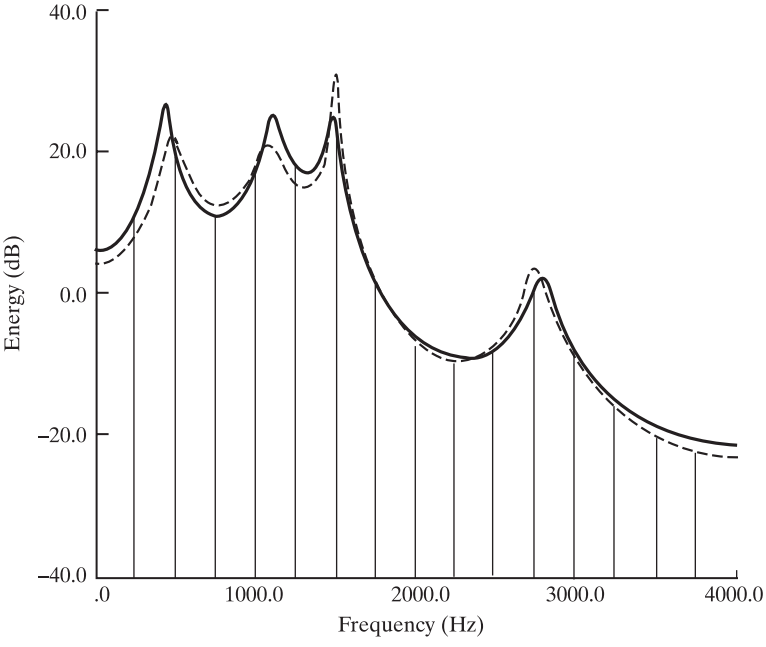


FIGURE 3.16 Example of the limitations of linear prediction spectral analysis. The solid line is the original 12-pole envelope, which goes through all the points. The dashed line is the 12-pole linear predictive model for $N = 30$ spectral lines. (Reproduced from El-Jaroudi & Makhoul, 1991; with permission of the IEEE.)

The solid line in the figure is the original 12-pole spectral envelope that passes through all the frequency points. The dashed envelope is the corresponding result obtained through linear predictive modeling performed on the periodic pulse train. This figure clearly illustrates the failure of the linear prediction method used to compute the all-pole model's coefficients.

The source of the problem cited here is traceable to the choice of the minimum mean-square error as the criterion for the linear predictive modeling of periodic waveforms. To overcome the problem, we need an error criterion that provides the basis for recovering the correct envelope from discrete spectral samples. A criterion that achieves this objective is the discrete version of the *Itakura–Saito distance measure*.[9]

To proceed with a derivation of this new error criterion, consider a real-valued, stationary stochastic process denoted by $u(n)$ that is periodic with period N. Expanding $u(n)$ into a Fourier series, we may write

$$u(n) = \frac{1}{N}\sum_{k=0}^{N-1} U_k \exp(\mathrm{j}n\omega_k), \qquad n = 0, 1, \ldots, N-1, \tag{3.148}$$

where $\omega_k = 2\pi k/N$ and

$$U_k = \sum_{n=0}^{N-1} u_n \exp(-\mathrm{j}n\omega_k), \qquad k = 0, 1, \ldots, N-1. \tag{3.149}$$

That is, $u(n)$ and U_k form a discrete Fourier transform (DFT) pair. Likewise, the autocorrelation function of $u(n)$ for lag m is expanded into the Fourier series

$$r(m) = \frac{1}{N}\sum_{k=0}^{N-1} S_k \exp(\mathrm{j}m\omega_k), \qquad m = 0, 1, \ldots, N-1, \tag{3.150}$$

where

$$S_k = \sum_{m=0}^{N-1} r(m)\exp(-\mathrm{j}m\omega_k), \qquad k = 0, 1, \ldots, N-1 \tag{3.151}$$

[9]Itakura and Saito (1970) were the first to show that the maximum-likelihood procedure provides a fundamental theoretic basis for the estimation of the spectral envelope of a stochastic process using linear predictive coding. In doing so, they introduced the Itakura–Saito distance measure as the error criterion for a spectral matching evaluation of speech signals viewed as sample functions of an autoregressive process. This new error criterion was originally formulated for continuous spectra, which is an appropriate model for unvoiced sounds. Subsequently, McAulay (1984) generalized the Itakura–Saito distance measure to periodic processes having arbitrary model spectra. Then, El-Jaroudi and Makhoul (1991) used the discrete version of the Itakura–Saito distance measure to develop a "discrete" all-pole model that overcomes the limitations of linear prediction in the modeling of voiced sounds.

In Section 3.11, we have focused on the discrete version of the Itakura–Saito distance measure. The continuous version of this error criterion follows from the negative of the log-likelihood function $L(\mathbf{a})$ in Eq. (3.156), presented on page 205. Specifically, if the number of frequency points, N, is permitted to approach infinity, then the summation in Eq. (3.156) is replaced by an integral. The negative of the resulting formula is indeed the continuous version of the Itakura–Saito distance measure.

is a spectral sample. That is, $r(m)$ and S_k also form a discrete Fourier transform pair of their own. The complex-valued random variables $U_0, U_1, \ldots, U_{N-1}$ defined in Eq. (3.149) are uncorrelated with each other, as shown by (see Problem 18, Chapter 1)

$$\mathbb{E}[U_k U_j^*] = \begin{cases} S_k & \text{for } j = k \\ 0 & \text{otherwise} \end{cases}. \tag{3.152}$$

With a parametric model of the stochastic process $u(n)$ in mind, we note that, in general, the spectrum of $u(n)$ depends functionally on a set of *spectral parameters* denoted by the vector

$$\mathbf{a} = [a_1, a_2, \ldots, a_M]^{\mathrm{T}},$$

where M is the order of the model. The problem of interest is to estimate the spectral parameter vector $\mathbf{a}$ by using the statistical information contained in the observed random variables $u(0), u(1), \ldots, u(N-1)$, or, equivalently, the complex random variables $U_0, U_1, \ldots, U_{N-1}$. For this estimation, we propose to use the maximum-likelihood procedure, which is known to have several optimality properties of an asymptotic kind (Van Trees, 1968; McDonough & Whalen, 1995; Quatieri, 2001). (A brief description of maximum-likelihood estimation is presented in Appendix D.)

To formulate this estimation procedure, we naturally need a probabilistic model for the stochastic process $u(n)$, and to make the formulation mathematically tractable, we assume a Gaussian distribution with zero mean. Thus, in light of the first line of Eq. (3.152), we express the "complex" probability density function of the random variable U_k, given the parameter vector $\mathbf{a}$, as (see Problem 18, Chapter 1)

$$f_U(U_k|\mathbf{a}) = \frac{1}{\pi S_k(\mathbf{a})} \exp\left(-\frac{|U_k|^2}{S_k(\mathbf{a})}\right), \qquad k = 0, 1, \ldots, N-1, \tag{3.153}$$

where we have used $S_k(\mathbf{a})$ to denote the kth spectral sample of the process to emphasize its functional dependence on the parameter vector $\mathbf{a}$. Since these random variables are uncorrelated by virtue of the second line of Eq. (3.152), the Gaussian assumption implies that they are also statistically independent. Hence, the joint probability density function of the random variables $U_0, U_1, \ldots, U_{N-1}$, given the parameter vector $\mathbf{a}$, is

$$\begin{aligned} f_U(U_0, U_1 \ \ldots, U_N|\mathbf{a}) &= \prod_{k=0}^{N-1} f_U(U_k|\,\mathbf{a}) \\ &= \prod_{k=0}^{N-1} \frac{1}{\pi S_k(\mathbf{a})} \exp\left(-\frac{|U_k|^2}{S_k(\mathbf{a})}\right). \end{aligned} \tag{3.154}$$

By definition, the *likelihood function*, denoted by $l(\mathbf{a})$ and viewed as a function of the parameter vector $\mathbf{a}$, is equal to the joint probability density function of Eq. (3.154). Hence, we write

$$l(\mathbf{a}) = \prod_{k=0}^{N-1} \frac{1}{\pi S_k(\mathbf{a})} \exp\left(-\frac{|U_k|^2}{S_k(\mathbf{a})}\right). \tag{3.155}$$

To simplify matters, we take logarithms and so express the *log-likelihood function* as

$$\begin{aligned} L(\mathbf{a}) &= \ln l(\mathbf{a}) \\ &= -\sum_{k=0}^{N-1}\left(\frac{|U_k|^2}{S_k(\mathbf{a})} + \ln S_k(\mathbf{a})\right), \end{aligned} \tag{3.156}$$

where we have ignored the constant $-N \ln \pi$, as it has no bearing on the problem.

For the *unconstrained problem* (i.e., in the absence of a parametric model), there are N unknown parameters S_k, $k = 0, 1, \ldots, N-1$. From Eq. (3.156), we readily see that the maximum-likelihood estimate of S_k is $|U_k|^2$. Hence, the use of $S_k = |U_k|^2$ in Eq. (3.156) yields the maximum value of the log-likelihood function:

$$L_{\max} = -\sum_{k=0}^{N-1}(1 + \ln|U_k|^2). \tag{3.157}$$

Accordingly, we define the "differential" log-likelihood function for the parameter vector $\mathbf{a}$ as

$$\begin{aligned} D_{\text{IS}}(\mathbf{a}) &= L_{\max} - L(\mathbf{a}) \\ &= \sum_{k=0}^{N-1}\left(\frac{|U_k|^2}{S_k(\mathbf{a})} - \ln\left(\frac{|U_k|^2}{S_k(\mathbf{a})}\right) - 1\right). \end{aligned} \tag{3.158}$$

Equation (3.158) is the discrete version of the Itakura–Saito distance measure (McAulay, 1984); the subscripts in $D_{IS}(\mathbf{a})$ refer to the originators of the continuous version of the measure. Note that $D_{IS}(\mathbf{a})$ is always nonnegative and is equal to zero only when $S_k(a) = |U_k|^2$ for all k.

Discrete All-Pole Model

Using the error criterion of Eq. (3.158), El-Jaroudi and Makhoul (1991) derived a parametric model of a spectral envelope, given a set of discrete spectral points. The new model is called a *discrete all-pole model*. The derivation of the model is based on a *matching condition* that equates the autocorrelation function $r(i)$ corresponding to the given discrete spectrum to the autocorrelation function $\hat{r}(i)$ corresponding to an all-pole model that is sampled at the same discrete frequencies of the given spectrum—hence the name of the model. This matching condition, in turn, results in a set of *nonlinear* equations relating the model parameters to the autocorrelation function of the given discrete spectrum. To simplify the solution of the nonlinear problem, the following property of sampled all-pole filters is used:

$$\sum_{k=0}^{M} a_k \hat{r}(i-k) = \hat{h}(-i) \quad \text{for all } i. \tag{3.159}$$

In Eq. (3.159), $a_0 = 1$ and

$$\hat{h}(-i) = \frac{1}{N}\sum_{m=1}^{N}\left(\frac{\exp(-\mathrm{j}\omega_m i)}{\sum_{k=0}^{M} a_k \exp(-\mathrm{j}\omega_m k)}\right). \tag{3.160}$$

The term $\hat{h}(-i)$ is the *time-reversed impulse response* of the discrete frequency-sampled all-pole filter. By substituting the all-pole property of Eq. (3.159) into the minimizing condition

$$\frac{\partial D_{\text{IS}}(\mathbf{a})}{\partial a_k} = 0 \quad \text{for} \quad k = 1, 2, \ldots, M,$$

we obtain the following set of equations relating the all-pole predictor coefficients to the given autocorrelation sequence:

$$\sum_{k=0}^{M} a_k r(i-k) = \hat{h}(-i), \quad 0 \le i \le M. \tag{3.161}$$

[For the derivation of Eqs. (3.159) through (3.161), see Problem 30.]

To compute the all-pole model parameters a_k, $k = 1, 2, \ldots, M$, a two-step iterative algorithm is used (El-Jaroudi and Makhoul, 1991):

- Given an estimate of the model, evaluate $\hat{h}(-i)$ using Eq. (3.160).
- Given the new estimate $\hat{h}(-i)$, solve the linearized equations (3.161) for a new estimate of the model parameters.

According to El-Jaroudi and Makhoul, the algorithm converges to a unique global minimum. It is also of interest to note that this algorithm results in the correct all-pole envelope for the problem depicted in Fig. 3.16. In general, the El-Jaroudi–Makhoul algorithm is less biased than the corresponding solution based on linear prodiction.

3.12 SUMMARY AND DISCUSSION

Linear prediction is basic to the model-building problem, given a set of physical data. In this chapter, we presented a detailed study of the linear prediction problem pertaining to wide-sense stationary stochastic processes. In particular, we used Wiener filter theory to develop optimum solutions for the two basic forms of linear prediction:

- Forward linear prediction, in which we are given the input sequence $u(n-1)$, $u(n-2), \ldots, u(n-M)$ and the requirement is to make an optimum prediction of the current sample $u(n)$ at time n.
- Backward linear prediction, in which we are given the input sequence $u(n)$, $u(n-1), \ldots, u(n-M+1)$ and the requirement is to make an optimum prediction of the old sample $u(n-M)$ at time $n-m$.

In both cases, the desired response is derived from the time series itself. In forward linear prediction, $u(n)$ acts as the desired response, whereas in backward linear prediction, $u(n-M)$ acts as the desired response.

The prediction process may be described in terms of a predictor or, equivalently, a prediction-error filter. These two linear systems differ from each other in their respective outputs. The output of a forward predictor is a one-step prediction of the input; the output of a forward prediction-error filter is the prediction error. In a similar way, we may distinguish between a backward predictor and a backward prediction-error filter.

The two structures most commonly used for building a prediction-error filter are as follows:

- An FIR filter, about which the issue of concern is the determination of the tap weights.
- A lattice filter, about which the issue of concern is the determination of the reflection coefficients.

These two sets of parameters are in fact uniquely related to each other via the Levinson–Durbin recursion.

The important properties of prediction-error filters may be summarized as follows:

- The forward prediction-error filter is minimum phase, which means that all the zeros of its transfer function lie inside the unit circle in the z-plane. The corresponding inverse filter, representing an autoregressive model of the input process, is therefore stable.
- The backward prediction-error filter is maximum phase, which means that all the zeros of its transfer function lie outside the unit circle in the z-plane. In this case, the inverse filter is unstable and therefore of no practical value.
- The forward prediction-error filter is a whitening filter, whereas the backward prediction-error filter is an anticausal whitening filter. (See Problem 14.)

The lattice predictor offers some highly desirable properties:

- *An order-recursive structure*, which means that the prediction order may be increased by adding one or more stages to the structure without destroying the previous calculations.
- *Modularity*, which is exemplified by the fact that all the stages of the lattice predictor have exactly the same physical structure.
- *Simultaneous computation of forward and backward prediction errors*, which provides for computational efficiency.
- *Statistical decoupling of the individual stages*, which is another way of saying that the backward prediction errors of varying orders produced by the different stages of the lattice predictor are uncorrelated with each other. This property, embodied in the Cholesky factorization, is exploited in the joint-estimation process, wherein the backward prediction errors are used to provide an estimate of some desired response.

PROBLEMS

1. The augmented Wiener–Hopf equations (3.14) of a forward prediction-error filter were derived by first optimizing the linear prediction filter in the mean-square-error sense and then combining the two resultants: the Wiener–Hopf equations for the tap-weight vector and the minimum mean-square prediction error. This problem addresses the issue of deriving those equations directly by proceeding as follows:

(a) Formulate the expression for the mean-square value of the forward prediction error as a function of the tap-weight vector of the forward prediction-error filter.

(b) Minimize this mean-square prediction error, subject to the constraint that the leading element of the tap-weight vector of the forward prediction-error filter is equal to unity.

[*Hint*: Use the method of Lagrange multipliers to solve the constrained optimization problem. For details of this method, see Appendix C. This hint also applies to part (b) of Problem 2.]

2. The augmented Wiener–Hopf equations (3.38) of a backward prediction-error filter were derived indirectly in Section 3.2. This problem addresses the issue of deriving those equations directly by proceeding as follows:

(a) Formulate the expression for the mean-square value of the backward prediction error in terms of the tap-weight vector of the backward prediction-error filter.

(b) Minimize this mean-square prediction error, subject to the constraint that the last element of the tap-weight vector of the backward prediction-error filter is equal to unity. [*Hint*: See the hint for Problem 1.]

3. (a) Equation (3.24) defines the Wiener–Hopf equations for backward linear prediction. This system of equations is reproduced here for convenience as

$$\mathbf{R}\mathbf{w}_b = \mathbf{r}^{\mathrm{B}*},$$

where $\mathbf{w}_b$ is the tap-weight vector of the predictor, $\mathbf{R}$ is the correlation matrix of the tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$, and $\mathbf{r}^{\mathrm{B}*}$ is the cross-correlation vector between these tap inputs and the desired response $u(n-M)$. Show that if the elements of the column vector $\mathbf{r}^{\mathrm{B}*}$ are rearranged in reverse order, the effect of this reversal is to modify the Wiener–Hopf equations as

$$\mathbf{R}^{\mathrm{T}}\mathbf{w}_b^{\mathrm{B}} = \mathbf{r}^{*}.$$

(b) Show that the inner products $\mathbf{r}^{\mathrm{BT}}\mathbf{w}_b$ and $\mathbf{r}^{\mathrm{T}}\mathbf{w}_b^{\mathrm{B}}$ are equal.

4. Consider a wide-sense stationary process $u(n)$ whose autocorrelation function has the following values for different lags:

$$\begin{aligned} r(0) &= 1; \\ r(1) &= 0.8; \\ r(2) &= 0.6; \\ r(3) &= 0.4. \end{aligned}$$

(a) Use the Levinson–Durbin recursion to evaluate the reflection coefficients κ_1, κ_2, and κ_3.

(b) Set up a three-stage lattice predictor for this process, using the values for the reflection coefficients found in part (a).

(c) Evaluate the average power of the prediction error produced at the output of each of the three stages in this lattice predictor. Hence, make a plot of prediction-error power versus prediction order. Comment on your results.

5. Consider the filtering structure described in Fig. P3.1, where the delay Δ is an integer greater than unity. The requirement is to choose the weight vector $\mathbf{w}$ so as to minimize the mean-square value of the estimation error $e(n)$. Find the optimum value of $\mathbf{w}(n)$.

6. Consider the linear prediction of a stationary autoregressive process $u(n)$ generated from the first-order difference equation

$$u(n) = 0.9u(n-1) + \nu(n),$$

where $\nu(n)$ is white noise of zero mean and unit variance.

(a) Determine the tap weights $a_{2,1}$ and $a_{2,2}$ of the forward prediction-error filter.

(b) Determine the reflection coefficients κ_1 and κ_2 of the corresponding lattice predictor.

Comment on your results in parts (a) and (b).

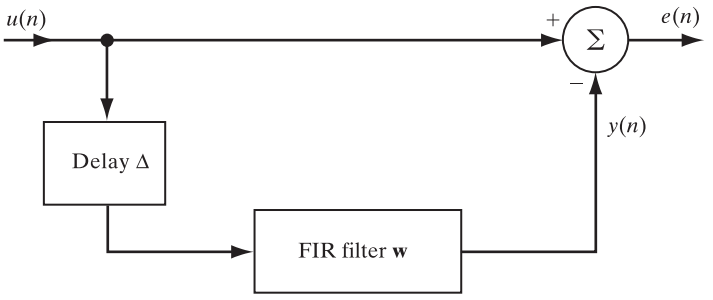


FIGURE P3.1

7. **(a)** A process $u_1(n)$ consists of a single sinusoidal process of complex amplitude α and angular frequency ω in additive white noise of zero mean and variance σ_ν^2, as given by

$$u_1(n) = \alpha e^{j\omega n} + \nu(n),$$

where

$$\mathbb{E}[\,|\alpha|^2\,] = \sigma_\alpha^2$$

and

$$\mathbb{E}[\,|\nu(n)|^2\,] = \sigma_\nu^2.$$

The process $u_1(n)$ is applied to a linear predictor of order M, optimized in the mean-square-error sense. Do the following:

(i) Determine the tap weights of the prediction-error filter of order M and the final value of the prediction-error power P_M.

(ii) Determine the reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$ of the corresponding lattice predictor.

(iii) How are the results in parts (i) and (ii) modified when we let the noise variance σ_ν^2 approach zero?

(b) Consider next an AR process described by

$$u_2(n) = -\alpha e^{j\omega} u_2(n-1) + \nu(n),$$

where, as before, $\nu(n)$ is additive white noise of zero mean and variance σ_ν^2. Assume that $0 < |\alpha| < 1$, but very close to unity. The process $u_2(n)$ is also applied to a linear predictor of order M, optimized in the mean-square-error sense.

(i) Determine the tap weights of the new prediction-error filter of order M.

(ii) Determine the reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$ of the corresponding lattice predictor.

(c) Use your results in parts (a) and (b) to compare the similarities and differences between the linear prediction of $u_1(n)$ and that of $u_2(n)$.

8. Equation (3.40) defines the Levinson–Durbin recursion for forward linear prediction. By rearranging the elements of the tap-weight vector $\mathbf{a}_m$ backward and then taking their complex conjugate, reformulate the Levinson–Durbin recursion for backward linear prediction as in Eq. (3.42).

9. Starting with the definition of Eq. (3.47) for Δ_{m-1}, show that Δ_{m-1} equals the cross-correlation between the delayed backward prediction error $b_{m-1}(n-1)$ and the forward prediction error $f_{m-1}(n)$.

10. Develop in detail the relationship between the Schur–Cohn method and the inverse recursion as outlined by Eqs. (3.97) through (3.100).

11. Consider an autoregressive process $u(n)$ of order two described by the difference equation

$$u(n) = u(n-1) - 0.5u(n-2) + v(n),$$

where $v(n)$ is a white-noise process of zero mean and variance 0.5.

(a) Find the average power of $u(n)$.

(b) Find the reflection coefficients κ_1 and κ_2.

(c) Find the average prediction-error powers P_1 and P_2.

12. Using the one-to-one correspondence between the two sequences of numbers $\{P_0, \kappa_1, \kappa_2\}$ and $\{r(0), r(1), r(2)\}$, compute the autocorrelation function values $r(1)$ and $r(2)$ that correspond to the reflection coefficients κ_1, and κ_2 found in Problem 11 for the second-order autoregressive process $u(n)$.

13. In Section 3.4, we presented a derivation of the minimum-phase property of a prediction-error filter by using Rouché's theorem. In this problem, we explore another derivation of this property—one based on proof by contradiction. Consider Fig. P3.2, which shows a prediction-error filter (of order M) represented as the cascade of two functional blocks, one with transfer function $C_i(z)$ and the other with its transfer function equal to the zero factor $(1 - z_i z^{-1})$. Let $S(\omega)$ denote the power spectral density of the process $u(n)$ applied to the input of the prediction-error filter.

(a) Show that the mean-square value of the forward prediction error $f_M(n)$ equals

$$\varepsilon = \int_{-\pi}^{\pi} S(\omega) \, |C_i(e^{j\omega})|^2 [1 - 2\rho_i \cos(\omega - \omega_i) + \rho_i^2] \, d\omega,$$

where $z_i = \rho_i e^{j\omega_i}$. Hence, evaluate the derivative $\partial\varepsilon/\partial\rho_i$.

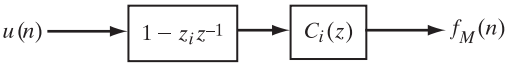


FIGURE P3.2

(b) Suppose that $\rho_i > 1$, so that the complex zero lies outside the unit circle. Show that, under this condition, $\partial\varepsilon/\partial\rho_i > 0$. Is such a condition possible when the filter operates optimally? What conclusion can you draw from your answers?

14. When an autoregressive process of order M is applied to a forward prediction-error filter of the same order, the output consists of white noise. Show that when such a process is applied to a backward prediction-error filter of order M, the output consists of an anticausal realization of white noise.

15. Consider a forward prediction-error filter characterized by a real-valued set of coefficients $a_{m,1}, a_{m,2}, \ldots, a_{m,m}$. Define a polynomial $\phi_m(z)$ as

$$\sqrt{P_m}\phi_m(z) = z^m + a_{m,1} z^{m-1} + \cdots + a_{m,m},$$

where P_m is the average prediction-error power of order m and z^{-1} is the unit-delay operator. [Note the difference between the definition of $\phi_m(z)$ and that of the corresponding transfer function $H_{f,m}(z)$ of the filter.] The filter coefficients bear a one-to-one correspondence with the sequence of autocorrelations $r(0), r(1), \ldots, r(m)$. Now define

$$S(z) = \sum_{i=-m}^{m} r(i) z^{-i},$$

and show that

$$\frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}} \phi_m(z)\phi_k(z^{-1})S(z)\,\mathrm{d}z = \delta_{mk},$$

where

$$\delta_{mk} = \begin{cases} 1, & k = m \\ 0, & k \neq m \end{cases}$$

is the Kronecker delta and the contour $\mathscr{C}$ is the unit circle. The polynomial ϕ_m is referred to as a *Szegö polynomial*. (See Appendix A for a review of contour integration.)

16. In a lattice-based structure for joint-process estimation, explain why the backward error prediction method is preferable to forward prediction.

17. For $m = 2$, compute $r(2)$ for the autocorrelation function and the reflection coefficients given P_0, κ_1, κ_2, and κ_3.

18. Find the inverse recursion using Levinson–Durbin recursion for the tap weights $a_{3,1}, a_{3,2}, a_{3,3}$, and $a_{4,4}$ of a prediction-error filter of order 3. Determine the corresponding reflection coefficient $\kappa_1, \kappa_2, \kappa_3$, and κ_4 for the order 4.

19. **(a)** Consider the matrix product **LR** that appears in the decomposition of Eq. (3.114), where the $(M + 1)$-by-$(M + 1)$ lower triangular matrix **L** is defined in Eq. (3.106) and **R** is the $(M + 1)$-by-$(M + 1)$ correlation matrix. Let **Y** denote this matrix product, and let y_{mk} denote the mkth element of **Y**. Show that

$$y_{mm} = P_m, \qquad m = 0, 1, \ldots, M,$$

where P_m is the prediction-error power for order m.

(b) Show that the mth column of matrix **Y** is obtained by passing the autocorrelation sequence $\{r(0), r(1), \ldots, r(m)\}$ through a corresponding sequence of backward prediction-error filters represented by the transfer functions $H_{b,o}(z), H_{b,1}(z), \ldots, H_{b,m}(z)$.

(c) Suppose that we apply the autocorrelation sequence $\{r(0), r(1), \ldots, r(m)\}$ to the input of a lattice predictor of order m. Show that the variables appearing at the various points on the lower line of the predictor at time m equal the elements of the mth column of matrix **Y**.

(d) For the situation described in part (c), show that the lower output of stage m in the predictor at time m equals P_m and that the upper output of this same stage at time $m + 1$ equals Δ_m^*. How is the ratio of these two outputs related to the reflection coefficient of stage $m + 1$?

(e) Use the results of part (d) to develop a recursive procedure for computing the sequence of reflection coefficients from the autocorrelation sequence.

20. Prove the following correlation properties of lattice filters:

(a) $\mathbb{E}[f_m(n)u^*(n-k)] = 0, \quad 1 \le k \le m$
$\mathbb{E}[b_m(n)u^*(n-k)] = 0, \quad 0 \le k \le m-1$

(b) $\mathbb{E}[f_m(n)u^*(n)] = \mathbb{E}[b_m(n)u^*(n-m)] = P_m$

(c) $\mathbb{E}[b_m(n)b_i^*(n)] = \begin{cases} P_m, & m = i \\ 0, & m \neq i \end{cases}$

(d) $\mathbb{E}[f_m(n)f_i^*(n-l)] = \mathbb{E}[f_m(n+l)f_i^*(n)] = 0, \quad 1 \le l \le m-i, \; m > i$

$\mathbb{E}[b_m(n)b_i^*(n-l)] = \mathbb{E}[b_m(n+l)b_i^*(n)] = 0, \quad 0 \le l \le m-i-1, \; m > i$

(e) $\mathbb{E}[f_m(n+m)f_i^*(n+i)] = \begin{cases} P_m, & m = i \\ 0, & m \neq i \end{cases}$

$\mathbb{E}[b_m(n+m)b_i^*(n+i)] = P_m, \quad m \ge i$

(f) $\mathbb{E}[f_m(n)b_i^*(n)] = \begin{cases} \kappa_i^* P_m, & m \ge i \\ 0, & m < i \end{cases}$

21. In Levinson–Durbin recursion, given the set of reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$, compute the corresponding set of tap weights $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$ for a prediction-error filter of final order M, where the remaining coefficient of the filter are 1. Given the set of tap weights $a_{M,1}$, $a_{M,2}, \ldots, a_{M,M}$ solve for the corresponding set of reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$. [*Hint*: We may solve this problem by applying the inverse form of the Levinson–Durbin recursion.]

22. Consider the problem of optimizing stage m of the lattice predictor. The cost function to be used in the optimization is described by

$$J_m(\kappa_m) = a\mathbb{E}[|f_m(n)|^2] + (1-a)\mathbb{E}[|b_m(n)|^2],$$

where a is a constant that lies between zero and unity and $f_m(n)$ and $b_m(n)$ denote the forward and backward prediction errors at the output of stage m, respectively.

(a) Show that the optimum value of the reflection coefficient κ_m for which J_m is at minimum is

$$\kappa_{m,o}(a) = -\frac{\mathbb{E}[b_{m-1}(n-1)f_{m-1}^*(n)]}{(1-a)\mathbb{E}[|f_{m-1}(n)|^2] + a\mathbb{E}[b_{m-1}(n-1)|^2]}.$$

(b) Evaluate $\kappa_{m,o}(a)$ for each of the following three special conditions:

(1) $a = 1$;
(2) $a = 2$;
(3) $a = \frac{1}{2}$.

Notes: When the parameter $a = 1$, the cost function reduces to

$$J_m(\kappa_m) = \mathbb{E}[|f_m(n)|^2].$$

We refer to this criterion as the *forward method.*

When the parameter $a = 0$, the cost function reduces to

$$J_m(\kappa_m) = \mathbb{E}[|b_m(n)|^2].$$

We refer to this second criterion as the *backward method*.

When the parameter $a = \frac{1}{2}$, the formula for $\kappa_{m,0}(a)$ reduces to the *Burg formula*:

$$\kappa_{m,o} = -\frac{2\mathbb{E}[b_{m-1}(n-1)f_{m-1}^*(n)]}{\mathbb{E}[|f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2]}, \qquad m = 1, 2, \ldots, M.$$

23. Let $\kappa_{m,o}^{(1)}$ and $\kappa_{m,o}^{(0)}$ denote the optimum values of the reflection coefficient κ_m for stage m of the lattice predictor, using the forward method and backward method, respectively, as determined in Problem 22.

(a) Show that the optimum value of $\kappa_{m,o}$ obtained from the Burg formula is the harmonic mean of the two values $\kappa_{m,o}^{(1)}$ and $\kappa_{m,o}^{(0)}$; that is,

$$\frac{2}{\kappa_{m,o}} = \frac{1}{\kappa_{m,o}^{(1)}} + \frac{1}{\kappa_{m,o}^{(0)}}.$$

(b) Using the result of part (a), show that

$$|\kappa_{m,o}| \leq 1, \quad \text{for all } m.$$

(c) For the case of a lattice predictor using the Burg formula, show that the mean-square values of the forward and backward prediction errors at the output of stage m are related to those at the input; that is, show that

$$\mathbb{E}[|f_m(n)|^2] = (1 - |\kappa_{m,o}|^2)\mathbb{E}[|f_{m-1}(n)|^2]$$

and

$$\mathbb{E}[|b_m(n)|^2] = (1 - |\kappa_{m,o}|^2)\mathbb{E}[|b_{m-1}(n-1)|^2].$$

24. Following the Levinson–Durbin algorithm presented in Section 3.3, we may express the reflection coefficient κ_m of stage m of a lattice predictor in the following equivalent ensemble-average forms:

$$\kappa_m = -\frac{\mathbb{E}[b_{m-1}(n-1)f_{m-1}^*(n)]}{\mathbb{E}[|b_{m-1}(n-1)|^2]} = -\frac{\mathbb{E}[b_{m-1}(n-1)f_{m-1}^*(n)]}{\mathbb{E}[|f_{m-1}(n)|^2]}$$
$$= -\frac{\mathbb{E}[b_{m-1}(n-1)f_{m-1}^*(n)]}{(\mathbb{E}[|f_{m-1}(n)|^2]\mathbb{E}[|b_{m-1}(n-1)|^2])^{1/2}}.$$

As a result, $|\kappa_m| \leq 1$ for all m. However, we find that when the estimation of the reflection coefficient κ_m is based on a finite number of data, only the temporal-average version of the last formula would maintain the property $|\kappa_m| \leq 1$ for all m. Demonstrate the validity of this statement.

25. Determine the (a) correlation matrix, (b) prediction-error power, and (c) prediction-error filter coefficients of a prediction-error filter as given in Eq. (3.14) that yields the following pair of simultaneous equations:

$$\begin{bmatrix} r(0) & r^*(1) \\ r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} a_{1,0} \\ a_{1,1} \end{bmatrix} = \begin{bmatrix} P_1 \\ 0 \end{bmatrix}$$

26. In Section 3.9, we considered the use of an all-pole lattice filter as the generator of an autoregressive process. This filter may also be used to efficiently compute the autocorrelation sequence $r(1), r(2), \ldots, r(m)$, normalized with respect to $r(0)$. The procedure involves initializing the states (i.e., unit-delay elements) of the lattice inverse filter to $1, 0, \ldots, 0$ and then allowing the filter to operate with zero input. In effect, this procedure provides a lattice interpretation of Eq. (3.67) that relates the autocorrelation sequence $\{r(0), r(1), \ldots, r(M)\}$ to

the augmented sequence of reflection coefficients $\{P_0, \kappa_1, \ldots, \kappa_M\}$. Demonstrate the validity of the procedure for the following values of final order M:

(a) $M = 1$. **(b)** $M = 2$. **(c)** $M = 3$.

27. Illustrate the second method for the application of the Levinson–Durbin recursion, the reflection coefficients κ_1, κ_2, and κ_3 and the average power P_0. Use these parameters to determine the corresponding tap weights $a_{3,1}, a_{3,2}, a_{3,3}$ and $a_{4,4}$ and the prediction-error power P_3 for a prediction-error filter of order 4.

28. Figure P3.3 shows the block diagram of a pole-zero lattice filter, which is an extended version of the all-pole, all-pass lattice filter of Fig. 3.10. The rationale for the extension shown in the figure is to realize the M-pole, M-zero transfer function

$$G(z) = G_0 \frac{\prod_{i=1}^{M}(1 - z/z_i)}{\prod_{i=1}^{M}(1 - z/p_i)},$$

where G_0 is a scaling factor. In light of the discussion presented in Section 3.9, the lattice filter rewired as in Fig. P3.3 can realize the prescribed poles of $G(z)$ through a proper choice of the reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$. Do the following:

(a) Given the pole locations $p_1, p_2, \ldots, p_M$ of the transfer function $G(z)$, discuss a procedure for determining the corresponding values of the reflection coefficients.

(b) Given the scaling factor G_0 and the zero locations $z_1, z_2, \ldots, z_M$ of the transfer function $G(z)$, develop a procedure for determining the regression coefficients $h_0, h_1, \ldots, h_M$.

(c) Can the structure of Fig. P3.3 realize a non-minimum-phase transfer function? Justify your answer.

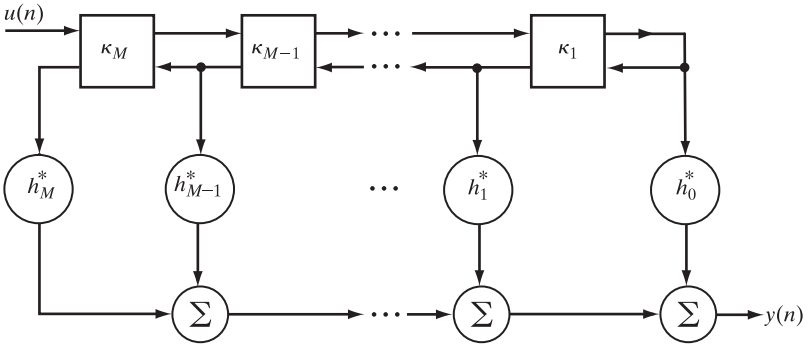


FIGURE P3.3

29. Continuing with Problem 28, determine the reflection coefficients and regression coefficients for the pole–zero lattice filter of Fig. P3.3 so as to realize the following transfer functions:

(a) $G(z) = \dfrac{10(1 + z/0.1)(1 + z/0.6)}{(1 - z/0.4)(1 + z/0.8)}$.

(b) $G(z) = \dfrac{10(1 + z + z^2)}{(1 - z/0.4)(1 + z/0.8)}.$

(c) $G(z) = \dfrac{10(1 + z/0.6)(1 + z/1.5)}{(1 - z/0.4)(1 + z/0.8)}.$

(d) $G(z) = \dfrac{10(1 + z + 0.5z^2)}{(1 - z/0.4)(1 + z/0.8)}.$

30. In this problem, we derive Eqs. (3.159) through (3.161) for computing the discrete all-pole filter, as well as the correlation matching condition.

(a) Let

$$A(e^{j\omega}) = \sum_{k=0}^{M} a_k e^{-j\omega k}$$

and

$$\hat{H}(e^{j\omega}) = \sum_{i=1}^{N} \hat{h}(i) e^{-j\omega i}.$$

Starting with the identity

$$\hat{H}(e^{j\omega_m}) A(e^{j\omega_m}) = 1$$

for $\omega = \omega_m$, derive Eqs. (3.159) and (3.160).

(b) Let $r(i)$ denote the autocorrelation function corresponding to the given discrete spectrum and $\hat{r}(i)$ denote the autocorrelation function corresponding to the all-pole model sampled at the same discrete frequencies as the given spectrum. Also, let

$$\begin{aligned} D(\omega) &= |A(e^{j\omega})|^2 \\ &= \sum_{k=0}^{M} d_k \cos(\omega k), \end{aligned}$$

where

$$d_0 = \sum_{k=0}^{M} a_k^2$$

and

$$d_i = 2 \sum_{k=0}^{M-i} a_k a_{k+1}, \qquad 1 \le i \le M.$$

By setting

$$\frac{\partial D_{\text{IS}}}{\partial d_i} = 0 \quad \text{for} \quad i = 0, 1, \ldots, M,$$

where D_{IS} is the discrete version of the Itakura–Saito distance measure, derive the correlation matching condition

$$\hat{r}(i) = r(i) \quad \text{for } 0 \le i \le M.$$

Although the correlation matching condition provides insight into the application of the Itakura–Saito distance measure, from a computational point of view it is more useful to apply the minimizing condition (El-Jaroudi & Makhoul, 1991)

$$\frac{\partial D_{\text{IS}}}{\partial a_i} = 0 \quad \text{for} \quad i = 0, 1, \ldots, M.$$

Hence, using the all-pole model characterized by

$$S_k(\mathbf{a}) = \frac{1}{\left| \sum_{k=0}^{M} a_k e^{-\text{j}\omega k} \right|^2}$$

in the formula of Eq. (3.158) for $D_{\text{IS}}(\mathbf{a})$, derive Eq. (3.161).

CHAPTER 4

Method of Steepest Descent

In this chapter, we begin the study of gradient-based adaptation by describing an old optimization technique known as the method of steepest descent. This method is basic to the understanding of the various ways in which gradient-based adaptation is implemented in practice.

The method of steepest descent is *recursive* in the sense that its formulation is represented by a feedback system whereby the computation of a filter proceeds iteratively in a step-by-step manner. When the method is applied to the Wiener filter, it provides us with an algorithmic solution that allows for the *tracking* of time variations in the signal's statistics without having to solve the Wiener–Hopf equations each time the statistics change. In the particular case of a stationary environment, we find that, starting from an arbitrary initial value of the tap-weight vector, the solution improves with the increased number of adaptation cycles. The important point to note is that, under the appropriate conditions, the solution so obtained converges to the Wiener solution (i.e., the minimum point of the ensemble-average error surface) without the analyst's having to invert the correlation matrix of the input vector.

4.1 BASIC IDEA OF THE STEEPEST-DESCENT ALGORITHM

Consider a cost function $J(\mathbf{w})$ that is a *continuously differentiable function* of some unknown weight vector $\mathbf{w}$. The function $J(\mathbf{w})$ maps the elements of $\mathbf{w}$ into real numbers. We want to find an optimal solution $\mathbf{w}_o$ that satisfies the condition

$$J(\mathbf{w}_o) \leq J(\mathbf{w}) \qquad \text{for all } \mathbf{w}, \tag{4.1}$$

which is a mathematical statement of *unconstrained optimization.*

A class of unconstrained optimization algorithms that is particularly well suited for adaptive filtering is based on the idea of *local iterative descent*:

> *Starting with an initial guess denoted by* $\mathbf{w}(0)$, *generate a sequence of weight vectors* $\mathbf{w}(1), \mathbf{w}(2), \ldots$, *such that the cost function* $J(\mathbf{w})$ *is reduced at each adaptation cycle of the algorithm; that is,*
>
> $$J(\mathbf{w}(n+1)) < J(\mathbf{w}(n)), \tag{4.2}$$
>
> *where* $\mathbf{w}(n)$ *is the old value of the weight vector and* $\mathbf{w}(n+1)$ *is its updated value.*

We hope that the algorithm will eventually converge onto the optimal value $\mathbf{w}_o$. [We say "hope" because there is a distinct possibility that the algorithm will diverge (i.e., become unstable), unless special precautions are taken.]

In a simple form of iterative descent known as the *method of steepest descent*, the successive adjustments applied to the weight vector $\mathbf{w}$ are in the direction of steepest descent—that is, in a direction opposite to the gradient vector of the cost function $J(\mathbf{w})$, which is denoted by $\nabla J(\mathbf{w})$. For convenience of presentation, we write

$$\begin{aligned}\mathbf{g} &= \nabla J(\mathbf{w}) \\ &= \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}.\end{aligned} \tag{4.3}$$

Accordingly, the steepest-descent algorithm is formally described by

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \frac{1}{2}\mu\mathbf{g}(n), \tag{4.4}$$

where n denotes the adaptation cycle (i.e., the time step in the iterative process), μ is a positive constant called the *step-size parameter*, and the factor $\frac{1}{2}$ is introduced for mathematical convenience. (The reason for introducing this factor will become apparent in the next section.) In going from adaptation cycle n to $n+1$, the algorithm applies the weight adjustment

$$\begin{aligned}\delta\mathbf{w}(n) &= \mathbf{w}(n+1) - \mathbf{w}(n) \\ &= -\frac{1}{2}\mu\mathbf{g}(n).\end{aligned} \tag{4.5}$$

To show that the formulation of the steepest-descent algorithm satisfies the condition of Eq. (4.2), we use a first-order Taylor series expansion around $\mathbf{w}(n)$ to obtain the approximation

$$J(\mathbf{w}(n+1)) \approx J(\mathbf{w}(n)) + \mathbf{g}^{\mathrm{H}}(n)\delta\mathbf{w}(n), \tag{4.6}$$

the use of which is justified for small μ and where the superscript H denotes Hermitian transposition. In Eq. (4.6), it is presumed that we are dealing with a complex-valued vector $\mathbf{w}$, which makes the gradient vector $\mathbf{g}$ complex valued, too—hence the use of the Hermitian transpose. The use of Eq. (4.5) in Eq. (4.6) yields

$$J(\mathbf{w}(n+1)) \approx J(\mathbf{w}(n)) - \frac{1}{2}\mu\|\mathbf{g}(n)\|^2,$$

which shows that $J(\mathbf{w}(n+1))$ is smaller than $J(\mathbf{w}(n))$ provided that the step-size parameter μ is positive. Hence, it follows that with increasing n, the cost function $J(n)$ progressively decreases, approaching the minimum value $J_{\min}$ as n approaches infinity.

4.2 THE STEEPEST-DESCENT ALGORITHM APPLIED TO THE WIENER FILTER

Consider a finite-duration impulse response (FIR) filter with tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$ and a corresponding set of *tap weights* $w_0(n), w_1(n), \ldots, w_{M-1}(n)$. The tap inputs represent samples drawn from a wide-sense stationary stochastic process of zero mean and correlation matrix $\mathbf{R}$. In addition to these inputs, the filter is supplied with a desired response $d(n)$ that provides a frame of reference for the optimum filtering action. Figure 4.1 depicts the filtering action described herein.

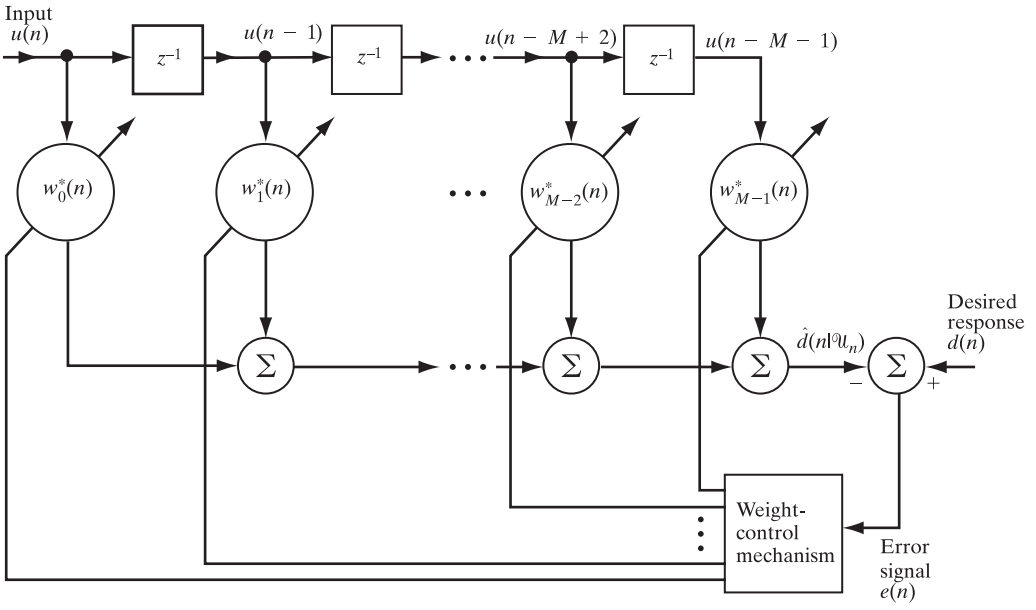


FIGURE 4.1 Structure of an adaptive FIR filter.

The vector of tap inputs at time n is denoted by $\mathbf{u}(n)$, and the corresponding *estimate* of the desired response at the filter output is denoted by $\hat{d}(n|\mathcal{U}_n)$, where $\mathcal{U}_n$ is the space spanned by the tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$. By comparing this estimate with the desired response $d(n)$, we produce an *estimation* error denoted by $e(n)$. We may thus write

$$\begin{aligned} e(n) &= d(n) - \hat{d}(n|\mathcal{U}_n) \\ &= d(n) - \mathbf{w}^{\mathrm{H}}(n)\mathbf{u}(n), \end{aligned} \tag{4.7}$$

where the term $\mathbf{w}^{\mathrm{H}}(n)\mathbf{u}(n)$ is the inner product of the tap-weight vector $\mathbf{w}(n)$ and the tap-input vector $\mathbf{u}(n)$. The expanded form of the tap-weight vector is described by

$$\mathbf{w}(n) = [w_0(n), w_1(n), \ldots, w_{M-1}(n)]^{\mathrm{T}},$$

where the superscript T denotes *transposition*, and that of the tap-input vector is correspondingly described by

$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, (n-M+1)]^{\mathrm{T}}.$$

If the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$ are jointly stationary, then the *mean-square error* or *cost function* $J(\mathbf{w}(n))$, or simply $J(n)$, at time n is a quadratic function of the tap-weight vector, so we may write [see Eq. (2.50)]

$$J(n) = \sigma_d^2 - \mathbf{w}^{\mathrm{H}}(n)\mathbf{p} - \mathbf{p}^{\mathrm{H}}\mathbf{w}(n) + \mathbf{w}^{\mathrm{H}}(n)\mathbf{R}\mathbf{w}(n), \tag{4.8}$$

where σ_d^2 = variance of the desired response $d(n)$,
$\mathbf{p}$ = cross-correlation vector between the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$, and
$\mathbf{R}$ = correlation matrix of the tap-input vector $\mathbf{u}(n)$.

From Chapter 2, we find that the gradient vector is given by

$$\nabla J(n) = \begin{bmatrix} \dfrac{\partial J(n)}{\partial a_0(n)} + \mathrm{j}\dfrac{\partial J(n)}{\partial b_0(n)} \\ \dfrac{\partial J(n)}{\partial a_1(n)} + \mathrm{j}\dfrac{\partial J(n)}{\partial b_1(n)} \\ \vdots \\ \dfrac{\partial J(n)}{\partial a_{M-1}(n)} + \mathrm{j}\dfrac{\partial J(n)}{\partial b_{M-1}(n)} \end{bmatrix} \tag{4.9}$$

$$= -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n),$$

where, in the expanded column vector, $\partial J(n)/\partial a_k(n)$ and $\partial J(n)/\partial b_k(n)$ are the partial derivatives of the cost function $J(n)$ with respect to the real part $a_k(n)$ and the imaginary part $b_k(n)$ of the kth tap weight $w_k(n)$, respectively, with $k = 1, 2, \ldots, M-1$. For the application of the steepest-descent algorithm, we assume that in Eq. (4.9) the correlation matrix $\mathbf{R}$ and the cross-correlation vector $\mathbf{p}$ are known, so that we may compute the gradient vector $\nabla J(n)$ for a given value of the tap-weight vector $\mathbf{w}(n)$. Thus, substituting Eq. (4.9) into Eq. (4.4), we may compute the updated value of the tap-weight vector $\mathbf{w}(n+1)$ by using the simple recursive relation

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu[\mathbf{p} - \mathbf{R}\mathbf{w}(n)], \qquad n = 0, 1, 2, \ldots, \tag{4.10}$$

which describes the mathematical formulation of the steepest-descent algorithm for Wiener filtering.

According to Eq. (4.10), the adjustment $\delta\mathbf{w}(n)$ applied to the tap-weight vector at time $n+1$ is equal to $\mu[\mathbf{p} - \mathbf{R}\mathbf{w}(n)]$. The adjustment may also be expressed as μ times the expectation of the inner product of the tap-input vector $\mathbf{u}(n)$ and the estimation error $e(n)$. (See Problem 7.) This suggests that we may use a bank of cross-correlators to compute the correction $\delta w(n)$ applied to the tap-weight vector $\mathbf{w}(n)$, as indicated in Fig. 4.2. In this figure, the elements of the correction vector $\delta\mathbf{w}(n)$ are denoted by $\delta w_0(n), \delta w_1(n), \ldots, \delta w_{M-1}(n)$.

Another point of interest is that we may view the steepest-descent algorithm of Eq. (4.10) as a *feedback model*, as illustrated by the *signal-flow graph* shown in Fig. 4.3. This model is multidimensional in the sense that the "signals" at the *nodes* of the graph consist of vectors and that the *transmittance* of each branch of the graph is a scalar or a square matrix. For each branch of the graph, the signal vector flowing out equals the signal vector flowing in, multiplied by the transmittance matrix of the branch. For two branches connected in parallel, the overall transmittance matrix equals the sum of the transmittance matrices of the individual branches. For two branches connected in cascade, the overall transmittance matrix equals the product of the individual transmittance matrices arranged in the same order as the pertinent branches. Finally, the symbol z^{-1} is the unit-delay operator, and $z^{-1}\mathbf{I}$ is the transmittance matrix of a unit-delay branch representing a delay of one adaptation cycle.

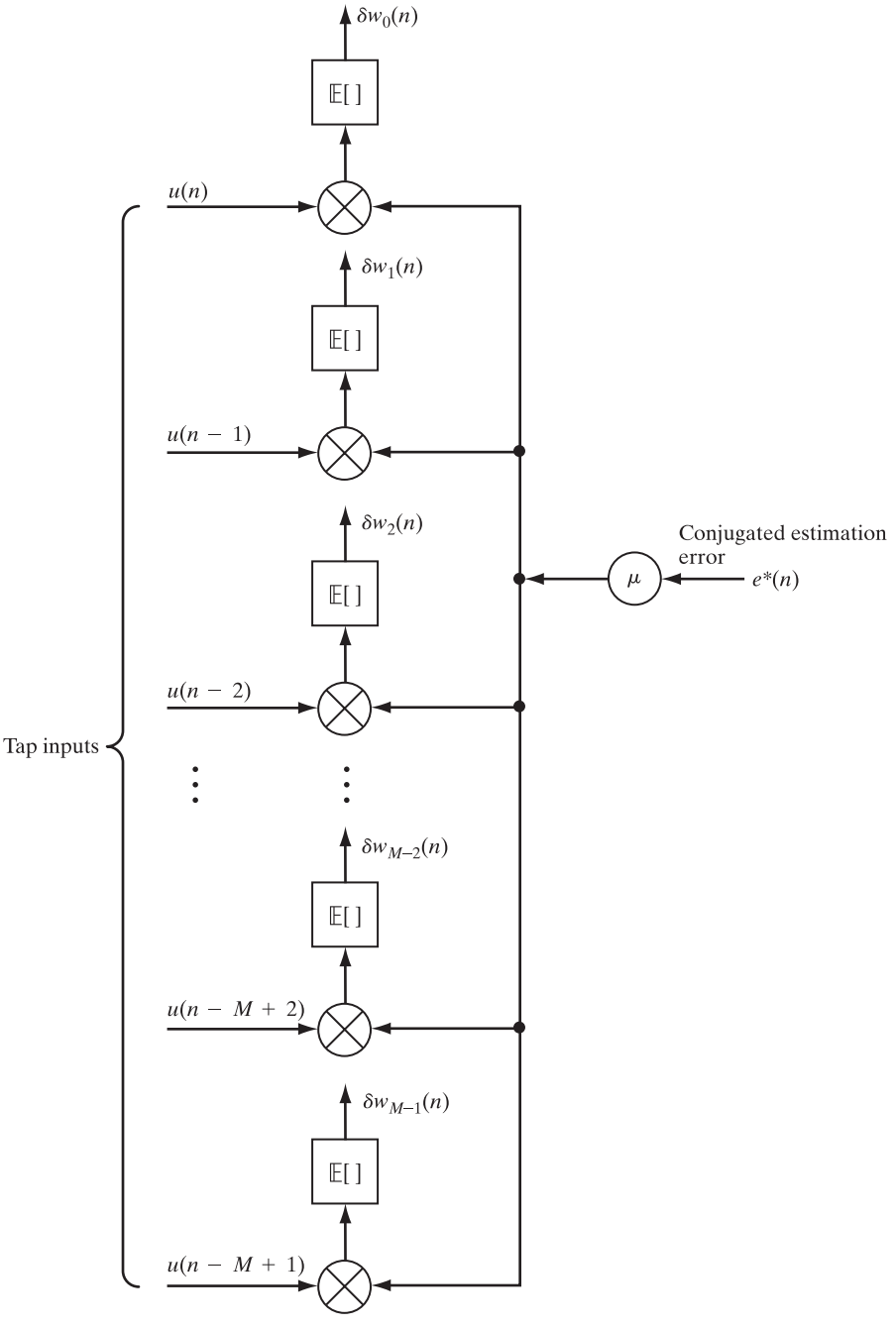


FIGURE 4.2 Bank of cross-correlators for computing the corrections to the elements of the tap-weight vector at time n.

4.3 STABILITY OF THE STEEPEST-DESCENT ALGORITHM

Since the steepest-descent algorithm involves the presence of *feedback*, as exemplified by the model of Fig. 4.3, the algorithm is subject to the possibility of becoming *unstable*. From the feedback model of that figure, we observe that the *stability performance* of the steepest-descent algorithm is determined by two factors: (1) the step-size parameter μ and (2) the correlation matrix $\mathbf{R}$ of the tap-input vector $\mathbf{u}(n)$. These two parameters completely control the transfer function of the *feedback loop*. To determine *the condition for the stability* of the steepest-descent algorithm, we examine the *natural modes* of the algorithm (Widrow, 1970). In particular, we use the representation of the correlation matrix $\mathbf{R}$ in terms of its eigenvalues and eigenvectors to define a transformed version of the tap-weight vector.

We begin the analysis by defining a *weight-error vector* at time n as

$$\mathbf{c}(n) = \mathbf{w}_o - \mathbf{w}(n), \tag{4.11}$$

where $\mathbf{w}_o$ is the optimum value of the tap-weight vector, as defined by the Wiener–Hopf equations (2.34). Then, eliminating the cross-correlation vector $\mathbf{p}$ between Eqs. (2.34) and (4.10) and rewriting the result in terms of the weight-error vector $\mathbf{c}(n)$, we get

$$\mathbf{c}(n+1) = (\mathbf{I} - \mu\mathbf{R})\mathbf{c}(n), \tag{4.12}$$

where $\mathbf{I}$ is the identity matrix. Equation (4.12) is represented by the feedback model shown in Fig. 4.4. This diagram further emphasizes the fact that the stability performance of the steepest-descent algorithm is dependent exclusively on μ and $\mathbf{R}$.

Using eigendecomposition, we may express the correlation matrix $\mathbf{R}$ as (see Appendix E)

$$\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{H}}. \tag{4.13}$$

The matrix $\mathbf{Q}$, called the *unitary matrix* of the transformation, has as its columns an orthogonal set of *eigenvectors* associated with the eigenvalues of the matrix $\mathbf{R}$. The

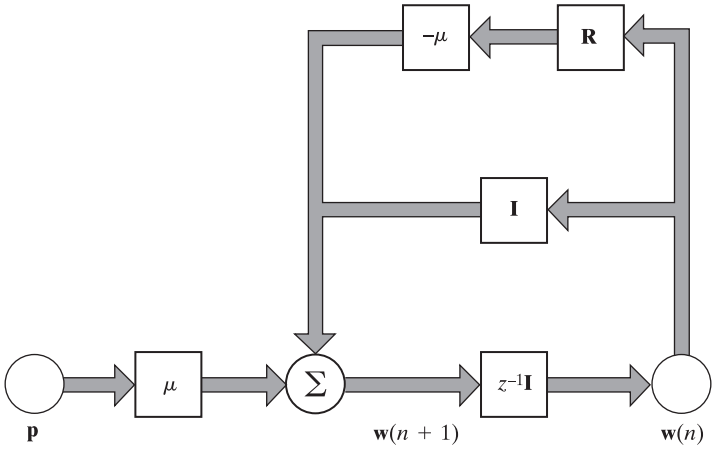


FIGURE 4.3 Signal-flow-graph representation of the steepest-descent algorithm based on Eq. (4.10).

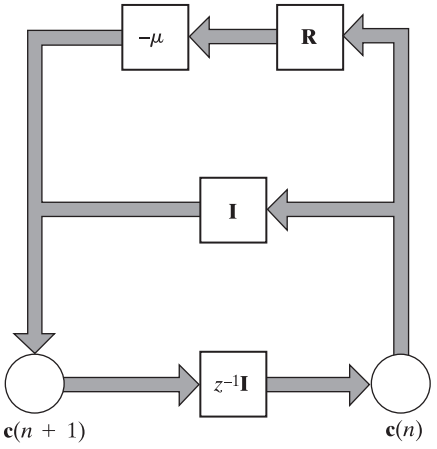


FIGURE 4.4 Signal-flow-graph representation of the steepest-descent algorithm based on the weight-error vector as shown in Eq. (4.12).

matrix $\mathbf{\Lambda}$ is a *diagonal* matrix and has as its diagonal elements the eigenvalues of the correlation matrix $\mathbf{R}$. These eigenvalues, denoted by $\lambda_1, \lambda_2, \ldots, \lambda_M$, are all positive and real. Each eigenvalue is associated with a corresponding eigenvector or column of matrix $\mathbf{Q}$. Substituting Eq. (4.13) into Eq. (4.12), we get

$$\mathbf{c}(n+1) = (\mathbf{I} - \mu\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{H}})\mathbf{c}(n). \tag{4.14}$$

Premultiplying both sides of this equation by $\mathbf{Q}^{\mathrm{H}}$ and using the property of the unitary matrix $\mathbf{Q}$ that $\mathbf{Q}^{\mathrm{H}}$ equals the inverse $\mathbf{Q}^{-1}$, we obtain

$$\mathbf{Q}^{\mathrm{H}}\mathbf{c}(n+1) = (\mathbf{I} - \mu\mathbf{\Lambda})\mathbf{Q}^{\mathrm{H}}\mathbf{c}(n). \tag{4.15}$$

We now define a new set of coordinates, using the definition of Eq. (4.11):

$$\begin{aligned}\mathbf{v}(n) &= \mathbf{Q}^{\mathrm{H}}\mathbf{c}(n)\\ &= \mathbf{Q}^{\mathrm{H}}[\mathbf{w}_o - \mathbf{w}(n)].\end{aligned} \tag{4.16}$$

Accordingly, we may rewrite Eq. (4.14) in the transformed form

$$\mathbf{v}(n+1) = (\mathbf{I} - \mu\mathbf{\Lambda})\mathbf{v}(n). \tag{4.17}$$

The initial value of $\mathbf{v}(n)$ is

$$\mathbf{v}(0) = \mathbf{Q}^{\mathrm{H}}[\mathbf{w}_o - \mathbf{w}(0)]. \tag{4.18}$$

Assuming that the initial tap-weight vector is zero, Eq. (4.18) reduces to

$$\mathbf{v}(0) = \mathbf{Q}^{\mathrm{H}}\mathbf{w}_o. \tag{4.19}$$

For the kth *natural mode* of the steepest-descent algorithm, we thus have

$$v_k(n+1) = (1 - \mu\lambda_k)v_k(n), \qquad k = 1, 2, \ldots, M \tag{4.20}$$

where λ_k is the kth eigenvalue of the correlation matrix $\mathbf{R}$. Equation (4.20) is represented by the scalar-valued feedback model of Fig. 4.5, where z^{-1} is the unit-delay

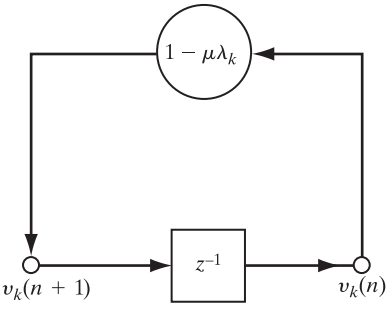


FIGURE 4.5 Signal-flow-graph representation of the kth natural mode of the steepest-descent algorithm based on Eq. (4.20).

operator. Clearly, the structure of this model is much simpler than that of the original matrix-valued feedback model of Fig. 4.3. These two models represent different, and yet equivalent, ways of viewing the steepest-descent algorithm.

Equation (4.20) is a homogeneous *difference equation of the first order*. Assuming that $v_k(n)$ has the initial value $v_k(0)$, we readily obtain the solution

$$v_k(n) = (1 - \mu\lambda_k)^n v_k(0), \qquad k = 1, 2, \ldots, M. \tag{4.21}$$

Since all eigenvalues of the correlation matrix $\mathbf{R}$ are positive and real, the response $v_k(n)$ will exhibit no oscillations. Furthermore, as illustrated in Fig. 4.6, the numbers generated by Eq. (4.21) represent a *geometric series* with a geometric ratio equal to $(1 - \mu\lambda_k)$. For *stability* or *convergence* of the steepest-descent algorithm, the magnitude of this geometric ratio must be less than unity for all k. That is, provided that we have

$$-1 < 1 - \mu\lambda_k < 1 \qquad \text{for all } k,$$

then, as the number of adaptation cycles, n, approaches infinity, all the natural modes of the steepest-descent algorithm die out, irrespective of the initial conditions. This is

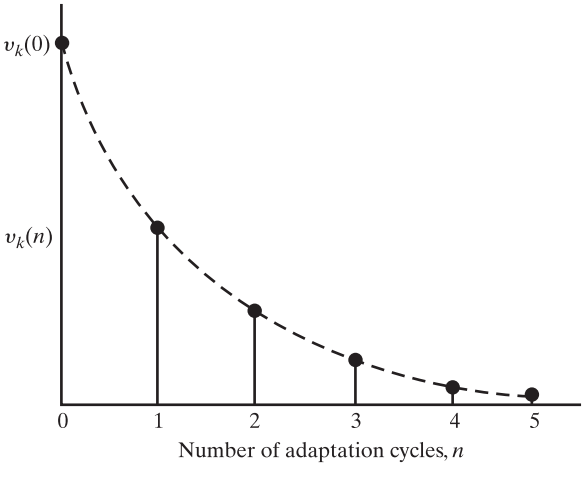


FIGURE 4.6 Variation of the kth natural mode of the steepest-descent algorithm with time, assuming that the magnitude of $1 - \mu\lambda_k$ is less than unity.

equivalent to saying that the tap-weight vector $\mathbf{w}(n)$ approaches the optimum Wiener solution $\mathbf{w}_o$ as n approaches infinity.

Since the eigenvalues of the correlation matrix $\mathbf{R}$ are all real and positive, it follows that a necessary and sufficient condition for the convergence or stability of the steepest-descent algorithm is that the step-size parameter μ satisfy the double inequality

$$0 < \mu < \frac{2}{\lambda_{\max}}, \tag{4.22}$$

where $\lambda_{\max}$ is the largest eigenvalue of the correlation matrix $\mathbf{R}$.

Referring to Fig. 4.6, we see that an exponential envelope of the *time constant* τ_k can be fitted to the geometric series by assuming the unit of time to be the duration of one adaptation cycle and by choosing τ_k such that

$$1 - \mu\lambda_k = \exp\left(-\frac{1}{\tau_k}\right).$$

Hence, the kth time constant can be expressed in terms of the step-size parameter μ and the kth eigenvalue as

$$\tau_k = \frac{-1}{\ln(1 - \mu\lambda_k)}. \tag{4.23}$$

The time τ_k defines the number of adaptation cycles required for the amplitude of the kth natural mode $\nu_k(n)$ to decay to $1/e$ of its initial value $\nu_k(0)$, where e is the base of the natural logarithm. For the special case of slow adaptation, for which the step-size parameter μ is small,

$$\tau_k \approx \frac{1}{\mu\lambda_k}, \qquad \mu \ll 1. \tag{4.24}$$

We may now formulate the transient behavior of the original tap-weight vector $\mathbf{w}(n)$. In particular, premultiplying both sides of Eq. (4.16) by $\mathbf{Q}$, using the fact that $\mathbf{Q}\mathbf{Q}^{\mathrm{H}} = \mathbf{I}$, and solving for $\mathbf{w}(n)$, we get

$$\begin{aligned}
\mathbf{w}(n) &= \mathbf{w}_o - \mathbf{Q}\mathbf{v}(n) \\
&= \mathbf{w}_o - [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M] \begin{bmatrix} \nu_1(n) \\ \nu_2(n) \\ \vdots \\ \nu_M(n) \end{bmatrix} \\
&= \mathbf{w}_o - \sum_{k=1}^{M} \mathbf{q}_k \nu_k(n),
\end{aligned} \tag{4.25}$$

where $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ are the eigenvectors associated, respectively, with the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$ of the correlation matrix $\mathbf{R}$ and the kth natural mode $\nu_k(n)$ is defined by Eq. (4.21). Thus, substituting Eq. (4.21) into Eq. (4.25), we find that the transient behavior of the ith tap weight is described by

$$w_i(n) = w_{oi} - \sum_{k=1}^{N} q_{ki} v_k(0)(1 - \mu\lambda_k)^n, \qquad i = 1, 2, \ldots, M, \tag{4.26}$$

where w_{oi} is the optimum value of the ith tap weight and q_{ki} is the corresponding ith element of the kth eigenvector $\mathbf{q}_k$.

Equation (4.26) shows that each tap weight in the steepest-descent algorithm converges as the weighted sum of exponentials of the form $(1 - \mu\lambda_k)^n$. The time τ_k required for each term to reach $1/e$ of its initial value is given by Eq. (4.23). However, the *overall time constant* τ_a, defined as the time required for the summation term in Eq. (4.26) to decay to $1/e$ of its initial value, cannot be expressed in a simple closed form similar to Eq. (4.23). Nevertheless, the *slowest rate of convergence* is attained when $q_{ki}v_k(0)$ is zero for all k except that mode corresponding to the smallest eigenvalue $\lambda_{\min}$ of matrix $\mathbf{R}$, so the upper bound on τ_a is defined by $-1/\ln(1 - \mu\lambda_{\min})$. The *fastest rate of convergence* is attained when all the $q_{ki}v_k(0)$ are zero except for that mode corresponding to the largest eigenvalue $\lambda_{\max}$, so the lower bound on τ_a is defined by $-1/\ln(1 - \mu\lambda_{\max})$. Accordingly, the overall time constant τ_a for any tap weight of the steepest-descent algorithm is bounded as follows (Griffiths, 1975):

$$\frac{-1}{\ln(1 - \mu\lambda_{\max})} \leq \tau_a \leq \frac{-1}{\ln(1 - \mu\lambda_{\min})}. \tag{4.27}$$

We see, therefore that when the eigenvalues of the correlation matrix $\mathbf{R}$ are widely spread (i.e., the correlation matrix of the tap inputs is ill conditioned), the settling time of the steepest-descent algorithm is limited by the smallest eigenvalues or, which is the same thing, the slowest modes.

Transient Behavior of the Mean-Square Error

We may develop further insight into the operation of the steepest-descent algorithm by examining the transient behavior of the mean-square error $J(n)$. From Eq. (2.56), we have

$$J(n) = J_{\min} + \sum_{k=1}^{M} \lambda_k |v_k(n)|^2, \tag{4.28}$$

where $J_{\min}$ is the minimum mean-square error. The transient behavior of the kth natural mode, $v_k(n)$, is defined by Eq. (4.21). Hence, substituting Eq. (4.21) into Eq. (4.28), we get

$$J(n) = J_{\min} + \sum_{k=1}^{M} \lambda_k (1 - \mu\lambda_k)^{2n} |v_k(0)|^2, \tag{4.29}$$

where $v_k(0)$ is the initial value of $v_k(n)$. When the steepest-descent algorithm is convergent [i.e., when the step-size parameter μ is chosen within the bounds defined by Eq. (4.22)], we see that, irrespective of the initial conditions,

$$\lim_{n \to \infty} J(n) = J_{\min}. \tag{4.30}$$

The curve obtained by plotting the mean-square error $J(n)$ versus the number of adaptation cycles, n, is called a *learning curve*. From Eq. (4.29), we may say:

The learning curve of the steepest-descent algorithm consists of a sum of exponentials, each of which corresponds to a natural mode of the algorithm.

In general, the number of natural modes equals the number of tap weights. In going from the initial value $J(0)$ to the final value $J_{\min}$, the exponential decay for the kth natural mode has a time constant given by

$$\tau_{k,\,\text{mse}} \approx \frac{-1}{2\ln(1 - \mu\lambda_k)}. \tag{4.31}$$

For small values of the step-size parameter μ, we may approximate this time constant as

$$\tau_{k,\,\text{mse}} \approx \frac{1}{2\mu\lambda_k}. \tag{4.32}$$

Equation (4.32) shows that the smaller we make the step-size parameter μ, the slower will be the rate of decay of each natural mode of the steepest-descent algorithm.

4.4 EXAMPLE

In this example, we examine the transient behavior of the steepest-descent algorithm applied to a predictor that operates on a real-valued *autoregressive (AR) process*. Figure 4.7 shows the structure of the predictor, assumed to contain two tap weights that are denoted by $w_1(n)$ and $w_2(n)$; the dependence of these tap weights on the number of adaptation cycles, n, emphasizes the transient condition of the predictor. The AR process $u(n)$ is described by the second-order difference equation

$$u(n) + a_1 u(n-1) + a_2 u(n-2) = \nu(n), \tag{4.33}$$

where the sample $\nu(n)$ is drawn from a white-noise process of zero mean and variance σ_ν^2. The AR parameters a_1 and a_2 are chosen so that the roots of the characteristic equation

$$1 + a_1 z^{-1} + a_2 z^{-2} = 0$$

are complex; that is, $a_1^2 < 4a_2$. The particular values assigned to a_1 and a_2 are determined by the desired *eigenvalue spread* $\chi(\mathbf{R})$, defined as the ratio of the largest eigenvalue of the correlation matrix $\mathbf{R}$ to the smallest eigenvalue. For specified values of a_1 and a_2, the variance σ_ν^2 of the white-noise $\nu(n)$ is chosen to make the process $u(n)$ have variance $\sigma_u^2 = 1$.

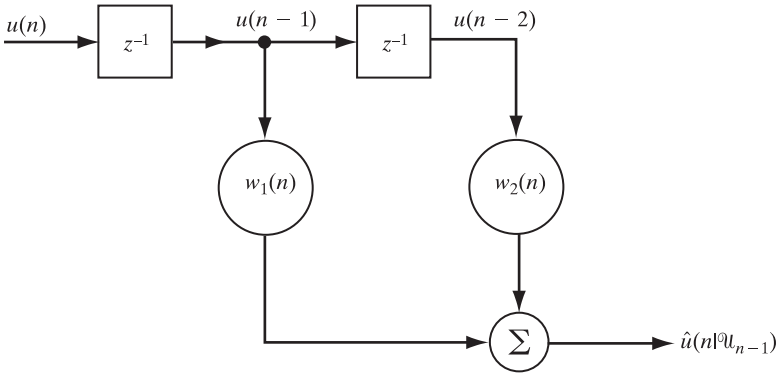


FIGURE 4.7 Two-tap predictor for real-valued input.

The requirement is to evaluate the transient behavior of the steepest-descent algorithm for the following conditions:

- Varying eigenvalue spread $\chi(\mathbf{R})$ and fixed step-size parameter μ.
- Varying step-size parameter μ and fixed eigenvalue spread $\chi(\mathbf{R})$.

Characterization of the AR Process

Since the predictor of Fig. 4.7 has two tap weights and the AR process $u(n)$ is real valued, it follows that the correlation matrix $\mathbf{R}$ of the tap inputs is a 2-by-2 symmetric matrix. That is,

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix},$$

where (see Chapter 1)

$$r(0) = \sigma_u^2$$

and

$$r(1) = -\frac{a_1}{1 + a_2}\,\sigma_u^2,$$

in each of which

$$\sigma_u^2 = \left(\frac{1 + a_2}{1 - a_2}\right)\frac{\sigma_\nu^2}{(1 + a_2)^2 - a_1^2}.$$

The two eigenvalues of $\mathbf{R}$ are

$$\lambda_1 = \left(1 - \frac{a_1}{1 + a_2}\right)\sigma_u^2$$

and

$$\lambda_2 = \left(1 + \frac{a_1}{1 + a_2}\right)\sigma_u^2.$$

Hence, the eigenvalue spread equals (assuming that a_1 is negative)

$$\chi(\mathbf{R}) = \frac{\lambda_1}{\lambda_2} = \frac{1 - a_1 + a_2}{1 + a_1 + a_2}.$$

The eigenvectors associated with the eigenvalues λ_1 and λ_2 are, respectively,

$$\mathbf{q}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

and

$$\mathbf{q}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$

both of which are normalized to unit length.

EXPERIMENT 1 Varying Eigenvalue Spread

In this experiment, the step-size parameter μ is fixed at 0.3, and the evaluations are made for the four sets of AR parameters given in Table 4.1.

For a given set of parameters, we use a two-dimensional plot of the transformed tap-weight error $v_1(n)$ versus $v_2(n)$ to display the transient behavior of the steepest-descent algorithm. In particular, the use of Eq. (4.21) yields

$$\begin{aligned} \mathbf{v}(n) &= \begin{bmatrix} v_1(n) \\ v_2(n) \end{bmatrix} \\ &= \begin{bmatrix} (1 - \mu\lambda_1)^n v_1(0) \\ (1 - \mu\lambda_2)^n v_2(0) \end{bmatrix}, \qquad n = 1, 2, \ldots . \end{aligned} \tag{4.34}$$

To calculate the initial value $\mathbf{v}(0)$, we use Eq. (4.19), assuming that the initial value $\mathbf{w}(0)$ of the tap-weight vector $\mathbf{w}(n)$ is zero. This equation requires knowledge of the optimum tap-weight vector $\mathbf{w}_o$. Now, when the two-tap predictor of Fig. 4.7 is optimized, with the second-order AR process of Eq. (4.33) supplying the tap inputs, we find that the optimum tap-weight vector is

$$\mathbf{w}_o = \begin{bmatrix} -a_1 \\ -a_2 \end{bmatrix}$$

and the minimum mean-square error is

$$J_{\min} = \sigma_v^2.$$

TABLE 4.1 Summary of Parameter Values Characterizing the Second-Order AR Modeling Problem

	AR parameters		Eigenvalues		Eigenvalue spread,	Minimum mean-square error,
Case	a_1	a_2	λ_1	λ_2	$\chi = \lambda_1/\lambda_2$	$J_{\min} = \sigma_v^2$
1	−0.1950	0.95	1.1	0.9	1.22	0.0965
2	−0.9750	0.95	1.5	0.5	3	0.0731
3	−1.5955	0.95	1.818	0.182	10	0.0322
4	−1.9114	0.95	1.957	0.0198	100	0.0038

Accordingly, the use of Eq. (4.19) yields the initial value

$$\begin{aligned}\mathbf{v}(0) &= \begin{bmatrix} v_1(0) \\ v_2(0) \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} -a_1 \\ -a_2 \end{bmatrix} \\ &= \frac{-1}{\sqrt{2}} \begin{bmatrix} a_1 + a_2 \\ a_1 - a_2 \end{bmatrix}.\end{aligned} \tag{4.35}$$

Thus, for the specified parameters, we use Eq. (4.35) to compute the initial value $\mathbf{v}(0)$ and then use Eq. (4.34) to compute $\mathbf{v}(1), \mathbf{v}(2), \ldots$. By joining the points defined by these values of $\mathbf{v}(n)$ for varying n, we obtain a *trajectory* that describes the transient behavior of the steepest-descent algorithm for the particular set of parameters.

It is informative to include in the two-dimensional plot of $v_1(n)$ versus $v_2(n)$ loci representing Eq. (4.28) for fixed values of n. For our example, Eq. (4.28) yields

$$J(n) - J_{\min} = \lambda_1 v_1^2(n) + \lambda_2 v_2^2(n). \tag{4.36}$$

When $\lambda_1 = \lambda_2$ and n is fixed, Eq. (4.36) represents a circle with center at the origin and radius equal to the square root of $[J(n) - J_{\min}]/\lambda$, where λ is the common value of the two eigenvalues. When, on the other hand, $\lambda_1 \neq \lambda_2$, Eq. (4.36) represents (for fixed n) an ellipse with major axis equal to the square root of $[J(n) - J_{\min}]/\lambda_2$ and minor axis equal to the square root of $[J(n) - J_{\min}]/\lambda_1$, with $\lambda_1 > \lambda_2$.

Case 1: Eigenvalue Spread $\chi(\mathbf{R}) = 1.22$. For the parameter values given for Case 1 in Table 4.1, the eigenvalue spread $\chi(\mathbf{R})$ equals 1.22; that is, the eigenvalues λ_1 and λ_2 are approximately equal. The use of these parameter values in Eqs. (4.34) and (4.35) yields the trajectory of $[v_1(n), v_2(n)]$ shown in Fig. 4.8(a), with n as running parameter. The use of the same parameter values in Eq. (4.36) yields the (approximately) circular loci shown for fixed values of $J(n)$, corresponding to $n = 0, 1, 2, 3, 4$.

We may also display the transient behavior of the steepest-descent algorithm by plotting the tap weight $w_1(n)$ versus $w_2(n)$. In particular, for our example, the use of Eq. (4.25) yields the tap-weight vector

$$\begin{aligned}\mathbf{w}(n) &= \begin{bmatrix} w_1(n) \\ w_2(n) \end{bmatrix} \\ &= \begin{bmatrix} -a_1 - (v_1(n) + v_2(n))/\sqrt{2} \\ -a_2 - (v_1(n) - v_2(n))/\sqrt{2} \end{bmatrix}.\end{aligned} \tag{4.37}$$

The corresponding trajectory of $[w_1(n), w_2(n)]$, with n as running parameter, obtained by using Eq. (4.28), is shown in Fig. 4.9(a) . Here, again, we have included the loci of $[w_1(n), w_2(n)]$ for fixed values of $J(n)$ corresponding to $n = 0, 1, 2, 3, 4, 5$. Note that these loci, unlike those of Fig. 4.8(a), are ellipsoidal.

Case 2: Eigenvalue Spread $\chi(\mathbf{R}) = 3$. The use of the parameter values for Case 2 of Table 4.1 in Eqs. (4.34) and (4.35) yields the trajectory of $[v_1(n), v_2(n)]$ shown in

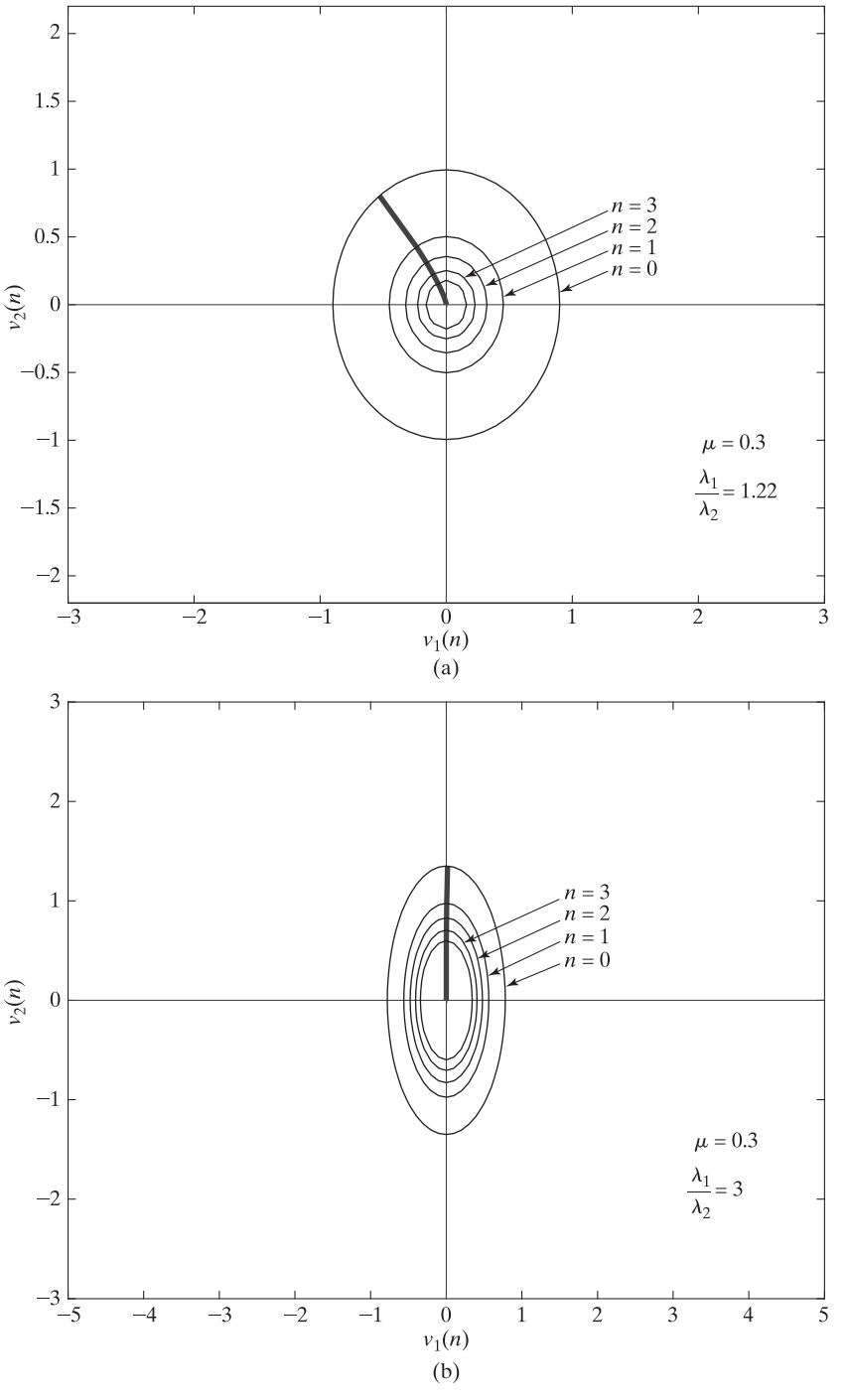
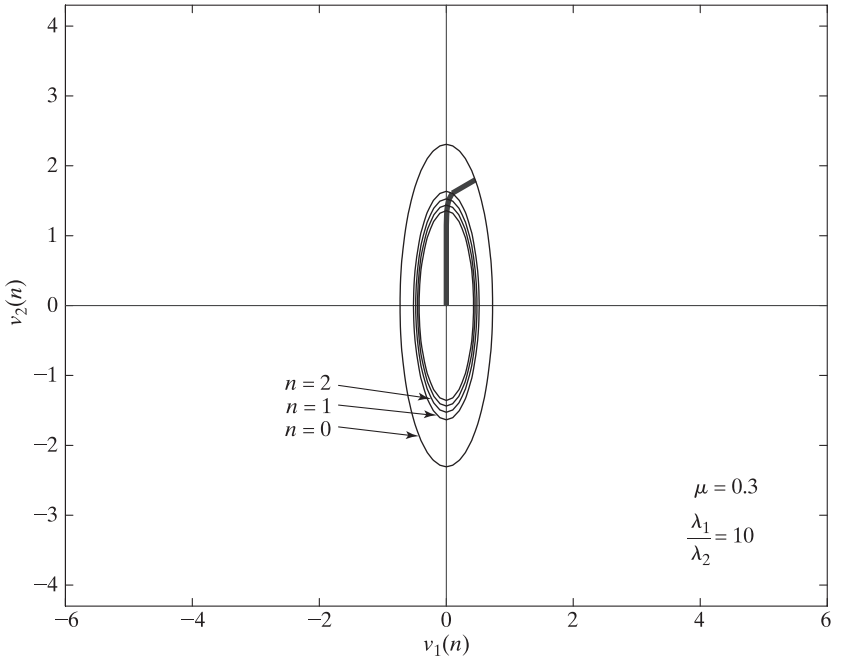
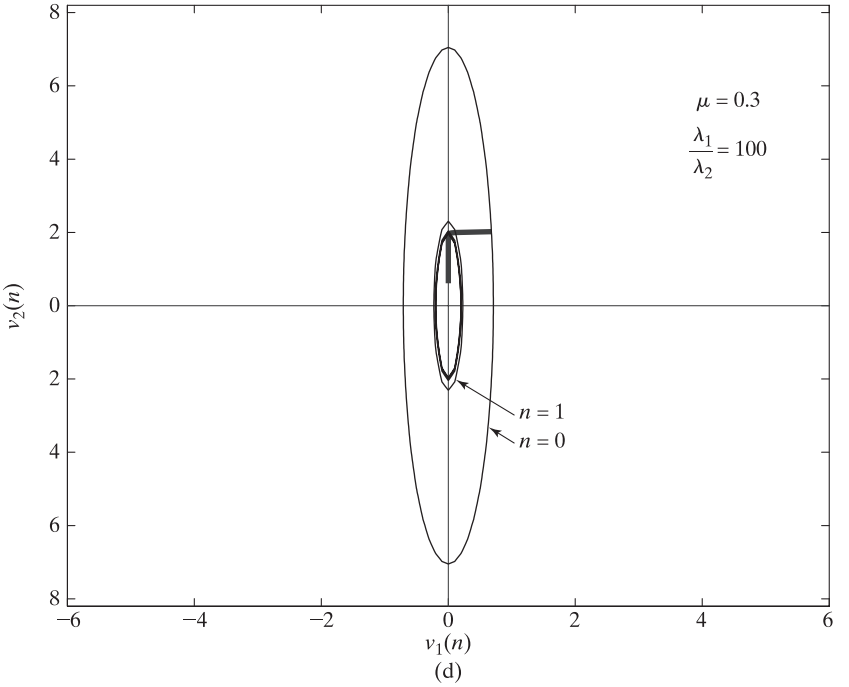


FIGURE 4.8 Loci of $v_1(n)$ versus $v_2(n)$ for the steepest-descent algorithm with step-size parameter $\mu = 0.3$ and varying eigenvalue spread: (a) $\chi(\mathbf{R}) = 1.22$; (b) $\chi(\mathbf{R}) = 3$; (c) $\chi(\mathbf{R}) = 10$; (d) $\chi(\mathbf{R}) = 100$.



(c)



(d)

FIGURE 4.8 (*continued from page 231*)

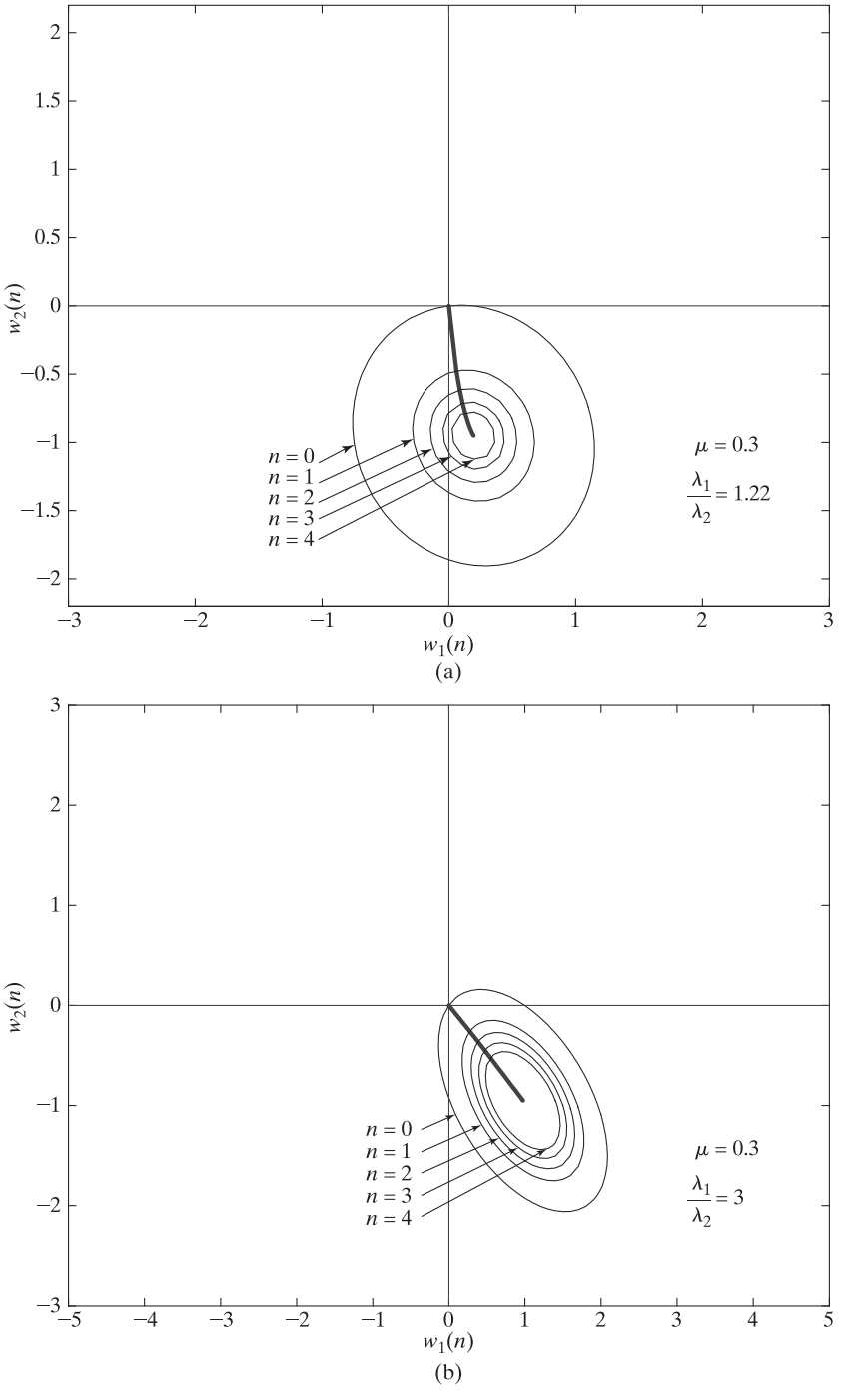
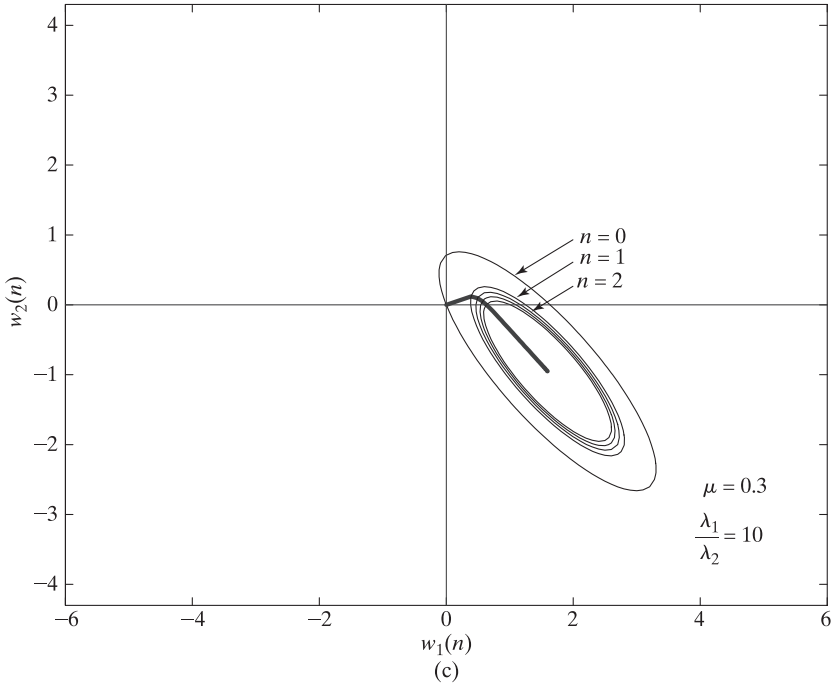
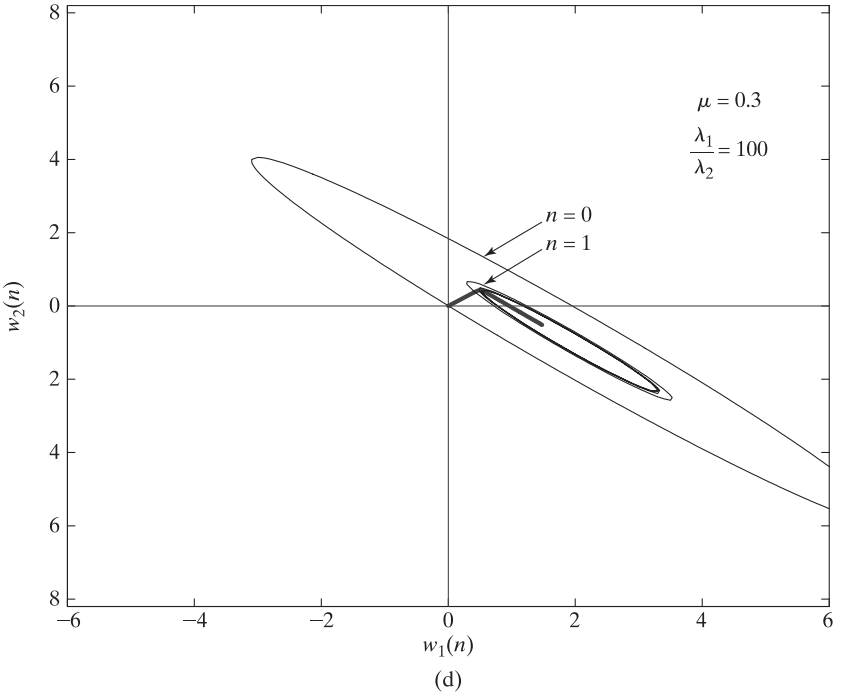


FIGURE 4.9 Loci of $w_1(n)$ versus $w_2(n)$ for the steepest-descent algorithm with step-size parameter $\mu = 0.3$ and varying eigenvalue spread: (a) $\chi(\mathbf{R}) = 1.22$; (b) $\chi(\mathbf{R}) = 3$; (c) $\chi(\mathbf{R}) = 10$; (d) $\chi(\mathbf{R}) = 100$.



(c)



(d)

FIGURE 4.9 (*continued from page 233*)

Fig. 4.8(b), with n as running parameter, and the use of the same parameter values in Eq. (4.36) yields the ellipsoidal loci shown for the fixed values of $J(n)$ for $n = 0, 1, 2, 3, 4, 5$. Note that, for this set of parameter values, the initial value $v_2(0)$ is approximately zero, so the initial value $\mathbf{v}(0)$ lies practically on the v_1-axis.

The corresponding trajectory of $[w_1(n), w_2(n)]$, with n as running parameter, is shown in Fig. 4.9b.

Case 3: Eigenvalue Spread $\chi(\mathbf{R}) = 10$. For this case, the application of Eqs. (4.34) and (4.35) yields the trajectory of $[v_1(n), v_2(n)]$ shown in Fig. 4.8(c), with n as running parameter, and the application of Eq. (4.36) yields the ellipsoidal loci included in the figure for fixed values of $J(n)$ for $n = 0, 1, 2, 3, 4, 5$. The corresponding trajectory of $[w_1(n), w_2(n)]$, with n as running parameter, is shown in Fig. 4.9(c).

Case 4: Eigenvalue Spread $\chi(\mathbf{R}) = 100$. For this case, the application of the preceding equations yields the results shown in Fig. 4.8(d) for the trajectory of $[v_1(n), v_2(n)]$ and the ellipsoidal loci for fixed values of $J(n)$. The corresponding trajectory of $[w_1(n), w_2(n)]$ is shown in Fig. 4.9(d).

In Fig. 4.10, we have plotted the mean-square error $J(n)$ versus n for the four eigenvalue spreads 1.22, 3, 10, and 100. We see that, as the eigenvalue spread increases (and the input process becomes more correlated), the minimum mean-square error $J_{\min}$ decreases. This observation makes intuitive sense: The predictor should do a better job tracking a strongly correlated input process than a weakly correlated one.

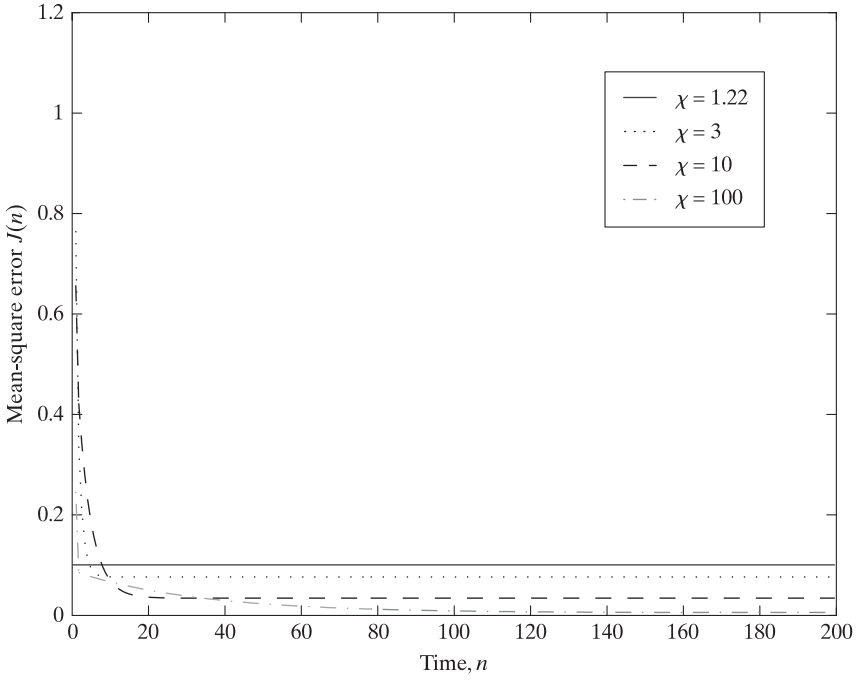


FIGURE 4.10 Learning curves of steepest-descent algorithm with step-size parameter $\mu = 0.3$ and varying eigenvalue spread.

EXPERIMENT 2 Varying Step-Size Parameter

In this experiment, the eigenvalue spread is fixed at $\chi(\mathbf{R}) = 10$, and the step-size parameter μ is varied. In particular, we examine the transient behavior of the steepest-descent algorithm for $\mu = 0.3$ and 1.0. The corresponding results in terms of the transformed tap-weight errors $v_1(n)$ and $v_2(n)$ are shown in parts (a) and (b) of Fig. 4.11, respectively. The results included in part (a) of this figure are the same as those in Fig. 4.8(c). Note that, in accordance with Eq. (4.22), the critical value of the step-size parameter equals $\mu_{\max} = 2/\lambda_{\max} = 1.1$, which is slightly in excess of the actual value $\mu = 1$ used in Fig. 4.11(b).

The results for $\mu = 0.3$ and 1.0 in terms of the tap weights $w_1(n)$ and $w_2(n)$ are shown in parts (a) and (b) of Fig. 4.12, respectively. Here again, the results included in part (a) of the figure are the same as those in Fig. 4.9(c).

Observations

On the basis of the results presented for Experiments 1 and 2, we may make the following observations:

1. The trajectory of $[v_1(n), v_2(n)]$, with the number of adaptation cycles n as running parameter, is normal to the locus of $[v_1(n), v_2(n)]$ for fixed $J(n)$. This statement also applies to the trajectory of $[w_1(n), w_2(n)]$ for fixed $J(n)$.
2. When the eigenvalues λ_1 and λ_2 are equal, the trajectory of $[v_1(n), v_2(n)]$ or that of $[w_1(n), w_2(n)]$, with n as running parameter, is practically a straight line. This situation is illustrated in Fig. 4.8(a) or 4.9(a), for which the eigenvalues λ_1 and λ_2 are approximately equal.
3. When the conditions are right for the initial value $\mathbf{v}(0)$ of the transformed tap-weight error vector $\mathbf{v}(n)$ to lie on the v_1-axis or v_2-axis, the trajectory of $[v_1(n), v_2(n)]$, with n as running parameter, is a straight line. This situation is illustrated in Fig. 4.8(b), where $v_1(0)$ is approximately zero. Correspondingly, the trajectory of $[w_1(n), w_2(n)]$, with n as running parameter, is also a straight line, as illustrated in Fig. 4.9(b).
4. Except for two special cases—(1) equal eigenvalues and (2) the right choice of initial conditions—the trajectory of $[v_1(n), v_2(n)]$, with n as running parameter, follows a curved path, as illustrated in Fig. 4.8(c). Correspondingly, the trajectory of $[w_1(n), w_2(n)]$, with n as running parameter, also follows a curved path, as illustrated in Fig. 4.9(c). When the eigenvalue spread is very high (i.e., the input data are very highly correlated), two things happen:
   - The error-performance surface assumes the shape of a deep valley.
   - The trajectories of $[v_1(n), v_2(n)]$ and $[w_1(n), w_2(n)]$ develop distinct bends.

   Both of these points are well illustrated in Figs. 4.8(d) and 4.9(d), respectively, for the case of $\chi(\mathbf{R}) = 100$.
5. The steepest-descent algorithm converges fastest when the eigenvalues λ_1 and λ_2 are equal or the starting point of the algorithm is chosen properly, for which cases the trajectory formed by joining the points $v(0), v(1), v(2), \ldots$, is a straight line, the shortest possible path.

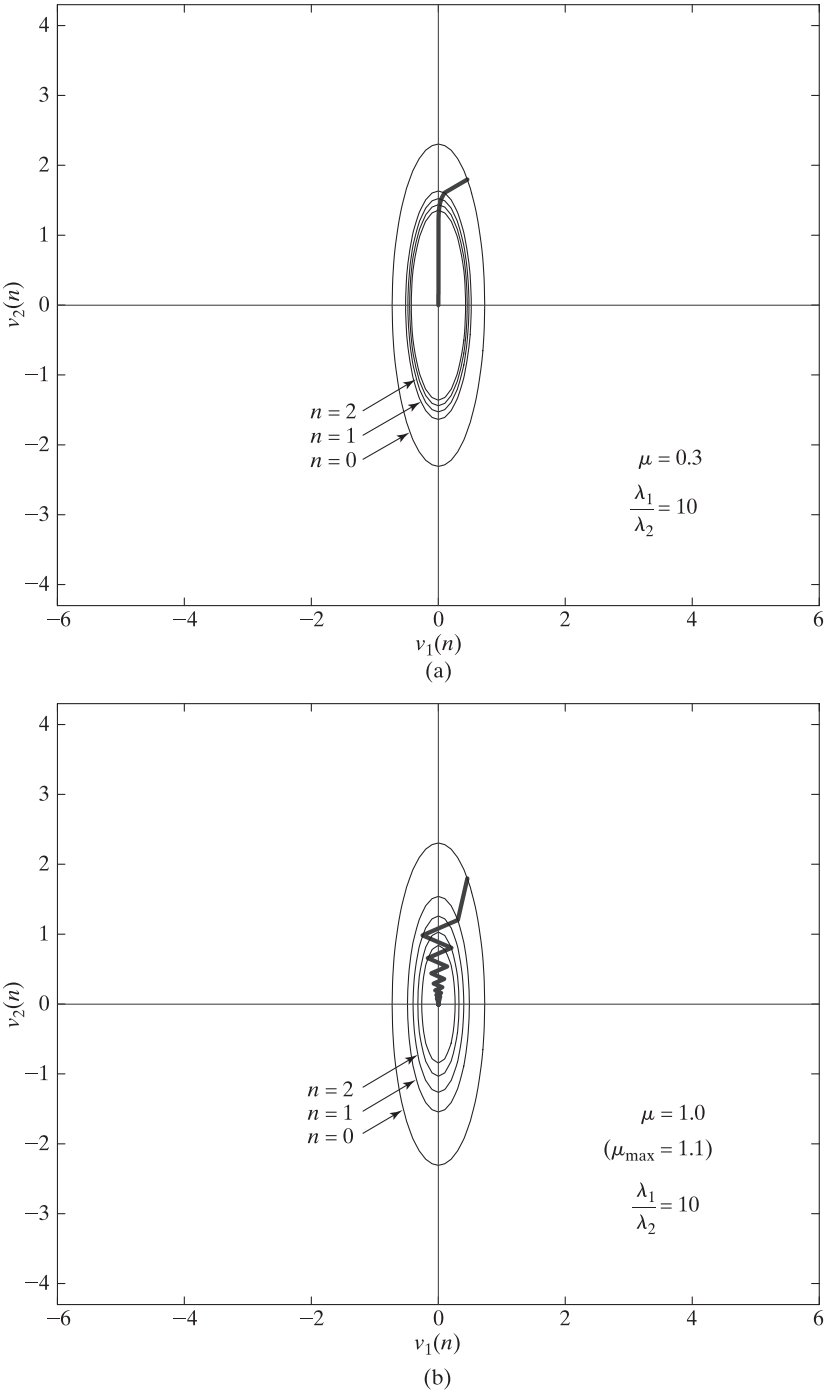


FIGURE 4.11 Loci of $v_1(n)$ versus $v_2(n)$ for the steepest-descent algorithm with eigenvalue spread $\chi(\mathbf{R}) = 10$ and varying step-size parameters: (a) overdamped, $\mu = 0.3$; (b) underdamped, $\mu = 1.0$.

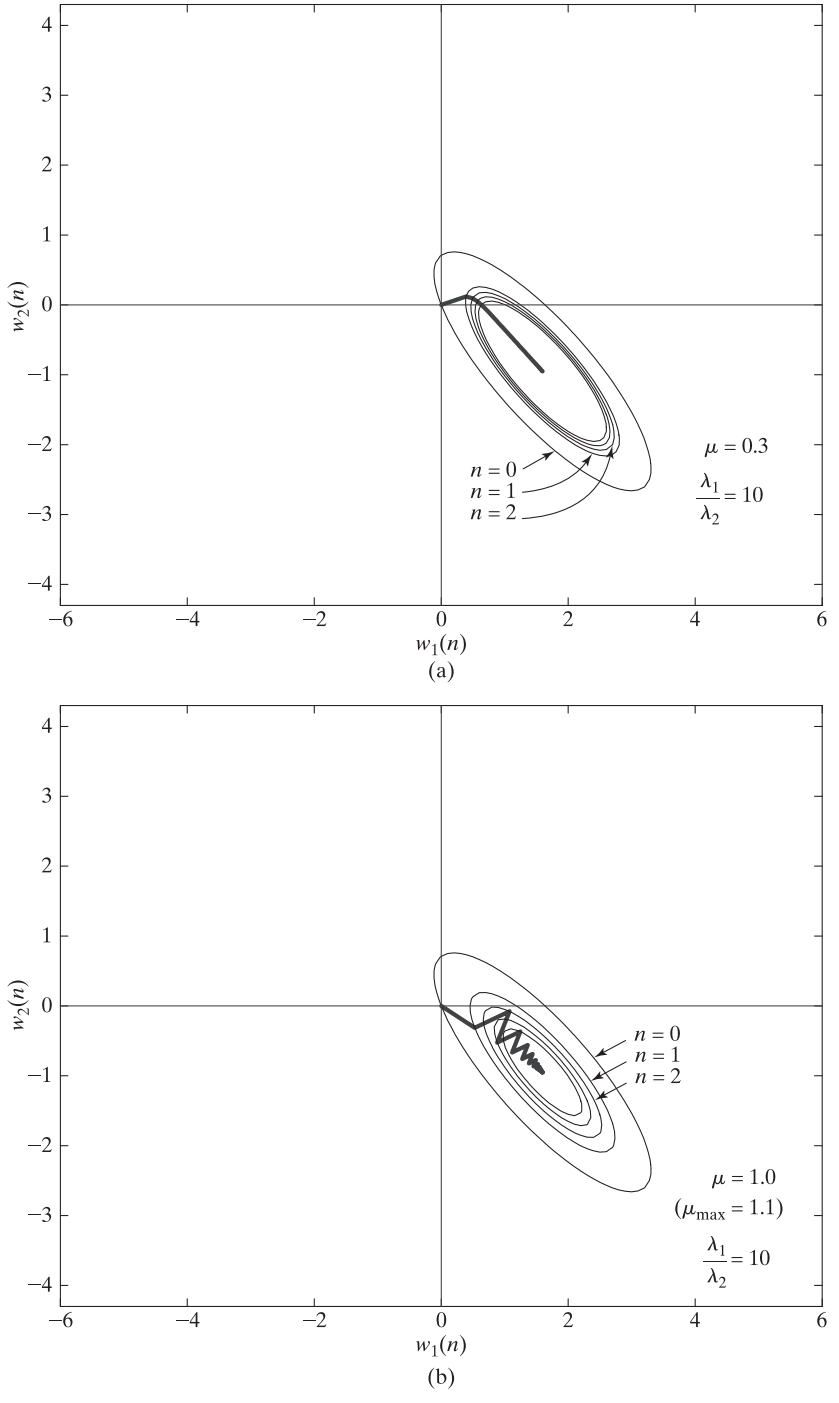


FIGURE 4.12 Loci of $w_1(n)$ versus $w_2(n)$ for the steepest-descent algorithm with eigenvalue spread $\chi(\mathbf{R}) = 10$ and varying step-size parameters: (a) overdamped, $\mu = 0.3$; (b) underdamped, $\mu = 1.0$.

6. For fixed step-size parameter μ, as the eigenvalue spread $\chi(\mathbf{R})$ increases (i.e., the correlation matrix $\mathbf{R}$ of the tap inputs becomes more ill conditioned), the ellipsoidal loci of $[v_1(n), v_2(n)]$ for fixed values of $J(n)$, for $n = 0, 1, 2, \ldots$, become increasingly narrower (i.e., the minor axis becomes smaller) and more crowded.
7. When the step-size parameter μ is small, the transient behavior of the steepest-descent algorithm is *overdamped*, in that the trajectory formed by joining the points $\mathbf{v}(0), \mathbf{v}(1), \mathbf{v}(2), \ldots$, follows a continuous path. When, on the other hand, μ approaches the maximum allowable value $\mu_{\max} = 2/\lambda_{\max}$, the transient behavior of the steepest-descent algorithm is *underdamped*, in that this trajectory exhibits oscillations. These two different forms of transient behavior are illustrated in Fig. 4.11 in terms of $v_1(n)$ and $v_2(n)$. The corresponding results in terms of $w_1(n)$ and $w_2(n)$ are presented in Fig. 4.12.

The conclusion to be drawn from these observations is that the transient behavior of the steepest-descent algorithm is highly sensitive to variations in both the step-size parameter μ and the eigenvalue spread of the correlation matrix of the tap inputs.

4.5 THE STEEPEST-DESCENT ALGORITHM VIEWED AS A DETERMINISTIC SEARCH METHOD

The error-performance surface of an adaptive FIR filter operating in a wide-sense stationary stochastic environment is a bowl-shaped (i.e., quadratic) surface with a distinct minimum point. The steepest-descent algorithm provides a local search method for seeking the minimum point of the error-performance surface, starting from an arbitrary initial point. For its operation, the steepest-descent algorithm depends on three quantities:

- The *starting point*, which is specified by the initial value $\mathbf{w}(0)$ of the tap-weight vector.
- The *gradient vector*, which, at a particular point on the error-performance surface (i.e., a particular value of the tap-weight vector), is uniquely determined by the cross-correlation vector $\mathbf{p}$ and the correlation matrix $\mathbf{R}$ that characterize the environment.
- The *step-size parameter* μ, which controls the size of the incremental change applied to the tap-weight vector of the FIR filter from one adaptation cycle of the algorithm to the next; for stability of the algorithm, μ must satisfy the condition of Eq. (4.22).

Once these three quantities are specified, the steepest-descent algorithm follows a distinct path in the multidimensional weight space, starting from the initial point $\mathbf{w}(0)$ and ultimately terminating on the optimum solution $\mathbf{w}_o$. In other words, the steepest-descent algorithm is a *deterministic search method* in weight space. This statement is confirmed by the experimental results presented in Section 4.4. In theory, the algorithm requires an "infinite" number of adaptation cycles to move from the starting point $\mathbf{w}(0)$ to the optimum point $\mathbf{w}_o$. However, in practice, we need to execute just a "finite" number of adaptation cycles of the algorithm for the FIR filter to attain a tap-weight vector close enough to the optimum solution $\mathbf{w}_o$—closeness is clearly a subjective matter that can be determined only by a designer's objective.

4.6 VIRTUE AND LIMITATION OF THE STEEPEST-DESCENT ALGORITHM

The important virtue of the steepest-descent algorithm is the *simplicity* of its implementation, which is readily seen from Eq. (4.10). However, as pointed out in Section 4.5, we may require a large number of adaptation cycles for the algorithm to converge to a point sufficiently close to the optimum solution $\mathbf{w}_o$. This performance limitation is due to the fact that the steepest-descent algorithm is based on a straight-line (i.e., first-order) approximation of the error-performance surface around the current point.

Newton's Method

To overcome the aforesaid limitation of the steepest-descent algorithm, we may use a quadratic (i.e., second-order) approximation of the error-performance surface around the current point denoted by $\mathbf{w}(n)$. To this end, we may invoke the second-order *Taylor series* expansion of the cost function $J(\mathbf{w})$ around $\mathbf{w}(n)$. Such an expansion requires knowledge of the *gradient* (first-order) as well as the *Hessian* (second-order) of the cost function $J(\mathbf{w})$. When the training data [i.e., u(n) and $d(n)$] are *complex*, evaluation of the Hessian is much more difficult than that of the gradient; for details, see the Taylor series expansion in Appendix B on the Wirtinger calculus. With this section being of rather limited scope, we simplify matters by focusing on real data.

Under this limitation, we may express the second-order Taylor series expansion of the cost function $J(\mathbf{w})$ as follows:

$$J(\mathbf{w}) \approx J(\mathbf{w}(n)) + (\mathbf{w} - \mathbf{w}(n))^{\mathrm{T}}\mathbf{g}(n) + \frac{1}{2}(\mathbf{w} - \mathbf{w}(n))^{\mathrm{T}}\mathbf{H}(n)(\mathbf{w} - \mathbf{w}(n)), \quad (4.38)$$

where the superscript T denotes matrix transposition, the vector

$$\mathbf{g}(n) = \left.\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}(n)} \quad (4.39)$$

is the gradient evaluated at $\mathbf{w}(n)$, and the matrix

$$\mathbf{H}(n) = \left.\frac{\partial^2 J(\mathbf{w})}{\partial \mathbf{w}^2}\right|_{\mathbf{w}=\mathbf{w}(n)} \quad (4.40)$$

is the Hessian of the cost function evaluated at $\mathbf{w}(n)$. The straight-line approximation of $J(\mathbf{w})$ around the current point $\mathbf{w}(n)$ is clearly a simplification of Eq. (4.38). Differentiating Eq. (4.38) with respect to $\mathbf{w}$ and setting the result to zero, we find that the next iterate (i.e., the updated point on the error-performance surface) is given by

$$\mathbf{w}(n + 1) = \mathbf{w}(n) - \mathbf{H}^{-1}\mathbf{g}(n), \quad (4.41)$$

where $\mathbf{H}^{-1}(n)$ is the *inverse* of the Hessian $\mathbf{H}(n)$. This iterative equation is the pure *Newton method* of optimization theory. (See Problem 15 for a modified form of Newton's algorithm.)

For the quadratic cost function of Eq. (4.8), the gradient vector is defined by Eq. (4.9). Moreover, differentiating the last line of Eq. (4.9) with respect to $\mathbf{w}(n)$, we get

$$\mathbf{H}(n) = 2\mathbf{R}. \quad (4.42)$$

That is, except for a scaling factor, the Hessian of the quadratic cost function of Eq. (4.8) is exactly equal to the correlation matrix $\mathbf{R}$ of the tap-input vector $\mathbf{u}(n)$. Hence, substituting Eqs. (4.9) and (4.42) into Eq. (4.41), we obtain

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) - \frac{1}{2}\mathbf{R}^{-1}(-2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n)) \\ &= \mathbf{R}^{-1}\mathbf{p} \\ &= \mathbf{w}_o. \end{aligned} \tag{4.43}$$

Equation (4.43) shows that Newton's method attains the optimum solution $\mathbf{w}_o$ from an arbitrary point $\mathbf{w}(n)$ on the error surface in a single adaptation cycle. However, this improvement in performance requires the inversion of the correlation matrix $\mathbf{R}$, the foregoing of which is the very thing that motivates the use of the steepest-descent algorithm.

The conclusion to be drawn from this section is that if computational simplicity is of paramount importance, then the method of steepest descent is the preferred iterative method for computing the tap-weight vector of an adaptive FIR filter operating in a wide-sense stationary environment. If, on the other hand, the rate of convergence is the issue of interest, then Newton's method or a modified version of it is the preferred approach.

4.7 SUMMARY AND DISCUSSION

The method of steepest descent provides a simple procedure for computing the tap-weight vector of a Wiener filter, given knowledge of two ensemble-average quantities:

- The correlation matrix of the tap-input vector.
- The cross-correlation vector between the tap-input vector and the desired response.

A critical feature of the method of steepest descent is the presence of feedback, which is another way of saying that the underlying algorithm is recursive in nature. As such, we have to pay particular attention to the issue of stability, which is governed by two parameters in the feedback loop of the algorithm:

- The step-size parameter μ.
- The correlation matrix $\mathbf{R}$ of the tap-input vector.

Specifically, the necessary and sufficient condition for stability of the algorithm is embodied in the condition

$$0 < \mu < \frac{2}{\lambda_{\max}},$$

where $\lambda_{\max}$ is the largest eigenvalue of the correlation matrix $\mathbf{R}$.

Moreover, depending on the value assigned to the step-size parameter μ, the transient response of the steepest-descent algorithm may exhibit one of three forms of behavior:

- *Underdamped response*, in which case the trajectory followed by the tap-weight vector toward the optimum Wiener solution exhibits oscillations; this response arises when the step-size parameter μ is large.

- *Overdamped response*, which is a nonoscillatory behavior that arises when μ is small.
- *Critically damped response*, which is the fine dividing line between the underdamped and overdamped conditions.

Unfortunately, in general, these conditions do not lend themselves to an exact mathematical analysis; they are usually evaluated by experimentation.

PROBLEMS

1. Consider a Wiener filtering problem characterized by the following values for the correlation matrix $\mathbf{R}$ of the tap-input vector $\mathbf{u}(n)$ and the cross-correlation vector $\mathbf{p}$ between $\mathbf{u}(n)$ and the desired response $d(n)$:

$$\mathbf{R} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix};$$
$$\mathbf{p} = \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix}.$$

(a) Suggest a suitable value for the step-size parameter μ that would ensure convergence of the method of steepest descent, based on the given value for matrix $\mathbf{R}$.

(b) Using the value proposed in part (a), determine the recursions for computing the elements $w_1(n)$ and $w_2(n)$ of the tap-weight vector $\mathbf{w}(n)$. For this computation, you may assume the initial values

$$w_1(0) = w_2(0) = 0.$$

(c) Investigate the effect of varying the step-size parameter μ on the trajectory of the tap-weight vector $\mathbf{w}(n)$ as n varies from zero to infinity.

2. Consider a Wiener filter characterized by the following values for the correlation matrix $\mathbf{R}$ of the tap-input vector $\mathbf{u}(n)$ and the cross-correlation vector $\mathbf{P}$ between $u(n)$ and the desired response $d(n)$:

$$R = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$
$$P = \begin{bmatrix} 0.25 \\ 0.5 \end{bmatrix}$$

Based on the given **R,** suggest a suitable value for the step-size parameter μ that would ensure convergence by the method of steepest descent.

3. From Problem 2, determine the recursions for $w_1(n)$ and $w_2(n)$ of the tap-weight vector $w(n)$. [*Hint*: Assume the initial values $w_1(0) = w_2(0) = 0$.]

4. Starting with the formula for the estimation error, viz.,

$$e(n) = d(n) - \mathbf{w}^{\mathrm{H}}(n)\mathbf{u}(n),$$

where $d(n)$ is the desired response, $\mathbf{u}(n)$ is the tap-input vector, and $\mathbf{w}(n)$ is the tap-weight vector in the FIR filter, show that the gradient of the instantaneous square error $|e(n)|^2$ equals

$$\hat{\nabla} J(n) = -2\mathbf{u}(n)d^*(n) + 2\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{w}(n).$$

5. In this problem, we explore another way of deriving the steepest-descent algorithm of Eq. (4.9) used to adjust the tap-weight vector in an FIR filter. The inverse of a positive-definite matrix may be expanded in a series as

$$\mathbf{R}^{-1} = \mu \sum_{k=0}^{\infty} (\mathbf{I} - \mu\mathbf{R})^k,$$

where $\mathbf{I}$ is the identity matrix and μ is a positive constant. To ensure that the series converges, the constant μ must lie inside the range

$$0 < \mu < \frac{2}{\lambda_{\max}},$$

where $\lambda_{\max}$ is the largest eigenvalue of the matrix $\mathbf{R}$. By using this series expansion for the inverse of the correlation matrix in the Wiener–Hopf equations, develop the recursion

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu[\mathbf{p} - \mathbf{R}\mathbf{w}(n)],$$

where

$$\mathbf{w}(n) = \mu \sum_{k=0}^{n-1} (\mathbf{I} - \mu\mathbf{R})^k \mathbf{p}$$

is the approximation to the Wiener solution for the tap-weight vector.

6. Justify by validation that the steepest-descent algorithm becomes unstable when the step-size parameter μ is assigned a negative value.

7. Consider a single-order autoregressive (AR) process $u(n) = -au(n) + u(n)$, where a is the AR parameter and $v(n)$ is a zero-mean white noise of variance σ_v^2. Set up a linear predictor of order one to compute the parameter a. [*Hint*: Use the method of steepest descent for the recursive computation of the Wiener solution for the parameter.]

8. Equation (4.29) defines the transient behavior of the mean-square error $J(n)$ for varying n that is produced by the steepest-descent algorithm. Let $J(0)$ and $J(\infty)$ denote the initial and final values of $J(n)$. Suppose that we approximate this transient behavior with the single exponential

$$J_{\text{approx}}(n) = [J(0) - J(\infty)]e^{-n/r} + J(\infty),$$

where τ is termed the *effective time constant*. Let τ be chosen such that

$$J_{\text{approx}}(1) = J(1).$$

Show that *the initial rate of convergence* of the steepest-descent algorithm, defined as the inverse of τ, is given by

$$\frac{1}{\tau} = \ln\left[\frac{J(0) - J(\infty)}{J(1) - J(\infty)}\right].$$

Using Eq. (4.29), find the value of $1/\tau$. Assume that the initial value $\mathbf{w}(0)$ is zero and that the step-size parameter μ is small.

9. Consider an autoregressive (AR) process of order one, described by the difference equation

$$u(n) = -au(n-1) + v(n),$$

where a is the AR parameter of the process and $v(n)$ is a zero-mean white noise of variance σ_v^2.

(a) Set up a linear predictor of order one to compute the parameter a. Specifically, use the method of steepest descent for the recursive computation of the Wiener solution for the parameter a.

(b) Plot the error-performance curve for this problem, identifying the minimum point of the curve in terms of known parameters.

(c) What is the condition on the step-size parameter μ to ensure stability? Justify your answer.

10. Figure 4.7 has two tap weights and the AR process $u(n)$ is real valued. The correlation matrix R of the tap inputs is a 2-by-2 symmetric matrix. Show that the eigenvectors associated with the eigenvalues λ_1 and λ_2 are normalized to unit length.

11. Examine the transient behavior of the steepest-descent algorithm applied to a predictor that operates on a real-valued AR with the second-order difference equation $u(n) - a_1u(n-1) + a_2u(n-2) = v(n)$.

12. Determine the correlation matrix $\mathbf{R}$ of a prediction-error filter applied to a third-order AR process $u(n) = -0.5u(n-1) + 0.5u(n-2) + 0.5u(n-3) + v(n)$ where $u(n)$ is white noise of zero mean and unit variance. [*Hint*: The method of steepest descent is used for recursive computation of the coefficient vector of the prediction-error filter.]

13. Consider a moving-average (MA) process of order one described by the difference equation

$$u(n) = v(n) - 0.2v(n-1),$$

where $v(n)$ is white noise of zero mean and unit variance. The MA process is applied to a forward linear predictor.

(a) The MA process $u(n)$ is approximated by a second-order AR process. Utilizing this approximation, determine the bounds on the step-size parameter μ of the steepest-descent algorithm used to recursively compute the weight vector of the predictor.

(b) Repeat the computation in part (a), but this time approximate $u(n)$ by a third-order AR process.

(c) Discuss the nature of differences between the results computed in parts (a) and (b).

14. An autoregressive moving-average (ARMA) process of order (1,1) is described by the difference equation

$$u(n) = -0.5u(n-1) + v(n) - 0.2v(n-1),$$

where $v(n)$ is white noise of zero mean and unit variance. The process $u(n)$ is applied to a prediction-error filter. To compute the coefficient vector of this filter, the ARMA process is approximated by a third-order AR process.

(a) Determine the coefficients of the approximating AR process and therefore the entries of the corresponding 3-by-3 correlation matrix $\mathbf{R}$.

(b) Compute the eigenvalues of the correlation matrix $\mathbf{R}$ determined in part (a).

(c) Determine the bounds on the step-size parameter μ of the steepest-descent algorithm used to recursively compute the coefficient vector of the prediction-error filter.

15. Introducing a step-size parameter μ into the formulation of Newton's equation, we modify Eq. (4.43) as follows:

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu\mathbf{R}^{-1}(\mathbf{p} - \mathbf{R}\mathbf{w}(n)) \\ &= (1-\mu)\mathbf{w}(n) + \mu\mathbf{R}^{-1}\mathbf{p}.\end{aligned}$$

Following a procedure similar to that described in Section 4.3 on the steepest-descent method, investigate the transient behavior of Newton's algorithm as formulated herein. That is, show that

(a) the transient behavior of Newton's algorithm is characterized by a single exponential whose time constant τ is defined by

$$(1-\mu)^{2k} = e^{-k/\tau};$$

(b) for μ small compared with unity, the time constant is

$$\tau \approx \frac{1}{2\mu}.$$

CHAPTER 5

Method of Stochastic Gradient Descent

In the preceding chapter, we studied the method of steepest descent for formulating recursive computation of the Wiener filter under the following two assumptions:

1. *Joint stationarity of the environment*, from which the regressor (i.e., input vector) $\mathbf{u}(n)$ and the corresponding desired response $d(n)$ are picked for $n = 1, 2, \ldots$.
2. *Information about the environment*, which is made up of the correlation matrix, $\mathbf{R}$, of $\mathbf{u}(n)$ and the cross-correlation vector, $\mathbf{p}$, between $\mathbf{u}(n)$ and $d(n)$.

Insofar as formulation of the cost function for the Wiener filter is concerned, namely, $J(\mathbf{w}(n))$, where $\mathbf{w}(n)$ is the unknown tap-weight vector, the information made up of $\mathbf{R}$ and $\mathbf{p}$ is perfect for the discussion at hand [see Eq. (4.8)]. As pointed out previously in Section 4.5, the method of steepest descent is *deterministic*, in the sense that this information is all that is required to determine the gradient (i.e., the search direction) from one adaptation cycle to the next.

Unfortunately, in many situations encountered in practice, the needed information about the environment contained in $\mathbf{R}$ and $\mathbf{p}$ is not available. We therefore have to look to a new class of algorithms with the built-in capability to *adapt* to statistical variations in the unknown environment. Such algorithms are collectively called *adaptive filtering algorithms*. There are two different methods for deriving these algorithms:

1. One based on the method of stochastic gradient descent, which is discussed in this chapter.
2. The other one based on the method of least squares, which is deferred to Chapter 9.

5.1 PRINCIPLES OF STOCHASTIC GRADIENT DESCENT

"Stochastic" is of Greek origin; it is a generic term, which signifies that we have a "random choice" in finding a gradient for an adaptive filtering algorithm to iterate from one adaptation cycle to the next. As different as they both are, the methods of stochastic gradient descent[1] and steepest descent do share a common property: They are both *local methods* of filtering.

[1]The method of stochastic gradient descent has a long history. It was originally introduced into the statistics literature by Robbins and Monro (1951), where the method was used for solving certain sequential parameter estimation problems. For detailed treatments of the method, the reader is referred to Kushner and Clark (1978), Kushner (1984), and Spall (2003).

Over and above its application to the widely used least-mean-square (LMS) algorithm and its variants, the method of stochastic gradient descent has been applied to stochastic control (Stengel, 1986), self-organized maximum eigenfiltering (Oja, 1982), nearest-neighbor clustering (Duda et al., 2001), back-propagation algorithm for supervised training of multilayer perceptrons (Rumelhart & McLelland, 1986), and reinforcement learning (Sutton, 1992).

Optimization and Complexity

In a rigorous sense, the method of stochastic gradient descent does not qualify to be viewed as an optimization algorithm for the following simple reason:

> *On account of its stochastic nature, stochastic gradient descent can never reach the desired optimum solution of a convex optimization problem; rather, once it reaches the local neighborhood of the optimum solution, it continually wanders around that solution in a random-walk fashion and therefore never settles down to an equilibrium point.*

As such, the method of stochastic gradient descent is *suboptimal*.

The method makes up for this computational deficiency, however, by offering the simplest possible form of complexity: *linear law of scaling* with respect to adjustable parameters. It is therefore popular in practice whenever computational complexity (among other issues) is of interest. This is particularly so when dealing with the processing of information-bearing data whose size grows continually over time.

Efficiency

Another key issue in the study of adaptive filtering algorithms is that of *efficiency*, which may be viewed as a representation of the cost involved in finding a satisfactory solution. There are various ways of measuring efficiency: the amount of computer run time, number of algorithmic adapatation cycles, and rate of convergence. In what follows, the latter measure will be used for the following reason:

> *The rate of convergence not only involves statistical learning theory but also encompasses the Wiener solution as the frame of reference for linear adaptive filtering algorithms, exemplified by the LMS and recursive least-squares (RLS) algorithms, operating in a stationary environment.*

Robustness

Yet another key issue in the study of adaptive filtering algorithms is that of *robustness*, which provides a criterion for assessing how a particular adaptive filtering algorithm maintains a satisfactory performance in the face of unknown disturbances that are typically expected to arise in practice. The classical approach used to evaluate robustness is a deterministic one, rooted in H^{∞} *theory*, which was first formulated in control theory (Zames, 1981); detailed treatment of this property is deferred to Chapter 11. For the present, it suffices to say that robustness and efficiency address two conflicting aspects of an adaptive filtering algorithm's behavior, more on which will also be said in Chapter 11.

Curse of Dimensionality

In general, the volume of a search space, inside which the gradient of a cost function is picked at each adaptation cycle of an optimization algorithm, may grow exponentially with dimensionality (i.e., number of adjustable parameters) of the algorithm. This kind of behavior is attributed to the *curse of dimensionality problem* (Bellman, 1961). In the case of stochastic gradient descent algorithms, which feature prominently throughout this book, algorithmic complexity follows a linear law, as pointed out at the beginning of this section. Fortunately, therefore, the curse of dimensionality is of no concern to the study of linear adaptive filtering algorithms rooted in stochastic gradient descent.

Time-Varying Problems

In many practical applications of adaptive filtering algorithms, the relevant environment is *nonstationary*. Therefore, its statistical characterization continually varies over time. In situations of this kind, the "best" solution to an adaptive filtering problem computed now may *not* be the best—or even a good—solution in the future. For an adaptive filtering algorithm to operate successfully in a nonstationary environment, it must have the capability to *track* statistical variations in the environment continually over time.

Monte Carlo Simulations

Given the realities just described, how do we study the performance of an adaptive filtering algorithm experimentally? The answer to this important question lies in the use of Monte Carlo simulations, which provide the experimenter the means not only to gain insight about that algorithm but also to compare it with a different adaptive filtering algorithm under various conditions. For this computer-oriented approach to be informative, it is essential that we do the following:

- First, use a number of independent Monte Carlo runs in the simulations large enough (at least 100 if not more) for the simulation results to be statistically reliable.
- Second, wherever possible, combine together both theory and numerical results in assessing the insights derived from the simulations.

With the material covered thus far on the method of stochastic gradient descent, the stage is set for two applications of the method to linear adaptive filtering.

5.2 APPLICATION 1: LEAST-MEAN-SQUARE (LMS) ALGORITHM

For our first application of the method of stochastic gradient descent, we have chosen the highly popular adaptive filtering algorithm, the LMS algorithm, which was pioneered by Widrow and Hoff (1960). Distinctive features of this algorithm can be summarized as follows:

1. The LMS algorithm is *simple*, meaning that computational complexity of the algorithm scales linearly with the dimensionality of the finite-duration impulse response (FIR) filter, around which the algorithm operates.

2. Unlike the Wiener filter, the algorithm does *not* require knowledge of statistical characteristics of the environment in which it operates.
3. The algorithm is *robust* in a deterministic sense (i.e., single realization of the algorithm) in the face of unknown environmental disturbances.
4. Last but by no means least, the algorithm does *not* require inversion of the correlation matrix of the regressor (i.e., input vector), which, therefore, makes it simpler than its counterpart, namely, the RLS algorithm.

Given this important set of properties, it is not surprising that the LMS algorithm is one of the most popular adaptive filtering algorithms in use.

Structural Description of the LMS Algorithm

Figure 5.1, depicted in three parts, addresses different perspectives of the underlying structure of the LMS algorithm.

The overall block diagram of Fig. 5.1(a) shows the three components that constitute the algorithm:

1. *FIR filter*, which operates on the regressor (input vector) $\mathbf{u}(n)$ to produce an *estimate* of the desired response, denoted by $\hat{d}(n|\mathcal{U}_n)$, where $\mathcal{U}_n$ denotes the space in which the input vector $\mathbf{u}(n)$ resides.
2. *Comparator*, which subtracts the estimate $\hat{d}(n|\mathcal{U}_n)$ from the desired response, $d(n)$, applied to the FIR filter at its output; the resultant is the *estimation error* (also referred to as the *error signal*), denoted by $e(n)$.
3. *Adaptive weight-control mechanism*, the function of which is to control the incremental adjustments applied to the individual tap weights of the FIR filter by exploiting information contained in the estimation error $e(n)$.

Details of the FIR *filter* are presented in Fig. 5.1(b). The tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$ form the elements of the M-by-1 *tap-input vector* $\mathbf{u}(n)$, where $M-1$ is the number of delay elements; these inputs span a multidimensional space denoted by $\mathcal{U}_n$. Correspondingly, the tap weights $\hat{w}_0(n), \hat{w}_1(n), \ldots, \hat{w}_{M-1}(n)$ form the elements of the M-by-1 tap-weight vector $\hat{\mathbf{w}}(n)$. The value computed for this vector using the LMS algorithm represents an estimate whose expected value may come close to the Wiener solution $\mathbf{w}_o$ (for a wide-sense stationary environment) as the number of adaptation cycles, n, approaches infinity.

Figure 5.1(c) presents details of the *adaptive weight-control mechanism*. Specifically, a scalar version of the *inner product* of the estimation error $e(n)$ and the tap input $u(n-k)$ is computed for $k = 0, 1, 2, \ldots, M-2, M-1$. The result so obtained defines the *correction* $\delta\hat{w}_k(n)$ applied to the tap weight $\hat{w}_k(n)$ at adaptation cycle $n+1$. The scaling factor used in this computation is denoted by a *positive* μ in Fig. 5.1(c) called the *step-size parameter,* which is real-valued.

Comparing the control mechanism of Fig. 5.1(c) for the LMS algorithm with that of Fig. 4.2 for the method of steepest descent, we see that the LMS algorithm uses the product $u(n-k)e^*(k)$ as an estimate of element k in the gradient vector $\nabla J(n)$ that characterizes the method of steepest descent. In other words, the expectation operator is removed from all the paths in Fig. 5.1(c). Accordingly, the recursive computation of each tap weight in the LMS algorithm suffers from *gradient noise*.

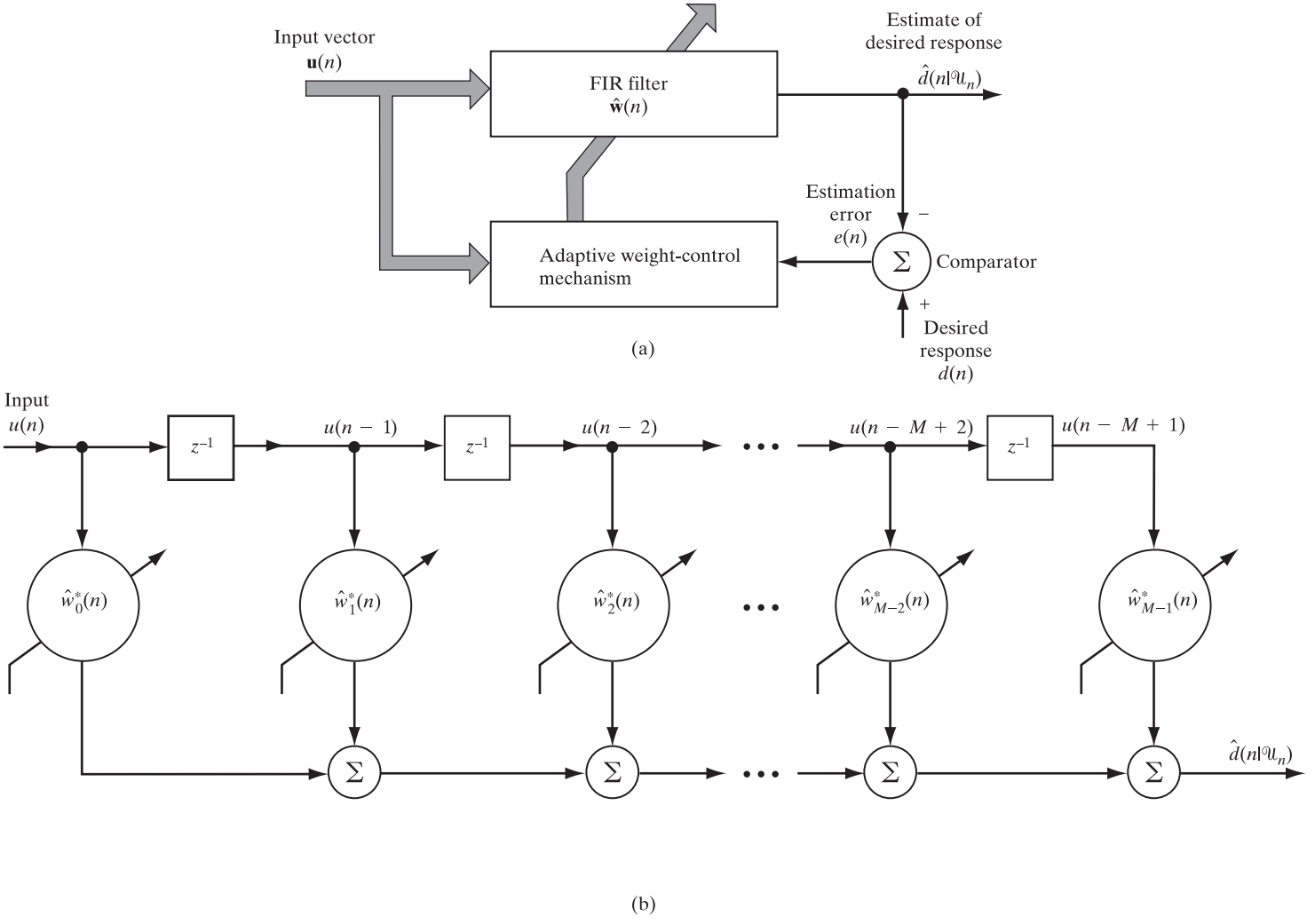


FIGURE 5.1 (*continued on the next page*)

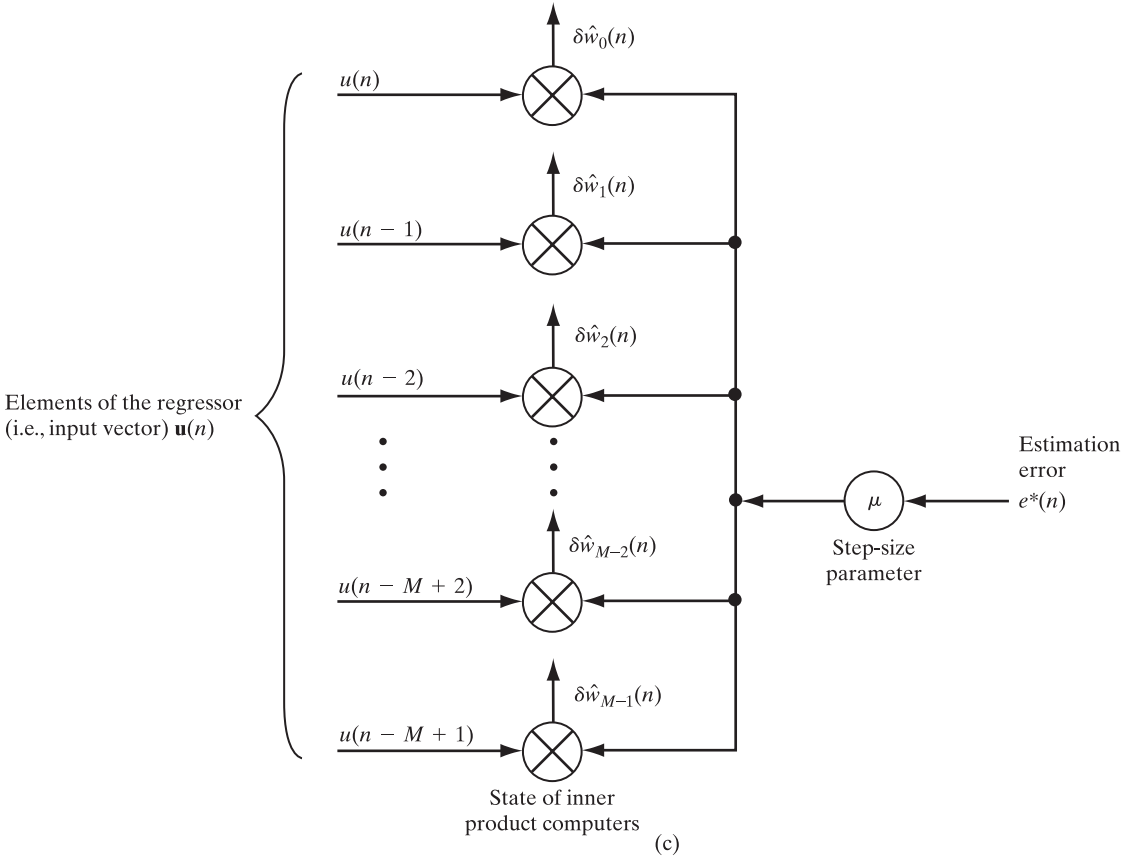


FIGURE 5.1 (a) Block diagram of adaptive FIR filter. (b) Detailed structure of the FIR filter component. (c) Detailed structure of the weight-control mechanism.

Throughout this chapter, it is assumed that the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$ are drawn from a jointly wide-sense stationary environment. In particular, the desired response $d(n)$ is *linearly* related to the input vector (i.e., regressor) $\mathbf{u}(n)$ by a multiple linear regression model whose parameter vector is unknown—hence the need for adaptive filtering. For such an environment, we know from Chapter 4 that the method of steepest descent computes a tap-weight vector $\mathbf{w}(n)$ that moves down the ensemble-average error-performance surface along a deterministic trajectory that terminates on the Wiener solution $\mathbf{w}_o$. The LMS algorithm, on the other hand, behaves differently because of the presence of gradient noise: Rather than terminating on the Wiener solution, the tap-weight vector $\hat{\mathbf{w}}(n)$ [different from $\mathbf{w}(n)$] computed by the LMS algorithm moves in the form of a *random walk* around the minimum point of the error-performance surface—that is, the Wiener solution.

In Eq. (2.3) of Chapter 2 on the Wiener filter, which is reproduced here for convenience, we defined the cost function as the mean-square value of the estimation error, as shown by

$$J = \mathbb{E}[|e(n)|^2], \tag{5.1}$$

where $\mathbb{E}$ is the statistical expectation operator. Therein, it was assumed that the Wiener filter operates on a wide-sense stationary environment, which results in a cost function, J, that is independent of time n, as shown in Eq. (5.1). To this end:

> *The expectation operator performs ensemble averaging over a large number of statistically independent realizations of the instantaneous value of the square estimation error, $|e(n)|^2$, which is performed at time n.*

Unfortunately, in practical applications of adaptive filtering, the use of ensemble averaging in the manner just described is not feasible. We say so because the whole motivation behind the deployment of adaptive filtering is to *adapt* to statistical variations of an unknown environment in an *on-line manner*, based on a single realization of the estimation error, $e(n)$, as it evolves across time n. Doing so is precisely what the method of stochastic gradient descent is all about. We may therefore proceed by ignoring the expectation operator in Eq. (5.1), thereby working on the following simplifed cross-function:

$$\begin{aligned} J_s(n) &= |e(n)|^2 \\ &= e(n)e^*(n), \end{aligned} \tag{5.2}$$

where the subscript s in $J_s(n)$ is intended to differentiate it from its ensemble-average counterpart, J. With the estimation error $e(n)$ being the *sample function* of a stochastic process, it follows that the cost function $J_s(n)$ is itself the sample value of a stochastic process. The derivative of $J_s(n)$ with respect to the kth tap weight of the FIR filter, $w_k(n)$, is therefore also stochastic, which is how it should be in the method of stochastic gradient descent.

Following the procedure described in Appendix B based on the Wirtinger calculus for computing gradients of complex data, the first use of which was discussed in

Chapter 2 on Wiener filters, we may express the partial derivative of the cost function $J_s(n)$ with respect to the complex conjugate of $w_k(n)$ as

$$\begin{aligned}\nabla J_{s,k}(n) &= \frac{\partial J_s(n)}{\partial w_k^*(n)} \\ &= -2u(n-k)e^*(n), \quad k = 0, 1, \ldots, M-1,\end{aligned} \tag{5.3}$$

which is exactly the same as that of Eq. (2.10) in Chapter 2, except for the expectation operator $\mathbb{E}$. Using the stochastic gradient of Eq. (5.3), we may now formulate the updating rule for the LMS algorithm as follows:

$$\hat{w}_k(n+1) = \hat{w}_k(n) - \frac{1}{2}\mu\nabla J_{s,k}(n), \tag{5.4}$$

where the scaling factor $\frac{1}{2}$ has been introduced merely for mathematical convenience. Thus, substituting Eq. (5.3) into Eq. (5.4), we obtain

$$\hat{w}_k(n+1) = \hat{w}_k(n) + \mu u(n-k)e^*(n), \quad k = 0, 1, \ldots, M-1, \tag{5.5}$$

where μ is a *fixed positive* step-size parameter.

Equation (5.5) is the scalar form of the LMS algorithm. To express it in vector form using matrix notation, let

$$\hat{\mathbf{w}}(n) = [\hat{w}_0(n), \hat{w}_1(n), \ldots, \hat{w}_{M-1}(n)]^{\mathrm{T}},$$

where the superscript T denotes *transposition*, and

$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M-1)]^{\mathrm{T}}.$$

We may thus rewrite Eq. (5.5) in the following compact form:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n), \tag{5.6}$$

where the asterisk denotes *complex conjugation*.

By definition, we have

$$e(n) = d(n) - \hat{d}(n\,|\,\mathcal{U}_n), \tag{5.7}$$

where

$$\hat{d}(n\,|\,\mathcal{U}_n) = \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n), \tag{5.8}$$

where the superscript H denotes *Hermitian transposition* (i.e., transposition combined with complex conjugation). Thus Eqs. (5.6) to (5.8), put together, define the LMS algorithm.

Another Way of Deriving the LMS Algorithm

The updated formula of Eq. (5.6) may also be obtained exactly from Eq. (4.10), describing iterative computation of the Wiener filter using the method of steepest descent that was covered in Chapter 4. Specifically, all that we have to do is to replace the correlation

matrix $\mathbf{R}$ and cross-correlation vector $\mathbf{p}$ with their respective instantaneous (sample) values; that is:

- The matrix $\mathbf{R}$ in Eq. (2.29) is replaced by the outer product $\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)$.
- The vector $\mathbf{p}$ in Eq. (2.32) is replaced by the product $\mathbf{u}(n)d^*(n)$.

Then, doing simple algebraic manipulations, we get the updated formula of Eq. (5.6). There is an important point that should be carefully noted here:

> Whatever connection that may have existed between the LMS algorithm and the Wiener filter, that connection is completely destroyed once the expectation operator $\mathbb{E}$ is removed in deriving the LMS algorithm from the Wiener filter.

How Do We Account for the Filtering Capability of the LMS Algorithm?

At first sight, it may appear that because instantaneous estimates of the correlation matrix $\mathbf{R}$ and cross-correlation vector $\mathbf{p}$ have relatively large variances, the LMS algorithm is incapable of performing a satisfactory filtering function. However, we must recognize that the LMS algorithm is *recursive*, as evidenced by Eq. (5.5) or, equivalently, Eq. (5.6). Accordingly, the LMS algorithm has a built-in *feedback mechanism*, whereby the instantaneous estimates of both $\mathbf{R}$ and $\mathbf{p}$ are *averaged over time* during the course of adaptation. Thus, although the trajectory followed by the LMS algorithm is entirely different from that produced by recursive computation of the Wiener filter, the LMS algorithm can produce a satisfactory performance under the following proviso:

> The step-size parameter, μ, is assigned a relatively small positive value.

TABLE 5.1 Summary of the LMS Algorithm

Parameters: M = number of taps (i.e., filter length)
μ = step-size parameter

$$0 < \mu < \frac{2}{\lambda_{\max}},$$

where $\lambda_{\max}$ is the maximum value of the correlation matrix of the tap inputs $u(n)$ and the filter length M is moderate to large.

Initialization: If prior knowledge of the tap-weight vector $\hat{\mathbf{w}}(n)$ is available, use it to select an appropriate value for $\hat{\mathbf{w}}(0)$. Otherwise, set $\hat{\mathbf{w}}(0) = \mathbf{0}$.

Data:

- Given $\mathbf{u}(n) = M$-by-1 tap-input vector at time n
  $= [u((n), u(n\text{-}1)), \ldots, u(n - M + 1)]^{\mathrm{T}}$
  $d(n)$ = desired response at time n.
- To be computed:
  $\hat{\mathbf{w}}(n + 1) =$ estimate of tap-weight vector at time $n + 1$.

Computation: For $n = 0, 1, 2, \ldots$, compute

$$e(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n)$$
$$\hat{\mathbf{w}}(n + 1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n).$$

It turns out that this provision is also responsible for robust behavior of the LMS algorithm, an issue that will be addressed in Chapter 11.

Summary of the LMS Algorithm

In Table 5.1 we present a summary of the LMS algorithm, which incorporates Eqs. (5.6) through (5.8) as well as the initialization of the algorithm. The table also includes a constraint on the permissible value of the step-size parameter, which is needed to ensure that the algorithm converges. More is said on this necessary condition for convergence of the LMS algorithm in Chapter 6.

5.3 APPLICATION 2: GRADIENT-ADAPTIVE LATTICE FILTERING ALGORITHM

Another algorithm that is rooted in the method of stochastic gradient descent is the gradient-adaptive lattice (GAL) algorithm. The GAL algorithm, due to Griffiths (1977, 1978), distinguishes itself from the LMS algorithm in that it is a *joint esimator*. To explain what we mean by this new terminology, consider the block diagram of Fig. 5.2, which is based on a *multistage lattice predictor*; this structure is a reproduction of Fig. 3.13. Unlike the LMS algorithm that is driven by the input signal directly, the adjustable tap weights in the GAL algorithm are driven by the backward prediction errors produced by the multistage lattice predictor. Accordingly, there are two consecutive stages in the GAL algorithm:

- Stage 1 involves recursive computation of the backward prediction errors, leading to recursive updates of the reflection coefficients.
- Stage 2 involves recursive updates of the tap weights in the GAL algorithm, followed by recursive estimates of the desired response.

Computations in these two stages are carried out jointly—hence the terminology "joint estimator."

Multistage Lattice Predictor

Figure 5.3 shows the block diagram of a single-stage lattice predictor, the input–output relation of which is characterized by a single parameter: the *reflection coefficient* κ_m. We assume that the input data are wide-sense stationary and that κ_m is complex valued. For the estimation of κ_m, we start with the cost function

$$J_{fb,\,m} = \frac{1}{2}\,\mathbb{E}[\,|f_m(n)|^2 + |b_m(n)|^2], \tag{5.9}$$

where $f_m(n)$ is the forward prediction error and $b_m(n)$ is the backward prediction error, both measured at the output of the lattice predictor; $\mathbb{E}$ is the statistical expectation operator; and the factor $\frac{1}{2}$ is introduced to simplify the presentation. From Section 3.8 we recall that the input–output relations of the lattice stage under consideration are described by (see Fig. 5.3)

$$f_m(n) = f_{m-1}(n) + \kappa_m^* b_{m-1}(n-1) \tag{5.10}$$

and

$$b_m(n) = b_{m-1}(n-1) + \kappa_m f_{m-1}(n). \tag{5.11}$$

Substituting Eqs. (5.10) and (5.11) into Eq. (5.9), we get

$$J_{fb,m} = \frac{1}{2}(\mathbb{E}[|f_{m-1}(n)|^2] + \mathbb{E}[|b_{m-1}(n-1)|^2])(1 + |\kappa_m|^2) + \kappa_m \mathbb{E}[f_{m-1}(n) b^*_{m-1}(n-1)] + \kappa^*_m \mathbb{E}[b_{m-1}(n-1) f^*_{m-1}(n)]. \tag{5.12}$$

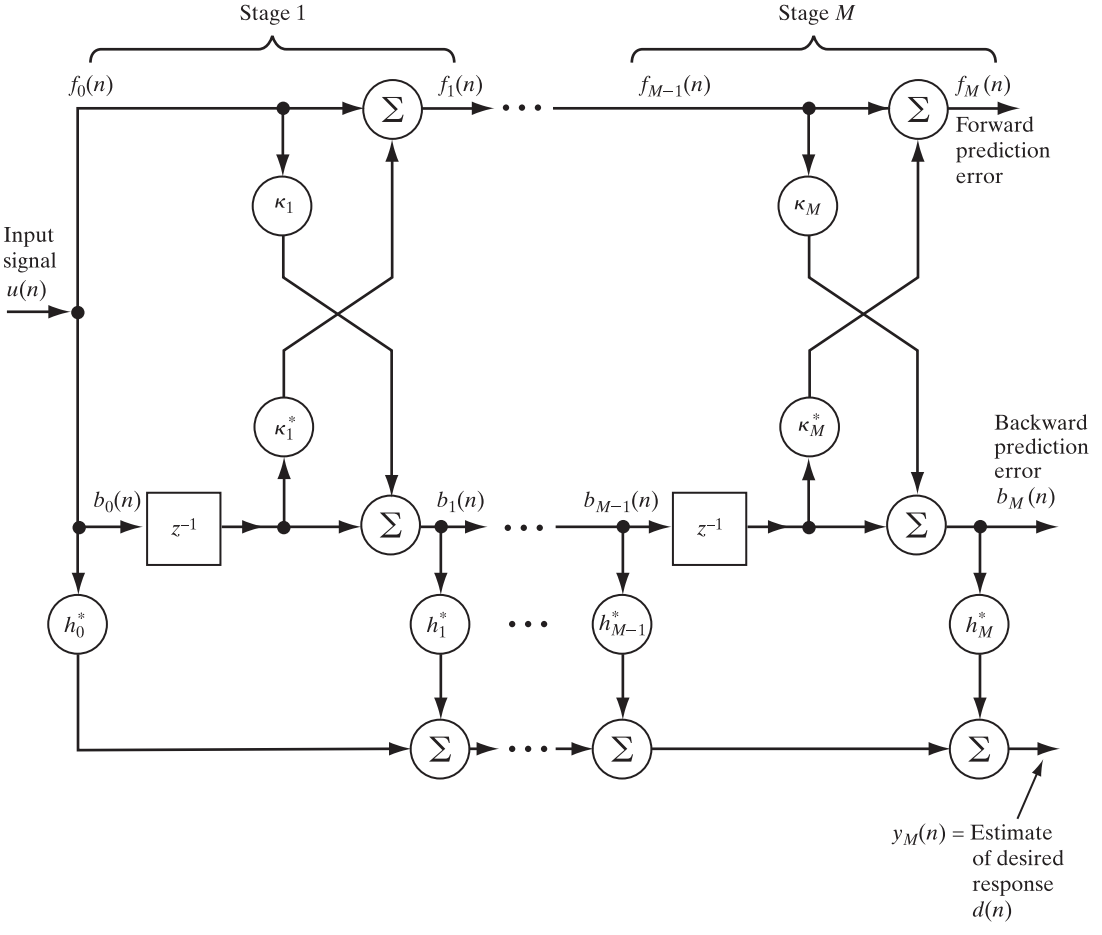


FIGURE 5.2 Lattice-based structure for joint-process estimation, with each lattice stage involving a single reflection coefficient.

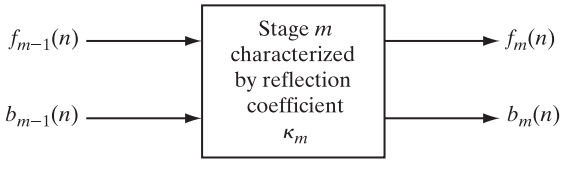


FIGURE 5.3 Block diagram of stage m in a lattice predictor; the relations defining the dependence of the output prediction variables $f_m(n)$ and $b_m(n)$ on the input prediction variables $f_{m-1}(n)$ and $b_{m-1}(n)$ are given in Eqs. (5.10) and (5.11).

Differentiating the cost function $J_{fb,\,m}$ with respect to the complex-valued reflection coefficient κ_m, we get

$$\frac{\partial J_{fb,\,m}}{\partial \kappa_m} = \kappa_m(\mathbb{E}[|f_{m-1}(n)|^2] + \mathbb{E}[|b_{m-1}(n-1)|^2]) + 2\mathbb{E}[b_{m-1}(n-1)f^*_{m-1}(n)]. \quad (5.13)$$

Putting this gradient equal to zero, we find that the *optimum* value of the reflection coefficient, for which the cost function $J_{fb,\,m}$ is minimum, is given by

$$\kappa_{m,\,o} = -\frac{2\mathbb{E}[b_{m-1}(n-1)f^*_{m-1}(n)]}{\mathbb{E}[|f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2]}. \quad (5.14)$$

Equation (5.14) for the optimum reflection coefficient $\kappa_{m,\,o}$ is known as the *Burg formula*[2] (Burg, 1968).

The Burg formula involves the use of ensemble averages. Assuming that the input signal $u(n)$ is ergodic, then, as discussed in Chapter 1, we may substitute time averages for the expectations in the numerator and denominator of the formula. We thus get the *Burg estimate* for the reflection coefficient $\kappa_{m,o}$ for stage m in the lattice predictor:

$$\hat{\kappa}_m(n) = -\frac{2\sum_{i=1}^{n} b_{m-1}(i-1)f^*_{m-1}(i)}{\sum_{i=1}^{n}(|f_{m-1}(i)^2| + |b_{m-1}(i-1)|^2)}. \quad (5.15)$$

Here, the dependence of the estimate $\hat{\kappa}_m$ on the data-block length n emphasizes the fact that the estimate is *data dependent*.

The GAL Algorithm

The formula of Eq. (5.15) is a *block estimator* for the reflection coefficient κ_m; it operates on the input prediction variables $f_{m-1}(i)$ and $b_{m-1}(i-1)$ for stage m of the lattice predictor on a block-by-block basis. We may reformulate this estimator into an equivalent *recursive* structure by following the procedure described next.

First, we define

$$\mathscr{E}_{m-1}(n) = \sum_{i=1}^{n}(|f_{m-1}(i)|^2 + |b_{m-1}(i-1)|^2), \quad (5.16)$$

which is the *total energy* of both the forward and delayed backward prediction errors at the input of the mth stage, measured up to and including time n. Isolating the term $|f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2$ from the rest of the summation in Eq. (5.16), we have a recursive formula for computing $\mathscr{E}_{m-1}(n)$, that is the summation in the denominator of Eq. (5.15):

$$\begin{aligned}\mathscr{E}_{m-1}(n) &= \sum_{i=1}^{n-1}(|f_{m-1}(i)|^2 + |b_{m-1}(i-1)|^2) + |f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2 \\ &= \mathscr{E}_{m-1}(n-1) + |f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2. \end{aligned} \quad (5.17)$$

[2]The 1968 paper by Burg is reproduced in the book edited by Childers (1978).

In a similar fashion, we may formulate the following recursive formula for computing the summation in the numerator of Eq. (5.15), which represents a time-average cross-correlation:

$$\sum_{i=1}^{n} b_{m-1}(i-1) f_{m-1}^{*}(i) = \sum_{i=1}^{n-1} b_{m-1}(i-1) f_{m-1}^{*}(i) + b_{m-1}(n-1) f_{m-1}^{*}(n). \quad (5.18)$$

Accordingly, substituting Eqs. (5.17) and (5.18) into Eq. (5.15), we may reformulate the Burg estimate for the reflection coefficient κ_m as

$$\hat{\kappa}_m(n) = -\frac{2\sum_{i=1}^{n-1} b_{m-1}(i-1) f_{m-1}^{*}(i) + 2b_{m-1}(n-1) f_{m-1}^{*}(n)}{\mathscr{E}_{m-1}(n-1) + |f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2}. \quad (5.19)$$

Equation (5.19) is still not quite in the right form for recursively computing the estimate $\hat{\kappa}_m(n)$. To arrive at such a recursive formula, we continue as follows:

1. Use the time-varying estimate $\hat{\kappa}_m(n-1)$ in place of κ_m to rewrite Eqs. (5.10) and (5.11) respectively, as

$$f_m(n) = f_{m-1}(n) + \hat{\kappa}_m^{*}(n-1) b_{m-1}(n-1) \quad (5.20)$$

and

$$b_m(n) = b_{m-1}(n-1) + \hat{\kappa}_m(n-1) f_{m-1}(n) \quad (5.21)$$

for $m = 1, 2, \ldots, M$. For practical justification as to why $\hat{\kappa}_m(n-1)$ rather than $\hat{\kappa}_m(n)$ is the correct choice in Eqs. (5.20) and (5.21), the reader is referred to Problem 8.

2. Use the rearranged forms of Eqs. (5.20) and (5.21) together with Eq. (5.17), to write

$$\begin{aligned}
2b_{m-1}(n-1) f_{m-1}^{*}(n) &= b_{m-1}(n-1) f_{m-1}^{*}(n) + f_{m-1}^{*}(n) b_{m-1}(n-1) \\
&= b_{m-1}(n-1)(f_m(n) - \hat{\kappa}_m^{*}(n-1) b_{m-1}(n-1))^{*} \\
&\quad + f_{m-1}^{*}(n)(b_m(n) - \hat{\kappa}_m(n-1) f_{m-1}(n)) \\
&= -\hat{\kappa}_m(n-1)(|f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2) \\
&\quad + (f_{m-1}^{*}(n) b_m(n) + b_{m-1}(n-1) f_m^{*}(n)) \\
&= -\hat{\kappa}_m(n-1)(\mathscr{E}_{m-1}(n-1) + |f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2) \\
&\quad + \hat{\kappa}_m(n-1)\mathscr{E}_{m-1}(n-1) + (f_{m-1}^{*}(n) b_m(n) + b_{m-1}(n-1) f_m^{*}(n)) \\
&= -\hat{\kappa}_m(n-1)\mathscr{E}_{m-1}(n) + \hat{\kappa}_m(n-1)\mathscr{E}_{m-1}(n-1) \\
&\quad + (f_{m-1}^{*}(n) b_m(n) + b_{m-1}(n-1) f_m^{*}(n)).
\end{aligned}$$

3. Use the relation just derived and the formula of Eq. (5.15) redefined for $\hat{\kappa}_m(n-1)$ to rewrite the numerator of Eq. (5.19) as

$$\begin{aligned}
&2\sum_{i=1}^{n-1} b_{m-1}(i-1)f^*_{m-1}(i) + 2b_{m-1}(n-1)f^*_{m-1}(n) \\
&= -\hat{\kappa}_m(n-1)\mathscr{E}_{m-1}(n-1) - \hat{\kappa}_m(n-1)\mathscr{E}_{m-1}(n) + \hat{\kappa}_m(n-1)\mathscr{E}_{m-1}(n-1) \\
&\quad + (f^*_{m-1}(n)b_m(n) + b_{m-1}(n-1)f^*_m(n)) \\
&= -\hat{\kappa}_m(n-1)\mathscr{E}_{m-1}(n) + (f^*_{m-1}(n)b_m(n) + b_{m-1}(n-1)f^*_m(n)).
\end{aligned}$$

Hence, substituting this last relation into the numerator of Eq. (5.19) and then simplifying terms, we get the recursive formula

$$\hat{\kappa}_m(n) = \hat{\kappa}_m(n-1) - \frac{f^*_{m-1}(n)b_m(n) + b_{m-1}(n-1)f^*_m(n)}{\mathscr{E}_{m-1}(n)}, \qquad m = 1, 2, \ldots, M. \tag{5.22}$$

To finalize the algorithmic formulation of the gradient lattice filter, we make two modifications to the recursive formulas of Eqs. (5.17) and (5.22) (Griffiths, 1977, 1978):

1. Introduce a *step-size parameter* $\tilde{\mu}$ to control the *adjustment* applied to each reflection coefficient in progressing from one adaptation cycle to the next:

$$\hat{\kappa}_m(n) = \hat{\kappa}_m(n-1) - \frac{\tilde{\mu}}{\mathscr{E}_{m-1}(n)}(f^*_{m-1}(n)b_m(n) + b_{m-1}(n-1)f^*_m(n)), \quad m = 1, 2, \ldots, M. \tag{5.23}$$

2. Modify the energy estimator of Eq. (5.17) by writing it in the form of a single-pole averaging filter that operates on the squared prediction errors, as shown by the convex combination

$$\mathscr{E}_{m-1}(n) = \beta\mathscr{E}_{m-1}(n-1) + (1-\beta)(|f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2), \tag{5.24}$$

where β is a new parameter lying in the interval $0 < \beta < 1$.

The original assumption behind the derivation of the recursive estimator of Eq. (5.22) is that it operates in a pseudostationary environment. In order to deal with statistical variations in a *nonstationary* environment, the modification described in Eq. (5.24) is introduced. The purpose of the modification is to equip the estimator with *memory* whereby the immediate past value of the prediction energy $\mathscr{E}_{m-1}$, as well as its present value, is used in computing the present value of the estimate $\hat{\kappa}_m(n)$.

Desired-Response Estimator

Turning next to the estimation of the desired response $d(n)$, we consider the structure shown in Fig. 5.4, which is that part of Fig. 5.2 that deals with this estimation for m stages, where $m = 0, 1, \ldots, M$. Here, we have an input vector (i.e., regressor) $\mathbf{b}_m(n)$ consisting of the backward prediction errors $b_0(n), b_1(n), \ldots, b_m(n)$ and a corresponding vector $\hat{\mathbf{h}}_m(n)$ made up of the parameters $\hat{h}_0, \hat{h}_1, \ldots, \hat{h}_m$. The regressor acts on the parameter vector to produce the output signal $y_m(n)$, which is an estimate of $d(n)$.

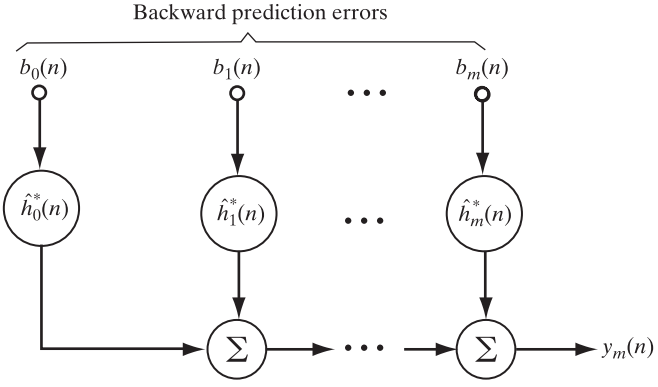


FIGURE 5.4 Desired-response estimator using a sequence of m backward prediction errors.

For the estimation of $\hat{\mathbf{h}}$, we may use a stochastic-gradient approach and proceed along the lines of the normalized LMS algorithm to be discussed in Section 7.1 of Chapter 7. First, referring to Fig. 5.4, we readily see that the order-update estimate of the desired response $d(n)$ is defined by

$$\begin{aligned} y_m(n) &= \sum_{k=0}^{m} \hat{h}_k^*(n) b_k(n) \\ &= \sum_{k=0}^{m-1} \hat{h}_k^*(n) b_k(n) + \hat{h}_m^*(n) b_m(n) \\ &= y_{m-1}(n) + \hat{h}_m^*(n) b_m(n). \end{aligned} \tag{5.25}$$

Correspondingly, the estimation error is defined by

$$e_m(n) = d(n) - y_m(n). \tag{5.26}$$

Following Problem 7, we may go on to express the time update for the mth regression coefficient as

$$\hat{h}_m(n+1) = \hat{h}_m(n) + \frac{\tilde{\mu}}{\|\mathbf{b}_m(n)\|^2} b_m(n) e_m^*(n), \tag{5.27}$$

where $\tilde{\mu}$ is the step-size parameter and the squared Euclidean norm $\|\mathbf{b}_m(n)\|^2$ is defined by the order update

$$\begin{aligned} \|\mathbf{b}_m(n)\|^2 &= \sum_{k=0}^{m} |b_k(n)|^2 \\ &= \sum_{k=0}^{m-1} |b_k(n)|^2 + |b_m(n)|^2 \\ &= \|\mathbf{b}_{m-1}(n)\|^2 + |b_m(n)|^2. \end{aligned} \tag{5.28}$$

Properties of the GAL Algorithm

The use of the time-varying step-size parameter $\mu_m(n) = \tilde{\mu}/\mathscr{E}_{m-1}(n)$ in the update equation for the reflection coefficient $\hat{\kappa}_m(n)$ introduces a form of *normalization* similar to that in the normalized LMS algorithm, discussed in Chapter 7. From Eq. (5.24), we see that, for small magnitudes of the forward and backward prediction errors, the value of the parameter $\mathscr{E}_{m-1}(n)$ is correspondingly small, or, equivalently, the step-size parameter $\tilde{\mu}(n)$ has a correspondingly large value. Such a behavior is desirable from a practical point of view. Basically, a small value for the prediction errors means that the adaptive lattice predictor is providing an accurate model of the external environment in which it is operating. Hence, if there is any increase in the prediction errors, it should be due to variations in that environment, in which case it is highly desirable for the adaptive lattice predictor to respond rapidly to such variations. This objective is indeed realized by having $\tilde{\mu}(n)$ assume a large value, which makes it possible for Eq. (5.23) in the GAL algorithm to provide an initially rapid convergence to the new environmental conditions. If, on the other hand, the input data applied to the adaptive lattice predictor are too noisy (i.e., if they contain a strong white-noise component in addition to the signal of interest), then the prediction errors produced by the adaptive lattice predictor are correspondingly large. In such a situation, the parameter $\mathscr{E}_{m-1}(n)$ has a large value, or, equivalently, the step-size parameter $\tilde{\mu}(n)$ has a small value. Accordingly, Eq. (5.23) in the GAL algorithm does *not* respond rapidly to variations in the external environment, which is precisely the way we would like the algorithm to behave (Alexander, 1986a).

Previously, we referred to the GAL algorithm as approximate in nature. We justify this remark on the following grounds:

- The order-update equations (5.20) and (5.21) for the forward and backward prediction errors, respectively, were postulated under the assumption that the signal $u(n)$ applied to the input of the multistage lattice predictor originates from a *stationary* source. To deal with a nonstationary environment, the energy estimator is modified into the form shown in Eq. (5.24), which accounts for statistical variations in the environment.
- Most importantly, there is no guarantee that, in a nonstationary environment, the decoupling property (i.e., orthogonality of the backward prediction errors) of the multistage lattice predictor in Fig. 5.2 is preserved by using the GAL algorithm.

The net result is that, although the convergence behavior of the GAL algorithm is faster than that of the LMS algorithm, it is inferior to exact recursive least-squares lattice algorithms to be discussed in Chapter 16. Nevertheless, the GAL algorithm offers a relatively simple implementation, which makes it attractive from a practical perspective.

Summary of the GAL Algorithm

Equations (5.20) and (5.21) are the order updates and Eqs. (5.23) and (5.24) are the time updates, of the multistage lattice predictor of the GAL algorithm. Equations (5.25) and (5.28) are the order updates and Eq. (5.27) is the time update of the algorithm's desired-response estimator, as summarized in Table 5.2, which also shows how to initialize the GAL algorithm.

For a well-behaved convergence of the GAL algorithm, it is recommended that we set $\tilde{\mu} < 0.1$.

TABLE 5.2 Summary of the GAL Algorithm

Parameters:
M = final prediction order
β = constant, lying in the range $(0, 1)$
$\tilde{\mu} < 0.1$
δ: small positive constant
a: another small positive constant

Multistage lattice predictor:

For prediction order $m = 1, 2, \ldots, M$, put

$$f_m(0) = b_m(0) = 0$$
$$\mathscr{E}_{m-1}(0) = a$$
$$\hat{\kappa}_m(0) = 0.$$

For time step $n = 1, 2, \ldots$, put

$$f_0(n) = b_0(n) = u(n), \qquad u(n) = \text{lattice predictor input.}$$

For prediction order $m = 1, 2, \ldots, M$ and time step $n = 1, 2, \ldots$, compute

$$\mathscr{E}_{m-1}(n) = \beta\mathscr{E}_{m-1}(n-1) + (1-\beta)(|f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2)$$
$$f_m(n) = f_{m-1}(n) + \hat{\kappa}_m^*(n-1)b_{m-1}(n-1)$$
$$b_m(n) = b_{m-1}(n-1) + \hat{\kappa}_m(n-1)f_{m-1}(n)$$
$$\hat{\kappa}_m(n) = \hat{\kappa}_m(n-1) - \frac{\tilde{\mu}}{\mathscr{E}_{m-1}(n)}[f_{m-1}^*(n)b_m(n) + b_{m-1}(n-1)f_m^*(n)].$$

Desired response estimator:

For prediction order $m = 0, 1, \ldots, M$, put

$$\hat{h}_m(0) = 0.$$

For time step $n = 0, 1, \ldots$, put

$$y_{-1}(n) = 0; \|\mathbf{b}_{-1}(n)\|^2 = \delta.$$

For prediction order $m = 0, 1, \ldots, M$ and time step $n = 0, 1, \ldots$, compute

$$y_m(n) = y_{m-1}(n) + \hat{h}_m^*(n)b_m(n)$$
$$e_m(n) = d(n) - y_m(n)$$
$$\|\mathbf{b}_m(n)\|^2 = \|\mathbf{b}_{m-1}(n)\|^2 + |b_m(n)|^2$$
$$\hat{h}_m(n+1) = \hat{h}_m(n) + \frac{\tilde{\mu}}{\|\mathbf{b}_m(n)\|^2}b_m(n)e_m^*(n).$$

5.4 OTHER APPLICATIONS OF STOCHASTIC GRADIENT DESCENT

Over and above Problems 5 and 6 on the leaky LMS and fourth-least-mean (FLM) algorithm, respectively, there are other applications of the method of stochastic gradient descent that are covered in subsequent chapters of the book.

In Chapter 7, we discuss the normalized LMS algorithm, which is intended to mitigate a limitation of the LMS algorithm that is attributed to the amplification of gradient noise. In effect, the step-size parameter μ is normalized with respect to the square Euclidean norm of the input vector, $\mathbf{u}(n)$, plus a small fixed parameter; in other words, the step-size parameter takes a time-varying normalized form.

As discussed in Section 5.1, one of the issues in using the method of stochastic gradient descent is that of having to track statistical variations encountered in a nonstationary environment. Although the LMS algorithm does have a built-in capability to

tackle this tracking problem, it is not adequate when dealing with the processing of data streams that, for example, are massive in size and change rapidly in time.

In Chapter 13, we describe a new stochastic gradient descent algorithm called the *incremental delta-bar-delta* (IDBD) *algorithm*. In this algorithm, the step-size parameter is not only time-varying but also takes the form of a vector whose individual elements correlate with factors of the input vector.

All the stochastic gradient descent algorithms discussed up to and including Chapter 11 are of the supervised learning variety. In Chapter 17, we shift gears by addressing blind adaptation that does not require supervision (i.e., desired response). With blind equalization as the application of interest, we describe the so-called *Bussgang algorithms*, the implementation of which involves the combined use of a zero-memory nonlinear estimator and the LMS algorithm.

5.5 SUMMARY AND DISCUSSION

In this chapter, we discussed the principles of the method of stochastic gradient descent, which provides the basis for a family of adaptive filtering algorithms exemplified by the LMS and GAL algorithms.

A distinctive feature of this first family of adaptive filtering algorithms is a computational complexity that is linear in the number of adjustable parameters (e.g., tap weights) of a finite-duration impulse response (FIR) filter, around which the adaptation takes place. Computational simplicity is particularly important when processing data streams that are massive in size and change rapidly in time. In applications of this kind, time is of the essence, hence the requirement to minimize the number of computations needed for each adaptation cycle.

Computational complexity governed by a linear law is an important characteristic of the LMS algorithm. Furthermore, the LMS algorithm has another important characteristic, namely robustness, which signifies the ability to deliver a satisfactory performance in the face of unknown disturbance that can arise in practice. A cautionary note is in order here: Robustness is always attained at the expense of efficiency. This trade-off is discussed in detail in Chapter 11.

Another requirement that needs to be addressed in adaptive filtering is that of tracking statistical variations in a nonstationary environment over time. Referring back to the example of having to process massive data that change rapidly, we may have to improve the tracking capability of the LMS algorithm. One clever way of accomplishing this requirement is to do two things:

- First, use a vectorized step-size parameter, the dimensionality of which is the same as that of the tap-weight vector to be estimated.
- Second, expand the LMS algorithm to assume the form of a "learning within learning" scheme.

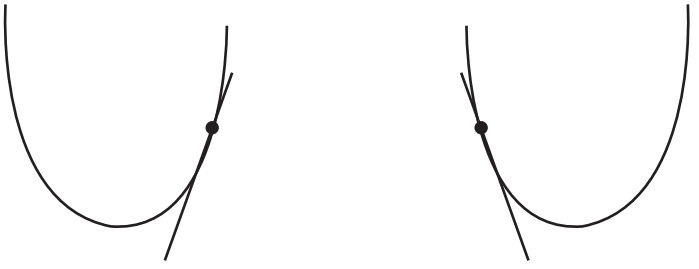
In such a scheme, each element of the tap-weight vector is assigned its own step-size parameter, thereby acquiring the capability to focus more on relevant parts of the incoming data and less on the irrelevant parts. The idea of learning within learning based on the LMS algorithm strengthens the signal-processing power to tackle applications that are beyond the capability of the algorithm in its traditional form. Most importantly,

as will be shown in Chapter 13, the modified algorithm remains a stochastic gradient descent algorithm.

To conclude, the method of stochastic gradient descent has a great deal to offer in the design of adaptive filtering algorithms.

PROBLEMS

1. Parts (a) and (b) of Fig. P5.1 depict two ways in which the gradient of a convex function can arise. In part (a), the gradient is positive; in part (b), it is negative. Demonstrate that despite this difference, algorithmic description of the method of gradient descent is exactly the same for both parts.



(a) (b)

FIGURE P5.1

2. How can we study the performance of an adaptive filtering algorithm experimentally? Give two applications of the method of stochastic gradient descent to linear adaptive filtering.

3. The update formula of the real-valued version of the so-called *sign-error LMS algorithm* is given by

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \mathbf{u}(n-i)\text{sgn}[e(n)], \quad i = 0,1,\ldots,M-1.$$

(a) Compare the computational complexity of this algorithm to that of the LMS algorithm.
(b) How is this update formula modified for the complex-valued version of the sign-error LMS?

4. Using the steepest descent method, derive the LMS algorithm by using iterative computation of the Wiener filter.

5. *The leaky LMS algorithm.* Consider the time-varying cost function

$$J(n) = |e(n)|^2 + \alpha \|\mathbf{w}(n)\|^2,$$

where $\mathbf{w}(n)$ is the tap-weight vector of an FIR filter, $e(n)$ is the estimation error, and α is a constant. As usual,

$$e(n) = d(n) - \mathbf{w}^{\text{H}}(n)\mathbf{u}(n),$$

where $d(n)$ is the desired response and $\mathbf{u}(n)$ is the tap-input vector. In the leaky LMS algorithm, the cost function $J(n)$ is minimized with respect to the weight vector $\mathbf{w}(n)$.

(a) Show that the time update for the tap-weight vector $\hat{\mathbf{w}}(n)$ is defined by

$$\hat{\mathbf{w}}(n+1) = (1 - \mu\alpha)\hat{\mathbf{w}}(n) + \mu \mathbf{u}(n)e^*(n).$$

(b) How would you modify the tap-input vector in the conventional LMS algorithm to get the equivalent result described in part (a) of the problem?

6. The instantaneous cost function of the *fourth-least-mean (FLM) algorithm* is defined by

$$J_{\mathbf{s}}(w) = |e(n)|^4.$$

(a) Show that the update formula for this algorithm is given by

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \mathbf{u}(n-i)e^*(n)|e(n)|^2 \quad i = 0, 1, \ldots, M-1.$$

(b) Compare the computational complexity of the FLM algorithm to that of the LMS algorithm.

(c) What is the advantage of the FLM over the LMS algorithm?

7. Compare the computational complexity of the FLM algorithm to that of the GAL algorithm.

8. As an alternative to the procedure used in Section 5.3 to derive the recursion of Eq. (5.22) for the GAL algorithm, it would be tempting to proceed as follows: First, $\hat{\kappa}_m(n)$ is used in place of $\hat{\kappa}_m(n-1)$ in Eqs. (5.20) and (5.21), as shown by

$$f_m(n) = f_{m-1}(n) + \hat{\kappa}_m^*(n) b_{m-1}(n-1);$$
$$b_m(n) = b_{m-1}(n-1) + \hat{\kappa}_m(n) f_{m-1}(n).$$

Next, through a process of cross-multiplication of terms applied to Eq. (5.15), followed by simplification of terms, the recursion

$$\hat{\kappa}_m(n) = \hat{\kappa}_m(n-1) - \frac{f_{m-1}^*(n) b_m(n) + b_{m-1}(n-1) f_m^*(n)}{\mathscr{E}_{m-1}(n-1)}, \qquad m = 1, 2, \ldots, M,$$

for updating the reflection coefficient is derived. The term $\mathscr{E}_{m-1}(n-1)$ is defined in Eq. (5.17) with n replaced by $n-1$.

(a) Derive this alternative recursion for $\hat{\kappa}_m(n)$.

(b) Given the equations presented in this problem, explain the reason why such an algorithm is computationally impractical, thereby justifying the practical validity of Eq. (5.22).

9. Compare the computational complexity of the LMS algorithm with that of the GAL algorithm for the same number of adjustable parameters.

CHAPTER 6

The Least-Mean-Square (LMS) Algorithm

In this relatively long chapter, we build on the derivation of the least-mean-square (LMS) algorithm as an application of the method of stochastic gradient descent that was presented in the preceding chapter. Specifically, we will expand on why the LMS algorithm is of fundamental importance in linear adaptive filtering in theory as well as application.

We begin the chapter with a signal-flow graph representation of the LMS algorithm, which clearly exhibits the fact that the LMS algorithm is basically a *nonlinear feedback control system*. With feedback known to be a "double-edged sword," it can work for us or against us. It is not surprising, therefore, to find that with the control mechanism being directly dependent on how the step-size parameter, μ, is chosen. This parameter plays a critical role in assuring convergence of the algorithm (i.e., its stability when viewed as a feedback control system) or in failing to do so.

The study of convergence is carried out under the umbrella of *statistical learning theory* of the LMS algorithm, which occupies a good part of the chapter. Although, indeed, the LMS algorithm is simple to formulate, its mathematical analysis is very difficult to carry out. Nevertheless, ideas related to *efficiency* of the algorithm that are derived from this theory are supported through the use of Monte Carlo simulations.

6.1 SIGNAL-FLOW GRAPH

For convenience of presentation, we reproduce the LMS algorithm summarized in Table 5.1, as follows:

$$y(n) = \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n), \tag{6.1}$$

$$e(n) = d(n) - y(n), \tag{6.2}$$

and

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^{*}(n), \tag{6.3}$$

where $\mathbf{u}(n)$ is the input vector (regressor), $d(n)$ is the corresponding desired response, and $\hat{\mathbf{w}}(n)$ is an estimate of the unknown tap-weight vector, $\mathbf{w}(n)$, of the linear multiple regression model used to represent the environment from which $\mathbf{u}(n)$ and $d(n)$ are jointly picked. The superscript H denotes *Hermitian transposition* (i.e., transposition combined with complex conjugation), and the asterisk denotes *complex conjugation*.

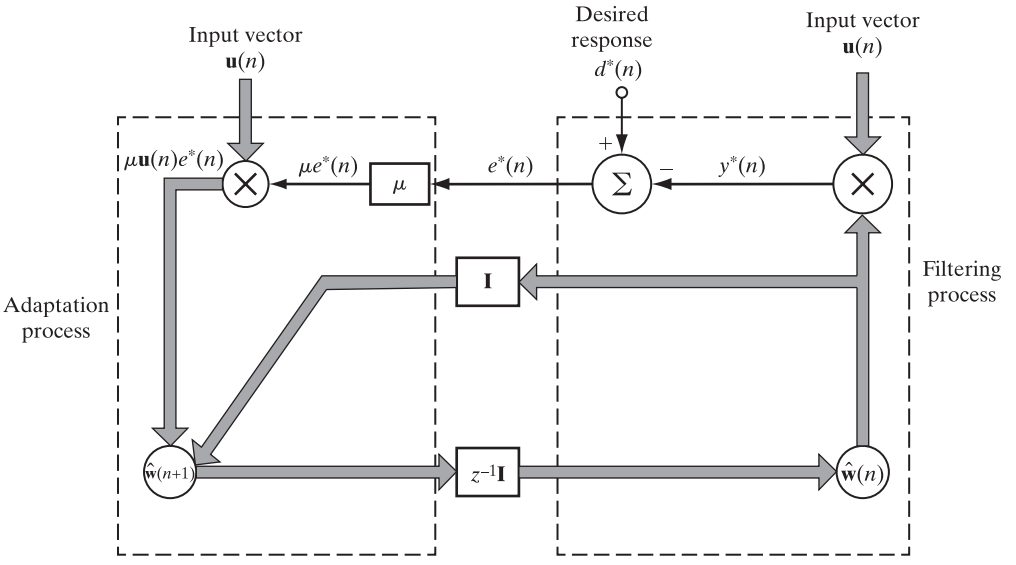


FIGURE 6.1 Signal-flow graph representation of the LMS algorithm, where $\mathbf{I}$ is the identity matrix and z^{-1} is the unit-time delay operator.

Given this set of equations, we may construct a *signal-flow graph* of the LMS algorithm as shown in Fig. 6.1. Based on this diagram, we see that the LMS algorithm embodies two basic processes that continually build on each other:

1. *Filtering process*. This first process involves two operations:
   - one computing the output, $y^*(n)$, of the finite-duration impulse response (FIR) filter within the algorithm, in response to the input signal $\mathbf{u}(n)$, and
   - the other one generating the estimation error, $e^*(n)$, by subtracting $y^*(n)$ from $d^*(n)$.
2. *Adaptation process.* This second process involves updating the present value of the estimated weight vector $\hat{\mathbf{w}}(n)$ by an "incremental" amount equal to the product term $\mu\mathbf{u}(n)e^*(n)$ to produce $\hat{\mathbf{w}}(n+1)$, where the incrementality is assured by choosing a small value for the step-size parameter, μ.

These two processes are identified on the right- and left-hand sides of Fig. 6.1, respectively.[1] Henceforth, each complete cycle of the LMS algorithm is referred to as an *adaptation cycle*.

From Fig. 6.1, we readily find that:

Computational complexity of the LMS algorithm scales linearly with dimensionality of the estimate $\hat{\mathbf{w}}(n)$.

This statement reemphasizes simplicity of the LMS algorithm that was pointed out in Chapter 5.

[1]The multiplier on the left in Fig. 6.1 signifies ordinary multiplication, and the multiplier on the right signifies inner-product formulation.

6.2 OPTIMALITY CONSIDERATIONS

There are two ways in which optimality of the LMS algorithm may be considered. The first involves data-induced perturbation, for which the LMS algorithm is optimal in a localized sense. The second is an overall sense, for which, as expected, the LMS algorithm is suboptimal.

Localized Optimality

In Chapter 5, we pointed out that the method of stochastic gradient descent is of a "local" kind, meaning that the search for a gradient vector is chosen not only randomly but also in a localized manner. Despite its stochastic behavior, the LMS algorithm is capable of exhibiting localized optimality in a Euclidean sense, provided that the *data-induced perturbation* from one adaptation cycle to the next one is small enough (Sayed, 2003). In a way, this requirement works in the best interests of the algorithm as an adaptive filter.

To explain what we have in mind here, we consider two different scenarios:

1. *Current estimation error.* Given the supervised-training set $\{\mathbf{u}(n), d(n)\}$, we write
$$e(n) = d(n) - \mathbf{w}^{\mathrm{H}}(n)\mathbf{u}(n), \tag{6.4}$$
where $\mathbf{w}(n)$ is an unknown tap-weight vector to be optimized in a localized Euclidean sense.
2. *Posterior estimation error.* In this second scenario, the requirement is to optimize the updated tap-weight vector $\mathbf{w}(n+1)$, for which we write
$$r(n) = d(n) - \mathbf{w}^{\mathrm{H}}(n+1)\mathbf{u}(n). \tag{6.5}$$
Here, it should be noted that the error $r(n)$ is different from the one-step prediction error that involves the product term $\mathbf{w}^{\mathrm{H}}(n)\mathbf{u}(n+1)$.

For the localized perturbation to be small, we require that the step-size parameter, μ, is small enough to satisfy the following condition:

$$|1 - \mu\|\mathbf{u}(n)\|^2| < 1 \quad \text{for all } \mathbf{u}(n),$$

where $\|\mathbf{u}(n)\|^2$ is the squared Euclidean norm of the input vector $\mathbf{u}(n)$. Equivalently, we may express this condition as follows:

$$0 < \frac{1}{2}\mu\|\mathbf{u}(n)\|^2 < 1 \quad \text{for all } \mathbf{u}(n). \tag{6.6}$$

We may now state the problem at hand:

> Find the optimum value of $\mathbf{w}(n+1)$ that minimizes the squared Euclidean between it and its past value $\mathbf{w}(n)$, subject to the condition described in Eq. (6.6).

In effect, the localized problem of interest is a *constrained optimization problem*.

To solve this problem, we use the well-known *method of Lagrange multipliers*, described in Appendix C. Specifically, we begin by introducing the *Lagrangian function*:

$$L(n) = \frac{1}{2}\|\mathbf{w}(n+1) - \mathbf{w}(n)\|^2 + \lambda(n)\left(r(n) - \left(1 - \frac{\mu}{2}\|\mathbf{u}(n)\|^2\right)e(n)\right), \qquad (6.7)$$

where $\lambda(n)$ is a *time-dependent Lagrangian multiplier*; $e(n)$ and $r(n)$ are defined in Eqs. (6.4) and (6.5). Applying Wirtinger calculus of Appendix B to differentiate $L(n)$ with respect to $\mathbf{w}^{\mathrm{H}}(n+1)$ and formally treating $\mathbf{w}(n)$ as a constant, we obtain the partial derivative

$$\frac{\partial L(n)}{\partial \mathbf{w}^{\mathrm{H}}(n+1)} = \mathbf{w}(n+1) - \mathbf{w}(n) - \lambda(n)\mathbf{u}(n).$$

Let $\hat{\mathbf{w}}(n+1)$ and $\hat{\mathbf{w}}(n)$ be the tap-weight vectors for which this partial derivative is zero. Accordingly, we obtain

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \lambda(n)\mathbf{u}(n). \qquad (6.8)$$

To find the corresponding value of the Lagrange multiplier, we substitute the solution of Eq. (6.8) in Eq. (6.7). Then, cancelling common terms in the resulting equation, we obtain

$$\lambda(n) = \mu e^*(n).$$

for which the Lagrangian function $L(n)$ attains its minimum value of zero. Recognizing that the parameter μ is a positive constant, it follows that the Lagrangian multiplier is complex valued for all n.

Finally, using the $\lambda(n)$ in Eq. (6.8), we get the desired solution to our constrained optimization problem, as shown by

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n),$$

which is a duplicate of the update formula previously described in Eq. (6.3), and with it, localized optimality of the LMS algorithm is justified.

Suboptimality Compared to the Wiener Solution

In Chapter 4, we demonstrated that the steepest-descent algorithmic representation of the Wiener filter approaches the *Wiener solution*, $\mathbf{w}_o$, in the limit as the number of adaptation cycles, n, approaches infinity, provided that the step-size parameter, μ, satisfies the condition of Eq. (4.22). Moreover, the learning curve approaches the asymptotic value $J_{\min}$.

In Section 6.4, we will show that, in contrast with the Wiener filter, the corresponding learning curve of the LMS algorithm approaches an asymptotic value $J(\infty)$ that exceeds $J_{\min}$ by an amount referred to as the *excess mean-square error*. It follows, therefore, that compared to the Wiener solution, the LMS algorithm is *suboptimal*.

In conceptual statistical terms, we may differentiate between these two learning curves as follows:

1. In the method of steepest descent, ensemble averaging is performed *before* computing the learning curve, as illustrated in Fig. 6.2(a); thereby, *perfect information* about the environment is contained in the correlation matrix $\mathbf{R}$ and cross-correlation vector $\mathbf{p}$, provided that the environment is wide-sense stationary. The learning curve for the method of steepest descent is therefore *deterministic* in nature.

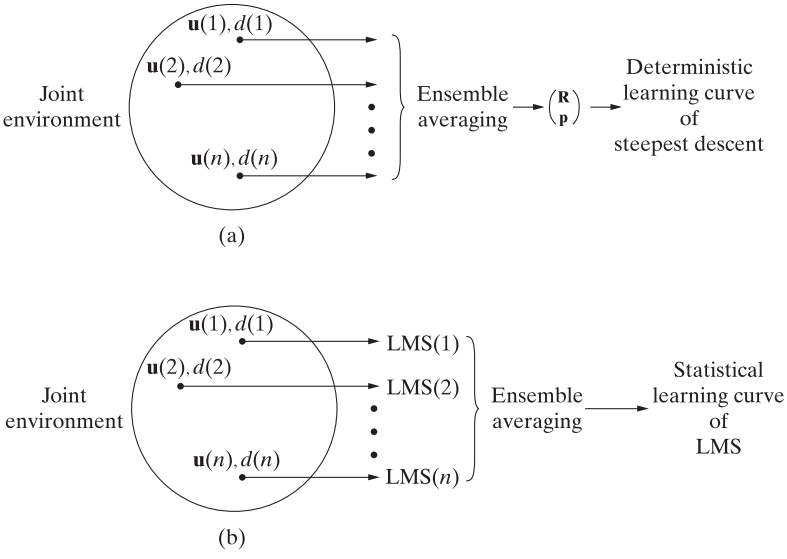


FIGURE 6.2 Illustrating the difference between learning curves for the same environment, assumed to be stationary: (a) method of steepest descent; (b) LMS.

2. On the other hand, in the LMS case, ensemble averaging is performed *after* computing the "noisy" learning curves of an ensemble of independently adapted FIR filters, as illustrated in Fig. 6.2(b); the noise is attributed to *gradient noise*. The learning curve of the LMS is therefore *statistical* in nature.

This important difference in ensemble averaging is what accounts for the excess mean-square error experienced in the LMS algorithm and therefore its suboptimality compared to the Wiener solution.

6.3 APPLICATIONS

Before proceeding further with a convergence analysis of the LMS algorithm, it is instructive to develop an appreciation for the versatility of this important signal-processing algorithm. We do that by presenting six widely different applications of the LMS algorithm.

Application 1: Canonical Model of the Complex LMS Algorithm

The LMS algorithm described in Eqs. (6.1) through (6.3) is *complex* in the sense that the input and output data as well as the tap weights are all complex valued. To emphasize the complex nature of the algorithm, henceforth we use the following complex notations:

Tap-input vector:

$$\mathbf{u}(n) = \mathbf{u}_I(n) + j\mathbf{u}_Q(n). \tag{6.9}$$

Desired response:

$$d(n) = d_I(n) + \mathrm{j}d_Q(n). \tag{6.10}$$

Tap-weight vector:

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}_I(n) + \mathrm{j}\hat{\mathbf{w}}_Q(n). \tag{6.11}$$

FIR filter output:

$$y(n) = y_I(n) + \mathrm{j}y_Q(n). \tag{6.12}$$

Estimation error:

$$e(n) = e_I + \mathrm{j}e_Q(n). \tag{6.13}$$

The subscripts I and Q denote "in-phase" and "quadrature" components, respectively—that is, real and imaginary parts, respectively. Using these definitions in Eqs. (6.1) through (6.3), expanding terms, and then equating real and imaginary parts, we get

$$y_I(n) = \hat{\mathbf{w}}_I^{\mathrm{T}}(n)\mathbf{u}_I(n) - \hat{\mathbf{w}}_Q^{\mathrm{T}}(n)\mathbf{u}_Q(n), \tag{6.14}$$

$$y_Q(n) = \hat{\mathbf{w}}_I^{\mathrm{T}}(n)\mathbf{u}_Q(n) + \hat{\mathbf{w}}_Q^{\mathrm{T}}(n)\mathbf{u}_I(n), \tag{6.15}$$

$$e_I(n) = d_I(n) - y_I(n), \tag{6.16}$$

$$e_Q(n) = d_Q(n) - y_Q(n), \tag{6.17}$$

$$\hat{\mathbf{w}}_I(n+1) = \hat{\mathbf{w}}_I(n) + \mu[e_I(n)\mathbf{u}_I(n) - e_Q(n)\mathbf{u}_Q(n)], \tag{6.18}$$

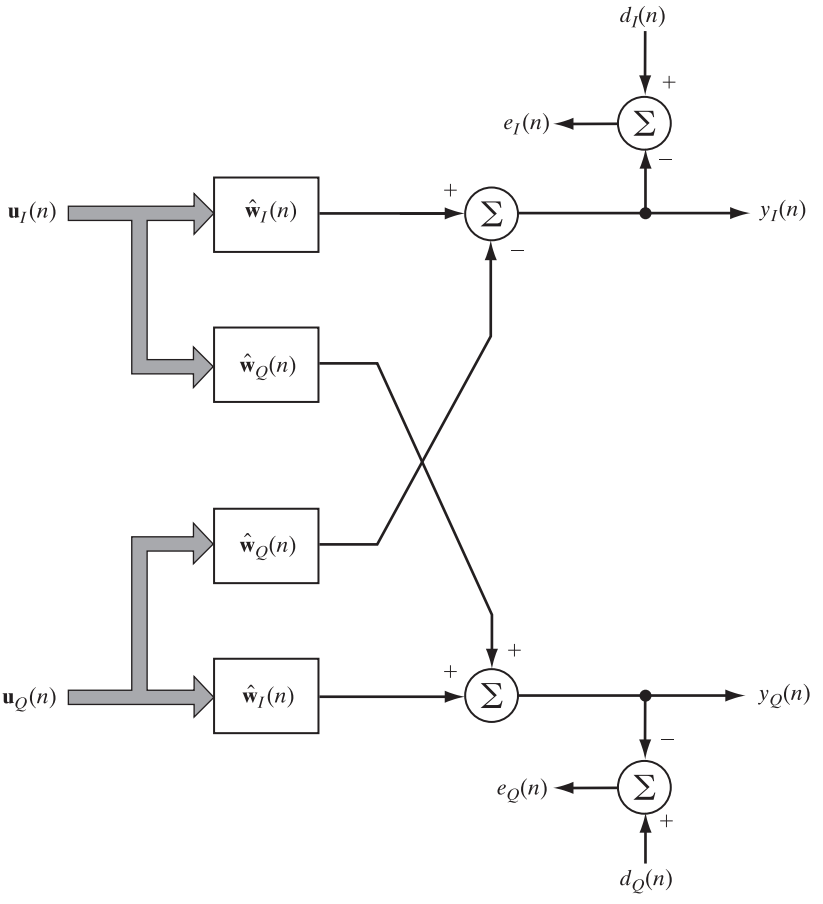
and

$$\hat{\mathbf{w}}_Q(n+1) = \hat{\mathbf{w}}_Q(n) + \mu[e_I(n)\mathbf{u}_Q(n) + e_Q(n)\mathbf{u}_I(n)], \tag{6.19}$$

where the superscript T denotes transposition. Equations (6.14) through (6.17), defining the error and output signals, are represented by the cross-coupled signal-flow graph shown in Fig. 6.3(a). The updated Eqs. (6.18) and (6.19) are likewise represented by the cross-coupled signal-flow graph shown in Fig. 6.3(b). The combination of this pair of signal-flow graphs constitutes the *canonical model* of the complex LMS algorithm. This model clearly illustrates that a complex LMS algorithm is equivalent to a set of four real LMS algorithms with *cross-coupling* between them, hence the computing power of the complex LMS algorithm.

The need for the canonical model of the complex LMS algorithm may arise, for example, in the *adaptive equalization* of a communication system for the transmission of binary data over a dispersive channel. To facilitate data transmission over the channel, some form of modulation is used, so that the spectral content of the transmitted signal resides inside the passband of the channel. Moreover, for spectral efficiency, modulation techniques such as quadrature phase-shift keying (QPSK) or M-ary quadrature amplitude modulation (QAM) are used, in which case the baseband version of the channel output assumes a complex form, which is the reason for the complex LMS algorithm. In any event, data transmission through the channel is limited by two factors:

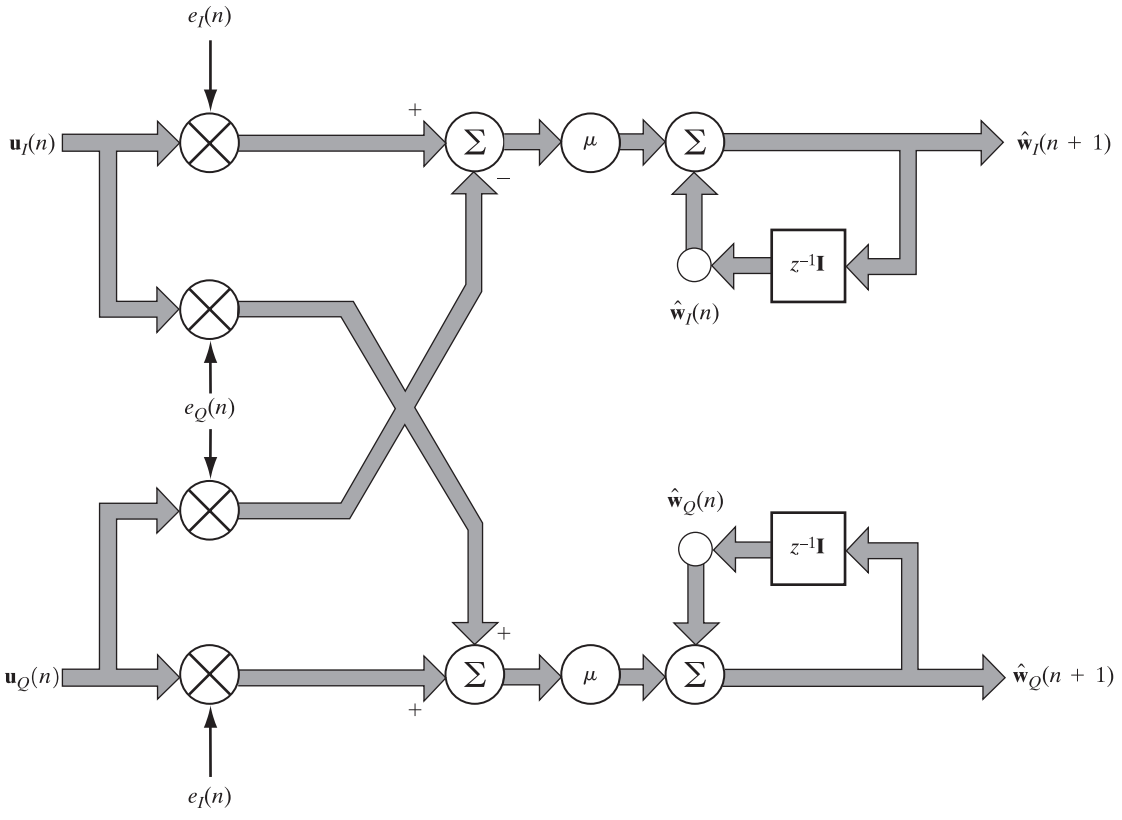
- *Intersymbol interference* (ISI), which is caused mainly by dispersion in the channel.
- *Thermal noise*, which is generated at the channel output (i.e., receiver input).



(a)

FIGURE 6.3 Canonical signal-flow graph representation of the complex LMS algorithm. (a) Error and output signals. *(continued on the next page.)*

For bandwidth-limited channels, we typically find that ISI is the chief determining factor in the design of high-data transmission systems. The adaptive equalizer is customarily placed in the receiver. With the channel output as the source of excitation applied to the equalizer, its free parameters are adjusted by means of the complex LMS algorithm to provide an estimate of each symbol transmitted. Provision for the desired response is made locally in the receiver. Specifically, during the *training mode*, a *replica* of the desired response is *stored* in the receiver. Naturally, the generator of this stored reference has to be synchronized with the known training sequence that is transmitted prior to data transmission. A widely used training sequence is a *pseudonoise (PN) sequence*, which has broadband noiselike properties; in reality, it is a deterministic waveform that repeats periodically. The PN sequence is generated by means of a feedback shift register that consists of a number of consecutive two-state memory stages (flip-flops) regulated



(b)

FIGURE 6.3 (b) Tap-weight update equation.

by a single timing clock. A feedback signal consisting of the modulo-two sum of the outputs of the memory stages is applied to the first memory stage of the shift register and thereby prevents it from emptying. Once the training mode is completed, data transmission over the channel begins. To allow the adaptive equalizer to track statistical variations of the channel during data transmission, the equalizer is switched to its decision-directed mode, more on which is said in Chapter 17.

Application 2: Adaptive Deconvolution for Processing of Time-Varying Seismic Data

In *exploration seismology*, we usually think of a layered model of the earth. In order to collect (i.e., record) seismic data for the purpose of characterizing such a model and thereby unraveling the complexities of the earth's surface, it is customary to use the *method of reflection seismology*, which involves the following:

- A source of seismic energy (e.g., dynamite or an air gun), which is typically activated on the earth's surface.

- The propagation of seismic waves from the interfaces between the earth's geological layers.
- Picking up and recording the seismic returns (i.e., reflections of seismic waves from the interfaces), which carry information about the subsurface structure of the earth.

An important issue in exploration seismology is the interpretation of seismic returns from the different geological layers of the earth. This interpretation is fundamental to the identification of crusted regions such as rocks, layers of sand, or sedimentary layers. The sedimentary layers are of particular interest, because they may contain hydrocarbon reservoirs. The idea of a layered-earth model plays a key role here.

Figure 6.4 depicts *an FIR model* for a layered earth; the model provides a local parameterization of the propagation (scattering) phenomenon in the earth's subsurface. According to the real-valued model shown in the figure, the input (downgoing) seismic wave $s(n)$ and the output (upgoing) seismic wave $u(n)$ are related by the convolution sum (assuming a model of length $M-1$)

$$u(n) = \sum_{k=0}^{M-1} w_k s(n-k),$$

where the sequence of tap weights, denoted by $w_0, w_1, \ldots, w_{M-1}$, represents the spatial mapping of the weighting or impulse response of the medium. The problem to be solved is one of system identification:

> Given a seismogram (i.e., a record of seismic returns) denoted by $u(n)$, estimate the impulse response w_n of the medium.

This estimation, called *seismic deconvolution*, removes the effect of convolving $s(n)$ with w_n. However, the problem is complicated by the fact that the input seismic

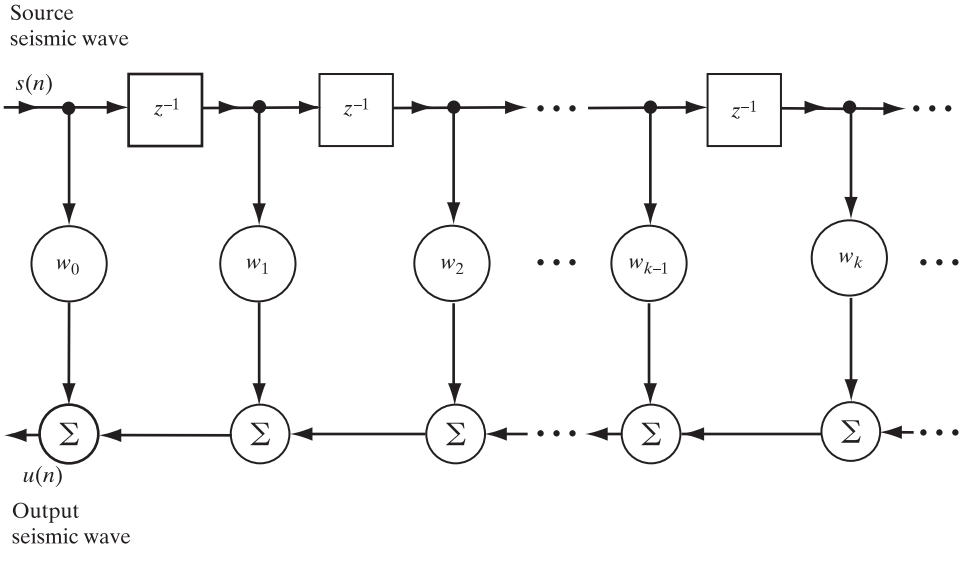


FIGURE 6.4 FIR model of layered earth.

wave $s(n)$ is actually unknown. To overcome this difficulty, we may use one of two procedures:

1. *Predictive deconvolution*,[2] which is so called because the procedure relies on linear prediction theory.
2. *Blind deconvolution*, which accounts for valuable phase information contained in the reflection seismogram—information that is ignored in predictive deconvolution.

In this section, we describe predictive convolution; discussion of blind deconvolution is deferred to Chapter 17. In both procedures, the LMS algorithm features prominently.

Predictive deconvolution rests on two hypotheses (Robinson & Durrani, 1986):

1. The *feedback hypothesis*, which treats w_n as the impulse response of an autoregressive (AR) model; the implication of this hypothesis is that the layered-earth model is minimum phase.
2. The *random hypothesis*, according to which the reflectivity function (i.e., the result of the deconvolution) is assumed to have the properties of white noise, at least within certain time gates.

Given the real-valued seismogram $u(n)$, we may use the real LMS algorithm to solve the predictive deconvolution problem by proceeding as follows (Griffiths et al., 1977):

- An M-dimensional operator $\hat{\mathbf{w}}(n)$ is used to generate a predicted trace from the data; that is,

$$u(n + \Delta) = \hat{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n), \tag{6.20}$$

  where

$$\hat{\mathbf{w}}(n) = [\hat{w}_0(n), \hat{w}_1(n), \ldots, \hat{w}_{M-1}(n)]^{\mathrm{T}},$$
$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^{\mathrm{T}},$$

  and $\Delta \geq 1$ is the *prediction depth*, or *decorrelation delay*, measured in units of the sampling period.
- The deconvolved trace $y(n)$ defining the difference between the input and predicted samples is evaluated:

$$y(n) = u(n) - \hat{u}(n).$$

- The operator $\hat{\mathbf{w}}(n)$ is updated:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu[u(n+\Delta) - \hat{u}(n+\Delta)]\mathbf{u}(n). \tag{6.21}$$

Equations (6.20) and (6.21) constitute the LMS-based adaptive seismic deconvolution algorithm. The adaptation is begun with an initial guess $\hat{\mathbf{w}}(0)$.

[2] A critique of predictive deconvolution is given by Schneider (1978, p. 29):

[A] work horse of statistical wavelet deconvolution for [several] decades has been the predictive deconvolution approach, which assumes the reflectivity function is statistically white and the convolutional wavelet to be minimum phase. To say that this has not been an effective tool is to condemn hundreds of thousands of miles of seismic processing and to deny untold millions of barrels of oil discovered from these data.

Application 3: Instantaneous Frequency Measurement

In this example, we study the use of the LMS algorithm as the basis for estimating the frequency content of a *narrowband signal* characterized by a rapidly varying power spectrum (Griffiths, 1975). In so doing, we illustrate the linkage between three basic ideas: an autoregressive (AR) model for describing a stochastic process, studied in Chapter 1; a linear predictor for analyzing the process, studied in Chapter 3; and the LMS algorithm for estimating the AR parameters.

By a narrowband signal, we mean a signal whose bandwidth Ω is small compared with the midband angular frequency ω_c, as illustrated in Fig. 6.5. A *frequency-modulated (FM) signal* is an example of a narrowband signal, provided that the carrier frequency is high enough. The *instantaneous frequency* (defined as the derivative of phase with respect to time) of an FM signal varies linearly with the modulating signal. Consider, then, a narrowband process $u(n)$ generated by a *time-varying* AR model of order M, as shown by the difference equation (assuming *real* data)

$$u(n) = -\sum_{k=1}^{M} a_k(n)u(n-k) + \nu(n), \tag{6.22}$$

where the $a_k(n)$ are the time-varying model parameters and $\nu(n)$ is a zero-mean white-noise process of time-varying variance $\sigma_\nu^2(n)$. The *time-varying AR (power) spectrum* of the narrowband process $u(n)$ is given by [see Eq. (3.101)]

$$S_{\mathrm{AR}}(\omega; n) = \frac{\sigma_\nu^2(n)}{\left|1 + \sum_{k=1}^{M} a_k(n)e^{-\mathrm{j}\omega k}\right|^2}, \qquad -\pi < \omega \leq \pi. \tag{6.23}$$

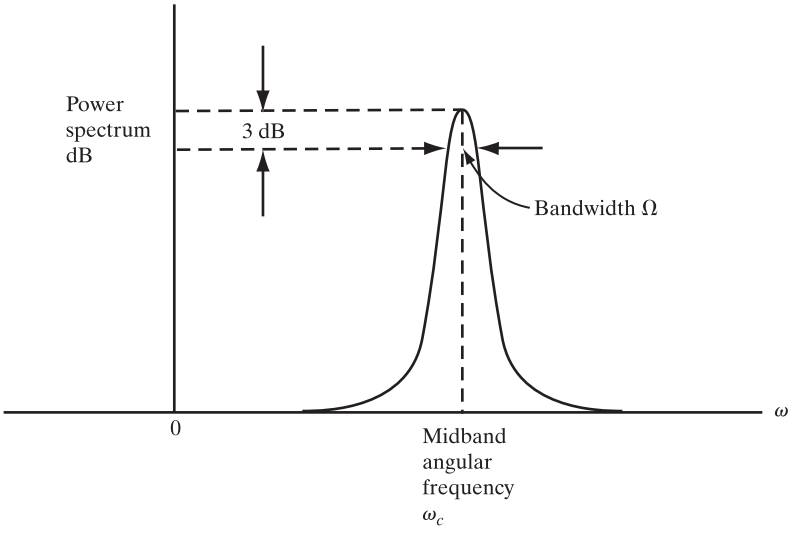


FIGURE 6.5 Definition of a narrowband signal in terms of its spectrum.

Note that an AR process whose poles form a cluster near the unit circle in the z-plane has the characteristics of a narrowband process.

To estimate the model parameters, we use an adaptive FIR filter employed as a linear predictor of order M. Let the tap weights of the predictor be denoted by $\hat{w}_k(n)$, $k = 1, 2, \ldots, M$. The tap weights are adapted continuously as the input signal $u(n)$ is received. In particular, we use the following LMS algorithm for adapting the tap weights:

$$\hat{w}_k(n+1) = \hat{w}_k(n) + \mu u(n-k) f_M(n), \qquad k = 1, 2, \ldots, M. \tag{6.24}$$

In this equation,

$$f_M(n) = u(n) - \sum_{k=1}^{M} \hat{w}_k(n) u(n-k) \tag{6.25}$$

is the *prediction error*. The tap weights of the adaptive predictor are related to the AR model parameters as follows:

$$-\hat{w}_k(n) = \text{estimate of } a_k(n) \text{ at adaptation cycle } n, \qquad \text{for } k = 1, 2, \ldots, M.$$

Moreover, the average power of the prediction error $f_M(n)$ provides an estimate of the noise variance $\sigma_\nu^2(n)$. Our interest is in locating the frequency of a narrowband signal. Accordingly, in what follows, we ignore the estimation of $\sigma_\nu^2(n)$. Specifically, we use only the tap weights of the adaptive predictor to define the *time-varying frequency function*

$$F(\omega; n) = \frac{1}{\left| 1 - \sum_{k=1}^{M} \hat{w}_k(n) e^{-j\omega k} \right|^2}. \tag{6.26}$$

Given the relationship between $\hat{w}_k(n)$ and $a_k(n)$, we see that the essential difference between the frequency function $F(\omega; n)$ in Eq. (6.26) and the AR power spectrum $S_{\text{AR}}(\omega; n)$ in Eq. (6.23) lies in their numerator scale factors. The numerator of $F(\omega; n)$ is a constant equal to unity, whereas that of $S_{\text{AR}}(\omega; n)$ is a time-varying constant equal to $\sigma_\nu^2(n)$. The advantages of $F(\omega; n)$ over $S_{\text{AR}}(\omega; n)$ are twofold: First, the 0/0 indeterminacy inherent in the narrowband spectrum of Eq. (6.23) is replaced by a "computationally tractable" limit of 1/0 in Eq. (6.26); second, the frequency function $F(\omega; n)$ is not affected by amplitude scale changes in the input signal $u(n)$, with the result that the peak value of $F(\omega; n)$ is related directly to the spectral width of the input signal.

We may use the function $F(\omega; n)$ to measure the instantaneous frequency of a frequency-modulated signal $u(n)$, provided that the following assumptions are justified (Griffiths, 1975):

- The adaptive predictor has been in operation sufficiently long, so as to ensure that any transients caused by the initialization of the tap weights have died out.
- The step-size parameter μ is chosen correctly for the prediction error $f_M(n)$ to be small for all n.
- The modulating signal is essentially constant over the sampling range of the adaptive predictor, which extends from adaptation cycle $(n - M)$ to adaptation cycle $(n - 1)$.

Given the validity of these assumptions, we find that the frequency function $F(\omega; n)$ has a peak at the instantaneous frequency of the input signal $u(n)$. Moreover, the LMS algorithm will track the time variation of the instantaneous frequency.

Application 4: Adaptive Noise Cancelling Applied to a Sinusoidal Interference

The traditional method of suppressing a *sinusoidal interference* corrupting an information-bearing signal is to use a fixed *notch filter* tuned to the frequency of the interference. To design the filter, we naturally need to know the precise frequency of the interference. But what if the notch is required to be very sharp and the interfering sinusoid is known to *drift* slowly? Clearly, then, we have a problem that calls for an *adaptive* solution. One such solution is provided by the use of adaptive noise cancelling, an application that is different from the previous three in that it is not based on a stochastic excitation.

Figure 6.6 shows the block diagram of a dual-input *adaptive noise canceller*. The primary input supplies an information-bearing signal and a sinusoidal interference that are uncorrelated with each other. The reference input supplies a correlated version of the sinusoidal interference. For the adaptive filter, we may use an FIR filter whose tap weights are adapted by means of the LMS algorithm. The filter uses the reference input to provide (at its output) an estimate of the sinusoidal interfering signal contained in the primary input. Thus, by subtracting the adaptive filter output from the primary input, the effect of the sinusoidal interference is diminished. In particular, an adaptive noise canceller using the LMS algorithm has two important characteristics (Widrow et al., 1976; Glover, 1977):

1. The canceller behaves as an adaptive notch filter whose null point is determined by the angular frequency ω_0 of the sinusoidal interference. Hence, the canceller is tunable, and the tuning frequency moves with ω_0.
2. The notch in the frequency response of the canceller can be made very sharp at precisely the frequency of the sinusoidal interference by choosing a small enough value for the step-size parameter μ.

Thus, unlike the situation with an ordinary notch filter, we have control over the frequency response of the adaptive noise canceller.

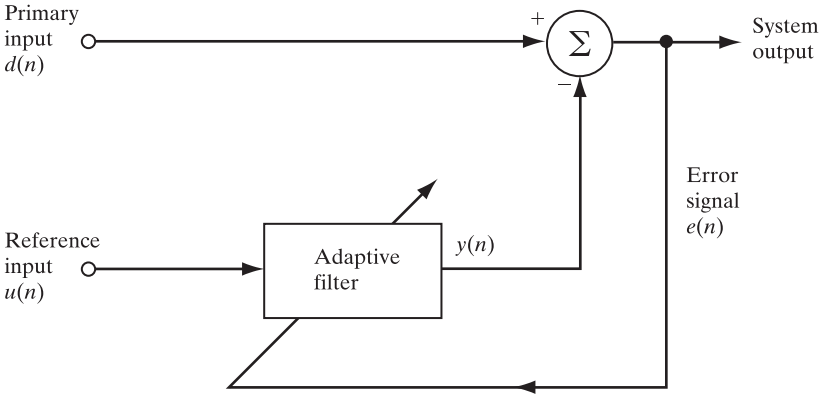


FIGURE 6.6 Block diagram of adaptive noise canceller.

In the application considered here, the input data are assumed to be real valued:

- *Primary input:*

$$d(n) = s(n) + A_0 \cos(\omega_0 n + \phi_0), \tag{6.27}$$

where $s(n)$ is an information-bearing signal, A_0 is the amplitude of the sinusoidal interference, ω_0 is the *normalized* angular frequency, and ϕ_0 is the phase.

- *Reference input:*

$$u(n) = A \cos(\omega_0 n + \phi), \tag{6.28}$$

where the amplitude A and the phase ϕ are different from those in the primary input, but the angular frequency ω_0 is the same.

Using the real form of the LMS algorithm, we describe the tap-weight update by means of the equations

$$y(n) = \sum_{i=0}^{M-1} \hat{w}_i(n) u(n - i), \tag{6.29}$$

$$e(n) = d(n) - y(n), \tag{6.30}$$

and

$$\hat{w}_i(n + 1) = \hat{w}_i(n) + \mu u(n - i) e(n), \qquad i = 0, 1, \ldots, M - 1, \tag{6.31}$$

where M is the length of the FIR filter and the constant μ is the step-size parameter. Note that the sampling period in the input data and in all other signals in the LMS algorithm is assumed to be unity for convenience of presentation; as mentioned previously, this practice is indeed followed throughout the book.

With a sinusoidal excitation as the input of interest, we restructure the block diagram of the adaptive noise canceller as in Fig. 6.7(a). According to this new representation, we may lump the sinusoidal input $u(n)$, the FIR filter, and the weight-update equation of the LMS algorithm into a single (open-loop) system. The adaptive system with input $e(n)$ and output $y(n)$ *varies with time* and cannot be represented by a transfer function. We may get around this difficulty as follows: With $z = e^{j\omega}$ and $z_0 = e^{j\omega_0}$, let the adaptive system be excited with $e(n) = z^n$. Then, the output $y(n)$ consists of three components: one proportional to z^n, the second proportional to $z^n(z_0^{2n})^*$, and the third proportional to $z^n(z_0^{2n})$. The first component represents a time-invariant system with the transfer function $G(z)$, which denotes the proportionality factor characterizing that component. The task at hand is to find $G(z)$.

To do so, we use the detailed signal-flow graph representation of the LMS algorithm depicted in Fig. 6.7(b) (Glover, 1977). In this diagram, we have singled out the ith tap weight for specific attention. The corresponding value of the tap input is

$$\begin{aligned} u(n - i) &= A \cos[\omega_0(n - i) + \phi] \\ &= \frac{A}{2}[e^{j(\omega_0 n + \phi_i)} + e^{-j(\omega_0 n + \phi_i)}], \end{aligned} \tag{6.32}$$

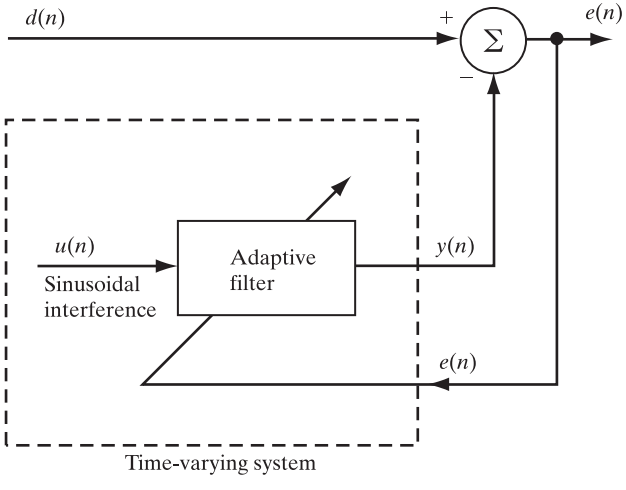


FIGURE 6.7 (a) New representation of adaptive noise canceller. *(continued on the next page.)*

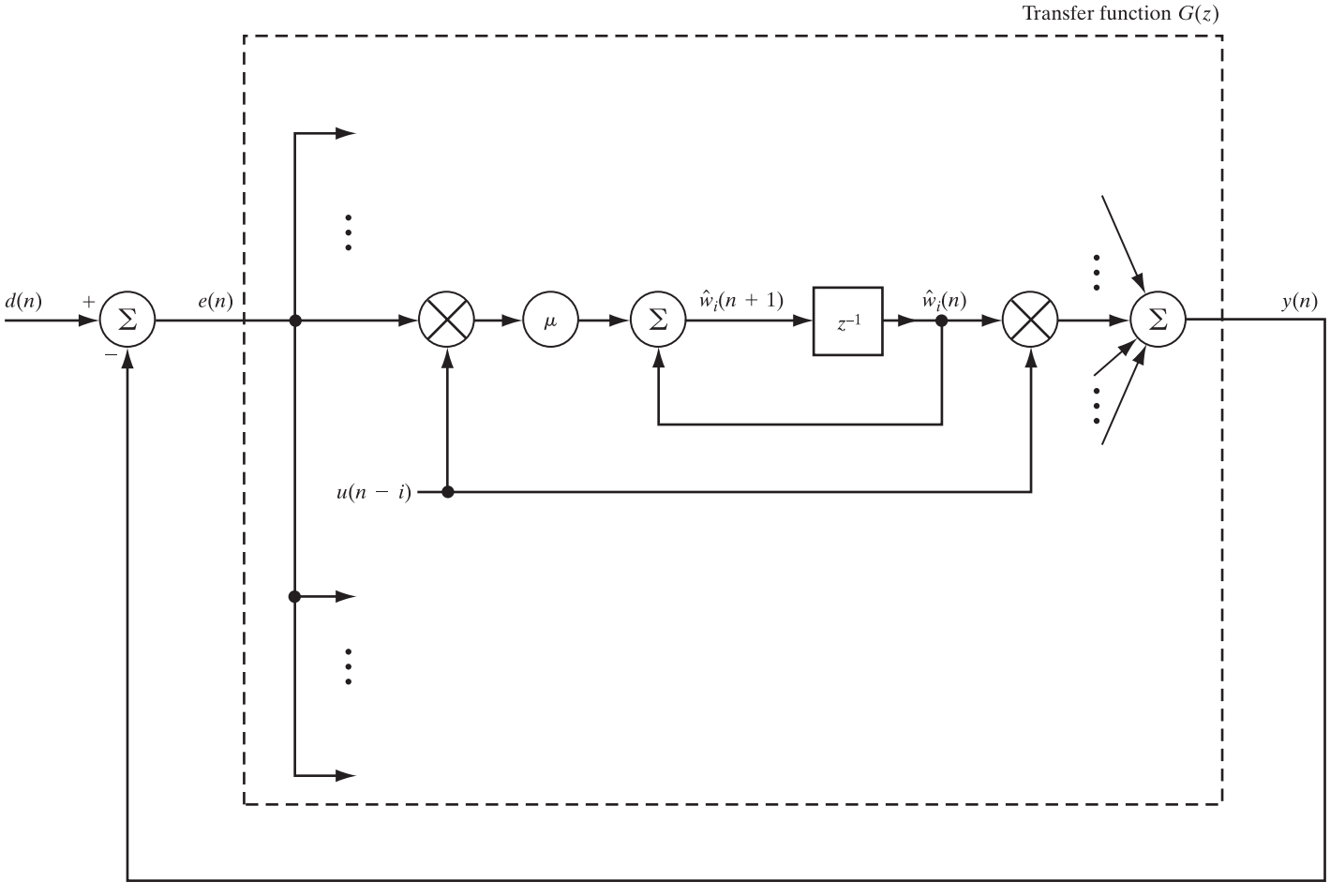


FIGURE 6.7 (b) Signal-flow graph representation of adaptive noise canceller, singling out the ith tap weight for detailed attention.

where

$$\phi_i = \phi - \omega_0 i.$$

In Fig. 6.7(b), the input $u(n - i)$ is multiplied by the estimation error $e(n)$. Hence, taking the z-transform of the product $u(n-i)e(n)$ and using $z[\cdot]$ to denote this operation, we obtain

$$z[u(n - i)e(n)] = \frac{A}{2} e^{j\phi_i} E(ze^{-j\omega_0}) + \frac{A}{2} e^{-j\phi_i} E(ze^{j\omega_0}), \tag{6.33}$$

where $E(ze^{-j\omega_0})$ is the z-transform $E(z)$ of $e(n)$ rotated counterclockwise around the unit circle through the angle ω_0. Similarly, $E(ze^{j\omega_0})$ represents a clockwise rotation through ω_0.

Next, taking the z-transform of Eq. (6.31), we get

$$z\hat{W}_i(z) = \hat{W}_i(z) + \mu z[u(n - i)e(n)], \tag{6.34}$$

where $\hat{W}_i(z)$ is the z-transform of $\hat{w}_i(n)$. Solving Eq. (6.34) for $\hat{W}_i(z)$ and using the z-transform given in Eq. (6.33), we get

$$\hat{W}_i(z) = \frac{\mu A}{2} \frac{1}{z - 1} [e^{j\phi_i} E(ze^{-j\omega_0}) + e^{-j\phi_i} E(ze^{j\omega_0})]. \tag{6.35}$$

We turn next to Eq. (6.29), which defines the adaptive filter output $y(n)$. Substituting Eq. (6.32) into Eq. (6.29), we obtain

$$y(n) = \frac{A}{2} \sum_{i=0}^{M-1} \hat{w}_i(n) [e^{j(\omega_0 n + \phi_i)} + e^{-j(\omega_0 n + \phi_i)}].$$

Evaluating the z-transform of $y(n)$ then yields

$$Y(z) = \frac{A}{2} \sum_{i=0}^{M-1} [e^{j\phi_i} \hat{W}_i(ze^{-j\omega_0}) + e^{j\phi_i} \hat{W}_i(ze^{-j\omega_0})]. \tag{6.36}$$

Thus, substituting Eq. (6.35) into Eq. (6.36), we obtain an expression for $Y(z)$ that consists of the sum of two components (Glover, 1977):

1. A *time-invariant component* defined by

$$\frac{\mu M A^2}{4} \left(\frac{1}{ze^{-j\omega_0} - 1} + \frac{1}{ze^{j\omega_0} - 1} \right),$$

which is independent of the phase ϕ_i and, therefore, the time index i.

2. A *time-varying component* that is dependent on the phase ϕ_i and hence on the variation with time i. This second component is scaled in amplitude by the factor

$$\beta(\omega_0, M) = \frac{\sin(M\omega_0)}{\sin \omega_0}.$$

For a given angular frequency ω_0, we assume that the total number of tap weights M in the FIR filter is large enough to satisfy the following approximation:

$$\frac{\beta(\omega_0, M)}{M} = \frac{\sin(M\omega_0)}{M \sin \omega_0} \approx 0. \tag{6.37}$$

Accordingly, we may justifiably ignore the time-varying component of the z-transform and so approximate $Y(z)$ by retaining the time-invariant component only:

$$Y(z) \approx \frac{\mu MA^2}{4} E(z)\left(\frac{1}{ze^{-j\omega_0} - 1} + \frac{1}{ze^{j\omega_0} - 1}\right). \tag{6.38}$$

The open-loop transfer function, relating $y(n)$ to $e(n)$, is therefore

$$\begin{aligned} G(z) &= \frac{Y(z)}{E(z)} \\ &\approx \frac{\mu MA^2}{4}\left(\frac{1}{ze^{-j\omega_0} - 1} + \frac{1}{ze^{j\omega_0} - 1}\right) \\ &= \frac{\mu MA^2}{2}\left(\frac{z\cos\omega_0 - 1}{z^2 - 2z\cos\omega_0 + 1}\right). \end{aligned} \tag{6.39}$$

The transfer function $G(z)$ has two complex-conjugate poles on the unit circle at $z = e^{\pm j\omega_0}$ and a real zero at $z = 1/\cos\omega_0$, as illustrated in Fig. 6.8(a). In other words, the adaptive noise canceller has a null point determined by the angular frequency ω_0 of the sinusoidal interference, as stated previously. (See Characteristic 1 on page 278.) Indeed, according to Eq. (6.39), we may view $G(z)$ as a pair of *integrators* that have been rotated by $\pm\omega_0$. In actuality, we see from Fig. 6.7(b) that it is the input that is first shifted in frequency by an amount $\pm\omega_0$ due to the first multiplication by the reference sinusoid $u(n)$, digitally integrated at zero frequency, and then shifted back again by the second multiplication. This overall operation is similar to a well-known technique in communications for obtaining a resonant filter that involves the combined use of two low-pass filters and heterodyning with sine and cosine at the resonant frequency (Wozencraft & Jacobs, 1965; Glover, 1977).

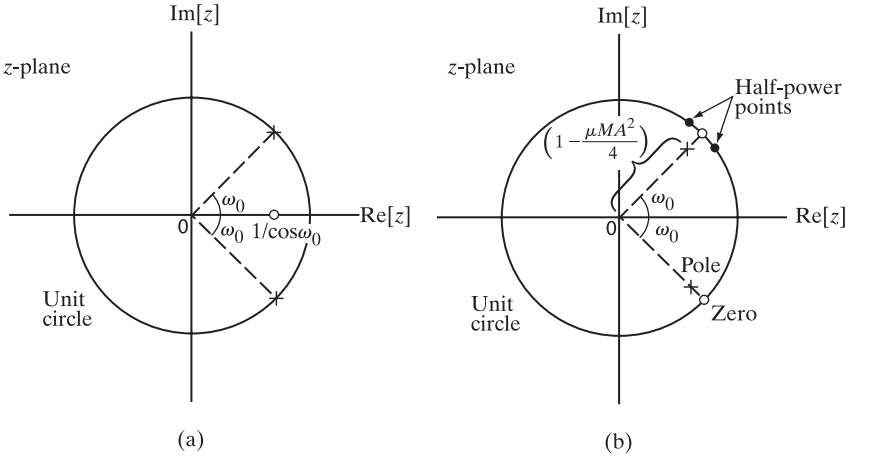


FIGURE 6.8 Approximate pole–zero patterns of (a) the open-loop transfer function $G(z)$ and (b) the closed-loop transfer function $H(z)$.

The model of Fig. 6.7(a) is recognized as a *closed-loop feedback system* whose transfer function $H(z)$ is related to the open-loop transfer function $G(z)$ via the equation

$$\begin{aligned} H(z) &= \frac{E(z)}{D(z)} \\ &= \frac{1}{1 + G(z)}, \end{aligned} \tag{6.40}$$

where $E(z)$ is the z-transform of the system output $e(n)$ and $D(z)$ is the z-transform of the system input $d(n)$. Accordingly, substituting Eq. (6.39) into Eq. (6.40), we get the approximate result

$$H(z) \approx \frac{z^2 - 2z\cos\omega_0 + 1}{z^2 - 2(1 - \mu MA^2/4)z\cos\omega_0 + (1 - \mu MA^2/2)}. \tag{6.41}$$

Equation (6.41) is the transfer function of a *second-order digital notch filter* with a notch at the normalized angular frequency ω_0. The zeros of $H(z)$ are at the poles of $G(z)$; that is, they are located on the unit circle at $z = e^{\pm j\omega_0}$. For a small value of the step-size parameter μ (i.e., a slow adaptation rate), such that

$$\frac{\mu MA^2}{4} \ll 1,$$

we find that the poles of $H(z)$ are located at

$$z \approx \left(1 - \frac{\mu MA^2}{4}\right)e^{\pm j\omega_0}. \tag{6.42}$$

In other words, the two poles of $H(z)$ lie inside the unit circle, a radial distance approximately equal to $\mu MA^2/4$ behind the zeros, as indicated in Fig. 6.8(b). The fact that the poles of $H(z)$ lie inside the unit circle means that the adaptive noise canceller is stable, as it should be for practical use in real time.

Figure 6.8(b) also includes the *half-power points* of $H(z)$. Since the zeros of $H(z)$ lie on the unit circle, the adaptive noise canceller has (in theory) a notch of infinite depth (in dB) at $\omega = \omega_0$. The *sharpness* of the notch is determined by the closeness of the poles of $H(z)$ to its zeros. The *3-dB bandwidth* B is determined by locating the two half-power points on the unit circle that are $\sqrt{2}$ times as far from the poles as they are from the zeros. Using this geometric approach, we find that the 3-dB bandwidth of the adaptive noise canceller is

$$B \approx \frac{\mu MA^2}{2} \text{ radians}. \tag{6.43}$$

Therefore, the smaller we make μ, the smaller the bandwidth B is, and the sharper the notch is. This confirms Characteristic 2 of the adaptive noise canceller that was mentioned on page 278. Its analysis is thereby completed.

Application 5: Adaptive Line Enhancement

The *adaptive line enhancer (ALE)*, illustrated in Fig. 6.9, is a system that may be used to detect a sinusoidal signal buried in a wideband noise background.[3] This figure shows that the ALE is in fact a degenerate form of the adaptive noise canceller in that its reference signal, instead of being derived separately, consists of a delayed version of the primary (input) signal. The delay, denoted by Δ in the figure, is called the *prediction depth*, or *decorrelation delay*, of the ALE, measured in units of the sampling period. The reference signal $u(n - \Delta)$ is processed by an FIR filter to produce an error signal $e(n)$, defined as the difference between the actual input $u(n)$ and the ALE's output $y(n) = u(n)$. The error signal $e(n)$ is, in turn, used to actuate the LMS algorithm for adjusting the M tap weights of the FIR filter.

Consider an input signal $u(n)$ that consists of a sinusoidal component $A \sin(\omega_0 n + \phi_0)$ buried in wideband noise $v(n)$; that is,

$$u(n) = A \sin(\omega_0 n + \phi_0) + v(n), \tag{6.44}$$

where ϕ_0 is an arbitrary phase shift and the noise $v(n)$ is assumed to have zero mean and variance σ_v^2. The ALE acts as a signal detector by virtue of two actions (Treichler, 1979):

- The prediction depth Δ is assigned a value large enough to *remove* the correlation between the noise $v(n)$ in the original input signal and the noise $v(n - \Delta)$ in the reference signal, while a simple phase shift equal to $\omega_0\Delta$ is introduced between the sinusoidal components in these two inputs.
- The tap weights of the FIR filter are adjusted by the LMS algorithm so as to minimize the mean-square value of the error signal and thereby compensate for the unknown phase shift $\omega_0\Delta$.

The net result of these two actions is the production of an output signal $y(n)$ that consists of a scaled sinusoid in noise of zero mean. In particular, when ω_0 is several multiples of π/M away from zero or π, we may express the output signal as (see Problem 3)

$$y(n) = aA \sin(\omega_0 n + \phi_0) + v_{\text{out}}(n), \tag{6.45}$$

where ϕ denotes a phase shift and $v_{\text{out}}(n)$ denotes the output noise. The scaling factor is defined by

$$a = \frac{(M/2)\text{SNR}}{1 + (M/2)\text{SNR}}, \tag{6.46}$$

where M is the length of the FIR filter and

$$\text{SNR} = \frac{A^2}{2\sigma_v^2} \tag{6.47}$$

[3]The ALE owes its origin to Widrow et al. (1975b). For a statistical analysis of its performance for the detection of sinusoidal signals in wideband noise, see Zeidler et al. (1978), Treichler (1979), and Rickard and Zeidler (1979). For a tutorial treatment of the ALE, see Zeidler (1990); the effects of signal bandwidth, input signal-to-noise ratio, noise correlation, and noise nonstationarity are explicitly considered in Zeidler's 1990 paper.

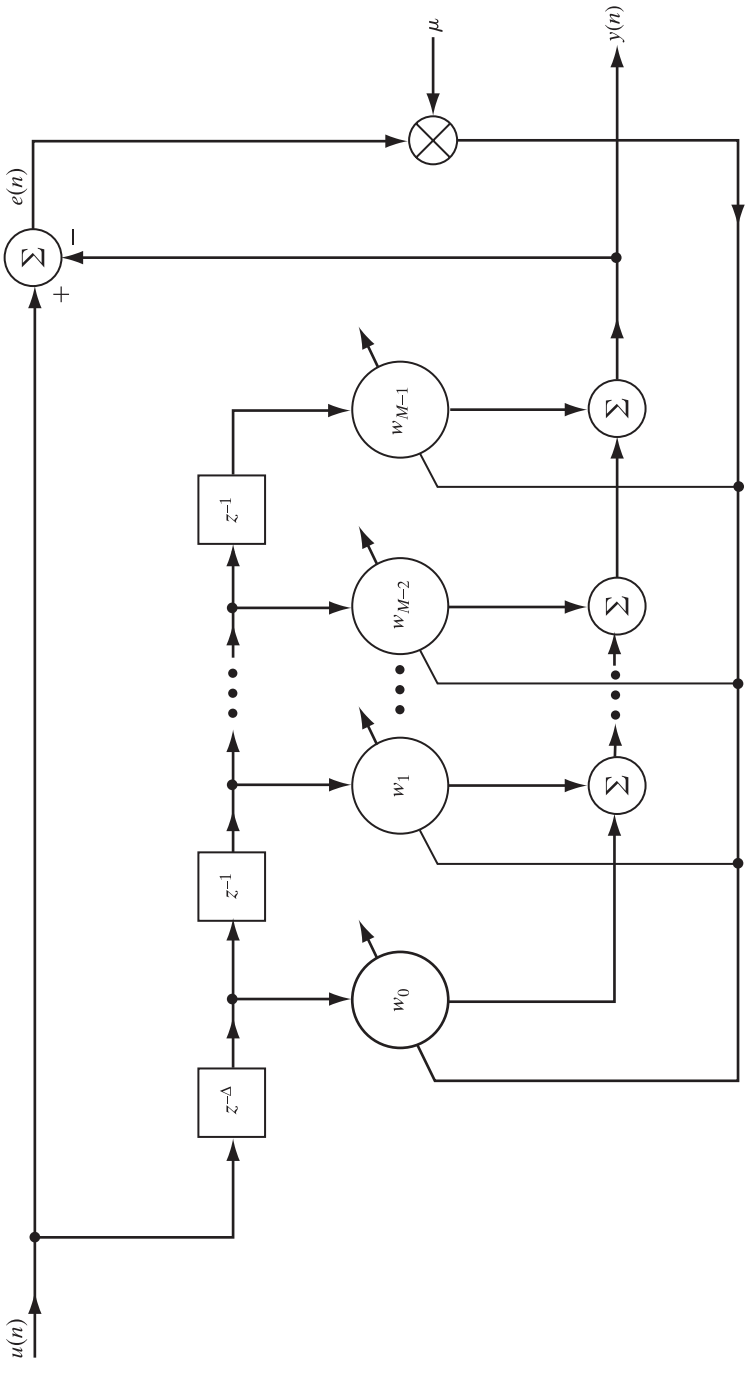


FIGURE 6.9 Adaptive line enhancer.

denotes the *signal-to-noise ratio* at the input of the ALE. According to Eq. (6.45), the ALE acts as a *self-tuning filter* whose frequency response exhibits a peak at the angular frequency ω_0 of the incoming sinusoid—hence the name "spectral line enhancer" or simply "line enhancer."

Rickard and Zeidler (1979) have also shown that the power spectral density of the ALE output $y(n)$ may be expressed as

$$S(\omega) = \frac{\pi A^2}{2}(a^2 + \mu\sigma_v^2 M)\delta(\omega - \omega_0) + \delta(\omega + \omega_0) + \mu\sigma_v^4 M + \frac{a^2\sigma_v^2}{M^2}\left[\frac{1 - \cos M(\omega - \omega_0)}{1 - \cos(\omega - \omega_0)} + \frac{1 + \cos M(\omega - \omega_0)}{1 + \cos(\omega - \omega_0)}\right], \qquad -\pi < \omega \leq \pi, \tag{6.48}$$

where $\delta(\cdot)$ denotes a Dirac delta function. To understand the composition of Eq. (6.48), we first note that an LMS algorithm operating in a stationary environment, the mean of the weight vector $\hat{\mathbf{w}}(n)$ approaches the Wiener solution $\mathbf{w}_o(n)$. A formal analysis of this behavior is presented in the next section; for now, it suffices to say that the steady-state model of the converged weight vector consists of the Wiener solution $\mathbf{w}_o$, acting in parallel with a slowly fluctuating, zero-mean random component $\hat{\mathbf{w}}_{\text{mis}}(n)$ due to gradient noise. The ALE may thus be modeled as shown in Fig. 6.10.

Recognizing that the ALE input itself consists of two components—sinusoid of angular frequency ω_0 and wideband noise $v(n)$ of zero mean and variance σ_v^2—we may distinguish four components in the power spectrum of Eq. (6.48) (Zeidler, 1990):

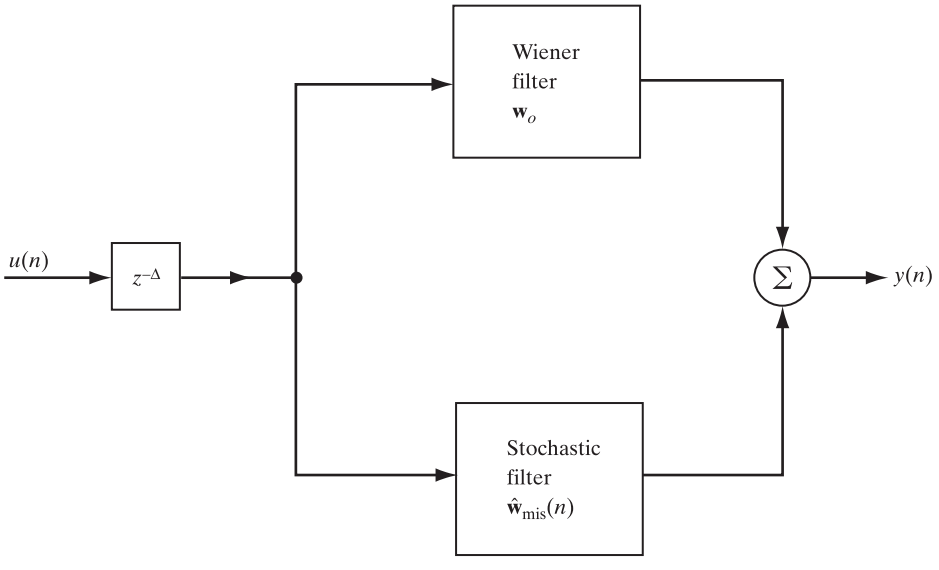


FIGURE 6.10 Model of the adaptive line enhancer.

- A sinusoidal component of angular frequency ω_0 and average power $\pi a^2 A^2/2$, which is the result of processing the input sinusoid by the Wiener filter represented by the weight vector $\mathbf{w}_o$.
- A sinusoidal component of angular frequency ω_0 and average power $\pi\mu A^2\sigma_v^2 M/2$, which is due to the stochastic filter represented by the weight vector $\hat{\mathbf{w}}_{\text{mis}}(n)$ acting on the input sinusoid.
- A wideband noise component of variance $\mu\sigma_v^4 M$, which is due to the action of the stochastic filter on the noise $v(n)$.
- A narrowband filtered noise component centered on ω_0, which results from processing the noise $v(n)$ by the Wiener filter.

These four components are depicted in Fig. 6.11. Thus, the power spectrum of the ALE output consists of a sinusoid at the center of a pedestal of narrowband filtered noise, the combination of which is embedded in a wideband noisy background. Most importantly,

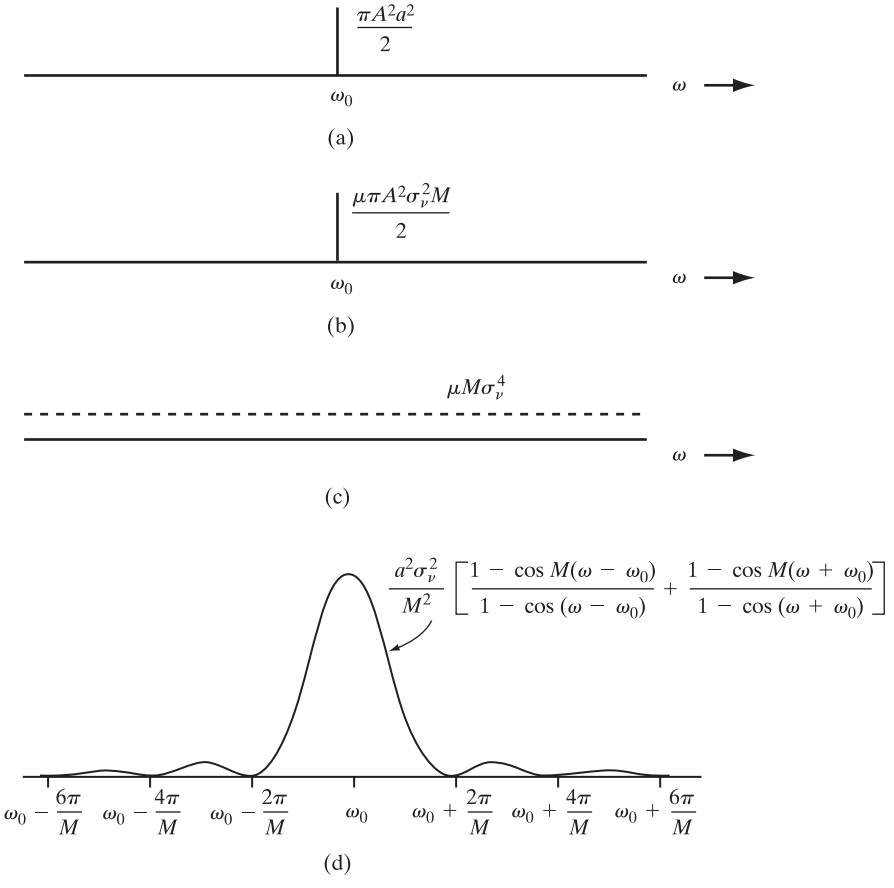


FIGURE 6.11 The four primary spectral components of the power spectral density at the ALE output. (a) Component due to Wiener filter acting on the input sinusoid. (b) Component due to stochastic filter acting on the input sinusoid. (c) Wideband noise due to the stochastic filter acting on the noise $v(n)$. (d) Narrowband noise due to the Wiener filter acting on $v(n)$.

when an adequate SNR exists at the ALE input, the ALE output is, on the average, approximately equal to the sinusoidal component present at the input, thereby providing a simple adaptive system for the detection of a sinusoid in wideband noise.

Application 6: Adaptive Beamforming

In this final example, we consider a spatial application of the LMS algorithm, namely, that of adaptive beamforming. In particular, we revisit the *generalized sidelobe canceller* (GSC) that was studied under the umbrella of Wiener filter theory in Chapter 2.

Figure 6.12 shows a block diagram of the GSC, the operation of which hinges on the combination of two actions:

1. The imposition of linear multiple constraints, designed to preserve an incident signal along a direction of interest.
2. The adjustment of some weights, in accordance with the LMS algorithm, so as to minimize the effects of interference and noise at the beamformer output.

The multiple linear constraints are described by an M-by-L matrix $\mathbf{C}$, on the basis of which a signal-blocking matrix $\mathbf{C}_a$ of size M-by-$(M-L)$ is defined by

$$\mathbf{C}_a^{\mathrm{H}}\mathbf{C} = \mathbf{O}. \tag{6.49}$$

In the GSC, the vector of weights assigned to the linear array of antenna elements is represented by

$$\mathbf{w}(n) = \mathbf{w}_q - \mathbf{C}_a\mathbf{w}_a(n), \tag{6.50}$$

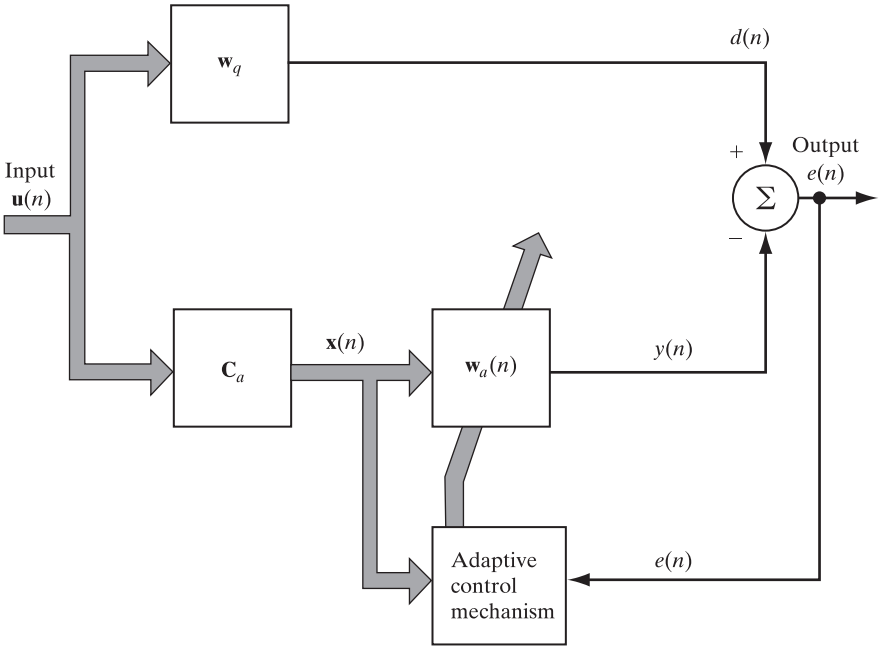


FIGURE 6.12 Block diagram of the GSC.

where $\mathbf{w}_a(n)$ is the *adjustable-weight vector* and $\mathbf{w}_q$ is the *quiescent-weight vector*. The latter component is defined in terms of the constraint matrix $\mathbf{C}$ by

$$\mathbf{w}_q = \mathbf{C}(\mathbf{C}^{\mathrm{H}}\mathbf{C})^{-1}\mathbf{g}, \tag{6.51}$$

where $\mathbf{g}$ is a prescribed gain vector.

The beamformer output is

$$\begin{aligned} e(n) &= \mathbf{w}^{\mathrm{H}}(n)\mathbf{u}(n) \\ &= (\mathbf{w}_q - \mathbf{C}_a\mathbf{w}_a(n))^{\mathrm{H}}\mathbf{u}(n) \\ &= \mathbf{w}_q^{\mathrm{H}}\mathbf{u}(n) - \mathbf{w}_a^{\mathrm{H}}(n)\mathbf{C}_a^{\mathrm{H}}\mathbf{u}(n). \end{aligned} \tag{6.52}$$

In words, the quiescent weight vector $\mathbf{w}_q$ influences that part of the input vector $\mathbf{u}(n)$ which lies in the subspace spanned by the columns of constraint matrix $\mathbf{C}$, whereas the adjustable weight vector $\mathbf{w}_a(n)$ influences the remaining part of the input vector $\mathbf{u}(n)$ that lies in the complementary subspace spanned by the columns of signal-blocking matrix $\mathbf{C}_a$. Note that $e(n)$ in Eq. (6.52) is the same as $y(n)$ in Eq. (2.107).

According to Eq. (6.52), the inner product $\mathbf{w}_q^{\mathrm{H}}\mathbf{u}(n)$ plays the role of the desired response:

$$d(n) = \mathbf{w}_q^{\mathrm{H}}\mathbf{u}(n).$$

By the same token, the matrix product $\mathbf{C}_a^{\mathrm{H}}\mathbf{u}(u)$ plays the role of the input vector for the adjustable weight vector $\mathbf{w}_a(n)$; to emphasize this point, we let

$$\mathbf{x}(n) = \mathbf{C}_a^{\mathrm{H}}\mathbf{u}(n). \tag{6.53}$$

We are now ready to formulate the LMS algorithm for the adaptation of weight vector $\mathbf{w}_a(n)$ in the GSC. Specifically, using Eqs. (6.52) and (6.53), we may go on to write

$$\begin{aligned} \mathbf{w}_a(n+1) &= \mathbf{w}_a(n) + \mu\mathbf{x}(n)e^*(n) \\ &= \mathbf{w}_a(n) + \mu\mathbf{C}_a^{\mathrm{H}}\mathbf{u}(n)(\mathbf{w}_q^{\mathrm{H}}\mathbf{u}(n) - \mathbf{w}_a^{\mathrm{H}}(n)\mathbf{C}_a^{\mathrm{H}}\mathbf{u}(n))^* \\ &= \mathbf{w}_a(n) + \mu\mathbf{C}_a^{\mathrm{H}}\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)(\mathbf{w}_q - \mathbf{C}_a\mathbf{w}_a(n)), \end{aligned} \tag{6.54}$$

where μ is the step-size parameter and all of the remaining quantities are displayed in the block diagram of Fig. 6.12.

6.4 STATISTICAL LEARNING THEORY

Derivation of the LMS algorithm was presented in Chapter 5. In this chapter, we have followed up with signal-flow graph representations, optimality considerations, and applications of the algorithm. The one topic yet to be covered is the underlying statistical learning theory of the LMS algorithm.

Much of the remainder of this chapter is devoted to this topic, in the study of which the step-size parameter, μ, plays a critical role. When μ is small, the LMS algorithm behaves like a *low-pass filter with a small cutoff frequency*. It is on the basis of

this behavior that the statistical learning theory of the algorithm addresses the following issues:

- Condition imposed on the μ for convergence.
- Misadjustment due to gradient noise.
- Efficiency of the algorithms.

All three issues are of practical importance.

What is also remarkable is the fact that under a slate of reasonable assumptions, there is a close mathematical relationship between the LMS statistical learning theory and the *Langevin theory of nonequilibrium thermodynamics*, which is discussed in Appendix F.

Basic Nonlinear Stochastic Difference Equation

To pave the way for a statistical analysis of the LMS algorithm, we find it more convenient to work with the weight-error vector rather than with the tap-weight vector itself. The rationale for this choice is that the error vector provides more useful information about the environment in a compact fashion than otherwise.

To this end, we introduce the following two definitions:

1. The *weight-error vector*

$$\boldsymbol{\varepsilon}(n) = \mathbf{w}_o - \hat{\mathbf{w}}(n), \tag{6.55}$$

where, as before, $\mathbf{w}_o$ denotes the optimum Wiener solution and $\hat{\mathbf{w}}(n)$ denotes the estimate of the tap-weight vector produced by the LMS algorithm at adaptation cycle n.

2. The *estimation error produced by the optimum Wiener filter*

$$e_o(n) = d(n) - \mathbf{w}_o^{\mathrm{H}}\mathbf{u}(n). \tag{6.56}$$

Then, using these two definitions in the update formula of Eq. (6.3) and simplifying the results, we obtain

$$\boldsymbol{\varepsilon}(n+1) = [\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]\boldsymbol{\varepsilon}(n) - \mu\mathbf{u}(n)e_o^*(n). \tag{6.57}$$

Equation (6.57) is the *nonlinear stochastic difference equation*, on which statistical learning theory of the LMS algorithm is based in what follows.

Kushner's Direct-Averaging Method

Equation (6.57) is a *stochastic difference equation* in the weight-error vector $\boldsymbol{\varepsilon}(n)$ with the characteristic feature of a system matrix equal to $[\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]$. To study the convergence behavior of such a stochastic algorithm in an *average sense*, we may invoke the *direct-averaging method* described in Kushner (1984). According to this method, the solution of the stochastic difference equation (6.57), operating under the assumption of a *small* step-size parameter μ, is (by virtue of the low-pass filtering action of the LMS algorithm) close to the solution of another stochastic difference equation whose system matrix is equal to the ensemble average, viz.,

$$\mathbb{E}[\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)] = \mathbf{I} - \mu\mathbf{R},$$

where $\mathbf{R}$ is the correlation matrix of the tap-input vector $\mathbf{u}(n)$. More specifically, we may replace the stochastic difference equation (6.57) with another stochastic difference equation described by

$$\boldsymbol{\varepsilon}_0(n+1) = (\mathbf{I} - \mu\mathbf{R})\boldsymbol{\varepsilon}_0(n) - \mu\mathbf{u}(n)e_o^*(n), \tag{6.58}$$

where, for reasons that will be become apparent presently, we have used the symbol $\boldsymbol{\varepsilon}_0(n)$ for the weight-error vector, which is different from that used in Eq. (6.57). Note, however, that the solutions of Eqs. (6.57) and (6.58) become equal for the limiting case of a vanishingly small step-size parameter μ.

Butterweck's Iterative Procedure

In an iterative procedure devised by Butterweck (1995, 2001, 2003), the solution of Eq. (6.58) is used as a starting point for generating a whole set of solutions of the original stochastic difference equation (6.57).[4] The accuracy of the solution so obtained improves with increasing adaptation cycle order. Thus, starting with the solution $\boldsymbol{\varepsilon}_0(n)$, we may express the solution of Eq. (6.57) as the sum of partial functions

$$\boldsymbol{\varepsilon}(n) = \boldsymbol{\varepsilon}_0(n) + \boldsymbol{\varepsilon}_1(n) + \boldsymbol{\varepsilon}_2(n) + \cdots, \tag{6.59}$$

where $\boldsymbol{\varepsilon}_0(n)$ is the *zero-order solution* of Eq. (6.57) for the limiting case of $\mu \longrightarrow 0$ and $\boldsymbol{\varepsilon}_1(n), \boldsymbol{\varepsilon}_2(n), \ldots$ are *higher-order corrections* to the zero-order solution for $\mu > 0$. If we now define the zero-mean difference matrix

$$\mathbf{P}(n) = \mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n) - \mathbf{R}, \tag{6.60}$$

then, substituting Eqs. (6.59) and (6.60) into Eq. (6.57) yields

$$\begin{aligned}\boldsymbol{\varepsilon}_0(n+1) + \boldsymbol{\varepsilon}_1(n+1) + \boldsymbol{\varepsilon}_2(n+1) + \cdots \\ = (\mathbf{I} - \mu\mathbf{R})[\boldsymbol{\varepsilon}_0(n) + \boldsymbol{\varepsilon}_1(n) + \boldsymbol{\varepsilon}_2(n) + \cdots] \\ - \mu\mathbf{P}(n)[\boldsymbol{\varepsilon}_0(n) + \boldsymbol{\varepsilon}_1(n) + \boldsymbol{\varepsilon}_2(n) + \cdots] - \mu\mathbf{u}(n)e_o^*(n),\end{aligned}$$

from which we readily deduce the set of coupled difference equations

$$\boldsymbol{\varepsilon}_i(n+1) = (\mathbf{I} - \mu\mathbf{R})\boldsymbol{\varepsilon}_i(n) + \mathbf{f}_i(n), \qquad i = 0, 1, 2, \ldots, \tag{6.61}$$

where the subscript i refers to the adaptation cycle order. The *driving force* $\mathbf{f}_i(n)$ for the difference equation (6.61) is defined by

$$\mathbf{f}_i(n) = \begin{cases} -\mu\mathbf{u}(n)e_o^*(n), & i = 0 \\ -\mu\mathbf{P}(n)\boldsymbol{\varepsilon}_{i-1}(n) & i = 1, 2, \ldots \end{cases}. \tag{6.62}$$

Thus, a time-varying system characterized by the stochastic difference equation (6.57) is transformed into a set of equations having the same basic format as that described in

[4]The material presented in this subsection is based on work done by Butterweck, which started with an article referring to the so-called independence theory in convergence analysis of the LMS algorithm as a disposable tool and then went on to the study of the steady-state analysis of long LMS algorithms (Butterweck, 2003, 2011).

Eq. (6.61), such that the solution to the ith equation in the set (i.e., step i in the iterative procedure) follows from the $(i-1)$th equation. In particular, the analysis of the LMS algorithm is reduced to a study of *the transmission of a stationary stochastic process through a low-pass filter with an extremely low cutoff frequency* as the step-size parameter μ approaches zero.

On the basis of Eq. (6.59), we may now express the correlation matrix of the weight-error vector $\boldsymbol{\varepsilon}(n)$ by a corresponding series as follows:

$$\begin{aligned}\mathbf{K}(n) &= \mathbb{E}[\boldsymbol{\varepsilon}(n)\boldsymbol{\varepsilon}^{\mathrm{H}}(n)] \\ &= \sum_i \sum_k \mathbb{E}[\boldsymbol{\varepsilon}_i(n)\boldsymbol{\varepsilon}_k^{\mathrm{H}}(n)], \qquad (i,k) = 0, 1, 2, \ldots .\end{aligned} \tag{6.63}$$

Expanding this series in light of the definitions given in Eqs. (6.61) and (6.62) and then grouping equal-order terms in the step-size parameter μ, we get the corresponding series expansion

$$\mathbf{K}(n) = \mathbf{K}_0(n) + \mu\mathbf{K}_1(n) + \mu^2\mathbf{K}_2(n) + \cdots, \tag{6.64}$$

where the various matrix coefficients are themselves defined as follows:

$$\mu^j\mathbf{K}_j(n) = \left\{ \begin{array}{ll} \mathbb{E}[\boldsymbol{\varepsilon}_0(n)\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)] & \text{for } j = 0 \\ \displaystyle\sum_i \sum_k \mathbb{E}[\boldsymbol{\varepsilon}_i(n)\boldsymbol{\varepsilon}_k^{\mathrm{H}}(n)] & \text{for all } (i,k) \geq 0 \\ & \text{such that } i + k = 2j - 1, 2j \end{array} \right\}. \tag{6.65}$$

The second part of Eq. (6.65) is understandable for $i + k = 2j$, dealing with the orders of magnitude $\boldsymbol{\varepsilon}_i(n)$ and $\boldsymbol{\varepsilon}_k(n)$. However, for a combination of terms with the same index i, but k replaced with $(k-1)$ such that the sum of the indices is $2j-1$, we would expect a higher result because $\boldsymbol{\varepsilon}_{k-1}(n)$ is an order of magnitude larger than $\boldsymbol{\varepsilon}_k(n)$. The surprisingly new result is ascribed to a very low degree of correlation between $\boldsymbol{\varepsilon}_{k-1}(n)$ and $\boldsymbol{\varepsilon}_k(n)$; that is, these two weight-error vectors are almost orthogonal. Note also that $\mathbf{K}_j$ is not independent of μ; rather, its Taylor expansion contains higher-order terms, which, in reality, makes the series expansion of $\mathbf{K}(n)$ less elegant than Eq. (6.64) may suggest.

The matrix coefficients in Eq. (6.65) are determined, albeit in a rather complex fashion, by the spectral and probability distribution of the environment in which the LMS algorithm operates. In a general setting with arbitrarily colored input signals, the calculation of $\mathbf{K}_j(n)$ for $j \geq 1$ can be rather tedious, except in some special cases (Butterweck, 1995). Nevertheless, Butterweck's procedure reveals an interesting structure in the statistical characteristics of the LMS algorithm.

Three Simplifying Assumptions

In much of what follows, we restrict the development of statistical LMS theory to small step sizes, embodied in the following assumptions:

> **Assumption 1.** *The step-size parameter μ is small, so the LMS algorithm acts as a low-pass filter with a low cutoff frequency.*

Under this assumption, we may use the zero-order terms $\boldsymbol{\varepsilon}_0(n)$ and $\mathbf{K}_0(n)$ as approximations to the actual $\boldsymbol{\varepsilon}(n)$ and $\mathbf{K}(n)$, respectively.

To illustrate the validity of Assumption 1, consider the simple example of an LMS algorithm using a single weight. For this example, the stochastic difference equation (6.57) simplifies to the scalar form

$$\varepsilon_0(n+1) = (1 - \mu\sigma_u^2)\varepsilon_0(n) + f_0(n)$$

where σ_u^2 is the variance of the input signal $u(n)$. This difference equation represents a low-pass filter whose transfer function has a single pole at

$$z = (1 - \mu\sigma_u^2).$$

For small μ, the pole lies inside of, and very close to, the unit circle in the z-plane, which implies a very low cutoff frequency.

Assumption 2. *The physical mechanism for generating the desired response $d(n)$ is described by a multiple linear regression model that is matched exactly by the Wiener filter; that is*,

$$d(n) = \mathbf{w}_o^{\mathrm{H}}\mathbf{u}(n) + e_o(n), \tag{6.66}$$

where the irreducible estimation error $e_o(n)$ is a white-noise process that is statistically independent of the input vector $\mathbf{u}(n)$.

The characterization of $e_o(n)$ as white noise means that its successive samples are uncorrelated, as shown by

$$\mathbb{E}[e_o(n)e_o^*(n-k)] = \begin{cases} J_{\min} & \text{for} \quad k = 0 \\ 0 & \text{for} \quad k \neq 0 \end{cases}. \tag{6.67}$$

The essence of the second assumption was discussed in Section 2.8, where we showed that, provided that the use of a multiple linear regression model is justified and the length of the Wiener filter is exactly equal to the order of the regression model, the estimation error $e_o(n)$ produced by the Wiener filter inherits the statistical characterization of the model error as white noise. [Note that the statistical independence of $e_o(n)$ from $\mathbf{u}(n)$ is stronger than the principle of orthogonality discussed in Chapter 2.]

(For experiments illustrating the validity of Assumptions 1 and 2, see Problem 18.)

The choice of a small step size according to Assumption 1 is certainly under the designer's control. To match the LMS algorithm's length to the order of the multiple regression model under Assumption 2 requires the use of a model selection criterion such as Akaike's information-theoretic criterion or Rissanen's minimum description-length criterion, which were discussed in Chapter 1. What if, for one reason or another, the multiple regression model described by Eq. (6.66) holds but the Wiener filter is mismatched to the model? In such a situation, the statistical characterization of the estimation error as white noise may not be justified. In this second possible scenario, we make the following assumption as an alternative to Assumption 2:

Assumption 3. *The input vector $\mathbf{u}(n)$ and the desired response $d(n)$ are jointly Gaussian.*

Stochastic processes produced by physical phenomena are often such that a Gaussian model is appropriate. Furthermore, the use of a Gaussian model to describe physical phenomena may be confirmed by experiments.

Thus, the *small step-size theory*, representing a special case of statistical learning of the LMS algorithm, applies to one of two possible scenarios: In one scenario, Assumption 2 holds, whereas in the other scenario, Assumption 3 holds. Between them, these two scenarios cover a wide range of environments in which the LMS algorithm operates. Most importantly, in deriving the small step-size theory, we avoid making any assumptions about the statistical independence of input data; we shall have more to say on this issue later on.

Natural Modes of the LMS Algorithm

Under Assumption 1, Butterweck's iterative procedure reduces to the following pair of equations:

$$\boldsymbol{\varepsilon}_0(n+1) = (\mathbf{I} - \mu\mathbf{R})\boldsymbol{\varepsilon}_0(n) + \mathbf{f}_0(n); \tag{6.68}$$

$$\mathbf{f}_0(n) = -\mu\mathbf{u}(n)e_o^*(n). \tag{6.69}$$

Before proceeding further, it is informative to transform the difference equation (6.68) into a simpler form by applying the *unitary similarity transformation* to the correlation matrix $\mathbf{R}$. (See Appendix E.) When we do so, we obtain

$$\mathbf{Q}^{\mathrm{H}}\mathbf{R}\mathbf{Q} = \boldsymbol{\Lambda}, \tag{6.70}$$

where $\mathbf{Q}$ is a *unitary matrix* whose columns constitute an orthogonal set of *eigenvectors* associated with the *eigenvalues* of the correlation matrix $\mathbf{R}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix consisting of the eigenvalues. To achieve the desired simplification, we also introduce the definition

$$\mathbf{v}(n) = \mathbf{Q}^{\mathrm{H}}\boldsymbol{\varepsilon}_0(n). \tag{6.71}$$

Accordingly, using Eqs. (6.70) and (6.71) and the defining property of the unitary matrix $\mathbf{Q}$, namely,

$$\mathbf{Q}\mathbf{Q}^{\mathrm{H}} = \mathbf{I}, \tag{6.72}$$

where $\mathbf{I}$ is the identity matrix, we may transform Eq. (6.68) into the form

$$\mathbf{v}(n+1) = (\mathbf{I} - \mu\boldsymbol{\Lambda})\mathbf{v}(n) + \boldsymbol{\phi}(n), \tag{6.73}$$

where the new vector $\boldsymbol{\phi}(n)$ is defined in terms of $\mathbf{f}_0(n)$ by the transformation

$$\boldsymbol{\phi}(n) = \mathbf{Q}^{\mathrm{H}}\mathbf{f}_0(n). \tag{6.74}$$

For a partial characterization of the stochastic force vector $\boldsymbol{\phi}(n)$, we may express its mean and correlation matrix over an ensemble of LMS algorithms as follows:

1. *The mean of the stochastic force vector $\boldsymbol{\phi}(n)$ is zero*:

$$\mathbb{E}[\boldsymbol{\phi}(n)] = \mathbf{0} \quad \text{for all } n. \tag{6.75}$$

2. *The correlation matrix of the stochastic force vector* $\boldsymbol{\phi}(n)$ *is a diagonal matrix*; that is,

$$\mathbb{E}[\boldsymbol{\phi}(n)\boldsymbol{\phi}^{\mathrm{H}}(n)] = \mu^2 J_{\min}\boldsymbol{\Lambda}, \tag{6.76}$$

where $J_{\min}$ *is the minimum mean-square error produced by the Wiener filter and* $\boldsymbol{\Lambda}$ *is the diagonal matrix of eigenvalues of the correlation matrix of the tap-input vector* $\mathbf{u}(n)$.

These two properties imply that the individual components of the transformed stochastic force vector $\boldsymbol{\phi}$ are *uncorrelated* with each other.

Property 1 follows immediately from the *principle of orthogonality* that is inherent in the Wiener filter. [See Eq. (2.11).] Specifically, using the defining equations (6.69) and (6.74), we write

$$\begin{aligned}\mathbb{E}[\boldsymbol{\phi}(n)] &= -\mu\mathbf{Q}^{\mathrm{H}}\mathbb{E}[\mathbf{u}(n)e_o^*(n)] \\ &= \mathbf{0},\end{aligned}$$

where the expectation term is zero by virtue of the principle of orthogonality.

The correlation matrix of $\boldsymbol{\phi}(n)$ is defined by

$$\mathbb{E}[\boldsymbol{\phi}(n)\boldsymbol{\phi}^{\mathrm{H}}(n)] = \mu^2\mathbf{Q}^{\mathrm{H}}\mathbb{E}[\mathbf{u}(n)e_o^*(n)e_o(n)\mathbf{u}^{\mathrm{H}}(n)]\mathbf{Q}. \tag{6.77}$$

To evaluate the expectation term in Eq. (6.77), we invoke Assumption 2 or Assumption 3, depending on which operational scenario is applicable:

- When the Wiener filter is perfectly matched to the multiple regression model of Eq. (6.66), the estimation error $e_o(n)$ is white (Assumption 2). Accordingly, we may factorize the expectation term in Eq. (6.77) as follows:

$$\begin{aligned}\mathbb{E}[\mathbf{u}(n)e_o^*(n)e_o(n)\mathbf{u}^{\mathrm{H}}(n)] &= \mathbb{E}[e_o^*(n)e_o(n)]\mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)] \\ &= J_{\min}\mathbf{R}.\end{aligned}$$

  Hence,

$$\begin{aligned}\mathbb{E}[\boldsymbol{\phi}(n)\boldsymbol{\phi}^{\mathrm{H}}(n)] &= \mu^2 J_{\min}\mathbf{Q}^{\mathrm{H}}\mathbf{R}\mathbf{Q} \\ &= \mu^2 J_{\min}\boldsymbol{\Lambda},\end{aligned}$$

  which demonstrates Property 2.

- When the Wiener filter is mismatched to the multiple regression model of Eq. (6.66), we invoke Assumption 3. Specifically, with the input data $\mathbf{u}(n)$ and $d(n)$ assumed to be jointly Gaussian, the estimation error $e_o(n)$ is likewise Gaussian. Then, applying the *Gaussian moment-factoring theorem* of Eq. (1.101), we may write

$$\begin{aligned}&\mathbb{E}[\mathbf{u}(n)e_o^*(n)e_o(n)\mathbf{u}^{\mathrm{H}}(n)] \\ &\qquad = \mathbb{E}[\mathbf{u}(n)e_o^*(n)]\mathbb{E}[e_o(n)\mathbf{u}^{\mathrm{H}}(n)] + \mathbb{E}[e_o^*(n)e_o(n)]\mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)],\end{aligned}$$

  which, by virtue of the principle of orthogonality, reduces to

$$\begin{aligned}\mathbb{E}[\mathbf{u}(n)e_o^*(n)e_o(n)\mathbf{u}^{\mathrm{H}}(n)] &= \mathbb{E}[e_o^*(n)e_o(n)]\mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)] \\ &= J_{\min}\mathbf{R}.\end{aligned}$$

Hence, using this result in Eq. (6.77), we get

$$\begin{aligned}\mathbb{E}[\boldsymbol{\phi}(n)\boldsymbol{\phi}^{\mathrm{H}}(n)] &= \mu^2 J_{\min}\mathbf{Q}^{\mathrm{H}}\mathbf{R}\mathbf{Q}\\ &= \mu^2 J_{\min}\boldsymbol{\Lambda},\end{aligned}$$

which again demonstrates Property 2.

According to Eq. (6.73), the number of natural modes constituting the transient response of the LMS algorithm is equal to the number of adjustable parameters in the filter. In particular, for the kth natural mode of the LMS algorithm, we have

$$v_k(n+1) = (1-\mu\lambda_k)v_k(n) + \phi_k(n), \qquad k = 1, 2, \ldots, M. \tag{6.78}$$

Comparing this equation with the corresponding equation (4.20) for the steepest-descent algorithm, we see that the transient behavior of the LMS algorithm differs from the steepest-descent algorithm for the Wiener filter in the presence of the stochastic force $\phi_k(n)$—a difference that has profound implications. In particular, from Eq. (6.78), it follows that the change in the natural model v_k from one adaptation cycle to the next is

$$\begin{aligned}\Delta v_k(n) &= v_k(n+1) - v_k(n)\\ &= -\mu\lambda_k v_k(n) + \phi_k(n), \qquad k = 1, 2, \ldots, M,\end{aligned} \tag{6.79}$$

which is naturally split into two parts: a *damping force*, denoted by $\mu\lambda_k v_k(n)$, and a *stochastic force*, denoted by $\phi_k(n)$.

On the Relationship between LMS Statistical Learning Theory and Langevin Equation of Nonequilibrium Thermodynamics

The linear difference equation (6.79) bears a close relationship with the Langevin equation of nonequilibrium thermodynamics, which is discussed in Appendix F. Indeed, Eq. (6.79) is the discrete-time version of the Langevin equation, which is demonstrated in Table 6.1. Just as the Langevin equation, driven by a stochastic force of its own, never reaches an equilibrium condition in thermodynamics, so it is with the LMS algorithm that never reaches an equilibrium condition in signal processing, as evidenced by Eq. (6.79). Moreover, just as the Langevin equation characterizes Brownian motion, so it is with the LMS algorithm that performs a Brownian motion of its own, a characterization that will be demonstrated experimentally in Sections 6.7 and 6.8.

To elaborate further on Eq. (6.79), we note that like $\phi_k(n)$, the natural mode $v_k(n)$ of the LMS algorithm is stochastic with a mean and mean-square value of its own.

TABLE 6.1 Analogy between the LMS Algorithm and the Langevin Equation

	LMS algorithm (discrete time, n)	Langevin equation (continuous time, t)
Stochastic force	$\phi_k(n)$	$\Gamma(t)$
Damping force	$-\mu\lambda_k$	$-\gamma$
Sample function	$\Delta v_k(n)$	$v(t)$

Let $v_k(0)$ denote the initial value of $v_k(n)$. We may then solve the stochastic difference equation (6.78) to produce the solution

$$v_k(n) = (1 - \mu\lambda_k)^n v_k(0) + \sum_{i=0}^{n-1} (1 - \mu\lambda_k)^{n-1-i} \phi_k(i), \tag{6.80}$$

where the first term is the *natural component* of $v_k(n)$ and the summation term is the *forced component*. Invoking the statistical properties of the stochastic force $\phi_k(n)$ described in Eqs. (6.75) and (6.76), we obtain the following formulas for the first two moments of the natural mode $v_k(n)$ for $k = 1, 2, \ldots, M$ (see Problem 11):

1. *Mean value:*

$$\mathbb{E}[v_k(n)] = v_k(0)(1 - \mu\lambda_k)^n. \tag{6.81}$$

2. *Mean-square value:*

$$\mathbb{E}[|v_k(n)|^2] = \frac{\mu J_{\min}}{2 - \mu\lambda_k} + (1 - \mu\lambda_k)^{2n}\left(|v_k(0)|^2 - \frac{\mu J_{\min}}{2 - \mu\lambda_k}\right). \tag{6.82}$$

We may thus summarize the exposition of small step-size theory of the LMS algorithm presented thus far as follows:

> Provided that the step-size parameter of an LMS algorithm is small, the natural modes of the algorithm execute Brownian motion about some fixed values, with the first and second moments of the natural modes being defined by Eqs. (6.81) and (6.82), respectively.

With this material at our disposal, we are ready to probe more deeply into statistical characterization of the LMS algorithm.

Learning Curves

It is common practice to use ensemble-average *learning curves* to study the statistical performance of adaptive filtering algorithms. In particular, we may identify two kinds of learning curves:

1. *The mean-square-error (MSE) learning curve*, which is based on ensemble averaging of the squared estimation error $|e(n)|^2$. This learning curve is thus a plot of the mean-square error

$$J(n) = \mathbb{E}[|e(n)|^2] \tag{6.83}$$

   versus the adaptation cycle n.

2. *The mean-square deviation (MSD) learning curve*, which is based on ensemble averaging of the squared error deviation $\|\boldsymbol{\varepsilon}(n)\|^2$. This second learning curve is thus a plot of the *mean-square deviation*

$$\mathcal{D}(n) = \mathbb{E}[\|\boldsymbol{\varepsilon}(n)\|^2] \tag{6.84}$$

   versus the adaptation cycle n.

Unlike the case of the steepest-descent algorithm, considered in Chapter 4, both the mean-square error $J(n)$ and mean-square deviation $\mathcal{D}(n)$ in the LMS algorithm depend on the adaptation cycle n, because the estimation error $e(n)$ and weight-error vector $\boldsymbol{\varepsilon}(n)$ are both *nonstationary processes*.

The estimation error produced by the LMS algorithm is expressed as

$$\begin{aligned} e(n) &= d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n) \\ &= d(n) - \mathbf{w}_o^{\mathrm{H}}\mathbf{u}(n) + \boldsymbol{\varepsilon}^{\mathrm{H}}(n)\mathbf{u}(n) \\ &= e_o(n) + \boldsymbol{\varepsilon}^{\mathrm{H}}(n)\mathbf{u}(n) \\ &\approx e_o(n) + \boldsymbol{\varepsilon}_o^{\mathrm{H}}(n)\mathbf{u}(n) \quad \text{for } \mu \text{ small,} \end{aligned} \tag{6.85}$$

where $e_o(n)$ is the estimation error produced by the Wiener filter and $\boldsymbol{\varepsilon}_0(n)$ is the zero-order weight-error vector of the LMS algorithm. Hence, the mean-square error produced by the corresponding LMS algorithm is given by

$$\begin{aligned} J(n) &= \mathbb{E}[|e(n)|^2] \\ &\approx \mathbb{E}[(e_o(n) + \boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)\mathbf{u}(n))(e_o^*(n) + \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}_0(n))] \\ &= J_{\min} + 2\mathrm{Re}\{\mathbb{E}[e_o^*(n)\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)\mathbf{u}(n)]\} + \mathbb{E}[\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}_0(n)], \end{aligned} \tag{6.86}$$

where $J_{\min}$ is the minimum mean-square error produced by the Wiener filter and $\mathrm{Re}\{\cdot\}$ denotes the real part of the quantity enclosed between the braces.

The second term on the right-hand side of Eq. (6.86) is zero for the following reasons, depending on which scenario applies:

- Under Assumption 2, the irreducible estimation error $e_o(n)$ produced by the Wiener filter is statistically independent of the input vector $\mathbf{u}(n)$. At adaptation cycle n, the zero-order weight-error vector $\boldsymbol{\varepsilon}_0(n)$ depends on past values of $e_o(n)$, a relationship that follows from the iterated use of Eq. (6.58). Hence, we may write

$$\begin{aligned} \mathbb{E}[e_o^*(n)\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)\mathbf{u}(n)] &= \mathbb{E}[e_o^*(n)]\mathbb{E}[\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)\mathbf{u}(n)] \\ &= 0. \end{aligned} \tag{6.87}$$

- The null result of Eq. (6.87) also holds under Assumption 3, For the kth components of $\boldsymbol{\varepsilon}_0(n)$ and $\mathbf{u}(n)$, we write the expectation

$$\mathbb{E}[e_o^*(n)\varepsilon_{0,k}^*(n)u(n-k)], \qquad k = 0, 1, \ldots, M-1.$$

  Assuming that the input vector $\mathbf{u}(n)$ and the desired response $d(n)$ are jointly Gaussian, and the estimation error $e_o(n)$ is therefore also Gaussian, then applying the identity described in Eq. (1.100), we immediately obtain

$$\mathbb{E}[e_o^*(n)\varepsilon_{0,k}^*(n)u(n-k)] = 0 \qquad \text{for all } k.$$

  Hence, Eq. (6.87) follows.

To evaluate the remaining term in Eq. (6.86), we use the fact that, as a consequence of Assumption 1, the variations of the weight-error vector $\boldsymbol{\varepsilon}_0(n)$ with time are slow compared with those of the input vector $\mathbf{u}(n)$; that is, the spectral content

of the input signal $u(n)$ is significantly different from that of the weight error $\varepsilon_k(n)$ for $k = 0, 1, \ldots, M-1$. Consequently, in light of the direct-averaging method, we may replace the stochastic product $\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)$ by its expected value and so write

$$\begin{aligned}\mathbb{E}[\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}_0(n)] &\approx \mathbb{E}[\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)\mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]\boldsymbol{\varepsilon}_0(n)] \\ &= \mathbb{E}[\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)\mathbf{R}\boldsymbol{\varepsilon}_0(n)].\end{aligned}$$

The *trace* of a scalar quantity is the same as the quantity itself; thus, taking the trace of this expectation and then interchanging the operations of expectation and trace yields

$$\begin{aligned}\mathbb{E}[\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}_0(n)] &\approx \mathrm{tr}\{\mathbb{E}[\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)\mathbf{R}\boldsymbol{\varepsilon}_0(n)]\} \\ &= \mathbb{E}\{\mathrm{tr}[\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)\mathbf{R}\boldsymbol{\varepsilon}_0(n)]\}.\end{aligned}$$

Next, from matrix algebra, we use the identity

$$\mathrm{tr}[\mathbf{AB}] = \mathrm{tr}[\mathbf{BA}],$$

where $\mathbf{A}$ and $\mathbf{B}$ are matrices of compatible dimensions. So putting $\mathbf{A} = \boldsymbol{\varepsilon}_0^{\mathrm{H}}$ and $\mathbf{B} = \mathbf{R}\boldsymbol{\varepsilon}_0$, we may write

$$\begin{aligned}\mathbb{E}[\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}_0(n)] &\approx \mathbb{E}\{\mathrm{tr}[\mathbf{R}\boldsymbol{\varepsilon}_0(n)\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)]\} \\ &= \mathrm{tr}\{\mathbb{E}[\mathbf{R}\boldsymbol{\varepsilon}_0(n)\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)]\} \\ &= \mathrm{tr}\{\mathbf{R}\mathbb{E}[\boldsymbol{\varepsilon}_0(n)\boldsymbol{\varepsilon}_0^{\mathrm{H}}(n)]\} \\ &= \mathrm{tr}[\mathbf{R}\mathbf{K}_0(n)], \qquad (6.88)\end{aligned}$$

where $\mathbf{R}$ is the correlation matrix of the tap inputs and $\mathbf{K}_0(n)$ is the zero-order approximation to the weight-error correlation matrix defined in the first line of Eq. (6.65).

Accordingly, using Eqs. (6.87) and (6.88) in Eq. (6.86), we may approximate the mean-square error produced by the LMS algorithm simply as

$$J(n) \approx J_{\min} + \mathrm{tr}[\mathbf{R}\mathbf{K}_0(n)]. \qquad (6.89)$$

Equation (6.89) indicates that, for all n, the mean-square value of the estimation error in the LMS algorithm consists of two components: the minimum mean-square error $J_{\min}$ and a component depending on the transient behavior of the zero-order weight-error correlation matrix $\mathbf{K}_0(n)$. Since the latter component is nonnegative definite for all n, *the LMS algorithm produces a mean-square error $J(n)$ that is in excess of the minimum mean-square error $J_{\min}$*. This statement confirms suboptimality of the LMS algorithm compared to the Wiener filter, which was stated previously is Section 6.2.

We now formally define the *excess mean-square error* as the difference between the mean-square error $J(n)$ produced by the LMS algorithm at adaptation cycle n and the minimum mean-square error $J_{\min}$ produced by the corresponding Wiener filter. Denoting the excess mean-square error by $J_{\mathrm{ex}}(n)$, we write

$$\begin{aligned}J_{\mathrm{ex}}(n) &= J(n) - J_{\min} \\ &\approx \mathrm{tr}[\mathbf{R}\mathbf{K}_0(n)]. \qquad (6.90)\end{aligned}$$

Employing the definition in the first line of Eq. (6.65) and proceeding in a manner similar to that described in Eq. (6.88), we may go on to express $J_{\mathrm{ex}}(n)$ as

$$\begin{aligned} J_{\text{ex}}(n) &\approx \text{tr}\{\mathbf{R}\mathbb{E}[\boldsymbol{\varepsilon}_0(n)\boldsymbol{\varepsilon}_0^{\text{H}}(n)]\} \\ &= \text{tr}\{\mathbf{R}\mathbb{E}[\mathbf{Q}\mathbf{v}(n)\mathbf{v}^{\text{H}}(n)\mathbf{Q}^{\text{H}}]\} \\ &= \mathbb{E}\{\text{tr}[\mathbf{R}\mathbf{Q}\mathbf{v}(n)\mathbf{v}^{\text{H}}(n)\mathbf{Q}^{\text{H}}]\} \\ &= \mathbb{E}\{\text{tr}[\mathbf{v}^{\text{H}}(n)\mathbf{Q}^{\text{H}}\mathbf{R}\mathbf{Q}\mathbf{v}(n)]\} \\ &= \mathbb{E}\{\text{tr}[\mathbf{v}^{\text{H}}(n)\boldsymbol{\Lambda}\mathbf{v}(n)]\} \\ &= \sum_{k=1}^{M} \lambda_k \mathbb{E}[|v_k(n)|^2]. \end{aligned} \tag{6.91}$$

Again, using $\boldsymbol{\varepsilon}_0(n)$ as an approximation for the weight-error vector $\boldsymbol{\varepsilon}(n)$ under Assumption 1, we may correspondingly use Eq. (6.71) to approximate the mean-square deviation of Eq. (6.84) as

$$\begin{aligned} \mathcal{D}(n) &\approx \mathbb{E}[\|\boldsymbol{\varepsilon}_0(n)\|^2] \\ &= \mathbb{E}[\|\mathbf{v}(n)\|^2] \\ &= \sum_{k=1}^{M} \mathbb{E}[|v_k(n)|^2], \end{aligned} \tag{6.92}$$

where, in the second line, we have used the fact that the Euclidean norm of a vector is invariant to rotation by a unitary similarity transformation. Now, let $\lambda_{\min}$ and $\lambda_{\max}$ denote the minimum and maximum eigenvalues, respectively, of the correlation matrix $\mathbf{R}$; that is,

$$\lambda_{\min} \leq \lambda_k \leq \lambda_{\max}, \qquad k = 1, 2, \ldots, M. \tag{6.93}$$

Using Eqs. (6.91) and (6.92), we may therefore bound the mean-square deviation as follows:

$$\lambda_{\min}\mathcal{D}(n) \leq J_{\text{ex}}(n) \leq \lambda_{\max}\mathcal{D}(n) \qquad \text{for all } n.$$

Equivalently, we may write

$$\frac{J_{\text{ex}}(n)}{\lambda_{\min}} \geq \mathcal{D}(n) \geq \frac{J_{\text{ex}}(n)}{\lambda_{\max}} \qquad \text{for all } n. \tag{6.94}$$

This twofold inequality shows that, for all n, the mean-square deviation $\mathcal{D}(n)$ is *lower bounded* by $J_{\text{ex}}(n)/\lambda_{\max}$ and *upper bounded* by $J_{\text{ex}}(n)/\lambda_{\min}$. Accordingly, we may make the statement:

> The mean-square deviation of the LMS algorithm decays with an increasing number of adaptation cycles in a manner similar to that of the excess mean-square error.

It therefore suffices to focus attention on the convergence behavior of $J_{\text{ex}}(n)$.

6.5 TRANSIENT BEHAVIOR AND CONVERGENCE CONSIDERATIONS

According to Eq. (6.81), the exponential factor $(1-\mu\lambda_k)^n$ governs the evolution of the mean of the kth natural mode of the LMS algorithm with adaptation cycle n. A necessary condition for this exponential factor to decay to zero is to have

$$-1 < 1 - \mu\lambda_k < +1 \qquad \text{for all } k,$$

which, in turn, imposes the following condition on the step-size parameter:

$$0 < \mu < \frac{2}{\lambda_{\max}}. \tag{6.95}$$

Here, $\lambda_{\max}$ is the largest eigenvalue of the correlation matrix $\mathbf{R}$. However, in studying the transient behavior of the LMS algorithm, we have to recall that the derivation of Eq. (6.81) is subject to the requirement that the step-size parameter μ be small. We may satisfy this requirement by assigning to μ a value that is small compared with the reciprocal of $\lambda_{\max}$. Then, for all k, the exponential factor $(1 - \mu\lambda_k)^n$ is assured of decaying to zero with increasing n, in which case we may write

$$\mathbb{E}([v_k(n)]) \to 0 \quad \text{as} \quad n \to \infty \qquad \text{for all } k. \tag{6.96}$$

Equivalently, by virtue of Eqs. (6.55) and (6.71) and the approximation of $\boldsymbol{\varepsilon}(n)$ by $\boldsymbol{\varepsilon}_0(n)$, we have

$$\mathbb{E}[\hat{\mathbf{w}}(n)] \to \mathbf{w}_o \quad \text{as} \quad n \to \infty, \tag{6.97}$$

where $\mathbf{w}_o$ is the Wiener solution. However, such a convergence criterion is of little practical value, since a sequence of asymptotically zero-mean random variables need not tend to zero.

We avoid this shortcoming of the convergence of the mean by considering a stronger criterion: *convergence in the mean square*, which is inherently linked to the ensemble-average learning curves and which therefore explains the practical importance of learning curves in the study of adaptive filters. Using Eqs. (6.82) and (6.91) in the first line of Eq. (6.90), we may express the mean-square error produced by the LMS algorithm for small μ compared to the permissible limit, $2/\lambda_{\max}$, as shown by

$$\begin{aligned} J(n) &= J_{\min} + \mu J_{\min} \sum_{k=1}^{M} \frac{\lambda_k}{2 - \mu\lambda_k} + \sum_{k=1}^{M} \lambda_k \left(|v_k(0)|^2 - \frac{\mu J_{\min}}{2 - \mu\lambda_k} \right) (1 - \mu\lambda_k)^{2n} \\ &\approx J_{\min} + \frac{\mu J_{\min}}{2} \sum_{k=1}^{M} \lambda_k + \sum_{k=1}^{M} \lambda_k \left(|v_k(0)|^2 - \frac{\mu J_{\min}}{2} \right) (1 - \mu\lambda_k)^{2n}. \end{aligned} \tag{6.98}$$

For most practical applications of the LMS algorithm, the error incurred in using the approximate formula in the second line of Eq. (6.98), compared to the first line of the equation, is small enough to be ignored.

The evolution of $J(n)$ with adaptation cycle n is governed by the exponential factor $(1 - \mu\lambda_k)^{2n}$ for all natural modes $k = 1, 2, \ldots, M$. Recalling that the derivation of Eq. (6.97) is subject to the requirement that the step-size parameter μ be small, we may again satisfy this requirement by choosing μ small compared with $1/\lambda_{\max}$. Then, under this condition, the exponential factor $(1 - \mu\lambda_k)^{2n}$ is assured of decaying to zero with increasing n for all k. Accordingly, we may characterize the learning curves of LMS algorithms for small μ by stating the following principle:

> The ensemble-average learning curve of an LMS algorithm does not exhibit oscillations; rather, it decays exponentially to the constant value

$$\begin{aligned} J(\infty) &= J_{\min} + \mu J_{\min} \sum_{k=1}^{M} \frac{\lambda_k}{2 - \mu\lambda_k} \\ &\approx J_{\min} + \frac{\mu J_{\min}}{2} \sum_{k=1}^{M} \lambda_k, \qquad \mu \text{ small.} \end{aligned} \tag{6.99}$$

Note, however, applicability of this principle is based on the zero-order solution $\boldsymbol{\varepsilon}_0(n)$, the use of which is justified only for small μ.[5]

As pointed out briefly in Section 5.2, the use of a small value for the step-size parameter μ is also responsible for robust behavior of the LMS algorithm, an important practical issue that will be discussed in detail in Chapter 11. We may therefore go on to say that assigning a small value to the μ is not only justifiable from a practical perspective but also from a statistical learning theoretic perspective of the LMS algorithm.

Misadjustment

We next characterize the LMS algorithm by introducing a new parameter: the *misadjustment*, which is formally defined as

$$\mathcal{M} = \frac{J_{\text{ex}}(\infty)}{J_{\min}}. \tag{6.100}$$

In words:

> The misadjustment is defined as the ratio of the steady-state value of the excess mean-square error $J_{\text{ex}}(\infty)$ to the minimum mean-square error $J_{\min}$.

Using Eqs. (6.99) and (6.100), we may thus write for small μ:

$$\mathcal{M} = \frac{\mu}{2}\sum_{k=1}^{M}\lambda_k. \tag{6.101}$$

The misadjustment $\mathcal{M}$ is a dimensionless parameter that provides a measure of how close the LMS algorithm is to optimality in the mean-square-error sense. The smaller $\mathcal{M}$ is compared with unity, the more accurate is the adaptive filtering action being performed by the LMS algorithm. It is customary to express $\mathcal{M}$ as a percentage. For example, a misadjustment of 10 percent means that the LMS algorithm produces a mean-square error (after adaptation is completed) that is 10 percent greater than the minimum mean-square error $J_{\min}$. Such performance is ordinarily considered to be satisfactory in practice.

From the eigendecomposition theory presented in Appendix E we recognize that the trace of matrix $\mathbf{R}$ is equal to the sum of its eigenvalues. We may therefore rewrite Eq. (6.101) in the equivalent form

$$\mathcal{M} = \frac{\mu}{2}\,\text{tr}[\mathbf{R}]. \tag{6.102}$$

This formula is the result of an eigendecomposition. It may also be derived directly without having to go through such an approach as discussed in Problem 12.

Regardless of which procedure is used to arrive at Eq. (6.102), we note that, for a stationary process composed of the tap inputs in the FIR filter of Fig. 5.1, the correlation

[5]No reliable statement can be made on the stability of the LMS algorithm for moderate and large values of μ. In this latter situation, the whole LMS algorithm (i.e., zero-order as well as higher-order solutions) would have to be considered. Unfortunately, under this complicated scenario, the statistical learning theory of the LMS algorithm becomes mathematically unmanageable.

matrix $\mathbf{R}$ is not only nonnegative definite, but also Toeplitz, with all of the elements on its main diagonal equal to $r(0)$. Since $r(0)$ is itself equal to the mean-square value of the input at each of the M taps in the FIR filter, we have

$$\begin{aligned}\text{tr}[\mathbf{R}] &= Mr(0)\\ &= \sum_{k=0}^{M-1} \mathbb{E}[|u(n-k)|^2]\\ &= \mathbb{E}[\|\mathbf{u}(n)\|^2].\end{aligned}$$

Thus, using the term "total tap-input power" to refer to the sum of the mean-square values of the tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$ in the FIR filter of the LMS algorithm, we may recast Eq. (6.102) for the misadjustment as shown by

$$\mathcal{M} = \frac{\mu}{2} \times (\text{total tap-input power}). \tag{6.103}$$

6.6 EFFICIENCY

In Chapter 5, on the method of stochastic gradient descent, we introduced the notion of efficiency as one of the principles of that method. Therein, we referred to the *rate of convergence* as a measure of *statistical efficiency*, which is formally defined for linear adaptive filtering algorithms as follows:

> The rate of convergence is a representative measure of the cost involved in converging to the Wiener solution by a linear adaptive filtering algorithm under the umbrella of statistical learning theory of that algorithm operating on a stationary environment.

For this measure to provide a common principle for comparative evaluation of different linear adaptive filtering algorithms, exemplified by the LMS algorithm studied in this chapter and the RLS algorithm to be studied in Chapter 10, it is understood that similar statistical assumptions are made on the environment, wherein observable data made up of the input vector and corresponding desired response are picked.

Time Constants of the Learning Curve

The rate of convergence of the LMS algorithm is determined by the time constants of its discrete transient response. To get a handle on this issue, we have to go back to Eq. (6.98), which embodies this transient response, herewith denoted by $t_k(n)$. Putting aside all the constants in this equation, we may express $t(n)$ in terms of the eigenvalues of the condition matrix $\mathbf{R}$ as follows:

$$\begin{aligned}t_k(n) &= (1-\mu\lambda_k)^{2n}\\ &= (1-2\mu\lambda_k+\mu^2\lambda_k^2)^n.\end{aligned}$$

Typically, with μ assigned a small value, the squared term $\mu^2\lambda_k^2$ is small enough to be ignored for all practical purposes. We may therefore approximate $t_k(n)$ as follows:

$$t_k(n) \approx (1 - 2\mu\lambda_k)^n \quad \text{for } k = 1, 2, \ldots, M. \tag{6.104}$$

Evaluating Eq. (6.104) at the initial condition $n = 0$ and adaptation cycle $n = 1$, we have

$$\begin{aligned} t_k(0) &= 1 \\ t_k(1) &= 1 - 2\mu\lambda_k. \end{aligned}$$

Let $\tau_{\text{mse},k}$ denote the *time constant* corresponding to the kth eigenvalue. Then, referring to Fig. 6.13, we find that

$$\begin{aligned} \frac{1}{\tau_{\text{mse},k}} &= t(0) - t(1) \\ &\approx 1 - (1 - 2\mu\lambda_k) \\ &= 2\mu\lambda_k, \quad k = 1, 2, \ldots, M. \end{aligned}$$

Equivalently, we may write

$$\tau_{\text{mse},k} \approx \frac{1}{2\mu\lambda_k}, \quad k = 1, 2, \ldots, M. \tag{6.105}$$

From this equation, we may draw the following conclusions (Widrow & Kamenetsky, 2003):

1. As the tap weights of the LMS algorithm converge toward the Wiener solution, the learning curve undergoes a *geometric progression* around the minimum mean-square error, $J_{\min}$.

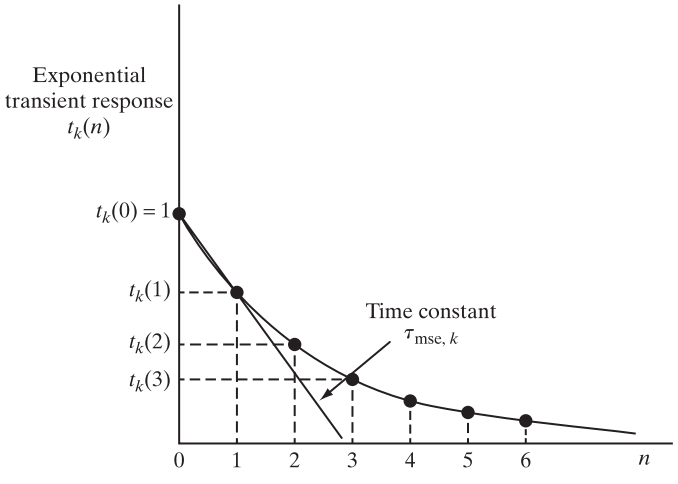


FIGURE 6.13 Exponential transient response, $t_k(n)$ pertaining to kth eigenvalue of the LMS algorithm, where n is the number of adaptation cycles normalized with respect to some arbitrary time interval for convenience of presentation.

2. The LMS algorithm is *sensitive to the eigenvalue spread* of the correlation matrix **R**, hence its potential vulnerability to eigenvalue spreads of a stationary process. (This issue will be discussed experimentally in Sections 6.7 and 6.8.)
3. Under worst conditions, the rate of convergence will be dominated by the smallest eigenvalue, denoted by $\lambda_{\min}$; the *slowest time constant* for the learning curve will therefore be approximately defined by

$$\tau_{\text{mse, min}} \approx \frac{1}{2\mu\lambda_{\min}},$$

which leads to a slow rate of convergence.

All along, we have to be aware of the stability constraint imposed on the LMS algorithm in Eq. (6.95), which, interestingly enough, involves the largest eigenvalue, $\lambda_{\max}$.

Relationship Between Misadjustment and Rate of Convergence

According to Eq. (6.103), the misadjustment factor of the LMS algorithm, $\mathcal{M}$, is directly proportional to the step-size parameter, μ; for $\mathcal{M}$ to be small, μ should be correspondingly kept small. In Chapter 11, we will show that by keeping μ small, robustness of the LMS algorithm is guaranteed in H^∞ theory. For this latter reason, we may reiterate that keeping μ small is in the best practical interests of the LMS algorithm.

However, the time constants of the learning curve and therefore the rate of convergence of the LMS algorithm are inversely proportional to the step-size parameter, μ. Accordingly, keeping μ small results in a long rate of convergence and, therefore, a less statistically efficient LMS algorithm.

In other words, keeping μ small, robustness of the LMS algorithm (with a small misadjustment factor) is traded off for efficiency. Detailed discussion of this trade-off is deferred to Chapter 11.

6.7 COMPUTER EXPERIMENT ON ADAPTIVE PREDICTION

For our first experiment with the LMS algorithm, we study a first-order, autoregressive (AR) process using *real-valued data*. The process is described by the following difference equation:

$$u(n) = -au(n-1) + \nu(n), \tag{6.106}$$

where a is the (one and only) parameter of the process and $\nu(n)$ is a zero-mean white-noise process of variance σ_ν^2. To estimate the parameter a, we use an adaptive predictor of order one, as depicted in Fig. 6.14. The *real* LMS algorithm for adaptation of the (one and only) tap weight of the predictor is written as

$$\hat{w}(n+1) = \hat{w}(n) + \mu u(n-1) f(n), \tag{6.107}$$

where

$$f(n) = u(n) - \hat{w}(n)u(n-1)$$

is the prediction error.

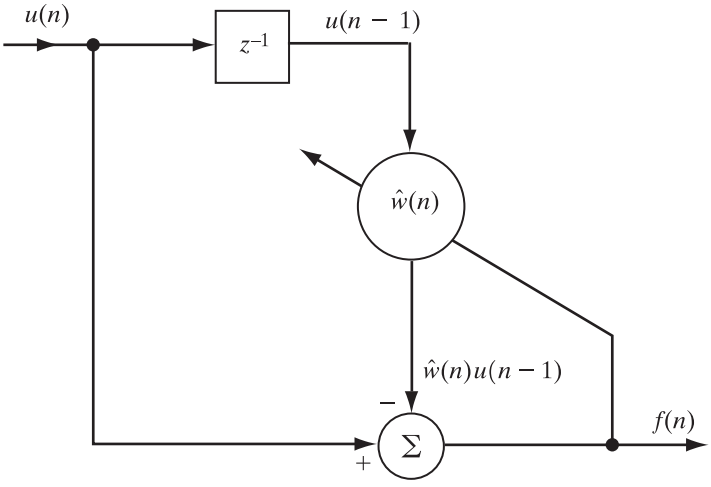


FIGURE 6.14 Adaptive first-order predictor.

The experiment is performed for the following parameterization of the AR process, assumed to have zero mean:

AR parameter, $a = -0.99$

Variance of the process, $\sigma_u^2 = 0.936$

The experiment has two objectives:

1. Study experimental plots of the algorithm's learning curves for varying μ.
2. Validate experimentally the statistical learning theory of the algorithm for small μ.

These two issues are addressed in this order in the discussion that follows.

Objective I. Learning Curves for Varying μ

Figure 6.15 shows experimental plots of the learning curves of the LMS algorithm [i.e., the mean-square error $J(n)$ versus the number of adaptation cycles, n] for the AR parameter a previously and varying step-size parameter μ. Specifically, the values used for μ are 0.01, 0.05, and 0.1. The ensemble averaging was performed over 100 independent Monte Carlo runs of the experiment. The plots of Fig. 6.15 confirm the following properties:

- As the step-size parameter μ is reduced, the rate of convergence of the LMS algorithm is correspondingly increased.
- A reduction in the step-size parameter μ also has the effect of reducing variation in the experimentally computed learning curve.

Objective 2. Validation of the Statistical Learning Theory

With the AR process of order one (i.e., $M = 1$), as described in Eq. (6.106), we note the following points for the problem at hand:

1. The correlation matrix $\mathbf{R}$ is a scalar with an eigenspectrum that consists of an eigenvalue λ_1 equal to σ_u^2 and an associated eigenvector $\mathbf{q}_1$ equal to unity.

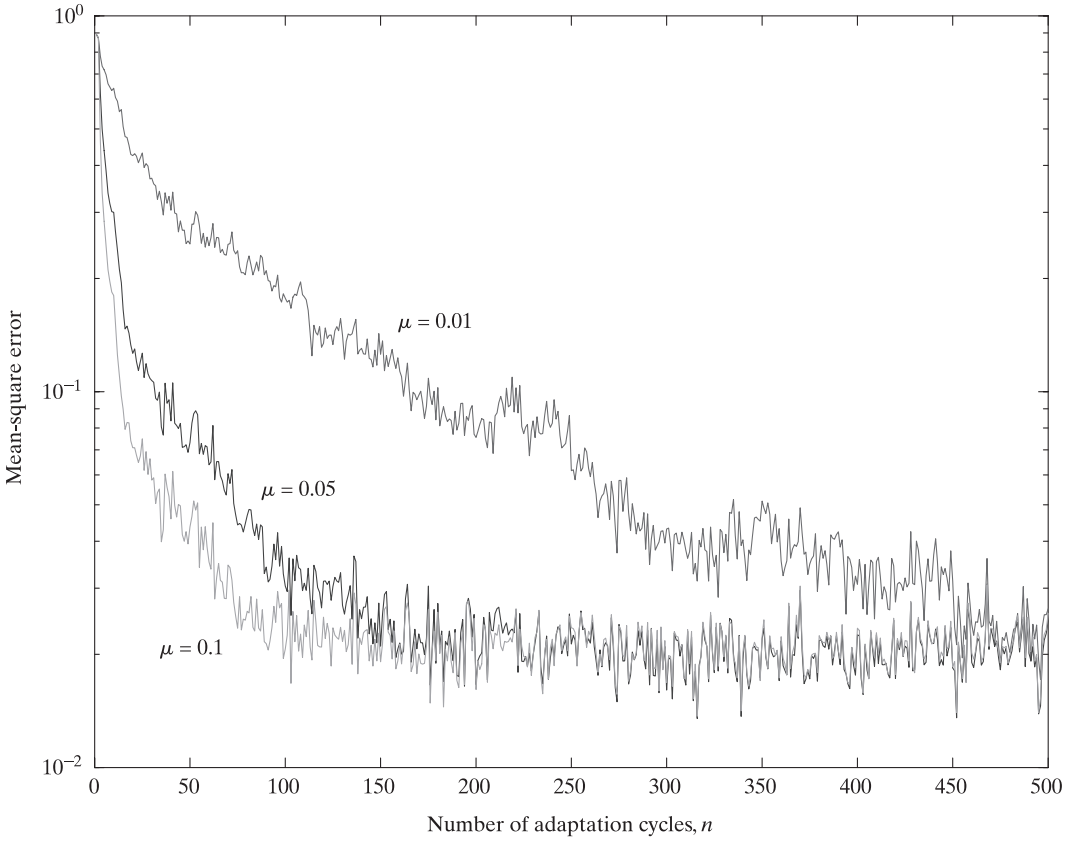


FIGURE 6.15 Experimental learning curves of adaptive first-order predictor for varying step-size parameter μ.

2. The Wiener solution w_o for the tap weight of the predictor is equal to $-a$.
3. The minimum mean-square error $J_{\min}$ is equal to σ_v^2, denoting the variance of the additive white noise $v(n)$.

These values are summarized here:

$$\left.\begin{aligned} \lambda_1 &= \sigma_u^2 \\ q_1 &= 1 \\ w_o &= -a \\ J_{\min} &= \sigma_v^2 \end{aligned}\right\}. \tag{6.108}$$

The first item to be checked is the choice of μ that justifies application of the small step-size theory developed in Section 6.5. As explained previously, this requirement may be satisfied by assigning to μ a value very small compared with $2/\lambda_{\max}$, where $\lambda_{\max}$ is the largest eigenvalue of $\mathbf{R}$. For this experiment, $\lambda_{\max} = \lambda_1$. With $\lambda_1 = \sigma_u^2$, the requirement of the small step-size theory is satisfied for the specified value of σ_u^2 by choosing $\mu = 0.001$.

With this background, we may now proceed with the second objective of the experiment, addressing two related aspects of the LMS statistical learning theory.

Random-Walk Behavior. Figure 6.16 plots a single realization of the LMS algorithm for $\mu = 0.001$. In this figure, we see that starting from the initial condition $\hat{w}(0) = 0$, the sample estimate of the tap weight, $\hat{w}(n)$, increases upward gradually with increasing number of adaptation cycles, n. After about $n = 500$, the estimate $\hat{w}(n)$ reaches a "quasi-steady-state" that is characterized by a random behavior around the optimum Wiener solution, $w_o = -a = 0.99$. The plot presented in Fig. 6.17 magnifies this random behavior, around the steady state, which has the appearance of a *random walk* with mean equal to 0.0439 and variance equal to 0.0074; these results were computed using 100 Monte Carlo runs. Thus, assuming that the additive noise in the update formula for $\hat{w}$ is Gaussian distributed, we may go on to say that the random walk assumes the form of Brownian motion.

Agreement between Theory and Experiment. Finally, another verification of the small step-size theory is presented in Fig. 6.18 on learning curves. The experimental curve, labeled "Experiment," was obtained by ensemble averaging the squared value of the prediction error $f(n)$ over 100 independent Monte Carlo runs for varying n. The theoretical curve, labeled "Theory," follows from Eq. (6.98), the second line of which, for the problem at hand, reduces to

$$J(n) \approx \sigma_v^2\left(1 + \frac{\mu}{2}\sigma_u^2\right) + \sigma_u^2\left(a^2 - \frac{\mu}{2}\sigma_v^2\right)(1 - \mu\sigma_u^2)^{2n}, \qquad \mu \text{ small.} \quad (6.109)$$

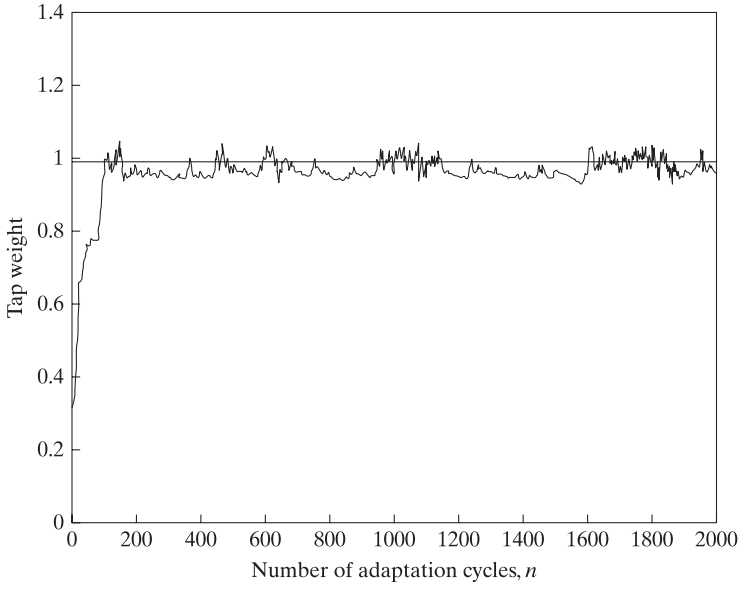


FIGURE 6.16 Transient response, displaying evolution of the tap weight of the adaptive one-step predictor across time, around the Wiener solution represented by the faint horizontal line.

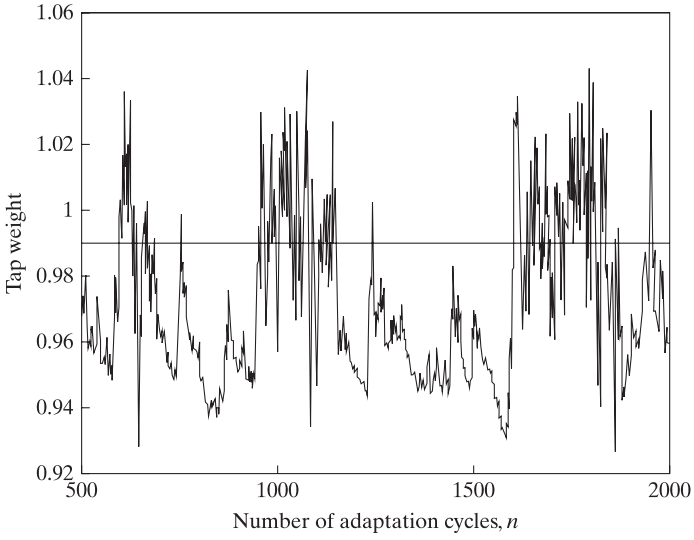


FIGURE 6.17 Illustrating the random behavior of the only tap weight of the predictor.

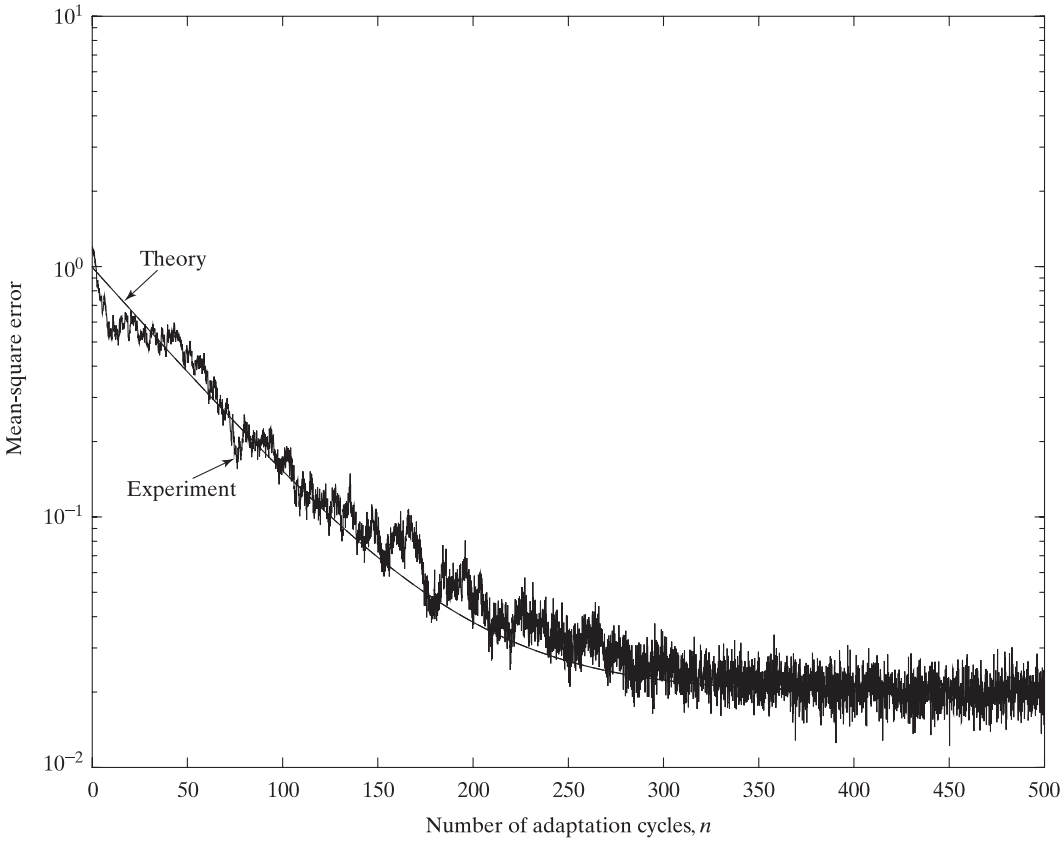


FIGURE 6.18 Comparison of experimental results with theory for the adaptive predictor, based on the mean-square error for $\mu = 0.001$.

For an AR process of order one, the variance of the white noise $\nu(n)$ is defined by [see Eq. (1.71)]

$$\sigma_\nu^2 = \sigma_u^2(1 - a^2). \tag{6.110}$$

Accordingly, the use of Eq. (6.110) in Eq. (6.109) yields

$$J(n) \approx \sigma_u^2(1 - a^2)\left(1 + \frac{\mu}{2}\sigma_u^2\right) + \sigma_u^2\left(a^2 + \frac{\mu}{2}a^2\sigma_u^2 - \frac{\mu}{2}\sigma_u^2\right)(1 - \mu\sigma_u^2)^{2n}, \quad \mu \text{ small.} \tag{6.111}$$

The experimental curve in Fig. 6.18 was computed using the parameter values: $a = -0.99$, $\sigma_u^2 = 0.936$, and $\mu = 0.001$. From the figure we observe that the agreement between theory and experiment is remarkably good for the entire learning curve, justifying validity of the LMS transient behavior presented in Section 6.5.

6.8 COMPUTER EXPERIMENT ON ADAPTIVE EQUALIZATION

In this second computer experiment, we study the use of the LMS algorithm for *adaptive equalization* of a linear dispersive channel that produces (unknown) distortion. Here, again, we assume that the data are all *real valued*. Figure 6.19 shows the block diagram of the system used to carry out the study. Random-number generator 1 provides the test signal x_n, used for probing the channel, whereas random-number generator 2 serves as the source of additive white noise $\nu(n)$ that corrupts the channel output. These two random-number generators are independent of each other. The adaptive equalizer has the task of correcting for the distortion produced by the channel in the presence of the additive white noise. Random-number generator 1, after suitable delay, also supplies the desired response applied to the adaptive equalizer in the form of a training sequence.

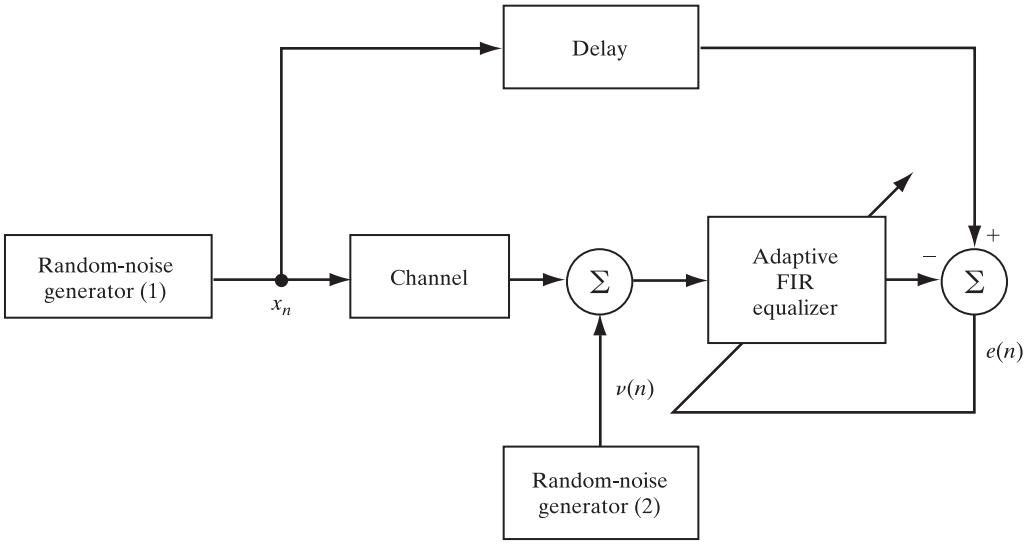


FIGURE 6.19 Block diagram of adaptive equalizer experiment.

The random sequence $\{x_n\}$ applied to the channel input consists of a *Bernoulli sequence*, with $x_n = \pm 1$ and the random variable x_n having zero mean and unit variance. The impulse response of the channel is described by the raised cosine[6]

$$h_n = \begin{cases} \frac{1}{2}\left[1 + \cos\left(\frac{2\pi}{W}(n-2)\right)\right], & n = 1, 2, 3, \\ 0, & \text{otherwise} \end{cases} \tag{6.112}$$

where the parameter W controls the amount of amplitude distortion produced by the channel, with the distortion increasing with W.

Equivalently, the parameter W controls the eigenvalue spread $\chi(\mathbf{R})$ of the correlation matrix of the tap inputs in the equalizer, with the eigenvalue spread increasing with W. The sequence $\nu(n)$, produced by the second random-number generator, has zero mean and variance $\sigma_\nu^2 = 0.001$.

The equalizer has $M = 11$ taps. Since the channel has an impulse response h_n that is symmetric about adaptation cycle $n = 2$, as depicted in Fig. 6.20(a), it follows that the optimum tap weights w_{on} of the equalizer are likewise symmetric about adaptation cycle $n = 5$, as depicted in Fig. 6.20(b). Accordingly, the channel input x_n is delayed by $\Delta = 2 + 5 = 7$ samples to provide the desired response for the equalizer. By selecting the

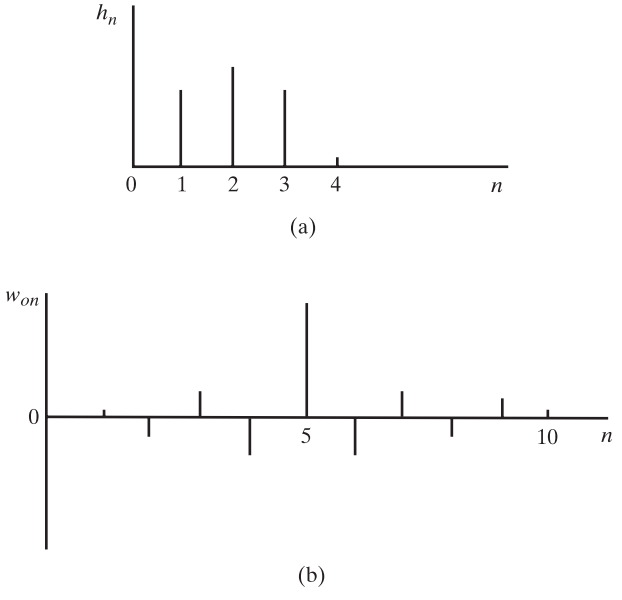


FIGURE 6.20 (a) Impulse response of channel; (b) impulse response of optimum FIR equalizer.

[6]The parameters specified in this experiment closely follow the paper by Satorius and Alexander (1979).

delay Δ to match the midpoint of the FIR equalizer, the LMS algorithm is enabled to provide an approximate inversion of both the minimum-phase and non-minimum-phase components of the channel response.

The experiment is in three parts that are intended to evaluate the response of the adaptive equalizer using the LMS algorithm to changes in the eigenvalue spread $\chi(\mathbf{R})$ and step-size parameter μ. Before proceeding to describe the results of the experiment, however, we compute the eigenvalues of the correlation matrix $\mathbf{R}$ of the 11 tap inputs in the equalizer.

Correlation Matrix of the Equalizer Input

The first tap input of the equalizer at adaptation cycle n is

$$u(n) = \sum_{k=1}^{3} h_k x(n - k) + \nu(n), \tag{6.113}$$

where all the parameters are real valued. Hence, the correlation matrix $\mathbf{R}$ of the 11 tap inputs of the equalizer, $u(n), u(n-1), \ldots, u(n-10)$, is a symmetric 11-by-11 matrix. Also, since the impulse response h_n has nonzero values only for $n = 1, 2, 3$, and the noise process $\nu(n)$ is white with zero mean and variance σ_ν^2, the correlation matrix $\mathbf{R}$ is *quintdiagonal*. That is, the only nonzero elements of $\mathbf{R}$ are on the main diagonal and the four diagonals directly above and below it, two on either side, as shown by the special structure

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) & r(2) & 0 & \cdots & 0 \\ r(1) & r(0) & r(1) & r(2) & \cdots & 0 \\ r(2) & r(1) & r(0) & r(1) & \cdots & 0 \\ 0 & r(2) & r(1) & r(0) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & r(0) \end{bmatrix}, \tag{6.114}$$

where

$$\begin{aligned} r(0) &= h_1^2 + h_2^2 + h_3^2 + \sigma_\nu^2, \\ r(1) &= h_1 h_2 + h_2 h_3, \end{aligned}$$

and

$$r(2) = h_1 h_3.$$

The variance $\sigma_\nu^2 = 0.001$; hence, h_1, h_2, and h_3 are determined by the value assigned to the parameter W in Eq. (6.112).

In Table 6.2, we have listed (1) values of the autocorrelation function $r(l)$ for lag $l = 0, 1, 2$, and (2) the smallest eigenvalue $\lambda_{\min}$, the largest eigenvalue $\lambda_{\max}$, and the eigenvalue spread $\chi(\mathbf{R}) = \lambda_{\max}/\lambda_{\min}$. We thus see that the eigenvalue spread ranges from 6.078 (for $W = 2.9$) to 46.821 (for $W = 3.5$).

Experiment 1: Effect of Eigenvalue Spread. For the first part of the experiment, the step-size parameter was held fixed at $\mu = 0.075$. This value is chosen in accordance with the requirement that the step-size parameter μ be small compared with the reciprocal of $\lambda_{\max}$, which denotes the largest eigenvalue of the correlation matrix $\mathbf{R}$. (Indeed, all

TABLE 6.2 Summary of Parameters for the Experiment on Adaptive Equalization

W	2.9	3.1	3.3	3.5
$r(0)$	1.096	1.157	1.226	1.302
$r(1)$	0.439	0.560	0.673	0.777
$r(2)$	0.048	0.078	0.113	0.151
λ_{min}	0.33	0.214	0.126	0.066
λ_{max}	2.030	2.376	2.726	3.071
$\chi(\mathbf{R}) = \lambda_{max}/\lambda_{min}$	6.078	11.124	21.713	46.822

three values of the step-size parameter μ used in the experiment satisfy this requirement, thereby assuring convergence.)

For each eigenvalue spread, an approximation to the ensemble-average learning curve of the adaptive equalizer is obtained by averaging the instantaneous-squared-error "$e^2(n)$ versus n" curve over 200 independent Monte Carlo runs of the computer experiment. The results of this computation are shown in Fig. 6.21, from which we see

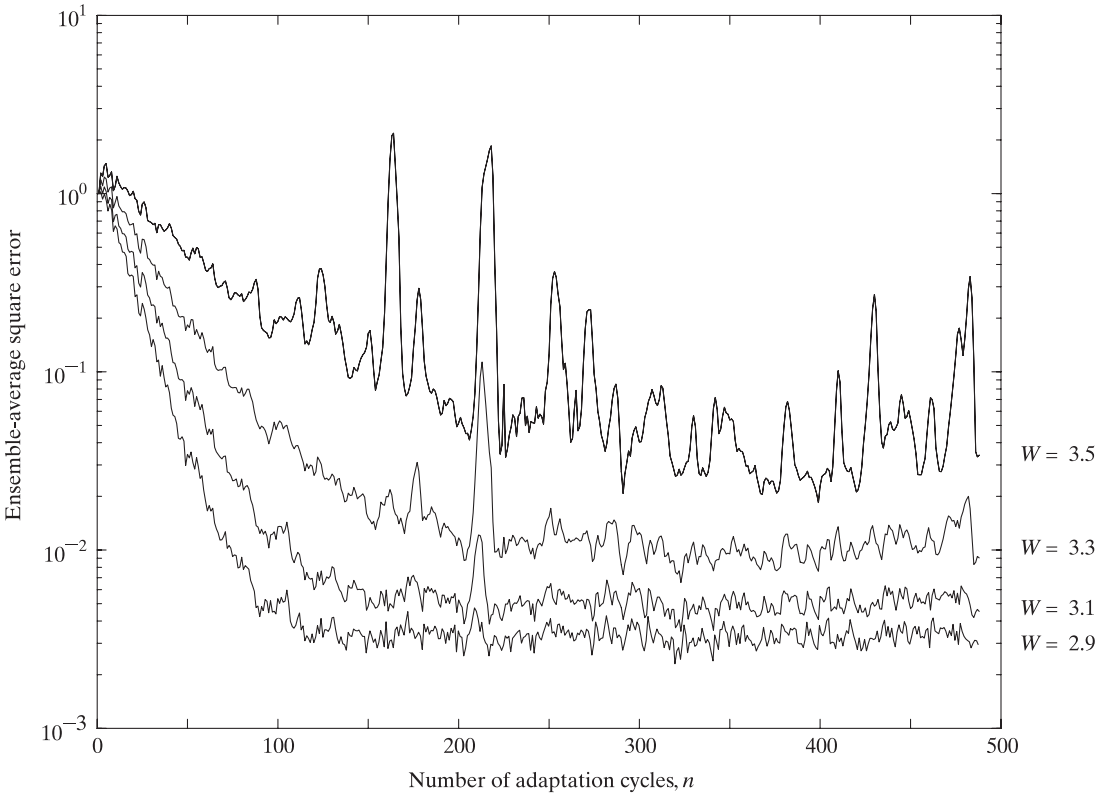


FIGURE 6.21 Learning curves of the LMS algorithm for an adaptive equalizer with number of taps $M = 11$, step-size parameter $\mu = 0.075$, and varying eigenvalue spread $\chi(\mathbf{R})$.

that increasing the eigenvalue spread $\chi(\mathbf{R})$ has the effect of slowing down the rate of convergence of the adaptive equalizer and also increasing the steady-state value of the average squared error. For example, when $\chi(\mathbf{R}) = 6.0782$, approximately 80 adaptation cycles are required for the adaptive equalizer to converge in the mean square, and the average squared error (after 500 adaptation cycles) equals approximately 0.003. On the other hand, when $\chi(\mathbf{R}) = 46.8216$ (i.e., the equalizer input is ill conditioned), the equalizer requires approximately 200 adaptation cycles to converge in the mean square, and the resulting average squared error (after 500 adaptation cycles) equals approximately 0.04.

In Fig. 6.22, we have plotted the ensemble-average impulse response of the adaptive equalizer after 1000 adaptation cycles for each of the four eigenvalue spreads of interest. As before, the ensemble averaging was carried out over 200 independent Monte Carlo runs of the experiment. We see that in each case the ensemble-average impulse response of the adaptive equalizer is very close to being symmetric with respect to the center tap, as expected. The variation in the impulse response from one eigenvalue

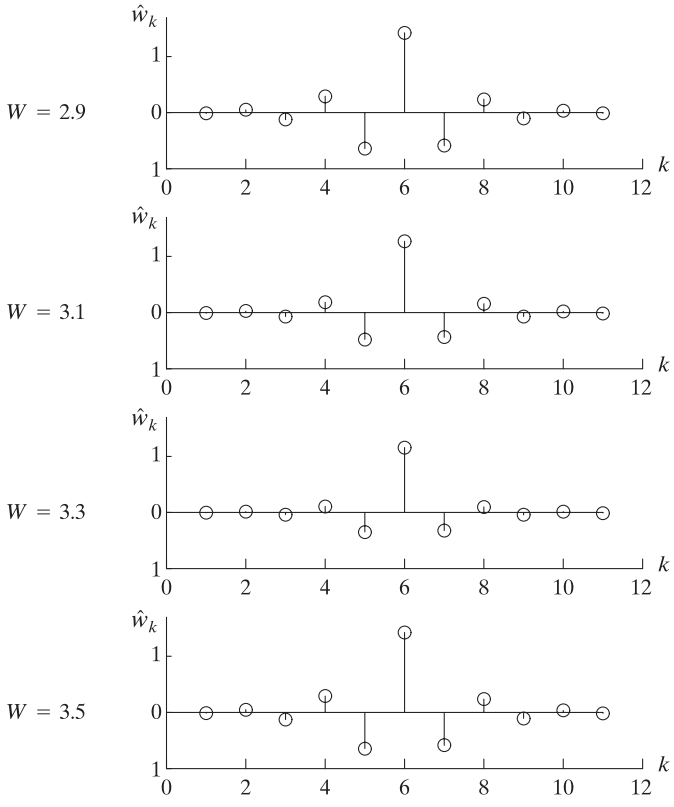


FIGURE 6.22 Ensemble-average impulse response of the adaptive equalizer (after 1000 adaptation cycles) for each of four different eigenvalue spreads.

spread to another merely reflects the effect of a corresponding change in the impulse response of the channel.

Experiment 2: Effect of Step-Size Parameter. For the second part of the experiment, the parameter W in Eq. (6.112) was fixed at 3.1, yielding an eigenvalue spread of 11.1238 for the correlation matrix of the tap inputs in the equalizer. The step-size parameter μ was this time assigned one of the three values 0.075, 0.025, and 0.0075.

Figure 6.23 shows the results of the computation. As before, each learning curve is the result of ensemble averaging the instantaneous-squared-error "$e^2(n)$ versus n" curve over 200 independent Monte Carlo runs of the computer experiment.

The results confirm that the rate of convergence of the adaptive equalizer is highly dependent on the step-size parameter μ. For a large step-size parameter ($\mu = 0.075$), the equalizer converged to steady-state conditions in approximately 120 adaptation

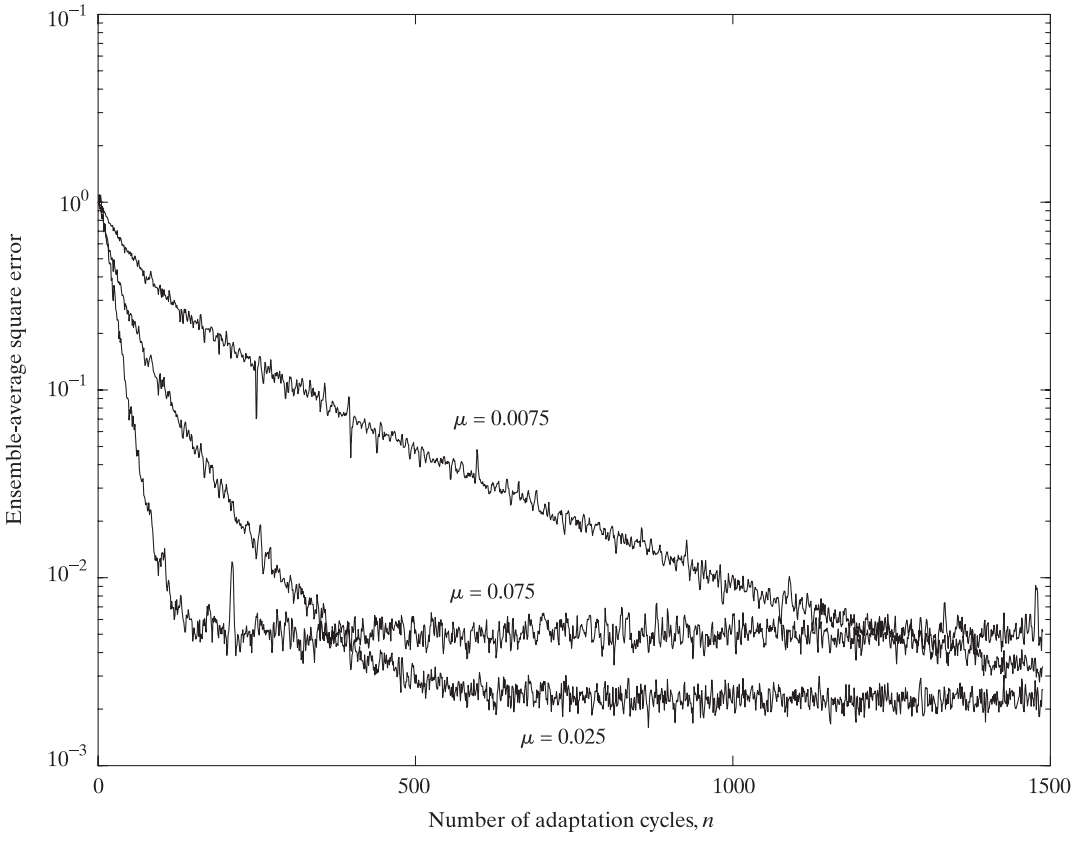


FIGURE 6.23 Learning curves of the LMS algorithm for an adaptive equalizer with the number of taps $M = 11$, fixed eigenvalue spread, and varying step-size parameter μ.

cycles. On the other hand, when μ was small (equal to 0.0075), the rate of convergence slowed down by more than an order of magnitude. The results also show that the steady-state value of the average squared error (and hence the misadjustment) increases with increasing μ.

Experiment 3: Validation of the Statistical Learning Theory. As with the computer experiment on adaptive prediction, we divide this third experiment on the validation of LMS statistical learning for the adaptive equalization into two parts.

Random-Walk Behavior. For this part of the experiment, we focus attention on quasi-steady-state response of the LMS algorithm, wherein a single sample function of the tap-weight vector, estimated by the LMS algorithm, performs a random behavior around the vectorized Wiener solution. With the tap-weight vector, $\hat{\mathbf{w}}(n)$, consisting of 11 elements, we have 11 such random behaviors, as shown plotted in Fig. 6.24. Table 6.3 summarizes the time-averaged deviation of the mean value of each estimated tap weight from its actual value and the corresponding time-averaged variance of each one of these plots; they were computed using 100 Monte Carlo runs. Based on these results, we may characterize the 11 random behaviors depicted in Fig. 6.24 as random walks. Moreover, assuming the stochastic force driving the random behaviors to be Gaussian distributed, we may describe each random behavior in Fig. 6.24 as an example of Brownian motion.

Agreement between Theory and Experiment. Applying the second line of Eq. (6.98) to the adaptive equalizer described in this section, we get the theoretic continuous curves plotted in Fig. 6.25 for the eigenvalue spread $W = 3.3$ and three different values of the step-size parameter, namely, $\mu = 0.0075, 0.025$, and 0.075. The corresponding experimental plots are also presented in parts (a), (b), and (c) of the figure, respectively; this second set of results was obtained using 400 Monte Carlo runs.

In light of the results presented in Fig. 6.25, we may make the following three observations:

1. For $\mu = 0.0075$, the experimental learning curve follows the theoretically derived learning curve very closely.
2. For $\mu = 0.025$, the agreement between theory and experiment remains good.
3. For $\mu = 0.075$, the transient part of the theoretic learning curve follows a good part of its experimental counterpart reasonably closely up to about 200 adaptation cycles, but thereafter, the theory fails significantly in computing the quasi-steady-state response of the algorithm.

Based on these results, we may go on to make two insightful conclusions:

1. When the step-size parameter, μ, is small, there is good agreement between the LMS statistical learning theory derived in this chapter and experiments performed using Monte Carlo simulations.

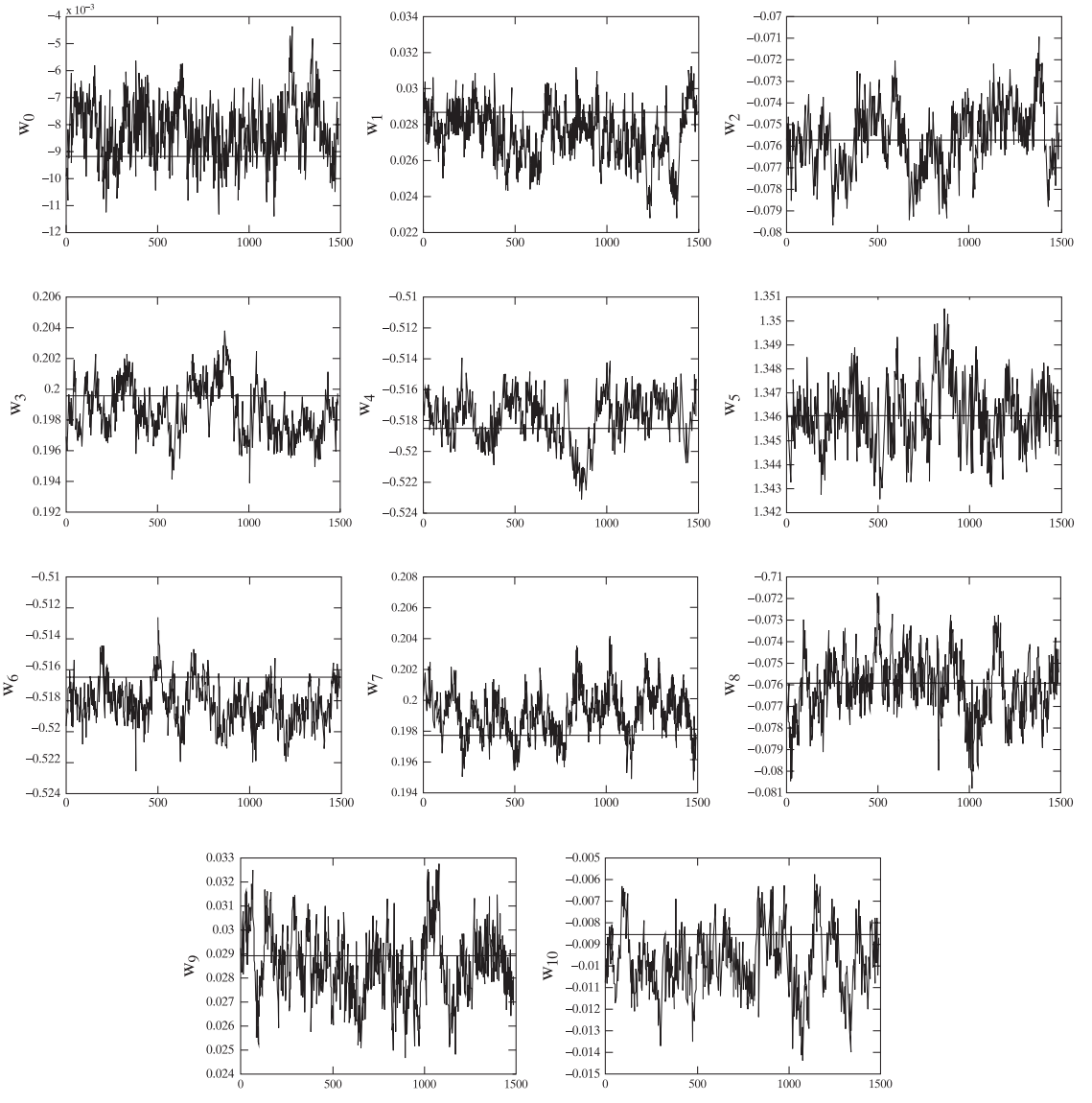


FIGURE 6.24 Eleven plots, displaying the random behavior of each tap weight, $\hat{w}_k(n)$, estimated by the LMS algorithm for $k = 0, 1, \ldots, 10$. The vertical axis in each plot describes the deviation of the estimated relevant weight in the LMS algorithm from its own Wiener solution. The horizontal axis of each plot indicates the number of adaptation cycles, n. [Note: The number of adaptation cycles is measured from when steady-state conditions (i.e., $n = 150$) are established.]

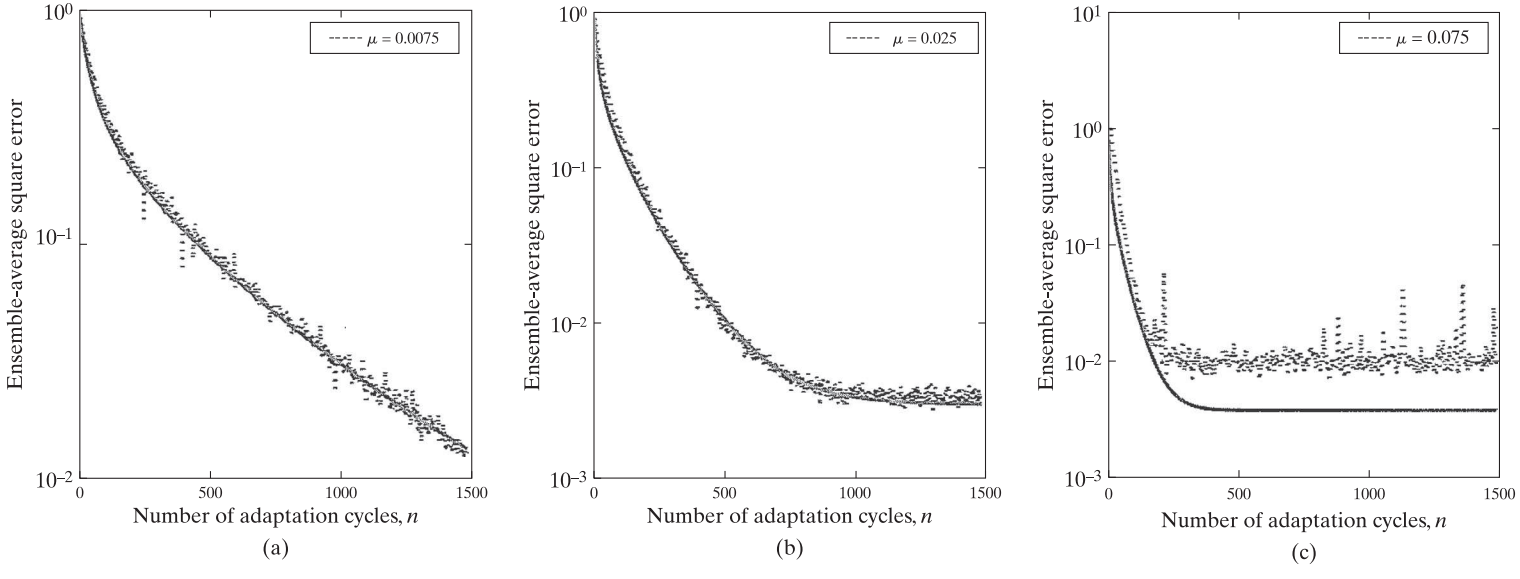


FIGURE 6.25 Comparative evaluation of the LMS statistical learning theory using Monte Carlo simulations for the eigenvolume spread $W = 3.3$ on three different values of the step-size parameter μ.

TABLE 6.3 Second-Order Statistics of Deviations in the 11 Tap Weights Estimated by the LMS Algorithm

	Mean	Variation
$\mathbf{w}_{o0} - \hat{w}_0$	−0.0010	1.146×10^{-6}
$\mathbf{w}_{o1} - \hat{w}_1$	0.0012	2.068×10^{-6}
$\mathbf{w}_{o2} - \hat{w}_2$	−0.0001	2.134×10^{-6}
$\mathbf{w}_{o3} - \hat{w}_3$	0.0010	2.701×10^{-6}
$\mathbf{w}_{o4} - \hat{w}_4$	−0.0006	2.229×10^{-6}
$\mathbf{w}_{o5} - \hat{w}_5$	−0.0001	1.737×10^{-6}
$\mathbf{w}_{o6} - \hat{w}_6$	0.0017	1.878×10^{-6}
$\mathbf{w}_{o7} - \hat{w}_7$	−0.0014	2.138×10^{-6}
$\mathbf{w}_{o8} - \hat{w}_8$	0.0001	1.880×10^{-6}
$\mathbf{w}_{o9} - \hat{w}_9$	0.0004	1.934×10^{-6}
$\mathbf{w}_{o10} - \hat{w}_{10}$	0.0012	2.068×10^{-6}

Note. Each deviation is defined with respect to the actual value of the tap weight $w_k, k = 0, 1, \ldots, 10$.

2. When the step-size parameter, μ, is large but small enough to assure convergence, the learning curve of the LMS algorithm appears to exhibit two distinct modes:[7]
   (a) *Initial mode*, which follows the statistical learning theory described in this chapter.
   (b) *Later mode*, which converges faster towards the quasi-steady-state condition of the algorithm.

6.9 COMPUTER EXPERIMENT ON A MINIMUM-VARIANCE DISTORTIONLESS-RESPONSE BEAMFORMER

For our final experiment, we consider the LMS algorithm applied to an adaptive minimum-variance distortionless-response (MVDR) beamformer consisting of a linear array of five uniformly spaced sensors (e.g., antenna elements), as depicted in Fig. 6.26. The spacing d between adjacent elements of the array equals one-half of the received wavelength, so as to avoid the appearance of grating lobes. The beamformer operates in an environment that consists of two components: a target signal impinging on the array along a direction of interest and a single source of interference coming from an unknown direction. It is assumed that these two components originate from independent sources and that the received signal includes additive white Gaussian noise at the output of each sensor.

The aims of the experiment are twofold:

- Examine the evolution with time of the adapted spatial response (pattern) of the MVDR beamformer for a prescribed target signal-to-interference ratio.

[7]A similar observation has been reported in Nascimento and Sayed (2000), where the second mode is explained by *almost sure convergence analysis*—that is, analysis based on the criterion that the LMS algorithm will converge with probability one.

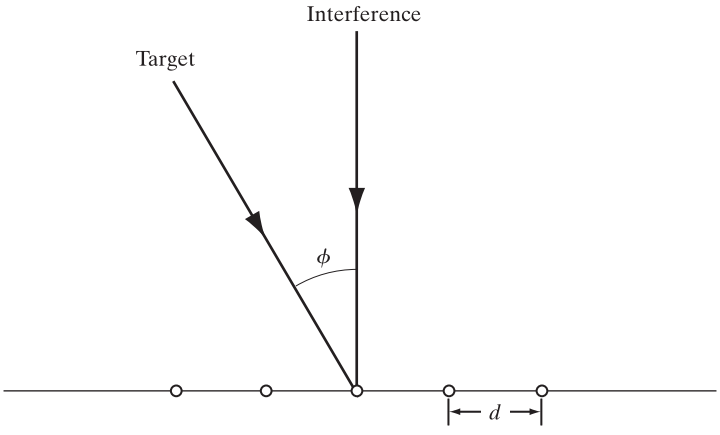


FIGURE 6.26 Linear array antenna.

- Evaluate the effect of varying the target-to-interference ratio on the interference-nulling performance of the beamformer.

The angles of incidence of the target and interfering signals, measured in radians with respect to the normal to the line of the array, are as follows:

- *Target signal:*

$$\phi_{\text{target}} = \sin^{-1}(-0.2).$$

- *Interference:*

$$\phi_{\text{interf}} = \sin^{-1}(0).$$

The design of the LMS algorithm for adjusting the weight vector of the adaptive MVDR beamformer follows the theory presented in Section 1.8. For the application at hand, the gain vector $\mathbf{g} = 1$.

Figure 6.27 shows the adapted spatial response of the MVDR beamformer for signal-to-noise ratio of 10 dB, varying interference-to-noise ratio (INR), and varying number of adaptation cycles. The spatial response is defined by $20 \log_{10} |\hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{s}(\theta)|^2$, where

$$\mathbf{s}(\theta) = [1, e^{-\mathrm{j}\theta}, e^{-\mathrm{j}2\theta}, e^{-\mathrm{j}3\theta}, e^{-\mathrm{j}4\theta}]^{\mathrm{T}}$$

is the steering vector. The electrical angle θ, measured in radians, is related to the angle of incidence ϕ by

$$\theta = \pi \sin \phi. \tag{6.115}$$

The weight vector $\hat{\mathbf{w}}(n)$ of the beamformer is computed with the use of the LMS algorithm with step-size parameter $\mu = 10^{-8}$, 10^{-9}, and 10^{-10} for INR = 20, 30, and

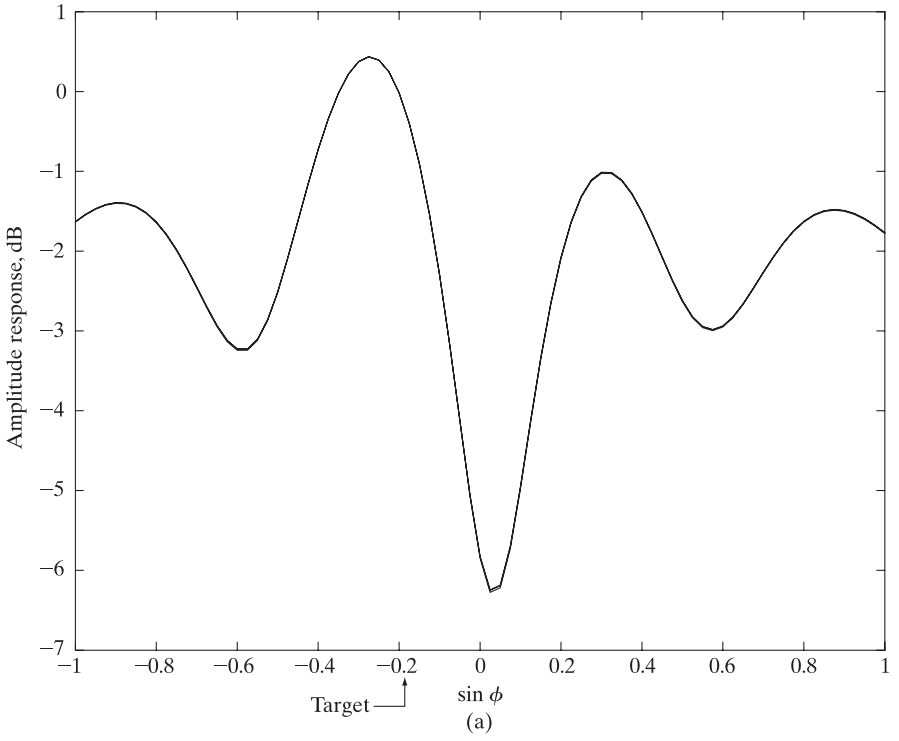


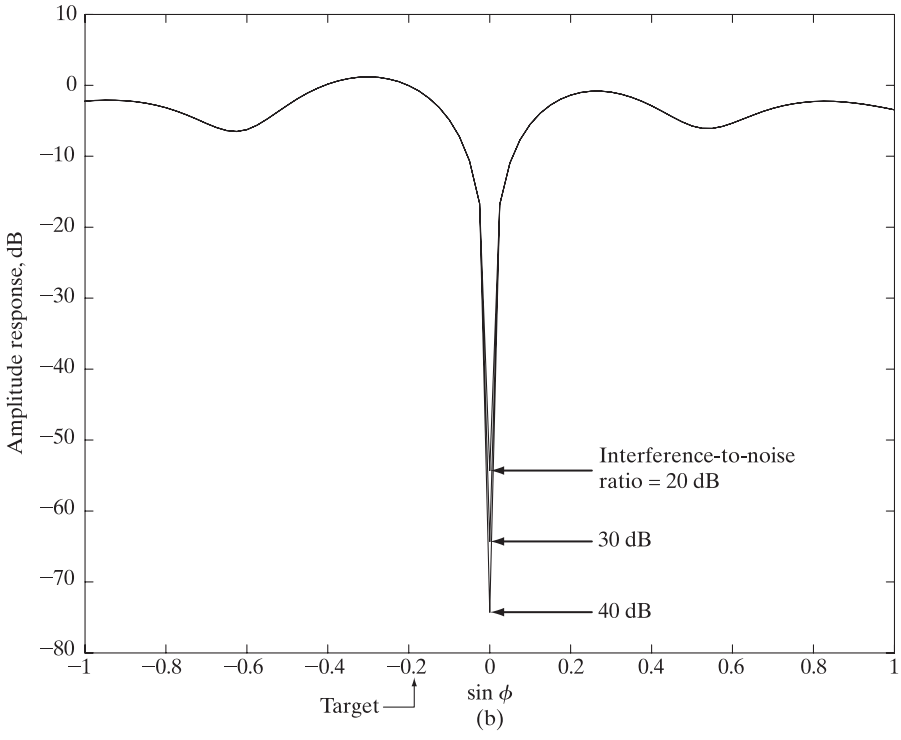
FIGURE 6.27 Adapted spatial response of MVDR beamformer for varying interference-to-noise ratio and varying number of adaptation cycles: (a) $n = 20$; (b) $n = 100$; (c) $n = 200$. In each part of the figure, the interference-to-noise ratio assumes one of three values; in part (a), the number of adaptation cycles is too small for these variations to have a noticeable effect. Parts (b) and (c) are shown on the next page.

40 dB, respectively. The reason for varying μ is to ensure convergence for a prescribed interference-to-noise ratio, as the largest eigenvalue λ_{max} of the correlation matrix of the input data depends on that ratio.

Figure 6.28 shows the adapted spatial response of the MVDR beamformer after 20, 25, and 30 adaptation cycles. The three curves of the figure pertain to INR = 20 dB and a fixed target signal-to-noise ratio of 10 dB.

On the basis of the results presented in Figs. 6.27 and 6.28, we may make the following observations:

- The spatial response of the MVDR beamformer is always held fixed at 0 dB along the prescribed angle of incidence $\phi_{target} = \sin^{-1}(-0.2)$, as required.
- The interference-nulling capability of the beamformer improves with (a) an increasing number of adaptation cycles (snapshots of data) and (b) an increasing interference-to-target signal ratio.



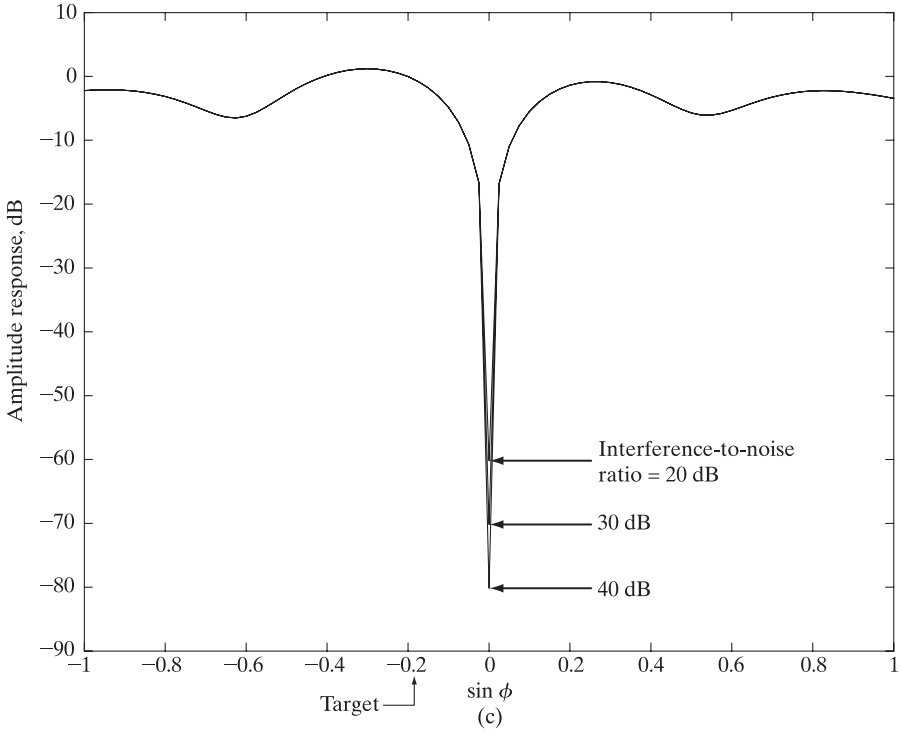


FIGURE 6.27 (*continued*)

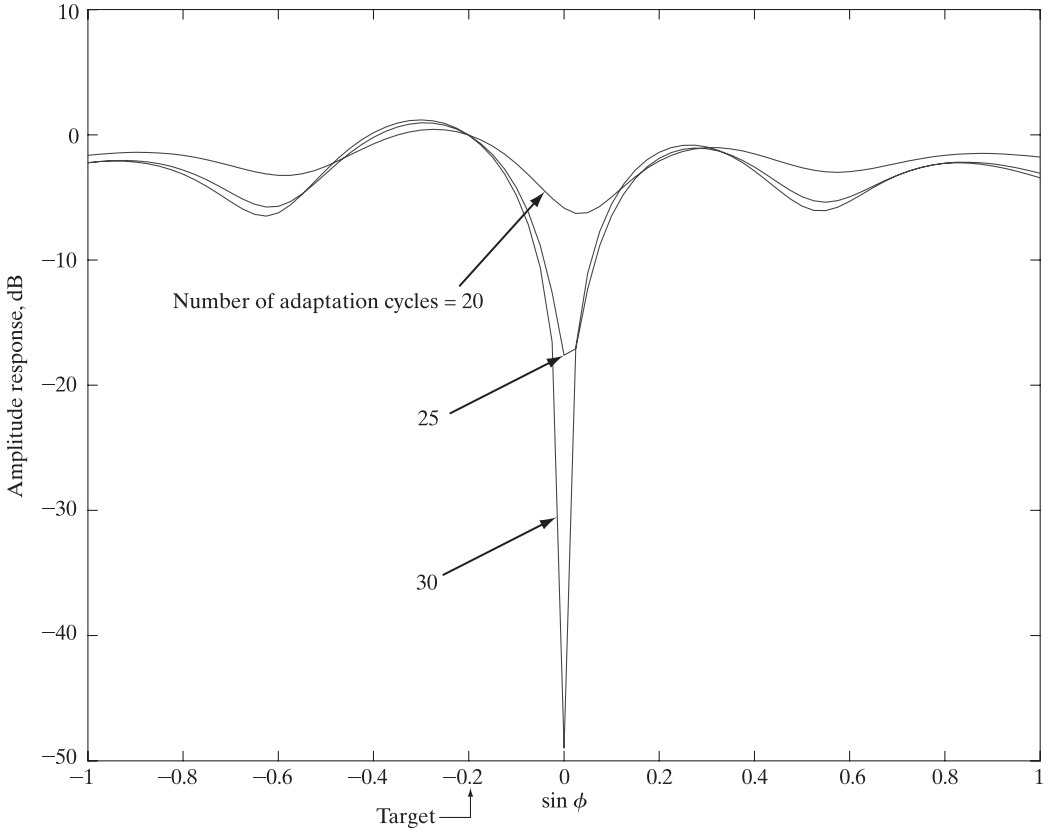


FIGURE 6.28 Adapted spatial response of MVDR beamformer for signal-to-noise ratio = 10 dB, interference-to-noise ratio = 20 dB, step-size parameter = 10^{-8}, and varying number of adaptation cycles.

6.10 SUMMARY AND DISCUSSION

In the first part of the chapter, we discussed optimality-related issues in two contexts: in a localized manner, wherein the LMS algorithm exhibits optimality in a certain Euclidean sense, and suboptimality compared with the Wiener solution in a quasi-steady state, where the suboptimality is attributed to gradient noise. In the second part, we followed up with applications of the LMS algorithm in equalization, deconvolution of seismic data, instantaneous frequency measurement, noise cancellation, line enhancement, and adaptive beamforming.

Highlights of the Small Step-Size Theory

Much of the second part of this chapter was devoted to the small step-size theory of LMS algorithms, the derivation of which builds on the Wiener filter theory of Chapter 2, providing a principled approach for evaluating the transient as well as the steady-state response of LMS algorithms under one of two plausible scenarios:

Scenario 1. The statistical dependence of the desired response $d(n)$ on the tap-input vector $\mathbf{u}(n)$ is governed by a multiple linear regression model. The Wiener filter,

used to compute the unknown parameter vector of the model, has a length equal to the model order. No other assumptions are made on the statistical characterization of the environment.

Scenario 2. The statistical dependence of $d(n)$ on $\mathbf{u}(n)$ is linear, but arbitrary. The desired response $d(n)$ and tap-input vector $\mathbf{u}$ are jointly Gaussian. No other assumptions are made on the statistical characterization of the environment.

The net result of the small step-size theory is that the ensemble-average learning curves of LMS algorithms for a given problem of interest exhibit a *deterministic* behavior in accordance with Eq. (6.98). Moreover, comparing that equation with Eq. (4.28), we see clearly that the ensemble-average learning curve of the LMS algorithm deviates from that of the steepest-descent algorithm operating on the corresponding Wiener filter, due to the presence of a stochastic driving force.

Another elegant feature of the small step-size theory is that it provides an insightful link of the stochastic behavior of LMS algorithms to Brownian motion, whose mathematical formulation is described by the Langevin equation.

Comparison with Independence Theory

Most importantly, the small step-size theory avoids shortcomings of the independence theory that are rooted in the statistical literature on the LMS algorithm (Widrow et al., 1976; Mazo, 1979; Gardner, 1984). The *independence theory* of the LMS algorithm makes the following assumptions:

- The tap-input vectors $\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(n)$ constitute a sequence of statistically independent vectors.
- At adaptation cycle n, the tap-input vector $\mathbf{u}(n)$ is statistically independent of all previous samples of the desired response, namely, $d(1), d(2), \ldots, d(n-1)$.
- Also at adaptation cycle n, the desired response $d(n)$ is dependent on the corresponding tap-input vector $\mathbf{u}(n)$, but it is statistically independent of all previous samples of the desired response.

The independence theory may be justified in certain applications, such as adaptive beamforming, where it is possible for successive snapshots of data (i.e., input vectors) received by an array of antenna elements from the surrounding environment to be independent of each other. However, for adaptive filtering applications in communications (e.g., signal prediction, channel equalization, and echo cancellation), the input vectors that direct the "hunting" of the weight vector toward the optimum Wiener solution are in fact statistically dependent. This dependence arises because of the *shifting property* of the input data. Specifically, the tap-input vector at adaptation cycle n is

$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^{\mathrm{T}}.$$

At adaptation cycle $n+1$, the vector takes on the new value

$$\mathbf{u}(n+1) = [u(n+1), u(n), \ldots, u(n-M+2)]^{\mathrm{T}}.$$

Thus, with the arrival of the new sample $u(n+1)$, the oldest sample $u(n-M+1)$ is discarded from $\mathbf{u}(n)$, and the remaining samples $u(n), u(n-1), \ldots, u(n-M+2)$ are shifted back in time by one unit to make room for the new sample $u(n+1)$. We see, therefore, that in a temporal setting, the tap-input vectors, and correspondingly the gradient directions computed by the LMS algorithm, are indeed *statistically dependent*.

The independence theory leads to conclusions about the transient as well as the steady-state response of the LMS algorithm that are similar to conclusions based on the small step-size theory. Nevertheless, the small step-size theory is the preferred approach to the statistical analysis of the LMS algorithm for three reasons: (1) It is grounded in principle, (2) it is insightful, and (3) it is easy to apply.

One other comment is in order. In deriving the small step-size theory, we ignored the higher-order terms $\boldsymbol{\varepsilon}_1(n), \boldsymbol{\varepsilon}_2(n), \ldots$, in the expansion of the weight-error vector $\boldsymbol{\varepsilon}(n)$ given in Eq. (6.59). However, when we use a large step size to accelerate the rate of convergence, the contributions of these higher-order terms to the statistical analysis of the LMS algorithm become significant and would therefore have to be included. The higher-order terms $\boldsymbol{\varepsilon}_1(n), \boldsymbol{\varepsilon}_2(n), \ldots$, have a stochastic nature, reflecting the increased noisy character of the learning curves of LMS algorithms with increasing step size. However, the inclusion of these higher-order terms makes statistical analysis of the LMS algorithm mathematically unmanageable.

PROBLEMS

1. What are the equations that define the operation of the LMS algorithm of the canonical model of the complex LMS algorithm?

2. Set up the equations that define the operation of the LMS algorithm that is used to implement adaptive noise cancelling applied to a sinusoidal interference.

3. Demonstrate that the LMS algorithm acts as a low-pass filter with a low cutoff frequency when the step-size parameter μ is small.

4. The zero-mean output $d(n)$ of an unknown real-valued system is represented by the *multiple linear regression model*

$$d(n) = \mathbf{w}_o^{\mathrm{T}}\mathbf{u}(n) + \nu(n),$$

where $\mathbf{w}_o$ is the (unknown) parameter vector of the model, $\mathbf{u}(n)$ is the input vector (regressor), and $\nu(n)$ is the sample value of an immeasurable white-noise process of zero mean and variance σ_ν^2. The block diagram of Fig. P6.1 shows the adaptive modeling of the unknown system, in which the adaptive FIR filter is controlled by a *modified* version of the LMS algorithm. In particular, the tap-weight vector $\mathbf{w}(n)$ of the FIR filter is chosen so as to minimize the index of performance

$$J(\mathbf{w}, K) = \mathbb{E}[e^{2K}(n)]$$

for $K = 1, 2, 3, \ldots$.

(a) By using the instantaneous gradient vector, show that the new adaptation rule for the corresponding estimate of the tap-weight vector is

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu K\mathbf{u}(n)e^{2K-1}(n),$$

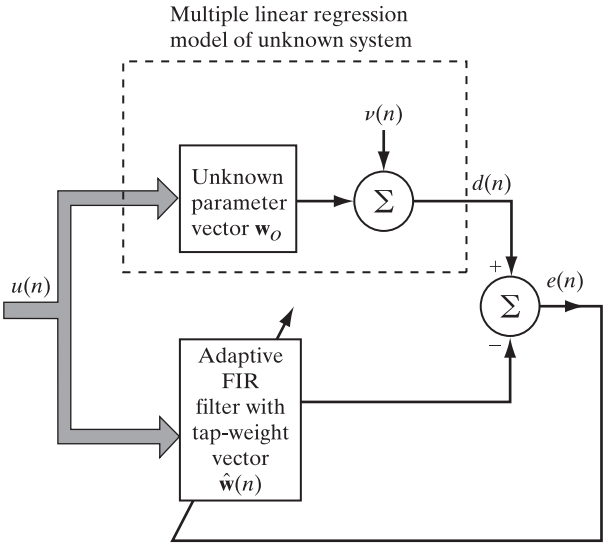


FIGURE P6.1

where μ is the step-size parameter and

$$e(n) = d(n) - \mathbf{w}^{\mathrm{T}}(n)\mathbf{u}(n)$$

is the estimation error.

(b) Assume that the weight-error vector

$$\boldsymbol{\varepsilon}(n) = \mathbf{w}_o - \hat{\mathbf{w}}(n)$$

is close to zero and that $\nu(n)$ is independent of $\mathbf{u}(n)$. Show that

$$\mathbb{E}[\boldsymbol{\varepsilon}(n+1)] = (\mathbf{I} - \mu K(2K-1)\mathbb{E}[\nu^{2K-2}(n)]\mathbf{R})\mathbb{E}[\boldsymbol{\varepsilon}(n)],$$

where $\mathbf{R}$ is the correlation matrix of the input vector $\mathbf{u}(n)$.

(c) Show that the modified LMS algorithm described in part (a) converges in the mean value if the step-size parameter μ satisfies the condition

$$0 < \mu < \frac{2}{K(2K-1)\mathbb{E}[\nu^{2(K-1)}(n)]\lambda_{\max}},$$

where $\lambda_{\max}$ is the largest eigenvalue of matrix $\mathbf{R}$.

(d) For $K = 1$, show that the results given in parts (a), (b), and (c) reduce to those of the traditional LMS algorithm.

5. (a) Let $\mathbf{m}(n)$ denote the *mean weight vector* in the LMS algorithm at adaptation cycle n; that is,

$$\mathbf{m}(n) = \mathbb{E}[\hat{\mathbf{w}}(n)].$$

Using the small step-size theory of Section 6.4, show that

$$\mathbf{m}(n) = (\mathbf{I} - \mu\mathbf{R})^n[\mathbf{m}(0) - \mathbf{m}(\infty)] + \mathbf{m}(\infty),$$

where μ is the step-size parameter, $\mathbf{R}$ is the correlation matrix of the input vector, and $\mathbf{m}(0)$ and $\mathbf{m}(\infty)$ are the initial and final values, respectively, of the mean weight vector.

(b) Show that, for convergence of the mean value $\mathbf{m}(n)$, the step-size parameter μ must satisfy the condition

$$0 < \mu < \frac{2}{\lambda_{\max}},$$

where $\lambda_{\max}$ is the largest eigenvalue of the correlation matrix $\mathbf{R}$.

6. Show that the physical mechanism for generating observable data is described by a multiple linear regression model that is matched exactly by the Wiener filter.

7. Consider the use of a white-noise sequence of zero mean and variance σ^2 as the input to the LMS algorithm. Evaluate
(a) the condition for convergence of the algorithm in the mean square, and
(b) the excess mean-square error.

8. For this set of specifications, study the following two different scenarios:

$$u_a(n) = \cos(1.2n) + 0.5\cos(0.1n),$$
$$u_b(n) = \cos(0.6n) + 0.5\cos(0.23n).$$

The first input, $u_a(n)$, has an eigenvalue spread $\chi(\mathbf{R}) = 2.9$, and the second input, $u_b(n)$, has an eigenvalue spread $\chi(\mathbf{R}) = 12.9$. Confirm both of these two spreads.

There are four distinct combinations to be considered:

Case 1: Minimum eigenvalue, for which we have

$$\mathbf{w}_o = \text{optimum tap-weight vector of the Wiener filter} = [-1, 1]^{\mathrm{T}}.$$

Show that $\mathbf{w}_o = \mathbf{q}_2$, where $\mathbf{q}_2$ is the eigenvector associated with the eigenvalue λ_2. Demonstrate the following:
(a) For the input $u_a(n)$, convergence of the LMS algorithm lies along the "slow" trajectory.
(b) For the input $u_b(n)$, convergence of the LMS algorithm is *decelerated* compared to part (a).
Illustrate both results diagrammatically.

Case 2: Maximum eigenvalue, for which we now have

$$\mathbf{w}_o = [1, 1]^{\mathrm{T}}.$$

Show that $\mathbf{w}_o = \mathbf{q}_1$, where $\mathbf{q}_1$ is the eigenvector associated with the eigenvalue λ_1. Demonstrate the following:
(a) For the input $u_a(n)$, convergence of the LMS algorithm lies along a "fast" trajectory.
(b) For the input $u_b(n)$, convergence of the LMS algorithm is *accelerated* compared to (a).
Again, illustrate your results diagrammatically.

9. Consider the operation of an adaptive line enhancer using the LMS algorithm under a low signal-to-noise ratio. The correlation matrix of the input vector is defined by

$$\mathbf{R} = \sigma^2\mathbf{I},$$

where $\mathbf{I}$ is the identity matrix. Show that the steady-state value of the weight-error correlation matrix $\mathbf{K}(n)$ is given by

$$\mathbf{K}(\infty) \cong \frac{\mu}{2} J_{\min}\mathbf{I},$$

where μ is the step-size parameter and $J_{\min}$ is the minimum mean-square error. You may assume that the number of taps in the adaptive FIR filter is large.

10. Starting with Eq. (6.58) for small step sizes, show that under steady-state conditions

$$\mathbf{R}\mathbf{K}_0(n) + \mathbf{K}_0(n)\mathbf{R} = \mu \sum_{l=0}^{\infty} J_{\min}^{(l)} \mathbf{R}^l,$$

where

$$J_{\min}^{(l)} = \mathbb{E}[e_o(n)e_o^*(n-l)], \qquad l = 0, 1, 2, \ldots$$

and

$$\mathbf{R}^l = \mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n-l)], \qquad l = 0, 1, 2, \ldots.$$

[Because of the structure of the sum term $\mathbf{R}\mathbf{K}_0(n) + \mathbf{K}_0(n)\mathbf{R}$, the equation involving this sum is called the *Lyapunov equation*.]

11. How does localized optimality work better compared to LMS algorithm as an adaptive filter?

12. Using the small step-size theory of LMS algorithms presented in Section 6.4, do the following:

(a) Show that $\mathscr{D}(\infty)$ is independent of the input signal; that is,

$$\mathscr{D}(\infty) = \frac{1}{2}\mu M J_{\min}.$$

(b) Derive the formula for the misadjustment

$$\mathscr{M} = \frac{\mu}{2}\,\mathrm{tr}[\mathbf{R}]$$

without having to go through the diagonalization of the correlation matrix $\mathbf{R}$.

13. **(a)** Show that the mean of the stochastic force vector $\phi(n)$ is zero.

(b) Show that the correlation matrix of the stochastic force vector $\phi(n)$ is a diagonal matrix.

14. The tandem connection of adaptive filters arises in some applications (e.g., acoustic echo cancellation). Consider, then, Fig. P6.2, which shows a simplified tandem configuration involving a pair of LMS adaptive filters (Ho, 2000). The input vector $\mathbf{u}(n)$ is applied simultaneously to both filters, and the error signal $e_1(n)$ produced by filter I serves the purpose of having a desired response for filter II.

(a) Formulate the update equations for the tandem configuration of the figure.

(b) Show that this tandem configuration converges in the mean square if both adaptive filters I and II converge in the mean square individually.

15. What does misadjustment refer to in the LMS algorithm? What is its significance?

Computer Experiments

16. In conducting a computer experiment that involves the generation of an AR process, sometimes not enough time is allowed for the transients to die out. The purpose of this experiment is to evaluate the effects of such transients on the operation of the LMS algorithm. Consider, then, the AR process $u(n)$ of order one described in Section 6.7. The parameters of this process are as follows:

AR parameter:	$a = -0.99$;
AR process variance:	$\sigma_u^2 = 0.936$;
Noise variance:	$\sigma_v^2 = 0.02$.

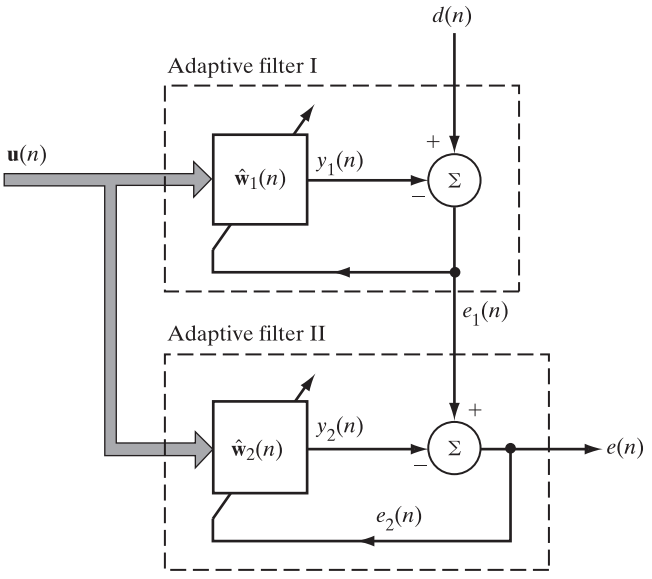


FIGURE P6.2

Generate the process $u(n)$ so described for $1 \le n \le 100$, assuming zero initial conditions. Use $u(n)$ as the input of a linear adaptive predictor that is based on the LMS algorithm together with a step-size parameter $\mu = 0.05$. In particular, plot the learning curve of the predictor by ensemble averaging over 100 independent realizations of the squared value of its output versus adaptation cycle n for $1 \le n \le 100$. Unlike the normal operation of the LMS algorithm, the learning curve so computed should start at the origin, rise to a peak, and then decay toward a steady-state value. Explain the reason for this phenomenon.

17. Consider an AR process $u(n)$ defined by the difference equation

$$u(n) = -a_1 u(n-1) - a_2 u(n-2) + \nu(n),$$

where $\nu(n)$ is an additive white noise of zero mean and variance σ_ν^2. The AR parameters a_1 and a_2 are both real valued:

$$a_1 = 0.1;$$
$$a_2 = -0.8.$$

(a) Calculate the noise variance σ_ν^2 such that the AR process $u(n)$ has unit variance. Hence, generate different realizations of the process $u(n)$.

(b) Given the input $u(n)$, an LMS algorithm of length $M = 2$ is used to estimate the unknown AR parameters a_1 and a_2. The step-size parameter μ is assigned the value 0.05. Justify the use of this design value in the application of the small step-size theory developed in Section 6.4.

(c) For one realization of the LMS algorithm, compute the prediction error

$$f(n) = u(n) - \hat{u}(n)$$

and the two tap-weight errors

$$\varepsilon_1(n) = -a_1 - \hat{w}_1(n)$$

and

$$\varepsilon_2(n) = -a_2 - \hat{w}_2(n).$$

Using power spectral plots of $f(n)$, $\varepsilon_1(n)$, and $\varepsilon_2(n)$, show that $f(n)$ behaves as white noise, whereas $\varepsilon_1(n)$ and $\varepsilon_2(n)$ behave as low-pass processes.

(d) Compute the ensemble-average learning curve of the LMS algorithm by averaging the squared value of the prediction error $f(n)$ over an ensemble of 100 different realizations of the filter.

(e) Using the small step-size statistical theory of Section 6.4, compute the theoretical learning curve of the LMS algorithm and compare your result against the measured result of part (d).

18. Consider a linear communication channel whose transfer function may take one of three possible forms:

(i) $H(z) = 0.25 + z^{-1} + 0.25z^{-2}$

(ii) $H(z) = 0.25 + z^{-1} - 0.25z^{-2}$

(iii) $H(z) = -0.25 + z^{-1} + 0.25z^{-2}$

The channel output, in response to the input x_n, is defined by

$$u(n) = \sum_k h_k x_{n-k} + \nu(n),$$

where h_n is the impulse response of the channel and $\nu(n)$ is additive white Gaussian noise with zero mean and variance $\sigma_\nu^2 = 0.01$. The channel input x_n consists of a Bernoulli sequence with $x_n = \pm 1$.

The purpose of the experiment is to design an adaptive equalizer trained by using the LMS algorithm with step-size parameter $\mu = 0.001$. In structural terms, the equalizer is built around an FIR filter with 21 taps. For desired response, a delayed version of the channel input, namely $x_{n-\Delta}$, is supplied to the equalizer. For each of the possible transfer functions listed under (i), (ii), and (iii), do the following:

(a) Determine the optimum value of the delay Δ that minimizes the mean-square error at the equalizer output.

(b) For the optimum delay Δ determined in part (a), plot the learning curve of the equalizer by averaging the squared value of the error signal over an ensemble of 100 independent Monte Carlo runs of the experiment.

19. In Section 6.4, we presented the small step-size statistical theory of LMS algorithms. The purpose of this section is to study what happens to the application of this theory when the assumption of a small step size is violated.

Repeat the experiment on the learning curve of the first-order adaptive predictor plotted in Fig. 6.18, but this time use the following values for the step-size parameter: 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3.

Comment on the results so obtained.

20. What is the analogy between the LMS algorithm (discrete time, n) and the Langevin equation (continuous time, t) in terms of stochastic force, damping force, and sample function?

21. The results presented in Fig. 6.25 for the LMS statistical learning theory for three different values of the step-size parameter, μ, were computed for the eigenvalue spread $W = 3.3$, experienced by the adaptive equalizer studies in Section 6.8.

(a) Repeat that experiment for the eigenvalue spread $W = 2.9, 3.1$, and 3.5.

(b) Comment on the results obtained in the context of the LMS statistical learning theory described in Section 6.4.

22. Repeat the computer experiment on MVDR beamforming described in Section 6.9 for a target signal-to-noise ratio of 10 dB, interference-to-noise ratio of 40 dB, and step-size parameter $\mu = 10^{-10}$. As before, the angle-of-arrival for the interference is

$$\phi_{\text{interf}} = \sin^{-1}(0).$$

This time, however, we investigate what happens to the spatial response of the beamformer as the target moves closer to the source of interference. Specifically, plot the spatial response for an increasing number of adaptation cycles for each of the following angles-of-arrival for the target:

(a) $\phi_{\text{target}} = \sin^{-1}(-0.15)$
(b) $\phi_{\text{target}} = \sin^{-1}(-0.10)$
(c) $\phi_{\text{target}} = \sin^{-1}(-0.05)$

Comment on your results.

CHAPTER 7

Normalized Least-Mean-Square (LMS) Algorithm and Its Generalization

In the traditional form of a least-mean-square (LMS) algorithm studied in Chapter 6, the *adjustment* applied to the tap-weight vector of the filter at adaptation cycle $n + 1$ consists of the product of three terms:

- The step-size parameter μ, which is under the designer's control.
- The tap-input vector $\mathbf{u}(n)$, which is supplied by a source of information.
- The estimation error $e(n)$ for real-valued data, or its complex conjugate $e^*(n)$ for complex-valued data, which is calculated at adaptation cycle n.

The adjustment is directly proportional to the tap-input vector $\mathbf{u}(n)$. Therefore, when $\mathbf{u}(n)$ is large, the LMS algorithm suffers from a *gradient noise amplification* problem. To overcome this difficulty, we may use the *normalized LMS* algorithm.[1] In particular, the adjustment applied to the tap-weight vector at adaptation cycle $n + 1$ is "normalized" with respect to the squared Euclidean norm of the tap-input vector $\mathbf{u}(n)$ at adaptation cycle n—hence the term "normalized."

This chapter is devoted to a discussion of the normalized LMS algorithm and its application to acoustic echo cancellation. The chapter also includes a discussion of the affine projection adaptive filter, which may be viewed as a generalization of the normalized LMS algorithm.

7.1 NORMALIZED LMS ALGORITHM: THE SOLUTION TO A CONSTRAINED OPTIMIZATION PROBLEM

In structural terms, the normalized LMS algorithm is exactly the same as the traditional LMS algorithm, as shown in the block diagram of Fig. 7.1. Both adaptive filtering algorithms are built around a finite-duration impulse response (FIR) filter, but differ only in the way in which the weight controller is mechanized. The M-by-1 tap-input vector $\mathbf{u}(n)$

[1]The stochastic-gradient-descent algorithm known as the normalized LMS algorithm was suggested independently by Nagumo and Noda (1967) and Albert and Gardner (1967). Nagumo and Noda did not use any special name for the algorithm, whereas Albert and Gardner referred to it as a "quick and dirty regression" scheme. It appears that Bitmead and Anderson (1980a, b) coined the name "normalized LMS algorithm."

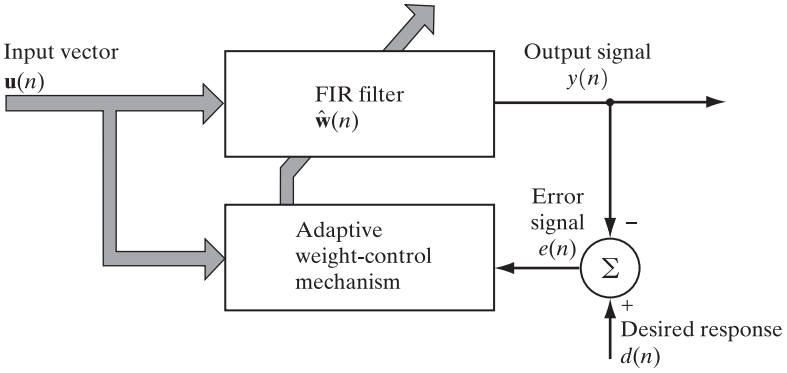


FIGURE 7.1 Block diagram of adaptive FIR filter.

produces an output $y(n)$ that is subtracted from the desired response $d(n)$ to produce the estimation error, or error signal, $e(n)$. In response to the combined action of the input vector $\mathbf{u}(n)$ and error signal $e(n)$, the weight controller applies a weight adjustment to the FIR filter. This sequence of events is repeated for a number of adaptation cycles until the filter reaches a steady state.

We may formulate the normalized LMS algorithm as a natural modification of the traditional LMS algorithm. (See Problem 1.) Alternatively, we may derive the normalized LMS algorithm in its own right; we follow the latter procedure here, as it provides insight into the operation of the filter.

The normalized LMS algorithm is a manifestation of the *principle of minimal disturbance*, which may be stated as follows:

> From one adaptation cycle to the next, the weight vector of an adaptive filter should be changed in a minimal manner, subject to a constraint imposed on the updated filter's output.

To cast this principle in mathematical terms, let $\hat{\mathbf{w}}(n)$ denote the old weight vector of the filter at adaptation cycle n and $\hat{\mathbf{w}}(n+1)$ denote its updated weight vector at adaptation cycle $n+1$. We may then formulate the criterion for designing the normalized LMS algorithm as that of constrained optimization: Given the tap-input vector $\mathbf{u}(n)$ and desired response $d(n)$, determine the updated tap-weight vector $\hat{\mathbf{w}}(n+1)$ so as to minimize the squared Euclidean norm of the change,

$$\delta\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n), \tag{7.1}$$

subject to the constraint

$$\hat{\mathbf{w}}^{\mathrm{H}}(n+1)\mathbf{u}(n) = d(n), \tag{7.2}$$

where the superscript H denotes Hermitian transposition (i.e., the operation of transposition combined with complex conjugation).

To solve this constrained optimization problem, we use the *method of Lagrange multipliers*, which is described in Appendix C for the general case of complex-valued

data. According to this method, the cost function for the problem at hand consists of two terms, given on the right-hand side of the equation

$$J(n) = \|\delta\hat{\mathbf{w}}(n+1)\|^2 + \mathrm{Re}[\lambda^*(d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n+1)\mathbf{u}(n))], \tag{7.3}$$

where λ is the complex-valued *Lagrange multiplier* and the asterisk denotes complex conjugation. The squared Euclidean norm $\|\delta\hat{\mathbf{w}}(n+1)\|^2$ is, naturally, real valued. The real-part operator, denoted by Re $[\cdot]$ and applied to the second term, ensures that the contribution of the constraint to the cost function is likewise real valued. The cost function $J(n)$ is a quadratic function in $\hat{\mathbf{w}}(n+1)$, as is shown by expanding Eq. (7.3) into

$$J(n) = (\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n))^{\mathrm{H}}(\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)) + \mathrm{Re}[\lambda^*(d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n+1)\mathbf{u}(n))]. \tag{7.4}$$

To find the optimum value of the updated weight vector that minimizes the cost function $J(n)$, we proceed as follows:

1. Differentiate the cost function $J(n)$ with respect to $\hat{\mathbf{w}}^{\mathrm{H}}(n+1)$. Then, following the Wirtinger calculus for differentiating a real-valued function with respect to a complex-valued weight vector and formally treating $\hat{\mathbf{w}}(n)$ as a constant, as in Appendix B, we get

$$\frac{\partial J(n)}{\partial \hat{\mathbf{w}}^{\mathrm{H}}(n+1)} = 2(\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)) - \lambda^*\mathbf{u}(n).$$

Setting this result equal to zero and solving for the optimum value $\hat{\mathbf{w}}(n+1)$, we obtain

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{1}{2}\lambda^*\mathbf{u}(n). \tag{7.5}$$

2. Solve for the unknown multiplier λ by substituting the result of step 1 [i.e., the weight vector $\hat{\mathbf{w}}(n+1)$] into the constraint of Eq. (7.2). Doing the substitution, we first write

$$\begin{aligned} d(n) &= \hat{\mathbf{w}}^{\mathrm{H}}(n+1)\mathbf{u}(n) \\ &= \left(\hat{\mathbf{w}}(n) + \frac{1}{2}\lambda^*\mathbf{u}(n)\right)^{\mathrm{H}}\mathbf{u}(n) \\ &= \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n) + \frac{1}{2}\lambda\mathbf{u}^{\mathrm{H}}(n)\mathbf{u}(n) \\ &= \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n) + \frac{1}{2}\lambda\|\mathbf{u}(n)\|^2. \end{aligned}$$

Then, solving for λ, we obtain

$$\lambda = \frac{2e(n)}{\|\mathbf{u}(n)\|^2}, \tag{7.6}$$

where

$$e(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n) \tag{7.7}$$

is the error signal.

3. Combine the results of steps 1 and 2 to formulate the optimal value of the incremental change, $\delta\hat{\mathbf{w}}(n+1)$. Specifically, from Eqs. (7.5) and (7.6), we have

$$\begin{aligned}\delta\hat{\mathbf{w}}(n+1) &= \hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)\\ &= \frac{1}{\|\mathbf{u}(n)\|^2}\,\mathbf{u}(n)e^*(n).\end{aligned} \tag{7.8}$$

In order to exercise control over the change in the tap-weight vector from one adaptation cycle to the next without changing the direction of the vector, we introduce a positive real scaling factor denoted by $\tilde{\mu}$. That is, we redefine the change simply as

$$\begin{aligned}\delta\hat{\mathbf{w}}(n+1) &= \hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)\\ &= \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}\,\mathbf{u}(n)e^*(n).\end{aligned} \tag{7.9}$$

Equivalently, we write

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}\,\mathbf{u}(n)e^*(n). \tag{7.10}$$

Indeed, this is the desired recursion for computing the M-by-1 tap-weight vector in the normalized LMS algorithm. Equation (7.10) clearly shows the reason for using the term "normalized": The product vector $\mathbf{u}(n)e^*(n)$ is normalized with respect to the squared Euclidean norm of the tap-input vector $\mathbf{u}(n)$.

Comparing the recursion of Eq. (7.10) for the normalized LMS algorithm with that of Eq. (5.6) for the traditional LMS algorithm, we may make the following observations:

- The adaptation constant $\tilde{\mu}$ for the normalized LMS algorithm is *dimensionless*, whereas the adaptation constant μ for the LMS algorithm has the dimensions of *inverse power*.
- Setting

$$\mu(n) = \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}, \tag{7.11}$$

  we may view the normalized LMS algorithm as an LMS algorithm with a *time-varying step-size parameter*.
- Most importantly, the normalized LMS algorithm exhibits a rate of convergence that is potentially faster than that of the traditional LMS algorithm for both uncorrelated and correlated input data (Nagumo & Noda, 1967; Douglas & Meng, 1994).

An issue of possible concern is that, in overcoming the gradient noise amplification problem associated with the traditional LMS algorithm, the normalized LMS algorithm introduces a problem of its own, namely, that when the tap-input vector $\mathbf{u}(n)$ is small, numerical difficulties may arise because then we have to divide by a small value for the

TABLE 7.1 Summary of the Normalized LMS Algorithm

Parameters: M = number of taps (i.e., filter length)

$\tilde{\mu}$ = adaptation constant

$$0 < \tilde{\mu} < 2\,\frac{\mathbb{E}[|u(n)|^2]\mathcal{D}(n)}{\mathbb{E}[|e(n)|^2]},$$

where

$\mathbb{E}[|e(n)|^2]$ = error signal power,
$\mathbb{E}[|u(n)|^2]$ = input signal power,
$\mathcal{D}(n)$ = mean-square deviation.

Initialization. If prior knowledge about the tap-weight vector $\hat{\mathbf{w}}(n)$ is available, use that knowledge to select an appropriate value for $\hat{\mathbf{w}}(0)$. Otherwise, set $\hat{\mathbf{w}}(0) = \mathbf{0}$.

Data

(a) Given: $\mathbf{u}(n)$ = M-by-1 tap input vector at time n.
$d(n)$ = desired response at time step n.

(b) To be computed: $\hat{\mathbf{w}}(n+1)$ = estimate of tap-weight vector at time step n + 1.

Computation: For $n = 0, 1, 2, \ldots$, compute

$$e(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n),$$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}\,\mathbf{u}(n)e^*(n).$$

squared norm $\|\mathbf{u}(n)\|^2$. To overcome this problem, we modify the recursion of Eq. (7.10) slightly to produce

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\delta + \|\mathbf{u}(n)\|^2}\,\mathbf{u}(n)e^*(n), \tag{7.12}$$

where $\delta > 0$. For $\delta = 0$, Eq. (7.12) reduces to the form given in Eq. (7.10).

The normalized LMS algorithm, based on Eq. (7.10), is summarized in Table 7.1. The upper bound on the normalized step-size parameter $\tilde{\mu}$ presented in the table is derived in the next section.

7.2 STABILITY OF THE NORMALIZED LMS ALGORITHM

Suppose that the physical mechanism responsible for generating the desired response $d(n)$ is governed by the multiple regression model, which is described by

$$d(n) = \mathbf{w}^{\mathrm{H}}\mathbf{u}(n) + v(n). \tag{7.13}$$

In this equation, $\mathbf{w}$ is the model's unknown parameter vector and $v(n)$ is the additive disturbance. The tap-weight vector $\hat{\mathbf{w}}(n)$ computed by the normalized LMS algorithm is an estimate of $\mathbf{w}$. The mismatch between $\mathbf{w}$ and $\hat{\mathbf{w}}(n)$ is measured by the *weight-error vector*

$$\boldsymbol{\varepsilon}(n) = \mathbf{w} - \hat{\mathbf{w}}(n).$$

Thus, subtracting Eq. (7.10) from $\mathbf{w}$, we get

$$\boldsymbol{\varepsilon}(n+1) = \boldsymbol{\varepsilon}(n) - \frac{\widetilde{\mu}}{\|\mathbf{u}(n)\|^2}\,\mathbf{u}(n)e^*(n). \tag{7.14}$$

As already stated, the underlying idea of a normalized LMS algorithm is that of minimizing the incremental change $\delta\hat{\mathbf{w}}(n+1)$ in the tap-weight vector of the filter from adaptation cycle n to adaptation cycle $n+1$, subject to a constraint imposed on the updated tap-weight vector $\hat{\mathbf{w}}(n+1)$. In light of this idea, it is logical that we base the stability analysis of the normalized LMS algorithm on the *mean-square deviation* [see Eq. (6.84)]

$$\mathcal{D}(n) = \mathbb{E}[\|\boldsymbol{\varepsilon}(n)\|^2]. \tag{7.15}$$

Taking the squared Euclidean norms of both sides of Eq. (7.14), rearranging terms, and then taking expectations, we get

$$\mathcal{D}(n+1) - \mathcal{D}(n) = \widetilde{\mu}^2\mathbb{E}\left[\frac{|e(n)|^2}{\|\mathbf{u}(n)\|^2}\right] - 2\widetilde{\mu}\,\mathbb{E}\left\{\mathrm{Re}\left[\frac{\xi_u(n)e^*(n)}{\|\mathbf{u}(n)\|^2}\right]\right\}, \tag{7.16}$$

where $\xi_u(n)$ is the *undisturbed error signal* defined by

$$\begin{aligned}\xi_u(n) &= (\mathbf{w} - \hat{\mathbf{w}}(n))^{\mathrm{H}}\mathbf{u}(n)\\ &= \boldsymbol{\varepsilon}^{\mathrm{H}}(n)\mathbf{u}(n).\end{aligned} \tag{7.17}$$

From Eq. (7.16), we readily see that the mean-square deviation $\mathcal{D}(n)$ decreases exponentially with increasing number of adaptation cycles n, and the normalized LMS algorithm is therefore *stable in the mean-square-error sense* (i.e., the convergence process is monotonic), provided that the normalized step-size parameter $\widetilde{\mu}$ is bounded as follows:

$$0 < \widetilde{\mu} < 2\,\frac{\mathrm{Re}\{\mathbb{E}[\xi_u(n)e^*(n)/\|\mathbf{u}(n)\|^2]\}}{\mathbb{E}[|e(n)|^2/\|\mathbf{u}(n)\|^2]}. \tag{7.18}$$

From Eq. (7.18), we also readily find that the largest value of the mean-square deviation $\mathcal{D}(n)$ is achieved at the midpoint of the interval defined therein. The optimal step-size parameter is therefore given by

$$\widetilde{\mu}_{\mathrm{opt}} = \frac{\mathrm{Re}\{\mathbb{E}[\xi_u(n)e^*(n)/\|\mathbf{u}(n)\|^2]\}}{\mathbb{E}[|e(n)|^2/\|\mathbf{u}(n)\|^2]}. \tag{7.19}$$

Special Environment: Real-Valued Data

For the case of real-valued data (e.g., in acoustic echo cancellation, considered in the next section), the normalized LMS algorithm of Eq. (7.10) takes the form

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\widetilde{\mu}}{\|\mathbf{u}(n)\|^2}\,\mathbf{u}(n)e(n). \tag{7.20}$$

Likewise, the optimal step-size parameter of Eq. (7.19) reduces to

$$\widetilde{\mu}_{\text{opt}} = \frac{\mathbb{E}[\xi_u(n)e(n)/\|\mathbf{u}(n)\|^2]}{\mathbb{E}[e^2(n)/\|\mathbf{u}(n)\|^2]}. \tag{7.21}$$

To make the computation of $\widetilde{\mu}_{\text{opt}}$ tractable, we now introduce three assumptions:

Assumption 1. *The fluctuations in the input signal energy $\|\mathbf{u}(n)\|^2$ from one adaptation cycle to the next are small enough to justify the approximations*

$$\mathbb{E}\left[\frac{\xi_u(n)e(n)}{\|\mathbf{u}(n)\|^2}\right] \approx \frac{\mathbb{E}[\xi_u(n)e(n)]}{\mathbb{E}[\|\mathbf{u}(n)\|^2]} \tag{7.22}$$

and

$$\mathbb{E}\left[\frac{e^2(n)}{\|\mathbf{u}(n)\|^2}\right] \approx \frac{\mathbb{E}[e^2(n)]}{\mathbb{E}[\|\mathbf{u}(n)\|^2]}. \tag{7.23}$$

Correspondingly, the formula of Eq. (7.21) approximates to

$$\widetilde{\mu}_{\text{opt}} \approx \frac{\mathbb{E}[\xi_u(n)e(n)]}{\mathbb{E}[e^2(n)]}. \tag{7.24}$$

Assumption 2. *The undisturbed error signal $\xi_u(n)$ is uncorrelated with the disturbance (noise) $v(n)$ in the multiple regression model for the desired response $d(n)$.*

The disturbed error signal $e(n)$ is related to the undisturbed error signal $\xi_u(n)$ by

$$e(n) = \xi_u(n) + v(n). \tag{7.25}$$

Using Eq. (7.25) and then invoking Assumption 2, we have

$$\begin{aligned}\mathbb{E}[\xi_u(n)e(n)] &= \mathbb{E}[\xi_u(n)(\xi_u(n) + v(n))] \\ &= \mathbb{E}[\xi_u^2(n)].\end{aligned} \tag{7.26}$$

Inserting Eq. (7.26) into Eq. (7.24), we may further simplify the formula for the optimal step size to

$$\widetilde{\mu}_{\text{opt}} \approx \frac{\mathbb{E}[\xi_u^2(n)]}{\mathbb{E}[e^2(n)]}. \tag{7.27}$$

Unlike the disturbed error signal $e(n)$, the undisturbed error signal $\xi_u(n)$ is inaccessible and, therefore, not directly measurable. To overcome this computational difficulty, we introduce one last assumption.

Assumption 3. *The spectral content of the input signal $u(n)$ is essentially flat over a frequency band larger than that occupied by each element of the weight-error vector $\boldsymbol{\varepsilon}(n)$, thereby justifying the approximation*

$$\begin{aligned}\mathbb{E}[\boldsymbol{\xi}_u^2(n)] &= \mathbb{E}[|\boldsymbol{\varepsilon}^{\mathrm{T}}(n)\mathbf{u}(n)|^2] \\ &\approx \mathbb{E}[\|\boldsymbol{\varepsilon}(n)\|^2]\mathbb{E}[u^2(n)] \\ &= \mathscr{D}(n)\mathbb{E}[u^2(n)], \end{aligned} \tag{7.28}$$

where $\mathscr{D}(n)$ *is the mean-square deviation*. The superscript T denotes transposition. Note that the approximate formula of Eq. (7.28) involves the input signal $u(n)$ rather than the tap-input vector $\mathbf{u}(n)$.

Assumption 3 is a statement of the low-pass filtering action of the LMS algorithm. Thus, using Eq. (7.28) in Eq. (7.26), we get the approximation

$$\widetilde{\mu}_{\text{opt}} \approx \frac{\mathscr{D}(n)\mathbb{E}[u^2(n)]}{\mathbb{E}[e^2(n)]}. \tag{7.29}$$

The practical virtue of the approximate formula for $\widetilde{\mu}_{\text{opt}}$ defined in Eq. (7.29) is borne out in the fact that simulations as well as real-time implementations have shown that Eq. (7.29) provides a good approximation for $\widetilde{\mu}_{\text{opt}}$ for the case of large filter lengths and speech inputs (Mader et al., 2000).

7.3 STEP-SIZE CONTROL FOR ACOUSTIC ECHO CANCELLATION

Almost all conversations are conducted in the presence of acoustic *echoes*. An echo may be *non-noticeable* or *distinct*, depending on the time delay involved. If the delay between the speech and its echo is short, the echo is non-noticeable, but perceived as a form of spectral distortion referred to as *reverberation*. If, on the other hand, the delay exceeds a few tens of milliseconds, the echo is distinctly noticeable.

Telecommunications environments, which are limited by the unavoidable presence of acoustic echoes, include the following (Sondhi & Berkley, 1980; Breining et al., 1999; Mader et al., 2000; Hänsler and Schmidt, 2004, 2008):

Telephone circuits. Every telephone set in a given geographical area is connected to a central office by a two-wire line called the *customer loop*, which serves the need for communication between two speakers. However, for telephone circuits longer than about 35 miles, a separate path is necessary for communication in either direction. Accordingly, there has to be provision for connecting a two-wire circuit to a four-wire circuit, which is accomplished by means of a hybrid transformer—basically, a bridge circuit with three ports (terminal pairs). When the bridge is not perfectly balanced, the input port becomes coupled to an output port, thereby giving rise to an electric echo.

Hands-free telephones. In the use of a hands-free telephone, we usually find that a loudspeaker and a microphone are placed a significant distance from each other, as in a teleconferencing environment. In such an environment, the microphone picks up not only the speech signal radiated by the loudspeaker, but also its echoes from the borders of the enclosure. As a result, the electroacoustic circuit may become unstable and produce a "howling" sound. Moreover, the users of the system become annoyed by having to listen to their own speech, delayed by the round-trip time of the system.

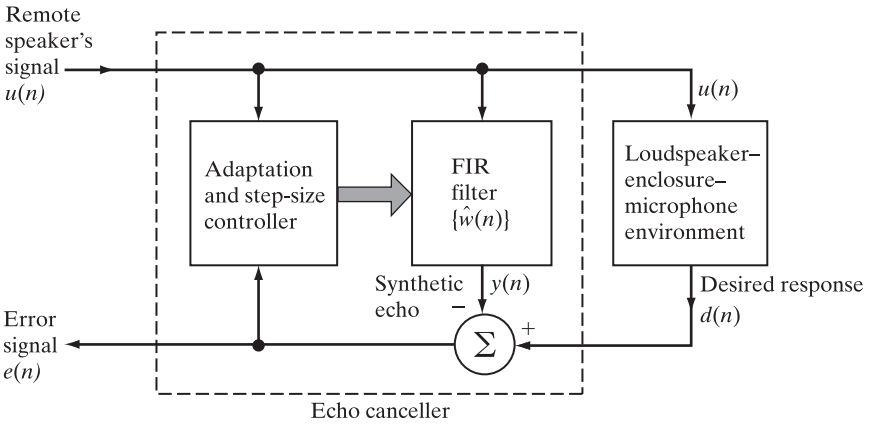


FIGURE 7.2 Structural diagram of an acoustic echo-control system.

To overcome the annoying presence of acoustic echoes in these telecommunication environments, the common practice is to use an adaptive *echo canceller*.[2] In telephone circuits, the adaptive echo canceller makes it possible to conduct an echo-free conversation between two speakers, regardless of the distance separating them. In hands-free telephones, the howling sound and a speaker's own delayed speech are both minimized.

The basic principle of acoustic cancellation may now be summarized as follows:

> Adaptively synthesize a replica of the echo and subtract it from the echo-corrupted signal.

In effect, acoustic echo cancellation is a form of the noise cancellation process described in Section 6.3.

To illustrate this principle, consider the block diagram shown in Figure 7.2. The diagram pertains to a hands-free telephone environment (Mader et al., 2000). Two main functional units are shown in the figure:

- Loudspeaker–enclosure–microphone (LEM).
- Echo canceller (EC).

A remote speaker's signal, denoted by $u(n)$, is radiated by the loudspeaker, picked up by the microphone, and convolved with the impulse response of the LEM to produce an output denoted by $d(n)$. The signal $d(n)$ is corrupted by an echo, due to reflections of $u(n)$ from the surrounding environment.

The echo canceller consists of two components: (1) an FIR filter and (2) an adaptation and step-size controller. The FIR filter convolves the remote speaker's

[2]For an exhaustive treatment of acoustic echo and noise control procedures and their practical implementations, see the book by Hänsler and Schmidt (2004).

signal $u(n)$ with its own impulse response $\hat{w}(n)$ to produce an estimate of the echo; this estimate is denoted by $y(n)$. Insofar as the operation of the echo canceller is concerned, the microphone output $d(n)$ constitutes the "desired response." The "synthetic echo" $y(n)$ produced by the FIR filter is subtracted from the microphone output $d(n)$ to produce an error signal $e(n)$. The remote speaker's signal $u(n)$ and the error signal $e(n)$ act on the adaptation and step-size controller to adjust the tap weights of the FIR filter so as to minimize the mean-square value of the error signal in accordance with the normalized LMS algorithm. The net result is that the error signal, constituting the output of the echo canceller, provides an estimate of the uncorrupted local speaker's signal.

Step-Size Control

The normalized LMS algorithm of Eq. (7.10) assumes the use of a constant value for the normalized step-size parameter $\widetilde{\mu}$. However, if the characterization of the disturbance $v(n)$ in the multiple regression model of Eq. (7.13) changes, which frequently happens in, for example, a hands-free telephone environment, the algorithm may malfunction due to the mismatch between the estimation errors $\xi_u(n)$ and $e(n)$. To be specific, suppose that there is an increase in the local disturbance $v(n)$. Then, since the error signal $e(n)$ grows with $v(n)$, we see from Eq. (7.29) that the increase in $e(n)$ results in a reduction in the upper bound on the step-size parameter $\widetilde{\mu}$. As a result, $\widetilde{\mu}$ may become oversized for the situation at hand, thereby leading to instability of the echo canceller.

Several factors are responsible for the generation of the disturbance $v(n)$ influencing the operation of an acoustic echo canceller (Breining et al., 1999):

- The speech signal produced by the local speaker leads to a disturbance of the adaptive filter. When both the local and the far-end speakers are simultaneously active, we have *double-talk*.
- There can be a permanent local noise (e.g., background noise generated in a car cabin), which also disturbs the error signal.
- In practice, it is difficult, if not impossible, to account for the complete impulse response of the surrounding environment, due to the large filter length required. The residual echo, with its origin in the part of the system that cannot be modeled, represents an additional source of local noise.
- Fixed-point digital signal processors are often used in the implementation of an adaptive filter, to limit the cost. The quantization noise of the fixed-point arithmetic used in the implementation is yet another source of local disturbance that affects the operation of the echo canceller.

To cope with these factors, there has to be a reduction in the step-size parameter $\widetilde{\mu}$. However, the use of a permanently small step-size parameter is undesirable for two reasons:

1. In the likely case of a large local disturbance, $\widetilde{\mu}$ may still be too high and therefore cause the adaptive filter to become unstable.

2. When the local disturbance is small, which is also likely to arise, the use of a small step size slows down the rate of convergence of the adaptive filter.

These are compelling reasons for the use of *step-size control*—hence the need for the replacement of $\widetilde{\mu}$ by a *time-varying* step-size parameter $\widetilde{\mu}(n)$ in Eq. (7.20).

A *two-stage control principle* is based on the equation

$$\widetilde{\mu}(n) = \begin{cases} 0 & \text{if } \widetilde{\mu}_{\text{opt}}(n) < \dfrac{1}{2}\,\widetilde{\mu}_{\text{fix}} \\ \widetilde{\mu}_{\text{fix}} & \text{otherwise} \end{cases}, \tag{7.30}$$

where $\widetilde{\mu}_{\text{fix}}$ is a *fixed* nonzero step size and $\widetilde{\mu}_{\text{opt}}(n)$ is the *optimal* step size for adaptation cycle n. According to Eq. (7.30), there is *no* filter adaptation if the computed value of $\widetilde{\mu}_{\text{opt}}(n)$ at adaptation cycle n is less than $\frac{1}{2}\widetilde{\mu}_{\text{fix}}$. Otherwise, $\widetilde{\mu}(n)$ is held constant at $\widetilde{\mu}_{\text{fix}}$. The use of Eq. (7.30) requires knowledge of the optimal step-size $\widetilde{\mu}_{\text{opt}}(n)$. Mader et al. (2000) get around this problem by using an approximation of the optimal step size instead. A scheme for obtaining this approximation is described next.

As remarked earlier, Eq. (7.29) provides a good approximation of $\widetilde{\mu}_{\text{opt}}(n)$ for speech inputs. According to that equation, the estimation of $\widetilde{\mu}_{\text{opt}}(n)$ is reduced to three separate estimation problems:

1. Estimation of the error signal power, $\mathbb{E}[e^2(n)]$.
2. Estimation of the input signal power, $\mathbb{E}[u(n)^2]$.
3. Estimation of the mean-square deviation, $\mathcal{D}(n)$.

The input signal $u(n)$ and the error signal $e(n)$ are both accessible; hence, the estimation of their respective average powers is straightforward. However, the estimation of $\mathcal{D}(n)$ needs more detailed consideration, because the parameter vector $\mathbf{w}$ of the multiple regression model characterizing the LEM environment is unknown.

For the short-term power estimation of a signal denoted by $x(n)$, we may use the idea of *convex combination* to formulate a first-order recursive procedure described by

$$\overline{x^2(n+1)} = (1-\gamma)x^2(n+1) + \gamma\overline{x^2(n)}, \tag{7.31}$$

where $\overline{x^2(n)}$ is the power estimate at time step n and γ is a smoothing constant. Typically, γ lies inside the interval [0.9, 0.999]. For a prescribed γ, we may thus use Eq. (7.31) to estimate the error signal power $\mathbb{E}[e^2(n)]$ and input signal power $\mathbb{E}[u^2(n)]$ by setting $x(n) = e(n)$ and $x(n) = u(n)$, respectively.

Turning next to the estimation of the mean-square deviation $\mathcal{D}(n)$, we may use a procedure involving the insertion of an *artificial delay* into the LEM system (Yamamoto & Kitayama, 1982; Mader et al., 2000). This procedure, in which the remote speaker's signal $u(n)$ is artificially delayed by M_D samples, is illustrated in Fig. 7.3. The delay is naturally modeled by the adaptive FIR filter in the echo canceller. Thus,

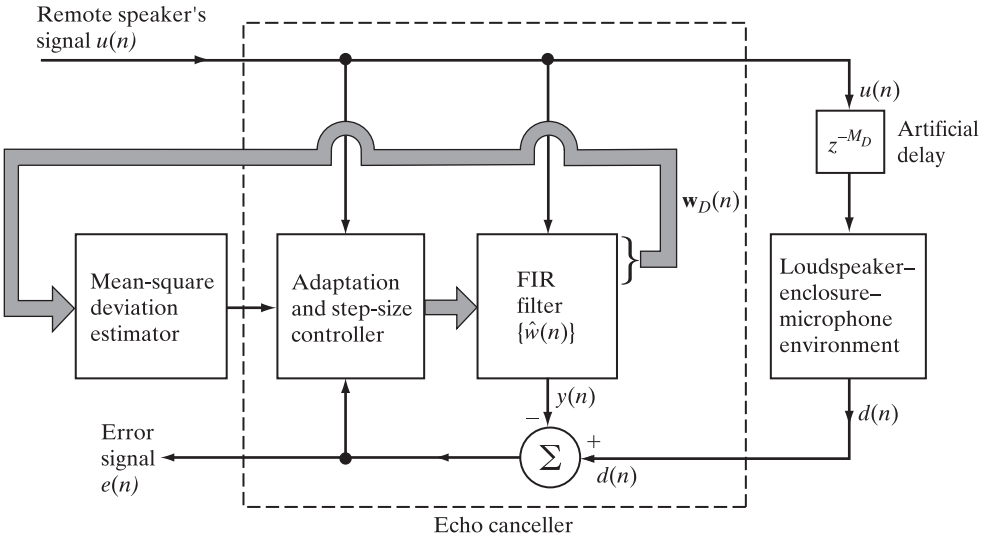


FIGURE 7.3 A structure incorporating the means for estimating the mean-square deviation $D(n)$; the block labeled z^{-M_D} represents an artificial delay of M_D samples.

the section of the "true" parameter vector $\mathbf{w}$ corresponding to the artificial delay is zero, in which case we may set the kth element of the error

$$\varepsilon_k(n) = -\hat{w}_k(n) \quad \text{for} \quad k = 0, \ldots, M_D - 1. \tag{7.32}$$

Next, we utilize the known property that an adaptive filter tends to spread the weight-error (filter mismatch) vector $\boldsymbol{\varepsilon}(n)$ evenly over its M taps (Yamamoto & Kitayama, 1982). Thus, using the known portion of the filter mismatch defined in Eq. (7.32), we may approximate the mean-square deviation as

$$\mathscr{D}(n) \approx \frac{M}{M_D} \sum_{k=0}^{M_D-1} \hat{w}_k^2(n), \tag{7.33}$$

where M is the length of the normalized LMS algorithm and $\hat{w}_k(n)$ is the kth component of the M-by-1 tap-weight vector $\hat{\mathbf{w}}(n)$. Finally, given the estimates of the signal powers $\mathbb{E}[e^2(n)]$ and $\mathbb{E}[u^2(n)]$ obtained by using Eq. (7.31) and the estimate of $\mathscr{D}(n)$ obtained by using Eq. (7.33), the optimal step size $\widetilde{\mu}_{\text{opt}}(n)$ is computed as $\mathscr{D}(n)\,\mathbb{E}[e^2(n)]/\mathbb{E}[e^2(n)]$.

In general, the step-size control based on the use of artificial delay as depicted in Fig. 7.3 delivers good performance. Specifically, when the error signal $e(n)$ increases due to a local excitation, the step-size parameter is reduced, thereby avoiding divergence of the filter. Unfortunately, however, a change in the LEM environment may lead to a *freezing* of the system in that a new adaptation of the filter and the "delay" coefficients

is prevented. To overcome this problem, an additional detector for LEM change is required, which enables the filter to readapt by either setting the delay coefficients or the step size to larger values.[3]

7.4 GEOMETRIC CONSIDERATIONS PERTAINING TO THE CONVERGENCE PROCESS FOR REAL-VALUED DATA

Returning to Eq. (7.9), which has to do with the weight-vector adjustment $\delta\hat{\mathbf{w}}(n+1)$ applied to a normalized LMS algorithm at adaptation cycle $n+1$, we make two observations:

1. The *direction* of the adjustment $\delta\hat{\mathbf{w}}(n+1)$ is the same as that of the input vector $\mathbf{u}(n)$.
2. The *size* of the adjustment $\delta\hat{\mathbf{w}}(n+1)$ is dependent on the *sample correlation coefficient* between the input vectors $\mathbf{u}(n)$ and $\mathbf{u}(n-1)$, which, for real-valued data, is defined by

$$\rho_{\text{sample}}(n) = \frac{\mathbf{u}^{\mathrm{T}}(n)\mathbf{u}(n-1)}{\|\mathbf{u}(n)\| \cdot \|\mathbf{u}(n-1)\|}. \tag{7.34}$$

(The reason for confining the discussion in this section to real-valued data is merely to ease the burden of geometric presentations that follow.)

[3]Mader et al. (2000) present an example that illustrates the "freezing" phenomenon, which arises in the artificial delay method.

Mader et al. also describe another method for estimating the mean-square deviation. The undisturbed estimation error $\xi_u(n)$ may be approximated by the error signal $e(n)$ provided that two conditions are satisfied:

- Sufficient excitation power is applied.
- The local speaker is not active, which requires the use of a reliable single-talk detection scheme.

When a remote single talk is present, a form of recursive smoothing is used to estimate the mean-square deviation; otherwise, the old value is maintained. Thus, using $\mathscr{D}_p(n)$ to denote the mean-square deviation based on power estimation, for real-valued data we may write (Mader et al., 2000)

$$\begin{aligned}\mathscr{D}_p(n) &= \frac{\overline{e^2(n)}\,|_{\nu(n)=0}}{\overline{u^2(n)}} \\ &\simeq \begin{cases} \gamma\mathscr{D}_p(n-1) + (1-\gamma)\dfrac{\overline{e^2(n)}}{\overline{u^2(n)}} & \text{if remote single talk is detected} \\ \mathscr{D}_p(n-1) & \text{otherwise} \end{cases},\end{aligned}$$

where γ is a positive constant less than unity. The advantage of step-size control based on this formula is that it does not lead to freezing of adaptation if the LEM system is disturbed.

From a geometric perspective, Eq. (7.34) has an insightful interpretation, as depicted in Fig. 7.4(a) for $\widetilde{\mu} = 1$ and $M = 3$. Elements of two M-dimensional spaces, namely, the input data space and the weight space, are shown in the figure. Specifically, π_n is the set of all weight vectors $\hat{\mathbf{w}}(n)$ that act on the input vector $\mathbf{u}(n)$ to produce the output $y(n)$, and similarly for π_{n-1}. The angle θ subtended between the hyperplanes π_n and π_{n-1} is the same as the angle between the input vectors $\mathbf{u}(n)$ and $\mathbf{u}(n-1)$. From signal-space theory, we know that the cosine of the angle θ between the vectors $\mathbf{u}(n)$ and $\mathbf{u}(n-1)$ is defined as the inner product $\mathbf{u}^T(n)\mathbf{u}(n-1)$, divided by the product of the Euclidean norms $\|\mathbf{u}(n)\|$ and $\|\mathbf{u}(n-1)\|$ (Wozencraft & Jacobs, 1965). Consequently, we may expand on observation 2 as follows:

2.1 When the angle θ is $\pm 90°$ [i.e, when the input vectors $\mathbf{u}(n)$ and $\mathbf{u}(n-1)$ are orthogonal to each other], the rate of convergence of the normalized LMS algorithm is the *fastest*.

2.2 When the angle θ is zero or 180° [i.e., when the input vectors $\mathbf{u}(n)$ and $\mathbf{u}(n-1)$ point in the same direction or in opposite directions], the rate of convergence of the normalized LMS algorithm is the *slowest*.

To guard against the possibility of the undesired situation described under observation 2.2—that is, to maintain the rate of convergence essentially constant, independently of the angle θ between the input vectors $\mathbf{u}(n)$ and $\mathbf{u}(n-1)$—we may use a generalization of the normalized LMS algorithm known as the *affine projection filter* (Ozeki & Umeda, 1984), considered in the next section.

A geometric description of the operating principle of this new adaptive filter is depicted in Fig. 7.4(b) for the case of $\widetilde{\mu} = 1$ and $M = 3$. In the figure, $\pi_n \cap \pi_{n-1}$ denotes the intersection of the hyperplanes π_n and π_{n-1}. Comparing the picture portrayed here with the corresponding picture of Fig. 7.4(a) pertaining to a normalized LMS algorithm, we observe that *in the weight space, the line joining* $\hat{\mathbf{w}}(n+1)$ *to* $\hat{\mathbf{w}}(n)$ *is normal to* $\pi_n \cap \pi_{n-1}$ *rather than* π_n.

In the context of a multidimensional space, we should distinguish between a *subspace* and an *affine subspace*. By definition, a subspace passes through the origin of the multidimensional space, whereas an affine subspace does not. Referring to Fig. 7.4(b), we note that the intersection of the hyperplanes π_n and π_{n-1} does *not* necessarily contain the origin of the M-dimensional weight space. Thus, $\pi_n \cap \pi_{n-1}$ is an affine subspace and hence the name "affine projection filter," the algorithmic formulation of which is discussed next.[4]

[4]The book edited by Gay and Benesty (2000) presents a detailed treatment of the affine projection filter, its "fast" implementations, and their applications to acoustic echoing and noise control for telecommunications. In addition to these monochannel acoustic echo cancellers, the book discusses multichannel echo cancellers and more elaborate systems involving the use of microphone arrays.

The affine projection filter and its application to echo cancellation are also discussed in the book by Benesty et al. (2001).

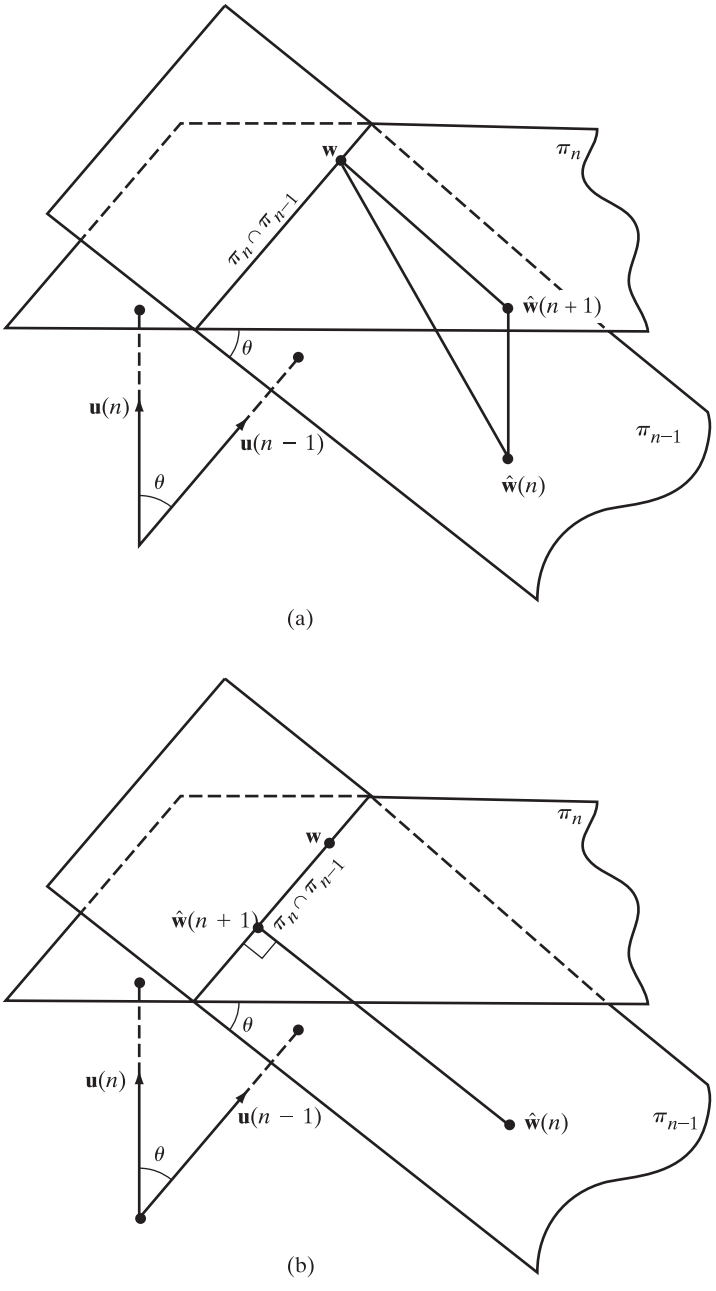


FIGURE 7.4 Geometric interpretation of (a) the normalized LMS algorithm and (b) the affine projection adaptive algorithm. In both parts of the figure, the vector $\mathbf{w}$ denotes the unknown parameter vector of a multiple regression model.

7.5 AFFINE PROJECTION ADAPTIVE FILTERS

In mathematical terms, we may formulate the criterion for designing an affine projection filter as one of optimization subject to multiple constraints, as follows:

> Minimize the squared Euclidean norm of the change in the weight vector
>
> $$\delta\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n), \tag{7.35}$$
>
> subject to the set of N constraints
>
> $$d(n-k) = \hat{\mathbf{w}}^{\mathrm{H}}(n+1)\mathbf{u}(n-k) \quad \text{for} \quad k = 0, 1, \ldots, N-1, \tag{7.36}$$
>
> where N is smaller than the dimensionality M of the input data space or, equivalently, the weight space.

This constrained optimization criterion includes that of the normalized LMS algorithm as a special case, namely, $N = 1$. We may view N, the number of constraints, as the *order* of the affine projection adaptive filter.

Following the method of *Lagrange multipliers with multiple constraints* described in Appendix C, we may combine Eqs. (7.35) and (7.36) to set up the following cost function for the affine projection filter:

$$J(n) = \|\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)\|^2 + \sum_{k=0}^{N-1} \mathrm{Re}[\lambda_k^*(d(n-k) - \hat{\mathbf{w}}^{\mathrm{H}}(n+1)\mathbf{u}(n-k))]. \tag{7.37}$$

In this function, the λ_k are the Lagrange multipliers pertaining to the multiple constraints. For convenience of presentation, we introduce the following definitions:

- An N-by-M *data matrix* $\mathbf{A}(n)$ whose Hermitian transpose is defined by
$$\mathbf{A}^{\mathrm{H}}(n) = [\mathbf{u}(n), \mathbf{u}(n-1), \ldots, \mathbf{u}(n-N+1)]. \tag{7.38}$$
- An N-by-1 *desired response vector* whose Hermitian transpose is defined by
$$\mathbf{d}^{\mathrm{H}}(n) = [d(n), d(n-1), \ldots, d(n-N+1)]. \tag{7.39}$$
- An N-by-1 *Lagrange vector* whose Hermitian transpose is defined by
$$\boldsymbol{\lambda}^{\mathrm{H}} = [\lambda_0, \lambda_1, \ldots, \lambda_{N-1}]. \tag{7.40}$$

Using these matrix definitions in Eq. (7.37), we may redefine the cost function in the more compact form

$$J(n) = \|\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)\|^2 + \mathrm{Re}[(\mathbf{d}(n) - \mathbf{A}(n)\hat{\mathbf{w}}(n+1))^{\mathrm{H}}\boldsymbol{\lambda}]. \tag{7.41}$$

Then, following the rules for differentiation with respect to a complex-valued vector of the Wirtinger calculus (see Appendix B), we find that the derivative of the cost function $J(n)$ with respect to the weight vector $\hat{\mathbf{w}}(n+1)$ is

$$\frac{\partial J(n)}{\partial \hat{\mathbf{w}}^{\mathrm{H}}(n+1)} = 2(\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)) - \mathbf{A}^{\mathrm{H}}(n)\boldsymbol{\lambda}.$$

Setting this derivative equal to zero, we get

$$\delta\hat{\mathbf{w}}(n+1) = \frac{1}{2}\mathbf{A}^{\mathrm{H}}(n)\boldsymbol{\lambda}. \tag{7.42}$$

To eliminate the Lagrange vector $\boldsymbol{\lambda}$ from Eq. (7.42), we first use the definitions of Eqs. (7.38) and (7.39) to rewrite Eq. (7.36) in the equivalent form

$$\mathbf{d}(n) = \mathbf{A}(n)\hat{\mathbf{w}}(n+1). \tag{7.43}$$

Premultiplying both sides of Eq. (7.42) by $\mathbf{A}(n)$ and then using Eqs. (7.35) and (7.43) to eliminate the updated weight vector $\hat{\mathbf{w}}(n+1)$ yields

$$\mathbf{d}(n) = \mathbf{A}(n)\hat{\mathbf{w}}(n) + \frac{1}{2}\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n)\boldsymbol{\lambda}, \tag{7.44}$$

from which we deduce the following:

- The difference between $\mathbf{d}(n)$ and $\mathbf{A}(n)\hat{\mathbf{w}}(n)$, based on the data available at adaptation cycle n, is the N-by-1 *error vector*

$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{A}(n)\hat{\mathbf{w}}(n). \tag{7.45}$$

- The matrix product $\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n)$ is an N-by-N matrix with an inverse denoted by $(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}$.

Thus, solving Eq. (7.44) for the Lagrange vector, we have

$$\boldsymbol{\lambda} = 2(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{e}(n). \tag{7.46}$$

Substituting this solution into Eq. (7.42) yields the optimum change in the weight vector:

$$\delta\hat{\mathbf{w}}(n+1) = \mathbf{A}^{\mathrm{H}}(n)(\mathbf{A}(n)(\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{e}(n). \tag{7.47}$$

Finally, we need to exercise control over the change in the weight vector from one adaptation cycle to the next, but keep the same direction. We do so by introducing the step-size parameter $\widetilde{\mu}$ into Eq. (7.47), yielding

$$\delta\hat{\mathbf{w}}(n+1) = \widetilde{\mu}\mathbf{A}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{e}(n). \tag{7.48}$$

Equivalently, we write

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \widetilde{\mu}\mathbf{A}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{e}(n), \tag{7.49}$$

which is the desired update equation for the affine projection adaptive filter. Table 7.2 presents a summary of this algorithm.

Affine Projection Operator

As explained in Section 7.4, the updated weight vector $\hat{\mathbf{w}}(n+1)$ is the result of an affine projection operator acting on $\hat{\mathbf{w}}(n)$. To determine this operator, we substitute Eq. (7.45) into Eq. (7.49), obtaining

$$\begin{aligned}\hat{\mathbf{w}}(n+1) = {} & [\,\mathbf{I} - \widetilde{\mu}\mathbf{A}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{A}(n)]\hat{\mathbf{w}}(n) \\ & + \widetilde{\mu}\mathbf{A}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{d}(n),\end{aligned} \tag{7.50}$$

where $\mathbf{I}$ is the identity matrix. Define the *projection operator*:

$$\mathbf{P} = \mathbf{A}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{A}(n), \tag{7.51}$$

TABLE 7.2 Summary of the Affine Projection Adaptive Filter

Parameters: M = number of taps
$\widetilde{\mu}$ = adaptation constant
N = number of multiple constraints, which defines the filter order

Initialization: If prior knowledge on the tap-weight vector $\hat{\mathbf{w}}(0)$ is available, use it to initialize the filter. Otherwise, set $\hat{\mathbf{w}}(0) = \mathbf{0}$.

Data:
(a) Given: $\mathbf{u}(n) = M$-by-1 tap-input vector at time step n
$= [u(n), u(n-1), \ldots, u(u-M+1)]^T$.
$d(n)$ = desired response and time step n.
(b) To be computed: $\hat{\mathbf{w}}(n+1)$ = estimate of Wiener solution at time step $n+1$.

Computation: n = 0, 1, 2, ...

$$\mathbf{A}^{\mathrm{H}}(n) = [\mathbf{u}(n), \mathbf{u}(n-1), \ldots, \mathbf{u}(n-z+1)]$$
$$\mathbf{d}^{\mathrm{H}}(n) = [d(n), d(n-1), \ldots, d(n-z+1)]$$
$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{A}(n)\hat{\mathbf{w}}(n)$$
$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \widetilde{\mu}\mathbf{A}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{e}(n).$$

which is uniquely determined by the data matrix $\mathbf{A}(n)$. For prescribed $\widetilde{\mu}$, $\mathbf{A}(n)$, and $\mathbf{d}(n)$, the *complement projector* $[\mathbf{I} - \widetilde{\mu}\mathbf{P}]$ acts on the old weight vector $\hat{\mathbf{w}}(n)$ to produce the updated weight vector $\hat{\mathbf{w}}(n+1)$. Most importantly, it is the presence of the second term in Eq. (7.50), namely, $\widetilde{\mu}\mathbf{A}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{d}(n)$ that makes the complement projection into an affine projection rather than just a projection.

In Chapter 9, dealing with the method of least squares, we show that, for N less than M, the matrix $\mathbf{A}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}$ is the *pseudoinverse* of the data matrix $\mathbf{A}(n)$. Using $\mathbf{A}^{+}(n)$ to denote this pseudoinverse, we may simplify the updated formula of Eq. (7.50) as

$$\hat{\mathbf{w}}(n+1) = [\mathbf{I} - \widetilde{\mu}\mathbf{A}^{+}(n)\mathbf{A}(n)]\hat{\mathbf{w}}(n) + \widetilde{\mu}\mathbf{A}^{+}(n)\mathbf{d}(n). \tag{7.52}$$

Indeed, it is because of the defining equation (7.52) that we may view the affine projection filter as an intermediate adaptive filter between the normalized LMS algorithm of Section 7.1 and the recursive least-squares (RLS) algorithm of Chapter 10, in terms of both computational complexity and performance.

Stability Analysis of the Affine Projection Adaptive Filter

As with the normalized LMS algorithm, we may base stability analysis of the affine projection adaptive filter on the mean-square deviation $\mathcal{D}(n)$ defined in Eq. (7.15). Subtracting Eq. (7.49) from the unknown weight vector $\mathbf{w}$ of the multiple regression model serving as a frame of reference, we may write

$$\boldsymbol{\varepsilon}(n+1) = \boldsymbol{\varepsilon}(n) - \widetilde{\mu}\mathbf{A}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{e}(n). \tag{7.53}$$

Using this updated equation in the definition of $\mathcal{D}(n)$, rearranging and simplifying terms, we get

$$\mathcal{D}(n+1) - \mathcal{D}(n) = \widetilde{\mu}^2 \mathbb{E}[\mathbf{e}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{e}(n)] - 2\widetilde{\mu}\mathbb{E}\{\mathrm{Re}[\boldsymbol{\xi}_{\mathrm{u}}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{e}(n)]\}, \tag{7.54}$$

where

$$\boldsymbol{\xi}_u(n) = \mathbf{A}(n)(\mathbf{w} - \hat{\mathbf{w}}(n)) \tag{7.55}$$

is the *undisturbed error vector*. From Eq. (7.54) we readily see that the mean-square deviation $\mathcal{D}(n)$ decreases monotonically with increasing n, provided that the step-size parameter $\widetilde{\mu}$ satisfies the condition

$$0 < \widetilde{\mu} < \frac{2\mathbb{E}\{\mathrm{Re}[\boldsymbol{\xi}_u^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{e}(n)]\}}{\mathbb{E}[\mathbf{e}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n)]^{-1}\mathbf{e}(n)]} \tag{7.56}$$

which contains the corresponding formula of Eq. (7.18) for the normalized LMS algorithm as a special case. The optimal step size is defined by

$$\widetilde{\mu}_{\mathrm{opt}} = \frac{\mathbb{E}\{\mathrm{Re}[\boldsymbol{\xi}_u^{\mathrm{H}}(n)\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n)^{-1}\mathbf{e}(n)]\}}{\mathbb{E}[\mathbf{e}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{e}(n)]}. \tag{7.57}$$

To simplify this formula, we expand on Assumptions 1 and 2 for the normalized LMS algorithm, proposing two more assumptions:

Assumption 4. *From one adaptation cycle to the next, the fluctuations in the inverse of the matrix product* $\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n)$, *which is common to the numerator and denominator of* $\widetilde{\mu}_{\mathrm{opt}}$, *are small enough to justify approximating* $\widetilde{\mu}_{\mathrm{opt}}$ *as*

$$\widetilde{\mu}_{\mathrm{opt}} \approx \frac{\mathbb{E}\{\mathrm{Re}[\boldsymbol{\xi}_u^{\mathrm{H}}(n)\mathbf{e}(n)]\}}{\mathbb{E}[\|\mathbf{e}(n)\|^2]}. \tag{7.58}$$

Assumption 5. *The undisturbed error vector* $\boldsymbol{\xi}_u(n)$ *is uncorrelated with the disturbance (noise) vector*

$$\boldsymbol{\nu}^{\mathrm{H}}(n) = [\nu(n), \nu(n-1), \ldots, \nu(n-N+1)]. \tag{7.59}$$

Assumption 3, generalized for complex data, applies equally well to the affine projection filter. Hence, invoking Assumptions 3 through 5, we may simplify Eq. (7.57) to

$$\begin{aligned}\widetilde{\mu}_{\mathrm{opt}} &\approx \frac{\mathbb{E}\{\|\boldsymbol{\xi}_u(n)\|^2\}}{\mathbb{E}[\|\mathbf{e}(n)\|^2]} \\ &= \frac{\sum_{k=0}^{N-1}\mathbb{E}[|\,\boldsymbol{\xi}_u(n-k)\,|^2]}{\sum_{k=0}^{N-1}\mathbb{E}[|\,e(n-k)\,|^2]} \\ &\approx \frac{\sum_{k=0}^{N-1}\mathcal{D}(n-k)\mathbb{E}[|\,u(n-k)\,|^2]}{\sum_{k=0}^{N-1}\mathbb{E}[|\,e(n-k)\,|^2]}.\end{aligned} \tag{7.60}$$

For real-valued data, putting the filter order $N = 1$ in Eq. (7.60) reduces that formula to Eq. (7.29) for the normalized LMS algorithm.

Summarizing Remarks on Affine Projection Adaptive Filters

The main motivation for using an affine projection adaptive filter is to provide an improvement in the rate of convergence over the normalized LMS algorithm. In this context, we may make the following observations on the convergence behavior of affine projection adaptive filters (Sankaran & Beex, 2000):

1. The learning curve of an affine projection adaptive filter consists of the sum of exponential terms.
2. An affine projection adaptive filter converges at a rate faster than that of the corresponding normalized LMS algorithm.
3. As more delayed versions of the tap-input vector $\mathbf{u}(n)$ are used (i.e., the filter order N is increased), the rate of convergence improves, but the rate at which improvement is attained decreases.

In any event, improved convergence behavior is accomplished at the expense of increased computational complexity.[5]

7.6 SUMMARY AND DISCUSSION

In this chapter, we extended the family of LMS algorithms by deriving the normalized LMS algorithm and the affine projection filter. The normalized LMS algorithm differs from the traditional LMS algorithm in the way in which the step size for

[5]The inversion of the N-by-N matrix product $\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n)$ in Eq. (7.49) for an affine projection filter raises the need for two practical modifications to that updated equation:

1. *Regularization.* In a noisy environment, the inversion of the N-by-N matrix product $\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n)$ may give rise to numerical difficulties. To guard against such a possibility, we modify that product by adding to it the term $\delta\mathbf{I}$, where δ is a small positive constant and $\mathbf{I}$ is the N-by-N identity matrix. This modification is referred to as *regularization,* on which more is said in Chapter 10.
2. *Fast implementation.* As the projection dimension N increases, so does the rate of convergence of the affine projection adaptive filter. Unfortunately, this improvement in performance is attained at the expense of a corresponding increase in computational complexity. To mitigate the computational complexity problem, we may follow one of two approaches:
   - Time-domain approach, which exploits the *time-shifting property* of the input vector $\mathbf{u}(n)$. This property embodies the fact that $\mathbf{u}(n)$ and its previous value, $\mathbf{u}(n-1)$, share the elements $u(n-1), \ldots, u(n-M+1)$, where M is the input space dimensionality. The *fast affine projection (FAP) filter* devised by Gay and Tavathia (1995) follows this approach.
   - Frequency-domain approach, which exploits a fast FIR filtering technique based on the idea of "fast convolution." It uses the fast Fourier transform algorithm. The *block exact fast affine projection (BEFAP) filter* devised by Tanaka et al. (1999) follows this second approach.

Unfortunately, regularization and fast implementation of an affine projection filter may conflict in that the aforementioned fast versions of the filter rely on a certain approximation that is violated if regularization is applied (Rombouts & Moonen, 2000).

controlling the adjustment to the filter's tap-weight vector is defined. Whereas in the traditional LMS algorithm the step size is a scalar parameter denoted by μ, in the general case of a normalized LMS algorithm it is defined by $\tilde{\mu}/(\|\mathbf{u}(n)\|^2 + \delta)$, where $\tilde{\mu}$ is dimensionless, $\|\mathbf{u}(n)\|$ is the Euclidean norm of the tap-input vector $\mathbf{u}(n)$, and δ is a small positive constant. The advantages of the normalized LMS algorithm are twofold:

1. The normalized LMS algorithm mitigates the gradient noise amplification problem, which can arise when the tap-input vector $\mathbf{u}(n)$ is large.
2. The rate of convergence of the normalized LMS algorithm is potentially faster than that of the traditional LMS algorithm for both uncorrelated and correlated input data.

The affine projection filter is a generalization of the normalized LMS algorithm. Specifically, the adjustment term $\tilde{\mu}\mathbf{u}(n)e^*(n)/(\|\mathbf{u}(n)\|^2 + \delta)$ applied to the tap-weight vector in the normalized LMS algorithm is replaced by the more elaborate term $\tilde{\mu}\mathbf{A}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n) + \delta\mathbf{I})^{-1}\mathbf{e}(n)$, where $\mathbf{I}$ is the identity matrix, δ is a small positive constant,

$$\mathbf{A}^{\mathrm{H}}(n) = [\mathbf{u}(n), \mathbf{u}(n-1), \mathbf{u}(n-N+1)],$$
$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{A}(n)\hat{\mathbf{w}}(n),$$

and

$$\mathbf{d}^{\mathrm{H}}(n) = [d(n), d(n-1), \ldots, d(n-N+1)].$$

Because of the use of $(N-1)$ past values of both the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$, the affine projection filter may be viewed as an adaptive filter that is intermediate between the normalized LMS algorithm and the recursive least-squares filter (to be discussed in Chapter 10). Consequently, the affine projection filter provides a significant improvement in convergence, which is attained unfortunately at the cost of increased computational complexity—simply put, there is no free lunch.

PROBLEMS

1. In Section 7.1, we presented a derivation of the normalized LMS algorithm in its own right. In this problem, we explore another derivation of that algorithm by modifying the method of steepest descent that led to the development of the traditional LMS algorithm. The modification involves writing the tap-weight vector update in the method of steepest descent as

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \frac{1}{2}\mu(n)\boldsymbol{\nabla}(n),$$

where $\mu(n)$ is a *time-varying step-size parameter* and $\nabla(n)$ is the gradient vector defined by

$$\boldsymbol{\nabla}(n) = 2[\mathbf{R}\mathbf{w}(n) - \mathbf{p}],$$

in which $\mathbf{R}$ is the correlation matrix of the tap-input vector $\mathbf{u}(n)$ and $\mathbf{p}$ is the cross-correlation vector between the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$.

(a) At time $n+1$, the mean-square error is defined by

$$J(n+1) = \mathbb{E}[|e(n+1)|^2],$$

where

$$e(n+1) = d(n+1) - \mathbf{w}^{\mathrm{H}}(n+1)\mathbf{u}(n+1).$$

Determine the value of the step-size parameter $\mu_o(n)$ that minimizes $J(n+1)$ as a function of $\mathbf{R}$ and $\nabla(n)$.

(b) Using instantaneous estimates for $\mathbf{R}$ and $\nabla(n)$ in the expression for $\mu_o(n)$ derived in part (a), determine the corresponding instantaneous estimate for $\mu_o(n)$. Hence, formulate the update equation for the tap-weight vector $\hat{\mathbf{w}}(n)$, and compare your result with that obtained for the normalized LMS algorithm.

2. Demonstrate that for real-valued data with the filter order $N = 1$, Eq. (7.60) reduces to Eq. (7.29) for the normalized LMS algorithm.

3. Two statisticians were asked to formulate the weight updates for the normalized LMS algorithm. Given the step-size parameter $\tilde{\mu}$, the tap-input vector

$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^{\mathrm{T}},$$

and the corresponding desired response $d(n)$, one statistician presented the update formula

(i) $$\hat{w}_k(n+1) = \hat{w}_k(n) + \frac{\tilde{\mu}}{|u(n-k)|^2}u(n-k)e^*(n), \qquad k = 0, 1, \ldots, M-1,$$

where

$$e(n) = d(n) - \sum_{k=0}^{M-1} \hat{w}_k^*(n)u(n-k).$$

The second statistician presented the different update formula

(ii) $$\hat{w}_k(n+1) = \hat{w}_k(n) + \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}u(n-k)e^*(n), \qquad k = 0, 1, \ldots, M-1,$$

where $e(n)$ is defined as before and $\|\mathbf{u}(n)\|$ is the Euclidean norm of $\mathbf{u}(n)$.

The difference between these two formulas lies in the way in which the normalization is performed. In light of the normalized LMS algorithm theory presented in Section 7.1, which of the two formulas is the correct one? Justify your answer.

4. (a) Explain in detail the procedure to find the optimum value of the updated weight vector.

(b) Explain the procedure for solving a constrained optimization problem.

5. Explain in detail how an affine projection adaptive filter converges at a rate faster than that of the corresponding normalized LMS algorithm.

6. Illustrate how an adaptive filter performs circular convolution instead of linear convolution.

7. The affine projection adaptive filtering algorithm is used to estimate the coefficients of an autoregressive (AR) process defined by

$$u(n) = \sum_{k=1}^{N-1} w_k^* u(n-k) + v(n),$$

where $v(n)$ is a zero-mean, white-Gaussian-noise process.

(a) Formulate the algorithm for computing the AR coefficients $w_1, w_2, \ldots, w_{N-1}$, assuming that the number of constraints used in the algorithm is N.

(b) Define the M-by-1 vector

$$\boldsymbol{\phi}(n)=[\mathbf{I} - \mathbf{A}^{\mathrm{H}}(n)(\mathbf{A}(n)\mathbf{A}^{\mathrm{H}}(n))^{-1}\mathbf{A}(n)]\mathbf{u}(n),$$

where

$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^{\mathrm{T}}$$

and

$$\mathbf{A}^{\mathrm{H}}(n) = [\mathbf{u}(n), \mathbf{u}(n-1), \ldots, \mathbf{u}(n-N+1)].$$

Show that the vector $\boldsymbol{\phi}(n)$ is a vector whose elements are estimates of zero-mean, white-Gaussian-noise processes.

8. In the normalized LMS algorithm, the condition $0 < \widetilde{\mu} < 2$ is often given as a necessary condition for its stability. Examine Eq. (7.19) for the normalized LMS algorithm. Discuss the situations for which the upper bound of 2 on $\widetilde{\mu}$ is justifiable.

9. Explain in detail the importance of normalization in the normalized LMS algorithm over an LMS algorithm.

Computer Experiments

10. In this problem, we revisit the computer experiment described in Problem 18 of Chapter 6. We are given the AR process

$$u(n) = -a_1 u(n-1) - a_2 u(n-2) + v(n),$$

where $a_1 = 0.1$ and $a_2 = -0.8$. The $v(n)$ is white noise with zero mean and a variance chosen to make the variance of $u(n)$ equal to unity.

(a) Plot the learning curve of the normalized LMS algorithm used to estimate the AR parameters a_1 and a_2. In this computation, use the following parameters:

$$\widetilde{\mu} = 0.2$$

and

$$\delta = 0.5.$$

For the plot, average the squared error signal $e(n)$ over an ensemble of 100 independent Monte Carlo runs of the experiment.

(b) Plot the corresponding errors in tap-weight estimates.

(c) Repeat the plots in parts (a) and (b) for the regularizing parameter $\delta = 0.25, 0.75$. What observation can you make on the effect of varying δ?

CHAPTER 8

Block-Adaptive Filters

In the traditional and normalized least-mean-square (LMS) algorithms described in Chapters 6 and 7, the tap weights (free parameters) of a finite-duration impulse response (FIR) filter are adapted in the time domain. Recognizing that the *Fourier transform* maps time-domain signals into the frequency domain and that the inverse Fourier transform provides the inverse mapping that takes us back into the time domain, we see that it is equally feasible to perform the adaptation of filter parameters in the frequency domain. In such a case, we speak of *frequency-domain adaptive filtering* (*FDAF*), the origin of which may be traced back to an early paper by Walzman and Schwartz (1973).

There are two motivations for seeking adaptation in the frequency domain in one form or another:

> **Motivation 1.** In certain applications, such as acoustic echo cancellation in teleconferencing, the adaptive filter is required to have a long impulse response (i.e., long memory) to cope with an equally long echo duration. When the LMS algorithm is adapted in the time domain, we find that the requirement of a long memory results in a significant increase in the computational complexity of the algorithm. Frequency-domain adaptive filters provide a possible solution to the computational complexity problem.
>
> **Motivation 2.** Self-orthogonalizing adaptive filtering, mechanized in a different way from that described under Motivation 1, is used to improve the convergence performance of the traditional LMS algorithm. We thus have a second motivation: a more uniform convergence rate is attained by exploiting the *orthogonality properties* of the discrete cosine transform (DCT).

Both of these approaches to linear adaptive filtering are discussed in this chapter.

The chapter also discusses the subband adaptive filter, which is different from a self-orthogonalizing adaptive filter. In a subband adaptive filter, filters with high stopband rejection are used to band-partition the input signal and hence possibly provide an improvement in convergence over the frequency-domain adaptive filter. Moreover, by decimating the subband signals (i.e., down-sampling to a lower rate), it is possible to achieve a significant reduction in computational complexity. Thus, a subband adaptive filter offers an attractive alternative to the frequency-domain adaptive filter.

Frequency-domain adaptive filtering (FDAF), self-orthogonalizing adaptive filtering, and subband adaptive filtering, collectively, form *block adaptive filtering*, the study of which in this chapter is confined to real-valued data in order to simplify the presentation.

8.1 BLOCK-ADAPTIVE FILTERS: BASIC IDEAS

In a *block-adaptive filter*, depicted in Fig. 8.1, the incoming data sequence $u(n)$ is sectioned into L-point blocks by means of a serial-to-parallel converter, and the blocks of input data so produced are applied to a finite-duration impulse response (FIR) filter of length M, one block at a time. The tap weights of the filter are updated after the collection of each block of data samples, so that adaptation of the filter proceeds on a block-by-block basis rather than on a sample-by-sample basis as in the traditional LMS algorithm (Clark et al., 1981; Shynk, 1992).

Following the notation introduced in previous chapters, let

$$\mathbf{u}(n) = [u(n),\, u(n-1), \ldots,\, u(n-M+1)]^{\mathrm{T}} \tag{8.1}$$

denote the input signal vector at time n, where the superscript T denotes *transposition*. Correspondingly, let

$$\hat{\mathbf{w}}(n) = [\hat{w}_0(n),\, \hat{w}_1(n), \ldots,\, \hat{w}_{M-1}(n)]^{\mathrm{T}} \tag{8.2}$$

denote the tap-weight vector of the filter at time n. Let k refer to the *block index*, which is related to the original sample time n as

$$\begin{aligned} n &= kL + i, \qquad && i = 0, 1, \ldots, L-1, \\ & && k = 1, 2, \ldots, \end{aligned} \tag{8.3}$$

where L is the *block length*. The input data for block k is thus defined by the set $\{\mathbf{u}(kL+i)\}_{i=0}^{L-1}$, which is written in matrix form as follows:

$$\mathbf{A}^{\mathrm{T}}(k) = [\mathbf{u}(kL),\, \mathbf{u}(kL+1), \ldots, \mathbf{u}(kL+L-1)]. \tag{8.4}$$

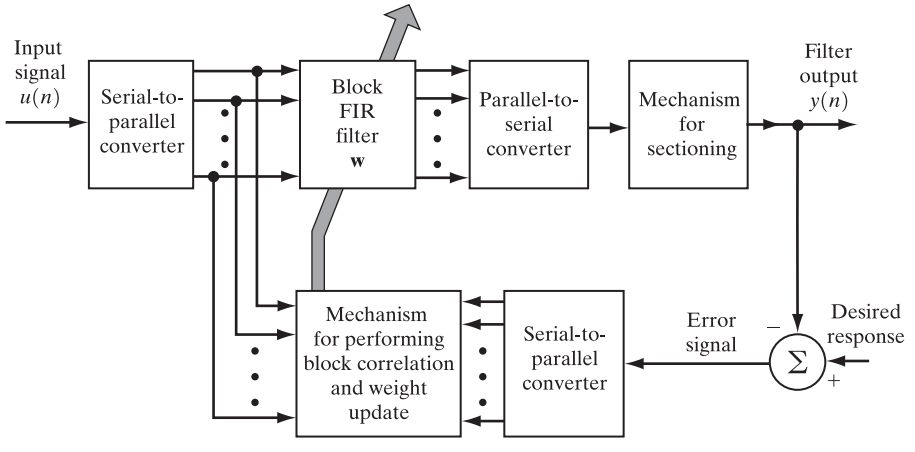


FIGURE 8.1 Block-adaptive filter.

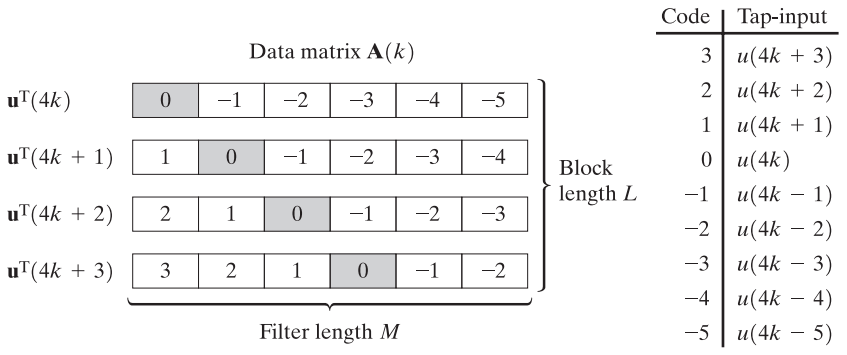


FIGURE 8.2 Illustrating the construction of data matrix $\mathbf{A}(k)$.

Over this block of input data, the tap-weight vector of the filter is held at the value $\hat{\mathbf{w}}(k)$, which is a rewrite of $\hat{\mathbf{w}}(n)$ for $n = k$. Figure 8.2 illustrates the construction of data matrix $\mathbf{A}(k)$ for filter length $M = 6$ and block length $L = 4$.

The output produced by the filter in response to the input signal vector $\mathbf{u}(kL + i)$ is defined by

$$\begin{aligned} y(kL + i) &= \hat{\mathbf{w}}^T(k)\mathbf{u}(kL + i) \\ &= \sum_{j=0}^{M-1} \hat{w}_j(k)u(kL + i - j), \qquad i = 0, 1, \ldots, L - 1. \end{aligned} \tag{8.5}$$

Let $d(kL + i)$ denote the corresponding value of the *desired response*. An *error signal*

$$e(kL + i) = d(kL + i) - y(kL + i) \tag{8.6}$$

is produced by comparing the filter output against the desired response. The error signal is thus permitted to vary at the sample rate, as in the traditional LMS algorithm. The error signal is sectioned into L-point blocks in a synchronous manner with the signal at the input end of the block-adaptive filter and is then used to compute the correction to be applied to the tap weights of the filter, as depicted in Fig. 8.1.

EXAMPLE 1

To illustrate the operation of the block-adaptive filter, consider the example of a filter for which the filter length M and block size L are both equal to 3. We may then express the output sequence computed by the filter for three consecutive data blocks, $k - 1$, k, and $k + 1$, as follows:

$$(k-1)\text{th block}\begin{cases} \begin{bmatrix} u(3k-3) & u(3k-4) & u(3k-5) \\ u(3k-2) & u(3k-3) & u(3k-4) \\ u(3k-1) & u(3k-2) & u(3k-3) \end{bmatrix} \begin{bmatrix} w_0(k-1) \\ w_1(k-1) \\ w_2(k-1) \end{bmatrix} = \begin{bmatrix} y(3k-3) \\ y(3k-2) \\ y(3k-1) \end{bmatrix}; \\ k\text{th block}\begin{cases} \begin{bmatrix} u(3k) & u(3k-1) & u(3k-2) \\ u(3k+1) & u(3k) & u(3k-1) \\ u(3k+2) & u(3k+1) & u(3k) \end{bmatrix} \begin{bmatrix} w_0(k) \\ w_1(k) \\ w_2(k) \end{bmatrix} = \begin{bmatrix} y(3k) \\ y(3k+1) \\ y(3k+2) \end{bmatrix}; \end{cases} \end{cases}$$

$$(k+1)\text{th block}\left\{\begin{bmatrix} u(3k+3) & u(3k+2) & u(3k+1) \\ u(3k+4) & u(3k+3) & u(3k+2) \\ u(3k+5) & u(3k+4) & u(3k+3) \end{bmatrix}\right.\begin{bmatrix} w_0(k+1) \\ w_1(k+1) \\ w_2(k+1) \end{bmatrix} = \begin{bmatrix} y(3k+3) \\ y(3k+4) \\ y(3k+5) \end{bmatrix}.$$

Note that the data matrix defined here is a Toeplitz matrix by virtue of the fact that the elements on any principal diagonal are all the same.

Block LMS Algorithm

From the derivation of the LMS algorithm presented in Chapter 5, we recall the following formula for the "adjustment" applied to the tap-weight vector from one adaptation cycle of the algorithm to the next (assuming real-valued data):

$$\begin{pmatrix}\text{adjustment to the} \\ \text{weight vector}\end{pmatrix} = \begin{pmatrix}\text{step-size} \\ \text{parameter}\end{pmatrix} \times \begin{pmatrix}\text{tap-input} \\ \text{vector}\end{pmatrix} \times (\text{error signal}).$$

Since, in the block LMS algorithm, the error signal is allowed to vary at the sampling rate, it follows that, for each block of data, we have different values of the error signal for use in the adaptive process. Accordingly, for the kth block, we may sum the product $\mathbf{u}(kL+i)e(kL+i)$ over all possible values of i and so define the following update equation for the tap-weight vector of the block LMS algorithm operating on real-valued data:

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + \mu\sum_{i=0}^{L-1}\mathbf{u}(kL+i)e(kL+i). \tag{8.7}$$

Here, μ is the step-size parameter. For convenience of presentation (which will become apparent in the next section), we rewrite Eq. (8.7) in the form

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + \mu\boldsymbol{\phi}(k). \tag{8.8}$$

The M-by-1 vector $\boldsymbol{\phi}(k)$ is a *cross-correlation* defined by

$$\begin{aligned}\boldsymbol{\phi}(k) &= \sum_{i=0}^{L-1}\mathbf{u}(kL+i)e(kL+i) \\ &= \mathbf{A}^{\mathrm{T}}(k)\mathbf{e}(k),\end{aligned} \tag{8.9}$$

where the L-by-M data matrix $\mathbf{A}(k)$ is defined in Eq. (8.4), and the L-by-1 vector

$$\mathbf{e}(k) = [e(kL), e(kL+1), \ldots, e(kL+L-1)]^{\mathrm{T}} \tag{8.10}$$

is the *error signal vector*.

A distinctive feature of the block LMS algorithm is that its design incorporates the *estimate* of the gradient vector,

$$\hat{\boldsymbol{\nabla}}(k) = -\frac{2}{L}\sum_{i=0}^{L-1}\mathbf{u}(kL+i)e(kL+i), \tag{8.11}$$

where the factor 2 is included to be consistent with the definition of the gradient vector used in Chapters 4 to 6, and the factor $1/L$ is included for $\hat{\boldsymbol{\nabla}}(k)$ to be an unbiased time average. Then we may reformulate the block LMS algorithm in terms of $\hat{\boldsymbol{\nabla}}(k)$ as

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) - \frac{1}{2}\mu_B \hat{\boldsymbol{\nabla}}(k), \tag{8.12}$$

where

$$\mu_B = L\mu. \tag{8.13}$$

The new constant μ_B may be viewed as the "effective" step-size parameter of the block LMS algorithm.

Convergence Properties of the Block LMS Algorithm

The block LMS algorithm has properties similar to those of the traditional LMS algorithm. The fundamental difference between them lies in the estimates of the gradient vector used in their respective implementations. Compared to the traditional LMS algorithm, we see from Eq. (8.11) that the block LMS algorithm uses a more *accurate* estimate of the gradient vector because of the time averaging, with the estimation accuracy increasing as the block size L is increased. However, this improvement does not imply faster adaptation, a fact that is revealed by examining the convergence properties of the block LMS algorithm.

We may proceed through a convergence analysis of the block LMS algorithm in a manner similar to that described in Chapter 6 for the traditional LMS algorithm. Indeed, such an analysis follows the same steps as those described there, except for a minor modification, namely, the summation of certain expectations over the index $i = 0, 1, \ldots, L-1$, which is related to the sample time n as in Eq. (8.3) and the use of which arises by virtue of Eq. (8.6). We may thus summarize the results of "small-step-size" statistical analysis of the block LMS algorithm as follows:

1. *Time constants.* The kth time constant of the block LMS algorithm is

$$\tau_{\text{mse,av}} = \frac{L}{2\mu_B \lambda_{\text{akv}}}, \qquad k = 1, 2, \ldots, M. \tag{8.14}$$

Here, λ_{akv} is the kth value of the M eigenvalues of the correlation matrix

$$\mathbf{R} = \mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{T}}(n)].$$

The corresponding formula for the kth time constant of the traditional LMS algorithm operating with step-size parameter μ is given by Eq. (6.105), which is reproduced here:

$$\tau_{\text{mse},k} = \frac{1}{2\mu\lambda_k}.$$

For given correlation matrix $\mathbf{R}$, the kth eigenvalue λ_k is the same for both the traditional LMS algorithm and the block LMS algorithm. Hence, comparing this formula with that of Eq. (8.14) and keeping the relation of Eq. (8.13) in mind, we find that for a given λ_k these two algorithms have the same kth time constant.

For the zero-order formula of Eq. (8.14) to hold, the effective step-size parameter μ_B of the block LMS algorithm has to be small compared to $2/\lambda_{\max}$, where $\lambda_{\max}$ is the largest eigenvalue of the correlation matrix $\mathbf{R}$. Suppose now that we have a requirement

for fast adaptation and therefore a small $\tau_{\text{mse},k}$, but the required block size L is so large that the value of μ_B calculated from Eq. (8.14) is correspondingly large. In such a situation, it is possible that the calculated value of μ_B is so large that the higher-order effects become significant enough for the use of the block LMS algorithm to be impractical because of instability problems.

2. *Misadjustment.* The misadjustment produced by the block LMS algorithm is

$$\mathcal{M} = \frac{\mu_B}{2L}\,\text{tr}[\mathbf{R}], \tag{8.15}$$

where tr[**R**] denotes the trace of the correlation matrix **R**. Here again, in light of Eq. (8.13), we find that the misadjustment of Eq. (8.15) is identical to that of Eq. (6.102) derived in Chapter 6 for the traditional LMS algorithm.

Choice of Block Size

Thus far in the discussion of block LMS algorithms, we have not imposed a restriction on the block size L in relation to the length M of the adaptive filter. In this context, we recognize three possible choices, each with its own practical implications:

1. $L = M$, which is the *optimal* choice from the viewpoint of computational complexity.
2. $L < M$, which offers the advantage of *reduced processing delay.* Moreover, by making the block size smaller than the filter length, we still have an adaptive filtering algorithm computationally more accurate than the traditional LMS algorithm.
3. $L > M$, which gives rise to *redundant operations in the adaptive process,* because the estimation of the gradient vector (computed over L points) now uses more information than the filter itself.

Henceforth, we confine our attention to the case of $L = M$, which is the preferred choice in most practical applications of block adaptive filtering.

When we speak of computational complexity, we mean the number of fast Fourier transformations and the size of each transformation that is needed for frequency-domain implementation of the block LMS algorithm.[1] This method of implementation is discussed next.

8.2 FAST BLOCK LMS ALGORITHM

Given an adaptive signal-processing application for which the block LMS algorithm is a satisfactory solution, the key question to be addressed is how to implement it in a computationally efficient manner. Referring to Eqs. (8.5) and (8.9), where

[1]Benesty et al. (2001) develop a general framework for frequency-domain adaptive filters, using a recursive least-squares criterion. Novel features of the theory presented therein include the following:

- The choice of block size is independent of the length of the adaptive filter.
- Various approximations applied to the theory lead to the fast block LMS algorithm, unconstrained block LMS algorithm, and multiple-delay adaptive filter.
- Some of these configurations are generalized to the multichannel scenario.

the computational burden of the block LMS algorithm lies, we observe the following:

- Equation (8.5) defines a *linear convolution* of the tap inputs and tap weights of the filter.
- Equation (8.9) defines a *linear correlation* between the tap inputs of the filter and the error signal.

From digital signal-processing theory, we know that the *fast Fourier transform (FFT) algorithm* provides a powerful tool for performing *fast convolution* and *fast correlation* (Oppenheim & Schafer, 1989). These observations point to a frequency-domain method for efficient implementation of the block LMS algorithm. Specifically, rather than performing the adaptation in the time domain as described in the previous section, the filter parameters are actually adapted in the frequency domain by using the FFT algorithm. The block LMS algorithm so implemented is referred to as the *fast block LMS algorithm*, which was developed independently by Clark et al. (1981, 1983) and Ferrara (1980).

From digital signal-processing theory, we also know that the overlap-save method and overlap-add method provide two efficient procedures for fast convolution—that is, the computation of linear convolution using the discrete Fourier transform (Oppenheim & Schafer, 1989). The overlap-save method is the more common one of the two for nonadaptive filtering. Also, it is noteworthy that although the filter can be implemented with any amount of overlap, the use of 50 percent overlap (i.e., block size equal to the number of weights) is the most efficient. Henceforth, we focus our attention on the overlap-save method with 50 percent overlap.[2]

According to the overlap-save method, the M tap weights of the filter are padded with an equal number of zeros, and an N-point FFT is used for the computation, where

$$N = 2M.$$

Thus, let the N-by-1 vector

$$\hat{\mathbf{W}}(k) = \text{FFT}\begin{bmatrix}\hat{\mathbf{w}}(k)\\ \mathbf{0}\end{bmatrix} \tag{8.16}$$

denote the FFT coefficients of the zero-padded, tap-weight vector $\hat{\mathbf{w}}(k)$. Here, $\mathbf{0}$ is the M-by-1 null vector and FFT [ ] denotes fast Fourier transformation. Note that the frequency-domain weight vector $\hat{\mathbf{W}}(k)$ is *twice* as long as the time-domain weight vector $\hat{\mathbf{w}}(k)$. Correspondingly, let

$$\mathbf{U}(k) = \text{diag}\{\text{FFT}\,[\underbrace{u(kM - M), \ldots, u(kM - 1)}_{(k-1)\text{th block}}, \underbrace{u(kM), \ldots, u(kM + M - 1)}_{k\text{th block}}]\} \tag{8.17}$$

denote an N-by-N diagonal matrix obtained by Fourier transforming two successive blocks of input data. We could use a vector to define the transformed version of the input signal vector $\mathbf{u}(M)$; however, for our present needs the matrix notation of Eq. (8.17) is

[2]In Sommen and Jayasinghe (1988), a simplified form of the overlap-add method is described, saving two inverse discrete Fourier transforms (DFTs).

considered to be more appropriate. Hence, applying the overlap-save method to the linear convolution of Eq. (8.5) yields the M-by-1 vector

$$\begin{aligned}\mathbf{y}^{\mathrm{T}}(k) &= [y(kM),\ y(kM+1),\ \ldots,\ y(kM+M-1)] \\ &= \text{last } M \text{ elements of IFFT}\,[\mathbf{U}(k)\hat{\mathbf{W}}(k)],\end{aligned} \tag{8.18}$$

where IFFT [ ] denotes inverse fast Fourier transformation. Only the last M elements in Eq. (8.18) are retained, because the first M elements correspond to a circular convolution.

Consider next the linear correlation of Eq. (8.9). For the kth block, define the M-by-1 desired response vector

$$\mathbf{d}(k) = [d(kM),\ d(kM+1), \ldots, d(kM+M-1)]^{\mathrm{T}} \tag{8.19}$$

and the corresponding M-by-1 error signal vector

$$\begin{aligned}\mathbf{e}(k) &= [e(kM),\ e(kM+1), \ldots, e(kM+M-1)]^{\mathrm{T}} \\ &= \mathbf{d}(k) - \mathbf{y}(k).\end{aligned} \tag{8.20}$$

Noting that, in implementing the linear convolution described in Eq. (8.18), the first M elements are discarded from the output, we may transform the error signal vector $\mathbf{e}(k)$ into the frequency domain as follows:

$$\mathbf{E}(k) = \text{FFT}\begin{bmatrix}\mathbf{0} \\ \mathbf{e}(k)\end{bmatrix}. \tag{8.21}$$

Next, recognizing that linear correlation is basically a "reversed" form of linear convolution, we find that applying the overlap-save method to the linear correlation of Eq. (8.9) yields

$$\boldsymbol{\phi}(k) = \text{first } M \text{ elements of IFFT}\,[\mathbf{U}^{\mathrm{H}}(k)\mathbf{E}(k)], \tag{8.22}$$

where the superscript H denotes *Hermitian transposition* (i.e., the operation of transposition combined with complex conjugation). Observe that, whereas in the case of linear convolution considered in Eq. (8.18) the first M elements are discarded, in the case of Eq. (8.22) the last M elements are discarded.

Finally, consider Eq. (8.8) for updating the tap-weight vector of the filter. Noting that in the definition of the frequency-domain weight vector [$\hat{\mathbf{W}}(k)$ of Eq. (8.16)] the time-domain weight vector $\hat{\mathbf{w}}(k)$ is followed by M zeros, we may correspondingly transform Eq. (8.8) into the frequency domain as follows:

$$\hat{\mathbf{W}}(k+1) = \hat{\mathbf{W}}(k) + \mu\text{FFT}\begin{bmatrix}\boldsymbol{\phi}(k) \\ \mathbf{0}\end{bmatrix}. \tag{8.23}$$

Equations (8.16) through (8.23), in that order, define the fast block LMS algorithm. Figure 8.3 shows a signal-flow graph representation of the fast block LMS algorithm (Shynk, 1992). This algorithm represents a precise frequency-domain implementation of the block LMS algorithm. As such, its convergence properties are identical to those of the block LMS algorithm discussed in Section 8.1.

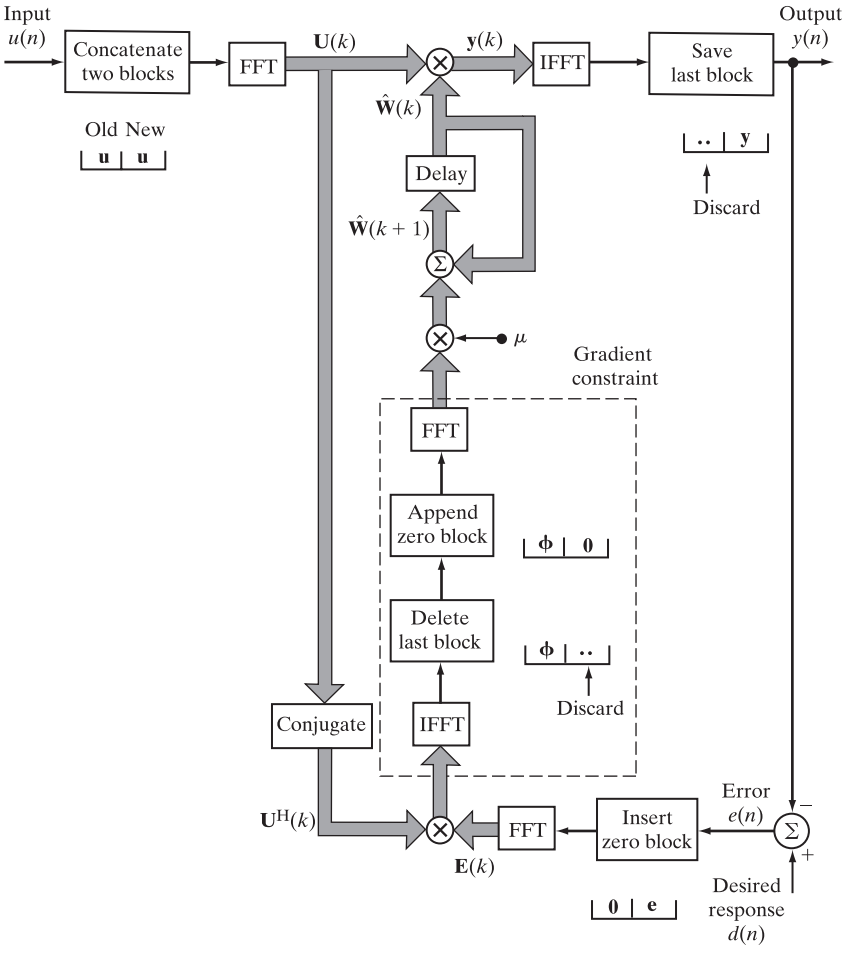


FIGURE 8.3 Overlap-save FDAF. This frequency-domain adaptive filter is based on the overlap-save sectioning procedure for implementing linear convolutions and linear correlations. (Taken from *IEEE SP Magazine* with permission of the IEEE.)

Computational Complexity

The computational complexity of the fast block LMS algorithm operating in the frequency domain is now compared with that of the traditional LMS algorithm operating in the time domain. The comparison is based on a count of the total number of multiplications involved in each of these two implementations for a block size M. Although in an actual implementation there are other factors to be considered (e.g., the number of additions and the storage requirements), the use of multiplications provides a reasonably accurate basis for comparing the computational complexity of the two algorithms.

Consider first the traditional LMS algorithm, with M tap weights operating on real data. In this case, M multiplications are performed to compute the output, and a further M multiplications are performed to update the tap weights, for a total of $2M$ multiplications per adaptation cycle. Hence, for a block of M output samples, the total number of multiplications is $2M^2$.

Now consider the fast block LMS algorithm. Each N-point FFT (and IFFT) requires approximately $N \log_2 N$ real multiplications (Oppenheim & Schafer, 1989), where $N = 2M$. According to the structure of the fast block LMS algorithm shown in Fig. 8.3, there are five frequency transformations performed, which therefore account for $5N \log_2 N$ multiplications. In addition, the computation of the frequency-domain output vector requires $4N$ multiplications, and so does the computation of the cross-correlations relating to the gradient vector estimation. Hence, the total number of multiplications performed in the fast block LMS algorithm is

$$\begin{aligned} 5N \log_2 N + 8N &= 10M \log_2(2M) + 16M \\ &= 10M \log_2 M + 26M. \end{aligned}$$

The *complexity ratio* for the fast block LMS to the traditional LMS algorithm is therefore (Shynk, 1992)

$$\begin{aligned} \text{complexity ratio} &= \frac{10M \log_2 M + 26M}{2M^2} \\ &= \frac{5 \log_2 M + 13}{M}. \end{aligned}$$

For example, for $M = 1024$, the use of this equation shows that the fast block LMS algorithm is roughly 16 times faster than the traditional LMS algorithm in computational terms. We may therefore make the following statement:

> The fast block LMS algorithm provides a computational cost per adaptation cycle below that of the traditional LMS algorithm.

Improvement in Convergence Rate

Basically, the fast block LMS algorithm is an efficient frequency-domain implementation of the block LMS algorithm. It therefore follows that both algorithms have the same convergence properties: necessary condition for convergence, rate of convergence, and misadjustment. However, the convergence rate of the fast block LMS algorithm can be improved by assigning a different step size to each adjustable weight without affecting the minimum mean-square error (Sommen et al., 1987; Lee & Un, 1989). In particular, the use of the overlap-save method is not only responsible for reducing the computational complexity of the implementation, but it also provides a practical basis for improving the convergence rate of the fast block LMS algorithm. The improvement is achieved by compensating for variations in the average signal power across the individual frequency bins. Specifically, we make all the modes of the adaptive process

converge essentially at the same rate by assigning to each weight an individual step-size parameter of its own, defined by

$$\mu_i = \frac{\alpha}{P_i}, \qquad i = 0, 1, \ldots, 2M - 1, \tag{8.24}$$

where α is a constant and P_i is an estimate of the average power in the ith frequency bin. Note that, for real signals, we have symmetry in the frequency domain, which means that the first M values of P_i in Eq. (8.24) suffice.

The condition of Eq. (8.24) applies when the environment in which the fast block LMS algorithm operates is wide-sense stationary. When, however, the environment is nonstationary, or when an estimate of the average input power in each bin is not available, we may use the following simple recursion (based on the idea of convex combination) to estimate the performance of the fast LMS block algorithm (Griffiths, 1978; Shynk, 1992):

$$P_i(k) = \gamma P_i(k - 1) + (1 - \gamma)|U_i(k)|^2, \qquad i = 0, 1, \ldots, 2M - 1. \tag{8.25}$$

Here, $U_i(k)$ is the input applied to the ith weight in the fast block LMS algorithm at time k, γ is a constant chosen in the range $0 < \gamma < 1$, and the parameter γ is a *forgetting factor* that controls the effective "memory" of the iterative process described in the equation. In particular, we may express the input power $P_i(k)$ as an exponentially weighted sum of the squares of the magnitude of the input values:

$$P_i(k) = (1 - \gamma)\sum_{l=0}^{\infty} \gamma^l |U_i(k - l)|^2. \tag{8.26}$$

Thus, given the estimate $P_i(k)$ of the average signal power in the ith bin, the step-size parameter μ is replaced by an M-by-M diagonal matrix in accordance with Eq. (8.24) as

$$\boldsymbol{\mu}(k) = \alpha \mathbf{D}(k), \tag{8.27}$$

where

$$\mathbf{D}(k) = \text{diag}[P_0^{-1}(k), P_1^{-1}(k), \ldots, P_{2M-1}^{-1}(k)]. \tag{8.28}$$

Correspondingly, the fast block LMS algorithm is modified as follows (Ferrara, 1985; Shynk, 1992):

1. In Eq. (8.22), involving the computation of the cross-correlation vector $\boldsymbol{\phi}(k)$, the product term $\mathbf{U}^{\text{H}}(k)\mathbf{E}(k)$ is replaced by $\mathbf{D}(k)\mathbf{U}^{\text{H}}(k)\mathbf{E}(k)$, yielding

$$\boldsymbol{\phi}(k) = \text{first } M \text{ elements of IFFT } [\mathbf{D}(k)\mathbf{U}^{\text{H}}(k)\mathbf{E}(k)]. \tag{8.29}$$

2. In Eq. (8.23), μ is replaced by the constant α; otherwise, the computation of the frequency-domain weight vector $\hat{\mathbf{W}}(k)$ is the same as before.

Table 8.1 presents a summary of the fast block LMS algorithm, incorporating the modifications described herein (Shynk, 1992).

TABLE 8.1 Summary of the Fast Block LMS Algorithm Based on Overlap-Save Sectioning (Assuming Real-Valued Data)

Initialization:

$$\hat{\mathbf{W}}(0) = 2M\text{-by-1 null vector}$$
$$P_i(0) = \delta_i, \text{ where the } \delta_i \text{ are small positive constants and } i = 0, \ldots, 2M-1$$

Notations:

$\mathbf{0} = M$-by-1 null vector
FFT = fast Fourier transformation
IFFT = inverse fast Fourier transformation
α = adaptation constant

Computation: For each new block of M input samples, compute:

Filtering:

$$\mathbf{U}(k) = \text{diag}\{\text{FFT}\,[u(kM-M), \ldots, u(kM-1), u(kM), \ldots, u(kM+M-1)]^{\mathrm{T}}\}$$
$$\mathbf{y}^{\mathrm{T}}(k) = \text{last } M \text{ elements of IFFT } [\mathbf{U}(k)\hat{\mathbf{W}}(k)].$$

Error estimation:

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{y}(k)$$
$$\mathbf{E}(k) = \text{FFT}\begin{bmatrix}\mathbf{0}\\ \mathbf{e}(k)\end{bmatrix}.$$

Signal-power estimation:

$$P_i(k) = \gamma P_i(k-1) + (1-\gamma)|U_i(k)|^2, \qquad i = 0, 1, \ldots, 2M-1$$
$$\mathbf{D}(k) = \text{diag}[P_0^{-1}(k), P_1^{-1}(k), \ldots, P_{2M-1}^{-1}(k)].$$

Tap-weight adaptation:

$$\boldsymbol{\phi}(k) = \text{first } M \text{ elements of IFFT}[\mathbf{D}(k)\mathbf{U}^{\mathrm{H}}(k)\mathbf{E}(k)]$$
$$\hat{\mathbf{W}}(k+1) = \hat{\mathbf{W}}(k) + \alpha\,\text{FFT}\begin{bmatrix}\boldsymbol{\phi}(k)\\ \mathbf{0}\end{bmatrix}.$$

8.3 UNCONSTRAINED FREQUENCY-DOMAIN ADAPTIVE FILTERS

The fast block LMS algorithm described by the signal-flow graph of Fig. 8.3 may be viewed as a constrained form of FDAF. Specifically, two of the five FFTs involved in the operation of the algorithm are needed to impose a *time-domain constraint* for the purpose of performing a linear correlation as specified in Eq. (8.9). The time-domain constraint consists of the following operations:

- Discarding the last M elements of the inverse FFT of $\mathbf{U}^{\mathrm{H}}(k)\mathbf{E}(k)$, as described in Eq. (8.22).
- Replacing the elements so discarded by a block of M zeros before reapplying the FFT, as described in Eq. (8.23).

The combination of operations described herein is contained inside the dashed rectangle of Fig. 8.3; this combination is referred to as a *gradient constraint*, in recognition of the fact that it is involved in computing an estimate of the gradient vector. Note that the gradient constraint is actually a time-domain constraint which basically ensures that

the $2M$ frequency-domain weights correspond to only M time-domain weights. This is the reason why a zero block is appended in the gradient constraint in the figure.

In the *unconstrained frequency-domain adaptive filter* (Mansour & Gray, 1982), the gradient constraint is removed completely from the signal-flow graph of Fig. 8.3. The net result is a simpler implementation that involves only three FFTs. Thus, the combination of Eqs. (8.22) and (8.23) in the fast block LMS algorithm is now replaced by the much simpler algorithm

$$\hat{\mathbf{W}}(k+1) = \hat{\mathbf{W}}(k) + \mu \mathbf{U}^{\mathrm{H}}(k)\mathbf{E}(k). \tag{8.30}$$

It is important to note, however, that the estimate of the gradient vector computed here no longer corresponds to a linear correlation as specified in Eq. (8.9); rather, we now have a circular correlation.

Consequently, we find that, in general, the unconstrained FDAF algorithm of Eq. (8.30) deviates from the fast block LMS algorithm, in that the tap-weight vector no longer approaches the Wiener solution as the number of block adaptation cycles approaches infinity (Sommen et al., 1987; Lee & Un, 1989; Shynk, 1992). Another point to note is that, although the convergence rate of the unconstrained FDAF algorithm is increased with time-varying step sizes, the improvement is offset by a worsening of the misadjustment. Indeed, according to Lee and Un (1989), the unconstrained algorithm requires twice as many adaptation cycles as the constrained algorithm to produce the same level of misadjustment.

8.4 SELF-ORTHOGONALIZING ADAPTIVE FILTERS

In the previous sections, we addressed the issue of how to use frequency-domain techniques to improve the computational accuracy of the LMS algorithm when the application of interest requires a long filter memory. In this section, we consider another important adaptive filtering issue, namely, that of improving the convergence properties of the LMS algorithm. This improvement is, however, attained at the cost of an increase in computational complexity.

To motivate the discussion, consider an input signal vector $\mathbf{u}(n)$ characterized by the correlation matrix $\mathbf{R}$. The *self-orthogonalizing adaptive filtering algorithm* for such a wide-sense stationary environment is described by (Chang, 1971; Cowan, 1987)

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \alpha \mathbf{R}^{-1}\mathbf{u}(n)e(n), \tag{8.31}$$

where $\mathbf{R}^{-1}$ is the inverse of the correlation matrix $\mathbf{R}$ and $e(n)$ is the error signal defined in the usual way. [Equation (8.31) is related to Newton's method, which was discussed in Section (4.6).] The constant α lies in the range $0 < \alpha < 1$; according to Cowan (1987), it may be set at the value

$$\alpha = \frac{1}{2M}, \tag{8.32}$$

where M is the filter length. An important property of the self-organizing filtering algorithm of Eq. (8.31) is that, in theory, it guarantees a constant rate of convergence, irrespective of the input statistics.

To prove this useful property, define the weight-error vector

$$\boldsymbol{\varepsilon}(n) = \mathbf{w}_o - \hat{\mathbf{w}}(n), \tag{8.33}$$

where the weight vector $\mathbf{w}_o$ is the Wiener solution. We may rewrite the algorithm of Eq. (8.31) in terms of $\boldsymbol{\varepsilon}(n)$ as

$$\boldsymbol{\varepsilon}(n+1) = \big(\mathbf{I} - \alpha\mathbf{R}^{-1}\mathbf{u}(n)\mathbf{u}^{\mathrm{T}}(n)\big)\boldsymbol{\varepsilon}(n) - \alpha\mathbf{R}^{-1}\mathbf{u}(n)e_o(n), \tag{8.34}$$

where $\mathbf{I}$ is the identity matrix and $e_o(n)$ is the optimum value of the error signal that is produced by the Wiener solution. Applying the statistical expectation operator to both sides of Eq. (8.34) and invoking Kushner's direct-averaging method under the assumption of small α (see Section 6.4), we obtain the following result:

$$\mathbb{E}[\boldsymbol{\varepsilon}(n+1)] \approx \big(\mathbf{I} - \alpha\mathbf{R}^{-1}\mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathbf{T}}(n)]\big)\mathbb{E}[\boldsymbol{\varepsilon}(n)] - \alpha\mathbf{R}^{-1}\mathbb{E}[\mathbf{u}(n)e_o(n)]. \tag{8.35}$$

We now recognize the following points (for real-valued data):

- From the definition of the correlation matrix for a wide-sense stationary process, we have

$$\mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{T}}(n)] = \mathbf{R}.$$

- From the principle of orthogonality, we have (see Section 2.2)

$$\mathbb{E}[\mathbf{u}(n)e_o(n)] = \mathbf{0}.$$

Accordingly, we may simplify Eq. (8.35) as follows:

$$\begin{aligned}\mathbb{E}[\boldsymbol{\varepsilon}(n+1)] &= \big(\mathbf{I} - \alpha\mathbf{R}^{-1}\mathbf{R}\big)\mathbb{E}[\boldsymbol{\varepsilon}(n)] \\ &= (1-\alpha)\mathbb{E}[\boldsymbol{\varepsilon}(n)].\end{aligned} \tag{8.36}$$

Equation (8.36) represents an autonomous first-order difference equation, the solution of which is

$$\mathbb{E}[\boldsymbol{\varepsilon}(n)] = (1-\alpha)^n\mathbb{E}[\boldsymbol{\varepsilon}(0)], \tag{8.37}$$

where $\boldsymbol{\varepsilon}(0)$ is the initial value of the weight-error vector. Hence, with the value of α lying in the range $0 < \alpha < 1$, we may write

$$\lim_{n\to\infty}\mathbb{E}[\boldsymbol{\varepsilon}(n)] = \mathbf{0} \tag{8.38}$$

or, equivalently,

$$\lim_{n\to\infty}\mathbb{E}[\hat{\mathbf{w}}(n)] = \mathbf{w}_o. \tag{8.39}$$

Most importantly, we note from Eq. (8.37) that the rate of convergence is completely independent of the input statistics, as stated previously.

EXAMPLE 2 White Noise Input

To illustrate the convergence properties of the self-organizing adaptive filtering algorithm, consider the case of a white-noise input, whose correlation matrix is defined by

$$\mathbf{R} = \sigma^2\mathbf{I}, \tag{8.40}$$

where σ^2 is the noise variance and $\mathbf{I}$ is the identity matrix. For this input, the use of Eq. (8.31) (with $\alpha = 1/2M$) yields

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{1}{2M\sigma^2}\mathbf{u}(n)e(n). \tag{8.41}$$

This algorithm is recognized as the traditional LMS algorithm with a step-size parameter defined by

$$\mu = \frac{1}{2M\sigma^2}. \tag{8.42}$$

In other words, for the special case of a white noise sequence characterized by an eigenvalue spread of unity, the traditional LMS algorithm behaves in the same way as the self-orthogonalizing adaptive filtering algorithm.

Two-Stage Adaptive Filter

This last example suggests that we may mechanize a self-orthogonalizing adaptive filter for an arbitrary environment by proceeding in two stages (Narayan et al., 1983; Cowan & Grant, 1985):

1. The input vector $\mathbf{u}(n)$ is transformed into a corresponding vector of uncorrelated variables.
2. The transformed vector is used as the input to an LMS algorithm.

From the discussion presented on eigenanalysis in Appendix E, we note that, in theory, the first objective may be realized by using the *Karhunen–Loève transform (KLT)*. Given an input vector $\mathbf{u}(n)$ of zero mean, drawn from a wide-sense stationary process, the ith output of the KLT is defined (for real-valued data) by

$$v_i(n) = \mathbf{q}_i^{\mathrm{T}}\mathbf{u}(n), \qquad i = 0, 1, \ldots, M-1, \tag{8.43}$$

where $\mathbf{q}_i$ is the eigenvector associated with the ith eigenvalue λ_i belonging to the correlation matrix $\mathbf{R}$ of the input vector $\mathbf{u}(n)$. The individual outputs of the KLT are zero-mean, uncorrelated variables, as shown by

$$\mathbb{E}[v_i(n)v_j(n)] = \begin{cases} \lambda_i, & j = i \\ 0, & j \neq i \end{cases}. \tag{8.44}$$

Accordingly, we may express the correlation matrix of the M-by-1 vector $\mathbf{v}(n)$ produced by the KLT as the diagonal matrix

$$\begin{aligned} \mathbf{\Lambda} &= \mathbb{E}[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)] \\ &= \mathrm{diag}[\lambda_0, \lambda_1, \ldots, \lambda_{M-1}]. \end{aligned} \tag{8.45}$$

The inverse of $\mathbf{\Lambda}$ is also a diagonal matrix:

$$\mathbf{\Lambda}^{-1} = \mathrm{diag}\left[\lambda_0^{-1}, \lambda_1^{-1}, \ldots, \lambda_{M-1}^{-1}\right]. \tag{8.46}$$

Consider now the self-orthogonalizing adaptive filtering algorithm of Eq. (8.31) with the transformed vector $\mathbf{v}(n)$ and its inverse correlation matrix $\mathbf{\Lambda}^{-1}$ used in place of $\mathbf{u}(n)$ and $\mathbf{R}^{-1}$, respectively. Under these new circumstances, Eq. (8.31) takes the form

$$\hat{\mathbf{w}}(n + 1) = \hat{\mathbf{w}}(n) + \alpha\mathbf{\Lambda}^{-1}\mathbf{v}(n)e(n), \tag{8.47}$$

the ith element of which may written as

$$\hat{w}_i(n + 1) = \hat{w}_i(n) + \frac{\alpha}{\lambda_i} v_i(n)e(n), \qquad i = 0, 1, \ldots, M - 1. \tag{8.48}$$

Note that the meaning of $\hat{\mathbf{w}}(n)$ in Eq. (8.47) is different from that in Eq. (8.31) because of the transformed input vector $\mathbf{v}(n)$. Equation (8.48) is immediately recognized as a *new* normalized form of the LMS algorithm. Normalization here means that each tap-weight is assigned its own step-size parameter that is related to the corresponding eigenvalue of the correlation matrix of the original input vector $\mathbf{u}(n)$. Thus, Eq. (8.48) takes care of the second point mentioned at the beginning of this subsection. Note, however, that the algorithm described herein is different from the traditional normalized LMS algorithm discussed in Chapter 7.

The KLT is a signal-dependent transformation, the implementation of which requires the estimation of the correlation matrix of the input vector, the diagonalization of this matrix, and the construction of the required basis vectors. These computations make the KLT impractical for real-time applications. Fortunately, the *discrete cosine transform* (*DCT*) provides a predetermined set of basis vectors that, together, are a good approximation of the KLT. Indeed, for a stationary zero-mean, first-order Markov process that is deemed to be sufficiently general in signal-processing studies, the DCT is asymptotically equivalent to the KLT,[3] both as the sequence length increases and as the adjacent correlation coefficient tends toward unity (Rao & Yip, 1990); the *adjacent correlation coefficient* of a stochastic process is defined as the autocorrelation function of the process for a unit lag, divided by the autocorrelation function of the process for zero lag (i.e., the mean-square value). Whereas the KLT is signal dependent, the DCT is signal independent and can therefore be implemented in a computationally efficient manner. The DCT basis vectors are a good approximation to the KLT for some signals.

We are now equipped with the tools we need to formulate a practical approximation to the self-orthogonalizing adaptive filter that combines the desirable properties of the DCT with those of the LMS algorithm. Figure 8.4 shows a block diagram of such a filter. The filter consists of two stages, with stage 1 providing the implementation of a *sliding* DCT algorithm and stage 2 implementing a *normalized* version of the LMS algorithm (Beaufays & Widrow, 1994; Beaufays, 1995a). In effect, stage 1 acts as a preprocessor that performs the orthogonalization of the input vector, albeit in an approximate manner.

[3]Grenander and Szegö (1958) and Gray (1972). Note that the asymptotic eigenvalue spread for the DFT with fast-order Markov inputs is much worse than that for the DCT, which makes the DCT the preferred approximation to the KLT. Specifically, for such inputs, the asymptotic eigenvalue spread is equal to $[(1 + \rho)/(1 - \rho)]$ for the DFT versus $(1 + \rho)$ for the DCT, where ρ is the adjacent correlation coefficient of the input (Beaufays, 1995a).

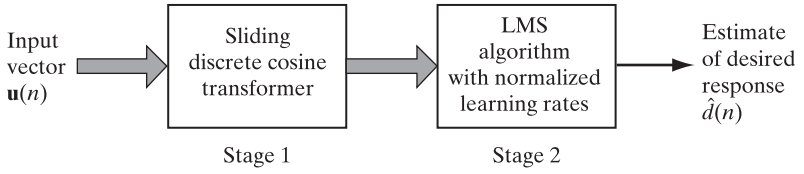


FIGURE 8.4 Block diagram of the DCT-LMS algorithm.

Sliding DCT

The DCT we have in mind for our present application uses a sliding window, with the computation being performed for *each* new input sample. This, in turn, enables the LMS algorithm (following the DCT) to operate at the incoming data rate as in its traditional form. Thus, unlike the fast block LMS algorithm, the FDAF algorithm described here is a nonblock algorithm and therefore not as computationally accurate.

From Fourier transform theory, we know that the discrete Fourier transform of an even function results in the discrete cosine transform. We may exploit this simple property to develop an efficient algorithm for computing the sliding DCT. To proceed, consider a sequence of M samples denoted by $u(n), u(n-1), \ldots, u(n-M+1)$. We may construct an extended sequence $a(i)$ that is symmetric about the point $n - M + 1/2$ as follows (see Fig. 8.5):

$$a(i) = \begin{cases} u(i), & i = n, n-1, \ldots, n-M+1 \\ u(-i+2n-2M+1), & i = n-M, n-M-1, \ldots, n-2M+1 \end{cases}. \tag{8.49}$$

For convenience of presentation, we define

$$W_{2M} = \exp\left(-\frac{\mathrm{j}2\pi}{2M}\right). \tag{8.50}$$

The mth element of the $2M$-point discrete Fourier transform (DFT) of the extended sequence in Eq. (8.49) at time n is defined by

$$A_m(n) = \sum_{i=n-2M+1}^{n} a(i) W_{2M}^{m(n-i)}. \tag{8.51}$$

Using Eq. (8.49) in Eq. (8.51), we may write

$$\begin{aligned} A_m(n) &= \sum_{i=n-M+1}^{n} a(i) W_{2M}^{m(n-i)} + \sum_{i=n-2M+1}^{n-M} a(i) W_{2M}^{m(n-i)} \\ &= \sum_{i=n-M+1}^{n} u(i) W_{2M}^{m(n-i)} + \sum_{i=n-2M+1}^{n-M} u(-i+2n-2M+1) W_{2M}^{m(n-i)} \\ &= \sum_{i=n-M+1}^{n} u(i) W_{2m}^{m(n-i)} + \sum_{i=n-M+1}^{n} u(i) W_{2M}^{m(i-n+2M-1)}. \end{aligned} \tag{8.52}$$

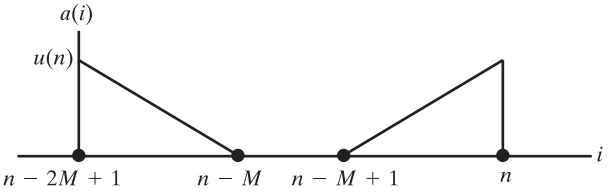


FIGURE 8.5 Construction of the extended sequence $a(i)$.

Factoring out the term $W_{2M}^{m(M-1/2)}$ and combining the two summations, we may redefine

$$\begin{aligned} A_m(n) &= W_{2M}^{m(M-1/2)} \sum_{i=n-M+1}^{n} u(i)\left(W_{2M}^{-m(i-n+M-1/2)} + W_{2M}^{m(i-n+M-1/2)} \right) \\ &= 2(-1)^m W_{2M}^{-m/2} \sum_{i=n-M+1}^{n} u(i) \cos\left(\frac{m(i-n+M-1/2)\pi}{M} \right), \end{aligned} \tag{8.53}$$

where, in the last line, we have used the definition of W_{2M} and Euler's formula for a cosine function. Except for a scaling factor, the summation in Eq. (8.53) is recognized as the DCT of the sequence $u(n)$ at time n; specifically, we have

$$C_m(n) = k_m \sum_{i=n-M+1}^{n} u(i) \cos\left(\frac{m(i-n+M-1/2)\pi}{M} \right), \tag{8.54}$$

where the constant

$$k_m = \begin{cases} 1/\sqrt{2}, & m = 0 \\ 1, & \text{otherwise} \end{cases}. \tag{8.55}$$

Accordingly, in light of Eqs. (8.53) and (8.54), the DCT of the sequence $u(n)$ is related to the DFT of the extended sequence $a(n)$ as follows:

$$C_m(n) = \frac{1}{2} k_m (-1)^m W_{2M}^{m/2} A_m(n), \qquad m = 0, 1, \ldots, M-1. \tag{8.56}$$

The DFT of the extended sequence $a(n)$ given in Eq. (8.52) may be viewed as the sum of two complementary DFTs:

$$A_m(n) = A_m^{(1)}(n) + A_m^{(2)}(n). \tag{8.57}$$

Here,

$$A_m^{(1)}(n) = \sum_{i=n-M+1}^{n} u(i) W_{2M}^{m(n-i)} \tag{8.58}$$

and

$$A_m^{(2)}(n) = \sum_{i=n-M+1}^{n} u(i) W_{2M}^{m(i-n+2M-1)}. \tag{8.59}$$

Consider first the DFT, $A_m^{(1)}(n)$. Separating out the sample $u(n)$, we may rewrite this DFT (computed at time n) as

$$A_m^{(1)}(n) = u(n) + \sum_{i=n-M+1}^{n-1} u(i) W_{2M}^{m(n-i)}. \tag{8.60}$$

Next, we note from Eq. (8.58) that the previous value of the DFT, computed at time $n-1$, is given by

$$\begin{aligned} A_m^{(1)}(n-1) &= \sum_{i=n-M}^{n-1} u(i) W_{2M}^{m(n-1-i)} \\ &= (-1)^m W_{2M}^{-m} u(n-M) + W_{2M}^{-m} \sum_{i=n-M+1}^{n} u(i) W_{2M}^{m(n-i)}, \end{aligned} \tag{8.61}$$

where, in the first term of the last line, we have used the fact that

$$W_{2M}^{mM} = e^{-\mathrm{j}m\pi} = (-1)^m.$$

Multiplying Eq. (8.61) by the factor W_{2M}^{m} and subtracting the result from Eq. (8.60), we get (after rearranging terms)

$$A_m^{(1)}(n) = W_{2M}^{m} A_m^{(1)}(n-1) + u(n) - (-1)^m u(n-M), \quad m = 0, 1, \ldots, M-1. \tag{8.62}$$

Equation (8.62) is a first-order difference equation that may be used to update the computation of $A_m^{(1)}(n)$, given its previous value $A_m^{(1)}(n-1)$, the new sample $u(n)$, and the very old sample $u(n-M)$.

Consider next the recursive computation of the second DFT, $A_m^{(2)}(n)$, defined in Eq. (8.59). We recognize that $W_{2M}^{2mM} = 1$ for all integer m. Hence, separating out the term that involves the sample $u(n)$, we may express this DFT in the form

$$A_m^{(2)}(n) = W_{2M}^{-m} u(n) + W_{2M}^{-m} \sum_{i=n-M+1}^{n-1} u(i) W_{2M}^{m(i-n)}. \tag{8.63}$$

Now, using Eq. (8.59) to evaluate this second DFT at time $n-1$ and then proceeding to separate out the term involving the sample $u(n-M)$, we may write

$$\begin{aligned} A_m^{(2)}(n-1) &= \sum_{i=n-M}^{n-1} u(i) W_{2M}^{m(i-n)} \\ &= W_{2M}^{mM} u(n-M) + \sum_{i=n-M+1}^{n-1} u(i) W_{2M}^{m(i-n)} \\ &= (-1)^m u(n-M) + \sum_{i=n-M+1}^{n-1} u(i) W_{2M}^{m(i-n)}. \end{aligned} \tag{8.64}$$

Multiplying Eq. (8.64) by the factor W_{2M}^{-m} and then subtracting the result from Eq. (8.63), we get (after rearranging terms)

$$A_m^{(2)}(n) = W_{2M}^{-m} A_m^{(2)}(n-1) + W_{2M}^{-m}\big(u(n) - (-1)^m u(n-M)\big). \tag{8.65}$$

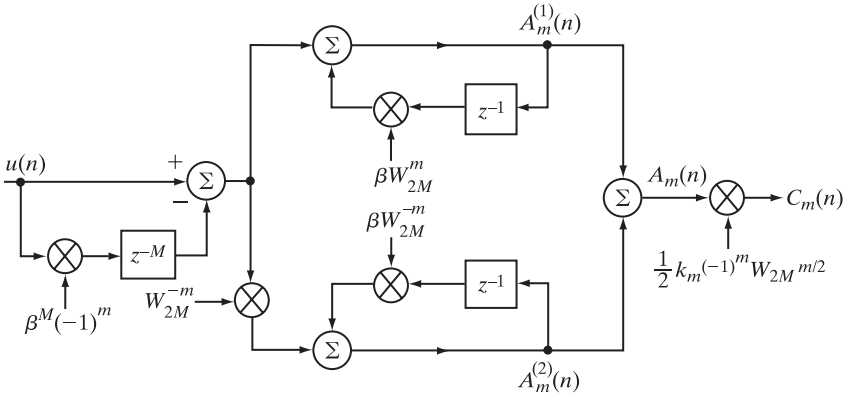


FIGURE 8.6 Indirect computation of the sliding discrete cosine transform.

Finally, using Eqs. (8.56), (8.57), (8.62), and (8.65), we may construct the block diagram shown in Fig. 8.6 for the recursive computation of the discrete cosine transform $C_m(n)$ of the sequence $u(n)$. The construction is simplified by noting the following two points:

1. The operations involving the present sample $u(n)$ and the old sample $u(n - M)$ are common to the computations of both discrete Fourier transforms $A_m^{(1)}(n)$ and $A_m^{(2)}(n)$—hence the common front end of Fig. 8.6.
2. The operator z^{-M} in the forward path in the figure is multiplied by $\beta^M(-1)^m$, where β is a new parameter. In contrast, the operator z^{-1} inside each of the two feedback loops in the figure is multiplied by β. The reason for including this new parameter is explained shortly.

The discrete-time network of Fig. 8.6 is called a *frequency-sampling filter*. It exhibits a form of structural symmetry that is inherited from the mathematical symmetry built into the definition of the discrete cosine transform.

The transfer function of the filter shown in Fig. 8.6 from the input $u(n)$ to the mth DCT output, $C_m(n)$, is given (with $\beta = 1$) by

$$H_m(z) = \frac{1}{2}\, k_m \left(\exp\left(-\frac{\mathrm{j}m\pi}{2M}\right) \frac{(-1)^m - z^{-M}}{1 - \exp\left(\dfrac{\mathrm{j}m\pi}{M}\right) z^{-1}} + \exp\left(\frac{\mathrm{j}m\pi}{2M}\right) \frac{(-1)^m - z^{-M}}{1 - \exp\left(-\dfrac{\mathrm{j}m\pi}{M}\right) z^{-1}} \right). \tag{8.66}$$

The common numerator of Eq. (8.66), namely, the factor $(-1)^m - z^{-M}$ represents a set of zeros that are uniformly spaced around the unit circle in the z-plane. These zeros are given by

$$z_m = \exp\left(\frac{\mathrm{j}\pi m}{M}\right), \qquad m = 0, \pm 1, \ldots, \pm(M - 1). \tag{8.67}$$

The first partial fraction in Eq. (8.66) has a single pole at $z = \exp(\mathrm{j}m\pi/M)$, whereas the second partial fraction has a single pole at $z_m = \exp(-\mathrm{j}m\pi/M)$. Accordingly, each of

these poles exactly cancels a particular zero of the numerator term. The net result is that the filter structure of Fig. 8.6 is equivalent to two banks of narrowband all-zero filters operating in parallel, with each filter bank corresponding to the M bins of the DCT. Figure 8.7(a) shows the frequency responses of the frequency-sampling filters pertaining to two adjacent bins of the DCT represented by the coefficients $m = 0$ and $m = 1$, and Fig. 8.7(b) shows the corresponding impulse responses of the filters for $M = 8$ bins.

With $\beta = 1$, the frequency-sampling filters described herein are "marginally" stable, because, for each bin of the DCT, the poles of the two feedback paths in Fig. 8.6 lie exactly on the unit circle, and round-off errors (however small) may give rise to instability by pushing one or the other (or both) of these poles outside the unit circle. This problem is alleviated by shifting the zeros of the forward path and the poles of the feedback paths slightly inside the unit circle (Shynk & Widrow, 1986)—hence the inclusion of parameter β in the figure, with $0 < \beta < 1$. For example, with $\beta = 0.99$, all of the poles and zeros of the two terms in the partial-fraction expansion of the transfer function $H_m(z)$ of a frequency-sampling filter, as in Eq. (8.66), are now made to lie on a circle with radius $\beta = 0.99$; the stability of the frequency-sampling filters is thereby ensured, even if exact pole–zero cancellations are not realized (Shynk, 1992).

Eigenvalue Estimation

The only issue that remains to be considered in the design of the DCT-LMS algorithm is how to estimate the eigenvalues of the correlation matrix $\mathbf{R}$ of the input vector $\mathbf{u}(n)$. These eigenvalues define the step sizes used to adapt the individual weights in the LMS algorithm of Eq. (8.48). Assuming that the stochastic process responsible for generating the input vector $\mathbf{u}(n)$ is ergodic, we may define an estimate of its correlation matrix $\mathbf{R}$ (for real-valued data) as

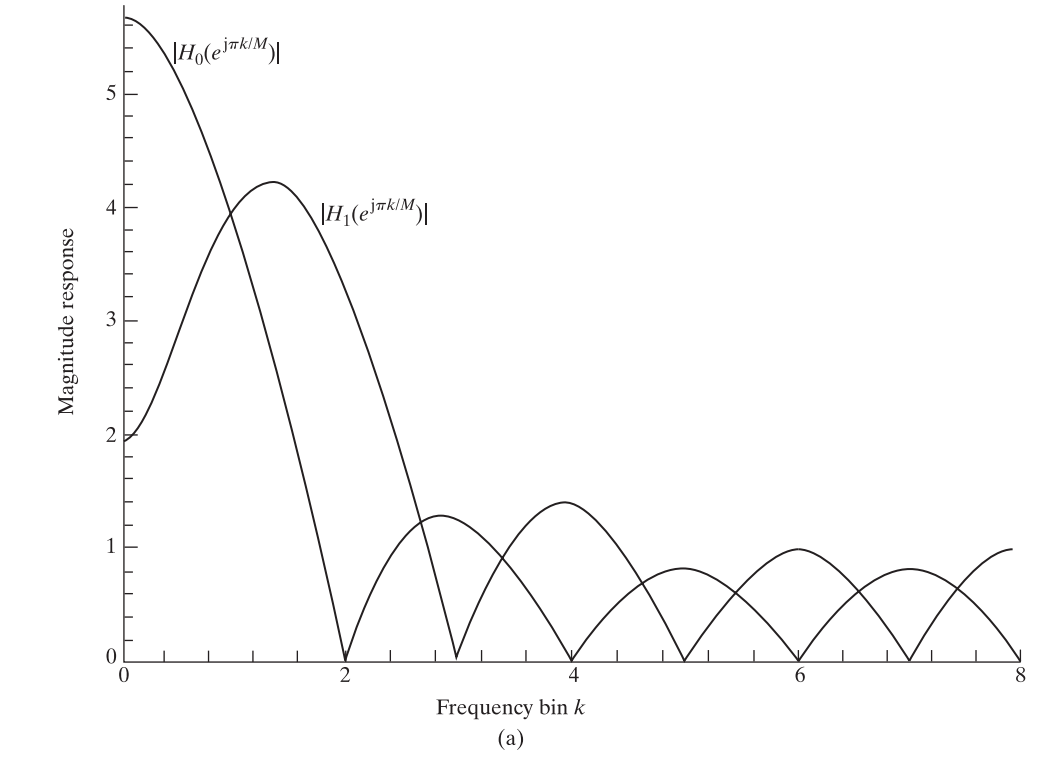
$$\hat{\mathbf{R}}(n) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{u}(i)\mathbf{u}^{\mathrm{T}}(i), \tag{8.68}$$

which is known as the *sample correlation matrix*. The coefficients of the DCT provide an approximation to the M-by-M matrix $\mathbf{Q}$ whose columns represent the eigenvectors associated with the eigenvalues of the correlation matrix $\mathbf{R}$. Let $\hat{\mathbf{Q}}$ denote this approximating matrix. Then the vector of outputs $C_0(n), C_1(n), \ldots, C_{M-1}(n)$ produced by the DCT in response to the input vector $\mathbf{u}(n)$ may be expressed as

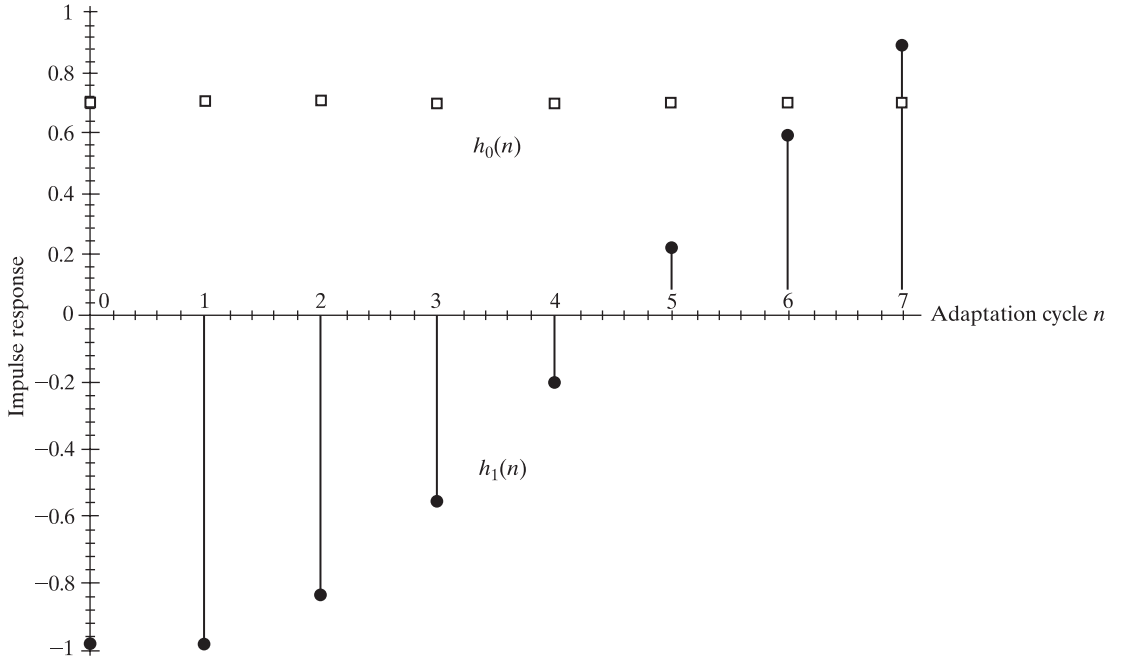
$$\begin{aligned} \hat{\mathbf{v}}(n) &= [C_0(n),\ C_1(n), \ldots, C_{M-1}(n)]^{\mathrm{T}} \\ &= \hat{\mathbf{Q}}\mathbf{u}(n). \end{aligned} \tag{8.69}$$

Furthermore, the approximation to the orthogonal transformation realized by the DCT may be written [in light of Eqs. (8.68) and (8.69)] in the form

$$\begin{aligned} \hat{\mathbf{\Lambda}}(n) &= \hat{\mathbf{Q}}\hat{\mathbf{R}}(n)\hat{\mathbf{Q}}^{\mathrm{T}} \\ &= \frac{1}{n} \sum_{i=1}^{n} \hat{\mathbf{Q}}\mathbf{u}(i)\mathbf{u}^{\mathrm{T}}(i)\mathbf{Q}^{\mathrm{T}} \\ &= \frac{1}{n} \sum_{i=1}^{n} \hat{\mathbf{v}}(i)\hat{\mathbf{v}}^{\mathrm{T}}(i). \end{aligned} \tag{8.70}$$



(a)



(b)

FIGURE 8.7 (a) Magnitude responses of frequency-sampling filters for $m = 0$ and $m = 1$. (b) Corresponding impulse responses $h_0(n)$ and $h_1(n)$ of the two frequency-sampling filters.

Equivalently, we have

$$\hat{\lambda}_m(n) = \frac{1}{n}\sum_{i=1}^{n} C_m^2(i), \qquad m = 0, 1, \ldots, M-1. \tag{8.71}$$

Equation (8.71) may be cast into a recursive form by writing

$$\begin{aligned}\hat{\lambda}_m(n) &= \frac{1}{n} C_m^2(n) + \frac{1}{n}\sum_{i=1}^{n-1} C_m^2(i) \\ &= \frac{1}{n} C_m^2(n) + \frac{n-1}{n} \cdot \frac{1}{n-1}\sum_{i=1}^{n-1} C_m^2(i).\end{aligned} \tag{8.72}$$

From the defining equation (8.71), we note that

$$\hat{\lambda}_m(n-1) = \frac{1}{n-1}\sum_{i=1}^{n-1} C_m^2(i).$$

Accordingly, we may rewrite Eq. (8.72) in the recursive form

$$\hat{\lambda}_m(n) = \hat{\lambda}_m(n-1) + \frac{1}{n}\left(C_m^2(n) - \hat{\lambda}_m(n-1)\right). \tag{8.73}$$

Equation (8.73) applies to a wide-sense stationary environment. To account for adaptive filtering operation in a nonstationary environment, we modify the recursive equation (8.73) to produce (Chao et al., 1990)

$$\hat{\lambda}_m(n) = \gamma\hat{\lambda}_m(n-1) + \frac{1}{n}\left(C_m^2(n) - \gamma\hat{\lambda}_m(n-1)\right), \qquad m = 0, 1, \ldots, M-1, \tag{8.74}$$

where γ is a *forgetting factor* that lies in the range $0 < \gamma < 1$. Equation (8.74) is the desired formula for recursive computation of the eigenvalues of the correlation matrix of the input vector $\mathbf{u}(n)$.

Summary of the DCT-LMS Algorithm

We are now ready to summarize the steps involved in computing the DCT-LMS algorithm. The summary is presented in Table 8.2, which follows from Figure 8.6, Eqs. (8.48) and (8.74), and Eqs. (8.62) and (8.65).

8.5 COMPUTER EXPERIMENT ON ADAPTIVE EQUALIZATION

In this computer experiment, we revisit the adaptive channel equalization example discussed in Section 6.7, where the traditional LMS algorithm was used to perform the adaptation. This time, we use the DCT-LMS algorithm derived in Section 8.4. (For details of the channel impulse response and the random sequence applied to the channel input, see Section 6.7.)

TABLE 8.2 Summary of the DCT-LMS algorithm:

Initialization:

For $m = 0, 1, \ldots, M-1$, set

$$A_m^{(1)}(0) = A_m^{(2)}(0) = 0$$
$$\hat{\lambda}_m(0) = 0$$
$$\hat{w}_m(0) = 0$$
$$k_m = \begin{cases} 1/\sqrt{2}, & m = 0 \\ 1, & \text{otherwise} \end{cases}$$

Selection of parameters:

$$\alpha = \frac{1}{2M}$$
$$\beta = 0.99$$
$$0 < \gamma < 1$$

Sliding DCT:

For $m = 0, 1, \ldots, M-1$, and $n = 1, 2, \ldots$, compute

$$A_m^{(1)}(n) = \beta W_{2M}^m A_m^{(1)}(n-1) + u(n) - \beta^M (-1)^m u(n-M)$$
$$A_m^{(2)}(n) = \beta W_{2M}^{-m} A_m^{(2)}(n-1) + W_{2M}^{-m}(u(n) - \beta^M (-1)^m u(n-M))$$
$$A_m(n) = A_m^{(1)}(n) + A_m^{(2)}(n)$$
$$C_m(n) = \frac{1}{2} k_m (-1)^m W_{2M}^{m/2} A_m(n),$$

where W_{2M} is defined by

$$W_{2M} = \exp\left(\frac{-\mathrm{j}2\pi}{2M}\right).$$

LMS algorithm:

$$y(n) = \sum_{m=0}^{M-1} C_m(n)\hat{w}_m(n)$$
$$e(n) = d(n) - y(n)$$
$$\hat{\lambda}_m(n) = \gamma\hat{\lambda}_m(n-1) + \frac{1}{n}(C_m^2(n) - \gamma\hat{\lambda}_m(n-1))$$
$$\hat{w}_m(n+1) = \hat{w}_m(n) + \frac{\alpha}{\hat{\lambda}_m(n)} C_m(n)e(n).$$

Note. In computing the updated weight $\hat{w}_m(n+1)$, care should be taken to prevent instability of the LMS algorithm, which can arise if some of the eigenvalue estimates are close to zero. Adding a small constant δ to $\hat{\lambda}_m(n)$ could do the trick, but it appears that a better strategy is to condition the correlation matrix of the input signal vector by adding a small amount of white noise (F. Beaufays, private communication, 1995).

The experiment is in two parts:

- In part 1, we study the transient behavior of the DCT-LMS algorithm for different values of the eigenvalue spread of the correlation matrix of the equalizer input.
- In part 2, we compare the transient behavior of the DCT-LMS algorithm to that of the traditional LMS algorithm.

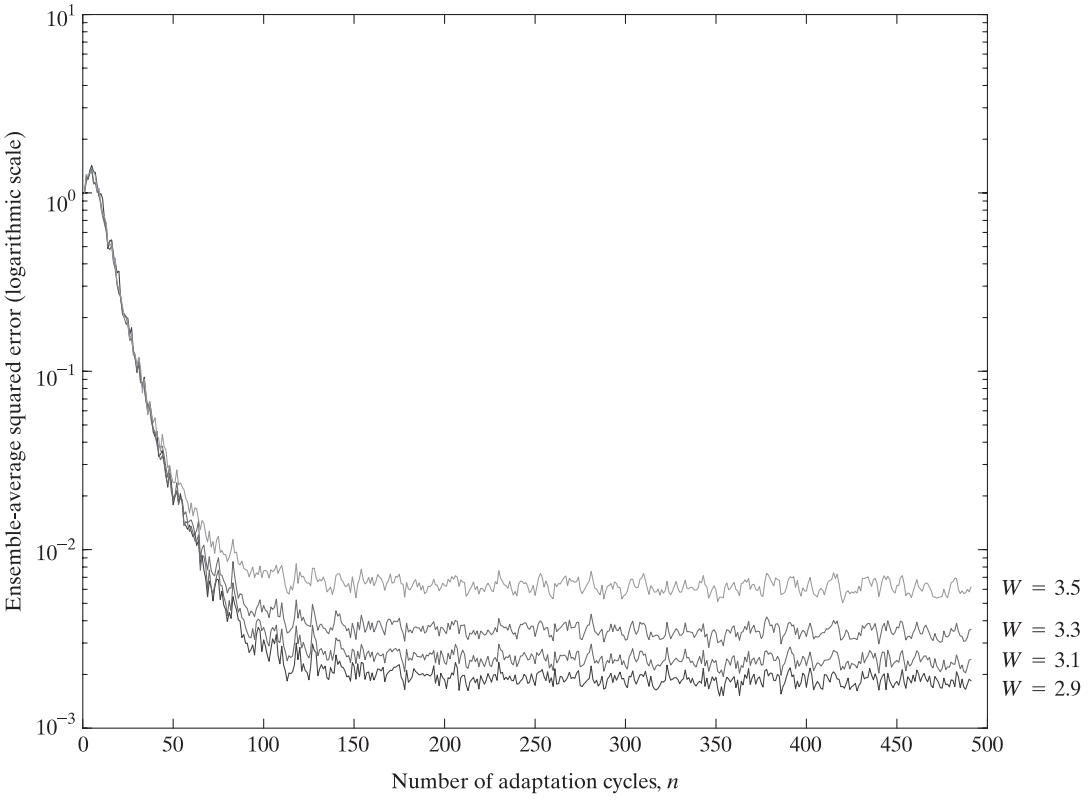


FIGURE 8.8 Learning curves of the DCT-LMS algorithm for varying eigenvalue spread $\chi(\mathbf{R})$.

Throughout these experiments, the signal-to-noise ratio is maintained at the high value of 30 dB, and the parameter β is set equal to 0.99.

Experiment 1: Transient Behavior of the DCT-LMS Algorithm. In Fig. 8.8, the ensemble-average learning curve of the DCT-LMS algorithm is plotted for varying channel parameter W. (The parameter W used in this experiment must not be confused with the parameter W used for the DCT.) Specifically, we have $W = 2.9$, 3.1, 3.3, and 3.5, a sequence that corresponds to the eigenvalue spread $\chi(\mathbf{R}) = 6.078$, 11.124, 21.713, and 46.822. (See Table 6.2.) The results presented in the figure clearly show that, unlike the traditional LMS algorithm, the ensemble-average transient behavior of the DCT-LMS algorithm is less sensitive to variations in the eigenvalue spread of the correlation matrix $\mathbf{R}$ of the input vector $\mathbf{u}(n)$ applied to the channel equalizer. This desirable property is due to the *orthonormalizing* action (i.e., orthogonalization combined with normalization) of the DCT as a preprocessor to the LMS algorithm.

Experiment 2: Comparison of the DCT-LMS Algorithm with Other Adaptive Filtering Algorithms. Figures 8.9(a) through 8.9(d) present a comparison of the ensemble-average error performance of the DCT-LMS algorithm to two other algorithms: the traditional LMS algorithm and the recursive least-squares (RLS) algorithm for four different values of channel parameter W. The operation of the traditional LMS algorithm follows the theory presented in Chapter 6. The theory of the RLS algorithm is presented in Chapter 10; we have included it here as another interesting frame of reference. On the basis of the results presented in Fig. 8.9, we may make the following observations on the transient performance of the three adaptive filtering algorithms considered here:

- The traditional LMS algorithm consistently behaves worst, in that it exhibits the slowest rate of convergence, the greatest sensitivity to variations in the parameter W [and therefore the greatest sensitivity to the eigenvalue spread $\chi(\mathbf{R})$], and the largest excess mean-square error.

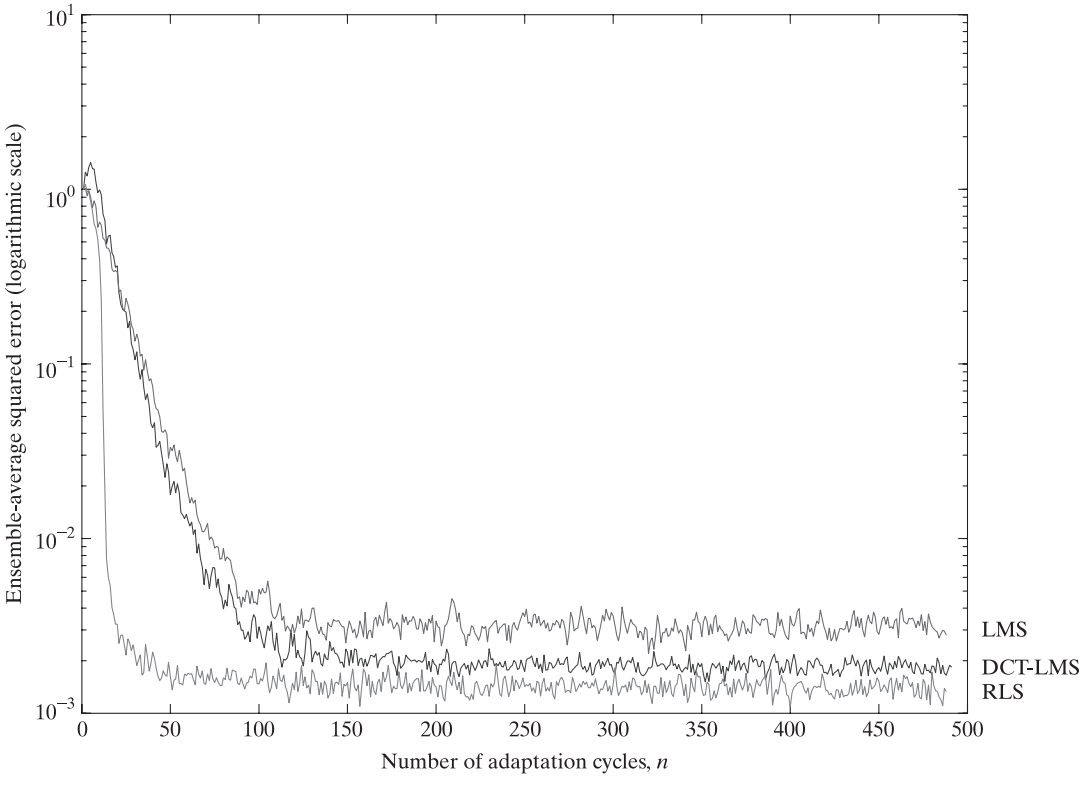


FIGURE 8.9 Comparison of the learning curves of the traditional LMS, DCT-LMS, and RLS algorithms: (a) $\chi(\mathbf{R}) = 6.078$. This figure is continued on the next two pages.

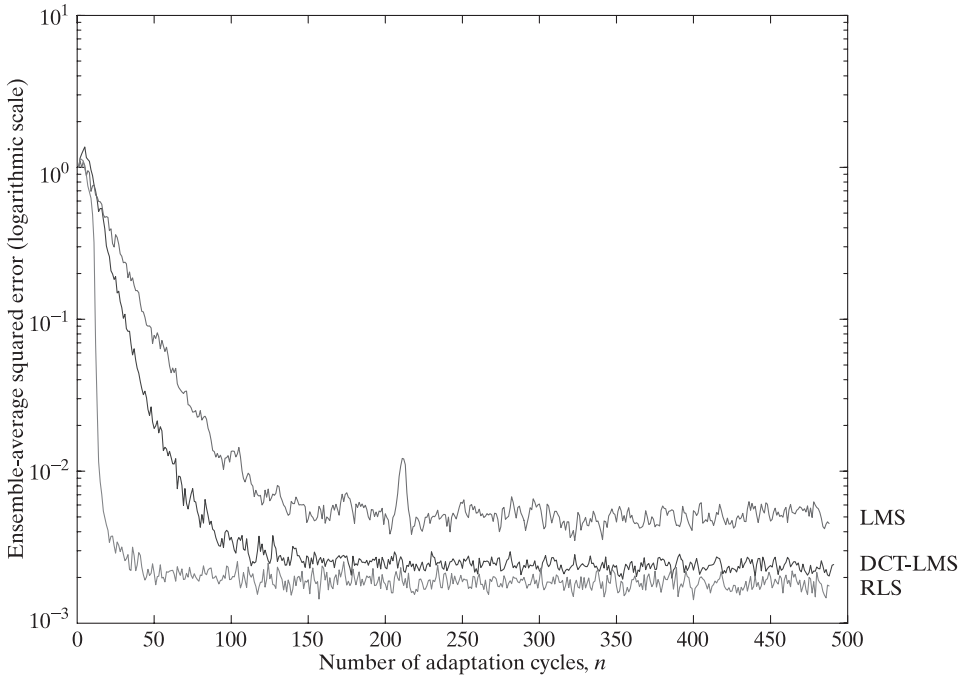


FIGURE 8.9**(b)** $\chi(\mathbf{R}) = 11.124$.

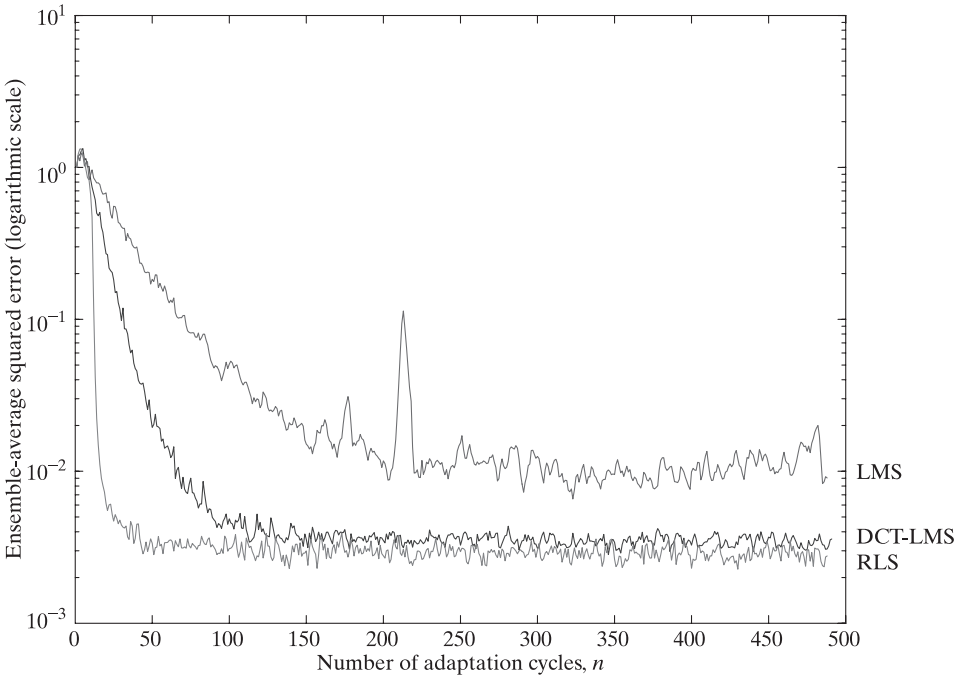


FIGURE 8.9**(c)** $\chi(\mathbf{R}) = 21.713$.

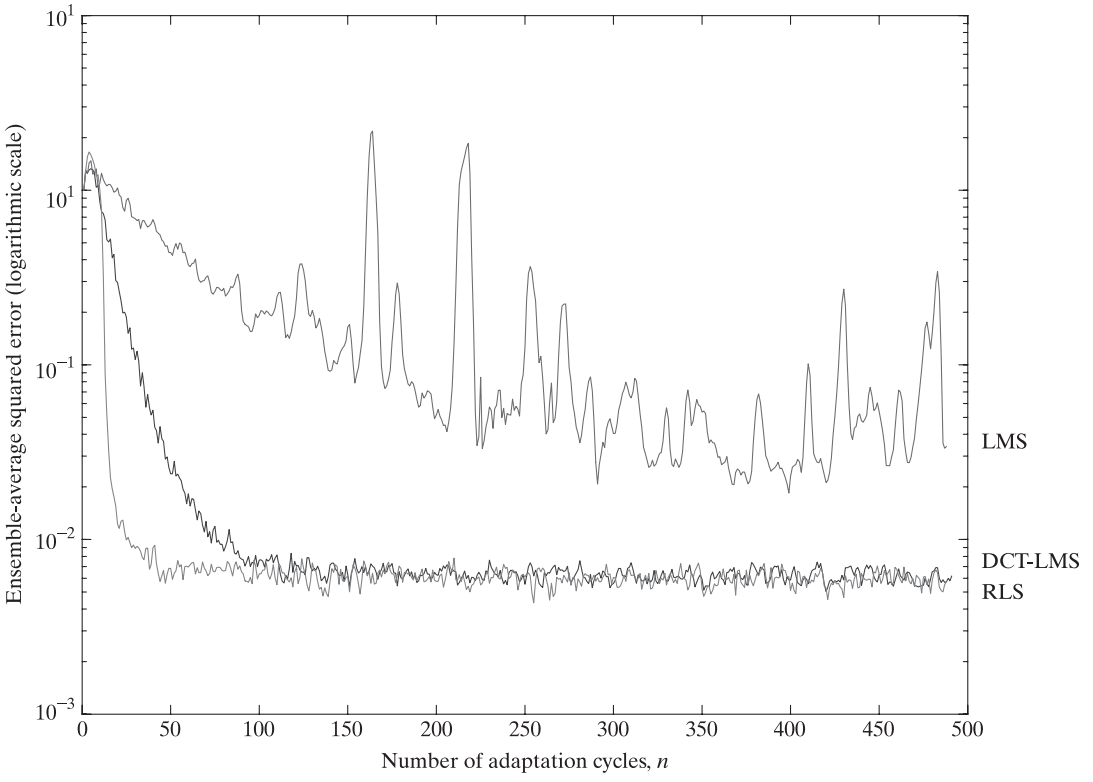


FIGURE 8.9(d) $\chi(\mathbf{R}) = 46.822$.

- The RLS algorithm consistently achieves the fastest rate of convergence and the smallest excess mean-square error, with the least sensitivity to variations in the eigenvalue spread $\chi(\mathbf{R})$.
- For a prescribed eigenvalue spread $\chi(\mathbf{R})$, the transient behavior of the DCT-LMS algorithm lies between those of the traditional LMS and RLS algorithms. Most importantly, however, we note the following:

  (i) The rate of convergence of the DCT-LMS algorithm is relatively insensitive to variations in the eigenvalue spread $\chi(\mathbf{R})$, as already noted under experiment 1.

  (ii) The excess mean-square error produced by the DCT-LMS algorithm is much smaller than that of the traditional LMS algorithm and close to that of the RLS algorithm.

Summarizing: the DCT-LMS algorithm improves statistical efficiency of the traditional LMS algorithm, bringing it closer to that of the RLS algorithm at the expense of increased computational complexity.

8.6 SUBBAND ADAPTIVE FILTERS

As with self-organizing adaptive filters, the motivation for the use of a subband adaptive filter, based on the LMS algorithm because of its simplicity of implementation, is to improve the convergence behavior of the filter.[4] Subband adaptive filters build on *multirate digital filters*, which involve an analysis and a synthesis section. The analysis section, illustrated in Fig. 8.10(a), consists of two functional blocks:

1. The *analysis filter bank*, which consists of a bank of L digital filters with a common input. The transfer functions of the analysis filters are denoted by

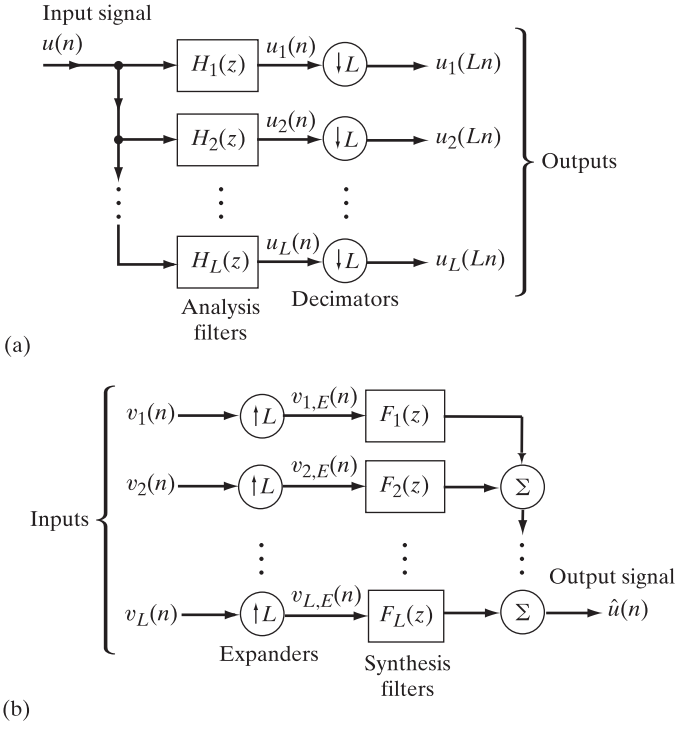


FIGURE 8.10 Multirate digital filter: (a) analysis section; (b) synthesis section.

[4]In a loose sense, if we were to add decimators and memory to the block labeled "discrete Fourier transform (DFT)" in Fig. 8.23 of the 1985 book by Widrow and Stearns, which depicts an adaptive filter with preprocessing to produce orthogonal signals, we get a system that looks somewhat similar to a subband adaptive filter. Clearly, this is not the real thing, but it could be viewed as a connection with historical interest.

In a strict sense, however, work on subband adaptive filtering is traceable to early papers by Kellermann (1985), Yasukawa and Shimada (1987), Chen et al. (1988), Gilloire and Vetterli (1992), and Petraglia and Mittra (1993). The 1992 paper by Gilloire and Vetterli is regarded by some researchers as a pioneering paper in subband adaptive filters; the paper was not perfect, but the ideas were right. Other contributions to the design of subband adaptive filters include the papers by de Courville and Duhamel (1998), and Pradhan and Reddy (1999) as well as Farhang-Boroujeny's (1998) book. The material presented in Section 8.6 is based on the paper by Pradhan and Reddy.

$H_1(z), H_2(z), \ldots, H_L(z)$, and are designed to have slightly overlapping frequency responses. The input signal $u(n)$ is thereby partitioned into a new set of signals denoted by $\{u_k(n)\}_{k=1}^{L}$, which are called *subband signals*.

2. The *bank of decimators*, which *down-sample* the subband signals by virtue of the fact that their bandwidths are all smaller than the bandwidth of the full-band signal $u(n)$. The kth L-fold decimator takes a subband signal $u_k(n)$ to produce an output signal

$$u_{k,\,\mathrm{D}}(n) = u_k(Ln), \qquad k = 1, 2, \ldots, L. \tag{8.75}$$

Only those samples of $u_k(n)$ that occur at instants of time equal to multiples of L are retained by the decimator. Figure 8.11 illustrates the decimation process for $L = 2$. In Fig. 8.11(a), the L-fold decimators are represented by downward arrows, followed by the decimation factor L.

The practical virtue of the analysis section of the multirate digital filter in Fig. 8.10(a) is that it permits the processing of each decimated signal $u_{k,\,D}(n)$ in such a way that the special properties of the kth decimated subband signal (e.g., its energy levels or perceptual significance) are exploited. The signals that result from this processing are then applied to the synthesis section of the multirate digital filter for further processing.

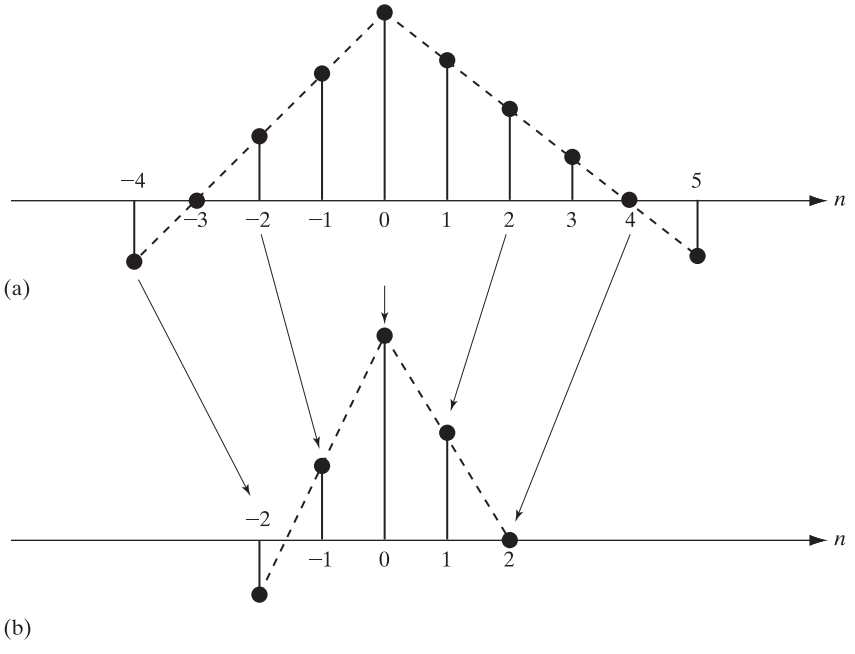


FIGURE 8.11 Decimation process for $L = 2$: (a) original sequence; (b) decimated sequence.

The synthesis section consists of two functional blocks of its own, as illustrated in Fig. 8.10(b):

1. The *bank of expanders*, which *up-sample* their respective inputs. The kth L-fold expander takes the input signal $v_k(n)$ to produce an output signal

$$v_{k,\,\mathrm{E}}(n) = \begin{cases} v_k(n/L) & \text{if } n \text{ is an integer multiple of } L \\ 0 & \text{otherwise} \end{cases}. \tag{8.76}$$

Figure 8.12 illustrates the expansion process for $L = 2$. In Fig. 8.12(b), the L-fold expanders are represented by upward arrows, followed by the expansion factor L. Each expander is essential to performing the process of *interpolation*; however, a filter is needed to convert the zero-valued samples of the expander into interpolated samples and thereby complete the interpolation. To explain this need, we recognize, from the time-frequency duality, which is an inherent property of Fourier transformation, that the spectrum $V_{k,\,\mathrm{E}}(e^{\mathrm{j}\omega})$ of the kth expander output is an L-fold compressed version of the spectrum $V_{k,\,\mathrm{E}}(e^{\mathrm{j}\omega})$ of the expander input.

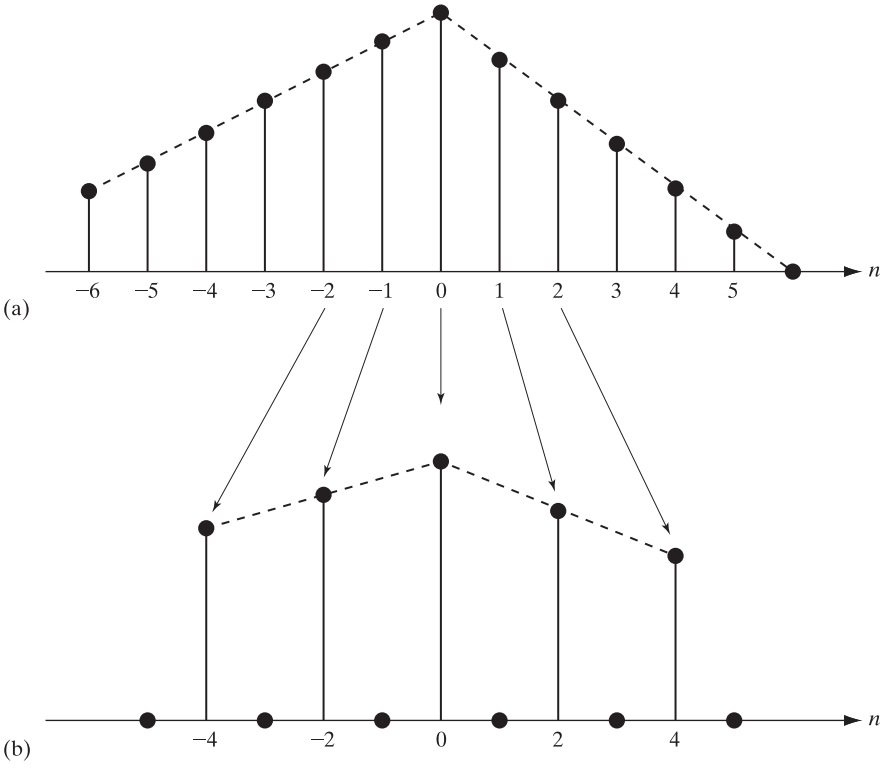


FIGURE 8.12 Expansion process for $L = 2$: (a) original sequence; (b) expanded sequence.

In particular, multiple *images* of the compressed spectrum are created by the expanders—hence the need for filters in the synthesis section to suppress the undesired images.

2. The *synthesis filter bank*, which consists of the parallel connection of a set of L digital filters with a common output. The transfer functions of the synthesis filters are denoted by $F_1(z), F_2(z), \ldots, F_L(z)$, and the resulting output of the synthesis section is denoted by $\hat{u}(n)$.

The output signal $\hat{u}(n)$ differs from the input signal $u(n)$ due to (1) the external processing performed on the decimated signals in the analysis section and (2) aliasing errors. In the context of our present discussion, aliasing refers to the phenomenon of a high-frequency component taking on the identity of a lower-frequency component in the spectrum of its decimated version. This phenomenon, arising because of the nonideal nature of the analysis filters, also includes the aliasing of a low-frequency band (e.g., H_1) into a high-frequency band (e.g., H_2 or H_3) because the decimation creates multiple copies. It is also possible for a low-frequency-band signal to show up somewhere else.

Let $T(z)$ denote the overall transfer function of the multirate digital filter. Suppose that, in the absence of any external processing performed on the output signals of the analysis section, the transfer functions $H_1(z), H_2(z), \ldots, H_L(z)$ of the analysis filters and the corresponding transfer functions $F_1(z), F_2(z), \ldots, F_L(z)$ of the synthesis filters are chosen in such a way that $T(z)$ is forced to be a pure delay, that is,

$$T(z) = cz^{-\Delta}, \tag{8.77}$$

where c is a scaling factor and Δ is the *processing delay* (i.e., *latency*) introduced by the cascaded use of analysis and synthesis filters. When this condition is satisfied, the alias-free multirate digital filter is said to have the *perfect reconstruction property*.

In subband adaptive filters, calculations of the error signals are performed at the decimator outputs of a multirate digital filter, whereby a significant reduction in computational complexity is achieved by virtue of the decimated sampling rate. Moreover, through the use of well-designed analysis and synthesis filters that closely satisfy the perfect reconstruction property of Eq. (8.77), it is possible to attain a significant improvement in convergence behavior of the adaptive filter. However, for certain applications, such as acoustic echo cancellation, upon which interest in subband adaptive filters is focused, it is important that the processing delay Δ be kept under control—a task that may not be easy to accomplish (Pradhan & Reddy, 1999). To describe the composition of a subband adaptive filter, we have chosen the system identification problem depicted in Fig. 8.13(a). We are given an unknown linear dynamic model with a long impulse response, and the problem is to design an LMS subband adaptive filter that provides an approximation to the unknown system such that the variance of the error signal $e(n)$ (i.e., the difference between the response of the unknown system, playing the role of a desired response, and the actual response of the LMS algorithm) is minimized. The subband adaptive filter performs the calculations of error signals in the subbands of a multirate digital filter, as illustrated in Fig. 8.13(b) for the example of two subbands (i.e., $L = 2$). Specifically, the output signals produced by the unknown system $H(z)$ and the LMS algorithm $\hat{W}(z)$ are (1) divided into subbands by the two pairs of analysis filters, $H_1(z)$ and

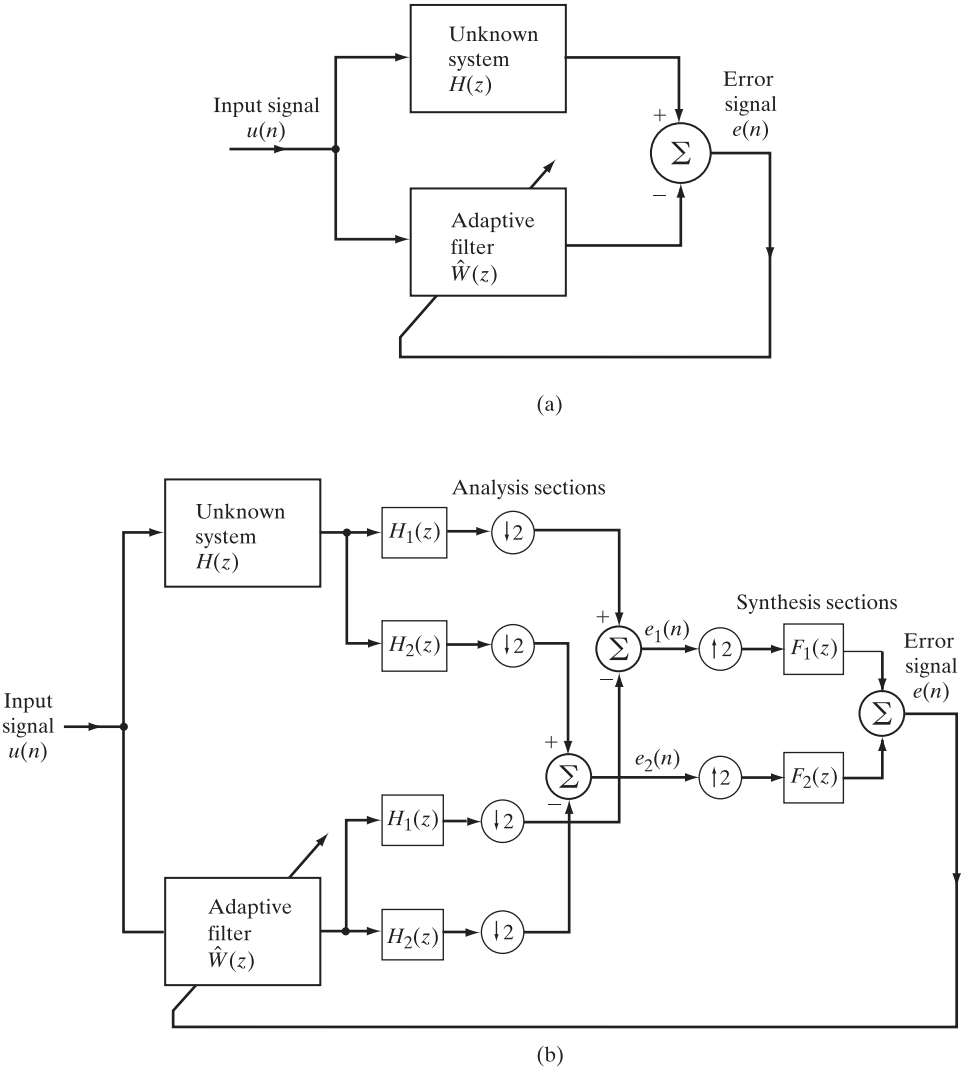


FIGURE 8.13 Subband adaptive filter for two subbands: (a) system identification; (b) subband implementation of the adaptive filter.

$H_2(z)$; (2) decimated by the factor $L = 2$; and then (3) subtracted to produce the pair of error signals, $e_1(n)$ and $e_2(n)$. Next, these two error signals are (1) expanded by the factor $L = 2$; (2) processed by the pair of synthesis filters, $F_1(z)$ and $F_2(z)$; and (3) finally combined to produce the error signal $e(n)$, which, in turn, is used to adjust the free parameters of the adaptive filter in accordance with the LMS algorithm. The analysis filters $H_1(z)$ and $H_2(z)$ and the synthesis filters $F_1(z)$ and $F_2(z)$ are chosen to form a perfect reconstruction pair.

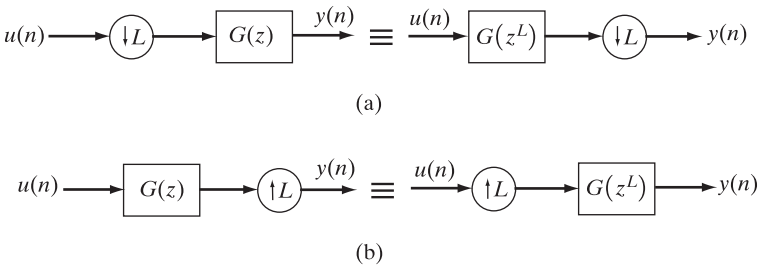


FIGURE 8.14 Noble identities for multirate systems: (a) identity 1; (b) identity 2.

A computationally efficient implementation of the filter banks is made possible by a polyphase decomposition, according to Bellanger et al. (1976). In subband adaptive filtering, the *polyphase decomposition* is applied to the LMS algorithm. Now, by definition, we have

$$\hat{W}(z) = \sum_{k=0}^{M-1} \hat{w}_k z^{-k}, \tag{8.78}$$

where $\{\hat{w}_k\}_{k=0}^{M}$ is the impulse response of the LMS algorithm of length M, assumed to be even. Hence, by separating the even-numbered coefficients of the impulse response $\hat{w}_n$ from the odd-numbered coefficients, we may rewrite Eq. (8.78) in the decomposed form

$$\hat{W}(z) = \hat{W}_1(z^2) + z^{-1}\hat{W}_2(z^2), \tag{8.79}$$

where, with $\hat{w}_{2k} = \hat{w}_{1,\,k}$,

$$\hat{W}_1(z^2) = \sum_{k=0}^{(M-2)/2} \hat{w}_{1,\,k} z^{-2k} \tag{8.80}$$

and, with $\hat{w}_{2k+1} = \hat{w}_{2,\,k}$,

$$\hat{W}_2(z^2) = z^{-1} \sum_{k=0}^{(M-2)/2} \hat{w}_{2,\,k} z^{-2k}. \tag{8.81}$$

The use of polyphase decomposition prepares the way for the application of *noble identities*,[5] which are depicted in Fig. 8.14. Identity 1, shown in Fig. 8.14(a), states that a filter of transfer function $G(z^L)$ followed by an L-fold decimator is equivalent to an L-fold decimator followed by a filter of transfer function $G(z)$. Identity 2, shown in Fig. 8.14(b), states that an L-fold expander followed by a filter of transfer function $G(z^L)$ is equivalent to a filter of transfer function $G(z)$ followed by an L-fold expander.

[5]For a detailed discussion of multirate systems and filter banks, including noble identities and polynomial decomposition, see the book by Vaidynathan (1993).

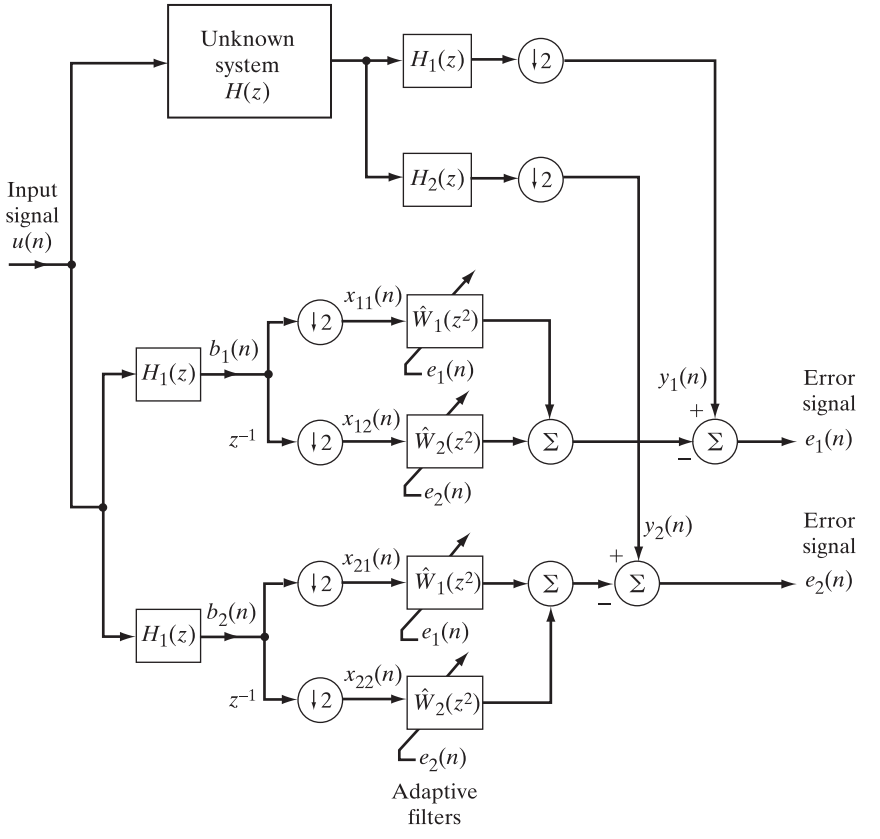


FIGURE 8.15 Modified form of the subband adaptive filter resulting from application of noble identity 1 to the filter of Fig. 8.13(b).

Accordingly, applying the polynomial decomposition and noble identity 1 to the subband adaptive filter of Fig. 8.13(b), we may restructure it into the form shown in Fig. 8.15 (Pradhan & Reddy, 1999). This new adaptive subband filter distinguishes itself in the following ways[6]:

- The components $x_{11}(n), x_{12}(n), x_{21}(n)$, and $x_{22}(n)$ are the subbands of the input signal $u(n)$; together, these components account for the output signals of the analysis filters $H_1(z)$ and $H_2(z)$.
- Two copies of $\hat{W}_1(z^2)$ and $\hat{W}_2(z^2)$, each of length $M/2$, where M is the length of $\hat{W}(z)$, are used in the configuration of Fig. 8.15.

[6]Another novel feature of the subband adaptive filter shown in Fig. 8.15 is that it avoids the use of "cross-filters" between the adjacent subbands. In Gilloire and Vetterli (1992), it is shown that the convergence performance of subband adaptive filters involving the use of cross-filters is worse than that of full-band adaptive filters.

Subband-LMS Adaptive Filtering Algorithm

The stage is now set for describing an algorithm for adapting the weights of the filters represented by $\hat{W}_1(z^2)$ and $\hat{W}_2(z^2)$ in the two-subband system of Fig. 8.15. From the figure, we find that the z-transforms of the error signals $e_1(n)$ and $e_2(n)$ are given, respectively, by

$$E_1(z) = Y_1(z) - X_{11}(z)\hat{W}_1(z^2) - X_{12}(z)\hat{W}_2(z^2) \tag{8.82}$$

and

$$E_2(z) = Y_2(z) - X_{21}(z)\hat{W}_1(z^2) - X_{22}(z)\hat{W}_2(z^2). \tag{8.83}$$

With a form of LMS adaptation in mind, we define the instantaneous cost function to be minimized as

$$J(n) = \frac{1}{2}\,\mathbf{e}^{\mathrm{T}}(n)\mathbf{P}\mathbf{e}(n),$$

where

$$\mathbf{e}(n) = [e_1(n),\, e_2(n)]^{\mathrm{T}}$$

is the error signal vector, and $\mathbf{P}$ is a positive definite matrix. Choosing

$$\mathbf{P} = \begin{bmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{bmatrix},$$

we may write

$$J(n) = \frac{1}{2}\,(\alpha_1 e_1^2(n) + \alpha_2 e_2^2(n)), \tag{8.84}$$

where the weighting coefficients α_1 and α_2 are proportional to the inverse powers of the signals $b_1(n)$ and $b_2(n)$ produced at outputs of the analysis filters $H_1(z)$ and $H_2(z)$ in Fig. 8.15, respectively. The gradient-based algorithm for adjusting the coefficients of the filters $\hat{W}_1(z^2)$ and $\hat{W}_2(z^2)$ is defined by

$$\hat{w}_{1,\,k}(n+1) = \hat{w}_{1,\,k}(n) - \mu\,\frac{\partial J(n)}{\partial \hat{w}_{1,\,k}(n)}, \qquad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{8.85}$$

and

$$\hat{w}_{2,\,k}(n+1) = \hat{w}_{2,\,k}(n) - \mu\,\frac{\partial J(n)}{\partial\, \hat{w}_{2,\,k}(n)}, \qquad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{8.86}$$

where $\hat{w}_{1,\,k}(n)$, and $\hat{w}_{2,\,k}(n)$ are the kth coefficients of the two filters $\hat{W}_1(z^2)$ and $\hat{W}_2(z^2)$, respectively, computed at adaptation cycle n. Using Eq. (8.84), the partial derivatives $\partial J(n)/\partial \hat{w}_{1,\,k}(n)$ and $\partial J(n)/\partial \hat{w}_{2,\,k}(n)$ are given, respectively, by

$$\frac{\partial J(n)}{\partial \hat{w}_{1,\,k}(n)} = \alpha_1 e_1(n)\frac{\partial e_1(n)}{\partial \hat{w}_{1,\,k}(n)} + \alpha_2 e_2(n)\frac{\partial e_2(n)}{\partial \hat{w}_{1,\,k}(n)} \tag{8.87}$$

and

$$\frac{\partial J(n)}{\partial \hat{w}_{2,\,k}(n)} = \alpha_1 e_1(n)\frac{\partial e_1(n)}{\partial \hat{w}_{2,\,k}(n)} + \alpha_2 e_2(n)\frac{\partial e_2(n)}{\partial \hat{w}_{2,\,k}(n)}. \tag{8.88}$$

Moreover, from Eqs. (8.82) and (8.83), we deduce the following set of partial derivatives:

$$\frac{\partial E_i(n)}{\partial \hat{w}_{j,\,k}} = -X_{ij}(z)z^{-k}, \quad \begin{matrix} i = 1, 2 \\ j = 1, 2 \\ k = 0, 1, \ldots, \dfrac{M}{2} - 1 \end{matrix}. \tag{8.89}$$

Taking the inverse z-transform of Eq. (8.89), we get

$$\frac{\partial e_i(n)}{\partial \hat{w}_{j,\,k}} = -x_{ij}(n - k), \quad \begin{matrix} i = 1, 2 \\ j = 1, 2 \\ k = 0, 1, \ldots, \dfrac{M}{2} - 1 \end{matrix}. \tag{8.90}$$

Substituting Eq. (8.90) into Eqs. (8.87) and (8.88), and then applying the LMS rule for adaptation we obtain the following subband LMS algorithm for the system of Fig. 8.15:

$$\hat{w}_{1,\,k}(n + 1) = \hat{w}_{1,\,k}(n) + \mu[\alpha_1 e_1(n)x_{11}(n - k) + \alpha_2 e_2(n)x_{21}(n - k)]; \tag{8.91}$$

$$\hat{w}_{2,\,k}(n + 1) = \hat{w}_{2,\,k}(n) + \mu[\alpha_1 e_1(n)x_{12}(n - k) + \alpha_2 e_2(n)x_{22}(n - k)]. \tag{8.92}$$

In Eqs. (8.91) and (8.92), the LMS algorithm for both subbands is assumed to have the same structure, with $k = 0, 1, \ldots, (M/2) - 1$.

The two-subband adaptive filter of Fig. 8.15 and the accompanying LMS algorithms of Eqs. (8.91) and (8.92) may be readily expanded to the general case of L subbands. The computational complexity of the resulting adaptive filter is nearly the same as that of the full-band LMS algorithm. Pradhan and Reddy (1999) present a convergence analysis of the new algorithm, including computer simulations and demonstrating the improved convergence behavior of the subband adaptive algorithm, compared to the traditional LMS algorithm.

8.7 SUMMARY AND DISCUSSION

Frequency-domain adaptive filtering (FDAF) techniques provide an alternative route to LMS adaptation in the time domain. The fast block LMS algorithm, based on the idea of block-adaptive filtering, provides a computationally efficient algorithm for building an adaptive FIR filter with a long memory. This algorithm exploits the computational advantage offered by a fast convolution technique known as the overlap-save method, which relies on the fast Fourier transform algorithm for its implementation. The fast block LMS algorithm exhibits convergence properties similar to those of the traditional LMS algorithm. In particular, the converged weight vector, misadjustment, and individual time constants of the fast block LMS algorithm are exactly the same as their counterparts in the traditional LMS algorithm. The main differences between the two algorithms are that (1) the fast block LMS algorithm

has a tighter stability bound than the traditional LMS algorithm and (2) the fast block LMS algorithm provides a more accurate estimate of the gradient vector than the traditional LMS algorithm does, with the accuracy of estimation increasing with the block size. Unfortunately, this improvement does not imply a faster convergence behavior, because the eigenvalue spread of the correlation matrix of the input vector, which determines the convergence behavior of the algorithm, is independent of the block size.

The other FDAF technique discussed in the chapter exploits the asymptotic equivalence of the discrete cosine transform to the statistically optimum Karhunen–Loéve transform (KLT). The algorithm, termed the DCT-LMS algorithm, provides a close approximation to the method of self-orthogonalizing adaptive filtering. Unlike the fast block LMS algorithm, the DCT-LMS algorithm is a nonblock algorithm that operates at the incoming data rate; therefore, it is not as computationally efficient as the fast block LMS algorithm.

The fast block LMS algorithm and the DCT-LMS algorithm have a feature in common: They are both convolution-based, FDAF algorithms. As an alternative, we may use *subband adaptive filters*, discussed in Section 8.6. One motivation for such an approach is to achieve computational efficiency by decimating the signals before performing the adaptive process. *Decimation* refers to the process of digitally converting the sampling rate of a signal of interest from a given rate to a lower rate. The use of this approach makes it possible to implement an adaptive FIR filter of long memory that is also computationally efficient. In this endeavor, the task of designing a single long filter is replaced by one of designing a bank of smaller filters that operate in parallel at a lower rate.

PROBLEMS

1. Demonstrate that the fast block LMS algorithm provides a computational cost per adaptation cycle below that of the traditional LMS algorithm.

2. The purpose of this problem is to develop a matrix formulation of the fast block LMS algorithm described by the signal-flow graph of Fig. 8.3.

(a) To define one time-domain constraint built into the operation of the algorithm, let

$$\mathbf{G}_1 = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix},$$

where $\mathbf{I}$ is the M-by-M identity matrix and $\mathbf{O}$ is the M-by-M null matrix. Show that the weight-update equation (8.23) may be rewritten in the compact form

$$\hat{\mathbf{W}}(k+1) = \hat{\mathbf{W}}(k) + \mu \mathbf{G}\mathbf{U}^{\mathrm{H}}(k)\mathbf{W}(k),$$

where the matrix $\mathbf{G}$, which represents a constraint imposed on the computation of the gradient vector, is defined in terms of $\mathbf{G}_1$ by the formula

$$\mathbf{G} = \mathbf{F}\mathbf{G}_1\mathbf{F}^{-1},$$

in which the matrix operator $\mathbf{F}$ signifies discrete Fourier transformation and $\mathbf{F}^{-1}$ signifies inverse discrete Fourier transformation.

(b) To define the other time-domain constraint built into the operation of the fast block LMS algorithm, let

$$\mathbf{G}_2 = [\mathbf{O}, \mathbf{I}],$$

where, as before, $\mathbf{I}$ and $\mathbf{O}$ denote the identity and null matrices, respectively. Show that Eq. (8.21) may be redefined in the compact form

$$\mathbf{E}(k) = \mathbf{F}\mathbf{G}_2^{\mathrm{T}}\,\mathbf{e}(k).$$

(c) Using the time-domain constraints represented by the matrices $\mathbf{G}_1$ and $\mathbf{G}_2$, formulate the corresponding matrix representations of the steps involved in the fast block LMS algorithm.

(d) What is the value of matrix $\mathbf{G}$ for which the fast block LMS algorithm reduces to the unconstrained FDAF algorithm of Section 8.3?

3. Explain the operation of a block-adaptive filter whose filter length M and block size L are 3. Also show that the data matrix defined is a Toeplitz matrix.

4. Figure P8.1 shows the block diagram of a *transform-domain LMS algorithm* (Narayan et al., 1983). The tap-input vector $\mathbf{u}(n)$ is first applied to a bank of bandpass digital filters, implemented

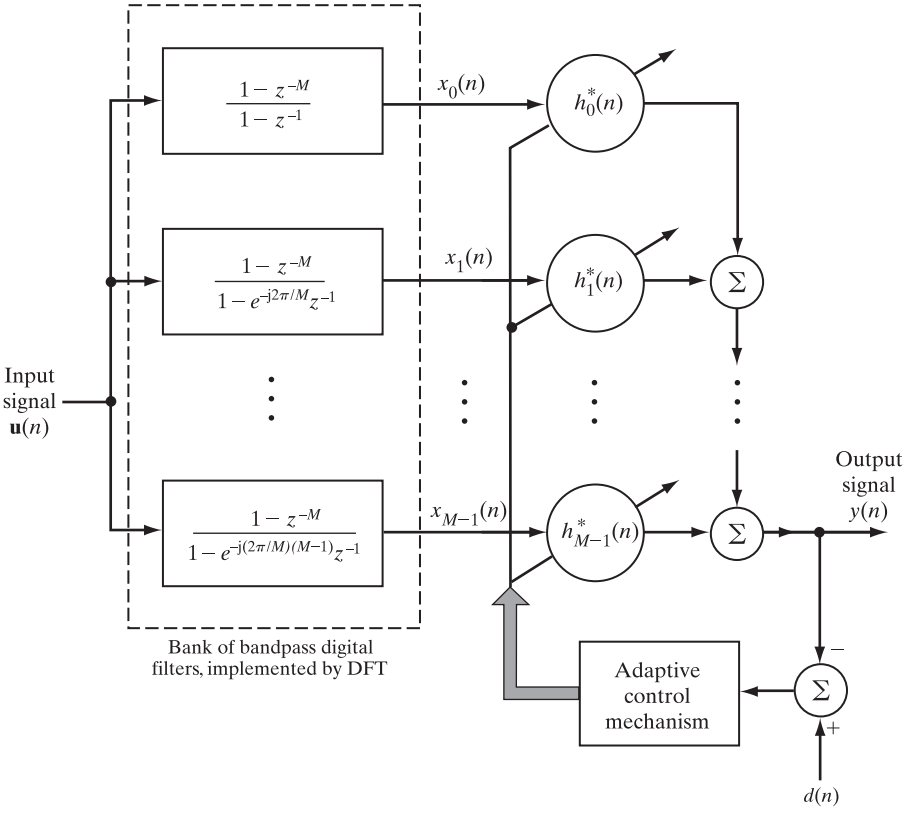


FIGURE P8.1

by means of the *discrete Fourier transform* (DFT). Let $\mathbf{x}(n)$ denote the transformed vector produced at the DFT output. In particular, element k of the vector $\mathbf{x}(n)$ is given by

$$x_k(n) = \sum_{i=0}^{M-1} u(n - i)e^{-j(2\pi/M)ik}, \qquad k = 0, 1, \ldots, M - 1,$$

where $u(n - i)$ is element i of the tap-input vector $\mathbf{u}(n)$. Each $x_k(n)$ is normalized with respect to an estimate of its average power. The inner product of the vector $\mathbf{x}(n)$ and a frequency-domain weight vector $\mathbf{h}(n)$ is formed, obtaining the filter output

$$y(n) = \mathbf{h}^H(n)\mathbf{x}(n).$$

The weight vector update equation is

$$\mathbf{h}(n + 1) = \mathbf{h}(n) + \mu\mathbf{D}^{-1}(n)\mathbf{x}(n)e^*(n),$$

where

$\mathbf{D}(n) = M$-by-M diagonal matrix whose kth element denotes the average power estimate of the DFT output $x_k(n)$ for $k = 0, 1, \ldots, M - 1$

and

μ = adaptation constant.

The asterisk in the update equation denotes complex conjugation, with the estimation error defined by

$$e(n) = d(n) - y(n),$$

where $d(n)$ is the desired response.

(a) Show that the DFT output $x_k(n)$ may be computed recursively, using the relation

$$x_k(n) = e^{j(2\pi/M)k}x_k(n - 1) + u(n) - u(n - M), \qquad k = 0, 1, \ldots, M - 1.$$

(b) Assuming that μ is chosen properly, show that the weight vector $\mathbf{h}(n)$ converges to the frequency-domain optimum solution

$$\mathbf{h}_o = \mathbf{Q}\mathbf{w}_o,$$

where $\mathbf{w}_o$ is the (time-domain) Wiener solution and $\mathbf{Q}$ is a unitary matrix defined by the DFT. Determine the components of $\mathbf{Q}$.

(c) The use of the matrix $\mathbf{D}^{-1}$ in controlling the adjustment applied to the frequency-domain weight vector, in conjunction with the DFT, has the approximate effect of prewhitening the tap-input vector $\mathbf{u}(n)$. Do the following:

(i) Demonstrate the prewhitening effect.

(ii) Discuss how prewhitening compresses the eigenvalue spread of the DFT output vector $\mathbf{x}(n)$.

(iii) The transform-domain LMS algorithm has a faster rate of convergence than the traditional LMS algorithm. Why?

5. The discrete cosine transform $C_m(n)$ of the sequence $u(n)$ may be decomposed as

$$C_m(n) = \frac{1}{2} k_m[C_m^{(1)}(n) + C_m^{(2)}(n)],$$

where k_m is defined by Eq. (8.55).

(a) Show that $C_m^{(1)}(n)$ and $C_m^{(2)}(n)$ may be computed recursively as

$$C_m^{(1)}(n) = W_{2M}^{m/2}[W_{2M}^{m/2}C_m^{(1)}(n-1) + (-1)^m u(n) - u(n-M)]$$

and

$$C_m^{(2)}(n) = W_{2M}^{-m/2}[W_{2M}^{-m/2}C_m^{(2)}(n-1) + (-1)^m u(n) - u(n-M)],$$

respectively, where

$$W_{2M} = \exp\left(-\frac{j2\pi}{2M}\right).$$

(b) How is the computation of $C_m(n)$ modified in light of the multiplying factors associated with the operator z^{-M} in the forward path and the operator z^{-1} in the feedback paths of Fig. 8.6, where $0 < \beta < 1$?

6. Show that the traditional LMS algorithm behaves the same as the self-orthogonalizing adaptive algorithm.

7. The subband LMS algorithm described in Eqs. (8.91) and (8.92) applies to real-valued data. Extend the algorithm to complex-valued data.

8. Illustrate the operation of a subband-LMS adaptive filtering algorithm.

CHAPTER 9

Method of Least Squares

In this chapter, we use a model-dependent procedure known as the *method of least squares* to solve the linear filtering problem, without invoking assumptions on the statistics of the inputs applied to the filter. To illustrate the basic idea of least squares, suppose we have a set of measurements $u(1), u(2), \ldots, u(N)$, made at times $t_1, t_2, \ldots, t_N$, respectively, and the requirement is to construct a curve that is used to *fit* these points in some optimum fashion. Let the time dependence of this curve be denoted by $f(t_i)$. According to the method of least squares, the "best" fit is obtained by *minimizing the sum of the squares of the difference* between $f(t_i)$ and $u(i)$ for $i = 1, 2, \ldots, N$.

The method of least squares may be viewed as an alternative to Wiener filter theory. Basically, Wiener filters are derived from *ensemble averages*, with the result that one filter (optimum in a statistical sense) is obtained for all realizations of the operational environment. The underlying environment is assumed to be wide-sense stationary. On the other hand, the method of least squares is *deterministic* in approach. Specifically, it involves the use of time averages, with the result that the filter depends on the number of samples used in the computation. In computational terms, the method of least squares is a *batch-processing approach*, in that the *least-squares filter* is designed by processing a batch (block) of input data. The filter is adapted to nonstationary data by repeating the computation on a block-by-block basis, which makes it computationally much more demanding than its recursive counterpart. Nevertheless, the batch-processing approach is becoming increasingly more attractive, due to the fact that computing power is no longer the impediment that it used to be in yesteryears.

We begin the discussion by outlining the essence of the least-squares estimation problem.

9.1 STATEMENT OF THE LINEAR LEAST-SQUARES ESTIMATION PROBLEM

Consider a physical phenomenon that is characterized by two sets of variables, $d(i)$ and $u(i)$. The variable $d(i)$ is observed at time i in *response* to the subset of variables $u(i)$, $u(i-1), \ldots, u(i-M+1)$, applied as *inputs*. That is, $d(i)$ is a function of the inputs $u(i), u(i-1), \ldots, u(i-M+1)$. This functional relationship is hypothesized to be *linear*. In particular, the response $d(i)$ is modeled as

$$d(i) = \sum_{k=0}^{M-1} w_{ok}^{*} u(i-k) + e_o(i), \tag{9.1}$$

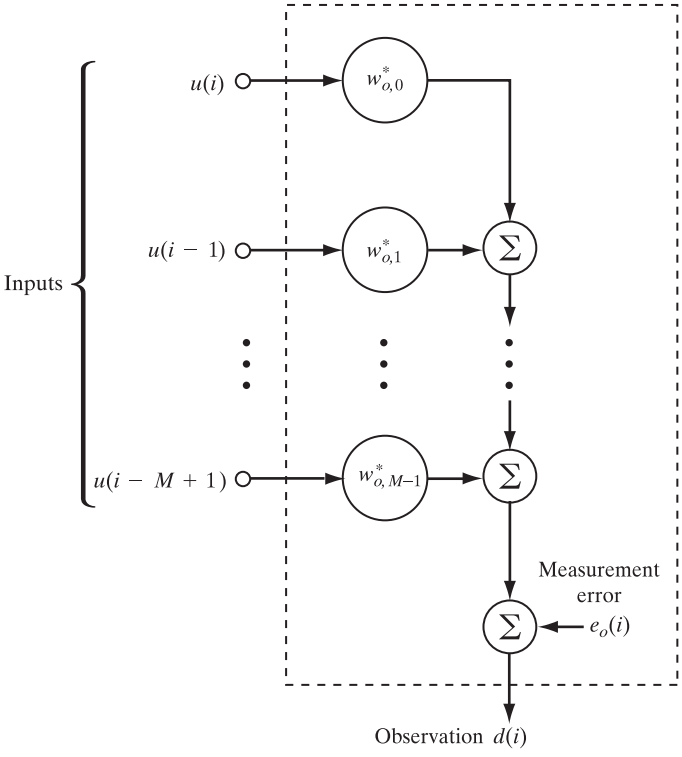


FIGURE 9.1 Multiple linear regression model.

where the w_{ok} are *unknown parameters* of the *model* and $e_o(i)$ represents the *measurement error* to which the statistical nature of the phenomenon is ascribed. The asterisk denotes *complex conjugation*. Each term in the summation in Eq. (9.1) represents a scalar inner product. In effect, the model of Eq. (9.1) says that the variable $d(i)$ may be determined as a linear combination of the input variables $u(i), u(i-1), \ldots, u(i-M+1)$, except for the error $e_o(i)$. This model, represented by the signal-flow graph shown in Fig. 9.1, is called a *multiple linear regression model*. (The multiple linear regression model was used in Chapter 2 and in Chapters 6 and 7 on least-mean-square (LMS) algorithms for the generation of training data.)

The *measurement error* $e_o(i)$ is an *unobservable* random variable that is introduced into the model to account for its inaccuracy. It is customary to assume that the measurement error process $e_o(i)$ is white with zero mean and variance σ^2. That is,

$$\mathbb{E}[e_o(i)] = 0 \qquad \text{for all } i$$

and

$$\mathbb{E}[e_o(i)e_o^*(k)] = \begin{cases} \sigma^2, & i = k \\ 0, & i \neq k \end{cases}.$$

The implication of this assumption is that we may rewrite Eq. (9.1) in the ensemble-average form

$$\mathbb{E}[d(i)] = \sum_{k=0}^{M-1} w_{ok}^* u(i-k),$$

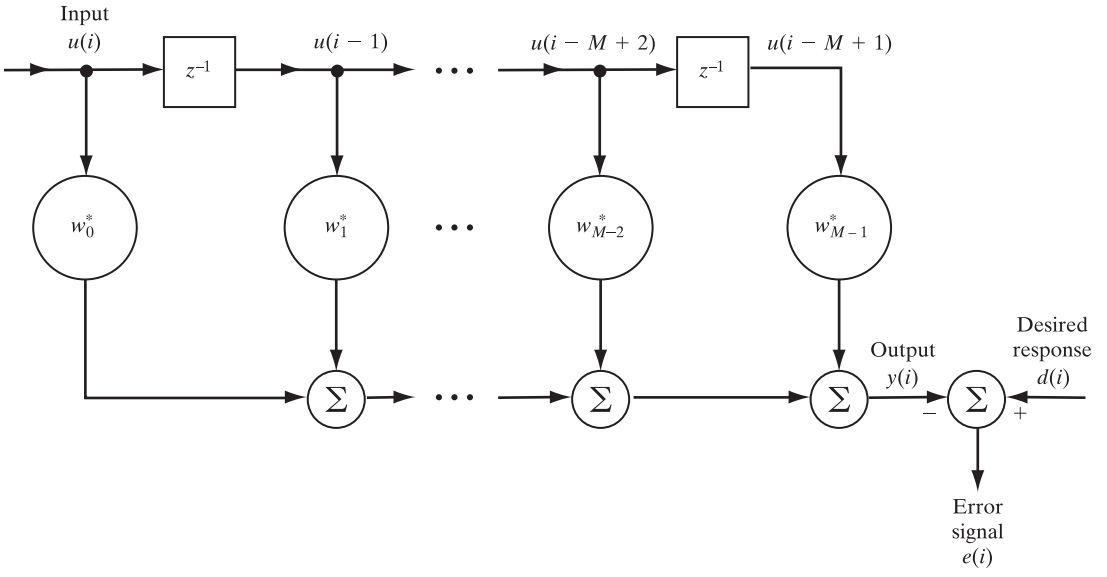


FIGURE 9.2 FIR filter model.

where the values of $u(i), u(i-1), \ldots, u(i-M+1)$ are *known* (i.e., deterministic). Hence, the mean of the response $d(i)$, in theory, is uniquely determined by the model.

The problem we have to solve is to *estimate* the unknown parameters w_{ok} of the multiple linear regression model of Fig. 9.1, given the two *observable* sets of variables: $u(i)$ and $d(i), i = 1, 2, \ldots, N$. To do this, we postulate the finite-duration impulse response (FIR) filter of Fig. 9.2 as the model of interest. By forming inner scalar products of the *tap inputs* $u(i), u(i-1), \ldots, u(i-M+1)$ and the corresponding *tap weights* $w_0, w_1, \ldots, w_{M-1}$, and by utilizing $d(i)$ as the *desired response*, we define the *estimation error* or *residual* $e(i)$ as the difference between the desired response $d(i)$ and the *filter output* $y(i)$; that is,

$$e(i) = d(i) - y(i), \tag{9.2}$$

where

$$y(i) = \sum_{k=0}^{M-1} w_k^* u(i-k). \tag{9.3}$$

Substituting Eq. (9.3) into Eq. (9.2) yields

$$e(i) = d(i) - \sum_{k=0}^{M-1} w_k^* u(i-k). \tag{9.4}$$

In the method of least squares, we choose the tap weights w_k of the FIR filter so as to minimize a cost function that consists of the *sum of error squares*, viz.,

$$\mathscr{E}(w_0, \ldots, w_{M-1}) = \sum_{i=i_1}^{i_2} |e(i)|^2, \tag{9.5}$$

where i_1 and i_2 define the index limits at which the error minimization occurs; this sum may also be viewed as an *error energy*. The values assigned to these limits depend on the

type of *data windowing* employed, to be discussed in Section 9.2. Basically, the problem we have to solve is to substitute Eq. (9.4) into Eq. (9.5) and then minimize the resulting cost function $\mathscr{E}(w_0, \ldots, w_{M-1})$ with respect to the tap weights of the FIR filter in Fig. 9.2. For this minimization, the tap weights of the filter $w_0, w_1, \ldots, w_{M-1}$ are held *constant* during the interval $i_1 \le i \le i_2$. The filter resulting from the minimization is called a *linear least-squares filter*.

9.2 DATA WINDOWING

Given M as the number of tap weights used in the FIR filter model of Fig. 9.2, the rectangular matrix constructed from the input data $u(1), u(2), \ldots, u(N)$ may assume different forms, depending on the values assigned to the limits i_1 and i_2 in Eq. (9.5). In particular, we may distinguish four different methods of *windowing* the input data:

1. The *covariance method*, which makes no assumptions about the data outside the interval $[1, N]$. Thus, by defining the limits of interest as $i_1 = M$ and $i_2 = N$, we may arrange the input data in the matrix form

$$\begin{bmatrix} u(M) & u(M+1) & \cdots & u(N) \\ u(M-1) & u(M) & \cdots & u(N-1) \\ \vdots & \vdots & \ddots & \vdots \\ u(1) & u(2) & \cdots & u(N-M+1) \end{bmatrix}.$$

2. The *autocorrelation method*, which makes the assumption that the data prior to time $i = 1$ and the data after $i = N$ are zero. Thus, with $i_1 = 1$ and $i_2 = N + M - 1$, the matrix of input data takes on the form

$$\begin{bmatrix} u(1) & u(2) & \cdots & u(M) & u(M+1) & \cdots & u(N) & 0 & \cdots & 0 \\ 0 & u(1) & \cdots & u(M-1) & u(M) & \cdots & u(N-1) & u(N) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u(1) & u(2) & \cdots & u(N-M+1) & u(N-M) & \cdots & u(N) \end{bmatrix}.$$

3. The *prewindowing method*, which makes the assumption that the input data prior to $i = 1$ are zero, but makes no assumption about the data after $i = N$. Thus, with $i_1 = 1$ and $i_2 = N$, the matrix of the input data assumes the form

$$\begin{bmatrix} u(1) & u(2) & \cdots & u(M) & u(M+1) & \cdots & u(N) \\ 0 & u(1) & \cdots & u(M-1) & u(M) & \cdots & u(N-1) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u(1) & u(2) & \cdots & u(N-M+1) \end{bmatrix}.$$

4. The *postwindowing method*, which makes no assumption about the data prior to time $i = 1$, but makes the assumption that the data after $i = N$ are zero. Thus, with $i_1 = M$ and $i_2 = N + M - 1$, the matrix of input data takes on the form

$$\begin{bmatrix} u(M) & u(M+1) & \cdots & u(N) & 0 & \cdots & 0 \\ u(M-1) & u(M) & \cdots & u(N-1) & u(N) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u(1) & u(2) & \cdots & u(N-M+1) & u(N-M) & \cdots & u(N) \end{bmatrix}.$$

The terms "covariance method" and "autocorrelation method" are commonly used in the speech-processing literature (Makhoul, 1975; Markel & Gray, 1976). However, the use of these two terms is *not* based on the standard definition of the covariance function as the correlation function with the means removed. Rather, the two terms derive their names from the way we interpret the meaning of the *known parameters* contained in the system of equations that results from minimizing the index of performance of Eq. (9.5). The covariance method derives its name from control theory, in which, with zero-mean tap inputs, these known parameters represent the elements of a *covariance matrix*. The autocorrelation method, on the other hand, derives its name from the fact that, for the conditions stated, the known parameters represent the *short-term autocorrelation function* of the tap inputs. It is of interest to note that, among the four windowing methods just described, the autocorrelation method is the only one that yields a *Toeplitz* correlation matrix for the input data.

In the remainder of this chapter, except for Problem 4, which deals with the autocorrelation method, we will be concerned exclusively with the covariance method. The prewindowing method is considered in subsequent chapters.

9.3 PRINCIPLE OF ORTHOGONALITY REVISITED

When we developed Wiener filter theory in Chapter 2, we proceeded by first deriving the principle of orthogonality (in the ensemble sense) for wide-sense stationary discrete-time stochastic processes, which were then used to derive the Wiener–Hopf equations that provide the mathematical basis of Wiener filters. In this chapter, we proceed in a similar fashion by first deriving the principle of orthogonality on the basis of time averages and then using the principle to derive a system of equations known as the normal equations, which affords the mathematical basis of linear least-squares filters. The development of this theory will be done for the covariance method.

The cost function, or the sum of the error squares, in the covariance method is defined by

$$\mathscr{E}(w_0, \ldots, w_{M-1}) = \sum_{i=M}^{N} |e(i)|^2. \tag{9.6}$$

By choosing the limits on the time index i in this way, we in effect make sure that, for each value of i, all the M tap inputs of the FIR filter in Fig. 9.2 have nonzero values. As mentioned previously, the problem we have to solve is to determine the tap weights of that FIR filter for which the sum of the error squares is minimum. Toward that end, we first rewrite Eq. (9.6) as

$$\mathscr{E}(w_0, \ldots, w_{M-1}) = \sum_{i=M}^{N} e(i)e^*(i), \tag{9.7}$$

where the estimation error $e(i)$ is defined in Eq. (9.4). Let the kth tap weight be expressed in terms of its real and imaginary parts as

$$w_k = a_k + \mathrm{j}b_k, \qquad k = 0, 1, \ldots, M-1. \tag{9.8}$$

Then, substituting Eq. (9.8) into Eq. (9.4), we get

$$e(i) = \mathrm{d}(i) - \sum_{k=0}^{M-1} (a_k - \mathrm{j}b_k)u(i-k). \tag{9.9}$$

We define the kth component of the gradient vector $\nabla\mathscr{E}$ as the derivative of the cost function $\mathscr{E}(w_0, \ldots, w_{M-1})$ with respect to the real and imaginary parts of tap weight w_k:

$$\nabla_k\mathscr{E} = \frac{\partial\mathscr{E}}{\partial a_k} + \mathrm{j}\frac{\partial\mathscr{E}}{\partial b_k}. \tag{9.10}$$

Substituting Eq. (9.7) into Eq. (9.10) and recognizing that the estimation error $e(i)$ is, in general, complex valued, we get

$$\nabla_k\mathscr{E} = -\sum_{i=M}^{n}\left[e(i)\frac{\partial e^*(i)}{\partial a_k} + e^*(i)\frac{\partial e(i)}{\partial a_k} + \mathrm{j}e(i)\frac{\partial e^*(i)}{\partial b_k} + \mathrm{j}e(i)\frac{\partial e(i)}{ab_k}\right]. \tag{9.11}$$

Next, differentiating $e(i)$ in Eq. (9.9) with respect to the real and imaginary parts of w_k, separately, we get the following four partial derivatives:

$$\begin{aligned}
\frac{\partial e(i)}{\partial a_k} &= -u(i-k);\\
\frac{\partial e^*(i)}{\partial a_k} &= -u^*(i-k);\\
\frac{\partial e(i)}{\partial b_k} &= \mathrm{j}u(i-k);\\
\frac{\partial e^*(i)}{\partial b_k} &= -\mathrm{j}u^*(i-k).
\end{aligned} \tag{9.12}$$

The substitution of these four partial derivatives into Eq. (9.11) yields the result

$$\nabla_k\mathscr{E} = -2\sum_{i=M}^{N} u(i-k)e^*(i). \tag{9.13}$$

For the minimization of the cost function $\mathscr{E}(w_0, \ldots, w_{M-1})$ with respect to the tap weights $w_0, \ldots, w_{M-1}$ of the FIR filter in Fig. 9.2, we require that the following conditions be satisfied simultaneously:

$$\nabla_k\mathscr{E} = 0, \qquad k = 0, 1, \ldots, M-1. \tag{9.14}$$

Let $e_{\min}(i)$ denote the special value of the estimation error $e(i)$ that results when the cost function $\mathscr{E}(w_0, \ldots, w_{M-1})$ is minimized (i.e., the FIR filter is optimized) in accordance with Eq. (9.14). Then, from Eq. (9.13), we readily see that the set of conditions in Eq. (9.14) is equivalent to the formula

$$\sum_{i=M}^{N} u(i-k)e_{\min}^*(i) = 0, \qquad k = 0, 1, \ldots, M-1. \tag{9.15}$$

Equation (9.15) is the mathematical description of the temporal version of the *principle of orthogonality*. The *time average*[1] on the left-hand side of Eq. (9.15) represents the cross-correlation between the tap input $u(i-k)$ and the minimum estimation error $e_{\min}(i)$ over the values of time i in the interval $[M, N]$, for a fixed value of k. Accordingly, we may state the *principle of orthogonality* as follows:

> The minimum-error time series $e_{\min}(i)$ is orthogonal to the time series $u(i-k)$ applied to tap k of an FIR filter of length M for $k = 0, 1, \ldots, M-1$ when the filter is operating in its least-squares condition.

This principle provides the basis of a simple *test* that we can carry out in practice to check whether or not the FIR filter is operating in its *least-squares condition*. We merely have to determine the time-average cross-correlation between the estimation error and the time series applied to *each* tap input of the filter. It is *only* when *all* these M cross-correlation functions are identically zero that we find that the cost function $\mathscr{E}(w_0, \ldots, w_{M-1})$ is minimal.

Corollary

Let $\hat{w}_0, \hat{w}_1, \ldots, \hat{w}_{M-1}$ denote the special values of the tap weights $w_0, w_1, \ldots, w_{M-1}$ that result when the FIR filter of Fig. 9.2 is optimized to operate in its least-squares condition. The filter output, obtained from Eq. (9.3), is

$$y_{\min}(i) = \sum_{k=0}^{M-1} \hat{w}_k^* u(i-k). \tag{9.16}$$

This output provides a *least-squares estimate* of the desired response $d(i)$; the estimate is said to be *linear* because it is a linear combination of the tap inputs $u(i), u(i-1), \ldots, u(i-M+1)$. Let $\mathcal{U}_i$ denote the space spanned by $u(i), \ldots, u(i-M+1)$. Let $\hat{d}(i \mid \mathcal{U}_i)$ denote the least-squares estimate of the desired response $d(i)$, given the tap inputs spanned by the space $\mathcal{U}_i$. We may then write

$$\hat{d}(i \mid \mathcal{U}_i) = y_{\min}(i), \tag{9.17}$$

or, equivalently,

$$\hat{d}(i \mid \mathcal{U}_i) = \sum_{k=0}^{M-1} \hat{w}_k^* u(i-k). \tag{9.18}$$

Now suppose we multiply both sides of Eq. (9.15) by $\hat{w}_k^*$ and then sum the result over the values of k in the interval $[0, M-1]$. We then get (after interchanging the order of summation)

$$\sum_{i=M}^{N} \left[\sum_{k=0}^{M-1} \hat{w}_k^* u(i-k) \right] e_{\min}^*(i) = 0. \tag{9.19}$$

[1]To be precise in the use of the term "time average," we should divide the sum on the left-hand side of Eq. (9.15) by the number of terms $N - M + 1$ used in the summation. Clearly, such an operation has no effect on the equation. We have chosen to ignore this scaling factor merely for convenience of presentation.

The summation term inside the square brackets on the left-hand side of Eq. (9.19) is recognized to be the least-squares estimate $\hat{d}(i \mid \mathcal{U}_i)$ of Eq. (9.18). Accordingly, we may simplify Eq. (9.19) to

$$\sum_{i=M}^{N} \hat{d}(i \mid \mathcal{U}_i) e_{\min}^{*}(i) = 0. \tag{9.20}$$

Equation (9.20) is a mathematical description of the *corollary to the principle of orthogonality*. We recognize that the time average on the left-hand side of the equation is the cross-correlation of the two time series: $\hat{d}(i \mid \mathcal{U}_i)$ and $e_{\min}(i)$. Accordingly, we may state the corollary to the principle of orthogonality as follows:

> When an FIR filter operates in its least-squares condition, the least-squares estimate of the desired response, produced at the filter output and represented by the time series $\hat{d}(i \mid \mathcal{U}_i)$, and the minimum estimation error time series $e_{\min}(i)$ are orthogonal to each other over time i.

A geometric illustration of this corollary is deferred to Section 9.6.

9.4 MINIMUM SUM OF ERROR SQUARES

The principle of orthogonality, given in Eq. (9.15), describes the least-squares condition of the FIR filter in Fig. 9.2 when the cost function $\mathscr{E}(w_0, \ldots, w_{M-1})$ is minimized with respect to the tap weights $w_0, \ldots, w_{M-1}$ in the filter. To find the minimum value of this cost function—that is, the *minimum sum of error squares* $\mathscr{E}_{\min}$—it is obvious that we may write

$$\underbrace{d(i)}_{\substack{\text{desired}\\ \text{response}}} = \underbrace{\hat{d}(i \mid \mathcal{U}_i)}_{\substack{\text{estimate of}\\ \text{desired}\\ \text{response}}} + \underbrace{e_{\min}(i)}_{\substack{\text{estimation}\\ \text{error}}}. \tag{9.21}$$

Evaluating the energy of the time series $d(i)$ for values of time i in the interval $[M, N]$ and using the corollary to the principle of orthogonality [i.e., Eq. (9.20)], we get the infinitively satisfying result

$$\mathscr{E}_d = \mathscr{E}_{\text{est}} + \mathscr{E}_{\min}, \tag{9.22}$$

where

$$\mathscr{E}_d = \sum_{i=M}^{N} |d(i)|^2, \tag{9.23}$$

$$\mathscr{E}_{\text{est}} = \sum_{i=M}^{N} |\hat{d}(i \mid \mathcal{U}_i)|^2, \tag{9.24}$$

and

$$\mathscr{E}_{\min} = \sum_{i=M}^{N} |e_{\min}(i)|^2. \tag{9.25}$$

Rearranging Eq. (9.22), we may express the minimum sum of error squares $\mathscr{E}_{\min}$ in terms of the energy $\mathscr{E}_d$ and the energy $\mathscr{E}_{\text{est}}$ contained in the time series $d(i)$ and $\hat{d}(i \mid \mathcal{U}_i)$, respectively:

$$\mathscr{E}_{\min} = \mathscr{E}_d - \mathscr{E}_{\text{est}}. \tag{9.26}$$

Clearly, given the specification of the desired response $d(i)$ for varying i, we may use Eq. (9.23) to evaluate the energy $\mathscr{E}_d$. As with the energy $\mathscr{E}_{\text{est}}$ contained in the time series $\hat{d}(i \mid \mathcal{U}_i)$ representing the estimate of the desired response, we shall defer its evaluation to the next section.

Since $\mathscr{E}_{\min}$ is nonnegative, it follows that the second term on the right-hand side of Eq. (9.26) can never exceed $\mathscr{E}_d$. Indeed, that term reaches the value of $\mathscr{E}_d$ only when the measurement error $e_o(i)$ in the multiple linear regression model of Fig. 9.1 is zero for all i, which is a practical impossibility.

Another case for which $\mathscr{E}_{\min}$ equals $\mathscr{E}_d$ occurs when the least-squares problem is *underdetermined*. Such a situation arises when there are fewer data points than parameters, in which case the estimation error, and therefore $\mathscr{E}_{\text{est}}$, is zero. Note, however, that when the least-squares problem is underdetermined, there is no unique solution to the problem. Discussion of this issue is deferred to the latter part of the chapter.

9.5 NORMAL EQUATIONS AND LINEAR LEAST-SQUARES FILTERS

There are two different, and yet basically equivalent, methods of describing the least-squares condition of the FIR filter in Fig. 9.2. The principle of orthogonality, described in Eq. (9.15), represents one method; the system of *normal equations* represents the other. (Interestingly enough, the system of normal equations derives its name from the corollary to the principle of orthogonality.) Naturally, we may derive this system of equations in its own independent way by formulating the gradient vector $\nabla\mathscr{E}$ in terms of the tap weights of the filter and then solving for the tap-weight vector $\hat{\mathbf{w}}$ for which $\nabla\mathscr{E}$ is zero. Alternatively, we may derive the system of normal equations from the principle of orthogonality. We shall pursue the latter (indirect) approach in this section, leaving the former (direct) approach to the interested reader as Problem 7.

The principle of orthogonality in Eq. (9.15) is formulated in terms of a set of tap inputs and the minimum estimation error $e_{\min}(i)$. Setting the tap weights in Eq. (9.4) to their least-squares values, we get

$$e_{\min}(i) = d(i) - \sum_{t=0}^{M-1} \hat{w}_t^* u(i - t), \tag{9.27}$$

where we have purposely used t as the dummy summation index on the right-hand side. Substituting Eq. (9.27) into Eq. (9.15) and then rearranging terms, we get a system of M simultaneous equations:

$$\sum_{t=0}^{M-1} \hat{w}_t \sum_{i=M}^{N} u(i - k)u^*(i - t) = \sum_{i=M}^{N} u(i - k)d^*(i), \qquad k = 0, \ldots, M - 1. \tag{9.28}$$

The two summations in Eq. (9.28) involving the index i represent time averages, except for a scaling factor. They have the following interpretations:

1. The time average (over i) on the left-hand side of the equation represents the *time-average autocorrelation function* of the tap inputs in the FIR filter of Fig. 9.2. In particular, we may write
$$\phi(t, k) = \sum_{i=M}^{N} u(i - k)u^*(i - t), \qquad 0 \leq (t, k) \leq M - 1. \tag{9.29}$$
2. The time average (also over i) on the right-hand side of Eq. (9.28) represents the *time-average cross-correlation* between the tap inputs and the desired response. In particular, we may write
$$z(-k) = \sum_{i=M}^{N} u(i - k)d^*(i), \qquad 0 \leq k \leq M - 1. \tag{9.30}$$

Accordingly, we may rewrite the system of simultaneous equations (9.28) as follows:

$$\sum_{t=0}^{M-1} \hat{w}_t \phi(t, k) = z(-k), \qquad k = 0, 1, \ldots, M - 1. \tag{9.31}$$

The system of equations (9.31) represents *the expanded system of the normal equations* for linear least-squares filters.

Matrix Formulation of the Normal Equations

We may recast the system of equations (9.31) in matrix form by first introducing the following definitions:

1. The M-by-M *time-average correlation matrix* of the tap inputs $u(i), u(i-1), \ldots, u(i-M+1)$:
$$\mathbf{\Phi} = \begin{bmatrix} \phi(0,0) & \phi(1,0) & \cdots & \phi(M-1,0) \\ \phi(0,1) & \phi(1,1) & \cdots & \phi(M-1,1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(0,M-1) & \phi(1,M-1) & \cdots & \phi(M-1,M-1) \end{bmatrix}. \tag{9.32}$$
2. The M-by-1 *time-average cross-correlation vector* between the tap inputs $u(i)$, $u(i-1), \ldots, u(i-M+1)$ and the desired response $d(i)$:
$$\mathbf{z} = [z(0), z(-1), \ldots, z(-M+1)]^{\mathrm{T}}. \tag{9.33}$$
3. The M-by-1 tap-weight vector of the least-squares filter:
$$\hat{\mathbf{w}} = [\hat{w}_0, \hat{w}_1, \ldots, \hat{w}_{M-1}]^{\mathrm{T}}, \tag{9.34}$$
where the superscript T denotes *transposition*.

We may now rewrite the system of M simultaneous equations (9.31) in terms of these matrix definitions simply as

$$\mathbf{\Phi}\hat{\mathbf{w}} = \mathbf{z}. \tag{9.35}$$

Equation (9.35) is *the matrix form of the normal equations for linear least-squares filters*.

Assuming that $\mathbf{\Phi}$ is nonsingular and therefore that the inverse matrix $\mathbf{\Phi}^{-1}$ exists, we may solve Eq. (9.35) for the tap-weight vector of the linear least-squares filter:

$$\hat{\mathbf{w}} = \mathbf{\Phi}^{-1}\mathbf{z}. \tag{9.36}$$

The condition for the existence of the inverse matrix $\mathbf{\Phi}^{-1}$ is discussed in Section 9.6.

Equation (9.36) is a very important result: It is the linear least-squares counterpart to the solution of the matrix form of the Wiener–Hopf equations (2.36). Basically, Eq. (9.36) states that the tap-weight vector $\hat{\mathbf{w}}$ of a linear least-squares filter is uniquely defined by the product of the inverse of the time-average correlation matrix $\mathbf{\Phi}$ of the tap inputs of the filter and the time-average cross-correlation vector $\mathbf{z}$ between the tap inputs and the desired response. Indeed, this equation is fundamental to the development of all recursive formulations of the linear least-squares filter, as pursued in subsequent chapters of the book.

Minimum Sum of Error Squares

Equation (9.26) defines the minimum sum of error squares $\mathscr{E}_{\min}$. We now complete the evaluation of $\mathscr{E}_{\min}$, expressed as the difference between the energy $\mathscr{E}_d$ of the desired response and the energy $\mathscr{E}_{\text{est}}$ of its estimate. Usually, $\mathscr{E}_d$ is determined from the time series representing the desired response. To evaluate $\mathscr{E}_{\text{est}}$, we write

$$\begin{aligned}
\mathscr{E}_{\text{est}} &= \sum_{i=M}^{N} |\hat{d}(i \mid \mathcal{U}_i)|^2 \\
&= \sum_{i=M}^{N} \sum_{t=0}^{M-1} \sum_{k=0}^{M-1} \hat{w}_t \hat{w}_k^* u(i-k)u^*(i-t) \\
&= \sum_{t=0}^{M-1} \sum_{k=0}^{M-1} \hat{w}_t \hat{w}_k^* \sum_{i=M}^{N} u(i-k)u^*(i-t),
\end{aligned} \tag{9.37}$$

where, in the second line, we have made use of Eq. (9.18). The innermost summation over time i in the final line of Eq. (9.37) represents the time-average autocorrelation function $\phi(t, k)$. [See Eq. (9.29).] Hence, we may rewrite Eq. (9.37) as

$$\begin{aligned}
\mathscr{E}_{\text{est}} &= \sum_{t=0}^{M-1} \sum_{k=0}^{M-1} \hat{w}_k^* \phi(t, k) \hat{w}_t \\
&= \hat{\mathbf{w}}^{\text{H}} \mathbf{\Phi} \hat{\mathbf{w}},
\end{aligned} \tag{9.38}$$

where $\hat{\mathbf{w}}$ is the least-squares tap-weight vector, $\mathbf{\Phi}$ is the time-average correlation matrix of the tap inputs, and the superscript H denotes *Hermitian transposition* (i.e., the operation of transposition combined with complex conjugation). We may further simplify the formula for $\mathscr{E}_{\text{est}}$ by noting that, from the normal equations (9.35), the matrix product $\mathbf{\Phi}\hat{\mathbf{w}}$ equals the cross-correlation vector $\mathbf{z}$. Accordingly, we have

$$\begin{aligned}
\mathscr{E}_{\text{est}} &= \hat{\mathbf{w}}^{\text{H}} \mathbf{z} \\
&= \mathbf{z}^{\text{H}} \hat{\mathbf{w}}.
\end{aligned} \tag{9.39}$$

Finally, substituting Eq. (9.39) into Eq. (9.26) and then using Eq. (9.36) for $\hat{\mathbf{w}}$, we get

$$\begin{aligned}
\mathscr{E}_{\min} &= \mathscr{E}_d - \mathbf{z}^{\text{H}} \hat{\mathbf{w}} \\
&= \mathscr{E}_d - \mathbf{z}^{\text{H}} \mathbf{\Phi}^{-1} \mathbf{z}.
\end{aligned} \tag{9.40}$$

Equation (9.40) is the formula for the minimum sum of error squares, expressed in terms of three known quantities: the energy $\mathscr{E}_d$ of the desired response, the time-average correlation matrix $\mathbf{\Phi}$ of the tap inputs, and the time-average cross-correlation vector $\mathbf{z}$ between the tap inputs and the desired response.

9.6 TIME-AVERAGE CORRELATION MATRIX Φ

The time-average correlation matrix, or simply the correlation matrix $\mathbf{\Phi}$ of the tap inputs, is shown in its expanded form in Eq. (9.32), with the element $\phi(t, k)$ defined in Eq. (9.29). The index k in $\phi(t, k)$ refers to the row number in the matrix $\mathbf{\Phi}$, and t refers to the column number. Let the M-by-1 tap-input vector

$$\mathbf{u}(i) = [u(i), u(i-1), \ldots, u(i-M+1)]^{\mathrm{T}}. \tag{9.41}$$

We may use Eqs. (9.29) and (9.41) to redefine the correlation matrix $\mathbf{\Phi}$ as the time average of the outer product $\mathbf{u}(i)\mathbf{u}^{\mathrm{H}}(i)$ over i as follows:

$$\mathbf{\Phi} = \sum_{i=M}^{N} \mathbf{u}(i)\mathbf{u}^{\mathrm{H}}(i). \tag{9.42}$$

To restate what we said earlier in footnote 1, the summation in Eq. (9.42) should be divided by the scaling factor $(N - M + 1)$ for the correlation matrix $\mathbf{\Phi}$ to be a time average in precise terms. In the statistics literature, the scaled form of $\mathbf{\Phi}$ is referred to as the *sample correlation matrix*. In any event, on the basis of the definition given in Eq. (9.42), we may readily establish the following properties of the correlation matrix:

Property 1. *The correlation matrix* $\mathbf{\Phi}$ *is Hermitian; that is,*

$$\mathbf{\Phi}^{\mathrm{H}} = \mathbf{\Phi}.$$

This property follows directly from Eq. (9.42).

Property 2. *The correlation matrix* $\mathbf{\Phi}$ *is nonnegative definite; that is,*

$$\mathbf{x}^{\mathrm{H}}\mathbf{\Phi}\mathbf{x} \geq 0$$

*for any M-by-*1 *vector* $\mathbf{x}$.

Using the definition of Eq. (9.42), we may derive Property 2 as follows:

$$\begin{aligned}
\mathbf{x}^{\mathrm{H}}\mathbf{\Phi}\mathbf{x} &= \sum_{i=M}^{N} \mathbf{x}^{\mathrm{H}}\mathbf{u}(i)\mathbf{u}^{\mathrm{H}}(i)\mathbf{x} \\
&= \sum_{i=M}^{N} [\mathbf{x}^{\mathrm{H}}\mathbf{u}(i)][\mathbf{x}^{\mathrm{H}}\mathbf{u}(i)]^{*} \\
&= \sum_{i=M}^{N} |\mathbf{x}^{\mathrm{H}}\mathbf{u}(i)|^{2} \geq 0.
\end{aligned}$$

Property 3. *The correlation matrix $\mathbf{\Phi}$ is nonsingular if and only if its determinant is nonzero.*

A square matrix is said to be nonsingular if its inverse exists. The *inverse* of $\mathbf{\Phi}$, denoted by $\mathbf{\Phi}^{-1}$, is defined by

$$\mathbf{\Phi}^{-1} = \frac{\text{adj}(\mathbf{\Phi})}{\det(\mathbf{\Phi})}, \tag{9.43}$$

where adj($\mathbf{\Phi}$) is the *adjoint matrix* of $\mathbf{\Phi}$ whose entries are the respective cofactors of the elements in det($\mathbf{\Phi}$). From Eq. (9.43) we readily see that the inverse matrix $\mathbf{\Phi}^{-1}$ exists if and only if $\det(\mathbf{\Phi}) \neq 0$. It follows therefore that the correlation matrix $\mathbf{\Phi}$ is nonsingular if and only if $\det(\mathbf{\Phi}) \neq 0$.

Property 4. *The eigenvalues of the correlation matrix $\mathbf{\Phi}$ are all real and nonnegative.*

The real requirement on the eigenvalues of $\mathbf{\Phi}$ follows from Property 1. The fact that all these eigenvalues are also nonnegative follows from Property 2.

Property 5. *The correlation matrix is the product of two rectangular Toeplitz matrices that are the Hermitian transpose of each other.*

The correlation matrix $\mathbf{\Phi}$ is, in general, non-Toeplitz, which is clearly seen by examining the expanded form of the matrix given in Eq. (9.32). The elements on the main diagonal, $\phi(0, 0), \phi(1, 1), \ldots, \phi(M-1, M-1)$, have different values; this also applies to secondary diagonals above or below the main diagonal. However, the matrix $\mathbf{\Phi}$ has a special structure in the sense that it is the product of two Toeplitz rectangular matrices. To prove this property, we first use Eq. (9.42) to express the matrix $\mathbf{\Phi}$ in the expanded form:

$$\mathbf{\Phi} = [\mathbf{u}(M), \mathbf{u}(M+1), \ldots, \mathbf{u}(N)] \begin{bmatrix} \mathbf{u}^{\mathrm{H}}(M) \\ \mathbf{u}^{\mathrm{H}}(M-1) \\ \vdots \\ \mathbf{u}^{\mathrm{H}}(N) \end{bmatrix}.$$

Next, for convenience of presentation, we introduce a *data matrix* $\mathbf{A}$, whose Hermitian transpose is defined by

$$\begin{aligned} \mathbf{A}^{\mathrm{H}} &= [\mathbf{u}(M), \quad \mathbf{u}(M+1), \quad \cdots, \quad \mathbf{u}(N)] \\ &= \begin{bmatrix} u(M) & u(M+1) & \cdots & u(N) \\ u(M-1) & u(M) & \cdots & u(N-1) \\ \vdots & \vdots & & \vdots \\ u(1) & u(2) & \cdots & u(N-M+1) \end{bmatrix}. \end{aligned} \tag{9.44}$$

The expanded matrix on the right-hand side of Eq. (9.44) is recognized to be the matrix of input data for the covariance method of data windowing. (See Section 9.2.) Thus, using the definition of Eq. (9.44), we may redefine $\mathbf{\Phi}$ in the compact form

$$\mathbf{\Phi} = \mathbf{A}^{\mathrm{H}}\mathbf{A}. \tag{9.45}$$

From the expanded form of the matrix given in the second line of Eq. (9.44), we see that $\mathbf{A}^H$ consists of an M-by-$(N - M + 1)$ *rectangular Toeplitz matrix*. The data matrix $\mathbf{A}$ itself is likewise an $(N - M + 1)$-by-M rectangular Toeplitz matrix. According to Eq. (9.45), therefore, the correlation matrix $\mathbf{\Phi}$ is the product of two rectangular Toeplitz matrices that are the Hermitian transpose of each other. This completes the derivation of Property 5.

9.7 REFORMULATION OF THE NORMAL EQUATIONS IN TERMS OF DATA MATRICES

The system of normal equations for a least-squares FIR filter is given by Eq. (9.35) in terms of the correlation matrix $\mathbf{\Phi}$ and the cross-correlation vector $\mathbf{z}$. We may reformulate the normal equations in terms of data matrices by using Eq. (9.45) for the correlation matrix $\mathbf{\Phi}$ of the tap inputs and a corresponding relation for the cross-correlation vector $\mathbf{z}$ between the tap inputs and the desired response. To do this, we introduce a *desired data vector* $\mathbf{d}$, consisting of the *desired response* $d(i)$ for values of i in the interval $[M, N]$:

$$\mathbf{d}^H = [d(M), d(M + 1), \ldots, d(N)]. \tag{9.46}$$

Note that we have purposely used Hermitian transposition rather than ordinary transposition in the definition of vector $\mathbf{d}$, to be consistent with the definition of the data matrix $\mathbf{A}$ in Eq. (9.44). With the definitions of Eqs. (9.44) and (9.46) at hand, we may now use Eqs. (9.30) and (9.33) to express the cross-correlation vector

$$\mathbf{z} = \mathbf{A}^H\mathbf{d}. \tag{9.47}$$

Furthermore, we may use Eqs. (9.45) and (9.47) in Eq. (9.35) and so express the system of normal equations in terms of the data matrix $\mathbf{A}$ and the desired data vector $\mathbf{d}$ as

$$\mathbf{A}^H\mathbf{A}\hat{\mathbf{w}} = \mathbf{A}^H\mathbf{d}.$$

From this equation we see that the system of equations used in the minimization of the cost function $\mathscr{E}$ may be represented by $\mathbf{A}\hat{\mathbf{w}} = \mathbf{d}$. Furthermore, assuming that the inverse matrix $(\mathbf{A}^H\,\mathbf{A})^{-1}$ exists, we may solve this system of equations by expressing the tap-weight vector as

$$\hat{\mathbf{w}} = (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H\mathbf{d}. \tag{9.48}$$

We may complete the reformulation of our results for the linear least-squares problem in terms of the data matrices $\mathbf{A}$ and $\mathbf{d}$ by substituting Eqs. (9.45) and (9.47) into Eq. (9.40) and Eq. (9.46) into Eq. (9.23). In so doing, we may rewrite the formula for the minimum sum of error squares as

$$\mathscr{E}_{\min} = \mathbf{d}^H\mathbf{d} - \mathbf{d}^H\mathbf{A}(\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H\mathbf{d}. \tag{9.49}$$

Although this formula looks somewhat cumbersome, its nice feature is that it is expressed explicitly in terms of the data matrix $\mathbf{A}$ and the desired data vector $\mathbf{d}$.

Projection Operator

Equation (9.48) defines the least-squares tap-weight vector $\hat{\mathbf{w}}$ in terms of the data matrix $\mathbf{A}$ and the desired data vector $\mathbf{d}$. The least-squares estimate of $\mathbf{d}$ is therefore given by

$$\begin{aligned}\hat{\mathbf{d}} &= \mathbf{A}\hat{\mathbf{w}} \\ &= \mathbf{A}(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}\mathbf{d}. \end{aligned} \tag{9.50}$$

Accordingly, we may view the multiple matrix product $\mathbf{A}(\mathbf{A}^{\mathrm{H}}\,\mathbf{A})^{-1}\,\mathbf{A}^{\mathrm{H}}$ as a *projection operator* onto the linear space spanned by the columns of the data matrix $\mathbf{A}$, which is the same space $\mathcal{U}_i$ mentioned previously for $i = N$. Denoting this projection operator by $\mathbf{P}$, we may thus write

$$\mathbf{P} = \mathbf{A}(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}. \tag{9.51}$$

The matrix difference

$$\mathbf{I} - \mathbf{A}(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}} = \mathbf{I} - \mathbf{P}$$

is the *orthogonal complement projector*. Note that both the projection operator and its complement are uniquely determined by the data matrix $\mathbf{A}$. The projection operator, $\mathbf{P}$, applied to the desired data vector $\mathbf{d}$, yields the corresponding estimate $\hat{\mathbf{d}}$. On the other hand, the orthogonal complement projector, $\mathbf{I} - \mathbf{P}$, applied to the desired data vector $\mathbf{d}$, yields the estimation error vector $\mathbf{e}_{\min} = \mathbf{d} - \hat{\mathbf{d}}$. Figure 9.3 illustrates the functions of the projection operator and the orthogonal complement projector as described herein.

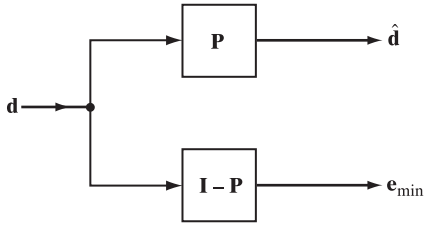


FIGURE 9.3 Projection operator $\mathbf{P}$ and orthogonal complement projector $\mathbf{I} - \mathbf{P}$.

EXAMPLE 1

Consider the example of a linear least-squares filter with two taps (i.e., $M = 2$) and a *real-valued* input time series consisting of four samples (i.e., $N = 4$). Hence, $N - M + 1 = 3$. The input data matrix $\mathbf{A}$ and the desired data vector $\mathbf{d}$ have the following values:

$$\begin{aligned}\mathbf{A} &= \begin{bmatrix} u(2) & u(1) \\ u(3) & u(2) \\ u(4) & u(3) \end{bmatrix} \\ &= \begin{bmatrix} 2 & 3 \\ 1 & 2 \\ -1 & 1 \end{bmatrix};\end{aligned}$$

$$\mathbf{d} = \begin{bmatrix} d(2) \\ d(3) \\ d(4) \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1/34 \end{bmatrix}.$$

The purpose of this example is to evaluate the projection operator and the orthogonal complement projector and use them to illustrate the principle of orthogonality.

The use of Eq. (9.51), reformulated for real data, yields the value of the projection operator as

$$\mathbf{P} = \mathbf{A}(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}} = \frac{1}{35}\begin{bmatrix} 26 & 15 & -2 \\ 15 & 10 & 5 \\ -3 & 5 & 34 \end{bmatrix}.$$

The corresponding value of the orthogonal complement projector is

$$\mathbf{I} - \mathbf{P} = \frac{1}{35}\begin{bmatrix} 9 & -15 & 3 \\ -15 & 25 & -5 \\ -3 & -5 & 1 \end{bmatrix}.$$

Accordingly, the estimate of the desired data vector and the estimation error vector have the following values, respectively:

$$\hat{\mathbf{d}} = \mathbf{P}\mathbf{d} = \begin{bmatrix} 1.91 \\ 1.15 \\ 0 \end{bmatrix};$$

$$\mathbf{e}_{\min} = (\mathbf{I} - \mathbf{P})\mathbf{d} = \begin{bmatrix} 0.09 \\ -0.15 \\ 0.03 \end{bmatrix}.$$

Figure 9.4 depicts three-dimensional geometric representations of the vectors $\hat{\mathbf{d}}$ and $\mathbf{e}_{\min}$ for the example just considered. The figure clearly shows that these two vectors are *normal* (i.e., *perpendicular*) to each other, in accordance with the corollary to the principle of orthogonality (hence the terminology "normal" equations). This condition is the geometric portrayal of the fact that, in a linear least-squares filter, the inner product $\mathbf{e}_{\min}^{\mathrm{H}}\mathbf{d}$ is zero. The figure also depicts the desired data vector $\mathbf{d}$ as the vector sum of the estimate $\hat{\mathbf{d}}$ and the error $\mathbf{e}_{\min}$. Note also that the vector $\mathbf{e}_{\min}$ is orthogonal to span($\mathbf{A}$), defined as the set of all linear combinations of the column vectors of the data matrix $\mathbf{A}$. The estimate $\hat{\mathbf{d}}$ is just one vector in the span($\mathbf{A}$).

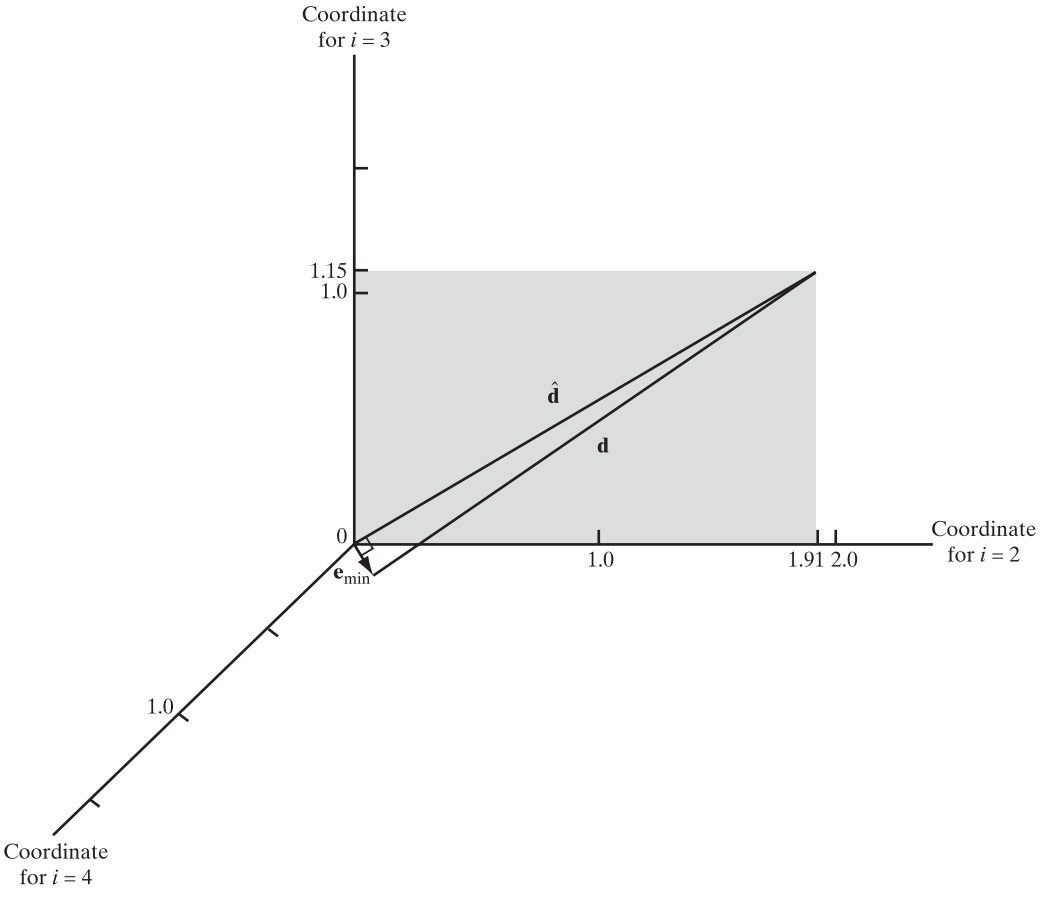


FIGURE 9.4 Three-dimensional geometric interpretations of vectors $\mathbf{d}$, $\hat{\mathbf{d}}$, and $\mathbf{e}_{\min}$ for Example 1.

Uniqueness Theorem

The linear least-squares problem of minimizing the sum of error squares $\mathscr{E}(n)$ always has a solution. That is, for given values of the data matrix $\mathbf{A}$ and the desired data vector $\mathbf{d}$, we can always find a vector $\hat{\mathbf{w}}$ that satisfies the normal equations. It is therefore important that we know if and when the solution is *unique*. This requirement is covered by the following *uniqueness theorem* (Stewart, 1973):

> The least-squares estimate $\hat{\mathbf{w}}$ is unique if and only if the nullity of the data matrix $\mathbf{A}$ equals zero.

Let $\mathbf{A}$ be a K-by-M matrix; in the case of the data matrix $\mathbf{A}$ defined in Eq. (9.44), let $K = N - M + 1$. We define the *null space* of matrix $\mathbf{A}$, written $\mathcal{N}(\mathbf{A})$, as the space of all vectors $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} = \mathbf{0}$. We define the *nullity* of matrix $\mathbf{A}$, written null($\mathbf{A}$), as the dimension of the null space $\mathcal{N}(\mathbf{A})$. In general,

$$\text{null}(\mathbf{A}) \neq \text{null}(\mathbf{A}^{\text{H}}).$$

In light of the uniqueness theorem, which is intuitively satisfying, we may expect a unique solution to the linear least-squares problem *only* when the data matrix $\mathbf{A}$ has *linearly independent columns*—that is, when the data matrix $\mathbf{A}$ is of *full column rank*. This implies that the matrix $\mathbf{A}$ has at least as many rows as columns; that is, $(N - M + 1) \geq M$. The latter condition means that the system of equations represented by $\mathbf{A}\hat{\mathbf{w}} = \mathbf{d}$ and used in the minimization is *overdetermined*, in that it has more equations than unknowns. Thus, provided that the data matrix $\mathbf{A}$ is of full column rank, the M-by-M matrix $\mathbf{A}^{\mathrm{H}}\mathbf{A}$ is *nonsingular*, and the least-squares estimate has the unique value given in Eq. (9.48).

When, however, the matrix $\mathbf{A}$ has *linearly dependent columns* (i.e., the matrix is *rank deficient*), the nullity of the matrix $\mathbf{A}$ is nonzero, and an infinite number of solutions can be found for minimizing the sum of error squares. In such a situation, the linear least-squares problem becomes quite involved, in that we now have the new problem of deciding which particular solution to adopt. We defer discussion of this issue to Section 9.14; in the meantime, we assume that the data matrix $\mathbf{A}$ is of full column rank, so that the least-squares estimate $\hat{\mathbf{w}}$ has the unique value defined by Eq. (9.48).

9.8 PROPERTIES OF LEAST-SQUARES ESTIMATES

The method of least squares has a strong intuitive feel that is reinforced by several outstanding properties, assuming that the data matrix $\mathbf{A}$ is known with *no* uncertainty. These properties, four in number, are described next (Miller, 1974; Goodwin & Payne, 1977).

Property 1. *The least-squares estimate $\hat{\mathbf{w}}$ is unbiased, provided that the measurement error process $e_o(i)$ has zero mean.*

From the multiple linear regression model of Fig. 9.1, we have [using the definitions of Eqs. (9.44) and (9.46)]

$$\mathbf{d} = \mathbf{A}\mathbf{w}_o + \boldsymbol{\varepsilon}_o, \tag{9.52}$$

where

$$\boldsymbol{\varepsilon}_o^{\mathrm{H}} = [e_o(M), e_o(M+1), \ldots, e_o(N)].$$

Hence, substituting Eq. (9.52) into Eq. (9.48), we may express the least-squares estimate as

$$\begin{aligned}\hat{\mathbf{w}} &= (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}\mathbf{A}\mathbf{w}_o + (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}\boldsymbol{\varepsilon}_o \\ &= \mathbf{w}_o + (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}\boldsymbol{\varepsilon}_o. \end{aligned} \tag{9.53}$$

The matrix product $(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}$ is a known quantity, since the data matrix $\mathbf{A}$ is completely defined by the set of given observations $u(1), u(2), \ldots, u(N)$. [See Eq. (9.44).] Hence, if the measurement error process $e_o(i)$ or, equivalently, the measurement error vector $\boldsymbol{\varepsilon}_o$, has zero mean, we find, by taking the expectation of both sides of Eq. (9.53), that the estimate $\hat{\mathbf{w}}$ is *unbiased*; that is,

$$\mathbb{E}[\hat{\mathbf{w}}] = \mathbf{w}_o. \tag{9.54}$$

Property 2. *When the measurement error process $e_o(i)$ is white with zero mean and variance σ^2, the covariance matrix of the least-squares estimate $\hat{\mathbf{w}}$ equals $\sigma^2\boldsymbol{\Phi}^{-1}$.*

Using the relation of Eq. (9.53), we find that the covariance matrix of the least-squares estimate $\hat{\mathbf{w}}$ equals

$$\begin{aligned}\text{cov}[\hat{\mathbf{w}}] &= \mathbb{E}[(\hat{\mathbf{w}} - \mathbf{w}_o)(\hat{\mathbf{w}} - \mathbf{w}_o)^{\mathrm{H}}] \\ &= \mathbb{E}[(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}\boldsymbol{\varepsilon}_o\boldsymbol{\varepsilon}_o^{\mathrm{H}}\mathbf{A}(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}] \\ &= (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}\mathbb{E}[\boldsymbol{\varepsilon}_o\boldsymbol{\varepsilon}_o^{\mathrm{H}}]\mathbf{A}(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}. \qquad (9.55)\end{aligned}$$

With the measurement error process $e_o(i)$ assumed to be white with zero mean and variance σ^2, we have

$$\mathbb{E}[\boldsymbol{\varepsilon}_o\boldsymbol{\varepsilon}_o^{\mathrm{H}}] = \sigma^2\mathbf{I}, \qquad (9.56)$$

where $\mathbf{I}$ is the identity matrix. Hence, Eq. (9.55) reduces to

$$\begin{aligned}\text{cov}[\hat{\mathbf{w}}] &= \sigma^2(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}\mathbf{A}(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1} \\ &= \sigma^2(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1} \\ &= \sigma^2\boldsymbol{\Phi}^{-1}, \qquad (9.57)\end{aligned}$$

which establishes Property 2.

Property 3. *When the measurement error process $e_o(i)$ is white with zero mean, the least-squares estimate $\hat{\mathbf{w}}$ is the best linear unbiased estimate.*

Consider any linear unbiased estimator that is defined by

$$\tilde{\mathbf{w}} = \mathbf{B}\mathbf{d}, \qquad (9.58)$$

where $\mathbf{B}$ is an M-by-$(N - M + 1)$ matrix. Substituting Eq. (9.52) into Eq. (9.58), we get

$$\tilde{\mathbf{w}} = \mathbf{B}\mathbf{A}\mathbf{w}_o + \mathbf{B}\boldsymbol{\varepsilon}_o. \qquad (9.59)$$

With the measurement error vector $\boldsymbol{\varepsilon}_{\mathrm{o}}$ assumed to have zero mean in accordance with Property 1, we find that the expectation of $\tilde{\mathbf{w}}$ is

$$\mathbb{E}[\tilde{\mathbf{w}}] = \mathbf{B}\mathbf{A}\mathbf{w}_o.$$

For the linear estimator $\tilde{\mathbf{w}}$ to be unbiased, we therefore require that the matrix $\mathbf{B}$ satisfy the condition

$$\mathbf{B}\mathbf{A} = \mathbf{I}.$$

Accordingly, we may rewrite Eq. (9.59) as

$$\tilde{\mathbf{w}} = \mathbf{w}_o + \mathbf{B}\boldsymbol{\varepsilon}_o.$$

The covariance matrix of $\tilde{\mathbf{w}}$ is

$$\begin{aligned}\text{cov}[\tilde{\mathbf{w}}] &= \mathbb{E}[(\tilde{\mathbf{w}} - \mathbf{w}_o)(\tilde{\mathbf{w}} - \mathbf{w}_o)^{\mathrm{H}}] \\ &= \mathbb{E}[\mathbf{B}\boldsymbol{\varepsilon}_o\boldsymbol{\varepsilon}_o^{\mathrm{H}}\mathbf{B}^{\mathrm{H}}] \\ &= \sigma^2\mathbf{B}\mathbf{B}^{\mathrm{H}}. \qquad (9.60)\end{aligned}$$

Here, we have made use of Eq. (9.56), which assumes that the elements of the measurement error vector $\boldsymbol{\varepsilon}_{\mathrm{o}}$ are uncorrelated and have a common variance σ^2; that is,

the measurement error process $e_o(i)$ is white. We next define a new matrix $\mathbf{\Psi}$ in terms of $\mathbf{B}$ as

$$\mathbf{\Psi} = \mathbf{B} - (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}. \tag{9.61}$$

Now we form the matrix product $\mathbf{\Psi}\mathbf{\Psi}^{\mathrm{H}}$ and note that $\mathbf{BA} = \mathbf{I}$:

$$\begin{aligned}\mathbf{\Psi}\mathbf{\Psi}^{\mathrm{H}} &= [\mathbf{B} - (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}][\mathbf{B}^{\mathrm{H}} - \mathbf{A}(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}] \\ &= \mathbf{B}\mathbf{B}^{\mathrm{H}} - \mathbf{B}\mathbf{A}(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1} - (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}\mathbf{B}^{\mathrm{H}} + (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1} \\ &= \mathbf{B}\mathbf{B}^{\mathrm{H}} - (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}.\end{aligned}$$

Since the diagonal elements of $\mathbf{\Psi}\mathbf{\Psi}^{\mathrm{H}}$ are always nonnegative, we may use this relation to write

$$\sigma^2 \operatorname{diag}[\mathbf{B}\mathbf{B}^{\mathrm{H}}] \geq \sigma^2 \operatorname{diag}[(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}]. \tag{9.62}$$

The term $\sigma^2\mathbf{B}\mathbf{B}^{\mathrm{H}}$ equals the covariance matrix of the linear unbiased estimate $\tilde{\mathbf{w}}$, as in Eq. (9.60). From Property 2, we also know that the term $\sigma^2(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}$ equals the covariance matrix of the least-squares estimate $\hat{\mathbf{w}}$. Thus, Eq. (9.62) shows that, within the class of linear unbiased estimates, the least-squares estimate $\hat{\mathbf{w}}$ is the "best" estimate of the unknown parameter vector $\mathbf{w}_o$ of the multiple linear regression model, in the sense that each element of $\hat{\mathbf{w}}$ has the smallest possible variance. Accordingly, we may make the following statement:

> When the measurement error process e_o contained in this model is white with zero mean, the least-squares estimate $\hat{\mathbf{w}}$ is the *b*est *l*inear *u*nbiased *e*stimate (BLUE).

Thus far, we have not made any assumption about the statistical distribution of the measurement error $e_o(i)$ other than that it is a zero-mean white-noise process. By making the further assumption that the measurement error $e_o(i)$ is *Gaussian* distributed, we obtain a stronger result on the optimality of the linear least-squares estimate, as discussed next.

Property 4. *When the measurement error process $e_o(i)$ is white and Gaussian with zero mean, the least-squares estimate $\hat{\mathbf{w}}$ achieves the Cramér–Rao lower bound for unbiased estimates.*

Let $f_E(\boldsymbol{\varepsilon}_o)$ denote the joint probability density function of the measurement error vector $\boldsymbol{\varepsilon}_o$. Let $\tilde{\mathbf{w}}$ denote any unbiased estimate of the unknown parameter vector $\mathbf{w}_o$ of the multiple linear regression model. Then the covariance matrix of $\tilde{\mathbf{w}}$ satisfies the inequality

$$\operatorname{cov}[\tilde{\mathbf{w}}] \geq \mathbf{J}^{-1}, \tag{9.63}$$

where

$$\operatorname{cov}[\tilde{\mathbf{w}}] = \mathbb{E}[(\tilde{\mathbf{w}} - \mathbf{w}_o)(\tilde{\mathbf{w}} - \mathbf{w}_o)^{\mathrm{H}}]. \tag{9.64}$$

The matrix $\mathbf{J}$ is called *Fisher's information matrix* and is defined by[2]

$$\mathbf{J} = \mathbb{E}\left[\left(\frac{\partial l}{\partial \mathbf{w}_o^{\mathrm{H}}}\right)\left(\frac{\partial l}{\partial \mathbf{w}_o}\right)\right], \tag{9.65}$$

where

$$l = \ln f_E(\boldsymbol{\varepsilon}_o) \tag{9.66}$$

[2]Fisher's information matrix is discussed in Appendix D for the case of real parameters.

is the *log-likelihood function*—that is, the natural logarithm of the joint probability density function of $\boldsymbol{\varepsilon}_o$.

Since the measurement error $e_o(n)$ is white, the elements of the vector $\boldsymbol{\varepsilon}_o$ are uncorrelated. Furthermore, since $e_o(n)$ is Gaussian, the elements of $\boldsymbol{\varepsilon}_o$ are statistically independent. With $e_o(i)$ assumed to be complex with a mean of zero and variance σ^2, we have (see Section 1.11)

$$f_E(\boldsymbol{\varepsilon}_o) = \frac{1}{(\pi\sigma^2)^{(N-M+1)}} \exp\left[-\frac{1}{\sigma^2}\sum_{i=M}^{N}|e_o(i)|^2\right]. \tag{9.67}$$

The log-likelihood function is therefore

$$\begin{aligned} l &= F - \frac{1}{\sigma^2}\sum_{i=M}^{N}|e_o(i)|^2 \\ &= F - \frac{1}{\sigma^2}\boldsymbol{\varepsilon}_o^{\mathrm{H}}\boldsymbol{\varepsilon}_o, \end{aligned} \tag{9.68}$$

where

$$F = -(N - M + 1)\ln(\pi\sigma^2)$$

is a constant. From Eq. (9.52), we have

$$\boldsymbol{\varepsilon}_o = \mathbf{d} - \mathbf{A}\mathbf{w}_o.$$

Using this relation in Eq. (9.68), we may rewrite l in terms of $\mathbf{w}_o$ as

$$l = F - \frac{1}{\sigma^2}\mathbf{d}^{\mathrm{H}}\mathbf{d} + \frac{1}{\sigma^2}\mathbf{w}_o^{\mathrm{H}}\mathbf{A}^{\mathrm{H}}\mathbf{d} + \frac{1}{\sigma^2}\mathbf{d}^{\mathrm{H}}\mathbf{A}\mathbf{w}_o - \frac{1}{\sigma^2}\mathbf{w}_o^{\mathrm{H}}\mathbf{A}^{\mathrm{H}}\mathbf{A}\mathbf{w}_o. \tag{9.69}$$

Differentiating l, defined in Eq. (9.68),with respect to $\mathbf{w}_o^{\mathrm{H}}$ and formally treating $\mathbf{w}_o$ as a constant in accordance with the Wirtinger calculus in Appendix B, we get

$$\begin{aligned} \frac{\partial l}{\partial \mathbf{w}_o^{\mathrm{H}}} &= \frac{1}{\sigma^2}\mathbf{A}^{\mathrm{H}}(\mathbf{d} - \mathbf{A}\mathbf{w}_o) \\ &= \frac{1}{\sigma^2}\mathbf{A}^{\mathrm{H}}\boldsymbol{\varepsilon}_o. \end{aligned} \tag{9.70}$$

Substituting Eq. (9.70) into Eq. (9.65) yields Fisher's information matrix for the problem at hand:

$$\begin{aligned} \mathbf{J} &= \frac{1}{\sigma^4}\mathbb{E}\left[\left(\frac{1}{\sigma^2}\mathbf{A}^{\mathrm{H}}\boldsymbol{\varepsilon}_o\right)\left(\frac{1}{\sigma^2}\mathbf{A}^{\mathrm{H}}\boldsymbol{\varepsilon}_o\right)^{\mathrm{H}}\right] \\ &= \frac{1}{\sigma^4}\mathbf{A}^{\mathrm{H}}\mathbb{E}[\boldsymbol{\varepsilon}_o\boldsymbol{\varepsilon}_o^{\mathrm{H}}]\mathbf{A} \\ &= \frac{1}{\sigma^2}\mathbf{A}^{\mathrm{H}}\mathbf{A} \\ &= \frac{1}{\sigma^2}\boldsymbol{\Phi}. \end{aligned} \tag{9.71}$$

Note that in the third line of Eq. (9.71) we have made use of Eq. (9.56) describing the assumption that the measurement error process $e_o(i)$ is white with zero mean and variance σ^2. Accordingly, the use of Eq. (9.63) shows that the covariance matrix of the unbiased estimate $\tilde{\mathbf{w}}$ satisfies the inequality

$$\text{cov}[\hat{\mathbf{w}}] \geq \sigma^2 \mathbf{\Phi}^{-1}. \tag{9.72}$$

However, from Property 2, we know that $\sigma^2\mathbf{\Phi}^{-1}$ equals the covariance matrix of the least-squares estimate $\hat{\mathbf{w}}$. Accordingly, $\hat{\mathbf{w}}$ achieves the Cramér–Rao lower bound. Moreover, using Property 1, we conclude by saying:

> When the measurement error process $e_o(i)$ is a zero-mean Gaussian white-noise process, the least-squares estimate $\hat{\mathbf{w}}$ is a *m*inimum-*v*ariance *u*nbiased *e*stimate (MVUE).

9.9 MINIMUM-VARIANCE DISTORTIONLESS RESPONSE (MVDR) SPECTRUM ESTIMATION

In the method of least squares, as described up to this point, no *constraints* are imposed on the solution. In certain applications, however, the use of such an approach may be unsatisfactory, in which case we may resort to a *constrained* version of the method. For example, in *adaptive beamforming* that involves spatial processing, we may wish to *minimize the variance (i.e., the average power) of a beamformer output, while a distortionless response is maintained along the direction of a target signal of interest*. Correspondingly, in the temporal counterpart to this problem, we may be required to *minimize the average power of a spectrum estimator, while a distortionless response is maintained at a particular frequency*. In such applications, the resulting solution is referred to as a *minimum-variance distortionless response (MVDR) estimator*, for obvious reasons. To be consistent with the material presented heretofore, we will formulate the temporal version of the MVDR algorithm.

Consider, then, an FIR filter, as depicted in Fig. 9.5. Let the filter output be denoted by $y(i)$. This output is in response to the tap inputs $u(i), u(i-1), \ldots, u(i-M)$. Specifically, we have

$$y(i) = \sum_{t=0}^{M} a_t^* u(i-t), \tag{9.73}$$

where $a_0, a_1, \ldots, a_m$ are the FIR filter coefficients. Then, assuming the use of the covariance method of data windowing, the requirement is to minimize the *output energy*

$$\mathscr{E}_{\text{out}} = \sum_{i=M+1}^{N} |y(i)|^2,$$

subject to the *constraint*

$$\sum_{k=0}^{M} a_k^* e^{-\mathrm{j}k\omega_0} = 1, \tag{9.74}$$

where ω_0 is an angular frequency of special interest. As in the traditional method of least squares, the filter coefficients $a_0, a_1, \ldots, a_M$ are held constant for the observation interval $1 \leq i \leq N$, where N is the total data length.

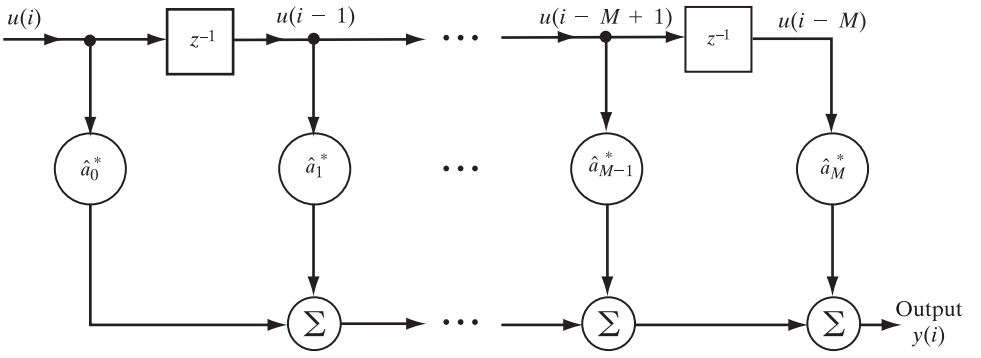


FIGURE 9.5 FIR filter.

To solve this *constrained minimization* problem, we use the *method of Lagrange multipliers*; this optimization procedure is discussed in Appendix C. Specifically, we define the *constrained cost function*

$$\mathscr{E} = \underbrace{\sum_{i=M+1}^{N} |y(i)|^2}_{\text{output energy}} + \underbrace{\lambda\left(\sum_{k=0}^{M} a_k^* e^{-\mathrm{j}k\omega_0} - 1\right)}_{\text{linear constraints}}, \tag{9.75}$$

where λ is a *complex Lagrange multiplier*. Note that in the constrained approach described herein, there is *no* desired response; in place of it, we have a set of linear constraints. Note also that in the absence of a desired response and therefore a frame of reference, the principle of orthogonality loses its meaning.

To solve for the optimum values of the filter coefficients, we first determine the gradient vector $\nabla\mathscr{E}$ with respect to a_k^* and then set it equal to zero. Thus, proceeding in a manner similar to that described in Section 9.3, we find that the kth element of the gradient vector for the constrained cost function of Eq. (9.75) is

$$\nabla_k\mathscr{E} = 2\sum_{i=M+1}^{N} u(i-k)y^*(i) + \lambda^* e^{-\mathrm{j}k\omega_0}. \tag{9.76}$$

Next, substituting Eq. (9.73) into Eq. (9.76) and rearranging terms, we get

$$\begin{aligned}\nabla_k\mathscr{E} &= 2\sum_{t=0}^{M} a_t \sum_{i=M+1}^{N} u(i-k)u^*(i-t) + \lambda^* e^{-\mathrm{j}k\omega_0} \\ &= 2\sum_{t=0}^{M} a_t \phi(t,k) + \lambda^* e^{-\mathrm{j}k\omega_0},\end{aligned} \tag{9.77}$$

where, in the first term of the second line, we have made use of the definition of Eq. (9.29) for the time-average autocorrelation function $\phi(t, k)$ of the tap inputs. To minimize the constrained cost function $\mathscr{E}$, we set

$$\nabla_k\mathscr{E} = 0, \qquad k = 0, 1, \ldots, M. \tag{9.78}$$

Accordingly, we find from Eq. (9.77) that the tap-weights of the *optimized* FIR filter satisfy the following system of $M+1$ simultaneous equations:

$$\sum_{t=0}^{M} \hat{a}_t \phi(t,k) = -\frac{1}{2}\lambda^* e^{-\mathrm{j}k\omega_0}, \qquad k = 0, 1, \ldots, M. \tag{9.79}$$

Using matrix notation, we may rewrite this system of equations in the compact form

$$\mathbf{\Phi}\hat{\mathbf{a}} = -\frac{1}{2}\lambda^* \mathbf{s}(\omega_0), \tag{9.80}$$

where $\mathbf{\Phi}$ is the $(M+1)$-by-$(M+1)$ time-average correlation matrix of the tap inputs, $\hat{\mathbf{a}}$ is the $(M+1)$-by-1 vector of optimum tap weights, and

$$\mathbf{s}(\omega_0) = [1, e^{-\mathrm{j}\omega_0}, \ldots, e^{-\mathrm{j}M\omega_0}]^{\mathrm{T}} \tag{9.81}$$

is the $(M+1)$-by-1 *fixed frequency vector*. Assuming that $\mathbf{\Phi}$ is nonsingular and therefore that its inverse $\mathbf{\Phi}^{-1}$ exists, we may solve Eq. (9.80) for the optimum tap-weight vector:

$$\hat{\mathbf{a}} = -\frac{1}{2}\lambda^* \mathbf{\Phi}^{-1}\mathbf{s}(\omega_0). \tag{9.82}$$

There remains only the problem of evaluating the Lagrange multiplier λ. To solve for λ, we use the linear constraint in Eq. (9.74) for the optimized FIR filter, written in matrix form as

$$\hat{\mathbf{a}}^{\mathrm{H}}\mathbf{s}(\omega_0) = 1. \tag{9.83}$$

Evaluating the *inner product* of the vector $\mathbf{s}(\omega_0)$ and the vector $\hat{\mathbf{a}}$ in Eq. (9.82), setting the result equal to unity and solving for λ, we get

$$\lambda^* = -\frac{2}{\mathbf{s}^{\mathrm{H}}(\omega_0)\mathbf{\Phi}^{-1}\mathbf{s}(\omega_0)}. \tag{9.84}$$

Finally, substituting this value of λ into Eq. (9.82), we obtain the MVDR solution:

$$\hat{\mathbf{a}} = \frac{\mathbf{\Phi}^{-1}\mathbf{s}(\omega_0)}{\mathbf{s}^{\mathrm{H}}(\omega_0)\mathbf{\Phi}^{-1}\mathbf{s}(\omega_0)}. \tag{9.85}$$

Thus, given the time-average correlation matrix $\mathbf{\Phi}$ of the tap inputs and the frequency vector $\mathbf{s}(\omega_0)$, we may use the *MVDR formula* of Eq. (9.85) to compute the optimum tap-weight vector $\hat{\mathbf{a}}$ of the FIR filter in Fig. 9.5.

Let $S_{\mathrm{MVDR}}(\omega_0)$ denote the minimum value of the output energy $\mathscr{E}_{\mathrm{out}}$, which results when the MVDR solution $\hat{\mathbf{a}}$ of Eq. (9.85) is used for the tap-weight vector under the condition that the response is tuned to the angular frequency ω_0. We may then write

$$S_{\mathrm{MVDR}}(\omega_0) = \hat{\mathbf{a}}^{\mathrm{H}}\mathbf{\Phi}\hat{\mathbf{a}}. \tag{9.86}$$

Substituting Eq. (9.85) into Eq. (9.86) and then simplifying the result, we finally get

$$S_{\text{MVDR}}(\omega_0) = \frac{1}{\mathbf{s}^{\text{H}}(\omega_0)\boldsymbol{\Phi}^{-1}\mathbf{s}(\omega_0)}. \tag{9.87}$$

Equation (9.87) may be given a more general interpretation. Suppose that we define a *frequency-scanning vector* as

$$\mathbf{s}(\omega) = [1, e^{-\text{j}\omega}, \ldots, e^{-\text{j}\omega M}]^{\text{T}}, \qquad -\pi < \omega \leq \pi, \tag{9.88}$$

where the angular frequency ω is now variable in the interval $(-\pi, \pi)$. For each ω, let the tap-weight vector of the FIR filter be assigned a corresponding MVDR estimate. The output energy of the optimized filter becomes a function of ω. Let $S_{\text{MVDR}}(\omega)$ describe this functional dependence; accordingly, we may write[3]

$$S_{\text{MVDR}}(\omega) = \frac{1}{\mathbf{s}^{\text{H}}(\omega)\boldsymbol{\Phi}^{-1}\mathbf{s}(\omega)}. \tag{9.89}$$

We refer to Eq. (9.89) as the *MVDR spectrum estimate* and to the solution given in Eq. (9.85) as the *MVDR estimate* of the tap-weight vector for $\omega = \omega_0$. Note that at any ω, the power due to other frequencies is minimized; hence, the MVDR spectrum computed in accordance with Eq. (9.89) exhibits relatively sharp peaks.

9.10 REGULARIZED MVDR BEAMFORMING

The MVDR formula of Eq. (9.85) also provides the basis for adaptive beamforming in radar, sonar, and wireless communications. The motivation for these applications is twofold:

- Fast speed of convergence.
- Ability to cope with a complicated interference environment in which the number of interfering sources is large.

Consider an adaptive beamformer built around a linear array of M antenna elements, as depicted in Fig. 9.6. Let $\mathbf{s}(\theta)$ denote the prescribed *beam-steering vector* (i.e., the vector of linear phase shifts applied to elements of the array) for a look direction of interest that is presented by the electrical angle θ lying in the range $-\pi/2 < \theta \leq \pi/2$. The angle θ is related to the actual direction of arrival φ, measured with respect to the normal to the array, by

$$\theta = \frac{2\pi d}{\lambda}\varphi, \tag{9.90}$$

where d is the interelement spacing between adjacent elements of the array and λ is the wavelength of the incident electromagnetic wave. (See "Background and Preview,"

[3]The method for computing the spectrum in Eq. (9.89) is also referred to in the literature as *Capon's method* (Capon, 1969). The term "minimum-variance distortionless response" owes its origin to Owsley (1985).

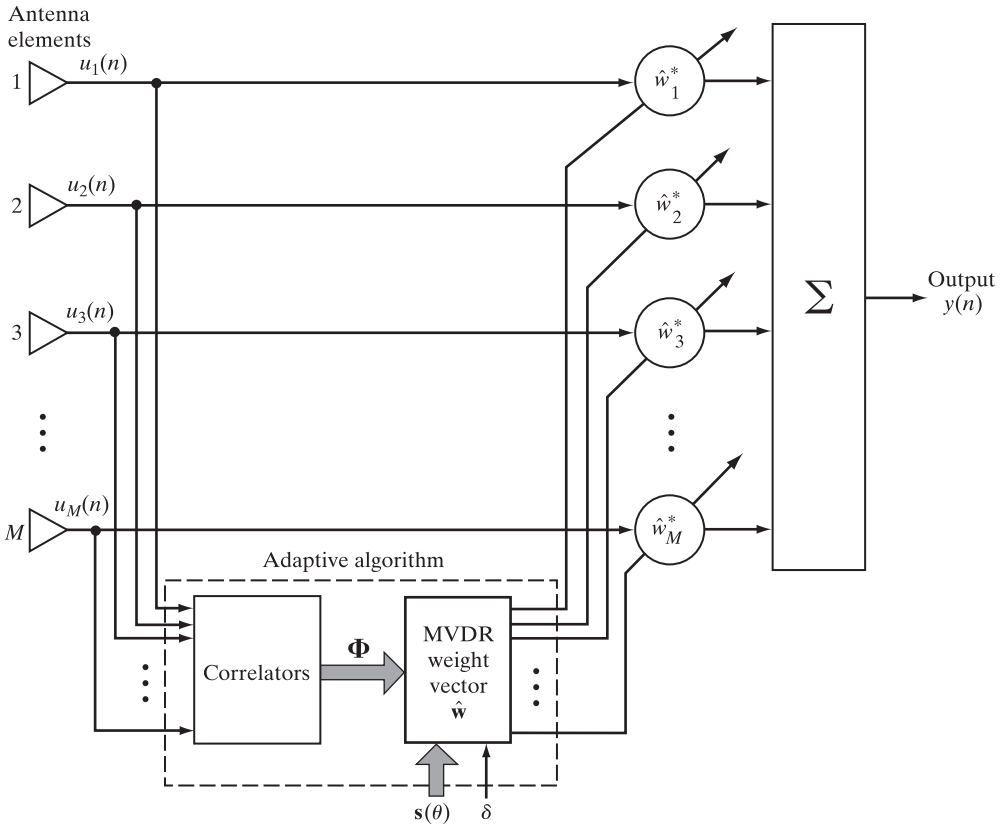


FIGURE 9.6 MVDR beamformer built around a linear array of antenna elements.

Fig. 9). Then, adapting Eq. (9.85) for the problem at hand, we may express the MVDR solution to the weight vector of the beamformer as

$$\hat{\mathbf{w}} = \frac{\mathbf{\Phi}^{-1}\mathbf{s}(\theta)}{\mathbf{s}^{\mathrm{H}}(\theta)\mathbf{\Phi}^{-1}\mathbf{s}(\theta)}, \tag{9.91}$$

which readily lends itself to a batch (block) form of implementation. Suppose now, we are given a block of input data denoted by $\{\mathbf{u}(n)\}_{n=1}^{K}$, where each of the K entries may be viewed as a "snapshot" of the environment. Given such a block of data, we compute the time-average correlation matrix $\mathbf{\Phi}$ by using the formula

$$\mathbf{\Phi} = \sum_{n=1}^{K} \mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n), \tag{9.92}$$

which is plugged into Eq. (9.91) to produce the corresponding value of the weight vector $\hat{\mathbf{w}}$. To cope with statistical variations in the environment, the whole "estimate and plug" procedure is repeated for every block of K snapshots. As mentioned in the introductory remarks to this chapter, although a block method is more computationally intensive than its recursive counterpart, computational complexity is no longer considered to be an issue of practical concern, due to continued advances in computer technology.

Regularization

The MVDR weight vector of Eq. (9.85) is the solution to an *ill-posed inverse estimation problem*; we say so for two reasons:

1. The problem is an inverse problem because, in going from the input data to the weight vector, we are actually proceeding in the direction opposite that of the physical process responsible for generating the data.
2. The problem is ill posed due to inadequacy of the input data and the presence of noise and interfering signals, the combination of which results in a nonunique solution.

Accordingly, using Eq. (9.91) may lead to an antenna pattern (i.e., a plot of the beamformer's power response versus the look direction) with high sidelobes that are unacceptable. Moreover, the antenna pattern may fluctuate randomly from one block of snapshots to the next, with a devastating effect on subsequent signal processing.

To mitigate the ill-posedness of an estimator (e.g., an adaptive beamformer), we need to *stabilize* the solution through the use of *regularization* (Tikhonov, 1963, 1973).[4] The basic principle is to augment the usual data-dependent cost function by a structure-dependent regularization component, the purpose of which is to impose a *smoothing* constraint on the input–output mapping realized by the estimator and thereby stabilize the solution. In words, the cost function for the regularized estimator is written as follows:

$$\begin{pmatrix}\text{regularized}\\ \text{cost}\\ \text{function}\end{pmatrix} = \begin{pmatrix}\text{data-dependent}\\ \text{cost function}\end{pmatrix} + \begin{pmatrix}\text{regularization}\\ \text{parameter}\end{pmatrix} \times \begin{pmatrix}\text{structure-}\\ \text{dependent}\\ \text{regularization}\\ \text{term}\end{pmatrix}. \quad (9.93)$$

In a sense, we may view the *regularization parameter* as an indicator of the sufficiency of the input data in specifying the solution. In particular, as the regularization parameter approaches zero, the solution is completely determined by the input data. On the other hand, as the regularization parameter approaches infinity, the smoothness constraint imposed through regularization is sufficient to specify the solution, which is another way of saying that the input data are unreliable. In practice, the regularization parameter is assigned a value somewhere between these two limiting conditions.

Regularized Adaptive Beamformer with Controlled Sidelobes

With regard to MVDR beamforming, we present a refined form of regularization designed with the following objective in mind (Hughes & McWhirter, 1995; McWhirter et al., 2000):

> **Objective.** Prevent the spatial response corresponding to the MVDR solution deviating unnecessarily from a quiescent response—the one that would have been specified for a nonadaptive beamformer if we had knowledge of the prevalent environment.

[4]For a detailed treatment of regularization theory, the reader is referred to Chapter 7 of the book by Haykin (2009).

Satisfying this objective has the effect of removing unnecessary degrees of freedom from the least-squares solution in a meaningful way. In particular, we may now formulate the problem of designing a regularized MVDR beamformer, configured in the manner shown in Fig. 9.6:

> Determine the optimum weight factor $\hat{\mathbf{w}}$ that minimizes the energy of the beamformer's output:
>
> $$\sum_{n=1}^{K} |\mathbf{w}^{\mathrm{H}}\mathbf{u}(n)|^2 \qquad \text{for a block of } K \text{ snapshots,}$$
>
> subject to the constraint
>
> $$\mathbf{w}^{\mathrm{H}}\mathbf{s}(\theta) = 1, \qquad \mathbf{s}(\theta) = \text{beam-steering vector,}$$
>
> and the requirement that the sidelobes be controlled in accordance with the objective outlined above.

In the case of a linear array, the constraint and the requirement on sidelobes are satisfied by the following choice of regularization (McWhirter et al., 2000):

$$\begin{aligned}\begin{pmatrix}\text{regularization}\\ \text{component}\end{pmatrix} &= \int_{-\pi/2}^{\pi/2} h(\theta)\,|(\mathbf{w}-\mathbf{w}_q)^{\mathrm{H}}\mathbf{s}(\theta)|^2 \mathrm{d}\theta \\ &= (\mathbf{w}-\mathbf{w}_q)^{\mathrm{H}}\mathbf{Z}(\mathbf{w}-\mathbf{w}_q). \end{aligned} \tag{9.94}$$

Here, $\mathbf{w}_q$ is the *quiescent* weight vector and

$$\mathbf{Z} = \int_{-\pi/2}^{\pi/2} h(\theta)\mathbf{s}(\theta)\mathbf{s}^{\mathrm{H}}(\theta)\mathrm{d}\theta \tag{9.95}$$

is an M-by-M matrix. The scalar function $h(\theta)$ is designed to apply a nonnegative weighting to the beamformer's field of view. Thus, incorporating the regularization component of Eq. (9.94) into the constrained cost function for the MVDR problem, in accordance with Eq. (9.93), we write

$$\mathscr{E}_{\mathrm{reg}}(\mathbf{w}) = \sum_{n=1}^{K} |\mathbf{w}^{\mathrm{H}}\mathbf{u}(n)|^2 + \lambda(\mathbf{w}^{\mathrm{H}}\mathbf{s}(\theta) - 1) + \delta(\mathbf{w}-\mathbf{w}_q)^{\mathrm{H}}\mathbf{Z}(\mathbf{w}-\mathbf{w}_q), \tag{9.96}$$

where λ is the Lagrange multiplier and δ is the regularization parameter. Following the generalized Wirtinger calculus in Appendix B, we differentiate $\mathscr{E}_{\mathrm{reg}}(\mathbf{w})$ with respect to $\mathbf{w}^{\mathrm{H}}$ and formally treat $\mathbf{w}$ as a constant. Then, setting the resulting partial derivative to zero and solving for the optimum $\mathbf{w}$, we get

$$\hat{\mathbf{w}} = (\boldsymbol{\Phi} + \delta\mathbf{Z})^{-1}\Big(\delta\mathbf{Z}\mathbf{w}_q - \lambda\mathbf{s}(\theta)\Big), \tag{9.97}$$

where $\boldsymbol{\Phi}$ is the time-average correlation matrix defined in Eq. (9.92). Using Eq. (9.97) to satisfy the constraint

$$\hat{\mathbf{w}}^{\mathrm{H}}\mathbf{s}(\theta) = 1$$

and solving for the Lagrange multiplier λ, we get

$$-\lambda = \frac{1 - \delta \mathbf{w}_q^{\mathrm{H}} \mathbf{Z}(\boldsymbol{\Phi} + \delta\mathbf{Z})^{-1}\mathbf{s}(\theta)}{\mathbf{s}^{\mathrm{H}}(\theta)(\boldsymbol{\Phi} + \delta\mathbf{Z})^{-1}\mathbf{s}(\theta)}, \tag{9.98}$$

where we have used the Hermitian property of the M-by-M augmented correlated matrix $\boldsymbol{\Phi} + \delta\mathbf{Z}$. Finally, substituting Eq. (9.98) into Eq. (9.97), we obtain the regularized MVDR solution:

$$\begin{aligned}\hat{\mathbf{w}} = {} & \frac{(\boldsymbol{\Phi} + \delta\mathbf{Z})^{-1}\mathbf{s}(\theta)}{\mathbf{s}^{\mathrm{H}}(\theta)(\boldsymbol{\Phi} + \delta\mathbf{Z})^{-1}\mathbf{s}(\theta)} + \delta(\boldsymbol{\Phi} + \delta\mathbf{Z})^{-1}\mathbf{Z}\mathbf{w}_q \\ & - \frac{\delta \mathbf{w}_q^{\mathrm{H}} \mathbf{Z}(\boldsymbol{\Phi} + \delta\mathbf{Z})^{-1}\mathbf{s}(\theta)(\boldsymbol{\Phi} + \delta\mathbf{Z})^{-1}\mathbf{s}(\theta)}{\mathbf{s}^{\mathrm{H}}(\theta)(\boldsymbol{\Phi} + \delta\mathbf{Z})^{-1}\mathbf{s}(\theta)}.\end{aligned} \tag{9.99}$$

The following points from Eq. (9.99) are noteworthy:

- In the limiting case of $\delta = 0$ (i.e., no regularization), Eq. (9.99) reduces to the unregularized MVDR solution defined in Eq. (9.85).
- Despite the complicated appearance of Eq. (9.99), practical implementation of this regularized solution is much less computationally demanding than a subspace projection algorithm based on singular-value decomposition (McWhirter et al., 2000). (Singular value decomposition is discussed in the next section.)

Figure 9.7, which plots the power response versus the actual direction of arrival, φ, illustrates the impressive performance that is attainable with a regularized MVDR

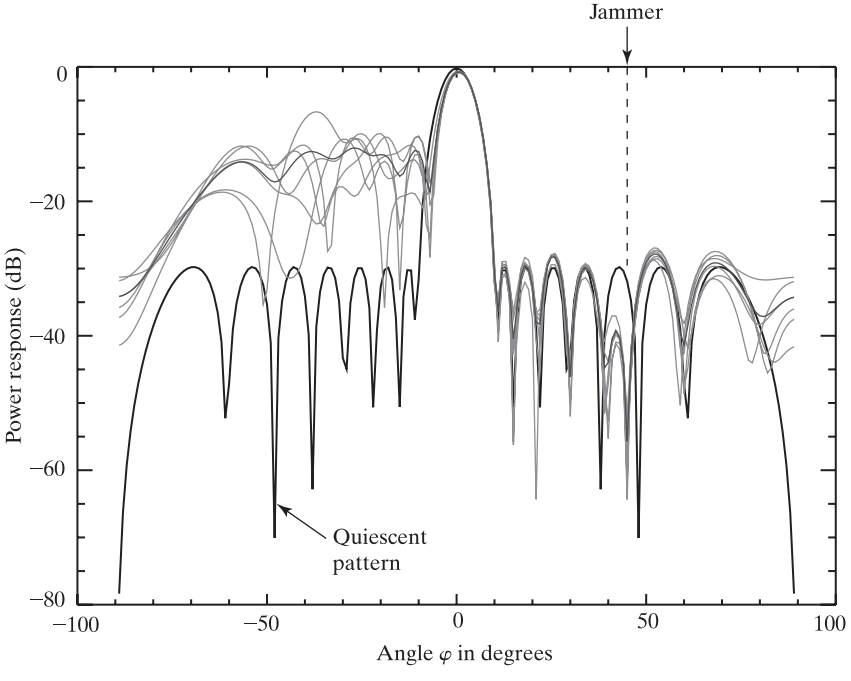


FIGURE 9.7 Sidelobe stabilization using the regularized MVDR beamformer. Solid curve shows the quiescent beampattern. (Reproduced with the permission of the British Crown.)

beamformer based on Eq. (9.99). The results presented herein, due to McWhirter et al. (2000), were computed for the following specifications:

- 16-element linear array with uniform half-wavelength spacing.
- Look direction $\varphi = 0$.
- Additive white Gaussian noise, spatially uncorrelated from one antenna element to the next.
- Single jammer at $\varphi = 45°$, with a power of 30 dB relative to the noise at each antenna element of the array.
- Number of snapshots used in the computation $K = 32$.
- Weighting function

$$h(\theta) = \begin{cases} \cos\theta & \text{for } \theta > 0° \\ 0 & \text{for } \theta < 0° \end{cases}.$$

  For $\theta > 0°$, the use of this weighting function constrains the antenna pattern to lie close to that of a fixed Chebyshev pattern with a sidelobe level of −30 dB. The antenna pattern was left unregularized for $\theta < 0°$ for illustration purposes only, as there was no jammer present in that field of view.
- The quiescent weight vector $\mathbf{w}_q$ is specified by the *Dolph–Chebyshev antenna pattern* (Dolph, 1946).
- Regularization parameter $\delta = 30K = 960$.

From the figure, we see that the use of regularization for $\varphi > 0°$ has a profound effect on the antenna pattern for the field of view. Specifically, the fluctuations in the weight vector from one block of input data to the next are dramatically reduced, while the sidelobes are controlled to lie close to the prescribed Dolph–Chebyshev levels. By the same token, for the unregularized field of view represented by $\varphi < 0°$ with no jammers, we see significant fluctuations in the antenna pattern, as well as a poor sidelobe performance. Moreover, the regularized beamformer is still producing a response of 0 dB in the prescribed look direction of $\varphi = 0°$ and forming a null in the direction of the jammer at $\varphi = 45°$.

9.11 SINGULAR-VALUE DECOMPOSITION

We complete the discussion of the method of least squares by describing a computational tool known as *singular-value decomposition*. The analytic power of this tool lies in the fact that it applies to square as well as rectangular matrices, be they real or complex. Hence, singular-value decomposition is extremely well suited to the numerical solution of linear least-squares problems, in the sense that *it can be applied directly to the data matrix*.

In Sections 9.5 and 9.7, we described two different forms of the normal equations for computing the linear least-squares solution:

1. The form given in Eq. (9.36), namely,

$$\hat{\mathbf{w}} = \boldsymbol{\Phi}^{-1}\mathbf{z},$$

where $\hat{\mathbf{w}}$ is the least-squares estimate of the unknown parameter vector of a multiple linear regression model, $\mathbf{\Phi}$ is the time-average correlation matrix of the tap inputs of an FIR filter performing the estimation, and $\mathbf{z}$ is the time-average cross-correlation vector between the tap inputs and the desired response.

2. The form given in Eq (9.48) *directly in terms of data matrices*, namely,

$$\hat{\mathbf{w}} = (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}\mathbf{d},$$

where $\mathbf{A}$ is the data matrix representing the time evolution of the tap-input vectors and $\mathbf{d}$ is the desired data vector representing the time evolution of the desired response.

These two forms are indeed mathematically equivalent, yet they point to different computational procedures for evaluating the least-squares solution $\hat{\mathbf{w}}$. Equation (9.36) requires knowledge of the time-average correlation matrix $\mathbf{\Phi}$ that involves computing the product of $\mathbf{A}^{\mathrm{H}}$ and $\mathbf{A}$. On the other hand, in Eq. (9.48), the entire term $(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}$ can be interpreted, in terms of the singular-value decomposition applied directly to the data matrix $\mathbf{A}$, in such a way that the solution computed for $\hat{\mathbf{w}}$ has *twice the number of correct digits* as the solution computed by means of Eq. (9.36) for the same numerical precision. To be specific, if we define the matrix

$$\mathbf{A}^{+} = (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}, \tag{9.100}$$

then we may rewrite Eq. (9.36) simply as

$$\hat{\mathbf{w}} = \mathbf{A}^{+}\mathbf{d}. \tag{9.101}$$

The matrix $\mathbf{A}^{+}$ is called the *pseudoinverse*, or the *Moore–Penrose generalized inverse*, of the matrix $\mathbf{A}$ (Stewart, 1973; Golub & Van Loan, 1996). Equation (9.101) represents a convenient way of saying:

The vector $\hat{\mathbf{w}}$ solves the linear least-squares problem.

Indeed, it was with the simple format of that equation in mind and also the desire to be consistent with definitions of the time-average correlation matrix $\mathbf{\Phi}$ and the cross-correlation vector $\mathbf{z}$ used in Section 9.5 that we defined the data matrix $\mathbf{A}$ and the desired data vector $\mathbf{d}$ in the manner shown in Eqs. (9.44) and (9.46), respectively.

In practice, we often find that the data matrix $\mathbf{A}$ contains linearly dependent columns. Consequently, we are faced with a new situation in which we now have to decide which of an infinite number of possible solutions to the least-squares problem to work with. This issue can indeed be resolved by using the singular-value decomposition technique to be described in Section 9.14.

The Singular-Value Decomposition Theorem

The *singular-value decomposition (SVD)* of a matrix is one of the most elegant algorithms in numerical algebra for providing quantitative information about the structure of a system of linear equations (Klema & Laub, 1980). The system of linear equations that is of specific interest to us is described by

$$\mathbf{A}\hat{\mathbf{w}} = \mathbf{d}, \tag{9.102}$$

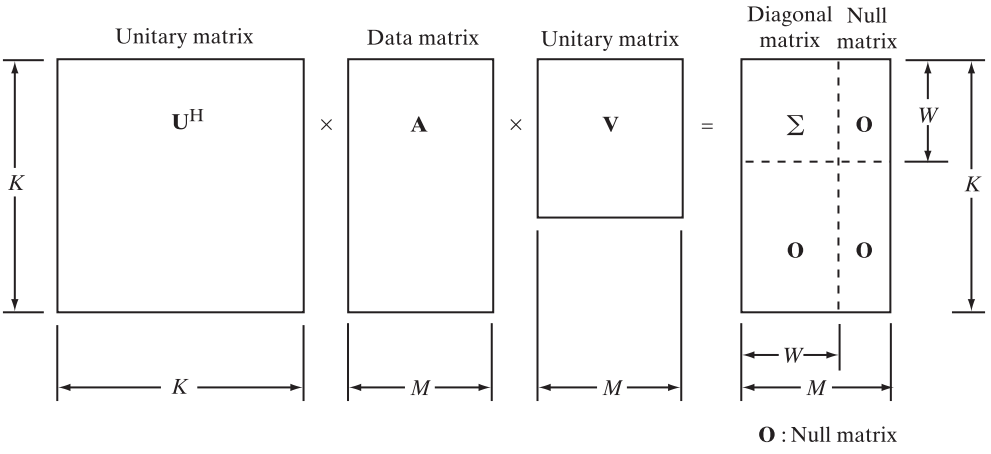


FIGURE 9.8 Diagrammatic interpretation of the singular-value decomposition theorem.

in which $\mathbf{A}$ is a K-by-M matrix, $\mathbf{d}$ is a K-by-1 vector, and $\hat{\mathbf{w}}$ (representing an estimate of the unknown parameter vector) is an M-by-1 vector. Equation (9.102) represents a simplified matrix form of the normal equations. In particular, premultiplication of both sides of the equation by the vector $\mathbf{A}^H$ yields the normal equations for the least-squares weight vector $\hat{\mathbf{w}}$, as defined in Eq. (9.48).

Given the data matrix $\mathbf{A}$, there are two unitary matrices $\mathbf{V}$ and $\mathbf{U}$ such that

$$\mathbf{U}^H\mathbf{A}\mathbf{V} = \begin{bmatrix} \mathbf{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{9.103}$$

where

$$\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_W) \tag{9.104}$$

is a diagonal matrix. The σ's are ordered as $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_W > 0$. Equation (9.103) is a mathematical statement of the *singular-value decomposition theorem*, also referred to as the *Autonne–Eckart–Young theorem* in recognition of its originators.[5]

Figure 9.8 presents a diagrammatic interpretation of the singular-value decomposition theorem, as described in Eq. (9.103). In this diagram, we have assumed that the number of rows, K, contained in the data matrix $\mathbf{A}$ is larger than the number of columns, M, and that the number of nonzero singular values W is less than M. We may, of course, restructure the diagrammatic interpretation of the singular-value decomposition theorem by expressing the data matrix in terms of the unitary matrices $\mathbf{U}$ and $\mathbf{V}$ and the diagonal matrix $\mathbf{\Sigma}$; this is left as an exercise for the reader.

The subscript W in Eq. (9.104) is the *rank* of matrix $\mathbf{A}$, written as rank($\mathbf{A}$); it is defined as the number of linearly independent columns in $\mathbf{A}$. Note that we always have rank($\mathbf{A}^H$) = rank($\mathbf{A}$).

[5]Singular-value decomposition was introduced in its general form by Autonne in 1902, and an important characterization of it was described by Eckart and Young (1936). For additional notes on the history of the singular-value decomposition, see Klema and Laub (1980).

Since it is possible to have $K > M$ or $K < M$, there are two distinct cases to be considered. We prove the singular-value decomposition theorem by considering both cases, independently of each other. For the case when $K > M$, we have an *overdetermined system*, in that we have more equations than unknowns. On the other hand, when $K < M$, we have an *underdetermined system*, in that we have more unknowns than equations. In what follows, we consider these two cases in turn.

Case 1: Overdetermined System. For the case when $K > M$, we form the M-by-M matrix $\mathbf{A}^{\mathrm{H}}\mathbf{A}$ by premultiplying the matrix $\mathbf{A}$ by its Hermitian transpose $\mathbf{A}^{\mathrm{H}}$. Since the matrix $\mathbf{A}^{\mathrm{H}}\mathbf{A}$ is Hermitian and nonnegative definite, its eigenvalues are all real nonnegative numbers. Let these eigenvalues be denoted by $\sigma_1^2, \sigma_2^2, \ldots, \sigma_M^2$, where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_W > 0$ and $\sigma_{W+1}, \sigma_{W+2}, \ldots$ are all zero, with $1 \leq W \leq M$. The matrix $\mathbf{A}^{\mathrm{H}}\mathbf{A}$ has the same rank as $\mathbf{A}$; hence, there are W nonzero eigenvalues. Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M$ denote a set of orthonormal eigenvectors of $\mathbf{A}^{\mathrm{H}}\mathbf{A}$ that are associated with the eigenvalues $\sigma_1^2, \sigma_2^2, \ldots, \sigma_M^2$, respectively. Also, let $\mathbf{V}$ denote the M-by-M unitary matrix whose columns are made up of the eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M$. Then, using the eigendecomposition of the matrix $\mathbf{A}^{\mathrm{H}}\mathbf{A}$, we may write

$$\mathbf{V}^{\mathrm{H}}\mathbf{A}^{\mathrm{H}}\mathbf{A}\mathbf{V} = \begin{bmatrix} \boldsymbol{\Sigma}^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{9.105}$$

Let the unitary matrix $\mathbf{V}$ be partitioned as

$$\mathbf{V} = [\mathbf{V}_1, \mathbf{V}_2], \tag{9.106}$$

where

$$\mathbf{V}_1 = [\mathbf{v}_1, \mathbf{v}_2, \ \ldots, \ \mathbf{v}_W] \tag{9.107}$$

is an M-by-W matrix and

$$\mathbf{V}_2 = [\mathbf{v}_{W+1}, \mathbf{v}_{W+2}, \ldots, \mathbf{v}_M] \tag{9.108}$$

is an M-by-$(M - W)$ matrix, with

$$\mathbf{V}_1^{\mathrm{H}}\mathbf{V}_2 = \mathbf{0}. \tag{9.109}$$

We may make the following two deductions from Eq. (9.105):

1. For matrix $\mathbf{V}_1$, we have

   $$\mathbf{V}_1^{\mathrm{H}}\mathbf{A}^{\mathrm{H}}\mathbf{A}\mathbf{V}_1 = \boldsymbol{\Sigma}^2.$$

   Consequently,

   $$\boldsymbol{\Sigma}^{-1}\mathbf{V}_1^{\mathrm{H}}\mathbf{A}^{\mathrm{H}}\mathbf{A}\mathbf{V}_1\boldsymbol{\Sigma}^{-1} = \mathbf{I}. \tag{9.110}$$

2. For matrix $\mathbf{V}_2$, we have

   $$\mathbf{V}_2^{\mathrm{H}}\mathbf{A}^{\mathrm{H}}\mathbf{A}\mathbf{V}_2 = \mathbf{0}.$$

   Consequently,

   $$\mathbf{A}\mathbf{V}_2 = \mathbf{0}. \tag{9.111}$$

We now define a new K-by-W matrix

$$\mathbf{U}_1 = \mathbf{A}\mathbf{V}_1\boldsymbol{\Sigma}^{-1}. \tag{9.112}$$

Then, from Eq. (9.110), it follows that

$$\mathbf{U}_1^{\mathrm{H}}\mathbf{U}_1 = \mathbf{I}, \tag{9.113}$$

which means that the columns of the matrix $\mathbf{U}_1$ are orthonormal with respect to each other. Next, we choose another K-by-$(K - W)$ matrix $\mathbf{U}_2$ such that the K-by-K matrix formed from $\mathbf{U}_1$ and $\mathbf{U}_2$, namely,

$$\mathbf{U} = [\mathbf{U}_1, \mathbf{U}_2], \tag{9.114}$$

is a unitary matrix. This means that

$$\mathbf{U}_1^{\mathrm{H}}\mathbf{U}_2 = \mathbf{0}. \tag{9.115}$$

Accordingly, we may use Eqs. (9.106), (9.114), (9.111), (9.112), and (9.115), in that order, and so write

$$\begin{aligned}
\mathbf{U}^{\mathrm{H}}\mathbf{A}\mathbf{V} &= \begin{bmatrix} \mathbf{U}_1^{\mathrm{H}} \\ \mathbf{U}_2^{\mathrm{H}} \end{bmatrix} \mathbf{A}[\mathbf{V}_1, \mathbf{V}_2] \\
&= \begin{bmatrix} \mathbf{U}_1^{\mathrm{H}}\mathbf{A}\mathbf{V}_1 & \mathbf{U}_1^{\mathrm{H}}\mathbf{A}\mathbf{V}_2 \\ \mathbf{U}_2^{\mathrm{H}}\mathbf{A}\mathbf{V}_1 & \mathbf{U}_2^{\mathrm{H}}\mathbf{A}\mathbf{V}_2 \end{bmatrix} \\
&= \begin{bmatrix} (\boldsymbol{\Sigma}^{-1}\mathbf{V}_1^{\mathrm{H}}\mathbf{A}^{\mathrm{H}})\mathbf{A}\mathbf{V}_1 & \mathbf{U}_1^{\mathrm{H}}(\mathbf{0}) \\ \mathbf{U}_2^{\mathrm{H}}(\mathbf{U}_1\boldsymbol{\Sigma}) & \mathbf{U}_2^{\mathrm{H}}(\mathbf{0}) \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix},
\end{aligned}$$

which proves Eq. (9.103) for the overdetermined case.

Case 2: Underdetermined System. Consider next the case when $K < M$. This time we form the K-by-K matrix $\mathbf{A}\mathbf{A}^{\mathrm{H}}$ by postmultiplying the matrix $\mathbf{A}$ by its Hermitian transpose $\mathbf{A}^{\mathrm{H}}$. The matrix $\mathbf{A}\mathbf{A}^{\mathrm{H}}$ is also Hermitian and nonnegative definite, so its eigenvalues are likewise real nonnegative numbers. The nonzero eigenvalues of $\mathbf{A}\mathbf{A}^{\mathrm{H}}$ are the *same* as those of $\mathbf{A}^{\mathrm{H}}\mathbf{A}$**.** We may therefore denote the eigenvalues of $\mathbf{A}\mathbf{A}^{\mathrm{H}}$ as $\sigma_1^2, \sigma_2^2, \ldots, \sigma_K^2$, where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_W > 0$ and $\sigma_{W+1}, \sigma_{W+2}, \ldots$ are all zero, with $1 \leq W \leq K$. Let $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K$ denote a set of orthonormal eigenvectors of the matrix $\mathbf{A}\mathbf{A}^{\mathrm{H}}$ that are associated with the eigenvalues $\sigma_1^2, \sigma_2^2, \ldots, \sigma_K^2$, respectively. Also, let $\mathbf{U}$ denote the unitary matrix whose columns are made up of the eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K$. Thus, using the eigendecomposition of $\mathbf{A}\mathbf{A}^{\mathrm{H}}$, we may write

$$\mathbf{U}^{\mathrm{H}}\mathbf{A}\mathbf{A}^{\mathrm{H}}\mathbf{U} = \begin{bmatrix} \boldsymbol{\Sigma}^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{9.116}$$

Let the unitary matrix $\mathbf{U}$ be partitioned as

$$\mathbf{U} = [\mathbf{U}_1, \mathbf{U}_2], \tag{9.117}$$

where

$$\mathbf{U}_1 = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_W], \tag{9.118}$$

$$\mathbf{U}_2 = [\mathbf{u}_{W+1}, \mathbf{u}_{W+2}, \ldots, \mathbf{u}_K], \tag{9.119}$$

and

$$\mathbf{U}_1^{\mathrm{H}}\mathbf{U}_2 = \mathbf{0}. \tag{9.120}$$

We may therefore make two deductions from Eq. (9.116):

1. For matrix $\mathbf{U}_1$, we have

$$\mathbf{U}_1^{\mathrm{H}}\mathbf{A}\mathbf{A}^{\mathrm{H}}\mathbf{U}_1 = \boldsymbol{\Sigma}^2.$$

Consequently,

$$\boldsymbol{\Sigma}^{-1}\mathbf{U}_1^{\mathrm{H}}\mathbf{A}\mathbf{A}^{\mathrm{H}}\mathbf{U}_1\boldsymbol{\Sigma}^{-1} = \mathbf{I}. \tag{9.121}$$

2. For matrix $\mathbf{U}_2$, we have

$$\mathbf{U}_2^{\mathrm{H}}\mathbf{A}\mathbf{A}^{\mathrm{H}}\mathbf{U}_2 = \mathbf{0}.$$

Consequently,

$$\mathbf{A}^{\mathrm{H}}\mathbf{U}_2 = \mathbf{0}. \tag{9.122}$$

We now define an M-by-W matrix

$$\mathbf{V}_1 = \mathbf{A}^{\mathrm{H}}\mathbf{U}_1\boldsymbol{\Sigma}^{-1}. \tag{9.123}$$

Then, from Eq. (9.121), it follows that

$$\mathbf{V}_1^{\mathrm{H}}\mathbf{V}_1 = \mathbf{I}, \tag{9.124}$$

which means that the columns of the matrix $\mathbf{V}_1$ are orthonormal with respect to each other. Next, we choose another M-by-$(M - W)$ matrix $\mathbf{V}_2$ such that the M-by-M matrix formed from $\mathbf{V}_1$ and $\mathbf{V}_2$, namely,

$$\mathbf{V} = [\mathbf{V}_1, \mathbf{V}_2], \tag{9.125}$$

is a unitary matrix. This means that

$$\mathbf{V}_2^{\mathrm{H}}\mathbf{V}_1 = \mathbf{0}. \tag{9.126}$$

Accordingly, we may use Eqs. (9.117), (9.125), (9.122), (9.123), and (9.126), in that order, and so write

$$\begin{aligned}
\mathbf{U}^{\mathrm{H}}\mathbf{A}\mathbf{V} &= \begin{bmatrix} \mathbf{U}_1^{\mathrm{H}} \\ \mathbf{U}_2^{\mathrm{H}} \end{bmatrix} \mathbf{A}[\mathbf{V}_1, \mathbf{V}_2] \\
&= \begin{bmatrix} \mathbf{U}_1^{\mathrm{H}}\mathbf{A}\mathbf{V}_1 & \mathbf{U}_1^{\mathrm{H}}\mathbf{A}\mathbf{V}_2 \\ \mathbf{U}_2^{\mathrm{H}}\mathbf{A}\mathbf{V}_1 & \mathbf{U}_2^{\mathrm{H}}\mathbf{A}\mathbf{V}_2 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{U}_1^{\mathrm{H}}\mathbf{A}(\mathbf{A}^{\mathrm{H}}\mathbf{U}_1\boldsymbol{\Sigma}^{-1}) & (\boldsymbol{\Sigma}\mathbf{V}_1^{\mathrm{H}})\mathbf{V}_2 \\ (\mathbf{0})\mathbf{V}_1 & (\mathbf{0})\mathbf{V}_2 \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.
\end{aligned}$$

This proves Eq. (9.103) for the underdetermined case, and with it, the proof of the singular-value decomposition theorem is completed.

Terminology and Relation to Eigenanalysis

The numbers $\sigma_1, \sigma_2, \ldots, \sigma_W$, constituting the diagonal matrix $\mathbf{\Sigma}$, are called the *singular values* of the matrix $\mathbf{A}$. The columns of the unitary matrix $\mathbf{V}$ (i.e., $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M$) are the *right singular vectors* of $\mathbf{A}$, and the columns of the second unitary matrix $\mathbf{U}$ (i.e., $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K$) are the *left singular vectors* of $\mathbf{A}$. We note from the preceding discussion that the right singular vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M$ are eigenvectors of $\mathbf{A}^{\mathrm{H}}\mathbf{A}$, whereas the left singular vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K$ are eigenvectors of $\mathbf{A}\mathbf{A}^{\mathrm{H}}$. Note that the number of positive singular values is equal to the rank of the data matrix $\mathbf{A}$. The singular-value decomposition therefore provides the basis of a practical method for determining the rank of a matrix.

Since $\mathbf{U}\mathbf{U}^{\mathrm{H}}$ equals the identity matrix, we find, from Eq. (9.103), that

$$\mathbf{A}\mathbf{V} = \mathbf{U}\begin{bmatrix} \mathbf{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Therefore,

$$\mathbf{A}\mathbf{v}_i = \sigma_i \mathbf{u}_i, \qquad i = 1, 2, \ldots, W,$$

and (9.127)

$$\mathbf{A}\mathbf{v}_i = \mathbf{0}, \qquad i = W + 1, \ldots, K.$$

Correspondingly, we may express the data matrix $\mathbf{A}$ in the expanded form

$$\mathbf{A} = \sum_{i=1}^{W} \sigma_i \mathbf{u}_i \mathbf{v}_i^{\mathrm{H}}. \tag{9.128}$$

Since $\mathbf{V}\mathbf{V}^{\mathrm{H}}$ equals the identity matrix, we also find from Eq. (9.103) that

$$\mathbf{U}^{\mathrm{H}}\mathbf{A} = \begin{bmatrix} \mathbf{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\mathbf{V}^{\mathrm{H}}$$

or, equivalently,

$$\mathbf{A}^{\mathrm{H}}\mathbf{U} = \mathbf{V}\begin{bmatrix} \mathbf{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

It follows that

$$\mathbf{A}^{\mathrm{H}}\mathbf{u}_i = \sigma_i \mathbf{v}_i, \qquad i = 1, 2, \ldots, W,$$

and (9.129)

$$\mathbf{A}^{\mathrm{H}}\mathbf{u}_i = \mathbf{0}, \qquad i = W + 1, \ldots, M.$$

In this case, we may express the Hermitian transpose of the data matrix $\mathbf{A}$ in the expanded form

$$\mathbf{A}^{\mathrm{H}} = \sum_{i=1}^{W} \sigma_i \mathbf{v}_i \mathbf{u}_i^{\mathrm{H}}, \tag{9.130}$$

which checks exactly with Eq. (9.128), as it should.

EXAMPLE 2

In this example, we use the SVD to deal with the different facets of *matrix rank*. Let $\mathbf{A}$ be a K-by-M data matrix with rank W. Then $\mathbf{A}$ is said to be of *full rank* if

$$W = \min(K, M).$$

Otherwise, the matrix $\mathbf{A}$ is *rank deficient*. As mentioned previously, the rank W is simply the number of nonzero singular values of matrix $\mathbf{A}$.

Consider next a computational environment that yields a numerical value for each element of $\mathbf{A}$ that is accurate to within $\pm\varepsilon$. Let $\mathbf{B}$ denote the approximate value of $\mathbf{A}$ so obtained. We define the ε-*rank* of matrix $\mathbf{A}$ (Golub & Van Loan, 1996) as

$$\operatorname{rank}(\mathbf{A}, \varepsilon) = \min_{\|\mathbf{A}-\mathbf{B}\| < \varepsilon} \operatorname{rank}(\mathbf{B}), \tag{9.131}$$

where $\|\mathbf{A} - \mathbf{B}\|$ is the *spectral norm* of the error matrix $\mathbf{A} - \mathbf{B}$ that results from the use of inaccurate computations. Extending the definition of the spectral norm of the matrix introduced in Appendix E to the situation at hand, we find that the spectral norm $\|\mathbf{A} - \mathbf{B}\|$ equals the largest singular value of the difference $\mathbf{A} - \mathbf{B}$. In any event, the K-by-M matrix $\mathbf{A}$ is said to be *numerically rank deficient* if

$$\operatorname*{rank}_{\|\mathbf{A}-\mathbf{B}\| < \varepsilon}(\mathbf{A}, \varepsilon) < \min(K, M).$$

The SVD provides a sensible method for characterizing the ε-rank and the numerical rank deficiency of the matrix, because the singular values resulting from its use indicate, in a simple fashion, how close a given matrix $\mathbf{A}$ is to another matrix $\mathbf{B}$ of lower rank.

9.12 PSEUDOINVERSE

Our interest in the SVD is to formulate a general definition of the pseudoinverse. Let $\mathbf{A}$ denote a K-by-M matrix that has the SVD described in Eq. (9.103). We define the pseudoinverse of the data matrix $\mathbf{A}$ (Stewart, 1973; Golub & Van Loan, 1996) as

$$\mathbf{A}^{+} = \mathbf{V} \begin{bmatrix} \boldsymbol{\Sigma}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{U}^{\mathrm{H}}, \tag{9.132}$$

where

$$\boldsymbol{\Sigma}^{-1} = \operatorname{diag}(\sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_W^{-1})$$

and W is the rank of $\mathbf{A}$. The pseudoinverse $\mathbf{A}^{+}$ may be expressed in the expanded form

$$\mathbf{A}^{+} = \sum_{i=1}^{W} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^{\mathrm{H}}. \tag{9.133}$$

We next identify two special cases that can arise.

Case 1: Overdetermined System. In this case, $K > M$, and we assume that the rank W equals M, so that the inverse matrix $(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}$ exists. The pseudoinverse of the data matrix $\mathbf{A}$ is then defined by

$$\mathbf{A}^{+} = (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}}. \tag{9.134}$$

To demonstrate the validity of this special formula, we note respectively from Eqs. (9.110) and (9.112) that

$$(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1} = \mathbf{V}_1\boldsymbol{\Sigma}^{-2}\mathbf{V}_1^{\mathrm{H}}$$

and

$$\mathbf{A}^{\mathrm{H}} = \mathbf{V}_1\boldsymbol{\Sigma}\mathbf{U}_1^{\mathrm{H}}.$$

Therefore, using this pair of relations, we may express the right-hand side of Eq. (9.134) as

$$\begin{aligned}(\mathbf{A}^{\mathrm{H}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{H}} &= (\mathbf{V}_1\boldsymbol{\Sigma}^{-2}\mathbf{V}_1^{\mathrm{H}})(\mathbf{V}_1\boldsymbol{\Sigma}\mathbf{U}_1^{\mathrm{H}}) \\ &= \mathbf{V}_1\boldsymbol{\Sigma}^{-1}\mathbf{U}_1^{\mathrm{H}} \\ &= \mathbf{V}\begin{bmatrix}\boldsymbol{\Sigma}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{U}^{\mathrm{H}} \\ &= \mathbf{A}^{+}.\end{aligned}$$

Case 2: Underdetermined System. In this second case, we have $M > K$, and we assume that the rank W equals K, so that the inverse matrix $(\mathbf{A}\mathbf{A}^{\mathrm{H}})^{-1}$ exists. The pseudoinverse of the data matrix $\mathbf{A}$ is now defined by

$$\mathbf{A}^{+} = \mathbf{A}^{\mathrm{H}}(\mathbf{A}\mathbf{A}^{\mathrm{H}})^{-1}. \tag{9.135}$$

To demonstrate the validity of this second special formula, we note from Eqs. (9.121) and (9.123), respectively, that

$$(\mathbf{A}\mathbf{A}^{\mathrm{H}})^{-1} = \mathbf{U}_1\boldsymbol{\Sigma}^{-2}\mathbf{U}_1^{\mathrm{H}}$$

and

$$\mathbf{A}^{\mathrm{H}} = \mathbf{V}_1\boldsymbol{\Sigma}\mathbf{U}_1^{\mathrm{H}}.$$

Therefore, using this pair of relations in the right-hand side of Eq. (9.135), we get

$$\begin{aligned}\mathbf{A}^{\mathrm{H}}(\mathbf{A}\mathbf{A}^{\mathrm{H}})^{-1} &= (\mathbf{V}_1\boldsymbol{\Sigma}\mathbf{U}_1^{\mathrm{H}})(\mathbf{U}_1\boldsymbol{\Sigma}^{-2}\mathbf{U}_1^{\mathrm{H}}) \\ &= \mathbf{V}_1\boldsymbol{\Sigma}^{-1}\mathbf{U}_1^{\mathrm{H}} \\ &= \mathbf{V}\begin{bmatrix}\boldsymbol{\Sigma}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{U}^{\mathrm{H}} \\ &= \mathbf{A}^{+}.\end{aligned}$$

Note that the pseudoinverse $\mathbf{A}^{+}$ as described in Eq. (9.132) or, equivalently, Eq. (9.133) is of general application in that it applies whether the data matrix $\mathbf{A}$ refers to an overdetermined or an underdetermined system and regardless of what the rank W is. Most importantly, it is numerically stable.

9.13 INTERPRETATION OF SINGULAR VALUES AND SINGULAR VECTORS

Consider a K-by-M data matrix $\mathbf{A}$, for which the SVD is given in Eq. (9.103) and the pseudoinverse is correspondingly given in Eq. (9.132). We assume that the system is overdetermined, and we define a K-by-1 vector $\mathbf{y}$ and an M-by-1 vector $\mathbf{x}$ that are related to each other by the transformation matrix $\mathbf{A}$:

$$\mathbf{y} = \mathbf{A}\mathbf{x}. \tag{9.136}$$

The vector $\mathbf{x}$ is constrained to have a Euclidean norm of unity; that is,

$$\|\mathbf{x}\| = 1. \tag{9.137}$$

Given the transformation of Eq. (9.136) and the constraint of Eq. (9.137), we wish to find the resulting locus of the points defined by the vector $\mathbf{y}$ in a K-dimensional space.

Solving Eq. (9.136) for $\mathbf{x}$, we get

$$\mathbf{x} = \mathbf{A}^{+}\mathbf{y}, \tag{9.138}$$

where $\mathbf{A}^{+}$ is the pseudoinverse of $\mathbf{A}$. Substituting Eq. (9.133) into Eq. (9.138), we get

$$\begin{aligned}\mathbf{x} &= \sum_{i=1}^{W} \frac{1}{\sigma_i}\mathbf{v}_i\mathbf{u}_i^{\mathrm{H}}\mathbf{y} \\ &= \sum_{i=1}^{W} \frac{(\mathbf{u}_i^{\mathrm{H}}\mathbf{y})}{\sigma_i}\mathbf{v}_i,\end{aligned} \tag{9.139}$$

where W is the rank of matrix $\mathbf{A}$ and the inner product $\mathbf{u}_i^{\mathrm{H}}\mathbf{y}$ is a scalar. Imposing the constraint of Eq. (9.137) on Eq. (9.139) and recognizing that the right singular vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_W$ form an orthonormal set, we obtain

$$\sum_{i=1}^{W} \frac{|\mathbf{y}^{\mathrm{H}}\mathbf{u}_i|^2}{\sigma_i^2} = 1. \tag{9.140}$$

Equation (9.140) defines the locus traced out by the tip of vector $\mathbf{y}$ in a K-dimensional space. Indeed, this is the equation of a *hyperellipsoid*.

To see this interpretation in a better way, we define the complex scalar

$$\begin{aligned}\zeta_i &= \mathbf{y}^{\mathrm{H}}\mathbf{u}_i \\ &= \sum_{k=1}^{K} y_k^{*} u_{ik}, \qquad i = 1, \ldots, W.\end{aligned} \tag{9.141}$$

In other words, ζ_i is a linear combination of all possible values of the elements of the left singular vector $\mathbf{u}_i$, so ζ_i is referred to as the "span" of $\mathbf{u}_i$. We may thus rewrite Eq. (9.140) as

$$\sum_{i=1}^{W} \frac{|\zeta_i|^2}{\sigma_i^2} = 1. \tag{9.142}$$

This is the equation of a hyperellipsoid with coordinates $|\zeta_1|, \ldots, |\zeta_W|$ and with semiaxes whose lengths are the singular values $\sigma_1, \ldots, \sigma_W$, respectively. Figure 9.9 illustrates the

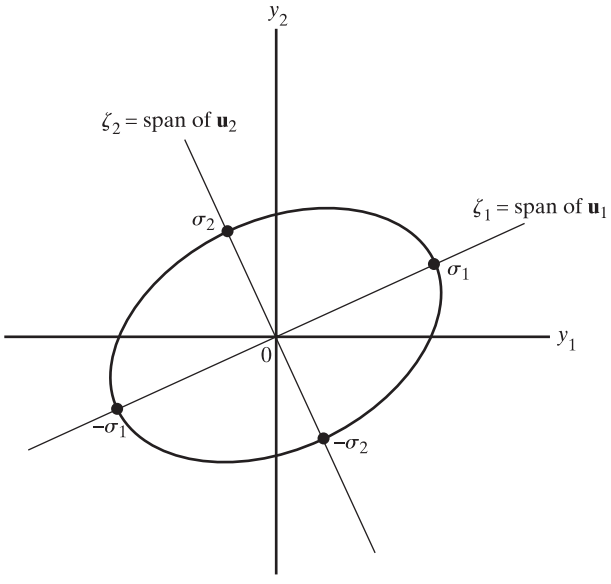


FIGURE 9.9 Locus of Eq. (9.140) for real data with $W = 2$ and $\sigma_1 > \sigma_2$.

locus traced out by Eq. (9.140) for the case of $W = 2$ and $\sigma_1 > \sigma_2$, assuming that the data matrix $\mathbf{A}$ is real.

9.14 MINIMUM-NORM SOLUTION TO THE LINEAR LEAST-SQUARES PROBLEM

Having equipped ourselves with the general definition of the pseudoinverse of a matrix $\mathbf{A}$ in terms of its SVD, we are now ready to tackle the solution to the linear least-squares problem even when null($\mathbf{A}$) $\neq \varnothing$, where $\varnothing$ denotes the *null set*. Recapping, we define the solution to the least-squares problem as in Eq. (9.101), reproduced here for convenience:

$$\hat{\mathbf{w}} = \mathbf{A}^{+}\mathbf{d}. \tag{9.143}$$

The pseudoinverse matrix $\mathbf{A}^{+}$ is itself defined by Eq. (9.132). We thus find that, out of the many vectors that solve the least-squares problem when null($\mathbf{A}$) $\neq \varnothing$, the one defined by Eq. (9.143) is *unique* in that it has the *shortest length possible in the Euclidean sense* (Stewart, 1973).

We prove this important result by manipulating the equation that defines the minimum value of the sum of error squares produced in the method of least squares. We note that both matrix products $\mathbf{V}\mathbf{V}^{\mathrm{H}}$ and $\mathbf{U}\mathbf{U}^{\mathrm{H}}$ equal identity matrices. Hence, we may start with Eq. (9.49), combine it with Eq. (9.48), and then write

$$\begin{aligned}
\mathscr{E}_{\min} &= \mathbf{d}^{\mathrm{H}}\mathbf{d} - \mathbf{d}^{\mathrm{H}}\mathbf{A}\hat{\mathbf{w}} \\
&= \mathbf{d}^{\mathbf{H}}(\mathbf{d} - \mathbf{A}\hat{\mathbf{w}}) \\
&= \mathbf{d}^{\mathrm{H}}\mathbf{U}\mathbf{U}^{\mathrm{H}}(\mathbf{d} - \mathbf{A}\mathbf{V}\mathbf{V}^{\mathrm{H}}\hat{\mathbf{w}}) \\
&= \mathbf{d}^{\mathrm{H}}\mathbf{U}(\mathbf{U}^{\mathrm{H}}\mathbf{d} - \mathbf{U}^{\mathrm{H}}\mathbf{A}\mathbf{V}\mathbf{V}^{\mathrm{H}}\hat{\mathbf{w}}).
\end{aligned} \tag{9.144}$$

Let

$$\begin{aligned}\mathbf{V}^{\mathrm{H}}\hat{\mathbf{w}} &= \mathbf{b} \\ &= \begin{bmatrix}\mathbf{b}_1 \\ \mathbf{b}_2\end{bmatrix}\end{aligned} \tag{9.145}$$

and

$$\begin{aligned}\mathbf{U}^{\mathrm{H}}\mathbf{d} &= \mathbf{c} \\ &= \begin{bmatrix}\mathbf{c}_1 \\ \mathbf{c}_2\end{bmatrix},\end{aligned} \tag{9.146}$$

where $\mathbf{b}_1$ and $\mathbf{c}_1$ are W-by-1 vectors and $\mathbf{b}_2$ and $\mathbf{c}_2$ are two other vectors. Thus, substituting Eqs. (9.103), (9.145), and (9.146) into Eq. (9.144), we get

$$\begin{aligned}\mathscr{E}_{\min} &= \mathbf{d}^{\mathrm{H}}\mathbf{U}\left(\begin{bmatrix}\mathbf{c}_1 \\ \mathbf{c}_2\end{bmatrix} - \begin{bmatrix}\boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{b}_1 \\ \mathbf{b}_2\end{bmatrix}\right) \\ &= \mathbf{d}^{\mathrm{H}}\mathbf{U}\begin{bmatrix}\mathbf{c}_1 - \boldsymbol{\Sigma}\mathbf{b}_1 \\ \mathbf{c}_2\end{bmatrix}.\end{aligned} \tag{9.147}$$

For $\mathscr{E}_{\min}$ to be minimum, we require that

$$\mathbf{c}_1 = \boldsymbol{\Sigma}\mathbf{b}_1 \tag{9.148}$$

or, equivalently,

$$\mathbf{b}_1 = \boldsymbol{\Sigma}^{-1}\mathbf{c}_1. \tag{9.149}$$

We observe that $\mathscr{E}_{\min}$ is independent of $\mathbf{b}_2$; hence, the value of $\mathbf{b}_2$ is arbitrary. However, if we let $\mathbf{b}_2 = \mathbf{0}$, we get the special result

$$\begin{aligned}\hat{\mathbf{w}} &= \mathbf{V}\mathbf{b} \\ &= \mathbf{V}\begin{bmatrix}\boldsymbol{\Sigma}^{-1}\mathbf{c}_1 \\ \mathbf{0}\end{bmatrix}.\end{aligned} \tag{9.150}$$

We may also express $\hat{\mathbf{w}}$ in the equivalent form

$$\begin{aligned}\hat{\mathbf{w}} &= \mathbf{V}\begin{bmatrix}\boldsymbol{\Sigma}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{c}_1 \\ \mathbf{c}_2\end{bmatrix} \\ &= \mathbf{V}\begin{bmatrix}\boldsymbol{\Sigma}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{U}^{\mathrm{H}}\mathbf{d} \\ &= \mathbf{A}^{+}\mathbf{d}.\end{aligned}$$

The last line is identical to Eq. (9.143), where the pseudoinverse $\mathbf{A}^{+}$ is defined by Eq. (9.132). In effect, we have shown that this value of $\hat{\mathbf{w}}$ does indeed solve the linear least-squares problem.

Moreover, the vector $\hat{\mathbf{w}}$ so defined is *unique*, in that it has the minimum Euclidean norm possible. In particular, since $\mathbf{V}\mathbf{V}^{\mathrm{H}} = \mathbf{I}$, we find from Eq. (9.150) that the squared Euclidean norm of $\hat{\mathbf{w}}$ equals

$$\|\hat{\mathbf{w}}\|^2 = \|\boldsymbol{\Sigma}^{-1}\mathbf{c}_1\|^2.$$

Consider now another possible solution to the linear least-squares problem—one that is defined by modifying Eq. (9.150) to write

$$\mathbf{w}' = \mathbf{V}\begin{bmatrix} \boldsymbol{\Sigma}^{-1}\mathbf{c}_1 \\ \mathbf{b}_2 \end{bmatrix}, \qquad \mathbf{b}_2 \neq \mathbf{0}.$$

The squared Euclidean norm of $\mathbf{w}'$ equals

$$\|\mathbf{w}'\|^2 = \|\boldsymbol{\Sigma}^{-1}\mathbf{c}_1\|^2 + \|\mathbf{b}_2\|^2.$$

We see, therefore, that for any $\mathbf{b}_2 \neq \mathbf{0}$,

$$\|\hat{\mathbf{w}}\| < \|\mathbf{w}'\|. \tag{9.151}$$

In sum, for an FIR filter, the tap-weight vector $\hat{\mathbf{w}}$ defined by Eq. (9.143) is a unique solution to the linear least-squares problem, even when $\text{null}(\mathbf{A}) \neq \varnothing$. *The vector* $\hat{\mathbf{w}}$ *is unique in the sense that it is the only tap-weight vector that simultaneously satisfies two requirements*: (1) *It produces the minimum sum of error squares, and* (2) *it has the smallest Euclidean norm possible.* We may therefore make the following statement:

> The special value of the tap-weight vector $\hat{\mathbf{w}}$ defined in Eq. (9.143) is called the *minimum-norm solution*.

Another Formulation of the Minimum-Norm Solution

We may develop an expanded formulation of the minimum-norm solution, depending on whether we are dealing with the overdetermined or underdetermined case. Accordingly, let us consider these two cases in turn.

Case 1: Overdetermined. For this case, the number of equations K is greater than the number of unknown parameters M. To proceed, then, we substitute Eq. (9.132) into Eq. (9.143); incorporate the partitioned forms of the unitary matrices $\mathbf{V}$ and $\mathbf{U}$, respectively, given in Eqs. (9.106) and (9.114); and then use Eq. (9.112) for $\mathbf{U}_1$. We may thus write

$$\begin{aligned} \hat{\mathbf{w}} &= (\mathbf{V}_1\boldsymbol{\Sigma}^{-1})(\mathbf{A}\mathbf{V}_1\boldsymbol{\Sigma}^{-1})^{\mathrm{H}}\mathbf{d} \\ &= \mathbf{V}_1\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}^{-1}\mathbf{V}_1^{\mathrm{H}}\mathbf{A}^{\mathrm{H}}\mathbf{d} \\ &= \mathbf{V}_1\boldsymbol{\Sigma}^{-2}\mathbf{V}_1^{\mathrm{H}}\mathbf{A}^{\mathrm{H}}\mathbf{d}. \end{aligned} \tag{9.152}$$

Hence, using the definition [see Eq. (9.107)]

$$\mathbf{V}_1 = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_W]$$

in Eq. (9.152), we get the following expanded formulation for $\hat{\mathbf{w}}$ for the overdetermined case:

$$\hat{\mathbf{w}} = \sum_{i=1}^{W} \frac{(\mathbf{v}_i^{\mathrm{H}}\mathbf{A}^{\mathrm{H}}\mathbf{d})}{\sigma_i^2}\mathbf{v}_i. \tag{9.153}$$

Case 2: Underdetermined. For this second case, the number of equations K is smaller than the number of unknowns M. Whereas the estimate $\hat{\mathbf{w}}$ for the overdetermined case is defined in terms of the $\mathbf{v}_i$ in Eq. (9.153), in the underdetermined case it is defined in terms of the $\mathbf{u}_i$. To this end, we look to Eq. (9.135) and the follow-up expression $\mathbf{U}_1\boldsymbol{\Sigma}^{-2}\mathbf{U}_1^{\mathrm{H}}$ as the equal to the inverse matrix $(\mathbf{A}\mathbf{A}^{\mathrm{H}})^{-1}$ for guidance. Specifically, using this background knowledge, we may redefine Eq. (9.143) as the expression for the estimate

$$\hat{\mathbf{w}} = \mathbf{A}^{\mathrm{H}}\mathbf{U}_1\boldsymbol{\Sigma}^{-2}\mathbf{U}_1^{\mathrm{H}}\mathbf{d}. \tag{9.154}$$

Substituting the definition [see Eq. (9.118)]

$$\mathbf{U}_1 = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_W]$$

into Eq. (9.154), we get the following expanded formulation for $\hat{\mathbf{w}}$ for the underdetermined case:

$$\hat{\mathbf{w}} = \sum_{i=1}^{W} \frac{(\mathbf{u}_i^{\mathrm{H}}\mathbf{d})}{\sigma_i^2}\mathbf{A}^{\mathrm{H}}\mathbf{u}_i. \tag{9.155}$$

Plainly, this formulation is different from that of Eq. (9.153) for the overdetermined case.

The important point to note here is that the expanded solutions of $\hat{\mathbf{w}}$ given in Eqs. (9.153) and (9.155) for the overdetermined and underdetermined systems, respectively, are both contained in the compact formula of Eq. (9.143). Indeed, from a numerical computation point of view, the use of Eq. (9.143) is the preferred method for computing the least-squares estimate $\hat{\mathbf{w}}$, be it for the overdetermined or underdetermined case.

9.15 NORMALIZED LMS ALGORITHM VIEWED AS THE MINIMUM-NORM SOLUTION TO AN UNDERDETERMINED LEAST-SQUARES ESTIMATION PROBLEM

In Chapter 7, we derived the normalized least-mean-square (LMS) algorithm as the solution to a constrained minimization problem. In this section, we revisit this algorithm in light of the theory developed on SVD. In particular, we show that the normalized LMS algorithm is indeed the minimum-norm solution to an underdetermined linear least-squares problem involving a single error equation with M unknowns, where M is the dimension of the tap-weight vector in the algorithm.

To be specific, consider the a posteriori error (see Section 5.2 on the LMS algorithm)

$$r(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n+1)\mathbf{u}(n), \tag{9.156}$$

where $d(n)$ is a desired response and $\mathbf{u}(n)$ is a tap-input vector, both measured at time n. The requirement is to find the tap-weight vector $\hat{\mathbf{w}}(n+1)$, measured at time $n+1$, such that the change in the tap-weight vector given by

$$\delta\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n) \tag{9.157}$$

is minimized, subject to the constraint

$$r(n) = 0. \tag{9.158}$$

Using Eq. (9.157) in Eq. (9.156), we may reformulate the a posteriori error as

$$r(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n) - \delta\hat{\mathbf{w}}^{\mathrm{H}}(n+1)\mathbf{u}(n). \tag{9.159}$$

We now recognize the customary definition of the estimation error, namely,

$$e(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n). \tag{9.160}$$

Hence, we may simplify Eq. (9.159) to

$$r(n) = e(n) - \delta\hat{\mathbf{w}}^{\mathrm{H}}(n+1)\mathbf{u}(n). \tag{9.161}$$

Taking the complex conjugation of both sides of Eq. (9.161), we note that the constraint of Eq. (9.158) is equivalent to setting

$$\mathbf{u}^{\mathrm{H}}(n)\delta\hat{\mathbf{w}}(n+1) = e^*(n). \tag{9.162}$$

Accordingly, we may restate our constrained minimization problem as follows:

> Find the minimum-norm solution for the change $\delta\hat{\mathbf{w}}(n+1)$ in the tap-weight estimate at time $n+1$ that satisfies the constraint
>
> $$\mathbf{u}^{\mathrm{H}}(n)\delta\hat{\mathbf{w}}(n+1) = e^*(n).$$

This problem is an underdetermined linear least-squares estimation problem. To solve it, we may use SVD, as described in Eq. (9.155). To help us in the application of this method, we employ Eq. (9.162) to make the identifications listed in Table 9.1 between the normalized LMS algorithm and linear least-squares estimation. In particular, we note that the normalized LMS algorithm has only one nonzero singular value equal to the norm of the tap-input vector $\mathbf{u}(n)$; that is, the rank $W = 1$. The corresponding left-singular vector is therefore simply equal to unity. Hence, with the aid of the table, the application of Eq. (9.155) yields

$$\delta\hat{\mathbf{w}}(n+1) = \frac{1}{\|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n). \tag{9.163}$$

This is precisely the result that we derived previously in Eq. (7.8) of Chapter 7.

We may next follow a reasoning similar to that described in deriving Eq. (7.8) and redefine the change $\delta\hat{\mathbf{w}}(n+1)$ by introducing a scaling factor $\tilde{\mu}$, as shown by

$$\delta\hat{\mathbf{w}}(n+1) = \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n).$$

TABLE 9.1 Summary of Correspondences Between Linear Least-Squares Estimation and Normalized LMS Algorithm

	Linear least-squares estimation (underdetermined)	Normalized LMS algorithm
Data matrix	$\mathbf{A}$	$\mathbf{u}^{\mathrm{H}}(n)$
Desired data vector	$\mathbf{d}$	$e^*(n)$
Parameter vector	$\hat{\mathbf{w}}$	$\delta\hat{\mathbf{w}}(n+1)$
Rank	W	1
Singular value	$\sigma_i, i = 1, \ldots, W$	$\|\mathbf{u}(n)\|$
Left singular vector	$\mathbf{u}_i, i = 1, \ldots, W$	1

Equivalently, we may write

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n). \tag{9.164}$$

By so doing, we are able to exercise control over the change in the tap-weight vector from one adaptation cycle to the next without changing the direction of the vector. Equation (9.164) is the tap-weight vector update for the normalized LMS algorithm, as presented in Eq. (7.10).

The important point to note from the discussion presented in this section is that the SVD provides an insightful link between the underdetermined form of linear least-squares estimation and LMS theory. In particular, we have shown that the weight update in the normalized LMS algorithm may indeed be viewed as the minimum-norm solution to an underdetermined form of the linear least-squares problem. The problem involves a single error equation with a number of unknowns equal to the dimension of the tap-weight vector in the algorithm.

9.16 SUMMARY AND DISCUSSION

In this chapter, we presented a detailed discussion of the method of least squares for solving the linear adaptive filtering problem through the use of a batch (block) processing approach. Among the distinguishing features of this approach are the following:

- It is a *model-dependent* procedure that operates on the input data on a block-by-block basis; the model on which the approach is based is a multiple linear regression model.
- It yields a solution for the tap-weight vector of an FIR filter that is the best linear unbiased estimate (BLUE), assuming that the measurement error process in the multiple linear regression model is white with zero mean.

The method of least squares is well suited for solving spectrum estimation/beamforming problems, such as those based on autoregressive (AR) and minimum-variance distortionless response (MVDR) models. For the efficient computation of the least-squares solution, the recommended procedure is to use singular value decomposition (SVD) that operates on the input data directly. The SVD is defined by the following parameters:

- A set of left singular vectors that form a unitary matrix.
- A set of right singular vectors that form another unitary matrix.
- A corresponding set of nonzero singular values.

The important advantage of using the SVD to solve a linear least-squares problem is that the solution, defined in terms of the pseudoinverse of the input data matrix, is numerically stable. We may therefore make the following statement:

> An algorithm is said to be *numerically stable*, or *robust*, if it does not introduce any more sensitivity to perturbation than that which is inherently present in the problem being studied.

Another useful application of the SVD is in *rank determination*. The *column rank* of a matrix is defined by the number of linearly independent columns of the matrix. Specifically, we say that an M-by-K matrix, with $M \geq K$, has *full* column rank if and only if it has K independent columns. In theory, the issue of full rank determination is a yes–no type of proposition, in the sense that either the matrix in question has full rank or it does not. In practice, however, the fuzzy nature of a data matrix and the use of inexact (finite-precision) arithmetic complicate the rank determination problem. The SVD provides a practical method for determining the rank of a matrix, given fuzzy data and round-off errors due to finite-precision computations.

We may conclude the chapter as follows:

> The method of least squares via the SVD provides an insightful mathematical link to the normalized LMS algorithm, in that that filter is the minimum-norm solution to an underdetermined least-squares estimation problem.

PROBLEMS

1. Consider a linear array consisting of M uniformly spaced sensors. The output of sensor k observed at time i is denoted by $u(k, i)$, where $k = 1, 2, \ldots, M$ and $i = 1, 2, \ldots, n$. In effect, the observations $u(1, i), u(2, i), \ldots, u(M, i)$ define "snapshot" i. Let $\mathbf{A}$ denote the n-by-M data matrix whose Hermitian transpose is defined by

$$\mathbf{A}^{\mathrm{H}} = \begin{bmatrix} u(1,1) & u(1,2) & \cdots & u(1,n) \\ u(2,1) & u(2,2) & \cdots & u(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ u(M,1) & u(M,2) & \cdots & u(M,n) \end{bmatrix},$$

where the number of columns equals the number of snapshots and the number of rows equals the number of sensors in the array. Demonstrate the following interpretations:

(a) The M-by-M matrix $\mathbf{A}^{\mathrm{H}}\mathbf{A}$ is the *spatial* correlation matrix with temporal averaging. This form of averaging assumes that the environment is temporally stationary.

(b) The n-by-n matrix $\mathbf{A}\mathbf{A}^{\mathrm{H}}$ is the *temporal* correlation matrix with spatial averaging. This form of averaging assumes that the environment is spatially stationary.

2. Estimate the desired data vector and the error vector for a linear least-squares filter with two taps (i.e., $M = 2$) and a real-valued input time series consisting of four samples (i.e., $N = 4$). The input data matrix $\mathbf{A}$ and the desired data vector $\mathbf{d}$ have the following values:

$$\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 2 & 1 \\ -1 & 2 \end{bmatrix} \qquad \mathbf{d} = \begin{bmatrix} 2 \\ 1 \\ 1/34 \end{bmatrix}$$

3. Problem 2 employed a 3-by-2 input data matrix and a 3-by-1 desired data vector to illustrate the corollary to the principle of orthogonality. Use the data given in that example to calculate the two tap weights of the linear least-squares filter.

4. In the autocorrelation method of linear prediction, we choose the tap-weight vector of a finite-duration impulse response (FIR) predictor to minimize the error energy

$$\mathscr{E}_f = \sum_{n=1}^{\infty} |f(n)|^2,$$

where $f(n)$ is the prediction error. Show that the transfer function $H(z)$ of the (forward) prediction-error filter is minimum phase, in that its roots must lie strictly inside the unit circle in the z-plane.

[*Hints*: **(1)** Express the transfer function $H(z)$ of order M (say) as the product of a simple zero factor $(1 - z_i z^{-1})$ and a function $H'(z)$. Then minimize the prediction-error energy with respect to the magnitude of zero z_i.

(2) Use the Cauchy–Schwartz inequality

$$\mathrm{Re}\left[\sum_{n=1}^{\infty} e^{j\theta} g(n-1) g^*(n)\right] \le \left[\sum_{n=1}^{\infty} |g(n)|^2\right]^{1/2} \left[\sum_{n=1}^{\infty} |e^{j\theta} g(n-1)|^2\right]^{1/2},$$

which holds if and only if $g(n) = e^{j\theta} g(n-1)$ for $n = 1, 2, \ldots, \infty$.]

5. Figure P9.1 shows a *forward prediction-error filter* with an FIR structure, with the tap inputs $u(i-1), u(i-2), \ldots, u(i-M)$ used to make a linear prediction of $u(i)$, denoted by $\hat{u}(i)$. The problem is to find the tap-weight vector $\hat{\mathbf{w}}$ that minimizes the sum of forward prediction-error squares

$$\mathscr{E}_f = \sum_{i=M+1}^{N} |f_M(i)|^2,$$

where $f_M(i)$ is the forward prediction error. Find the following parameters:

(a) The M-by-M correlation matrix of the tap inputs of the predictor.

(b) The M-by-1 cross-correlation vector between the tap inputs of the predictor and the desired response $u(i)$.

(c) The minimum value of $\mathscr{E}_f$.

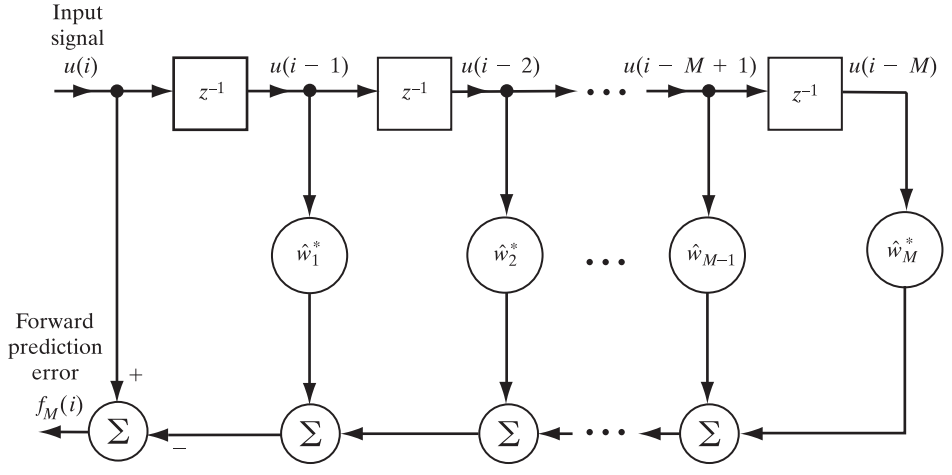


FIGURE P9.1

6. Figure P9.2 shows a *backward prediction-error filter* with an FIR structure, with the tap inputs $u(i-M+1), \ldots, u(i-1), u(i)$ used to make a linear prediction of the input $u(i-M)$,

denoted by $\hat{u}(i - M)$. The problem is to find the tap-weight vector $\hat{\mathbf{w}}$ that minimizes the sum of backward prediction-error squares

$$\mathscr{E}_b = \sum_{i=M+1}^{N} |b_M(i)|^2,$$

where $b_M(i)$ is the backward prediction error. Find the following parameters:

(a) The M-by-M correlation matrix of the tap inputs.

(b) The M-by-1 correlation vector between the tap inputs and the desired response $u(i-M)$.

(c) The minimum value of $\mathscr{E}_b$.

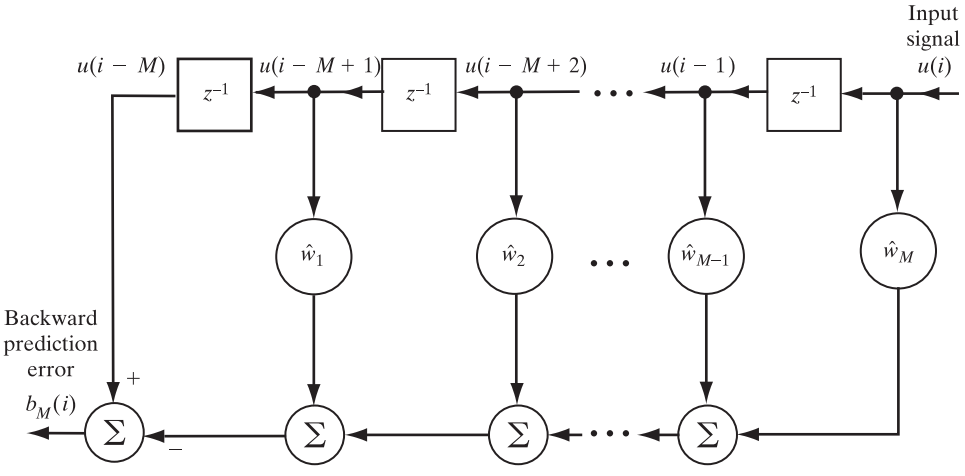


FIGURE P9.2

7. Demonstrate how the autocorrelation method is the only one that yields a Toeplitz correlation matrix for the input data.

8. A simplified form of regularization applied to the MVDR beamformer is to modify the formula of Eq. (9.92) for computation of the M-by-M time-average correlation matrix $\mathbf{\Phi}$ into

$$\mathbf{\Phi} = \sum_{n=1}^{K} \mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n) + \delta\mathbf{I},$$

where $\mathbf{I}$ is the M-by-M identity matrix and δ is the regularization parameter. For obvious reasons, this form of regularization is referred to as *diagonal loading*.

Show that an MVDR beamformer regularized in this manner is derived by minimizing the cost function

$$\mathscr{E}_{\mathrm{reg}} = \sum_{n=1}^{K} |\mathbf{w}^{\mathrm{H}}\mathbf{u}(n)|^2 + \lambda(\mathbf{w}^{\mathrm{H}}\mathbf{s}(\theta) - 1) + \delta\|\mathbf{w}\|^2$$

with respect to the weight vector $\mathbf{w}$. Following the notation described in Section 9.10, $\mathbf{s}(\theta)$ is the beam-steering vector, λ is the Lagrange multiplier, and δ is the regularization parameter.

9. Consider an autoregressive spectrum estimation procedure that relies on the combined use of *forward and backward linear prediction* (FBLP). Figures P9.1 and P9.2 show the operation of FIR filters as forward and backward prediction-error filters, respectively. The cost function to be minimized is defined by

$$\mathscr{E} = \sum_{i=M+1}^{N} (|f_M(i)|^2 + |b_M(i)|^2),$$

where $f_M(i)$ and $b_M(i)$ are the forward and backward prediction errors, respectively.

(a) Derive the formula for the tap-weight vector $\hat{\mathbf{w}}$ that minimizes the cost function $\mathscr{E}$.

(b) Determine the minimum value of the cost function, denoted by $\mathscr{E}_{\min}$.

(c) Define the $(M+1)$-by-1 tap-weight vector of the prediction-error filter

$$\hat{\mathbf{a}} = \begin{bmatrix} 1 \\ -\hat{\mathbf{w}} \end{bmatrix}.$$

Suppose we write

$$\mathbf{\Phi}\hat{\mathbf{a}} = \begin{bmatrix} \mathscr{E}_{\min} \\ \mathbf{0} \end{bmatrix},$$

where $\mathbf{0}$ is the null vector. Determine the augmented correlation matrix $\mathbf{\Phi}$, and show that $\mathbf{\Phi}$ is *Hermitian persymmetric*; that is, show that

$$\phi(k, t) = \phi^*(t, k), \qquad 0 \leq (t, k) \leq M$$

and

$$\phi(M - k, M - t) = \phi^*(t, k), \qquad 0 \leq (t, k) \leq M.$$

10. Consider the generic signal-to-interference-ratio maximization problem[6]

$$\max_{w}\left(\frac{\mathbf{w}^{\mathrm{H}}\mathbf{s}\mathbf{s}^{\mathrm{H}}\mathbf{w}}{\mathbf{w}^{\mathrm{H}}\mathbf{R}\mathbf{w}}\right),$$

subject to the auxiliary linear constraints

$$\mathbf{C}_{N-1}^{\mathrm{H}}\mathbf{w} = \mathbf{f}_{N-1},$$

[6]The constrained signal-to-interference maximization problem described in Problem 10 goes back to an early paper by Frost (1972). Abramovich (2000) studies the probability distributions for the "loss" factor produced by using a maximum-likelihood estimate of the correlation matrix $\mathbf{R}$ generated by K training samples of the input vector $\mathbf{u}(n)$, namely,

$$\hat{\mathbf{R}} = \frac{1}{K}\sum_{n=1}^{K}\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n),$$

or by using the diagonally "loaded" estimate

$$\hat{\mathbf{R}}_L = \hat{\mathbf{R}} + \delta\mathbf{I},$$

where $\mathbf{I}$ is the identity matrix and δ is the loading factor (regularization parameter). The statistical analysis presented by Abramovich is of particular interest to the performance analysis of linearly constrained sample matrix inversion algorithms used in airborne and over-the-horizon radar applications with a limited size of training samples.

where $\mathbf{w}$ is an M-by-1 weight vector of an adaptive beamformer, $\mathbf{s}$ is an M-by-1 complex-valued beam-steering vector, $\mathbf{R}$ is the M-by-M unknown correlation matrix of the interference, $\mathbf{C}_{N-1}$ is the M-by-$(N-1)$ matrix of linear constraints, and $\mathbf{f}_{N-1}$ is the $(N-1)$-by-1 constraint vector.

(a) Show that the solution to this constrained optimization problem is given by

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{C}(\mathbf{C}^{\mathrm{H}}\mathbf{R}^{-1}\mathbf{C})^{-1}\mathbf{f},$$

where $\mathbf{C}$ is the M-by-N matrix obtained by augmenting the matrix $\mathbf{C}_{N-1}$ with the steering vector $\mathbf{s}$ and the vector $\mathbf{f}$ is formed by augmenting the vector $\mathbf{f}_{N-1}$ with the fixed value f_0 of the gain (radiation pattern) in the direction of the steering vector $\mathbf{s}$; that is,

$$\mathbf{w}^{\mathrm{H}}\mathbf{s} = f_0.$$

(b) Find the maximum signal-to-interference ratio that results from the use of the optimum solution defined in part (a).

(c) What is the optimum weight vector when the auxiliary linear constraints are zero? What is the corresponding value of the maximum signal-to-interference ratio?

(d) Describe a procedure for estimating the unknown correlation matrix $\mathbf{R}$.

11. Calculate the singular values and singular vectors of

$$\mathbf{A} = \begin{bmatrix} 2 & -1 \\ 0.5 & 1 \end{bmatrix}$$

by Eigen decomposition of the matrix product $\mathbf{A}^{\mathrm{T}}\mathbf{A}$

12. What are the important advantages of using singular-value decomposition?

13. Show that when the measurement error process $e_{\mathrm{o}}(i)$ is a zero-mean Gaussian white-noise process, the least-squares estimate $\hat{\mathbf{w}}$ is a minimum-variance unbiased estimate.

14. In this problem, using the idea of singular-value decomposition, we explore the derivation of the weight update for the normalized LMS algorithm described in Eq. (7.12). The problem may be viewed as an extension of the discussion presented in Section 9.15. Find the minimum-norm solution for the coefficient vector

$$\mathbf{c}(n+1) = \begin{bmatrix} \delta\hat{\mathbf{w}}(n+1) \\ 0 \end{bmatrix}$$

that satisfies the equation

$$\mathbf{x}^{\mathrm{H}}(n)\mathbf{c}(n+1) = e^*(n),$$

where

$$\mathbf{x}(n) = \begin{bmatrix} \mathbf{u}(n) \\ \sqrt{\delta} \end{bmatrix}.$$

Hence, show that

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\delta + \|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n),$$

where $\delta > 0$, and $\tilde{\mu}$ is the step-size parameter.

15. Give the solution of the Wiener-Hopf equations in matrix form.

CHAPTER 10

The Recursive Least-Squares (RLS) Algorithm

In this chapter, we extend the use of the method of least squares to develop a *recursive* algorithm for the design of adaptive finite-duration impulse response (FIR) filters such that, given the least-squares estimate of the tap-weight vector of the filter at adaptation cycle $n-1$, we may compute the updated estimate of the vector at adaptation cycle n upon the arrival of new data. We refer to the resulting algorithm as the *recursive least-squares (RLS) algorithm*.

We begin the development of the RLS algorithm by reviewing some basic relations that pertain to the method of least squares. Then, by exploiting a relation in matrix algebra known as the *matrix inversion lemma*, we develop the RLS algorithm. An important feature of this algorithm is that its rate of convergence is typically an order of magnitude faster than that of the simple LMS algorithm, due to the fact that the RLS algorithm *whitens* the input data by using the inverse correlation matrix of the data, assumed to be of zero mean. This improvement in performance, however, is achieved at the expense of an increase in computational complexity of the RLS algorithm.

10.1 SOME PRELIMINARIES

In *recursive* implementations of the method of least squares, we start the computation with *prescribed initial conditions* and use the information contained in new data samples to *update* the old estimates. We therefore find that the length of observable data is variable. Accordingly, we express the *cost function* to be minimized as $\mathscr{E}(n)$, where n is the variable length of the observable data. Also, it is customary to introduce a *weighting factor* into the definition of $\mathscr{E}(n)$. We thus write

$$\mathscr{E}(n) = \sum_{i=1}^{n} \beta(n,i)\,|e(i)|^2, \tag{10.1}$$

where $e(i)$ is the difference between the *desired response* $d(i)$ and the *output* $y(i)$ produced by an FIR filter whose tap inputs (at time i) equal $u(i), u(i-1), \ldots, u(i-M+1)$, as in Fig. 10.1. That is,

$$\begin{aligned} e(i) &= d(i) - y(i) \\ &= d(i) - \mathbf{w}^{\mathrm{H}}(n)\mathbf{u}(i), \end{aligned} \tag{10.2}$$

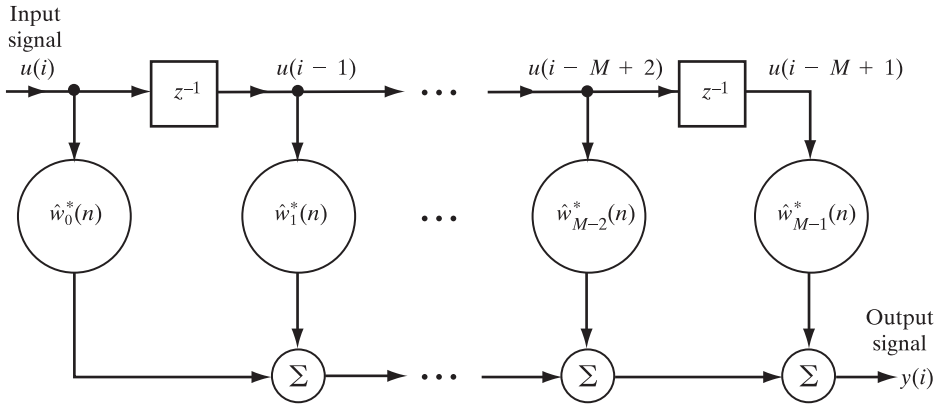


FIGURE 10.1 FIR filter with time-varying tap weights.

where $\mathbf{u}(i)$ is the *tap-input vector at time i*, defined by

$$\mathbf{u}(i) = [u(i), u(i-1), \ldots, u(i-M+1)]^{\mathrm{T}}, \tag{10.3}$$

and $\mathbf{w}(n)$ is the *tap-weight vector at time n*, defined by

$$\mathbf{w}(n) = [w_0(n), w_1(n), \ldots, w_{M-1}(n)]^{\mathrm{T}}. \tag{10.4}$$

The superscript T denotes transposition, and the superscript H denotes *Hermitian transposition* (i.e., the operation of transposition combined with complex conjugation). Note that the tap weights of the FIR filter remain *fixed* during the observation interval $1 \le i \le n$ for which the cost function $\mathscr{E}(n)$ is defined.

The weighting factor $\beta(n, i)$ in Eq. (10.1) has the property that

$$0 < \beta(n, i) \le 1, \qquad i = 1, 2, \ldots, n. \tag{10.5}$$

The use of the weighting factor $\beta(n, i)$, in general, is intended to ensure that data in the distant past are "forgotten" in order to afford the possibility of following the statistical variations of the observable data. A special form of weighting that is commonly used is the *exponential weighting factor*, or *forgetting factor*, defined by

$$\beta(n, i) = \lambda^{n-i}, \qquad i = 1, 2, \ldots, n, \tag{10.6}$$

where λ is a positive constant close to, but less than, unity. When $\lambda = 1$, we have the ordinary method of least squares. The inverse of $1 - \lambda$ is, roughly speaking, a measure of the *memory* of the algorithm. The special case $\lambda = 1$ corresponds to *infinite memory*.

Regularization

Least-square estimation, like the method of least squares, is an ill-posed inverse problem, in that we are given input data consisting of a tap-input vector $\mathbf{u}(n)$ and the corresponding desired response $d(n)$ for varying n, and the requirement is to

estimate the unknown parameter vector of a multiple linear regression model that relates $d(n)$ to $\mathbf{u}(n)$.

The ill-posed nature of least-squares estimation is due to the following reasons:

- There is insufficient information in the input data to reconstruct the input–output mapping uniquely.
- The unavoidable presence of noise or imprecision in the input data adds uncertainty to the reconstructed input–output mapping.

To make the estimation problem "well posed," some form of prior information about the input–output mapping is needed. This, in turn, means that formulation of the cost function must be expanded to take the prior information into account.

To satisfy that objective, we expand the cost function to be minimized as the sum of two components:

$$\mathscr{E}(n) = \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2 + \delta \lambda^n \|\mathbf{w}(n)\|^2. \tag{10.7}$$

Here, we assume the use of prewindowing, which means that the input signal is windowed before application of the FIR filter. The two components of the cost function are as follows:

1. The *sum of weighted error squares*,

$$\sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2 = \sum_{i=1}^{n} \lambda^{n-i} |d(i) - \mathbf{w}^{\mathrm{H}}(n)\mathbf{u}(i)|^2,$$

which is data dependent. This component measures the exponentially weighted error between the desired response $d(i)$ and the actual response of the filter, $y(i)$, which is related to the tap-input vector $\mathbf{u}(i)$ by the formula

$$y(i) = \mathbf{w}^{\mathrm{H}}(n)\mathbf{u}(i).$$

2. The *regularizing term*,

$$\delta \lambda^n \|\mathbf{w}(n)\|^2 = \delta \lambda^n \mathbf{w}^{\mathrm{H}}(n)\mathbf{w}(n),$$

where δ is a positive real number called the *regularization parameter*. Except for the factor $\delta \lambda^n$, the regularizing term depends solely on the tap-weight vector $\mathbf{w}(n)$. The regularizing term is included in the cost function to stabilize the solution to the recursive least-squares problem by smoothing the solution.

In a strict sense, the term $\delta \lambda^n \|\mathbf{w}(n)\|^2$ is a "rough" form of regularization, for two reasons. First, the exponential weighting factor λ lies in the interval $0 < \lambda \leq 1$; hence, for λ less than unity, λ^n tends to zero for large n, which means that the beneficial effect of adding $\delta \lambda^n \|\hat{\mathbf{w}}(n)\|^2$ to the cost function is forgotten with time. Second, and more important, the regularizing term should be of the form $\delta \|\mathbf{D}F(\hat{\mathbf{w}})\|^2$, where $F(\hat{\mathbf{w}})$ is the input–output

map realized by the RLS algorithm and $\mathbf{D}$ is the *differential operator*.[1] Nevertheless, the regularizing term in Eq. (10.7) is commonly used in formulating the RLS algorithm.

Reformulation of the Normal Equations

Expanding Eq. (10.7) and collecting terms, we find that the effect of including the regularizing term $\delta\lambda^n\|\mathbf{w}(n)\|^2$ in the cost function $\mathscr{E}(n)$ is equivalent to a reformulation of the M-by-M time-average correlation matrix of the tap-input vector $\mathbf{u}(i)$:

$$\boldsymbol{\Phi}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)\mathbf{u}^{\mathrm{H}}(i) + \delta\lambda^n\mathbf{I}. \tag{10.8}$$

In this equation, $\mathbf{I}$ is the M-by-M identity matrix. Note that the addition of the regularizing term also has the effect of making the correlation matrix $\Phi(n)$ nonsingular at all stages of the computation, starting from $n = 0$. A correlation matrix modified as in Eq. (10.8) is said to be *diagonally loaded.*

The M-by-1 time-average cross-correlation vector $\mathbf{z}(n)$ between the tap inputs of the FIR filter and the desired response is unaffected by the use of regularization, as is shown by the formula

$$\mathbf{z}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)d^*(i), \tag{10.9}$$

where, again, the use of prewindowing is assumed and the asterisk denotes complex conjugation.

According to the method of least squares discussed in Chapter 9, the optimum value of the M-by-1 tap-weight vector $\hat{\mathbf{w}}(n)$, for which the cost function $\mathscr{E}(n)$ of Eq. (10.7) attains its minimum value, is defined by the normal equations. For the recursive least-squares problem, the normal equations are written in matrix form as

$$\boldsymbol{\Phi}(n)\hat{\mathbf{w}}(n) = \mathbf{z}(n), \tag{10.10}$$

where $\boldsymbol{\Phi}(n)$ and $\mathbf{z}(n)$ are now defined by Eqs. (10.8) and (10.9), respectively.

Recursive Computations of $\Phi(n)$ and $z(n)$

Isolating the term corresponding to $i = n$ from the rest of the summation on the right-hand side of Eq. (10.8), we may write

$$\boldsymbol{\Phi}(n) = \lambda\left[\sum_{i=1}^{n-1} \lambda^{n-1-i}\mathbf{u}(i)\mathbf{u}^{\mathrm{H}}(i) + \delta\lambda^{n-1}\mathbf{I}\right] + \mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n). \tag{10.11}$$

[1]Regularization theory is credited to Tikhonov (1963). For a detailed discussion of the topic, see Tikhonov and Arsenin (1977), Kirsch (1996), and Haykin (2009).

The analytic approach used to handle the proper regularizing term $\delta\|\mathbf{D}F(\hat{\mathbf{w}})\|^2$ builds on the idea of a *function space*, which refers to a *normed space* of functions. In such a space of many (strictly speaking, infinitely many) dimensions, a continuous function is represented by a "vector." By using this geometric image, an insightful link is established between linear differential operators and matrices (Lanczos, 1964). Thus, the symbol $\|\cdot\|$ denotes a norm imposed on the function space to which $\mathbf{D}F(\hat{\mathbf{w}})$ belongs.

By definition, the expression inside the brackets on the right-hand side of Eq. (10.11) equals the correlation matrix $\mathbf{\Phi}(n-1)$. Hence, we have the following recursion for updating the value of the correlation matrix of the tap inputs:

$$\mathbf{\Phi}(n) = \lambda\mathbf{\Phi}(n-1) + \mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n). \tag{10.12}$$

Here, $\mathbf{\Phi}(n-1)$ is the "old" value of the correlation matrix, and the matrix product $\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)$ plays the role of a "correction" term in the updating operation. Note that the recursion of Eq. (10.12) holds, irrespective of the initializing condition.

Similarly, we may use Eq. (10.9) to derive the following recursion for updating the cross-correlation vector between the tap inputs and the desired response:

$$\mathbf{z}(n) = \lambda\mathbf{z}(n-1) + \mathbf{u}(n)d^*(n). \tag{10.13}$$

To compute the least-square estimate $\hat{\mathbf{w}}(n)$ for the tap-weight vector in accordance with Eq. (10.8), we have to determine the inverse of the correlation matrix $\mathbf{\Phi}(n)$. In practice, however, we usually try to avoid performing such an operation, as it can be quite time consuming, particularly if the number of tap weights, M, is high. Also, we would like to be able to compute the least-squares estimate $\hat{\mathbf{w}}(n)$ for the tap-weight vector recursively for $n = 1, 2, \ldots, \infty$. We can realize both of these objectives by using a basic result in matrix algebra known as the *matrix inversion lemma*, which we discuss next.

10.2 THE MATRIX INVERSION LEMMA

Let $\mathbf{A}$ and $\mathbf{B}$ be two positive-definite M-by-M matrices related by

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^{\mathrm{H}}, \tag{10.14}$$

where $\mathbf{D}$ is a positive-definite N-by-M matrix and $\mathbf{C}$ is an M-by-N matrix. According to the *matrix inversion lemma*, we may express the inverse of the matrix $\mathbf{A}$ as

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^{\mathrm{H}}\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^{\mathrm{H}}\mathbf{B}. \tag{10.15}$$

The proof of this lemma is established by multiplying Eq. (10.14) by Eq. (10.15) and recognizing that the product of a square matrix and its inverse is equal to the identity matrix. (See Problem 2.) The matrix inversion lemma states that if we are given a matrix $\mathbf{A}$, as defined in Eq. (10.14), we can determine its inverse $\mathbf{A}^{-1}$ by using the relation expressed in Eq. (10.15). In effect, the lemma is described by that pair of equations. The matrix inversion lemma is also referred to in the literature as *Woodbury's identity*.[2]

In the next section, we show how the matrix inversion lemma can be applied to obtain a recursive equation for computing the least-squares solution $\hat{\mathbf{w}}(n)$ for the tap-weight vector.

[2]The exact origin of the matrix inversion lemma is not known. Householder (1964) attributes it to Woodbury (1950). At any rate, the lemma was first applied in the filtering literature by Kailath, who used a form of it to prove the equivalence of the Wiener filter and the maximum-likelihood procedure for estimating the output of a random, linear, time-invariant channel that is corrupted by additive white Gaussian noise (Kailath, 1960). Early use of the matrix inversion lemma was also made by Ho (1963). Another interesting application of the matrix inversion lemma was made by Brooks and Reed, who used it to prove the equivalence of the Wiener filter, the maximum signal-to-noise ratio filter, and the likelihood ratio processor for detecting a signal amidst additive white Gaussian noise (Brooks & Reed, 1972). (See Problem 18 of Chapter 2.)

10.3 THE EXPONENTIALLY WEIGHTED RLS ALGORITHM

With the correlation matrix $\mathbf{\Phi}(n)$ assumed to be nonsingular and therefore invertible, we may apply the matrix inversion lemma to the recursive equation (10.12). We first make the following identifications:

$$\begin{aligned}\mathbf{A} &= \mathbf{\Phi}(n);\\ \mathbf{B}^{-1} &= \lambda\mathbf{\Phi}(n-1);\\ \mathbf{C} &= \mathbf{u}(n);\\ \mathbf{D} &= 1.\end{aligned}$$

Then, substituting these definitions into the matrix inversion lemma, we obtain the following recursive equation for the inverse of the correlation matrix:

$$\mathbf{\Phi}^{-1}(n) = \lambda^{-1}\mathbf{\Phi}^{-1}(n-1) - \frac{\lambda^{-2}\mathbf{\Phi}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{\Phi}^{-1}(n-1)}{1+\lambda^{-1}\mathbf{u}^{\mathrm{H}}(n)\mathbf{\Phi}^{-1}(n-1)\mathbf{u}(n)}. \tag{10.16}$$

For convenience of computation, let

$$\mathbf{P}(n) = \mathbf{\Phi}^{-1}(n) \tag{10.17}$$

and

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^{\mathrm{H}}\mathbf{P}(n-1)\mathbf{u}(n)}. \tag{10.18}$$

Using these definitions, we may rewrite Eq. (10.16) as

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{P}(n-1). \tag{10.19}$$

The M-by-M matrix $\mathbf{P}(n)$ is referred to as the *inverse correlation matrix*.[3] The M-by-1 vector $\mathbf{k}(n)$ is referred to as the *gain vector*, for reasons that will become apparent later in the section. Equation (10.19) is the *Riccati equation* for the RLS algorithm.

By rearranging Eq. (10.18), we obtain

$$\begin{aligned}\mathbf{k}(n) &= \lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{P}(n-1)\mathbf{u}(n)\\ &= [\lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{P}(n-1)]\mathbf{u}(n).\end{aligned} \tag{10.20}$$

We see from Eq. (10.19) that the expression inside the brackets on the right-hand side of the last line of Eq. (10.20) equals $\mathbf{P}(n)$. Hence, we may simplify Eq. (10.20) to

$$\mathbf{k}(n) = \mathbf{P}(n)\mathbf{u}(n). \tag{10.21}$$

This result, together with $\mathbf{P}(n) = \mathbf{\Phi}^{-1}(n)$, may be used as the definition of the gain vector:

$$\mathbf{k}(n) = \mathbf{\Phi}^{-1}(n)\mathbf{u}(n). \tag{10.22}$$

[3]The matrix $\mathbf{P}(n)$ may also be viewed as the covariance matrix of the RLS estimate $\hat{\mathbf{w}}(n)$, normalized with respect to the noise variance σ^2 in the multiple linear regression model of Fig. 9.1. This interpretation of $\mathbf{P}(n)$ follows from Property 2 of linear least-squares estimates described in Section 9.8 of Chapter 9.

In other words, the gain vector $\mathbf{k}(n)$ is defined as the tap-input vector $\mathbf{u}(n)$, transformed by the inverse of the correlation matrix $\mathbf{\Phi}(n)$.

Time Update for the Tap-Weight Vector

Next, we wish to develop a recursive equation for updating the least-squares estimate $\hat{\mathbf{w}}(n)$ of the tap-weight vector. To do this, we use Eqs. (10.8), (10.13), and (10.17) to express the least-squares estimate of the tap-weight vector at adaptation cycle n as

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \mathbf{\Phi}^{-1}(n)\mathbf{z}(n)\\ &= \mathbf{P}(n)\mathbf{z}(n)\\ &= \lambda\mathbf{P}(n)\mathbf{z}(n-1) + \mathbf{P}(n)\mathbf{u}(n)d^*(n).\end{aligned} \tag{10.23}$$

Substituting Eq. (10.19) for $\mathbf{P}(n)$ in the first term only on the right-hand side of Eq. (10.23), we get

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \mathbf{P}(n-1)\mathbf{z}(n-1) - \mathbf{k}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{P}(n-1)\mathbf{z}(n-1)\\ &\quad + \mathbf{P}(n)\mathbf{u}(n)d^*(n)\\ &= \mathbf{\Phi}^{-1}(n-1)\mathbf{z}(n-1) - \mathbf{k}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{\Phi}^{-1}(n-1)\mathbf{z}(n-1)\\ &\quad + \mathbf{P}(n)\mathbf{u}(n)d^*(n)\\ &= \hat{\mathbf{w}}(n-1) - \mathbf{k}(n)\mathbf{u}^{\mathrm{H}}(n)\hat{\mathbf{w}}(n-1) + \mathbf{P}(n)\mathbf{u}(n)d^*(n).\end{aligned} \tag{10.24}$$

Finally, using the fact that $\mathbf{P}(n)\mathbf{u}(n)$ equals the gain vector $\mathbf{k}(n)$, as in Eq. (10.21), we get the desired recursive equation for updating the tap-weight vector:

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)[d^*(n) - \mathbf{u}^{\mathrm{H}}(n)\hat{\mathbf{w}}(n-1)]\\ &= \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n).\end{aligned} \tag{10.25}$$

Here,

$$\begin{aligned}\xi(n) &= d(n) - \mathbf{u}^{\mathrm{T}}(n)\hat{\mathbf{w}}^*(n-1)\\ &= d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n-1)\mathbf{u}(n)\end{aligned} \tag{10.26}$$

is the a priori *estimation error*. The inner product $\hat{\mathbf{w}}^{\mathrm{H}}(n-1)\mathbf{u}(n)$ represents an estimate of the desired response $d(n)$, based on the *old* least-squares estimate of the tap-weight vector that was made at time $n-1$.

Equation (10.25) for the adjustment of the tap-weight vector and Eq. (10.26) for the a priori estimation error suggest the block-diagram representation depicted in Fig. 10.2(a) for the *recursive least-squares (RLS) algorithm.*

The a priori estimation error $\xi(n)$ is, in general, different from the *a posteriori estimation error*

$$e(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n), \tag{10.27}$$

the computation of which involves the *current* least-squares estimate of the tap-weight vector available at time n. Indeed, we may view $\xi(n)$ as a "tentative" value of $e(n)$ before updating the tap-weight vector. Note, however, that in the least-squares optimization that led to the recursive algorithm of Eq. (10.25), we actually minimized the cost function $\mathscr{E}(n)$ based on $e(n)$ and *not* $\xi(n)$. Note also that the error signal $e(n)$ in the RLS algorithm is defined differently from that in the LMS algorithm.

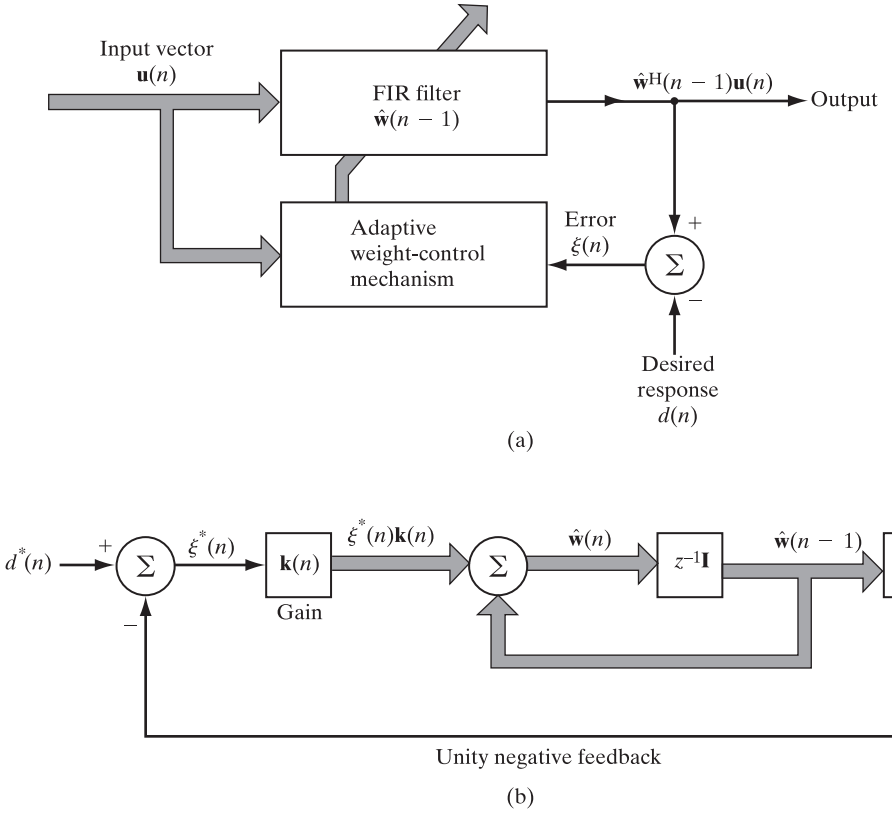


FIGURE 10.2 Representations of the RLS algorithm: (a) block diagram; (b) signal-flow graph.

Signal-Flow Graph

Figure 10.2(b) depicts a signal-flow graph representation of the RLS algorithm. Specifically, Eq. (10.26), representing the *filtering process*, and Eq. (10.25), representing the *adaptation process*, are both accounted for in the figure. However, the figure is *incomplete* because computation of the gain vector, $\mathbf{k}(n)$, is *not* represented in the figure. Examining Eq. (10.21), we readily see that this computation follows a *square law*. In other words, the complete signal-flow graph representation of the RLS algorithm is computationally more demanding than the signal-flow graph representative of the LMS algorithm in Fig. 6.1.

Summary of the RLS Algorithm

Table 10.1 provides a summary of the RLS algorithm, making use of Eqs. (10.18), (10.26), (10.25), and (10.19), in that order.

Examining Eq. (10.18), we see that the product term $\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)$ is common to the numerator and denominator. Computation of the gain vector $\mathbf{k}(n)$ may therefore be simplified in two stages. First, the vector $\boldsymbol{\pi}(n)$ is introduced to denote the common term, $\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)$. Second, the scaled vector $\boldsymbol{\pi}(n)/(1+\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\pi}(n))$ is used to compute $\mathbf{k}(n)$.

TABLE 10.1 Summary of the RLS Algorithm

Initialize the algorithm by setting

$$\hat{\mathbf{w}}(0) = \mathbf{0},$$
$$\mathbf{P}(0) = \delta^{-1}\mathbf{I},$$

and

$$\delta = \begin{cases} \text{small positive constant for high SNR} \\ \text{large positive constant for low SNR} \end{cases}.$$

For each instant of time, $n = 1, 2, \ldots$, compute

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1 + \lambda^{-1}\mathbf{u}^{\mathrm{H}}\mathbf{P}(n-1)\mathbf{u}(n)}$$
$$\xi(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n-1)\mathbf{u}(n),$$
$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n),$$

and

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{P}(n-1).$$

This two-stage computation of $\mathbf{k}(n)$ is preferred over the direct computation of $\mathbf{k}(n)$ using Eq. (10.18) from a finite-precision arithmetic point of view. (A more detailed discussion of this issue is deferred to Chapter 13.)

To initialize the RLS algorithm, we need to specify two quantities:

1. The *initial weight vector* $\hat{\mathbf{w}}(0)$. The customary practice is to set $\hat{\mathbf{w}}(0) = 0$.
2. The *initial correlation matrix* $\mathbf{\Phi}(0)$. Setting $n = 0$ in Eq. (10.8), we find that, with the use of prewindowing, we obtain

$$\mathbf{\Phi}(0) = \delta\mathrm{I},$$

where δ is the regularization parameter. The parameter δ should be assigned a small value for high signal-to-noise ratio (SNR) and a large value for low SNR, which may be justified on regularization grounds.

Further justification for the choice of δ as described here is presented in the next section.

10.4 SELECTION OF THE REGULARIZATION PARAMETER

In a detailed study reported in Moustakides (1997), the convergence behavior of the RLS algorithm was evaluated for a stationary environment, with particular reference to two variable parameters:

- The *signal-to-noise ratio* (SNR) of the tap-input data, which is determined by the prevalent operating conditions.
- The *regularization parameter* δ, which is under the designer's control.

To set the stage for a summary of the findings presented in Moustakides' work, let $\mathbf{F}(x)$ denote a matrix function of x, and let $f(x)$ denote a nonnegative scalar function of x, where the variable x assumes values in some set $\mathcal{A}_x$. We may then introduce the following definition:

$$\mathbf{F}(x) = \mathbf{\Theta}(f), \tag{10.28}$$

where there exist constants c_1 and c_2 that are independent of the variable x, such that

$$c_1 f(x) \leq \|\mathbf{F}(x)\| \leq c_2 f(x) \qquad \text{for all} \quad x \in \mathcal{A}_x, \tag{10.29}$$

and where $\|\mathbf{F}(x)\|$ is the *matrix norm* of $\mathbf{F}(x)$, which is itself defined by

$$\|\mathbf{F}(x)\| = (\text{tr}[\mathbf{F}^{\text{H}}(x)\mathbf{F}(x)])^{1/2}. \tag{10.30}$$

In Eq. (10.28), $\mathbf{\Theta}(f)$ is some function of $f(x)$. The significance of the definition introduced in this equation will become apparent presently.

As pointed out in Section 10.3, initialization of the RLS algorithm includes setting the initial value of the time-average correlation matrix

$$\mathbf{\Phi}(0) = \delta\mathbf{I}.$$

The dependence of the regularization parameter δ on SNR is given detailed meaning in Moustakides (1997). In particular, $\mathbf{\Phi}(0)$ is reformulated as

$$\mathbf{\Phi}(0) = \mu^{\alpha}\mathbf{R}_0, \tag{10.31}$$

where

$$\mu = 1 - \lambda \tag{10.32}$$

and $\mathbf{R}_0$ is a deterministic positive-definite matrix defined by

$$\mathbf{R}_0 = \sigma_u^2\mathbf{I}, \tag{10.33}$$

in which σ_u^2 is the variance of a data sample $u(n)$. Thus, according to Eqs. (10.31) and (10.33), the regularization parameter δ is defined by

$$\delta = \sigma_u^2\mu^{\alpha}. \tag{10.34}$$

[In Chapter 13, it is shown that the factor $1 - \lambda$ in the RLS algorithm plays a role similar to the step-size parameter μ in the LMS algorithm—hence the notation introduced in Eq. (10.32).]

The parameter α provides the mathematical basis for distinguishing the initial value of the correlation matrix $\mathbf{\Phi}(n)$ as small, medium, or large. In particular, for situations in which

$$\mu \in [0, \mu_0] \quad \text{with} \quad \mu_0 \ll 1 \tag{10.35}$$

we may distinguish three scenarios in light of the definition introduced in Eq. (10.31):

1. $\alpha > 0$, which corresponds to a small initial value $\mathbf{\Phi}(0)$.
2. $0 > \alpha \geq -1$, which corresponds to a medium initial value $\mathbf{\Phi}(0)$.
3. $-1 \geq \alpha$, which corresponds to a large initial value $\mathbf{\Phi}(0)$.

With these definitions and the three distinct initial conditions at hand, we may now summarize the findings reported in Moustakides (1997) on the selection of the regularization parameter δ in initializing the RLS algorithm for situations that are governed by Eq. (10.35):

1. *High SNR*
   When the noise level in the tap inputs is low (i.e., the input SNR is high, on the order of 30 dB or more), the RLS algorithm exhibits an exceptionally fast rate of convergence, provided that the correlation matrix is initialized with a small enough norm. Typically, this requirement is satisfied by setting $\alpha = 1$. As α is reduced toward zero [i.e., the matrix norm of $\mathbf{\Phi}(0)$ is increased], the convergence behavior of the RLS algorithm deteriorates.
2. *Medium SNR*
   In a medium SNR environment (i.e., the input SNR is on the order of 10 dB), the rate of convergence of the RLS algorithm is worse than the optimal rate for the high-SNR case, but the convergence behavior of the RLS algorithm is essentially insensitive to variations in the matrix norm of $\mathbf{\Phi}(0)$ for $-1 \leq \alpha < 0$.
3. *Low SNR*
   Finally, when the noise level in the tap inputs is high (i.e., the input SNR is on the order of −10 dB or less), it is preferable to initialize the RLS algorithm with a correlation matrix $\mathbf{\Phi}(0)$ with a large matrix norm (i.e., $\alpha \leq -1$), since this condition may yield the best overall performance.

These remarks hold for a stationary environment or a slowly time-varying one. If, however, there is an abrupt change in the state of the environment and the change takes place when the RLS algorithm has reached a steady state, then the filter interprets the abrupt change as renewed initialization with a "large" initial $\mathbf{\Phi}(0)$ wherein $n = 0$ corresponds to the instant at which the environment switched to a new state. In such a situation, the recommended practice is to *stop the operation of the RLS algorithm and restart anew by initializing it with a small* $\mathbf{\Phi}(0)$.

10.5 UPDATE RECURSION FOR THE SUM OF WEIGHTED ERROR SQUARES

The minimum value of the sum of weighted error squares, $\mathscr{E}_{\min}(n)$, results when the tap-weight vector is set equal to the least-squares estimate $\hat{\mathbf{w}}(n)$. To compute $\mathscr{E}_{\min}(n)$, we may then use the relation [see first line of Eq. (9.40)]

$$\mathscr{E}_{\min}(n) = \mathscr{E}_d(n) - \mathbf{z}^{\mathrm{H}}(n)\hat{\mathbf{w}}(n), \tag{10.36}$$

where $\mathscr{E}_d(n)$ is defined (using the notation of this chapter) by

$$\begin{aligned} \mathscr{E}_d(n) &= \sum_{i=1}^{n} \lambda^{n-i} |d(i)|^2 \\ &= \lambda \mathscr{E}_d(n-1) + |d(n)|^2. \end{aligned} \tag{10.37}$$

Therefore, substituting Eqs. (10.13), (10.25), and (10.37) into Eq. (10.36), we get

$$\begin{aligned}\mathscr{E}_{\min}(n) &= \lambda[\mathscr{E}_d(n-1) - \mathbf{z}^{\mathrm{H}}(n-1)\hat{\mathbf{w}}(n-1)] \\ &\quad + d(n)[d^*(n) - \mathbf{u}^{\mathrm{H}}(n)\hat{\mathbf{w}}(n-1)] \\ &\quad - \mathbf{z}^{\mathrm{H}}(n)\mathbf{k}(n)\xi^*(n),\end{aligned} \tag{10.38}$$

where, in the last term, we have restored $\mathbf{z}(n)$ to its original form. By definition, the expression inside the first set of brackets on the right-hand side of Eq. (10.38) equals $\mathscr{E}_{\min}(n-1)$. Also, by definition, the expression inside the second set of brackets equals the complex conjugate of the a priori estimation error $\xi(n)$. For the last term, we use the definition of the gain vector $\mathbf{k}(n)$ to express the inner product

$$\begin{aligned}\mathbf{z}^{\mathrm{H}}(n)\mathbf{k}(n) &= \mathbf{z}^{\mathrm{H}}(n)\boldsymbol{\Phi}^{-1}(n)\mathbf{u}(n) \\ &= [\boldsymbol{\Phi}^{-1}(n)\mathbf{z}(n)]^{\mathrm{H}}\mathbf{u}(n) \\ &= \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n),\end{aligned}$$

where (in the second line) we have used the Hermitian property of the correlation matrix $\boldsymbol{\Phi}(n)$ and (in the third line) we have used the fact that $\boldsymbol{\Phi}^{-1}(n)\mathbf{z}(n)$ equals the least-squares estimate $\hat{\mathbf{w}}(n)$. Accordingly, we may simplify Eq. (10.38) to

$$\begin{aligned}\mathscr{E}_{\min}(n) &= \lambda\mathscr{E}_{\min}(n-1) + d(n)\xi^*(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n)\xi^*(n) \\ &= \lambda\mathscr{E}_{\min}(n-1) + \xi^*(n)[d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n)] \\ &= \lambda\mathscr{E}_{\min}(n-1) + \xi^*(n)e(n),\end{aligned} \tag{10.39}$$

where $e(n)$ is the a posteriori estimation error. Equation (10.39) is the recursion for updating the sum of weighted error squares. We thus see that the product of the complex conjugate of $\xi(n)$ and $e(n)$ represents the correction term in this update. Note that the product is real valued, which implies that we always have

$$\xi(n)e^*(n) = \xi^*(n)e(n). \tag{10.40}$$

Conversion Factor

The formula of Eq. (10.39) involves two different estimation errors: the a priori estimation error $\xi(n)$ and the a posteriori estimation error $e(n)$, which are naturally related. To establish the relationship between these two estimation errors, we start with the defining equation (10.27) and substitute the update equation (10.25), obtaining

$$\begin{aligned}e(n) &= d(n) - [\hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n)]^{\mathrm{H}}\mathbf{u}(n) \\ &= d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n-1)\mathbf{u}(n) - \mathbf{k}^{\mathrm{H}}(n)\mathbf{u}(n)\xi(n) \\ &= (1 - \mathbf{k}^{\mathrm{H}}(n)\mathbf{u}(n))\xi(n),\end{aligned} \tag{10.41}$$

where, in the last line, we have made use of the definition given in Eq. (10.26). The ratio of the a posteriori estimation $e(n)$ to the a priori estimation $\xi(n)$ is called the *conversion factor*, denoted by $\gamma(n)$. We may thus write

$$\begin{aligned}\gamma(n) &= \frac{e(n)}{\xi(n)} \\ &= 1 - \mathbf{k}^{\mathrm{H}}(n)\mathbf{u}(n),\end{aligned} \tag{10.42}$$

the value of which is uniquely determined by the gain vector $\mathbf{k}(n)$ and the tap-input vector $\mathbf{u}(n)$.

10.6 EXAMPLE: SINGLE-WEIGHT ADAPTIVE NOISE CANCELLER

Consider the *single-weight, dual-input adaptive noise canceller* depicted in Fig. 10.3. The two inputs are represented by a *primary signal* $d(n)$, which consists of an *information-bearing signal* component and an additive *interference*, and a *reference signal* $u(n)$, which is correlated with the interference and has no detectable contribution to the information-bearing signal. The requirement is to exploit the properties of the reference signal in relation to the primary signal to suppress the interference at the output of the adaptive noise canceller.

Application of the RLS algorithm of Table 10.1 yields the following set of equations for this canceller (after rearranging terms):

$$k(n) = \left[\frac{1}{\lambda\hat{\sigma}_u^2(n-1) + |u(n)|^2}\right]u(n); \tag{10.43}$$

$$\xi(n) = d(n) - \hat{w}^*(n-1)u(n); \tag{10.44}$$

$$\hat{w}(n) = \hat{w}(n-1) + k(n)\xi^*(n); \tag{10.45}$$

$$\hat{\sigma}_u^2(n) = \lambda\hat{\sigma}_u^2(n-1) + |u(n)|^2. \tag{10.46}$$

In the last equation, $\hat{\sigma}_u^2(n)$, an estimate of the variance of the zero-mean reference signal $u(n)$, is the inverse of $P(n)$, the scalar version of the matrix $\mathbf{P}(n)$ in the RLS algorithm; that is,

$$\hat{\sigma}_u^2(n) = P^{-1}(n). \tag{10.47}$$

It is informative to compare the algorithm described in Eqs. (10.43) through (10.46) with its counterpart obtained using the normalized LMS algorithm; the version of the normalized LMS algorithm of particular interest in the context of our present situation is

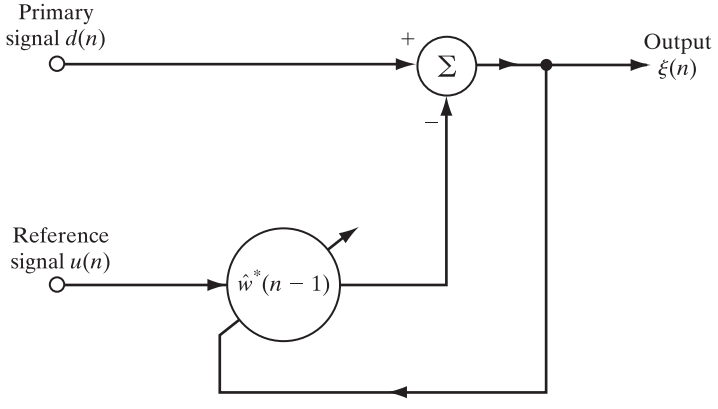


FIGURE 10.3 Single-weight adaptive noise canceller.

that given in Eq. (7.12). The *major difference* between these two algorithms is that the constant δ in the normalized LMS algorithm is replaced by the time-varying term $\lambda\hat{\sigma}_u^2(n-1)$ in the denominator of the gain factor $k(n)$ that controls the correction applied to the tap weight in Eq. (10.45).

10.7 STATISTICAL LEARNING THEORY

In this section, we discuss the convergence behavior of the RLS algorithm in a stationary environment, assuming that the exponential weighting factor λ is unity. (The case of λ less than unity is considered in Chapter 13.) To pave the way for the discussion, we make three assumptions, all of which are reasonable in their own ways.

Assumption 1 *The desired response $d(n)$ and the tap-input vector $\mathbf{u}(n)$ are related by the multiple linear regression model*

$$d(n) = \mathbf{w}_o^{\mathrm{H}}\mathbf{u}(n) + e_o(n), \tag{10.48}$$

where $\mathbf{w}_o$ is the regression parameter vector and $e_o(n)$ is the measurement noise. The noise $e_o(n)$ is white with zero mean and variance σ_o^2, which makes it independent of the regressor $\mathbf{u}(n)$.

The relationship expressed in Eq. (10.48) is depicted in Fig. 10.4, which is a reproduction of Fig. 9.1, used in the study of the method of least squares in Chapter 9.

Assumption 2 *The input signal vector $\mathbf{u}(n)$ is drawn from a stochastic process, which is ergodic in the autocorrelation function.*

The implication of Assumption 2 is that we may substitute time averages for ensemble averages, as discussed in Chapter 1. In particular, we may express the ensemble-average correlation matrix of the input vector $\mathbf{u}(n)$ as

$$\mathbf{R} \approx \frac{1}{n}\mathbf{\Phi}(n) \quad \text{for} \quad n > M, \tag{10.49}$$

where $\mathbf{\Phi}(n)$ is the time-average correlation matrix of $\mathbf{u}(n)$ and the requirement $n > M$ ensures that the input signal spreads across all the taps of the FIR filter. The approximation of Eq. (10.49) improves with an increasing number of adaptation cycles, n.

Assumption 3 *Fluctuations in the weight-error vector $\boldsymbol{\varepsilon}(n)$ are slow compared with those of the input signal vector $\mathbf{u}(n)$.*

The justification for Assumption 3 is to recognize that the weight-error vector $\boldsymbol{\varepsilon}(n)$ is the accumulation of a series of changes extending over n adaptation cycles of the RLS algorithm. This property is shown by

$$\begin{aligned}\boldsymbol{\varepsilon}(n) &= \mathbf{w}_o - \hat{\mathbf{w}}(n)\\ &= \boldsymbol{\varepsilon}(0) - \sum_{i=1}^{n}\mathbf{k}(i)\,\xi^*(i),\end{aligned} \tag{10.50}$$

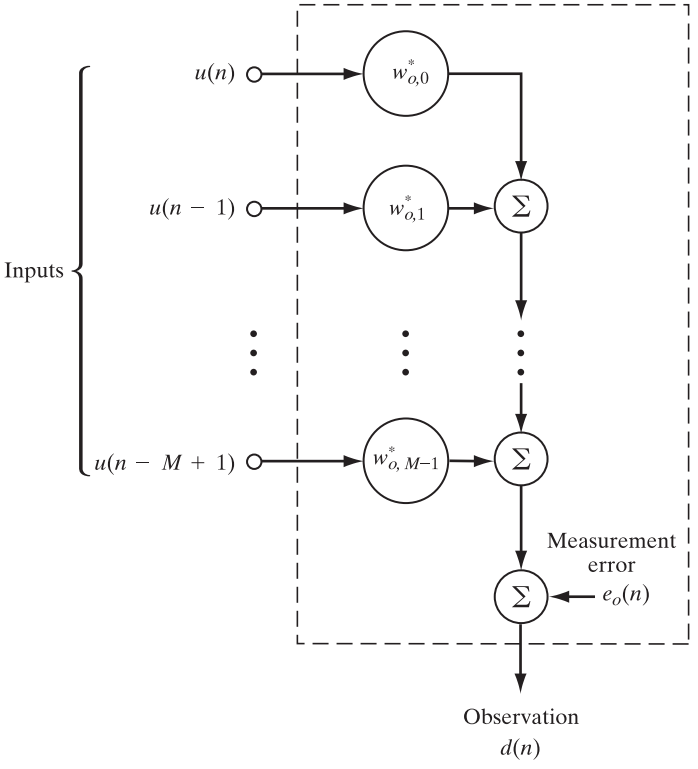


FIGURE 10.4 Multiple linear regression model.

which follows from Eq. (10.25). (See Problem 6.) Although both $\mathbf{k}(i)$ and $\xi(i)$ depend on $\mathbf{u}(i)$, the summation in Eq. (10.50) has a "smoothing" effect on $\boldsymbol{\varepsilon}(n)$. In effect, the RLS algorithm acts as a time-varying low-pass filter.

No further assumptions on the statistical characterization of $\mathbf{u}(n)$ and $d(n)$ are made in what follows.

Convergence in the Mean

Solving the normal equations (10.10) for $\hat{\mathbf{w}}(n)$, we may write

$$\hat{\mathbf{w}}(n) = \boldsymbol{\Phi}^{-1}(n)\mathbf{z}(n), \quad n > M, \tag{10.51}$$

where, for $\lambda = 1$,

$$\boldsymbol{\Phi}(n) = \sum_{i=1}^{n} \mathbf{u}(i)\mathbf{u}^{\mathrm{H}}(i) + \boldsymbol{\Phi}(0) \tag{10.52}$$

and

$$\mathbf{z}(n) = \sum_{i=1}^{n} \mathbf{u}(i)d^*(i). \tag{10.53}$$

Subsituting Eq. (10.48) into Eq. (10.53) and then using Eq. (10.52), we get

$$\begin{aligned}\mathbf{z}(n) &= \sum_{i=1}^{n}\mathbf{u}(i)\,\mathbf{u}^{\mathrm{H}}(i)\mathbf{w}_o + \sum_{i=1}^{n}\mathbf{u}(i)e_o^*(i)\\ &= \boldsymbol{\Phi}(n)\mathbf{w}_o - \boldsymbol{\Phi}(0)\mathbf{w}_o + \sum_{i=1}^{n}\mathbf{u}(i)e_o^*(i),\end{aligned} \tag{10.54}$$

where, in the last line, we have made use of Eq. (10.8) with $\lambda = 1$. This, in turn, means that we may rewrite Eq. (10.51) as

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \boldsymbol{\Phi}^{-1}(n)\boldsymbol{\Phi}(n)\mathbf{w}_o - \boldsymbol{\Phi}^{-1}(n)\boldsymbol{\Phi}(0)\mathbf{w}_o + \boldsymbol{\Phi}^{-1}(n)\sum_{i=1}^{n}\mathbf{u}(i)e_o^*(i)\\ &= \mathbf{w}_o - \boldsymbol{\Phi}^{-1}(n)\boldsymbol{\Phi}(0)\mathbf{w}_o + \boldsymbol{\Phi}^{-1}(n)\sum_{i=1}^{n}\mathbf{u}(i)e_o^*(i).\end{aligned} \tag{10.55}$$

Taking the expectation of both sides of Eq. (10.55) and invoking Assumptions 1 and 2, we may write

$$\begin{aligned}\mathbb{E}[\hat{\mathbf{w}}(n)] &\approx \mathbf{w}_o - \frac{1}{n}\mathbf{R}^{-1}\boldsymbol{\Phi}(0)\mathbf{w}_o\\ &= \mathbf{w}_o - \frac{\delta}{n}\mathbf{R}^{-1}\mathbf{w}_o\\ &= \mathbf{w}_o - \frac{\delta}{n}\mathbf{p}, \quad n > M,\end{aligned} \tag{10.56}$$

where $\mathbf{p}$ is the ensemble-average cross-correlation vector between the desired response $d(n)$ and input vector $\mathbf{u}(n)$. Equation (10.56) states that the RLS algorithm is convergent in the mean value. For finite n greater than the filter length M, the estimate $\hat{\mathbf{w}}(n)$ is *biased*, due to the initialization of the algorithm by setting $\boldsymbol{\Phi}(0) = \delta\mathbf{I}$, but the bias decreases to zero as n approaches infinity.

Convergence in the Mean Square

The weight-error correlation matrix is defined by

$$\begin{aligned}\mathbf{K}(n) &= \mathbb{E}[\boldsymbol{\varepsilon}(n)\boldsymbol{\varepsilon}^{\mathrm{H}}(n)]\\ &= \mathbb{E}[(\mathbf{w}_o - \hat{\mathbf{w}}(n))(\mathbf{w}_o - \hat{\mathbf{w}}(n))^{\mathrm{H}}].\end{aligned} \tag{10.57}$$

Substituting Eq. (10.55) into Eq. (10.57) and ignoring the effects of initialization, which is justified for $n > M$, we get

$$\mathbf{K}(n) = \mathbb{E}\left[\boldsymbol{\Phi}^{-1}(n)\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{u}(i)\mathbf{u}^{\mathrm{H}}(j)\boldsymbol{\Phi}^{-1}(n)e_o^*(i)e_o(j)\right].$$

Under Assumption 1, the input vector $\mathbf{u}(n)$, and therefore $\boldsymbol{\Phi}^{-1}(n)$, is independent of the measurement noise $e_o(n)$. Accordingly, we may express $\mathbf{K}(n)$ as the product of two expectations:

$$\mathbf{K}(n) = \mathbb{E}\left[\boldsymbol{\Phi}^{-1}(n)\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{u}(i)\,\mathbf{u}^{\mathrm{H}}(j)\boldsymbol{\Phi}^{-1}(n)\right]\mathbb{E}[e_o^*(i)e_o(j)].$$

Since, under Assumption 1, the measurement noise $e_o(n)$ is white, we have

$$\mathbb{E}[e_o^*(i)e_o(j)] = \begin{cases} \sigma_o^2 & \text{for} \quad i = j \\ 0 & \text{otherwise} \end{cases}, \tag{10.58}$$

where σ_o^2 is the variance of $e_o(n)$. Hence, the expression for the weight-error correlation matrix reduces to

$$\begin{aligned} \mathbf{K}(n) &= \sigma_o^2 \mathbb{E}\left[\boldsymbol{\Phi}^{-1}(n) \sum_{i=1}^{n} \mathbf{u}(i)\mathbf{u}^{\mathrm{H}}(i)\boldsymbol{\Phi}^{-1}(n) \right] \\ &= \sigma_o^2 \mathbb{E}[\boldsymbol{\Phi}^{-1}(n)\boldsymbol{\Phi}(n)\boldsymbol{\Phi}^{-1}(n)] \\ &= \sigma_o^2 \mathbb{E}[\boldsymbol{\Phi}^{-1}(n)]. \end{aligned}$$

Finally, invoking Assumption 2, embodied in Eq. (10.49), we may write[4]

$$\mathbf{K}(n) = \frac{1}{n}\sigma_o^2 \mathbf{R}^{-1}, \qquad n > M. \tag{10.59}$$

The *mean-square deviation* is defined by

$$\begin{aligned} \mathcal{D}(n) &= \mathbb{E}[\boldsymbol{\varepsilon}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}(n)] \\ &= \mathrm{tr}[\mathbf{K}(n)], \end{aligned} \tag{10.60}$$

where $\mathrm{tr}[\,\cdot\,]$ denotes the trace operator. In light of Eq. (10.59), the mean-square deviation of the RLS algorithm is

$$\begin{aligned} \mathcal{D}(n) &= \frac{1}{n}\sigma_o^2 \,\mathrm{tr}\,[\mathbf{R}^{-1}] \\ &= \frac{1}{n}\sigma_o^2 \sum_{i=1}^{M} \frac{1}{\lambda_i}, \qquad n > M, \end{aligned} \tag{10.61}$$

where the λ_i are the eigenvalues of the ensemble-average correlation matrix $\mathbf{R}$.

On the basis of Eq. (10.61), we may now make the following two important observations for $n > M$:

1. The mean-square deviation $\mathcal{D}(n)$ is *magnified by the inverse of the smallest eigenvalue* $\lambda_{\min}$. Hence, to a first order of approximation, the sensitivity of the RLS algorithm to eigenvalue spread is determined initially in proportion to the inverse

[4]The relation

$$\mathbb{E}[\boldsymbol{\Phi}^{-1}(n)] = \frac{1}{n}\mathbf{R}^{-1} \quad \text{for } n > M$$

is also justified in Appendix H on *complex Wishart distributions*. The correlation matrix $\boldsymbol{\Phi}^{-1}(n)$ is described by a complex Wishart distribution under the following conditions:

- The input vectors $\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(n)$ are independent and identically distributed (i.i.d.).
- The input vectors $\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(n)$ are drawn from a stochastic process with a multivariate Gaussian distribution of zero mean and an ensemble-average correlation matrix $\mathbf{R}$.

These two assumptions hold in an array-processing system that operates in a Gaussian environment.

of the smallest eigenvalue. Therefore, ill-conditioned least-squares problems may lead to poor convergence properties.

2. The mean-square deviation $\mathcal{D}(n)$ decays almost linearly with the number of adaptation cycles, n. Hence, the estimate $\hat{\mathbf{w}}(n)$ produced by the RLS algorithm converges in the norm (i.e., mean square) to the parameter vector $\mathbf{w}_o$ of the multiple linear regression model almost linearly with time.

Learning Curve

In the RLS algorithm, there are two types of error: the a priori estimation error $\xi(n)$ and the a posteriori estimation error $e(n)$. Given the initial conditions of Section 10.3, we find that the mean-square values of these two errors vary differently with time n. At time $n = 1$, the mean-square value of $\xi(n)$ becomes *large*—equal to the mean-square value of the desired response $d(n)$—and then *decays* with increasing n. The mean-square value of $e(n)$, on the other hand, becomes *small* at $n = 1$ and then *rises* with increasing n, until a point is reached for large n for which $e(n)$ is equal to $\xi(n)$. Accordingly, the choice of $\xi(n)$ as the error of interest yields a learning curve for the RLS algorithm that has the same general shape as that for the LMS algorithm. By choosing $\xi(n)$, we thus can make a direct graphical comparison between the learning curves of the RLS and LMS algorithms. We therefore base a computation of the ensemble-average learning curve of the RLS algorithm on the a priori estimation error $\xi(n)$, as follows:

$$J'(n) = \mathbb{E}[|\xi(n)|^2]. \tag{10.62}$$

The prime in the symbol $J'(n)$ is intended to distinguish the mean-square value of $\xi(n)$ from that of $e(n)$.

Eliminating the desired response $d(n)$ between Eqs. (10.26) and (10.48), we may express the a priori estimation error

$$\begin{aligned}\xi(n) &= e_o(n) + [\mathbf{w}_o - \hat{\mathbf{w}}(n-1)]^{\mathrm{H}}\mathbf{u}(n)\\ &= e_o(n) + \boldsymbol{\varepsilon}^{\mathrm{H}}(n-1)\mathbf{u}(n),\end{aligned} \tag{10.63}$$

where the second term involving $\boldsymbol{\varepsilon}(n-1)$ is an undisturbed estimation error. Substituting Eq. (10.63) into Eq. (10.62) and then expanding terms, we get

$$\begin{aligned}J'(n) = {} & \mathbb{E}[|e_o(n)|^2] + \mathbb{E}[\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}(n-1)\boldsymbol{\varepsilon}^{\mathrm{H}}(n-1)\mathbf{u}(n)]\\ & + \mathbb{E}[\boldsymbol{\varepsilon}^{\mathrm{H}}(n-1)\mathbf{u}(n)e_o^*(n)] + \mathbb{E}[e_o(n)\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}(n-1)].\end{aligned} \tag{10.64}$$

Examining the four expectation terms in Eq. (10.64), we may now make four observations:

1. The expectation of $|e_o(n)|^2$ is σ_o^2, by virtue of Eq. (10.58).
2. The second expectation is expressed as

$$\begin{aligned}\mathbb{E}[\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}(n-1)\boldsymbol{\varepsilon}^{\mathrm{H}}(n-1)\mathbf{u}(n)] &= \mathbb{E}[\mathrm{tr}\,\{\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}(n-1)\boldsymbol{\varepsilon}^{\mathrm{H}}(n-1)\mathbf{u}(n)\}]\\ &= \mathbb{E}[\mathrm{tr}\{\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}(n-1)\boldsymbol{\varepsilon}^{\mathrm{H}}(n-1)\}]\\ &= \mathrm{tr}\{\mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}(n-1)\boldsymbol{\varepsilon}^{\mathrm{H}}(n-1)]\}.\end{aligned}$$

Under Assumption 3, the outer product $\boldsymbol{\varepsilon}(n-1)\boldsymbol{\varepsilon}^{\mathrm{H}}(n-1)$ of the weight-error vector at time step n fluctuates at a slower rate than the outer product $\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)$. Applying Kushner's direct-averaging method, discussed in Chapter 6, we may write

$$\begin{aligned}\mathbb{E}[\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}(n-1)\boldsymbol{\varepsilon}^{\mathrm{H}}(n-1)\mathbf{u}(n)] &\approx \mathrm{tr}\{\mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]\mathbb{E}[\boldsymbol{\varepsilon}(n-1)\boldsymbol{\varepsilon}^{\mathrm{H}}(n-1)]\}\\ &= \mathrm{tr}\,[\mathbf{RK}(n-1)].\end{aligned} \tag{10.65}$$

Substituting Eq. (10.59) into Eq. (10.65) yields

$$\begin{aligned}\mathbb{E}[\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}(n-1)\boldsymbol{\varepsilon}^{\mathrm{H}}(n-1)\mathbf{u}(n)] &\approx \frac{1}{n}\sigma_o^2\mathrm{tr}[\mathbf{R}\mathbf{R}^{-1}]\\ &= \frac{1}{n}\sigma_o^2\mathrm{tr}[\mathbf{I}]\\ &= \frac{M}{n}\sigma_o^2, \quad n > M,\end{aligned} \tag{10.66}$$

where M is the filter length.

3. The third expectation is zero, for two reasons. First, viewing time step n as the present, we see that the weight-error vector $\boldsymbol{\varepsilon}(n-1)$ depends on past values of the input vector $\mathbf{u}(n)$ and measurement noise $e_o(n)$. [This statement follows from Eq. (10.50).] Second, under Assumption 1, $\mathbf{u}(n)$ and $e_o(n)$ are statistically independent, and $e_o(n)$ has zero mean. Hence, we may write

$$\begin{aligned}\mathbb{E}[\boldsymbol{\varepsilon}^{\mathrm{H}}(n-1)\mathbf{u}(n)e_o^*(n)] &= \mathbb{E}[\boldsymbol{\varepsilon}^{\mathrm{H}}(n-1)\mathbf{u}(n)]\ \mathbb{E}[e_o^*(n)]\\ &= 0.\end{aligned} \tag{10.67}$$

4. Finally, the fourth expectation has the same mathematical form as the third expectation, except for a trivial complex conjugation. Hence, the fourth expectation is also zero.

Using these results in Eq. (10.64), we obtain the following result:

$$J'(n) \approx \sigma_o^2 + \frac{M}{n}\sigma_o^2, \qquad n > M. \tag{10.68}$$

Convergence analysis of the RLS algorithm in this section assumes that the exponential weighting factor λ equals unity (i.e., the algorithm operates with infinite memory). As mentioned at the beginning of the section, the case of λ lying in the range $0 < \lambda < 1$ is deferred to Chapter 13.

10.8 EFFICIENCY

Now that we know how to formulate the learning curve of the RLS algorithm so as to be on a similar mathematical framework as the LMS algorithm, we are in the position to investigate its *statistical efficiency*.

To proceed, consider the much simplified Eq. (10.68), on the basis of which we may identify three distinctive properties of the RLS algorithm, operating in a stationary environment:

Property 1 *As the number of adaptation cycles, n, approaches infinity, the mean-square error, J'(n), approaches the optimal solution defined by the variance of the measurement error, $e_o(n)$, namely, $\sigma_o^2(n)$.*

To justify this property, consider the ideal condition under which the adjustable FIR filter used in the RLS algorithm has exactly the same length as the multiple linear regression model in Fig. 9.1. We may then invoke the critically matched case described in Section 2.6 of Chapter 2, wherein the minimum mean-square error of the Wiener filter has the irreducible value σ_o^2. Under this ideal condition, we thus have the following corollary to Property 1:

Corollary *As the number of adaptation cycles, n, approaches infinity, the RLS algorithm approaches the Wiener solution, in which case the misadjustment is zero.*

Property 2 *The ensemble-average learning curve of the RLS algorithm converges toward the final (i.e., Wiener) solution in about 2M adaptation cycles, where M is the length of the FIR filter built into the RLS algorithm.*

This property is justified on account of Monte Carlo simulations. Consequently, we may go on to say that the rate of convergence of the RLS algorithm is typically more than an order of magnitude faster than that of the LMS algorithm for the same stationary environment.

Property 3 *Convergence analysis of the RLS algorithm in the mean-square sense is essentially independent of the eigenvalues of the correlation matrix* $\mathbf{R}$ *of the input vector* $\mathbf{u}(n)$.

This property follows from Eq. (10.68). Accordingly, comparing Eq. (10.68) with the corresponding formula of Eq. (6.98) for the LMS algorithm, we may say:

The RLS algorithm is less sensitive to the eigenvalue spread of the correlation matrix $\mathbf{R}$ than the LMS algorithm.

We may also make the following final statement:

In general, the RLS algorithm is more statistically efficient than the LMS algorithm when both algorithms operate in the same stationary environment.

This statement is intuitively satisfying for the following reason: In statistical terms, the RLS algorithm is a second-order estimator, whereas the LMS algorithm is a first-order estimator.

In the next chapter on robustness, we will present a comparative evaluation of the LMS and RLS algorithms not only in terms of efficiency but also robustness, when they both operate in the same stationary environment.

10.9 COMPUTER EXPERIMENT ON ADAPTIVE EQUALIZATION

For this computer experiment, we use the RLS algorithm, with the exponential weighting factor $\lambda = 1$, for the adaptive equalization of a linear dispersive communication channel. The LMS version of this study was presented in Section 6.8. The block diagram of the system used in the study is depicted in Fig. 10.5. Two independent random-number generators are used—one, denoted by x_n, for probing the channel and the other, denoted by $v(n)$, for simulating the effect of additive white noise at the receiver input. The sequence x_n is a Bernoulli sequence with $x_n = \pm 1$; the random variable x_n has zero mean and unit variance. The second

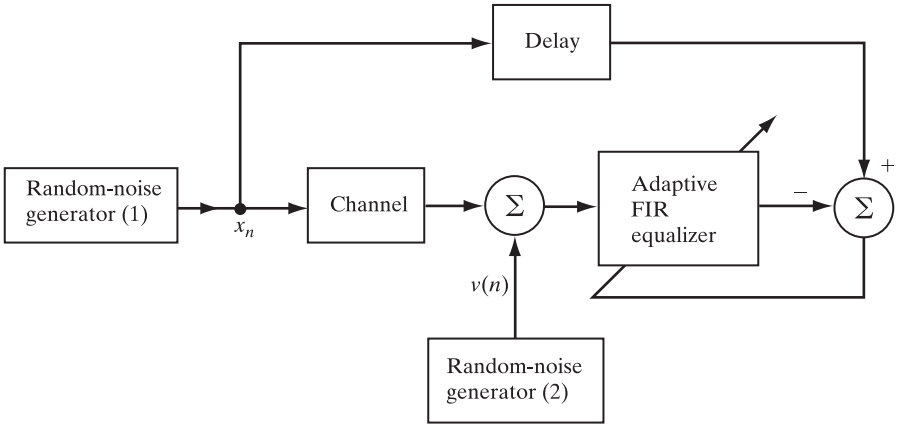


FIGURE 10.5 Block diagram of adaptive equalizer for computer experiment.

sequence $v(n)$ has zero mean, and its variance σ_v^2 is determined by the desired signal-to-noise ratio. The equalizer has 11 taps. The impulse response of the channel is defined by

$$h_n = \begin{cases} \frac{1}{2}\left[1 + \cos\left(\frac{2\pi}{W}(n-2)\right)\right], & n = 1, 2, 3 \\ 0 & \text{otherwise} \end{cases}, \tag{10.69}$$

where W controls the amount of amplitude distortion and, therefore, the eigenvalue spread produced by the channel. The channel input x_n, after a delay of seven samples, provides the desired response for the equalizer. (See Section 6.8 for details.)

The experiment is in two parts: In part 1 the signal-to-noise ratio (SNR) is high, and in part 2, it is low. In both parts of the experiment, the regularization parameter $\delta = 0.004$. (The procedure described in Section 10.4 for selecting δ for varying SNR does not apply here because of the choice $\lambda = 1$.)

1. Signal-to-Noise Ratio = 30 dB. The results of the experiment for a fixed signal-to-noise ratio of 30 dB (equivalently, a variance $\sigma_v^2 = 0.001$) and varying W or eigenvalue spread $\chi(\mathbf{R})$ were presented in Chapter 8. (See Fig. 8.9.) The four parts of that figure correspond to the parameter $W = 2.9, 3.1, 3.3$, and 3.5, or equivalently, $\chi(\mathbf{R}) = 6.078, 11.124, 21.713$, and 46.822, respectively. (See Table 6.2 for details.) Each part of the figure includes learning curves for the LMS, DCT-LMS, and RLS algorithms. The set of results pertaining to the RLS algorithm for the four different values of eigenvalue spread $\chi(\mathbf{R})$ is reproduced in Fig. 10.6. For comparison, the corresponding set of results for the LMS algorithm (with step-size parameter $\mu = 0.075$) is shown in Fig. 6.21. On the basis of the results presented in these two figures, we may make the following observations:

- Convergence of the RLS algorithm is attained in about 20 adaptation cycles, approximately twice the number of taps in the FIR equalizer.
- The rate of convergence of the RLS algorithm is relatively insensitive to variations in the eigenvalue spread $\chi(\mathbf{R})$, compared with the convergence of the LMS algorithm.

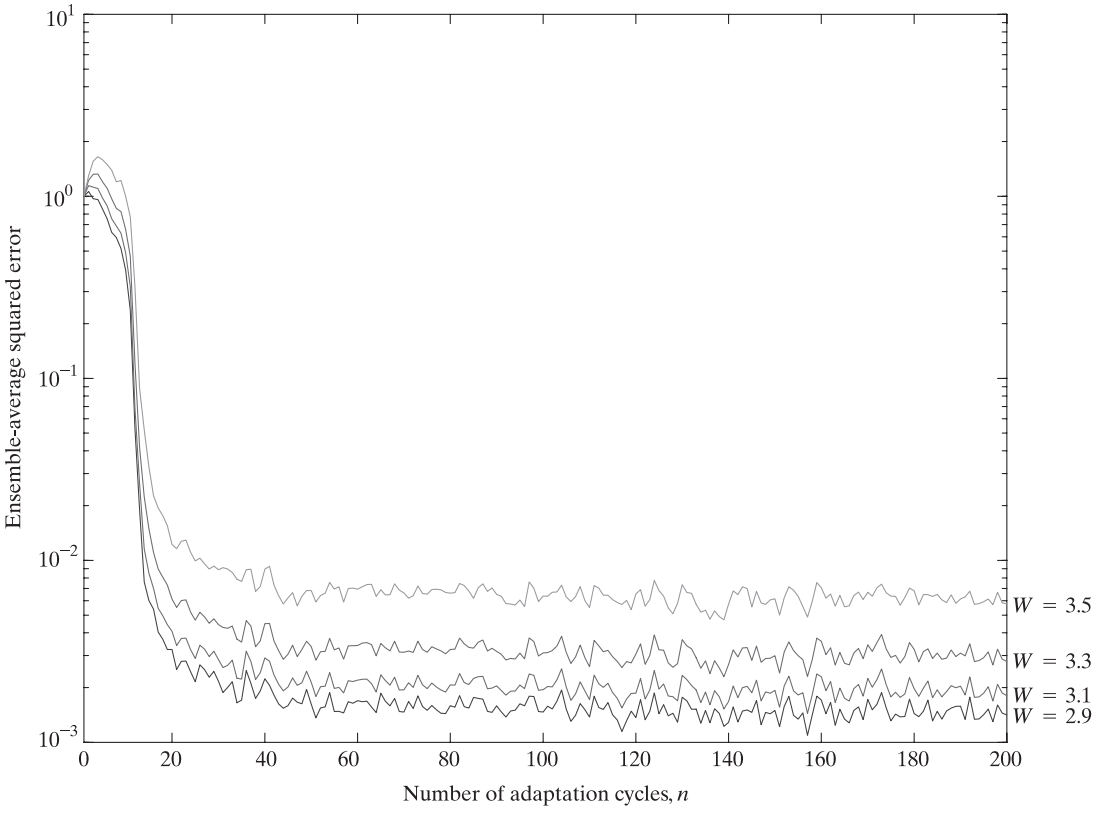


FIGURE 10.6 Learning curves for the RLS algorithm with four different eigenvalue spreads, $\delta = 0.004$, and $\lambda = 1.0$.

- The RLS algorithm converges much faster than the LMS algorithm; compare the results shown in Fig. 10.6 with those of the LMS algorithm plotted in Fig. 6.21.
- The steady-state value of the ensemble-average squared error produced by the RLS algorithm is much smaller than that of the LMS algorithm, confirming what we said earlier: The RLS algorithm produces zero misadjustment, at least in theory.

The results presented in Fig. 10.6 compared with those in Fig. 6.21 clearly show the much faster rate of convergence of the RLS over the LMS algorithm; for that rate to be realized, however, the signal-to-noise ratio has to be high. This advantage is lost when the signal-to-noise ratio is not high, as demonstrated next.

2. Signal-to-Noise Ratio = 10 dB. Figure 10.7 shows the learning curves for the RLS algorithm and the LMS algorithm (with the step-size parameter $\mu = 0.075$) for $W = 3.1$ and a signal-to-noise ratio of 10 dB. Insofar as the rate of convergence is concerned, we now see that the RLS and LMS algorithms perform in roughly the same manner, both requiring about 40 adaptation cycles to converge; however, neither algorithm supplies a good solution to the channel equalization problem.

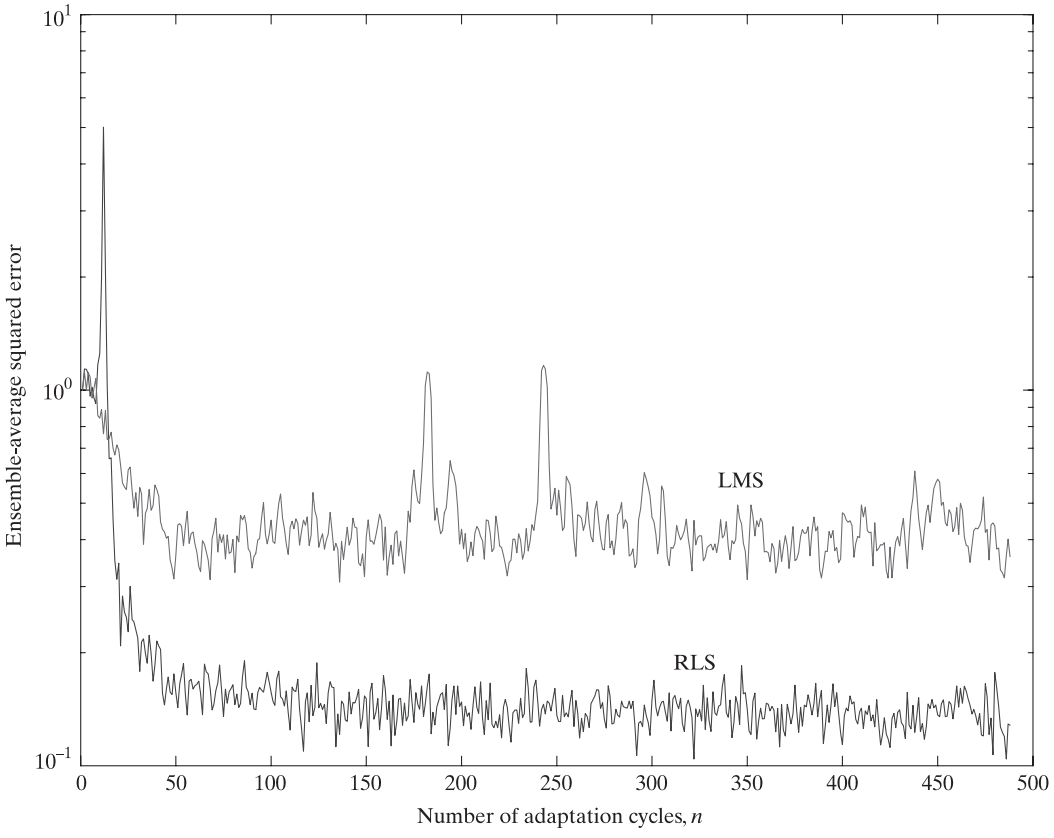


FIGURE 10.7 Learning curves for the RLS and LMS algorithms for $W = 3.1$ [i.e., eigenvalue spread $\chi(\mathbf{R}) = 11.124$] and SNR = 10 dB. RLS: $\delta = 0.004$ and $\lambda = 1.0$. LMS: Step-size parameter $\mu = 0.075$.

10.10 SUMMARY AND DISCUSSION

In this chapter, we derived the recursive least-squares (RLS) algorithm as a natural extension of the method of least squares. The derivation was based on a lemma in matrix algebra known as the matrix inversion lemma.

The fundamental difference between the RLS algorithm and the LMS algorithm may be summarized as follows: The step-size parameter μ in the LMS algorithm is replaced by $\mathbf{\Phi}^{-1}(n)$—that is, the inverse of the correlation matrix of the input vector $\mathbf{u}(n)$, which has the effect of *whitening* the tap inputs. This modification has a profound impact on the statistical efficiency, exemplified by convergence behavior, of the RLS algorithm for a stationary environment, which can be as follows:

1. The rate of convergence of the RLS algorithm is typically more than an order of magnitude faster than that of the LMS algorithm.
2. The rate of convergence of the RLS algorithm is essentially invariant to the eigenvalue spread (i.e., condition number) of the ensemble-average correlation matrix $\mathbf{R}$ of the input vector $\mathbf{u}(n)$.

3. The excess mean-square error $J'_{\text{ex}}(n)$ of the RLS algorithm converges to zero as the number of adaptation cycles, n, approaches infinity.

The operation of the RLS algorithm summarized herein applies to a stationary environment with the exponential weighting factor $\lambda = 1$. For the case of $\lambda \neq 1$, Properties 1 and 2 still hold, but the excess mean-square error $J'_{\text{ex}}(n)$ is no longer zero. In any event, computation of the mean-square error $J'(n)$, produced by the RLS algorithm, is based on the a priori estimation error $\xi(n)$.

With the material on the statistical efficiency of the RLS algorithm covered in this chapter and the corresponding material on the LMS algorithm covered in Chapter 6, the stage is set for the next chapter to make a detailed comparative evaluation between these two algorithms, operating in the same stationary environment. Therein, we will focus attention not only on statistical efficiency but also on robustness of the two basic linear adaptive filtering algorithms. These two properties will feature in the comparative study because of their practical importance. Naturally, computation complexity will also feature in the study.

PROBLEMS

1. To permit a recursive implementation of the method of least squares, the window or weighting function $\beta(n, i)$ must have a suitable structure. Assume that

$$\beta(n, i) = \lambda(i)\beta(n, i - 1), \qquad i = 1, \ldots, n,$$

where $\beta(n, n) = 1$. Show that

$$\beta(n, i) = \prod_{k=i+1}^{n} \lambda^{-1}(k).$$

What is the form of $\lambda(k)$ for which $\beta(n, i) = \lambda^{n-i}$ is obtained?

2. Derive the Riccati equation for the RLS algorithm.

3. Derive the inverse correlation matrix using the exponentially weighted RLS algorithm for a correlation matrix $\Phi(n) = \mathrm{u}(n)\mathrm{u}^{\mathrm{H}}(n) + \delta\mathrm{I}$ where $\mathbf{u}(n)$ is a tap-input vector and δ is a small positive constant.

4. What is the major difference between the LMS and RLS algorithms?

5. Show how the validity of the matrix inversion lemma can be applied to obtain a recursive equation for computing the least-squares solution $\hat{w}(n)$ for the tap weight vector.

6. How do we solve a constrained minimization problem for the optimum values of the filter coefficients in the MVDR spectrum estimation method?

7. The RLS algorithm differs from the LMS algorithm in a fundamental respect: The step-size parameter μ in the LMS algorithm is replaced by the inverse correlation matrix $\boldsymbol{\Phi}^{-1}(n)$.
   (a) Show that this replacement has a self-orthogonalizing effect in the RLS algorithm, which is expressed by

$$\mathbb{E}[\boldsymbol{\varepsilon}(n)] \approx \left(1 - \frac{1}{n}\right)\mathbb{E}[\boldsymbol{\varepsilon}(n - 1)] \quad \text{for large } n,$$

   where

$$\boldsymbol{\varepsilon}(n) = \mathbf{w}_o - \hat{\mathbf{w}}(n).$$

   For this derivation you may invoke Assumptions 1 and 2 of Section 10.7.

(b) In light of the result derived in part (a), distinguish between the RLS algorithm and the self-orthogonalizing adaptive filter discussed in Section 8.4.

8. In this problem, we use the complex Wishart distribution of Appendix H to study the convergence behavior of the RLS algorithm applied to an array signal processor that satisfies the following assumptions:

i. The observables $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N$ at the outputs of the N sensors constituting the array are i.i.d.

ii. The observables are drawn from a stochastic process with multivariate Gaussian distribution of zero mean and ensemble-average correlation matrix $\mathbf{R}$.

(a) Apply the complex Wishart distribution to obtain the expectation

$$\mathbb{E}[\boldsymbol{\Phi}^{-1}(n)] = \frac{1}{n}\mathbf{R}^{-1}.$$

(b) Using the result of part (a), derive the corresponding expression for the mean-square deviation for the RLS algorithm operating in the environment described in the aforementioned two assumptions.

Computer Experiments

9. Problem 17 of Chapter 6 addresses the application of the LMS algorithm to the design of a linear predictor operating on an autoregressive process of order two. Using the RLS algorithm, repeat parts (b) through (e) of the computer experiment described therein.

10. Explain the close relationship of the LMS algorithm to stochastic approximation when there is a decrease in the value of the step-size parameter μ with an increasing number of adaptation cycles.

11. Problem 20 of Chapter 6 addresses the application of the LMS algorithm to the study of an MVDR beamformer. Using the RLS algorithm, repeat the computer experiment described therein.

CHAPTER 11

Robustness

At this particular juncture in the book, it is highly instructive that we pause and look back on the major issues covered thus far. They are summarized here as follows:

- Adoption of the method of stochastic gradient descent for the least-mean-square (LMS) algorithm and the method of least squares for the recursive least-squares (RLS) algorithm.
- Computational complexity: linear law for the LMS algorithm, and square law for the RLS algorithm.
- Statistical efficiency measured with respect to the Wiener solution in a stationary environment: slow rate of convergence for the LMS algorithm and much faster rate of convergence for the RLS algorithm.

Now, then, recalling the factors itemized in the *Background and Preview* chapter, a factor yet to be discussed in depth is *robustness*. It is this very topic that will occupy much of our attention in this chapter. Specifically, the properties of statistical efficiency, computational complexity, and robustness, in the final analysis, provide the frame of reference for making a decision in favor of the LMS or RLS algorithm for an application of interest.

11.1 ROBUSTNESS, ADAPTATION, AND DISTURBANCES

With robustness as the topic of interest in this chapter, it is appropriate that we reproduce the statement on robustness made in the *Background and Preview* chapter:

> For an adaptive filter to be *robust*, small disturbances (i.e., disturbances with small energy) can only result in small estimation errors. The disturbances may arise from a variety of sources: internal or external to the filter.

The key words to be noted in this statement are: robust, adaptive, and disturbances.

From a practical perspective, the environmental presence of disturbances is unavoidable. To expand on what is being said here: If stochasticity of the environment is confined to Gaussianity, whiteness, or such other statistical characterizations, then adaptive filtering would be relegated to merely a problem in statistical parameter estimation, which is clearly not the goal of adaptive filtering in the first place. To be more specific, for adaptive filtering algorithms to occupy a distinctive place of their own in

the literature across the board, they would have to be capable of dealing with *noisy data of unknown statistics*. In other words, we may state (Hassibi, 2003):

> The process of adaptive filtering is much more related to robustness with respect to statistical variations than it is to optimality with respect to a prespecified statistical model.

In general, this statement says that we may have to opt for *suboptimality* in order to be assured of robustness in the face of disturbances. Given this practical reality, it is unfortunate that in the adaptive filtering literature, not enough attention has been given to robustness.

11.2 ROBUSTNESS: PRELIMINARY CONSIDERATIONS ROOTED IN H^∞ OPTIMIZATION

To set the stage for what H^∞ optimization stands for, it is enriching that we refer back to the first derivation of the LMS algorithm that was carried out in a "heuristic" manner in Section 5.2 of Chapter 5. Therein, for the cost function to be minimized with respect to the adjustable tap weights of the finite-duration impulse response (FIR) filter built into the algorithm, we used the squared modulus of the instantaneous estimation errors. In so doing, any connection between the LMS algorithm and the Wiener filter was completely severed. This prompts us to raise the following question:

> If the LMS algorithm is not optimum in the mean-square-error sense as in the case of Wiener filtering, what then is the actual criterion on the basis of which the LMS algorithm is optimized?

It turns out that the answer to this fundamental question lies in H^∞ optimization.[1]

[1] H^∞ optimization was first introduced in control theory to design "robust controllers." As mentioned previously in the section on historical notes in the *Background and Preview* chapter, the pioneering journal papers on H^∞ optimization are pioneered by Zames (1981), Zames and Francis (1983), and Francis and Zames (1984).

For linear time invariant (LTI) systems, it can be readily shown that the maximum energy gain is the maximum of the squared modulus of the LTI's transfer function over all frequencies, which, in turn, is the H^∞ norm. The original papers of Zames and Francis dealt with LTI systems, so they naturally used the H^∞ norm. Consequently, the H^∞ norm for robustness stuck, despite the fact that researchers have moved beyond LTI systems. In this modern setting, there is no transfer function, and the H^∞ norm for robustness does not apply in mathematical terms.

Today, therefore, when we speak of the H^∞ norm, what we actually mean is the $\mathcal{H}^2$-*induced norm*. The reason for saying so is that, by definition, H^∞ stands for the maximum of the ratio of two $\mathcal{H}^2$ norms, each of which represents energy. To be more precise, H^∞ is a member of the mathematical *Hardy spaces*, so named by Riesz (1923) to honor a paper written by Hardy (1915). Specifically:

- The H in H^∞ stands for Hardy spaces.
- The superscript, ∞, stands for the fact that H^∞ is the space of all analytic functions in complex variable theory that lie outside the unit circle with some finite magnitude on the unit circle (Duren, 2000).

For a tutorial exposition on H^∞ optimization, see the paper by Kwakernaak (1993); for the first journal paper on robustress of the LMS algorithm, see Hassibi et al. (1996). For books on the subject, see Francis (1987), and Basar and Bernhard (1991).

Uncertainties in Adaptive Filtering

In Section 11.1, we emphasized that in seeking a robust solution to adaptive filtering, we have to pay particular attention to the unavoidable presence of environmental disturbances. To this end, there are two issues to be considered:

1. In focusing on a sample realization of an adaptive filtering algorithm, it is logical to ignore underlying statistical assumptions and, therefore, adopt a *deterministic* approach to the adaptive filtering problem under a worst-case scenario, so as to safeguard against unknown sources of disturbances.
2. There is a need to examine the adaptive filtering problem itself as a possible source of disturbances.

To deal with the first issue, suppose that we have a set of training data $\{\mathbf{u}(n), d(n)\}$, which may fit into a *multiple linear regression model* described by the following equation[2]:

$$d(n) = \mathbf{w}^{\mathrm{H}}\mathbf{u}(n) + v(n), \tag{11.1}$$

where $\mathbf{u}(n)$ is the regressor, $d(n)$ is the desired response, $\mathbf{w}$ is a vector of unknown parameters to be estimated, the superscript H denotes *Hermitian transposition* (i.e., transposition combined with complex conjugation), and the additive term $v(n)$ is an unknown disturbance that arises on account of the following sources:

- *Measurement noise.* The source of this noise is attributed to the use of imperfect sensors.
- *Environmental modeling errors.* For example, the true model of the training data, $\{\mathbf{u}(n), d(n)\}$, may be actually characterized by an infinite-duration impulse response (IIR); to simplify matters, the tail end of the model's impulse response is ignored, so as to accommodate the use of a finite-duration impulse response (FIR) model for mathematical simplicity.
- *Other disturbances.* These disturbances may originate from other unknown sources.

Turning to the second issue, let $\hat{\mathbf{w}}(n)$ denote a *recursive estimate* of the unknown parameter vector $\mathbf{w}$ in Eq. (11.1), which is being computed using an adaptive filtering algorithm of interest by processing the complete training data $\{\mathbf{u}(i), d(i)\}$ for all $i \leq n$. The recursive estimate, $\hat{\mathbf{w}}(n)$, is *strictly causal*, which means that updating the old estimate $\hat{\mathbf{w}}(n-1)$ to $\hat{\mathbf{w}}(n)$ depends only on the current input $\mathbf{u}(n)$ at adaptation cycle n. In any event, the *weight-error vector*, denoted by $\tilde{\mathbf{w}}(n)$, is defined by[3]

$$\tilde{\mathbf{w}}(n) = \mathbf{w} - \hat{\mathbf{w}}(n). \tag{11.2}$$

[2]Equation (11.1) is of the same mathematical form as that of Eq. (6.6) in Chapter 6, but with an important difference to be noted: No assumptions are made in Eq. (11.1) on statistical characterization of the additive term, $v(n)$.

[3]The weight-error vector $\tilde{\mathbf{w}}(n)$ should not be confused with the weight-error vector $\boldsymbol{\varepsilon}(n)$ introduced in Chapter 6. In contrast to $\tilde{\mathbf{w}}(n)$, the error $\boldsymbol{\varepsilon}(n)$ is defined with respect to the Wiener solution, $\mathbf{w}_o$.

Typically, the chosen initial value, $\hat{\mathbf{w}}(0)$, used in the recursive estimation is different from $\mathbf{w}$. Accordingly, we have another source of disturbance to account for, namely,

$$\tilde{\mathbf{w}}(0) = \mathbf{w} - \hat{\mathbf{w}}(0). \tag{11.3}$$

Hence, in evaluating the deterministic behavior of an adaptive filtering algorithm based on a recursive estimation strategy, the two disturbances to be accounted for are:

1. The additive disturbance $v(n)$ in the regression model of Eq. (11.1).
2. The initial weight-error vector $\tilde{\mathbf{w}}(0)$ in Eq. (11.3).

Formulation of the H^∞-Optimization Problem

Let $\mathcal{T}$ denote a *causal estimator* that maps the above-mentioned disturbances on the input of a recursive estimation strategy to estimation errors at its output, as depicted in Fig. 11.1. Note that $\mathcal{T}$ is a function of the strategy used to compute the estimate $\hat{\mathbf{w}}(n)$ at adaptation cycle n. We may thus introduce the following definition:

> The energy gain of a causal estimator is the ratio of estimation-error energy at the output of the estimator to total disturbance energy at its input.

Clearly, the notion of *energy gain* so defined is dependent on the disturbances that are *unknown*. To remove this dependence, we introduce the following definition:

> The H^∞ norm of the causal estimator $\mathcal{T}$ is the maximum energy gain computed over all possible disturbance sequences of fixed energy.

Henceforth, we use the symbol γ^2 to denote the maximum energy gain, where the superscript 2 stands for the 2 in the $\mathcal{H}^2$-induced norm.

In solving the *H^∞-estimation problem*, the objective may now be stated as follows:

> Find the causal estimator that guarantees the smallest H^∞ norm over all possible estimators.

The optimal γ^2 resulting from this finding is denoted by γ^2_{opt}, on the basis of which we may go one step further to make the following statement[4]:

> For the causal estimator, depicted in Fig. 11.1, to be robust in the H^∞ sense, it is necessary and sufficient for the condition
>
> $$\gamma^2_{\text{opt}} \leq 1$$
>
> to be satisfied for all possible uncertainties.

[4]The following comparison is noteworthy:

- In statistical learning theory, the emphasis is on the expectation operator, where we treat an ensemble of independent realizations of an adaptive filtering algorithm.
- In contrast, in robustness, the emphasis is on having to consider the effort of all conceivable disturbances that could affect a single realization of the algorithm.

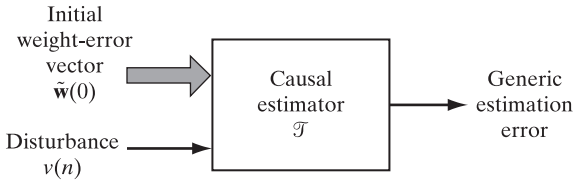


FIGURE 11.1 Formulation of the optimal H^∞ estimation problem. The generic estimation error at the causal estimator's output could be the weight-error vector, the uncorrupted output error, etc.

What if this condition is not satisfied? In such a situation, the causal estimator is said to be *less robust* than an estimator for which $\gamma_{\text{opt}}^2 < 1$. Another related point that should be carefully noted: If γ_{opt}^2 happens to be greater than 1, it simply means that there exist some, but not all, disturbances that are amplified by the estimator.

11.3 ROBUSTNESS OF THE LMS ALGORITHM

With the material covered in the preceding section, the stage is now set for us to show that the LMS algorithm is indeed robust in the H^∞ sense, provided that the step-size parameter, μ, used in the algorithm is small enough to satisfy a certain condition to be determined.[5]

The H^∞ optimal estimator so found is of a *minimax* nature. More specifically, we may view the H^∞ optimal estimation problem as a game-theoretic problem (Basar & Bernhard, 1991):

> Nature, acting as the opponent, has access to the unknown disturbances, thereby maximizing the energy gain. On the other hand, the designer has the choice of finding a causal estimator (algorithm) that minimizes the energy gain.

Since no assumptions are made about the disturbances, an H^∞ estimator has to account for all possible disturbances. Accordingly, such an estimator may be "overconservative"; that is, it is a "worst-case" estimator.

As a measurable error signal directly related to the weight-error vector $\tilde{\mathbf{w}}(n)$, we introduce the *undisturbed estimation error*, or *undisturbed error signal*, defined as

$$\begin{aligned}\xi_u(n) &= (\mathbf{w} - \hat{\mathbf{w}}(n))^{\mathrm{H}}\mathbf{u}(n) \\ &= \tilde{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n).\end{aligned} \tag{11.4}$$

The term "undisturbed," signified by subscript u, is used here to distinguish the estimation error $\xi_u(n)$ from the estimation error $e(n)$ in Chapter 5. Specifically, $\xi_u(n)$ compares the filter's response $\hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n)$ with the "undisturbed" response $\mathbf{w}^{\mathrm{H}}\mathbf{u}(n)$ of the multiple

[5]The H^∞ optimality of the LMS algorithm was first described in a conference paper by Hassibi et al. (1993), which was followed by a journal paper in 1996. In a related context, the interested reader may refer to Hassibi (2003) for a philosophical discussion of this same topic.

It should also be noted that H^∞-optimal adaptive filtering algorithms are not unique. Specifically, H^∞ optimality may be generalized, for example, to include the normalized LMS algorithm. (See Problem 7.)

linear regression model, rather than the desired response $d(n)$. Indeed, from the defining equations (5.7) and (11.4), we readily find that the two estimation errors $\xi_u(n)$ and $e(n)$ are related by

$$\xi_u(n) = e(n) - v(n), \tag{11.5}$$

where $v(n)$ is the additive disturbance in the multiple linear regression model.

Cauchy–Schwarz Inequality for Computing the H^∞ Norm

To compute the H^∞ norm for any estimator, we need to compute the worst-case energy gain from the disturbances to the undisturbed estimation error of Eq. (11.4). To attempt to bound this energy gain, we next propose to apply the *Cauchy–Schwarz inequality* to the inner product $\widetilde{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n)$. For a statement of this inequality, consider the inner product of two complex vectors $\mathbf{a}$ and $\mathbf{b}$ of compatible dimensions. The Cauchy–Schwarz inequality states that the absolute value of the inner product $\mathbf{a}^{\mathrm{H}}\mathbf{b}$ has upper bound $\|\mathbf{a}\|\|\mathbf{b}\|$; that is,

$$|\mathbf{a}^{\mathrm{H}}\mathbf{b}|^2 \le \|\mathbf{a}\|^2\|\mathbf{b}\|^2.$$

For the problem at hand, we have

$$\mathbf{a} = \widetilde{\mathbf{w}}(n)$$

and

$$\mathbf{b} = \mathbf{u}(n).$$

Hence, application of the Cauchy–Schwarz inequality to the problem at hand yields

$$|\widetilde{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n)|^2 \le \|\widetilde{\mathbf{w}}(n)\|^2\|\mathbf{u}(n)\|^2$$

or, equivalently, from the definition of Eq. (11.4),

$$|\xi_u(n)|^2 \le \|\widetilde{\mathbf{w}}(n)\|^2\|\mathbf{u}(n)\|^2. \tag{11.6}$$

Suppose next we choose a *positive* real number μ that satisfies the condition

$$0 < \mu < \frac{1}{\|\mathbf{u}(n)\|^2}.$$

Clearly, if this condition holds, then we may recast the inequality of Eq. (11.6) as follows:

$$|\xi_u(n)|^2 \le \mu^{-1}\|\widetilde{\mathbf{w}}(n)\|^2. \tag{11.7}$$

Furthermore, we can go one step further: Given that the inequality of Eq. (11.7) holds for an arbitrary estimate $\hat{\mathbf{w}}(n)$, it will still hold if the right-hand side of the inequality is increased by the squared amplitude of the disturbance $|v(n)|^2$; that is,

$$|\xi_u(n)|^2 \le \mu^{-1}\|\widetilde{\mathbf{w}}(n)\|^2 + |v(n)|^2, \tag{11.8}$$

which brings the disturbance $v(n)$ into the analysis.

Thus far in the discussion, we have not said how the estimate $\hat{\mathbf{w}}(n)$ is to be computed. We now take care of this matter by using the LMS algorithm with the positive

real number μ as its step-size parameter. The update formula for the estimate $\hat{\mathbf{w}}(n)$ is defined in Eq. (5.5). With the use of this formula, it is a straightforward matter to show that the inequality of Eq. (11.8) is tightened by adding the term $\mu^{-1}\|\tilde{\mathbf{w}}(n+1)\|^2$ to the left-hand side, provided that μ is less than $1/\|\mathbf{u}(n)\|^2$. (For a proof of this statement, see Problem 6.) Thus, the tightened inequality is

$$\mu^{-1}\|\tilde{\mathbf{w}}(n+1)\|^2 + |\xi_u(n)|^2 \leq \mu^{-1}\|\tilde{\mathbf{w}}(n)\|^2 + |\nu(n)|^2, \tag{11.9}$$

which now includes all the disturbances and parameters of interest.

As the adaptive filtering process progresses across time, the input vector $\mathbf{u}(n)$ naturally varies with discrete time n. Suppose that we run the LMS algorithm for $N+1$ adaptation cycles, starting from $n=0$ with the initial condition $\hat{\mathbf{w}}(0)$, thereby generating the sequence of estimates $\{\hat{\mathbf{w}}(0), \hat{\mathbf{w}}(1), \ldots, \hat{\mathbf{w}}(N)\}$ and the corresponding sequence of undisturbed estimation errors $\{\xi_u(0), \xi_u(1), \ldots, \xi_u(N)\}$. The members of both causal sequences satisfy the inequality of Eq. (11.9), provided that for any integer N the step-size parameter μ itself satisfies the condition

$$0 < \mu < \min_{1 \leq n \leq N} \frac{1}{\|\mathbf{u}(n)\|^2}. \tag{11.10}$$

Equation (11.10) should be true for all n. However, since the denominator on the right-hand side of this equation is the norm of the regression (input) vector $\mathbf{u}(n)$, it follows that the step-size parameter, μ, need not vanish.[6] The left-hand side of the inequality is thereby justified.

Robustness of the LMS Algorithm in the H^∞ Sense Confirmed

Provided, then, that the condition of Eq. (11.10) on the step-size parameter μ holds, starting from the initial condition $\hat{\mathbf{w}}(0)$ and summing the two sides of the inequality in Eq. (11.9) over the interval $0 \leq n \leq N$, we get (after cancelling common terms)

$$\mu^{-1}\|\tilde{\mathbf{w}}(N)\|^2 + \sum_{n=0}^{N} |\xi_u(n)|^2 \leq \mu^{-1}\|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=0}^{N} |\nu(n)|^2.$$

Clearly, if this inequality holds, then we certainly have

$$\sum_{n=0}^{N} |\xi_u(n)|^2 \leq \mu^{-1}\|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=0}^{N} |\nu(n)|^2. \tag{11.11}$$

[6]The decrease in the value of the step-size parameter μ with an increasing number of adaptation cycles in accordance with Eq. (11.10) brings to mind the close relationship of the LMS algorithm to stochastic approximation (Robbins & Monro, 1951; Sakrison, 1966), which is defined by

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{1}{n}\mathbf{u}(n)e^*(n),$$

where $1/n$ plays the role of a time-dependent step-size parameter.

Previously, we introduced γ^2 in Section 11.2 to denote the H^∞ norm (i.e., maximum energy gain). Accordingly, using γ^2_{LMS} to denote the H^∞ norm of the LMS algorithm with step-size parameter μ, we may use Eq. (11.11) to formally write

$$\gamma^2_{\text{LMS}}(\mu) = \sup_{\mathbf{w},\, \nu \in \mathcal{H}^2} \frac{\sum_{n=0}^{N} |\xi_u(n)|^2}{\mu^{-1}\|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=0}^{N} |\nu(n)|^2}, \tag{11.12}$$

where $\mathcal{H}^2$ denotes the *space of all square-summable causal sequences*, and "sup" stands for *supremum* (i.e., the lowest upper bound). The denominator of Eq. (11.12) denotes the total disturbance energy, made up of two components: $\mu^{-1}\|\hat{\mathbf{w}}(0)\|^2$ is the disturbance energy due to the choice of initial condition $\hat{\mathbf{w}}(0)$, and $\sum_{n=0}^{N} |\nu(n)|^2$ is the energy of the disturbance $\nu(n)$ in the multiple linear regression model. The numerator of Eq. (11.12) is the energy of the undisturbed estimation error $\xi_u(n)$ produced by the LMS algorithm at its output.

Equation (11.11) has an important practical interpretation:

> Provided that the step-size parameter μ of the LMS algorithm satisfies the condition of Eq. (11.11), then, no matter how different the initial weight vector $\hat{\mathbf{w}}(0)$ is from the unknown parameter vector $\mathbf{w}$ of the multiple linear regression model, and irrespective of the value of the additive disturbance $\nu(n)$, the error energy produced at the output of the LMS algorithm will never exceed the sum of the two disturbance energies at the filter's input, produced by the choice of initial condition $\hat{\mathbf{w}}(0)$ and the presence of input disturbance $\nu(n)$.

This statement explains the reason for the *robust* behavior of the LMS algorithm as it endeavors to estimate the unknown parameter vector $\mathbf{w}$ in the face of unavoidable disturbances affecting the operation of the algorithm.

From Eqs. (11.11) and (11.12), we readily see that

$$\gamma^2_{\text{LMS}}(\mu) \le 1 \quad \text{for} \quad 0 < \mu \le \min_{1 \le n \le N} \frac{1}{\|\mathbf{u}(n)\|^2} \quad \text{and any integer } N. \tag{11.13}$$

We have thus shown that the maximum energy gain (i.e., H^∞ norm) for the LMS algorithm is bounded by unity; in other words, the estimation error energy never exceeds the disturbance energy. The surprising fact that emerges from the discussion presented herein is that the maximum energy gain for the LMS algorithm is exactly unity. Moreover, it turns out that there exists *no* other algorithm that can achieve a maximum energy gain strictly less than unity. This implies that *the LMS algorithm is H^∞ optimal*.

More on Robustness of the LMS Algorithm

To demonstrate this remarkable property of the LMS algorithm, we shall now introduce a disturbance sequence such that the maximum energy gain can be made arbitrarily close to unity for *any* algorithm. To this end, envision a disturbance $\nu(n)$ that satisfies the setting

$$\nu(n) = -\xi_u(n) \quad \text{for all } n.$$

At first sight, such a setting may appear unrealistic. However, it is indeed conceivable, as it lies within the unrestricted model of the disturbance $v(n)$ described in Eq. (11.1). Under this special setting, we readily deduce from Eq. (11.5) that $e(n) = 0$ for all n.

Now it is not difficult to see that any algorithm that yields a bounded maximum energy gain will not change its estimate of the unknown weight vector $\mathbf{w}$ when confronted with the error signal $e(n) = 0$. (Otherwise, we could get nonzero estimation errors when the disturbance energy is zero, thus leading to an infinite energy gain.) For example, this is certainly true of the LMS algorithm described by the update equation (5.5) in Chapter 5. We therefore conclude that for this particular disturbance, and for *any* algorithm that yields a finite maximum energy gain, we have

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(0) \qquad \text{for all } n.$$

Since $\xi_u(n) = -v(n)$, the energy gain of Eq. (11.12) for this particular disturbance takes the following special form:

$$\gamma^2(\mu) = \frac{\sum_{n=0}^{N} |\xi_u(n)|^2}{\mu^{-1}\|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=0}^{N} |\xi_u(n)|^2} \qquad \text{for } \xi_u(n) = -v(n). \tag{11.14}$$

When

$$\lim_{N\to\infty} \sum_{n=0}^{N} \|\mathbf{u}(n)\|^2 < \infty \tag{11.15}$$

(i.e., the input vectors $\mathbf{u}(n)$ are *excitatory*), it follows that, for any positive constant Δ, we can find a parameter vector $\mathbf{w}$ and an integer N such that

$$\sum_{n=0}^{N} |\xi_u(n)|^2 = \sum_{n=0}^{N} |(\mathbf{w} - \hat{\mathbf{w}}(n))^H \mathbf{u}(n)|^2 \geq \frac{1}{\Delta\mu} \|\mathbf{w} - \hat{\mathbf{w}}(0)\|^2 = \frac{1}{\Delta\mu}\|\tilde{\mathbf{w}}(0)\|^2.$$

Using these choices in Eq. (11.14), cancelling common terms, and keeping in mind the bounds on γ^2 we may write

$$\frac{1}{1+\Delta} \leq \gamma^2 \leq 1.$$

We can now see that for the particular disturbance, $\xi_u(n) = -\gamma^2(n)$, the maximum energy gain of any algorithm can be made arbitrarily close to unity by allowing the constant Δ to approach zero. Since the "worst-case" disturbance for any algorithm must yield a maximum energy gain that is no smaller than this special value, we may say the following:

> The maximum energy gain for *any* algorithm, which corresponds to the "worst-case" disturbance, can never be less than unity.

However, Eq. (11.13) teaches us the following:

> The maximum energy gain for the LMS algorithm does *not* exceed unity.

Thus, the two statements (just made) imply that the LMS algorithm is indeed H^∞ optimal; that is, it minimizes the maximum energy gain, and that

$$\gamma_{\text{opt}}^2 = 1.$$

Summarizing Remarks

The material just presented is important for the following reasons:

- Since the time that adaptive equalizers were first used for telephone channels, it was recognized that an adaptive equalizer based on the LMS algorithm was robust with respect to disturbances on a telephone channel.
- The preceding practical observation is confirmed theoretically by the fact that the LMS algorithm is in fact *optimal in the H^∞ sense*.

Credit must therefore be given to the insightful vision of Widrow and Hoff in pioneering the LMS algorithm in 1960.

11.4 ROBUSTNESS OF THE RLS ALGORITHM

Now that we have demonstrated the H^∞ optimality of the LMS algorithm, our next task is twofold[7]:

1. To work on robustness of the RLS algorithm.
2. To compare robustness of the RLS algorithm to that of the LMS algorithm.

To this end, the model used for robustness of the RLS algorithm embodies the same pair of disturbances introduced in Section 11.3 for the LMS algorithm; that is, we have the additive disturbance, $v(n)$, defined in Eq. (11.1), and the error, $\tilde{\mathbf{w}}(0)$, in the initial estimate of the unknown tap-weight vector $\mathbf{w}$ as defined in Eq. (11.2). However, with the RLS algorithm being different from the LMS algorithm, the undisturbed estimation error for the RLS algorithm is defined as follows:

$$\xi_u(n) = (\mathbf{w} - \hat{\mathbf{w}}(n-1))^{\text{H}}\mathbf{u}(n), \tag{11.16}$$

which differs from the case of LMS algorithms: $\hat{\mathbf{w}}(n-1)$ is used in place of $\hat{\mathbf{w}}(n)$ to be consistent with the RLS theory presented in Chapter 10. Correspondingly, $\xi_u(n)$ is related to $v(n)$ as

$$\xi_u(n) = \xi(n) - v(n), \tag{11.17}$$

where $\xi(n)$ is the a priori estimation error.

To study robustness of the RLS algorithm we need to compute bounds on the maximum energy gain from the input disturbances to the estimation errors. To obtain these bounds, we will build on the following lemma, the proof of which is presented as Problem 9.

[7]The treatment of H^∞ theory applied to the RLS algorithm described herein follows Hassibi and Kailath (2001).

Lemma. Consider the RLS algorithm whose weight vector is recursively updated in accordance with the equation

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n),$$

where $\mathbf{k}(n) = \mathbf{\Phi}^{-1}(n)\mathbf{u}(n)$ is the gain vector. Then, assuming that the time-average correlation matrix $\mathbf{\Phi}(n)$ is invertible and the exponential weighting factor λ is equal to unity, we may write

$$\tilde{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{\Phi}(n)\tilde{\mathbf{w}}(n) + \frac{|\xi(n)|^2}{r(n)} = \tilde{\mathbf{w}}^{\mathrm{H}}(n-1)\mathbf{\Phi}(n-1)\tilde{\mathbf{w}}(n-1) + |\nu(n)|^2, \quad (11.18)$$

where, for $\lambda = 1$, the denominator term $r(n)$ in left-hand side of the equation is defined by

$$r(n) = 1 + \mathbf{u}^{\mathrm{H}}(n)\mathbf{\Phi}^{-1}(n-1)\mathbf{u}(n).$$

This term is the reciprocal of the conversion factor $\gamma(n)$ defined in Eq. (10.42); note, however, the $\gamma(n)$ of Eq. (10.42) must not be confused with the H^∞ norm introduced in this chapter.

Upper Bound on the H^∞ Norm of the RLS Algorithm

The rationale behind introducing the lemma of Eq. (11.18) is that it provides a basis for finding an upper bound in the H^∞ norm of the RLS algorithm. Thus, summing the two sides of Eq. (11.18) for all values of n in the interval $1 \le n \le N$, we get (after cancelling common terms)

$$\tilde{\mathbf{w}}^{\mathrm{H}}(N)\mathbf{\Phi}(N)\tilde{\mathbf{w}}(N) + \sum_{n=1}^{N}\frac{|\xi(n)|^2}{r(n)} = \tilde{\mathbf{w}}^{\mathrm{H}}(0)\mathbf{\Phi}(0)\tilde{\mathbf{w}}(0) + \sum_{n=1}^{N}|\nu(n)|^2. \quad (11.19)$$

In the RLS algorithm, the correlation matrix $\mathbf{\Phi}(n)$ is initialized with the value

$$\mathbf{\Phi}(0) = \delta\mathbf{I},$$

where δ is the regularization parameter. (See Section 10.3.) Accordingly, we may rewrite Eq. (11.19) as

$$\tilde{\mathbf{w}}^{\mathrm{H}}(N)\mathbf{\Phi}(N)\tilde{\mathbf{w}}(N) + \sum_{n=1}^{N}\frac{|\xi(n)|^2}{r(n)} = \delta\|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=1}^{N}|\nu(n)|^2. \quad (11.20)$$

The right-hand side of Eq. (11.20) is precisely the disturbance error energy that we are seeking. The left-hand side, however, is not quite the estimation error energy. To isolate the estimation error energy, we need to apply a few tricks. First, note that if we define

$$\bar{r} = \max_n r(n),$$

then we can rewrite Eq. (11.20) in the form of an inequality:

$$\tilde{\mathbf{w}}^{\mathrm{H}}(N)\mathbf{\Phi}(N)\tilde{\mathbf{w}}(N) + \frac{1}{\bar{r}}\sum_{n=1}^{N}|\xi(n)|^2 \le \delta\|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=1}^{N}|\nu(n)|^2 \quad (11.21)$$

or, more simply,

$$\frac{1}{\bar{r}}\sum_{n=1}^{N}|\xi(n)|^2 \leq \delta\|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=1}^{N}|\nu(n)|^2.$$

From Eq. (11.17), we have

$$\xi(n) = \xi_u(n) + \nu(n).$$

Hence, we may go one step further to write

$$\frac{1}{\bar{r}}\sum_{n=1}^{N}|\xi_u(n) + \nu(n)|^2 \leq \delta\|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=1}^{N}|\nu(n)|^2. \tag{11.22}$$

To isolate the estimation error energy, we now note that, for any positive parameter $\alpha > 0$, we have the following inequality:

$$|a + b|^2 \geq \left(1 - \frac{1}{\alpha}\right)|a|^2 - (1 - \alpha)|b|^2,$$

where a and b are an arbitrary pair of variables. Applying this inequality, with $a = \xi_u(n)$ and $b = \nu(n)$, to Eq. (11.22), we obtain (after a rearrangement of terms)

$$\frac{1 - \frac{1}{\alpha}}{\bar{r}}\sum_{n=1}^{N}|\xi_u(n)|^2 \leq \delta\|\tilde{\mathbf{w}}(0)\|^2 + \left(1 + \frac{1 - \alpha}{\bar{r}}\right)\sum_{n=1}^{N}|\nu(n)|^2, \tag{11.23}$$

where the undisturbed estimation error energy term, $\sum_{n=1}^{N}|\xi_u(n)|^2$, is now isolated. Dividing both sides of Eq. (11.23) by $\left(1 - \frac{1}{\alpha}\right)/\bar{r}$ and assuming that $\alpha > 1$, we obtain,

$$\sum_{n=1}^{N}|\xi_u(n)|^2 \leq \frac{\bar{r}\delta}{1 - \frac{1}{\alpha}}\|\tilde{\mathbf{w}}(0)\|^2 + \frac{\alpha^2 + \alpha(\bar{r} - 1)}{\alpha - 1}\sum_{n=1}^{N}|\nu(n)|^2.$$

To obtain the "tightest" possible bound on $\sum_{n=1}^{N}|\xi_u(n)|^2$, we minimize over $\alpha > 1$ and the summation $\sum_{n=1}^{N}|\nu(n)|^2$ in the above inequality. In fact, it is not hard to show that

$$\min_{\alpha>1}\left(\frac{\alpha^2 + (\bar{r} - 1)\alpha}{\alpha - 1}\right) = (1 + \sqrt{\bar{r}})^2$$

and

$$\arg\min_{\alpha>1}\left(\frac{\alpha^2 + (\bar{r} - 1)\alpha}{\alpha - 1}\right) = 1 + \sqrt{\bar{r}}.$$

Utilizing these results, we may now go on to write

$$\sum_{n=1}^{N}|\xi_u(n)|^2 \leq \frac{\bar{r}\delta}{1 - \frac{1}{1 + \sqrt{\bar{r}}}}\|\tilde{\mathbf{w}}(0)\|^2 + (1 + \sqrt{\bar{r}})^2\sum_{n=1}^{N}|\nu(n)|^2$$

$$= \sqrt{\bar{r}}(1 + \sqrt{\bar{r}})\delta\|\tilde{\mathbf{w}}(0)\|^2 + (1 + \sqrt{\bar{r}})^2 \sum_{n=1}^{N} |\nu(n)|^2$$
$$\leq (1 + \sqrt{\bar{r}})^2 \left(\delta\|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=1}^{N} |\nu(n)|^2 \right).$$

We have thus upper bounded the energy gain of the RLS algorithm as follows:

$$\frac{\sum_{n=1}^{N} |\xi_u(n)|^2}{\delta\|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=1}^{N} |\nu(n)|^2} \leq (1 + \sqrt{\bar{r}})^2. \tag{11.24}$$

Since this inequality is true for all disturbances, we clearly have

$$\sup_{\mathbf{w},\, \nu \in \mathcal{H}^2} \left(\frac{\sum_{n=1}^{N} |\xi_u(n)|^2}{\delta\|\tilde{w}(0)\|^2 + \sum_{n=1}^{N} |\nu(n)|^2} \right) \leq (1 + \sqrt{\bar{r}})^2, \tag{11.25}$$

which is the desired upper bound on the maximum energy gain (or H^∞ norm) of the RLS algorithm.

Lower Bound on the H^∞ Norm for the RLS Algorithm

The strategy we used to obtain a lower bound on the maximum energy gain of the LMS algorithm in Section 11.3 was to construct a suitable disturbance signal and to compute its energy gain; the rationale for doing this is that the energy gain for any disturbance signal serves as a lower bound on the maximum (or worst-case) energy gain. We shall adopt a similar strategy now for the RLS algorithm.

To this end, note that we can always choose the disturbance sequence $v(n)$ to be such that the inequality in Eq. (11.21) is achieved: All we need to do is choose the $v(n)$ in such a way that $\xi(n) = \xi_u(n) + \bar{v}(n)$ is zero *except* for the time instant for which $\bar{r} = \max_n r(n)$ is achieved. Furthermore, we can always choose the unknown tap-weight vector $\mathbf{w}$ such that $\tilde{\mathbf{w}}(N) = \mathbf{w} - \hat{\mathbf{w}}(N) = \mathbf{0}$. Denoting this particular disturbance signal by $\{\hat{\mathbf{w}}, \bar{v}(n)\}$, we may then use Eq. (11.20) to write

$$\frac{1}{\bar{r}} \sum_{n=1}^{N} |\xi_u(n) + \bar{v}(n)|^2 = \delta\|\bar{\tilde{\mathbf{w}}}(0)\|^2 + \sum_{n=1}^{N} |\bar{v}(n)|^2, \tag{11.26}$$

where the new initial condition $\bar{\tilde{\mathbf{w}}}(0)$ corresponds to the choice $\bar{\mathbf{w}}$. We now proceed with a *dual* argument for α that gave us the upper bound of Eq. (11.25). Specifically, for all $\alpha > 0$, we have

$$|a + b|^2 \leq \left(1 + \frac{1}{\alpha}\right)|a|^2 + (1 + \alpha)|b|^2.$$

Applying this inequality, with $a = \xi_u(n)$ and $b = v(n)$, to Eq. (11.26), we obtain (after a rearrangement of terms)

$$\frac{1+\frac{1}{\alpha}}{\bar{r}}\sum_{n=1}^{N}|\xi_u(n)|^2 \geq \delta\|\bar{\tilde{\mathbf{w}}}(0)\|^2 + \left(1-\frac{1+\alpha}{\bar{r}}\right)\sum_{n=1}^{N}|\bar{\nu}(n)|^2.$$

Dividing both sides of this inequality by $\left(1+\frac{1}{\alpha}\right)/\bar{r}$, we obtain

$$\sum_{n=1}^{N}|\xi_u(n)|^2 \geq \frac{\bar{r}\delta}{1+\frac{1}{\alpha}}\|\bar{\tilde{\mathbf{w}}}(0)\| + \frac{-\alpha^2+\alpha(\bar{r}-1)}{\alpha+1}\sum_{n=1}^{N}|\bar{\nu}(n)|^2. \tag{11.27}$$

To obtain the "tightest" possible bound on $\sum_{n=1}^{N}|\xi_u(n)|^2$, we maximize over $\alpha>0$ and the summation $\sum_{n=1}^{N}|\bar{\nu}(n)|^2$ on the right-hand side of Eq. (11.27). Here again, it is not hard to show that

$$\min_{\alpha>0}\left(\frac{-\alpha^2+(\bar{r}-1)\alpha}{\alpha+1}\right) = (\sqrt{\bar{r}}-1)^2$$

and

$$\arg\min_{a>0}\left(\frac{-\alpha^2+(\bar{r}-1)\alpha}{\alpha+1}\right) = \sqrt{\bar{r}}-1.$$

Therefore, using this pair of results in Eq. (11.27) yields

$$\begin{aligned}\sum_{n=1}^{N}|\xi_u(n)|^2 &\geq \frac{\bar{r}\delta}{1+\frac{1}{\sqrt{\bar{r}}-1}}\|\bar{\tilde{\mathbf{w}}}(0)\|^2 + (\sqrt{\bar{r}}-1)^2\sum_{n=1}^{N}|\bar{\nu}(n)|^2\\ &= \sqrt{\bar{r}}(\sqrt{\bar{r}}-1)\delta\|\bar{\tilde{\mathbf{w}}}(0)\|^2 + (\sqrt{\bar{r}}-1)^2\sum_{n=1}^{N}|\bar{\nu}(n)|^2\\ &\geq (\sqrt{\bar{r}}-1)^2\left(\delta\|\bar{\tilde{\mathbf{w}}}(0)\|^2 + \sum_{n=1}^{N}|\bar{\nu}(n)|^2\right).\end{aligned}$$

Accordingly, we may go on to write the following expression for the lower bound on the energy gain produced by the RLS algorithm:

$$\begin{aligned}\gamma_{\text{RLS}}^2 &= \sup_{\mathbf{w},\,\nu\in\mathcal{H}^2}\left(\frac{\sum_{n=1}^{N}|\xi_u(n)|^2}{\delta\|\bar{\tilde{\mathbf{w}}}(0)\|^2 + \sum_{n=1}^{N}|\nu(n)|^2}\right)\\ &\geq \left(\sqrt{\bar{r}}-1\right)^2.\end{aligned} \tag{11.28}$$

Combining Lower and Upper Bounds

With the results of Eqs. (11.25) and (11.28) at hand, we may now combine this pair of equations in the manner shown here:

$$\left(\sqrt{\bar{r}}-1\right)^2 \leq \gamma_{\text{RLS}}^2 \leq \left(\sqrt{\bar{r}}+1\right)^2. \tag{11.29}$$

The upper and lower bounds in the RLS algorithm shown in Eq. (11.29) appear to be relatively tight (especially when the term $\bar{r}$ is large) since they differ by 2 in that

$$\left(\sqrt{\bar{r}} + 1\right) - \left(\sqrt{\bar{r}} - 1\right) = 2.$$

11.5 COMPARATIVE EVALUATIONS OF THE LMS AND RLS ALGORITHMS FROM THE PERSPECTIVE OF ROBUSTNESS

Now that we have completed the analysis of the LMS and RLS algorithms in terms of their respective H^∞ norms (i.e., γ^2), we are now ready to make the following comparisons between them:

1. Unlike the LMS algorithm, the maximum energy gain, γ^2_{RLS} in Eq. (11.28), can exceed unity, which means that the RLS algorithm can amplify disturbance signals. It follows therefore that, in general, the RLS algorithm is less robust to uncertainties than the LMS algorithm.
2. The maximum energy gain of the RLS algorithm is dependent on the input data because both the lower and upper bounds in Eq. (11.29) depend on the term

$$\bar{r} = \max_n [1 + \mathbf{u}^{\text{H}}(n)\boldsymbol{\Phi}^{-1}(n-1)\mathbf{u}(n)],$$

which is data dependent. Accordingly, unlike the LMS algorithm, robustness of the RLS algorithm depends on the input data, which should not be surprising because the RLS algorithm is based on the method of least-squares that is model-dependent.

3. Noting that

$$\begin{aligned}\bar{r} &\geq r(0)\\ &= 1 + \delta^{-1}\|\mathbf{u}(0)\|^2,\end{aligned}$$

where δ is the regularization parameter in the RLS algorithm, we may redefine the upper bound on the H^∞ norm of the RLS algorithm in Eq. (11.29) by writing

$$\gamma^2_{\text{RLS}} \leq \left(\sqrt{1 + \delta^{-1}\|\mathbf{u}(0)\|^2} + 1\right)^2. \tag{11.30}$$

This upper bound clearly shows that the smaller we make the regularization parameter δ, the less robust the RLS algorithm is. In a way, the observation just made, albeit in a different way, is reminiscent of the behavior of the LMS algorithm: The larger we make the step-size parameter μ, the less robust the LMS algorithm is.

11.6 RISK-SENSITIVE OPTIMALITY

One other noteworthy issue related to robustness that distinguishes the LMS algorithm from its RLS counterpart is *risk-sensitive optimality*. To expand on this issue, consider the regression model of Eq. (11.1), reproduced here for convenience of presentation:

$$d(n) = \mathbf{w}^{\text{H}}\mathbf{u}(n) + v(n),$$

where $\{\mathbf{u}(n), d(n)\}_{n=1}^{N}$ denotes the training data set, the unknown vector $\mathbf{w}$ parameterizes the multiple linear regression model, and $v(n)$ is additive disturbance. Adopting a stochastic interpretation of the risk-sensitive optimality problem, assume that the sources of uncertainties, represented by $\mathbf{w}$ and $v(n)$ in the model, are Gaussian distributed. Specifically:

- $\mathbf{w}$ has zero mean and correlation matrix is $\mu\mathbf{I}$, where μ is an adjustable positive parameter and $\mathbf{I}$ is the identity matrix.
- $v(n)$ has zero mean and unit variance; moreover, it is assumed to be independently and identically distributed (i.i.d.) Gaussian.

Furthermore, let $y(n)$, denoting the output of the estimator of interest (i.e., linear adaptive filtering algorithm), be dependent on past values of the desired response, $d(n)$, as shown by

$y(1) = 0,$
$y(2)$ depends on $d(1)$,
$y(3)$ depends on $d(1)$ and $d(2)$,

and so on for n up to and equal to N, denoting the training data length. Then, under the stochastic scenario just described, we may make the following two statements (Hassibi, 2003):

1. The LMS algorithm recursively solves the risk-sensitive optimization problem

$$\min_{\{y(n)\}_{n=1}^{N}} \mathbb{E}_{\{\mathbf{u}(n),\, d(n)\}_{n=1}^{N}} \left[\exp\left(\sum_{i=1}^{N} (y(i) - \mathbf{w}^{\mathrm{H}}\mathbf{u}(i))^2 \right) \right], \tag{11.31}$$

provided that the step-size parameter, μ, satisfies the condition

$$\mu \leq \frac{1}{\|\mathbf{u}(n)\|^2} \quad \text{for all } n.$$

2. Moreover, provided that the input vector $\mathbf{u}(n)$ is excitatory, then there is no estimator that renders the *cost function* defined by the expectation over the entire data set, namely,

$$\mathbb{E}_{\{d(n),\, \mathbf{u}(n)\}_{n=1}^{N}} \left[\exp\left(\frac{1}{\gamma^2} \sum_{i=1}^{N} (y(i) - \mathbf{w}^{\mathrm{H}}\mathbf{u}(i))^2 \right) \right], \tag{11.32}$$

to be finite for any $\gamma < 1$.

Correspondingly, under the same Gaussian assumptions imposed on the disturbances, the RLS algorithm minimizes the mean-square-error cost function:

$$\mathbb{E}_{\{\mathbf{u}(i),\, d(n)\}_{n=1}^{N}} \left[\sum_{i=1}^{N} (y(i) - \mathbf{w}^{\mathrm{H}}\mathbf{u}(i))^2 \right]. \tag{11.33}$$

Comparing the cost function of Eq. (11.32) for the LMS algorithm with that of Eq. (11.33) for the RLS algorithm, we readily see that because of the exponential function in Eq. (11.32), the LMS algorithm places a much larger penalty on large values of

the estimation error, given by the difference, $d(n) - \mathbf{w}^{H}\mathbf{u}(n)$. Stated in a comparative way, we may say that the LMS algorithm is more sensitive to large prediction errors than the RLS algorithm in the following sense:

> Whereas the RLS algorithm is concerned with the frequent occurrence of estimation errors that are of moderate values, the LMS algorithm is more concerned with the occasional occurrence of large estimation errors.

It is for this reason that the formula of Eq. (11.32), exemplified by the LMS algorithm, is referred to as a *risk-sensitive criterion*.[8]

11.7 TRADE-OFFS BETWEEN ROBUSTNESS AND EFFICIENCY

Having studied the comparative robustness of the LMS and RLS algorithms in this chapter, when the key issue of interest is robustness in the face of unknown disturbances, the LMS algorithm outperforms the RLS algorithm in general. But, then, there is another key practical issue, namely *efficiency*, which provides a statistical yardstick of its own for assessing the cost of finding an acceptable solution to an adaptive filtering problem.

As a reminder, the idea of efficiency was discussed in Chapter 6 on the LMS algorithm and Chapter 10 on the RLS algorithm. Therein, we established that the statistical efficiency of an adaptive filtering algorithm is measured in terms of the *rate of convergence*, defined as the number of adaptation cycles needed for the algorithm to relax to a steady state on or close to the Wiener solution starting from some initial condition. Typically, relaxation (i.e., convergence) of the LMS algorithm is *slower* than that of the RLS algorithm by more than an order of magnitude. It follows therefore that this time around, when the issue of interest is performance, the RLS algorithm outperforms the LMS algorithm.

These two conflicting scenarios point to the lack of a "universal" best adaptive filtering algorithm for all conceivable problems that can arise in practice. In particular, the lack of universality may be viewed as a manifestation of the so-called *no-free-lunch theorem* for optimization, which may be succinctly stated as follows (see Wolpert and Macreedy, 1997):

> An elevated performance of an algorithm over one class of problems is guaranteed to be offset over another class.

Accordingly, in the context of adaptive filtering algorithms, we have a fundamental *trade-off* between deterministic robustness in the face of uncertainties on the one hand and statistical efficiency on the other. Resolution of this trade-off can only be attained by considering the practical realities of the application of interest.

[8] In Whittle (1990), estimators that minimize the criterion of Eq. (11.32), parameterized by positive values of γ, are said to constitute a *family of risk-averse estimators*.

In a historical context, it should also be noted that the idea of an exponential quadratic cost was first described in classic control theory by Jacobson (1973).

Lessons Drawn from the No-Free-Lunch Theorem

Typically, but not always, robustness overrides efficiency in many applications of adaptive filtering. The intuitive rationale for saying so is that, from an engineering perspective, it would be preferable to employ an adaptive filtering algorithm that is robust and therefore capable of providing a relatively satisfactory performance when the algorithm is expected to operate under unknown disturbances. Generally speaking, in a nonstationary environment expected to suffer from unknown disturbances, robustness is frequently traded off for performance by choosing the LMS algorithm over its RLS counterpart; this issue is discussed in more detail in Chapter 13.

Another issue that weighs on this choice, favoring LMS, is *complexity*. From the discussions presented in Chapters 6 and 10, recall that computational complexity of the LMS algorithm scales *linearly* with respect to the size of the FIR filter (i.e., the number of tap weights), whereas computational complexity of the RLS algorithm follows a *square law*. Here again, the simplicity of the LMS algorithm overrides the RLS algorithm.

It is for these two compelling reasons that the LMS algorithm and its variants are the most widely used adaptive filtering algorithms for signal processing and control applications (Widrow & Kamenetsky, 2003).

Having stressed popularity of the LMS algorithm, however, we should not overlook applications of the RLS algorithm where the need for performance overrides robustness. For example, in wireless communications, where time is of the essence, the fast rate of convergence of the RLS algorithm makes it the method of choice over the relatively slow rate of convergence of the LMS algorithm for adaptive equalization.

This advantage of the RLS algorithm over the LMS algorithm is well illustrated in the typical example of Fig. 11.2 for a Digital Enhanced Cordless Telecommunications telephone (Fuhl, 1994). In this figure, we see that the RLS algorithm has converged after about 10 bits of data, whereas the LMS algorithm requires almost 300 bits of data to converge. The large "residual" error of the RLS algorithm in Figure 11.2, compared to

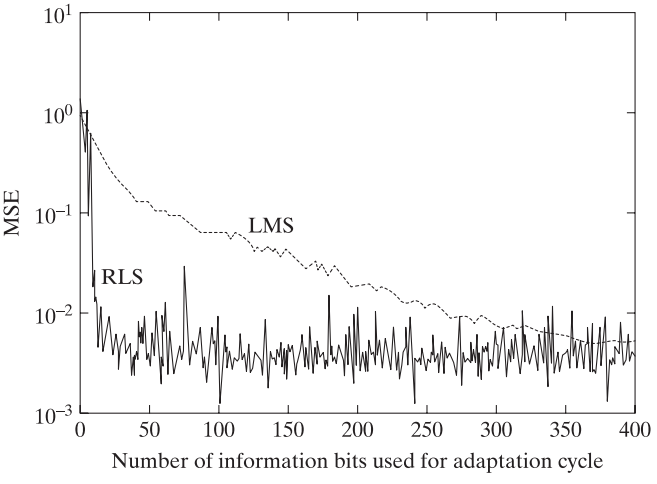


FIGURE 11.2 Mean-square error (MSE) as a function of the number of adaptation cycles for a special kind of equalizer using feedback in decision making, hence the name "decision feedback equalizer". LMS: $\mu = 0.03$; RLS: $\lambda = 0.99$; $\delta = 10^{-9}$. See also Molisch (2011).

the LMS algorithm, is attributed to environmental fluctuations that appear to affect the RLS algorithm more than the LMS algorithm. This observation confirms the inferior robust behavior of the RLS algorithm compared to the LMS algorithm. Notwithstanding this relative weakness of the RLS algorithm due to temporal variations of the wireless channel, from a practical perspective we find that the fast rate of convergence of the RLS algorithm is deemed to be more important than the extremely small residual error rate of the LMS algorithm. In other words, in this wireless communication example, statistical efficiency of the RLS algorithm is favored over robustness of the LMS algorithm.

11.8 SUMMARY AND DISCUSSION

By introducing a detailed treatment of robustness of an adaptive filtering algorithm in this chapter, in effect, we have evened out the playing field as a counterpart to statistical efficiency considered in previous chapters. In this expanded field, we may therefore make two important remarks insofar as the LMS and RLS algorithms are concerned:

1. First, the LMS algorithm is *model-independent* in statistical terms. Hence, with reliance on the method of stochastic gradient descent for its derivation, it is destined to provide a suboptimal solution. By assigning a small enough value to its step-size parameter, μ, its robustness in the H^{∞} sense is assured. But, the price paid for this practical benefit is a degraded statistical efficiency, which shows up in a relatively long rate of convergence.
2. Second, the RLS algorithm is *model-dependent* in that its derivation is based on the method of least squares, which is pivoted on a multiple linear regression model for describing the desired response. Accordingly, the RLS algorithm is optimal insofar as statistical efficiency is concerned, which manifests itself in a rate of convergence that is faster than that of the LMS algorithm by more than an order of magnitude. However, the price paid for this desirable property is increased sensitivity to disturbances, which results in a degraded robust behavior in general.

What we have just described here is merely another example of what the no-free-lunch thereom teaches us.

To conclude, we may say that in the final analysis, the decision in favor of the LMS or RLS algorithm for linear adaptive filtering hinges on the application of interest and the environment in which it needed to operate.

PROBLEMS

Notes. In the next four problems, we revisit robustness of the LMS algorithm, with each problem casting a perspective of its own on robustness. The first three problems, viewed together, reinforce robustness of the LMS algorithm, albeit in a way different from that presented in Section 11.3. The fourth problem provides further insight on robustness of the LMS algorithm. To simplify matters, the training data are real valued.

1. Consider the LMS algorithm, for which the update formula is given by

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \mathbf{u}(n)e(n),$$

where the error signal

$$e(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n)$$

and the desired response

$$d(n) = \mathbf{w}^{\mathrm{T}}\mathbf{u}(n) + v(n).$$

The superscript T denotes *transposition*. The $\mathbf{w}$ is an unknown weight vector, for which $\hat{\mathbf{w}}(n)$ is the estimate computed by the algorithm at adaptation cycle n. The unknown $\mathbf{w}$ and $v(n)$ are responsible for disturbances affecting the algorithm's behavior.

Define the weight-error vector

$$\tilde{\mathbf{w}}(n) = \mathbf{w} - \hat{\mathbf{w}}(n).$$

Let the estimate of the desired response, $d(n)$, be defined by

$$\hat{d}(n) = \hat{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n)$$

and the step-size parameter, μ, satisfy the condition

$$\mu < \frac{1}{\|\mathbf{u}(n)\|^2} \quad \text{for all } n.$$

The requirement is to show that the following inequality

$$\frac{\sum_{n=1}^{N} \xi_u^2(n)}{\mu^{-1}\|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=0}^{N} v^2(n)} \leq 1$$

holds in accordance with the terminology introduced in Section 11.3.

(a) Starting with the scaled expression, $\mu^{-\frac{1}{2}}\tilde{\mathbf{w}}(n)$, show that

$$\mu^{-\frac{1}{2}}\tilde{\mathbf{w}}(n+1) = \mu^{-\frac{1}{2}}\tilde{\mathbf{w}}(n) - \mu^{-\frac{1}{2}}\mathbf{u}(n)(d(n) - \hat{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n)).$$

(b) Correspondingly, show that

$$v(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n) - \tilde{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n).$$

(c) Having evaluated both the squared Euclidean norm of the equation under part (a) and the squared magnitude of the equation under part (b), and, then, having subtracted the results so obtained, show that

$$\mu^{-1}\|\tilde{\mathbf{w}}(n+1)\|^2 - v^2(n) = \mu^{-1}\|\tilde{\mathbf{w}}(n)\|^2 - (\tilde{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n))^2 - (1 - \mu\|\mathbf{u}(n)\|^2)(d(n) - \tilde{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n))^2.$$

(d) Add up the equations under part (c) from adaptation cycle $n = 0$ to N to obtain the following equation after cancelling common terms:

$$\mu^{-1}\|\tilde{\mathbf{w}}(N)\|^2 - \sum_{n=0}^{N} v^2(n) = \mu^{-1}\|\tilde{\mathbf{w}}(0)\|^2 - \sum_{n-0}^{N} (\tilde{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n))^2 - \sum_{n=0}^{N} (1 - \mu\|\mathbf{u}(n)\|^2)(d(n) - \tilde{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n))^2.$$

(e) Next, reformulate the equation under part (d) so as to take the form of the rational function

$$\frac{\left(\mu^{-1}\|\tilde{\mathbf{w}}(N)\|^2 + \sum_{n=0}^{N} (\tilde{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n))^2 + \sum_{n-0}^{N} (1 - \mu\|\mathbf{u}(n)\|^2)(d(n) - \tilde{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n))^2\right)}{\mu^{-1}\|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=0}^{N} v^2(n)} = 1.$$

(f) From the standpoint of robustness, the only term that matters in the numerator of the equation in part (e) is the second term. Explain why.

With the answer to the question at hand and recognizing the condition imposed on the step-size parameter, μ, proceed on to express the prescribed solution to the problem namely,

$$\frac{\sum_{n=0}^{N} \xi_u^2(n)}{\mu^{-1}\|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=0}^{N} v^2(n)} \leq 1.$$

2. In this second problem, we reverse direction. Specifically, the starting point is described by the inequality

$$\frac{\sum_{n=0}^{i} (\hat{d}(n) - \mathbf{w}^{\mathrm{T}}\mathbf{u}(n))^2}{\mu^{-1}\mathbf{w}^{\mathrm{T}}\mathbf{w} + \sum_{n=0}^{i-1} v^2(n)} \leq 1,$$

where, in the numerator, i denotes an arbitrary instant of time. The requirement is to show that for this inequality to hold, the step-size parameter, μ, must satisfy the following condition:

$$\mu \leq \frac{1}{\mathbf{u}^{\mathrm{T}}(i)\mathbf{u}(i)} \quad \text{for all } i.$$

(a) Show that the above-mentioned inequality is equivalently expressed as follows:

$$\mu^{-1}\mathbf{w}^{\mathrm{T}}\mathbf{w} + \sum_{n=0}^{i-1} (d(n) - \mathbf{w}^{\mathrm{T}}\mathbf{u}(n))^2 - \sum_{n=0}^{i} (\hat{d}(n) - \mathbf{w}^{\mathrm{T}}\mathbf{u}(n))^2 \geq 0$$

for all values of $\mathbf{w}$.

(b) The expression on the left-hand side of the inequality under (a) is an *indefinite quadratic form*. Why?

For the quadratic form to be positive, it has to have a minimum over all $\mathbf{w}$. To this end, differentiate the inequality twice to obtain the following result:

$$\mu^{-1}\mathbf{I} - \mathbf{u}(i)\mathbf{u}^{\mathrm{T}}(i) \geq 0,$$

where $\mathbf{I}$ is the identity matrix.

(c) Recognizing that the outer product $\mathbf{u}(i)\mathbf{u}^{\mathrm{T}}(i)$ is a matrix of rank one, hence, show that the step-size parameter, μ, must satisfy the following condition:

$$\mu \leq \frac{1}{\mathbf{u}^{\mathrm{T}}(i)\mathbf{u}(i)} \quad \text{for all } i$$

to validate the inequality under part (b). With i being arbitrary, this is precisely the prescribed condition for the problem.

3. Reconfirm with reference to the robustness of the LMS algorithm that "the maximum energy gain for any algorithm, which corresponds to the 'worst case' disturbance, can never be less than unity."

4. To expand on the quadratic form in part (a) of Problem 2 being indefinite, differentiate this inequality once with respect to the unknown vector $\mathbf{w}$ to find the optimizing $\mathbf{w}$, as shown by

$$\mathbf{w} = [\mu\mathbf{I} - \mathbf{u}(i)\mathbf{u}^{\mathrm{T}}(i)]^{-1}\left(\sum_{n=0}^{i-1} e(n)\mathbf{u}(n) - \hat{d}(i)\mathbf{u}(i)\right),$$

where $e(n)$ is the error signal defined by

$$e(n) = d(n) - \hat{d}(n).$$

5. Explain in detail how the robustness of the RLS algorithm may be studied. Compute the bounds on the maximum energy gain from the input disturbances to the estimation errors.

6. Derive the condition for the members of both causal sequences to satisfy the Cauchy–Schwarz Inequality for computing the H^{∞} Norm.

7. In this problem, we study robustness of the *normalized LMS algorithm*. The procedure to be followed is similar to that described in Problem 1.

Specifically, for real-valued data, the update formula for the normalized algorithm is given by

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}\mathbf{u}(n)}{\|\mathbf{u}(n)\|^2}e(n),$$

where $\tilde{\mu}$ is a new step-size parameter.

(a) Show that the condition for robustness of the normalized algorithm is described by the inequality:

$$\tilde{\mu} < 1.$$

(b) As pointed out in Chapter 7, to guard against the potential possibility of the input vector $\mathbf{u}(n)$ assuming a value very close to zero, the update formula is modified as follows:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}\mathbf{u}(n)}{\varepsilon + \|\mathbf{u}(n)\|^2},$$

where ε is a small positive constant. How is the condition on the step-size parameter, $\tilde{\mu}$, modified due to the inclusion of ε?.

8. Based on the results derived in this chapter on robustness of the LMS and normalized LMS algorithms, and the corresponding ones derived in Chapters 6 and 7 based on statistical learning theory, we may categorize the conditions imposed on their step-size parameter, as summarized in Table P11.1.

(a) Which particular set, those based on robustness or statistical learning theory, is more stringent from a practical standpoint?

(b) Discuss practical implications of the answer in part (a).

TABLE P11.1

	Robustness	Statistical learning theory
LMS algorithm	$0 < \mu < \dfrac{1}{\|\mathbf{u}(n)\|^2}$	$0 < \mu < \dfrac{2}{\lambda_{\max}}$
Normalized LMS algorithm	$0 < \mu < 1$	$0 < \mu < 2$

9. Show the significance of the LMS algorithm to stochastic approximation by using Eq. (11.10).

10. Show that the analysis of the decision in favor of the LMS or RLS algorithm for linear adaptive filtering hinges on the application of interest and the environment in which it needs to operate.

CHAPTER 12

Finite-Precision Effects

From a practical perspective, the study of adaptive filters would be incomplete without some discussion of the effects of *quantization*, or *round-off, errors* that arise when such filters are implemented digitally.

The theory of adaptive filtering developed in previous chapters assumes the use of an *analog model* (i.e., a model with infinite precision) for the samples of input data as well as for the internal algorithmic calculations. This assumption is made in order to take advantage of well-understood continuous mathematics. Adaptive filter theory, however, cannot be applied to the construction of an adaptive filter directly; rather, it provides an *idealized mathematical framework* for such a construction. In particular, in a *digital* implementation of an adaptive filtering algorithm as ordinarily employed in practice, the input data and internal calculations are all quantized to a *finite precision* that is determined by design and cost considerations. Consequently, the quantization process has the effect of causing the performance of a digital implementation of the algorithm to deviate from its theoretic continuous value. The nature of this deviation is influenced by a combination of factors:

- Design details of the adaptive filtering algorithm.
- Degree of ill conditioning (i.e., the eigenvalue spread) in the underlying correlation matrix that characterizes the input data.
- Form of the numerical computation (fixed point or floating point) employed.

It is important for us to understand the numerical properties of adaptive filtering algorithms, as doing so would obviously help us in meeting design specifications. Moreover, the cost of a digital implementation of an algorithm is influenced by the *number of bits* (i.e., the precision) available for performing the numerical computations associated with the algorithm. Generally speaking, the cost of implementation increases with the number of bits employed; therefore, there is ample practical motivation for using the minimum number of bits possible.

We begin our study of the numerical properties of adaptive filtering algorithms by examining the sources of quantization error and the related issues of numerical stability and accuracy.

12.1 QUANTIZATION ERRORS

In the digital implementation of an adaptive filter, essentially two sources of quantization error arise:

1. *Analog-to-digital conversion.* Given that the input data are in analog form, we may use an analog-to-digital converter for their numerical representation. For our present discussion, we assume a quantization process with a *uniform step size* δ and a set of *quantizing levels* positioned at $0, \pm\delta \pm 2\delta, \ldots$. Figure 12.1 illustrates the input–output characteristic of a typical uniform quantizer. Consider a particular sample at the quantizer input, with an amplitude that lies in the range $i\delta - (\delta/2)$ to $i\delta + (\delta/2)$, where i is an integer (positive, negative, or zero) and $i\delta$ defines the *quantizer output*. Such a quantization process introduces a region of uncertainty of width δ, centered on $i\delta$. Let η denote the quantization error. Correspondingly, the quantizer input is $i\delta + \eta$, where $-\delta/2 \leq \eta \leq \delta/2$. When the quantization is fine enough (say, the number of quantizing levels is 64 or more) and the signal spectrum is sufficiently rich, the distortion produced by the quantizing process may be modeled as an additive independent source of white noise with zero mean and variance determined by the quantizer step size δ (Gray, 1990). It is customary to

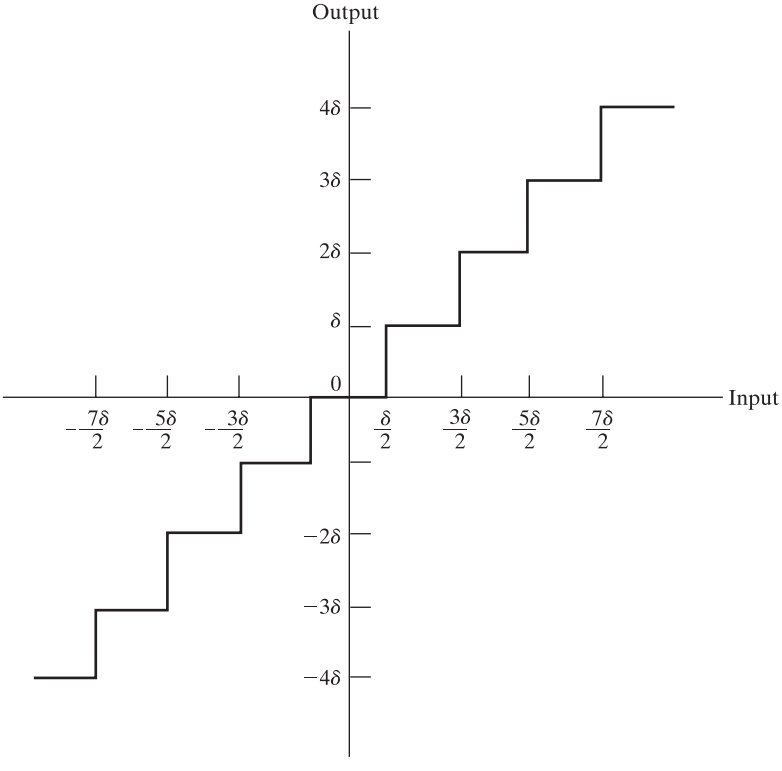


FIGURE 12.1 Input–output characteristic of a uniform quantizer.

assume that the quantization error η is *uniformly distributed* over the range from $-\delta/2$ to $\delta/2$. The variance of the quantization error is therefore given by

$$\begin{aligned}\sigma^2 &= \int_{-\delta/2}^{\delta/2} \frac{1}{\delta}\eta^2 \mathrm{d}\eta \\ &= \frac{\delta^2}{12}.\end{aligned} \tag{12.1}$$

We assume that the quantizer input is properly scaled, so that it lies inside the interval $(-1, +1]$. With each quantizing level represented by B bits plus a sign, the quantizer step size is

$$\delta = 2^{-B}. \tag{12.2}$$

Substituting Eq. (12.2) into Eq. (12.1), we find that the quantization error resulting from the digital representation of input analog data has the variance

$$\sigma^2 = \frac{2^{-2B}}{12}. \tag{12.3}$$

2. *Finite-wordlength arithmetic.* In a digital machine, a finite wordlength is commonly used to store the result of internal arithmetic calculations. Assuming that *no* overflow takes place during the course of computation, additions do not introduce any error (if fixed-point arithmetic is used), whereas each multiplication introduces an error after the product is quantized. The statistical characterization of finite-wordlength arithmetic errors may be quite different from that of analog-to-digital conversion errors. Finite-wordlength arithmetic errors may have a *nonzero mean*, which results from either rounding off or truncating the output of a multiplier so as to match the prescribed wordlength.

The presence of finite-wordlength arithmetic raises serious concern in the digital implementation of an adaptive filter, particularly when the tap weights (coefficients) of the filter are updated on a continuous basis. The digital version of the filter exhibits a specific *response*, or *propagation*, to such errors, causing its performance to deviate from the ideal (i.e., infinite-precision) form of the filter. Indeed, it is possible for the deviation to be of a catastrophic nature, in the sense that the errors resulting from the use of finite-precision arithmetic may accumulate without bound. If such a situation is allowed to persist, the filter is ultimately driven into an overflow condition, and the algorithm is said to be *numerically unstable*. Clearly, for an adaptive filter to be of practical value, it has to be numerically stable. An adaptive filter is said to be *numerically stable* if the use of finite-precision arithmetic results in deviations from the infinite-precision form of the filter that are *bounded*. It is important to recognize that numerical stability is an inherent characteristic of an adaptive filter. In other words, if an adaptive filter is numerically unstable, then increasing the number of bits used in a digital implementation of the filter will not make that implementation stable.

Another issue that requires attention in the digital implementation of an adaptive filtering algorithm is *numerical accuracy*. Unlike numerical stability, however, numerical

accuracy of an adaptive filter is determined by the number of bits used to implement internal calculations of the filter. The larger the number of bits used, the smaller is the deviation from the ideal performance, and the more accurate is the digital implementation of the filter. In practical terms, it is meaningful to speak of the numerical accuracy of an adaptive filter only if the filter is numerically stable.

For the remainder of this chapter, we discuss the numerical properties of adaptive filtering algorithms and related issues. We begin with the least-mean-square (LMS) algorithm and then move on to recursive least-squares (RLS) algorithms.

12.2 LEAST-MEAN-SQUARE (LMS) ALGORITHM

To simplify the discussion of finite-precision effects on the performance of the LMS algorithm,[1] we will depart from the practice followed in previous chapters and assume that the input data, and therefore the filter coefficients, are all *real valued*. This assumption, made merely for convenience of presentation, will in no way affect validity of the findings presented herein.

A block diagram of the *finite-precision LMS algorithm* is depicted in Fig. 12.2. Each of the blocks (operators) labeled Q represents a *quantizer*. Each Q introduces a *quantization*, or *round-off, error* of its own. We may describe the input–output relations of the quantizers operating in the figure as follows:

1. For the input quantizer connected to $\mathbf{u}(n)$, we have

$$\begin{aligned}\mathbf{u}_q(n) &= Q[\mathbf{u}(n)] \\ &= \mathbf{u}(n) + \boldsymbol{\eta}_u(n),\end{aligned} \tag{12.4}$$

   where $\boldsymbol{\eta}_u(n)$ is the *input quantization error vector*.

2. For the quantizer connected to the desired response $d(n)$, we have

$$\begin{aligned}d_q(n) &= Q[d(n)] \\ &= d(n) + \eta_d(n),\end{aligned} \tag{12.5}$$

   where $\eta_d(n)$ is the *desired-response quantization error*.

3. For the quantized tap-weight vector $\hat{\mathbf{w}}_q(n)$, we write

$$\begin{aligned}\hat{\mathbf{w}}_q(n) &= Q[\hat{\mathbf{w}}(n)] \\ &= \hat{\mathbf{w}}(n) + \Delta\hat{\mathbf{w}}(n),\end{aligned} \tag{12.6}$$

[1]The first treatment of finite-precision effects in the LMS algorithm was presented by Gitlin et al. (1973). Subsequently, more detailed treatments of these effects were presented by Weiss and Mitra (1979), Caraiscos and Liu (1984), and Alexander (1987). The paper by Caraiscos and Liu considers steady-state conditions, whereas the paper by Alexander is broader in scope, in that it considers transient conditions. The problem of finite-precision effects in the LMS algorithm is also discussed in Cioffi (1987) and in Sherwood and Bershad (1987). Another problem encountered in the practical use of the LMS algorithm is parameter drift, which is discussed in detail in Sethares et al. (1986). The material presented in Section 12.2 is very much influenced by the contents of these papers. In our presentation, we assume the use of *fixed-point arithmetic*. Error analysis of the LMS algorithm for *floating-point arithmetic* is discussed in Caraiscos and Liu (1984).

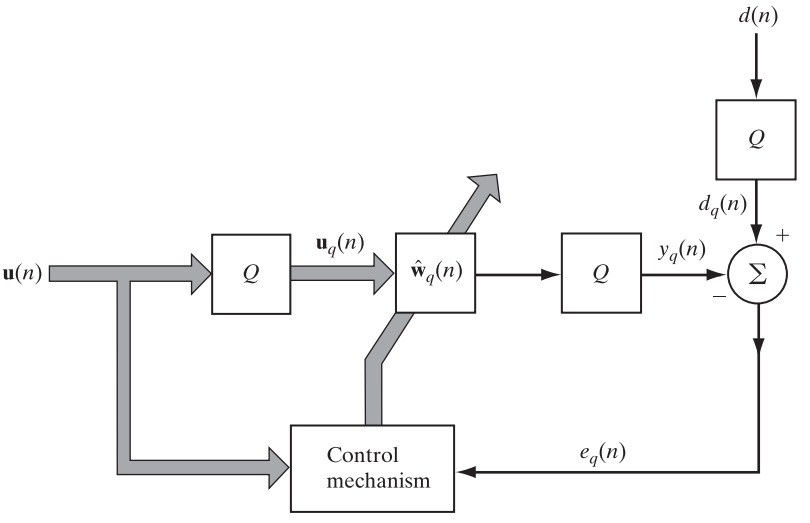


FIGURE 12.2 Block diagram representation of the finite-precision form of the LMS algorithm.

where $\hat{\mathbf{w}}(n)$ is the tap-weight vector in the infinite-precision LMS algorithm and $\Delta\hat{\mathbf{w}}(n)$ is the *tap-weight error vector* resulting from quantization.

4. For the quantizer connected to the output of the finite-duration impulse response (FIR) filter represented by the quantized tap-weight vector $\hat{\mathbf{w}}_q(n)$, we write

$$\begin{aligned} y_q(n) &= Q[\mathbf{u}_q^{\mathrm{T}}(n)\hat{\mathbf{w}}_q(n)] \\ &= \mathbf{u}_q^{\mathrm{T}}(n)\hat{\mathbf{w}}_q(n) + \eta_y(n), \end{aligned} \tag{12.7}$$

where $\eta_y(n)$ is the *filtered-output quantization error* and the superscript T denotes *transposition*.

The finite-precision LMS algorithm is described by the pair of relations

$$e_q(n) = d_q(n) - y_q(n) \tag{12.8}$$

and

$$\hat{\mathbf{w}}_q(n+1) = \hat{\mathbf{w}}_q(n) + Q[\mu e_q(n)\mathbf{u}_q(n)], \tag{12.9}$$

where $y_q(n)$ is as defined in Eq. (12.7). The quantizing operation indicated on the right-hand side of Eq. (12.9) is not shown explicitly in Fig. 12.2; nevertheless, it is basic to the operation of the finite-precision LMS algorithm. The use of Eq. (12.9) has the following practical implication: The product $\mu e_q(n)\mathbf{u}_q(n)$, representing a scaled version of the gradient vector estimate, is quantized *before* being added to the contents of the *tap-weight accumulator*. Because of hardware constraints, this form of digital implementation is preferred to the alternative method of operating the tap-weight accumulator in double-precision mode and then quantizing the tap weight to single precision at the accumulator output.

In a statistical analysis of the finite-precision LMS algorithm, it is customary to make the following assumptions:

1. The input data are properly scaled so as to *prevent overflow* of the elements of the quantized tap-weight vector $\hat{\mathbf{w}}_q(n)$ and the quantized output $y_q(n)$ during filtering.
2. Each data sample is represented by B_D bits plus a sign, and each tap weight is represented by B_W bits plus a sign. Thus, the quantization error associated with a B_D-plus-sign bit number (i.e., a sample datum) has the variance

$$\sigma_D^2 = \frac{2^{-2B_D}}{12}. \tag{12.10}$$

   Similarly, the quantization error associated with a B_W-plus-sign bit number (i.e., the tap weight) has the variance

$$\sigma_W^2 = \frac{2^{-2B_W}}{12}. \tag{12.11}$$

3. The elements of the input quantization error vector $\boldsymbol{\eta}_u(n)$ and the desired-response quantization error $\eta_d(n)$ are *white-noise* sequences, independent of the signals and independent from each other. Moreover, they have zero mean and variance σ_D^2.
4. The output quantization error $\eta_y(n)$ is a white-noise sequence, independent of the input signals and other quantization errors. It has a mean of zero and a variance equal to $c\sigma_D^2$, where c is a constant that depends on the way in which the inner product $\mathbf{u}_q^{\mathrm{T}}(n)\hat{\mathbf{w}}_q(n)$ is computed. If the individual scalar products in $\mathbf{u}_q^{\mathrm{T}}(n)\hat{\mathbf{w}}_q(n)$ are all computed without quantization and then summed, and if the final result is quantized in B_D bits plus a sign, then the constant c is unity, and the variance of $\eta_y(n)$ is σ_D^2, as defined in Eq. (12.10). If, in contrast, the individual scalar products in $\mathbf{u}^{\mathrm{T}}(n)\hat{\mathbf{w}}_q(n)$ are quantized and then summed, the constant c is M, and the variance of $\eta_y(n)$ is $M\sigma_D^2$, where M is the number of taps in the FIR filter implementation of the LMS algorithm.
5. The small step-size theory of Section 6.4, dealing with the infinite-precision LMS algorithm, is invoked.

Total Output Mean-Square Error

The filtered output $y_q(n)$ produced by the finite-precision LMS algorithm presents a *quantized estimate* of the desired response. The *total output error* is therefore equal to the difference $d(n) - y_q(n)$. Using Eq. (12.7), we may express this error as

$$\begin{aligned} e_{\text{total}}(n) &= d(n) - y_q(n) \\ &= d(n) - \mathbf{u}_q^{\mathrm{T}}(n)\hat{\mathbf{w}}_q(n) - \eta_y(n). \end{aligned} \tag{12.12}$$

Substituting Eqs. (12.4) and (12.6) into Eq. (12.12) and ignoring all quantization error terms higher than first order, we get

$$e_{\text{total}}(n) = [d(n) - \mathbf{u}^{\mathrm{T}}(n)\hat{\mathbf{w}}(n)] - [\Delta\hat{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n) + \boldsymbol{\eta}_u^{\mathrm{T}}(n)\hat{\mathbf{w}}(n) + \eta_y(n)]. \tag{12.13}$$

The term inside the first set of square brackets on the right-hand side of Eq. (12.13) is the estimation error $e(n)$ in the infinite-precision LMS algorithm. The term inside the second set of square brackets is entirely due to quantization errors in the finite-precision LMS algorithm. Because of Assumptions 3 and 4 (i.e., the quantization errors $\boldsymbol{\eta}_u$ and η_y are independent of the input signals and of each other), the quantization error-related terms $\Delta\hat{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n)$ and $\eta_y(n)$ are uncorrelated with each other. Basically, for the same reason, the infinite-precision estimation error $e(n)$ is uncorrelated with both $\boldsymbol{\eta}_u^{\mathrm{T}}(n)\hat{\mathbf{w}}(n)$ and $\eta_y(n)$. By invoking the small step-size theory of Chapter 6, we may write

$$\mathbb{E}[e(n)\Delta\hat{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n)] = \mathbb{E}[\Delta\hat{\mathbf{w}}^{\mathrm{T}}(n)]\mathbb{E}[e(n)\mathbf{u}(n)].$$

Moreover, by invoking this same theory, we may show that the expectation $\mathbb{E}[\Delta\hat{\mathbf{w}}(n)]$ is zero. (See Problem 2.) Hence, $e(n)$ and $\Delta\hat{\mathbf{w}}^{\mathrm{T}}(n)u(n)$ are also uncorrelated. In other words, the infinite-precision estimation error $e(n)$ is uncorrelated with all three quantization-error-related terms—$\Delta\hat{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{u}(n)$, $\boldsymbol{\eta}_u^{\mathrm{T}}(n)\hat{\mathbf{w}}(n)$, and $\eta_y(n)$—in Eq. (12.13).

Using these observations and assuming that the step-size parameter μ is small, Caraiscos and Liu (1984) showed that the *total output mean-square error* produced in the finite-precision algorithm has the *steady-state* composition

$$\mathbb{E}[e_{\text{total}}^2(n)] = J_{\min}(1 + \mathcal{M}) + \xi_1(\sigma_w^2, \mu) + \xi_2(\sigma_D^2). \tag{12.14}$$

The first term on the right-hand side of Eq. (12.14), $J_{\min}(1 + \mathcal{M})$, is the mean-square error of the infinite-precision LMS algorithm. In particular, $J_{\min}$ is the *minimum mean-square error* of the optimum Wiener filter, and $\mathcal{M}$ is the *misadjustment* of the infinite-precision LMS algorithm. The second term, $\xi_1(\sigma_w^2, \mu)$, arises because of the error $\Delta\hat{\mathbf{w}}(n)$ in the quantized tap-weight vector $\hat{\mathbf{w}}_q(n)$. This contribution to the total output mean-square error is *inversely proportional to the step-size parameter* μ. The third term, $\xi_2(\sigma_D^2)$, arises because of two quantization errors: the error $\boldsymbol{\eta}_u(n)$ in the quantized input vector $\mathbf{u}_q(n)$ and the error $\eta_y(n)$ in the quantized filter output $y_q(n)$. However, unlike $\xi_1(\sigma_w^2, \mu)$, this final contribution to the total output mean-square error is, to a first order of approximation, independent of the step-size parameter μ.

From the infinite-precision statistical learning theory of the LMS algorithm presented in Chapter 6, we know that decreasing μ reduces the misadjustment $\mathcal{M}$ and thus leads to an improved performance of the algorithm. In contrast, the inverse dependence of the contribution $\xi_1(\sigma_w^2, \mu)$ on μ in Eq. (12.14) indicates that decreasing μ has the effect of increasing the deviation from infinite-precision performance. In practice, therefore, the step-size parameter μ may be decreased only to a level at which the degrading effects of quantization errors in the tap weights of the finite-precision LMS algorithm become significant.

Since the misadjustment $\mathcal{M}$ decreases with μ and the contribution $\xi_1(\sigma_w^2, \mu)$ increases with reduced μ, we may (in theory) find an optimum value of μ for which the total output mean-square error in Eq. (12.14) is minimized. However, it turns out that this minimization results in an optimum value μ_o for the step-size parameter μ that is too small to be of practical value. In other words, such a μ_o does not permit the LMS algorithm to converge completely. Indeed, Eq. (12.14) for calculating the total output mean-square error is valid only for a μ that is well in excess of μ_o. Such a choice of μ is

necessary to prevent the occurrence of a phenomenon known as stalling, described later in the section.

Deviations During the Convergence Period

Equation (12.14) describes the general structure of the total output mean-square error of the finite-precision LMS algorithm, assuming that the algorithm has reached a steady state. During the convergence period of the algorithm, however, the situation is more complicated.

A detailed treatment of the transient adaptation properties of the finite-precision LMS algorithm is presented in Alexander (1987). In particular, a general formula is derived for the tap-weight misadjustment, or *perturbation*, of the finite-precision LMS algorithm, which is then measured with respect to the tap-weight solution computed from the infinite-precision form of the algorithm. The tap-weight misadjustment is defined by

$$\mathcal{W}(n) = \mathbb{E}[\Delta\hat{\mathbf{w}}^{\mathrm{T}}(n)\Delta\hat{\mathbf{w}}(n)], \tag{12.15}$$

where the *tap-weight error vector* is itself defined by [see Eq. (12.6)]

$$\Delta\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}_q(n) - \hat{\mathbf{w}}(n). \tag{12.16}$$

The tap-weight vectors $\hat{\mathbf{w}}_q(n)$ and $\hat{\mathbf{w}}(n)$ refer to the finite-precision and infinite-precision forms of the LMS algorithm, respectively. To determine $\mathcal{W}(n)$, the weight update equation (12.9) is written as

$$\hat{\mathbf{w}}_q(n+1) = \hat{\mathbf{w}}_q(n) + \mu e_q(n)\mathbf{u}_q(n) + \boldsymbol{\eta}_w(n), \tag{12.17}$$

where $\boldsymbol{\eta}_\omega(n)$ is the *gradient quantization error vector*, which results from quantizing the product $\mu e_q(n)\mathbf{u}_q(n)$ that represents a scaled version of the gradient vector estimate. The individual elements of $\boldsymbol{\eta}_\omega(n)$ are assumed to be uncorrelated in time and with each other and also are assumed to have a common variance σ_w^2. For this assumption to be valid, the step-size parameter μ must be large enough to prevent stalling from occurring. (Stalling is described later in the section.)

Applying an orthogonal transformation to $\hat{\mathbf{w}}_q(n)$ in Eq. (12.17) in a manner similar to that described in Section 6.4, we may study the propagation characteristics of the tap-weight misadjustment $\mathcal{W}(n)$ during adaptation and during the steady state. Using what amounts to such an approach, Alexander (1987) derived the following important theoretic results, supported by computer simulation:

1. The tap weights in the LMS algorithm are the *most sensitive* of all parameters to quantization. For the case of *uncorrelated input data*, the variance σ_w^2 [that enters the statistical characterization of the tap-weight update equation (12.17)] is proportional to the reciprocal of the product $r(0)\mu$, where $r(0)$ is the average input power and μ is the step-size parameter. For the case of *correlated input data*, the variance σ_w^2 is proportional to the reciprocal of $\mu\lambda_{\min}$, where $\lambda_{\min}$ is the smallest eigenvalue of the correlation matrix $\mathbf{R}$ of the input data vector $\mathbf{u}(n)$.
2. For uncorrelated input data, the adaptation time constants of the tap-weight misadjustment $\mathcal{W}(n)$ are heavily dependent on the step-size parameter μ.

3. For correlated input data, the adaptation time constants of $\mathcal{W}(n)$ are heavily dependent on the interaction between μ and the minimum eigenvalue $\lambda_{\min}$.

From a design point of view, it is important to recognize that the step-size parameter μ should not be chosen too small—in spite of the infinite-precision theory of the LMS algorithm, which advocates a small value for μ. Moreover, the more ill conditioned the input process $u(n)$, the more pronounced the finite-precision effects in a digital implementation of the LMS algorithm will be.

Leaky LMS Algorithm

To further stabilize the digital implementation of the LMS algorithm, we may use a technique known as *leakage*.[2] Basically, leakage prevents overflow in a limited-precision environment by providing a compromise between minimizing the mean-square error and containing the energy in the impulse response of the adaptive filter. However, overflow is prevented only at the twin expenses of an increase in hardware cost and a degradation in performance, compared with the infinite-precision form of the traditional LMS algorithm.

In the leaky LMS algorithm, the cost function

$$J(n) = e^2(n) + \alpha\|\hat{\mathbf{w}}(n)\|^2, \tag{12.18}$$

where α is a positive control parameter that is minimized with respect to the tap-weight vector $\hat{\mathbf{w}}(n)$. The first term on the right-hand side of the equation is the squared estimation error, and the second term is the energy in the tap-weight vector $\hat{\mathbf{w}}(n)$. The minimization described herein (for real data) yields the following time update for the tap-weight vector (see Problem 5, Chapter 5):

$$\hat{\mathbf{w}}(n+1) = (1 - \mu\alpha)\hat{\mathbf{w}}(n) + \mu e(n)\mathbf{u}(n). \tag{12.19}$$

Here, α is a constant that satisfies the condition

$$0 \leq \alpha < \frac{1}{\mu}.$$

Except for the *leakage factor* $(1 - \mu\alpha)$ associated with the first term on the right-hand side of Eq. (12.19), the algorithm is of the same mathematical form as the traditional LMS algorithm.

Note that inclusion of the leakage factor $(1 - \mu\alpha)$ in Eq. (12.19) has the equivalent effect of adding a white-noise sequence of zero mean and variance α to the input

[2]Leakage may be viewed as a technique for increasing the robustness of an algorithm (Ioannou & Kokotovic, 1983; Ioannou, 1990). For a historical account of the leakage technique in the context of adaptive filtering, see Cioffi (1987). For discussions of the leakage LMS algorithm, see Widrow and Stearns (1985) and Cioffi (1987).

Unfortunately, the leaky LMS algorithm suffers from a degradation of performance due to the addition of bias to the weight estimates. Nascimento and Sayed (1999) address the two issues of bias and weight drift by proposing a modification to the LMS algorithm called the circular-leaky LMS algorithm. This new adaptive filtering algorithm solves the weight-drift problem without introducing bias into the weight estimates, yet the computational complexity of the algorithm is essentially the same as that of the traditional LMS algorithm.

process $u(n)$. This suggests another method for stabilizing a digital implementation of the LMS algorithm: A relatively weak white-noise sequence (of variance α), known as *dither*, may be added to the input process $u(n)$, and samples of the combination may then be used as tap inputs (Werner, 1983).

Stalling

Stalling, also called *lockup*, is a phenomenon that is not evident from Eq. (12.14), but that may arise in a digital implementation of the LMS algorithm. Stalling occurs when the gradient estimate is *not* sufficiently noisy. To be specific, a digital implementation of the LMS algorithm *stops adapting*, or *stalls*, whenever the correction term $\mu e_q(n)u_q(n-i)$ for the ith tap weight in the update equation (12.9) is smaller than the *least significant bit (LSB)* of the tap weight; mathematically (Gitlin et al., 1973),

$$|\mu e_q(n_0)u_q(n_0 - i)| \leq \text{LSB}, \tag{12.20}$$

where n_0 is the adaptation cycle at which the ith tap weight stops adapting. Suppose that the condition of Eq. (12.20) is first satisfied for the ith tap weight. Then, to a first order of approximation, we may replace $u_q\ (n_0 - i)$ by its *root-mean-square* (rms) value, A_{rms}. Accordingly, using this value in the equation, we get the following relation for the rms value of the quantized estimation error when adaptation in the digitally implemented LMS algorithm stops:

$$|e_q(n)| \leq \frac{\text{LSB}}{\mu A_{\text{rms}}} = e_D(\mu). \tag{12.21}$$

The quantity $e_D(\mu)$, defined on the right-hand side of Eq. (12.21), is called the *digital residual error*.

To prevent the algorithm from stalling due to digital effects, the digital residual error $e_D(\mu)$ must be made as small as possible. According to Eq. (12.21), this requirement may be satisfied in one of two ways:

1. The LSB may be reduced by picking a sufficiently large number of bits for the digital representation of each tap weight.
2. The step-size parameter μ may be made as large as possible, while still guaranteeing that the algorithm converges.

Another method of preventing stalling is to insert dither at the input of the quantizer that feeds the tap-weight accumulator (Sherwood & Bershad, 1987). Dither, consisting of a random sequence, essentially "linearizes" the quantizer. In other words, the addition of dither guarantees that the quantizer input is *noisy* enough for the gradient quantization error vector $\boldsymbol{\eta}_w$ to be again modeled as white noise (i.e., the elements of $\boldsymbol{\eta}_w$ are uncorrelated in time and with each other, and they have a common variance σ_w^2). When dither is used in the manner described here, it is desirable to minimize its effect on the overall operation of the LMS algorithm. This is commonly achieved by shaping the power spectrum of the dither so that it is effectively rejected by the algorithm at its output.

Parameter Drift

In addition to the numerical problems associated with the LMS algorithm, one other rather subtle problem is encountered in practical applications of the algorithm: Certain classes of input excitation can lead to *parameter drift*; that is, parameter estimates or tap weights in the LMS algorithm attain arbitrarily large values despite bounded inputs, bounded disturbances, and bounded estimation errors (Sethares et al., 1986). Although such an unbounded behavior may be unexpected, it is possible for the parameter estimates to drift to infinity while all the observable signals in the algorithm converge to zero. Parameter drift in the LMS algorithm may be viewed as a *hidden form of instability*, since the tap weights represent "internal" variables of the algorithm. Parameter drift may result in new numerical problems, increased sensitivity to unmodeled disturbances, and degraded long-term performance.

In order to appreciate the subtleties of the parameter drift problem, we need to introduce some new concepts relating to the parameter space. We therefore digress briefly from the issue at hand to do so.

A sequence of information-bearing tap-input vectors $\mathbf{u}(n)$ for varying adaptation cycle n may be used to partition the *real M-dimensional parameter space* $\mathbb{R}^M$ into orthogonal subspaces, where M is the number of tap weights (i.e., the available number of degrees of freedom). The aim of this partitioning is to convert an adaptive filtering algorithm (e.g., the LMS algorithm) into simpler subsystems and thereby provide a closer linkage between the transient behavior of the parameter estimates and the filter excitations. The partitioning we have in mind is depicted in Fig. 12.3. In particular, we may identify the following subspaces of $\mathbb{R}^M$:

1. *The unexcited subspace.* Let the M-by-1 vector $\mathbf{z}$ be any element of the parameter space $\mathbb{R}^M$ that satisfies two conditions:
   - The Euclidean norm of the vector $\mathbf{z}$ is unity; that is,
   $$\|\mathbf{z}\| = 1.$$
   - The vector $\mathbf{z}$ is orthogonal to the tap-input vector $\mathbf{u}(n)$ for all but a finite number of adaptation cycles n; that is,
   $$\mathbf{z}^{\mathrm{T}}\mathbf{u}(n) \neq 0 \qquad \text{only finitely often.} \tag{12.22}$$

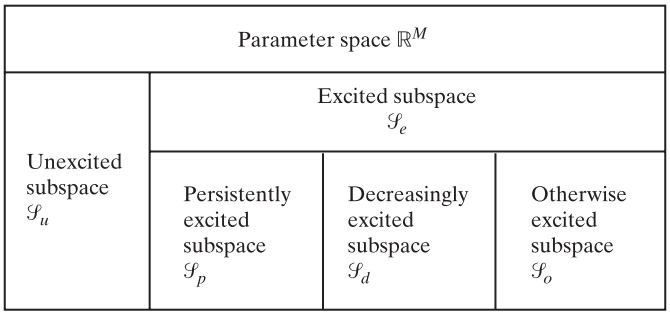


FIGURE 12.3 Decomposition of parameter space $\mathbb{R}^M$, based on excitation.

Let $\mathcal{S}_u$ denote the subspace of $\mathbb{R}^M$ that is spanned by the set of all such vectors $\mathbf{z}$. The subspace $\mathcal{S}_u$ is called the *unexcited subspace*, in the sense that it spans those *directions* in the parameter space $\mathbb{R}^M$ that are excited only *finitely often*.

2. *The excited subspace.* Let $\mathcal{S}_e$ denote the *orthogonal complement* of the unexcited subspace $\mathcal{S}_u$. Clearly, $\mathcal{S}_e$ is also a subspace of the parameter space $\mathbb{R}^M$, containing those directions in the parameter space $\mathbb{R}^M$ that are excited *infinitely often*. Thus, except for the null vector, every element $\mathbf{z}$ *belonging to the subspace* $\mathcal{S}_e$ satisfies the condition

$$\mathbf{z}^{\mathrm{T}}\mathbf{u}(n) \neq 0 \qquad \text{infinitely often.} \tag{12.23}$$

The subspace $\mathcal{S}_e$ is called the *excited subspace*.

The subspace $\mathcal{S}_e$ may itself be decomposed into three orthogonal subspaces of its own, depending on the effects of different types of excitation on the behavior of the adaptive filtering algorithm. The three subspaces of $\mathcal{S}_e$ may be identified as follows (Sethares et al., 1986):

- *The persistently excited subspace.* Let $\mathbf{z}$ be any vector of unit norm that lies in the excited subspace $\mathcal{S}_e$. For any positive integer m and any $\alpha > 0$, choose the vector $\mathbf{z}$ such that

$$\mathbf{z}^{\mathrm{T}}\mathbf{u}(i) > \alpha \qquad \text{for } n \leq i \leq n + m \text{ and for all but a finite number of } n. \tag{12.24}$$

  Given the integer m and the constant α, let $\mathcal{S}_p(m, \alpha)$ be the subspace spanned by all such vectors $\mathbf{z}$ that satisfy Eq. (12.24). Then there exist a finite m_0 and a positive α_0 for which the subspace $\mathcal{S}_p(m_0, \alpha_0)$ is *maximal*. In other words, $\mathcal{S}_p(m_0, \alpha_0)$ contains $\mathcal{S}_p(m, \alpha)$ for all $m > 0$ and for all $\alpha > 0$. The subspace $\mathcal{S}_p \equiv \mathcal{S}_p\,(m_0, \alpha_0)$ is called the *persistently excited subspace*, and m_0 is called the *interval of excitation*. For every direction $\mathbf{z}$ that lies in the persistently excited subspace $\mathcal{S}_p$, there is an excitation of level α_0 at least once in all but a finite number of intervals of length m_0. In the persistently excited subspace, we are therefore able to find a tap-input vector $\mathbf{u}(n)$ *rich* enough to excite all the internal modes that govern the transient behavior of the adaptive filtering algorithm being probed (Narendra & Annaswamy, 1989).
- *The subspace of decreasing excitation.* Consider a sequence $u(i)$ for which

$$\left(\sum_{i=1}^{\infty} |\,u(i)\,|^p\right)^{1/p} < \infty. \tag{12.25}$$

  Such a sequence is said to be an element of the *normed linear space* $\mathbb{L}^p$ for $1 < p < \infty$. The norm of this new space is defined by

$$\|\mathbf{u}\|_p = \left(\sum_{i=1}^{\infty} |\,u(i)\,|^p\right)^{1/p}. \tag{12.26}$$

  Note that if the sequence $u(i)$ is an element of $\mathbb{L}^p$ for $1 < p < \infty$, then

$$\lim_{n \to \infty} u(n) = 0. \tag{12.27}$$

Let $\mathbf{z}$ be any unit-norm vector that lies in the excited subspace $\mathcal{S}_e$ such that, for $1 < p < \infty$, the sequence $\mathbf{z}^T\mathbf{u}(n)$ lies in the normed linear space $\mathbb{L}^p$. Let $\mathcal{S}_d$ be the subspace that is spanned by all such vectors $\mathbf{z}$. The subspace $\mathcal{S}_d$ is called the *subspace of decreasing excitation*, in the sense that each direction of $\mathcal{S}_d$ is decreasingly excited. For any vector $\mathbf{z} \neq \mathbf{0}$, the two conditions

$$|\mathbf{z}^T\mathbf{u}(n)| = \alpha > 0 \qquad \text{infinitely often}$$

and

$$\lim_{n\to\infty} \mathbf{z}^T\mathbf{u}(n) = 0$$

cannot be satisfied simultaneously. In actuality, we find that the subspace of decreasing excitation, $\mathcal{S}_d$, is orthogonal to the subspace of persistent excitation, $\mathcal{S}_p$.

- *The otherwise excited subspace.* Let $\mathcal{S}_p \cup \mathcal{S}_d$ denote the *union* of the persistently excited subspace $\mathcal{S}_p$ and the subspace of decreasing excitation, $\mathcal{S}_d$. Let $\mathcal{S}_o$ denote the orthogonal complement of $\mathcal{S}_p \cup \mathcal{S}_d$ that lies in the excited subspace $\mathcal{S}_e$. The subspace $\mathcal{S}_o$ is called the *otherwise excited subspace*. Any vector that lies in the subspace $\mathcal{S}_o$ is not unexciting, not persistently exciting, and not in the normal linear space $\mathbb{L}^p$ for any finite p. An example of such a signal is the sequence

$$\mathbf{z}^T\mathbf{u}(n) = \frac{1}{\ln(1+n)}, \qquad n = 1, 2, \ldots. \tag{12.28}$$

Returning to our discussion of the parameter drift problem in the LMS algorithm, we find that, for bounded excitations and bounded disturbances in the case of unexcited and persistently exciting subspaces, the parameter estimates resulting from the application of the LMS algorithm are indeed bounded. However, in the decreasing and otherwise excited subspaces, parameter drift may occur (Sethares et al., 1986). A common method of counteracting the parameter drift problem in the LMS algorithm is to introduce leakage into the tap-weight update equation of the algorithm. Indeed, that is another reason for using the leaky LMS algorithm that was described previously.

12.3 RECURSIVE LEAST-SQUARES (RLS) ALGORITHM

The RLS algorithm offers an alternative to the LMS algorithm as a tool for the solution of linear adaptive filtering problems. From the discussion presented in Chapter 10, we know that the RLS algorithm is characterized by a fast rate of convergence that is relatively insensitive to eigenvalue spread of the underlying correlation matrix of the input data and by a negligible misadjustment (zero for a stationary environment without disturbances). Moreover, although the algorithm is computationally demanding (in the sense that its computational complexity is on the order of M^2, where M is the dimension of the tap-weight vector), its mathematical formulation, and therefore its implementation, is relatively simple. However, there is a numerical instability problem to be considered when the RLS algorithm is implemented in finite-precision arithmetic.

Basically, the *numerical instability*, or *explosive divergence*, of the RLS algorithm is of a nature similar to that experienced in Kalman filtering, of which the RLS algorithm is a special case; Kalman filtering is studied in Chapter 14. Indeed, the problem may be traced to the fact that the time-updated matrix $\mathbf{P}(n)$ in the Riccati equation is computed as the difference between two nonnegative definite matrices, as indicated in Eq. (10.19). Accordingly, the algorithm diverges explosively when the matrix $\mathbf{P}(n)$ loses the property of positive definiteness or Hermitian symmetry. This is precisely what happens in the usual formulation of the RLS algorithm described in Table 10.1 (Verhaegen, 1989).

How, then, can the RLS algorithm be formulated so that the Hermitian symmetry of the matrix $\mathbf{P}(n)$ is preserved despite the presence of numerical errors? For obvious practical reasons, it would also be satisfying if the solution to this fundamental problem could be attained in a computationally efficient manner. With these issues in mind, we present in Table 12.1 a particular version of the RLS algorithm from Yang (1994) that describes a computationally efficient procedure[3] for preserving the Hermitian symmetry of $\mathbf{P}(n)$ by design. (When reading Table 12.1 and the rest of this chapter, the asterisk denotes *complex conjugation*, and the superscript H denotes *Hermitian transposition* [i.e., transposition combined with complex conjugation].) Computational efficiency of this algorithm is improved because the algorithm simply computes the upper or lower triangular part of the matrix $\mathbf{P}(n)$, as signified by the operator Tri $\{\ \}$, and then fills in the rest of the matrix to preserve Hermitian symmetry. Moreover, division by λ is replaced by multiplication with the precomputed value of λ^{-1}.

Error-Propagation Model

We now turn our attention to the error-propagation model.[4] According to the algorithm of Table 12.1, the recursions involved in the computation of the inverse correlation matrix $\mathbf{P}(n)$ proceed as follows:

$$\boldsymbol{\pi}(n) = \mathbf{P}(n-1)\mathbf{u}(n); \tag{12.29}$$

$$r(n) = \frac{1}{\lambda + \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\pi}(n)}; \tag{12.30}$$

$$\mathbf{k}(n) = r(n)\boldsymbol{\pi}(n); \tag{12.31}$$

$$\mathbf{P}(n) = \mathrm{Tri}\{\lambda^{-1}[\mathbf{P}(n-1) - \mathbf{k}(n)\boldsymbol{\pi}^{\mathrm{H}}(n)]\}. \tag{12.32}$$

[3]Verhaegen (1989) describes another symmetry-preserving version of the RLS algorithm. Verhaegen's version is less efficient than Yang's version in computational terms; however, both versions exhibit the same numerical behavior.

[4]The error-propagation model of RLS algorithms considered in this chapter examines the "linearized" round-off propagation mechanism and focuses on the property of exponential stability. In reality, however, round-off error propagation in an RLS algorithm is a nonlinear mechanism. Accordingly, the implication of using the linearized approach is *local* exponential stability of the RLS algorithm, with the result that there is no indication as to how small the accumulated error should be so that the ignored influence of the nonlinear (second-order) terms in Eqs. (12.35) and (12.37) does not destroy the stability of the filter. This nonlinear issue, which is of both theoretical and practical importance, is studied in Liavas and Regalia (1999). In their paper, bounds are derived on the word-length requirements to ensure bounded-error accumulation and consistency in a finite-precision implementation of the traditional RLS algorithm. The bounds are expressed in terms of the exponential weighting factor and input signal conditioning.

TABLE 12.1 Summary of a Computationally Efficient Symmetry-Preserving Version of the RLS Algorithm

Initialize the algorithm by setting

$$\mathbf{P}(0) = \delta^{-1}\mathbf{I}, \qquad \delta = \text{ small regularization parameter}$$
$$\hat{\mathbf{w}}(0) = \mathbf{0}$$

For each adaptation cycle, $n = 1, 2, \ldots$, compute

$$\boldsymbol{\pi}(n) = \mathbf{P}(n-1)\mathbf{u}(n)$$
$$r(n) = \frac{1}{\lambda + \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\pi}(n)}$$
$$\mathbf{k}(n) = r(n)\boldsymbol{\pi}(n)$$
$$\xi(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n-1)\mathbf{u}(n)$$
$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n)$$
$$\mathbf{P}(n) = \mathrm{Tri}\{\lambda^{-1}[\mathbf{P}(n-1) - \mathbf{k}(n)\boldsymbol{\pi}^{\mathrm{H}}(n)]\}$$

In Eq. (12.32), λ is the exponential weighting factor. Now, consider the propagation of a *single* quantization error at adaptation cycle $n-1$ to subsequent recursions, under the assumption that no other quantization errors are made. In particular, let

$$\mathbf{P}_q(n-1) = \mathbf{P}(n-1) + \boldsymbol{\eta}_p(n-1), \tag{12.33}$$

where the *error matrix* $\boldsymbol{\eta}_p(n-1)$ arises from the quantization of $\mathbf{P}(n-1)$. The corresponding quantized value of $\boldsymbol{\pi}(n)$ is

$$\boldsymbol{\pi}_q(n) = \boldsymbol{\pi}(n) + \boldsymbol{\eta}_p(n-1)\mathbf{u}(n). \tag{12.34}$$

Let $r_q(n)$ denote the quantized value of $r(n)$. Then, using the defining equation (12.30), followed by the application of Eq. (12.34), we may write

$$\begin{aligned}
r_q(n) &= \frac{1}{\lambda + \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\pi}_q(n)} \\
&= \frac{1}{\lambda + \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\pi}(n) + \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\eta}_p(n-1)\mathbf{u}(n)} \\
&= \frac{1}{\lambda + \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\pi}(n)}\left(1 + \frac{\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\eta}_p(n-1)\mathbf{u}(n)}{\lambda + \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\pi}(n)}\right)^{-1} \\
&= \frac{1}{\lambda + \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\pi}(n)} - \frac{\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\eta}_p(n-1)\mathbf{u}(n)}{(\lambda + \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\pi}(n))^2} + O(\boldsymbol{\eta}_p^2) \\
&= r(n) - \frac{\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\eta}_p(n-1)\mathbf{u}(n)}{(\lambda + \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\pi}(n))^2} + O(\boldsymbol{\eta}_p^2),
\end{aligned} \tag{12.35}$$

where $O(\boldsymbol{\eta}_p^2)$ denotes the order of magnitude, $\|\boldsymbol{\eta}_p\|^2$.

In an ideal situation, the infinite-precision scalar quantity $r(n)$ is *nonnegative*, taking on values between zero and $1/\lambda$. On the other hand, if $\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\pi}(n)$ is small compared with λ, and if λ itself is small enough compared with unity, then, according to Eq. (12.35), in a finite-precision environment it is possible for the quantity $r_q(n)$ to take on a *negative* value larger in magnitude than $1/\lambda$. When this happens, the RLS algorithm exhibits explosive divergence (Bottomley & Alexander, 1989).[5]

The quantized value of the gain vector $\mathbf{k}(n)$ is written as

$$\begin{aligned}\mathbf{k}_q(n) &= r_q(n)\boldsymbol{\pi}_q(n)\\ &= \mathbf{k}(n) + \boldsymbol{\eta}_k(n),\end{aligned} \tag{12.36}$$

where

$$\boldsymbol{\eta}_k(n) = r(n)(\mathbf{I} - \mathbf{k}(n)\mathbf{u}^{\mathrm{H}}(n))\boldsymbol{\eta}_p(n-1)\mathbf{u}(n) + O(\boldsymbol{\eta}_p^2) \tag{12.37}$$

is the *gain vector quantization error*. Finally, using Eq. (12.32), we find that the quantization error incurred in computing the updated inverse-correlation matrix $\mathbf{P}(n)$ is

$$\boldsymbol{\eta}_p(n) = \lambda^{-1}(\mathbf{I} - \mathbf{k}(n)\mathbf{u}^{\mathrm{H}}(n))\boldsymbol{\eta}_p(n-1)(\mathbf{I} - \mathbf{k}(n)\mathbf{u}^{\mathrm{H}}(n))^{\mathrm{H}}, \tag{12.38}$$

where the term $O(\boldsymbol{\eta}_p^2)$ has been ignored.

On the basis of Eq. (12.38), it would be tempting to conclude that $\boldsymbol{\eta}_p^{\mathrm{H}}(n) = \boldsymbol{\eta}_p(n)$, and therefore that the RLS algorithm of Table 12.1 is *Hermitian-symmetry preserving*, if we can assume that the condition $\boldsymbol{\eta}_p^{\mathrm{H}}(n-1) = \boldsymbol{\eta}_p(n-1)$ holds at the previous adaptation cycle. We are justified in making this assertion by virtue of the fact there is *no* blowup in this formulation of the RLS algorithm, as is demonstrated in what follows. (It is also assumed that there is no stalling.)

Equation (12.38) defines the *error propagation mechanism* for the RLS algorithm summarized in Table 12.1 on the basis of a single quantization error in $\mathbf{P}(n-1)$. The matrix $\mathbf{I} - \mathbf{k}(n)\mathbf{u}^{\mathrm{H}}(n)$ in Eq. (12.38) plays a crucial role in the way in which the single quantization error $\boldsymbol{\eta}_p(n-1)$ propagates through the algorithm. Using the original definition given in Eq. (10.22) for the gain vector, namely,

$$\mathbf{k}(n) = \boldsymbol{\Phi}^{-1}(n)\mathbf{u}(n), \tag{12.39}$$

we may write

$$\mathbf{I} - \mathbf{k}(n)\mathbf{u}^{\mathrm{H}}(n) = \mathbf{I} - \boldsymbol{\Phi}^{-1}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n). \tag{12.40}$$

Next, from Eq. (10.12), we have

$$\boldsymbol{\Phi}(n) = \lambda\boldsymbol{\Phi}(n-1) + \mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n). \tag{12.41}$$

Multiplying both sides of Eq. (12.41) by the inverse matrix $\boldsymbol{\Phi}^{-1}(n)$ and rearranging terms, we get

$$\mathbf{I} - \boldsymbol{\Phi}^{-1}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n) = \lambda\boldsymbol{\Phi}^{-1}(n)\boldsymbol{\Phi}(n-1). \tag{12.42}$$

[5]According to Bottomley and Alexander (1989), evolution of the quantized term $r_q(n)$ provides a good indication of explosive divergence because this term grows large and then suddenly becomes negative.

Comparing Eqs. (12.40) and (12.42), we readily see that

$$\mathbf{I} - \mathbf{k}(n)\mathbf{u}^{\mathrm{H}}(n) = \lambda\boldsymbol{\Phi}^{-1}(n)\boldsymbol{\Phi}(n-1). \tag{12.43}$$

Suppose now we consider the effect of the quantization error $\boldsymbol{\eta}_p(n_0)$ induced at adaptation cycle $n_0 \leq n$. Then, when the RLS algorithm of Table 12.1 is used and the matrix $\mathbf{P}(n)$ remains Hermitian, according to the error-propagation model of Eq. (12.38), it follows that the effect of the quantization error $\eta_p(n_0)$ becomes modified at adaptation cycle n, yielding

$$\boldsymbol{\eta}_p(n) = \lambda^{-(n-n_0)}\boldsymbol{\varphi}(n, n_0)\boldsymbol{\eta}_p(n_0)\boldsymbol{\varphi}^{\mathrm{H}}(n, n_0), \qquad n \geq n_0, \tag{12.44}$$

where

$$\boldsymbol{\varphi}(n, n_0) = (\mathbf{I} - \mathbf{k}(n)\mathbf{u}^{\mathrm{H}}(n)) \cdots (\mathbf{I} - \mathbf{k}(n_0+1)\mathbf{u}^{\mathrm{H}}(n_0+1)) \tag{12.45}$$

is a *transition matrix*. The repeated use of Eq. (12.43) in Eq. (12.45) leads us to express the transition matrix in the equivalent form

$$\boldsymbol{\varphi}(n, n_0) = \lambda^{n-n_0}\boldsymbol{\Phi}^{-1}(n)\boldsymbol{\Phi}(n_0). \tag{12.46}$$

The correlation matrix is defined by [ignoring the regularization term in Eq. (10.8)]

$$\boldsymbol{\Phi}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)\mathbf{u}^{\mathrm{H}}(i). \tag{12.47}$$

On the basis of this definition, the tap-input vector $\mathbf{u}(n)$ is said to be *uniformly persistently exciting* for sufficiently large n if there exist a, b, and N such that $0 < a < b < \infty$ and the following condition is satisfied (Ljung & Ljung, 1985):

$$a\mathbf{I} \leq \boldsymbol{\Phi}(n) \leq b\mathbf{I} \qquad \text{for all } n \geq N. \tag{12.48}$$

The notation used in Eq. (12.48) is shorthand for saying that the matrix $\boldsymbol{\Phi}(n)$ is positive definite. The condition for persistent excitation guarantees not only the positive definiteness of $\boldsymbol{\Phi}(n)$ but also that its matrix norm is uniformly bounded for $n \geq N$; that is,

$$\|\boldsymbol{\Phi}^{-1}(n)\| \leq \frac{1}{a} \qquad \text{for } n > N. \tag{12.49}$$

Returning to the transition matrix $\boldsymbol{\varphi}(n, n_0)$ of Eq. (12.46) and invoking the *mutual consistency*[6] property of a matrix norm, we may write

$$\|\boldsymbol{\varphi}(n, n_0)\| \leq \lambda^{n-n_0}\|\boldsymbol{\Phi}^{-1}(n)\| \cdot \|\boldsymbol{\Phi}(n_0)\|. \tag{12.50}$$

Next, invoking Eq. (12.49), we may rewrite Eq. (12.50) as

$$\|\boldsymbol{\varphi}(n, n_0)\| \leq \frac{\lambda^{n-n_0}}{a}\|\boldsymbol{\Phi}(n_0)\|. \tag{12.51}$$

[6]Consider two matrices $\mathbf{A}$ and $\mathbf{B}$ of compatible dimensions. The mutual consistency property states that

$$\|\mathbf{AB}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{B}\|.$$

(See Property 7 of Section E.2 in Appendix E.)

Finally, we may use the error-propagation equation (12.44) to express the Euclidean norm of $\boldsymbol{\eta}_p(n)$ as

$$\|\boldsymbol{\eta}_p(n)\| \leq \lambda^{-(n-n_0)}\|\boldsymbol{\varphi}(n, n_0)\| \cdot \|\boldsymbol{\eta}_p(n-1)\| \cdot \|\boldsymbol{\varphi}^{\mathrm{H}}(n, n_0)\|,$$

which, in light of Eq. (12.51), may be rewritten as

$$\|\boldsymbol{\eta}_p(n)\| \leq \lambda^{n-n_0}M, \qquad n \geq n_0, \tag{12.52}$$

where

$$M = \frac{1}{a^2}\|\boldsymbol{\Phi}(n_0)\|^2\|\boldsymbol{\eta}_p(n-1)\| \tag{12.53}$$

is a positive number. Equation (12.53) states that the RLS algorithm of Table 12.1 is *exponentially stable* in the sense that a single quantization error $\boldsymbol{\eta}_p(n_0)$ occurring in the inverse correlation matrix $\mathbf{P}(n_0)$ at adaptation cycle n_0 decays exponentially, provided that $\lambda < 1$ (i.e., provided that the algorithm has *finite memory*).[7] In other words, the propagation of a single error through this formulation of the traditional RLS algorithm with finite memory is *contractive*. Computer simulations validating this result are presented in Verhaegen (1989).

Note, however, that the single-error propagation for the case of *growing memory* (i.e., $\lambda = 1$) is *not* contractive. The reason is that when $\lambda = 1$, neither $\boldsymbol{\varphi}(n, n_0) \leq \mathbf{I}$ nor $\|\boldsymbol{\varphi}(n, n_0)\| \leq 1$ holds, even if the input vector $\mathbf{u}(n)$ is persistently exciting. Consequently, the accumulation of numerical errors may cause the algorithm to be divergent (Yang, 1994). In an independent study, Slock and Kailath (1991) also pointed out that the error-propagation mechanism in the RLS algorithm with $\lambda = 1$ is unstable and of a random-walk type. Moreover, there is experimental evidence for this numerical divergence, which is reported in Ardalan and Alexander (1987).

Stalling

As with the LMS algorithm, a second form of divergence, referred to as *stalling*, occurs when the tap weights in the RLS algorithm stop adapting. In particular, stalling occurs when the quantized elements of the matrix $\mathbf{P}(n)$ become very small, such that multiplication by $\mathbf{P}(n)$ is equivalent to multiplication by a zero matrix (Bottomley & Alexander, 1989). Clearly, stalling may arise no matter how the RLS algorithm is implemented.

[7]The first rigorous proof that single-error propagation in the RLS algorithm is exponentially stable for $\lambda < 1$ was presented in Ljung and Ljung (1985). This was followed by a more detailed investigation by Verhaegen (1989). Reconfirmation that the error-propagation mechanism in the RLS algorithm is exponentially stable was subsequently presented in Slock and Kailath (1991) and Yang (1994).

Stalling is directly linked to the exponential weighting factor λ and the variance σ_u^2 of the input data $u(n)$. Assuming that λ is close to unity, we find, from the definition of the correlation matrix $\mathbf{\Phi}(n)$, that the expectation of $\mathbf{\Phi}(n)$ is given by

$$\mathbb{E}[\mathbf{\Phi}(n)] \approx \frac{\mathbf{R}}{1-\lambda} \qquad \text{for large } n. \tag{12.54}$$

[The justification for this approximate relation is deferred to Chapter 13; see Eq. (13.49).] For λ close to unity,

$$\mathbb{E}[\mathbf{P}(n)] = \mathbb{E}[\mathbf{\Phi}^{-1}(n)] \approx (\mathbb{E}[\mathbf{\Phi}(n)])^{-1}. \tag{12.55}$$

Hence, using Eq. (12.54) in Eq. (12.55), we get

$$\mathbb{E}[\mathbf{P}(n)] \approx (1-\lambda)\mathbf{R}^{-1} \qquad \text{for large } n, \tag{12.56}$$

where $\mathbf{R}^{-1}$ is the inverse of matrix $\mathbf{R}$. Assuming that the tap-input vector $\mathbf{u}(n)$ is drawn from a wide-sense stationary process with zero mean, we may write

$$\mathcal{R} = \frac{1}{\sigma_u^2}\mathbf{R}, \tag{12.57}$$

where $\mathcal{R}$ is a *normalized correlation matrix* with diagonal elements equal to unity and off-diagonal elements less than or equal to unity and σ_u^2 is the variance of an input data sample $u(n)$. We may therefore rewrite Eq. (12.56) as

$$\mathbb{E}[\mathbf{P}(n)] \approx \left(\frac{1-\lambda}{\sigma_u^2}\right)\mathcal{R}^{-1} \qquad \text{for large } n. \tag{12.58}$$

Equation (12.58) reveals that the RLS algorithm may stall if the exponential weighting factor λ is close to unity or if the input data variance σ_u^2 is large. Accordingly, we may prevent stalling of the RLS algorithm by using a sufficiently large number of accumulator bits in the computation of the inverse correlation matrix $\mathbf{P}(n)$, thereby adding complexity to the practical implementation of the algorithm.

12.4 SUMMARY AND DISCUSSION

In this chapter, we discussed the numerical stability of the LMS and RLS algorithms.

The LMS algorithm is *numerically robust*. When the algorithm is operating in a limited-precision environment, the point to note is that the step-size parameter μ may be decreased only to a level at which the degrading effects of round-off errors in the tap weights of the finite-precision LMS algorithm become significant. Moreover, a finite-precision implementation of the algorithm may be improved by incorporating *leakage* into it.

The RLS algorithm, on the other hand, is prone to *numerical instability*, or *explosive divergence*, which may be traced to the manner in which its correlation matrix $\mathbf{P}(n)$ is time-updated in Eq. (10.19). This equation involves calculating the difference

between two nonnegative definite matrices, with the result that $\mathbf{P}(n)$ may no longer be nonnegative definite, hence the occurrence of numerical instability. This deficiency may be ameliorated by exploiting the use of square-root filtering at the expense of increased computational complexity. (The topic of square-root filtering is discussed in Chapter 14 on Kalman filtering, of which the RLS algorithm is a special case.)

To summarize, when we are confronted with a requirement for the use of finite-precision arithmetic in linear adaptive filtering, the LMS algorithm generally is the preferred choice over the RLS algorithm for two reasons:

1. The LMS algorithm is model-independent, a characterstic inherited from the use of stochastic gradient method.
2. The LMS algorithm has simplified computational complexity.

PROBLEMS

1. Illustrate the assumptions in a statistical analysis of the finite-precision LMS algorithm.
2. Show that the total output mean-square error produced in the finite-precision algorithm has a steady-state composition.
3. How can a leaky LMS algorithm be prevented from stalling due to digital effects?
4. The error-propagation model of RLS algorithms considered in this chapter examines the "linearized" round-off propagation mechanism and focuses on the property of exponential stability. In reality, however, round-off error propagation in an RLS algorithm is a nonlinear mechanism. Explain.
5. **(a)** Justify the small step-size theory of Section 6.4, dealing with the infinite-precision LMS algorithm.
   (b) How can stalling be prevented in the RLS algorithm?
6. Suggest an appropriate method for stabilizing a digital implementation of the LMS algorithm whose leakage factor $(1 - \mu\alpha)$ in Eq. (12.19) has the equivalent effect of adding a white-noise sequence of zero mean and variance a to the input process $u(n)$.
7. Derive the formula to prevent the RLS algorithm from stalling due to digital effects by making the digital residual error $e_{\mathrm{D}}(\mu)$ as small as possible.

CHAPTER 13

Adaptation in Nonstationary Environments

In much of the material on linear adaptive filtering covered in previous chapters, we focused attention on the least-mean-square (LMS) and recursive least-square (RLS) algorithms. Specifically, we considered the *average behavior* of these two algorithms, operating in a stationary environment. In such an environment, the error-performance surface is fixed, and the essential requirement is to process the training data in a step-by-step fashion, seeking the minimum point of the surface and thereby assuring optimum or near-optimum performance. With the Wiener filter as the frame of reference, that minimum point is defined by the *Wiener solution*.

Unfortunately, many of the environments encountered in practice are *nonstationary*, for which the Wiener solution takes on a time-varying form. In such a real-world application, an adaptive filtering algorithm has an added task to perform:

> Track the continually time-varying position of the minimum point on the error-performance surface.

In other words, the algorithm is now required to continually track statistical variations of the environment, under the proviso that these variations are "slow enough" for algorithmic tracking to be practically feasible.

Tracking is a *steady-state phenomenon*, to be contrasted with convergence, which is a *transient phenomenon*. It follows that, for an adaptive filter to exercise its tracking capability, it must first pass from the transient mode to the steady-state mode of operation, and there must be provision for continuous adjustment of the free parameters of the filter. Moreover, we may state that, in general, the rate of convergence and the tracking capability are two different properties of the algorithm. In particular, an adaptive filtering algorithm with good convergence properties does *not* necessarily possess a fast tracking capability, and vice versa.

13.1 CAUSES AND CONSEQUENCES OF NONSTATIONARITY

Nonstationarity of the environment, in which an adaptive filtering algorithm operates, is attributed to one of two causes:

1. *The source supplying the desired response for the algorithm is not only noisy but also time varying.* For example, this kind of situation arises in a *system-identification*

problem, where, as mentioned, the system under study is not only noisy but also time varying. In this situation, the correlation matrix of the tap inputs of the adaptive filtering algorithm remains fixed (as in a stationary input), whereas the cross-correlation vector between the tap inputs and the desired response assumes a time-varying form.

2. *The stochastic source of tap inputs applied to the algorithm is nonstationary.* The second situation arises, for example, when the adaptive filtering algorithm is used to *equalize a time-varying channel*. This time, both the correlation matrix of the tap inputs and their cross-correlation vector with the desired response assume time-varying forms.

The tracking details of a time-varying system are therefore not only dependent on the type of adaptive filtering algorithm employed in the study but also problem specific.

To study adaptation in a nonstationary environment with emphasis on tracking, we may identify two fundamental issues:

Issue 1: *How do we choose between the LMS and RLS algorithms as the basis for tracking?*

Issue 2: *How do we automatically tune the adaptation parameter of the tracking algorithm to realize the best possible performance?*

Issue 1 is covered in the first part of the chapter, consisting of Sections 13.2 to 13.7, and Issue 2 is addressed in the second part of the chapter, consisting of Sections 13.8 to 13.11.

13.2 THE SYSTEM IDENTIFICATION PROBLEM

To study the tracking of an unknown time-varying system with system identification as the problem of interest, we will focus on a popular time-varying model for a nonstationary environment. The model is governed by two basic processes.

1. *First-order Markov process.* The unknown dynamic equations of the environment are modeled by a finite-duration impulse response (FIR) filter whose tap-weight vector $\mathbf{w}_o(n)$ (i.e., impulse response) undergoes a *first-order Markov process*, written in vector form as

$$\mathbf{w}_o(n+1) = a\mathbf{w}_o(n) + \boldsymbol{\omega}(n), \tag{13.1}$$

where a is a fixed parameter of the model and $\boldsymbol{\omega}(n)$ is the *process noise vector*, assumed to be of zero mean and correlation matrix $\mathbf{R}_\omega$. In physical terms, the tap-weight vector $\mathbf{w}_o(n)$ may be viewed as originating from the process noise $\boldsymbol{\omega}(n)$, whose individual elements are applied to a bank of one-pole low-pass filters. Each such filter has a transfer function equal to $1/(1 - az^{-1})$, where z^{-1} is the unit-delay operator. It is assumed that the value of parameter a is very close to unity. The significance of this assumption is that the bandwidth of the low-pass filters is very much smaller than the incoming data rate. Equivalently, we may say that many adaptation cycles of the Markov model are required to produce a significant change in the tap-weight vector $\mathbf{w}_o(n)$.

2. *Multiple regression.* The observable feature of the environment—the desired response, denoted by $d(n)$—is governed by the multiple linear regression model

$$d(n) = \mathbf{w}_o^{\mathrm{H}}(n)\mathbf{u}(n) + \nu(n), \tag{13.2}$$

where $\nu(n)$ is the *measurement noise*, assumed to be white with zero mean and variance σ_ν^2, and the superscript H denotes *Hermitian transposition* (i.e., transposition combined with complex conjugation). Thus, even though both the input vector $\mathbf{u}(n)$ and the noise $\nu(n)$ are stationary, the model output $d(n)$ is a *nonstationary* random process by virtue of the fact that $\mathbf{w}_o(n)$ varies with time. Herein lies the challenge posed to the adaptive filter.

Figure 13.1, depicting Eqs. (13.1) and (13.2), is referred to as the unknown dynamic model.

Henceforth in this chapter, we consider system identification as the adaptive filtering task of interest. Specifically, given the observable $d(n)$ as the desired response for an input vector $\mathbf{u}(n)$, the requirement is to design an adaptive filter that tracks statistical variations in the Markov model's impulse response vector $\mathbf{w}_o(n)$. This system identification problem is depicted in Fig. 13.2. The error signal involved in the adaptive process is defined by

$$\begin{aligned} e(n) &= d(n) - y(n) \\ &= \mathbf{w}_o^{\mathrm{H}}(n)\mathbf{u}(n) + \nu(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n), \end{aligned} \tag{13.3}$$

where $\hat{\mathbf{w}}(n)$ is the tap-weight vector of the adaptive filter, which is assumed to have both an FIR structure and the same number of taps, M, as the unknown dynamic model represented by $\mathbf{w}_o(n)$. The tap-weight vector $\mathbf{w}_o(n)$ represents the "target" to be tracked by the filter. Whenever $\hat{\mathbf{w}}(n)$ equals $\mathbf{w}_o(n)$, the minimum mean-square error produced by the adaptive filter equals the irreducible error variance σ_ν^2.

According to Fig. 13.2, the desired response $d(n)$ applied to the adaptive filter is the observable of the environment. Since the environment is time varying, the desired response is correspondingly nonstationary. Hence, with the correlation matrix of the tap

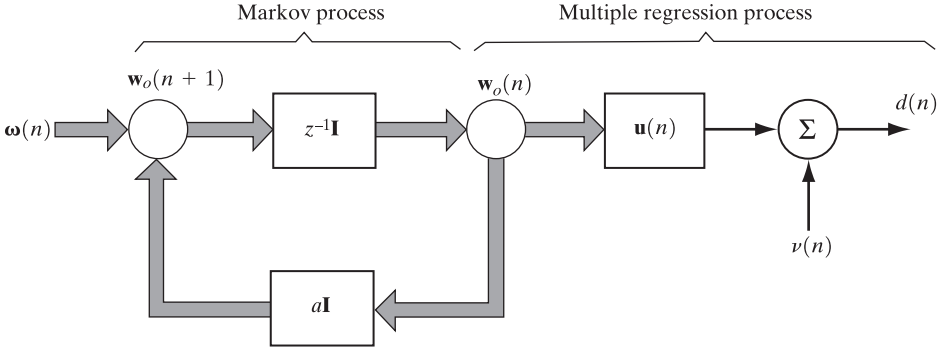


FIGURE 13.1 Linear dynamic model of a nonstationary environment.

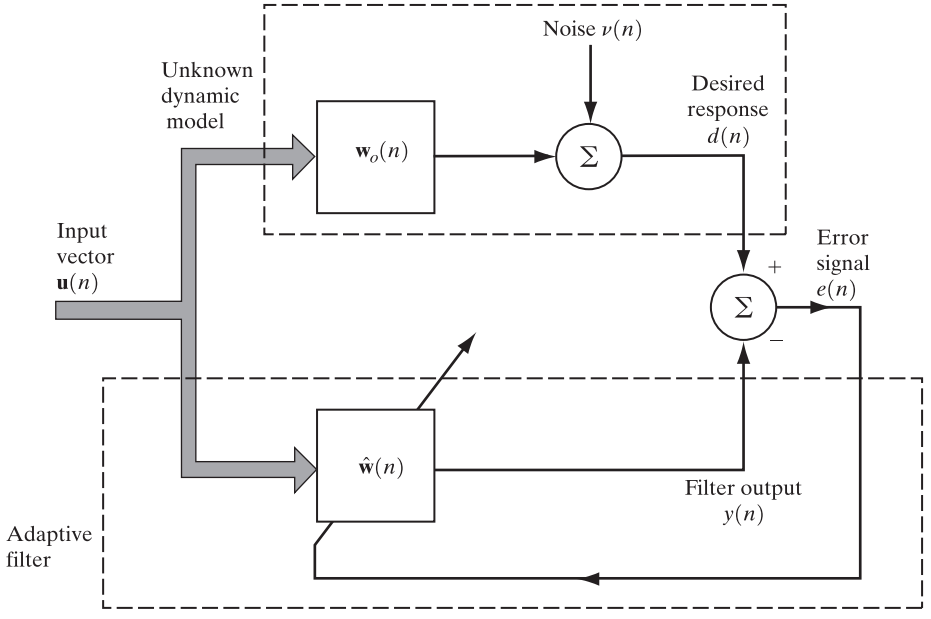


FIGURE 13.2 System identification using an adaptive filter; both $\mathbf{w}_o(n)$ and $\hat{\mathbf{w}}(n)$ are assumed to be of length M.

inputs having the fixed value $\mathbf{R}_u$, we find that the adaptive filter has a quadratic bowl-shaped error-performance surface whose position in weight space is in a permanent state of motion, hence the requirement to track the motion.

Assumptions

Typically, the time variations of the process noise vector $\boldsymbol{\omega}(n)$ in the unknown dynamic model of Fig. 13.1 are slow (i.e., bounded), which makes it possible for the adaptive FIR filter using the LMS algorithm or RLS algorithm to *track* the statistical variations in the dynamic behavior of the unknown environment.

To proceed with a tracking analysis of the LMS and RLS algorithms applied to the system identification problem of Fig. 13.2, the following conditions are assumed throughout the chapter:

1. The process noise $\boldsymbol{\omega}(n)$ is white with zero mean and correlation matrix $\mathbf{R}_\omega(n)$.
2. The measurement noise $\nu(n)$ is white with zero mean and variance σ_ν^2.
3. The process noise $\boldsymbol{\omega}(n)$ and measurement noise $\nu(n)$ are independent of each other.
4. The measurement matrix, denoted by $\mathbf{u}^{\mathrm{H}}(n)$, is independent of both the measurement noise $\nu(n)$ and the process noise $\boldsymbol{\omega}(n)$.

13.3 DEGREE OF NONSTATIONARITY

In order to provide a clear definition of the rather ambiguous concepts of "slow" and "fast" statistical variations of the model, we may introduce the notion of the *degree of nonstationarity* (Macchi, 1995). In the context of the unknown dynamic model described in Fig. 13.1, the degree of nonstationarity, denoted by α, is formally defined as the square root of the ratio of two quantities: the expectation of the squared magnitude of the inner product of the process noise $\boldsymbol{\omega}(n)$ and the vector $\mathbf{u}(n)$, and the average power of the measurement noise $\nu(n)$. That is,

$$\alpha = \left(\frac{\mathbb{E}[\,|\boldsymbol{\omega}^{\mathrm{H}}(n)\mathbf{u}(n)|^2]}{\mathbb{E}[\,|\nu(n)|^2]} \right)^{1/2}. \tag{13.4}$$

The degree of nonstationarity is therefore a characteristic of Fig. 13.1 acting alone and has nothing to do with the adaptive filter.

The numerator in Eq. (13.4) may be rewritten as follows in light of the assumption that $\boldsymbol{\omega}(n)$ is essentially independent of $\mathbf{u}(n)$:

$$\begin{aligned}
\mathbb{E}[\,|\boldsymbol{\omega}^{\mathrm{H}}(n)\mathbf{u}(n)|^2] &= \mathbb{E}[\boldsymbol{\omega}^{\mathrm{H}}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\omega}(n)] \\
&= \mathrm{tr}\,\{\mathbb{E}[\boldsymbol{\omega}^{\mathrm{H}}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\omega}(n)]\} \\
&= \mathbb{E}\,\{\mathrm{tr}\,[\boldsymbol{\omega}^{\mathrm{H}}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\omega}(n)]\} \\
&= \mathbb{E}\{\mathrm{tr}\,[\boldsymbol{\omega}(n)\boldsymbol{\omega}^{\mathrm{H}}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]\} \\
&= \mathrm{tr}\,\{\mathbb{E}[\boldsymbol{\omega}(n)\boldsymbol{\omega}^{\mathrm{H}}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]\} \\
&= \mathrm{tr}\,\{\mathbb{E}[\boldsymbol{\omega}(n)\boldsymbol{\omega}^{\mathrm{H}}(n)]\mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]\} \\
&= \mathrm{tr}\,[\mathbf{R}_{\omega}\mathbf{R}_{u}].
\end{aligned} \tag{13.5}$$

Here, tr $[\cdot]$ denotes the *trace* of the matrix enclosed inside the square brackets, $\mathbf{R}_u$ is the correlation matrix of the vector $\mathbf{u}(n)$, and $\mathbf{R}_\omega$ is the correlation matrix of the process noise vector $\boldsymbol{\omega}(n)$. The denominator in Eq. (13.4) is simply the variance σ_ν^2 of the zero-mean measurement noise $\nu(n)$. Accordingly, we may reformulate the degree of nonstationary for the Markov model of Fig. 13.1 simply as

$$\alpha = \frac{1}{\sigma_\nu}(\mathrm{tr}\,[\mathbf{R}_u\mathbf{R}_\omega])^{1/2} = \frac{1}{\sigma_\nu}(\mathrm{tr}\,[\mathbf{R}_\omega\mathbf{R}_u])^{1/2}. \tag{13.6}$$

The degree of nonstationarity, α, bears a useful relation to the misadjustment $\mathcal{M}$ of the adaptive filter. To bring out this relation, we first note that the minimum mean-square error $J_{\min}$ the adaptive filter in Fig. 13.2 can even attain is equal to the variance σ_ν^2 of the measurement noise $\nu(n)$. Next, we note that the best that the adaptive filter can ever do in tracking the time-varying system of Fig. 13.1 is to produce a weight-error vector $\boldsymbol{\varepsilon}(n)$ that is equal to the process noise vector $\boldsymbol{\omega}(n)$. Then, setting the correlation matrix of $\boldsymbol{\varepsilon}(n)$ equal to the correlation matrix of $\boldsymbol{\omega}(n)$, and recalling the definition of misadjustment as the ratio of the excess mean-square error to the minimum mean-square error $J_{\min} = \sigma_\nu^2$, we may use Eq. (6.100) to write

$$\mathcal{M} = \frac{J_{\text{ex}}(\infty)}{J_{\min}} \geq \frac{\text{tr}[\mathbf{R}_u \mathbf{R}_\omega]}{\sigma_\nu^2} = \alpha^2. \tag{13.7}$$

In other words, the square root of the misadjustment $\mathcal{M}$ places an upper bound on the degree of nonstationarity, hence the importance of α.

We shall have more to say on misadjustment as a measure of tracking assessment in the next section. For now, we may use Eq. (13.7) to make two noteworthy remarks:

1. For slow statistical variations of the environment, α is small. This means that it should be possible to build an adaptive filter that can track the unknown dynamic model of Fig. 13.1.
2. When the statistical variations of the environment are too fast, α may be greater than unity. In such a case, the misadjustment produced by the adaptive filter exceeds 100 percent, which means that there is no advantage to be gained in building an adaptive filter to solve the tracking problem: end of story.

13.4 CRITERIA FOR TRACKING ASSESSMENT

With the state of the unknown dynamic model in Fig. 13.1 denoted by $\mathbf{w}_o(n)$, and with the tap-weight vector of the adaptive FIR filter in Fig. 13.2 denoted by $\hat{\mathbf{w}}(n)$, we formally define the *weight-error vector* as

$$\boldsymbol{\varepsilon}(n) = \mathbf{w}_o(n) - \hat{\mathbf{w}}(n). \tag{13.8}$$

On the basis of $\boldsymbol{\varepsilon}(n)$, we may go on to define two figures of merit for assessing the tracking capability of the adaptive filter.

1. *Mean-Square Deviation*

A commonly used figure of merit for tracking assessment is the *mean-square deviation* (MSD) between the actual weight vector $\mathbf{w}_o(n)$ of the unknown dynamic model and the adjusted weight vector $\hat{\mathbf{w}}(n)$ of the adaptive filter, defined by

$$\begin{aligned} \mathcal{D}(n) &= \mathbb{E}[\|\mathbf{w}_o(n) - \hat{\mathbf{w}}(n)\|^2] \\ &= \mathbb{E}[\|\boldsymbol{\varepsilon}(n)\|^2], \end{aligned} \tag{13.9}$$

where the number of adaptation cycles, n, is assumed to be large enough for the adaptive filter's transient mode of operation to have finished. Following steps similar to those presented in Eq. (13.5), we may reformulate Eq. (13.9) into the equivalent form

$$\mathcal{D}(n) = \text{tr}[\mathbf{K}(n)], \tag{13.10}$$

where

$$\mathbf{K}(n) = \mathbb{E}[\boldsymbol{\varepsilon}(n)\boldsymbol{\varepsilon}^{\text{H}}(n)] \tag{13.11}$$

is the correlation matrix of the weight-error vector $\boldsymbol{\varepsilon}(n)$. Clearly, the mean-square deviation $\mathcal{D}(n)$ should be small for good tracking performance.

The weight-error vector may be expressed as the sum of two components:

$$\boldsymbol{\varepsilon}(n) = \boldsymbol{\varepsilon}_1(n) + \boldsymbol{\varepsilon}_2(n), \tag{13.12}$$

where

$$\boldsymbol{\varepsilon}_1(n) = \mathbb{E}[\hat{\mathbf{w}}(n)] - \mathbf{w}(n) \tag{13.13}$$

is the *weight vector noise*, in which $\mathbb{E}[\hat{\mathbf{w}}(n)]$ is the ensemble-average value of the tap-weight vector and

$$\boldsymbol{\varepsilon}_2(n) = \mathbf{w}_o(n) - \mathbb{E}[\hat{\mathbf{w}}(n)] \tag{13.14}$$

is the *weight vector lag*. Invoking the assumptions made in Section 13.2, we have

$$\mathbb{E}[\boldsymbol{\varepsilon}_1^{\mathrm{H}}(n)\boldsymbol{\varepsilon}_2(n)] = \mathbb{E}[\boldsymbol{\varepsilon}_2^{\mathrm{H}}(n)\boldsymbol{\varepsilon}_1(n)] = 0. \tag{13.15}$$

Accordingly, we may express the mean-square deviation as

$$\mathcal{D}(n) = \mathcal{D}_1(n) + \mathcal{D}_2(n). \tag{13.16}$$

The first term in Eq. (13.16), $\mathcal{D}_1(n)$, is called the *estimation variance* and is due to the weight vector noise $\boldsymbol{\varepsilon}_1(n)$; it is defined by

$$\mathcal{D}_1(n) = \mathbb{E}[\|\boldsymbol{\varepsilon}_1(n)\|^2]. \tag{13.17}$$

$\mathcal{D}_1(n)$ is always present, even in the stationary case. The second term, $\mathcal{D}_2(n)$, is called the *lag variance* and is due to the weight vector lag; it is defined by

$$\mathcal{D}_2(n) = \mathbb{E}[\|\boldsymbol{\varepsilon}_2(n)\|^2]. \tag{13.18}$$

The presence of $\mathcal{D}_2(n)$ is testimony to the nonstationary nature of the environment. The decomposition of the mean-square deviation $\mathcal{D}(n)$ into the estimation variance $\mathcal{D}_1(n)$ and the lag variance $\mathcal{D}_2(n)$, as described in Eq. (13.16), is called the *decoupling property* (Macchi, 1986a, b).

2. *Misadjustment*

Another commonly used figure of merit for assessing the tracking capability of an adaptive filter is the *misadjustment*, which is defined by

$$\mathcal{M}(n) = \frac{J_{\mathrm{ex}}(n)}{\sigma_\nu^2}, \tag{13.19}$$

where $J_{\mathrm{ex}}(n)$ is the excess (residual) mean-square error of the adaptive filter, measured with respect to the variance σ_ν^2 of the white noise component $\nu(n)$ at the output of the unknown dynamic model in Fig. 13.1. Here again, it is assumed that the number of adaptation cycles, n, is large enough for the transient period to have ended. We then find that, for good tracking performance, the misadjustment $\mathcal{M}(n)$ should be small compared with unity; that is, $J_{\mathrm{ex}}(n)$ is smaller than σ_ν^2 for all n.

As with the mean-square deviation, the excess mean-square error $J_{\mathrm{ex}}(n)$ may be expressed as the sum of two components, $J_{\mathrm{ex}\,1}(n)$ and $J_{\mathrm{ex}\,2}(n)$, by virtue of the assumptions

made in Section 13.2. The first component, $J_{\text{ex}\,1}(n)$, due to the weight vector noise $\boldsymbol{\varepsilon}_1(n)$, is called the *estimation noise*. The second component, $J_{\text{ex}\,2}(n)$, due to the weight vector lag $\boldsymbol{\varepsilon}_2(n)$, is called the *lag noise*. The presence of the term $J_{\text{ex}\,2}(n)$ is attributed directly to the nonstationary nature of the environment. Correspondingly, we may express the misadjustment as

$$\mathcal{M}(n) = \mathcal{M}_1(n) + \mathcal{M}_2(n), \tag{13.20}$$

where $\mathcal{M}_1(n) = J_{\text{ex}1}(n)/\sigma_\nu^2$ and $\mathcal{M}_2(n) = J_{\text{ex}2}(n)/\sigma_\nu^2$. Here, the first term, $\mathcal{M}_1(n)$, is called the *noise misadjustment*, and the second term, $\mathcal{M}_2(n)$, is called the *lag misadjustment*. Thus, the decoupling property is true for the misadjustment, too, in that the estimation noise and lag noise are decoupled in power.

In general, both figures of merit, $\mathcal{D}(n)$ and $\mathcal{M}(n)$, depend on the number of adaptation cycles, n. Moreover, they highlight different aspects of the tracking problem in a complementary way, as subsequent analysis will reveal.

13.5 TRACKING PERFORMANCE OF THE LMS ALGORITHM

To proceed with a study of the tracking problem, consider the system model of Fig. 13.2, in which the adaptive (FIR) filter is implemented using the LMS algorithm. According to this algorithm, the tap-weight vector of the adaptive filter is updated by the formula

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n), \tag{13.21}$$

where μ is the step-size parameter and the asterisk denotes *complex conjugation*. [See Eq. (5.6).] Substituting Eq. (13.3) for the error signal $e(n)$ into Eq. (13.21), we may reformulate the LMS algorithm in the expanded form

$$\hat{\mathbf{w}}(n+1) = [\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^{\text{H}}(n)]\hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)\mathbf{u}^{\text{H}}(n)\mathbf{w}_o(n) + \mu\mathbf{u}(n)\nu^*(n). \tag{13.22}$$

Next, using Eq. (13.1) for describing the first-order Markov model, Eq. (13.8) for defining the weight-error vector, and Eq. (13.22) just derived for the LMS algorithm, we may describe the steady-state evolution of the LMS algorithm in terms of the weight-error vector as follows:

$$\begin{aligned}\boldsymbol{\varepsilon}(n+1) &= \mathbf{w}_o(n+1) - \hat{\mathbf{w}}(n+1) \\ &= [\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^{\text{H}}(n)]\boldsymbol{\varepsilon}(n) - (1-a)\mathbf{w}_o(n) - \mu\mathbf{u}(n)\nu^*(n) + \boldsymbol{\omega}(n),\end{aligned} \tag{13.23}$$

where $\mathbf{I}$ is the identity matrix. The linear stochastic difference equation (13.23) provides a complete description of the LMS algorithm embedded in the system model of Fig. 13.2. A general tracking theory of the model described in Eq. (13.23), however, is yet to be developed. With this goal in mind, the usual procedure is to assume that the model parameter, a, in Eq. (13.1) is close enough to one to ignore the term $(1-a)\,\mathbf{w}_o(n)$; in so doing, we have in fact developed a tracking theory for a random-walk model (Macchi, 1995). Thus, setting $a = 1$, Eq. (13.23) simplfies to

$$\boldsymbol{\varepsilon}(n+1) = [\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^{\text{H}}(n)]\boldsymbol{\varepsilon}(n) - \mu\mathbf{u}(n)\nu^*(n) + \boldsymbol{\omega}(n). \tag{13.24}$$

Typically, the step-size parameter μ is assigned a small value in order to realize good tracking performance. Under this condition, we may solve Eq. (13.24) for the weight-error vector $\boldsymbol{\varepsilon}(n)$ by invoking the *direct-averaging method* (Kushner, 1984), which was discussed previously in Chapter 6. Specifically, we may state that, for small μ, the solution $\boldsymbol{\varepsilon}(n)$ to the linear stochastic difference equation (13.24) is "close" to the solution of another linear stochastic difference equation that is obtained by replacing the system matrix $[\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]$ with its ensemble average $(\mathbf{I} - \mu\mathbf{R}_u)$, where $\mathbf{R}_u$ is the correlation matrix of the input vector $\mathbf{u}(n)$. We may thus write the new stochastic difference equation as

$$\boldsymbol{\varepsilon}_0(n + 1) = (\mathbf{I} - \mu\mathbf{R}_u)\boldsymbol{\varepsilon}_0(n) - \mu\mathbf{u}(n)\nu^*(n) + \boldsymbol{\omega}(n), \tag{13.25}$$

where $\boldsymbol{\varepsilon}_0(n)$ is the small-step-size approximation to the weight-error vector $\boldsymbol{\varepsilon}(n)$.

Now we apply the unitary similarity transformation to the correlation matrix $\mathbf{R}_u$, given by

$$\mathbf{Q}^{\mathrm{H}}\mathbf{R}_u\mathbf{Q} = \boldsymbol{\Lambda}_u, \tag{13.26}$$

where $\boldsymbol{\Lambda}_u = \{\lambda_{u,k}\}_{k=1}^{M}$ is a diagonal matrix consisting of the eigenvalues of the correlation matrix and $\mathbf{Q}$ is a unitary matrix whose columns constitute an orthogonal set of eigenvectors associated with those eigenvalues.

Then, following a procedure similar to that described in Section 6.4 on statistical learning theory of the LMS algorithm, we may transform the stochastic difference equation (13.25) into a system of approximately decoupled difference equations, viz.,

$$v_k(n + 1) = (1 - \mu\lambda_{u,k})v_k(n) + \phi_k(n), \quad k = 1, 2, \ldots, M, \tag{13.27}$$

where the *natural mode* $v_k(n)$ is the kth component of the transformed vector

$$\boldsymbol{\nu}(n) = \mathbf{Q}^{\mathrm{H}}\boldsymbol{\varepsilon}_0(n) \tag{13.28}$$

and $\phi_k(n)$ is a *stochastic force* with zero mean, defined by

$$\phi_k(n) = -\mu\mathbf{q}_k^{\mathrm{H}}\mathbf{u}(n)\nu^*(n) + \mathbf{q}_k^{\mathrm{H}}\boldsymbol{\omega}(n), \tag{13.29}$$

where $\mathbf{q}_k$ is the eigenvector associated with the eigenvalue $\lambda_{u,k}$. Recognizing that the components, $\mathbf{u}(n)$ and $\boldsymbol{\omega}(n)$, are statistically independent, we may project them separately on the eigenvector $\mathbf{q}_k$ and thereby express the variance of $\phi_k(n)$ as

$$\sigma_{\phi_k}^2 \approx \mu^2\sigma_\nu^2\lambda_{u,k} + \lambda_{\omega,k}, \qquad k = 1, 2, \ldots, M. \tag{13.30}$$

where $\lambda_{\omega,k}$ is the eigenvalue of the process noise, $\boldsymbol{\omega}_n$, projected onto $\mathbf{q}_k$.

The system of natural modes described in Eq. (13.27) provides the mathematical framework for assessing the tracking performance of the LMS algorithm applied to the system identification problem of Fig. 13.2.[1]

[1]As remarked previously, tracking behavior of the LMS algorithm assumes that convergence of the algorithm has been completed. Under this condition, we see the following difference:

- In Section 6.4, the steady state of the algorithm involves eigenvalues of the correlation matrix $\mathbf{R}_u$, pertaining to the input vector.
- For the system-identification problem considered herein, steady state of the algorithm also involves eigenvalues of the correlation matrix $\mathbf{R}_\omega$, pertaining to the process noise vector.

This difference will become apparent in the next subsection.

Mean-Square Deviation

According to Eq. (6.92), the mean-square deviation of the LMS algorithm, under the assumption of a small step size, is defined by

$$\begin{aligned}\mathcal{D}(n) &\approx \mathbb{E}[\|\boldsymbol{\varepsilon}_0(n)\|^2] \\ &= \sum_{k=1}^{M} \mathbb{E}[|v_k(n)|^2],\end{aligned} \tag{13.31}$$

which, after a large number of adaptation cycles, n, takes the value

$$\mathcal{D}(n) \approx \frac{\mu}{2} M\sigma_\nu^2 + \frac{1}{2\mu}\sum_{k=1}^{M}\frac{\lambda_{\omega,k}}{\lambda_{u,k}}. \tag{13.32}$$

[The derivation of Eq. (13.32) is presented as Problem 5.] Equivalently, we may express the mean-square deviation as

$$\mathcal{D}(n) \approx \frac{\mu}{2} M\sigma_\nu^2 + \frac{1}{2\mu}\mathrm{tr}[\mathbf{R}_u^{-1}\mathbf{R}_\omega] \quad \text{for } n \text{ large}, \tag{13.33}$$

where $\mathrm{tr}[\cdot]$ denotes the trace operator.

The first term in Eq. (13.33), $\mu M\sigma_\nu^2/2$, is the estimation variance due to the measurement noise $\nu(n)$; this term varies *linearly* with the step-size parameter μ. The second term, $\mathrm{tr}[\mathbf{R}_u^{-1}\mathbf{R}_\omega]/2\mu$, is the lag variance due to the process noise vector $\boldsymbol{\omega}(n)$; this term varies *inversely* with the step-size parameter μ, thereby permitting a faster tracking speed.

Let μ_{opt} denote the optimum value of the step-size parameter for which the mean-square deviation attains its minimum value $\mathcal{D}_{\min}$. This optimum condition is realized when the estimation variance and lag variance contribute equally to the mean-square deviation. From Eq. (13.33), we readily find that

$$\mu_{\text{opt}} \approx \frac{1}{\sigma_\nu\sqrt{M}}(\mathrm{tr}[\mathbf{R}_u^{-1}\mathbf{R}_\omega])^{1/2} \tag{13.34}$$

and

$$\mathcal{D}_{\min} \approx \sigma_\nu\sqrt{M}(\mathrm{tr}[\mathbf{R}_u^{-1}\mathbf{R}_\omega])^{1/2}. \tag{13.35}$$

Misadjustment of the LMS Algorithm

To evaluate the misadjustment of the LMS algorithm for the system identification scenario described in Fig. 13.2, we use Eq. (6.91) to write

$$\mathcal{M} \approx \frac{1}{\sigma_\nu^2}\sum_{k=1}^{M}\lambda_{u,k}\mathbb{E}[|v_k(n)|^2], \tag{13.36}$$

which, after a large number of adaptation cycles, yields the approximate result,

$$\mathcal{M} \approx \frac{\mu}{2}\sum_{k=1}^{M}\lambda_{u,k} + \frac{1}{2\mu\sigma_\nu^2}\sum_{k=1}^{M}\lambda_{\omega,k}. \tag{13.37}$$

[The derivation of Eq. (13.37) is presented as Problem 6.] Equivalently, we may express the misadjustment of the LMS algorithm as

$$\mathcal{M} \approx \frac{\mu}{2}\text{tr}[\mathbf{R}_u] + \frac{1}{2\mu\sigma_v^2}\text{tr}[\mathbf{R}_\omega] \quad \text{for } n \text{ large}. \tag{13.38}$$

The first term, $\mu\text{tr}[\mathbf{R}_u]/2$, is the noise misadjustment caused by the measurement noise $v(n)$; this term is of the same form as in a stationary environment, which is not surprising. The second term, $\text{tr}[\mathbf{R}_{\boldsymbol{\omega}}]/2\mu\sigma_v^2$, is the lag misadjustment caused by the process noise vector $\boldsymbol{\omega}(n)$; this term is representative of nonstationarity in the environment.

The noise misadjustment varies *linearly* with the step-size parameter μ, whereas the lag misadjustment varies *inversely* with μ. The optimum value of the step-size parameter, μ_{opt}, for which the misadjustment attains its minimum value $\mathcal{M}_{\min}$, occurs when the estimation noise and lag noise have equal power. We thus readily find from Eq. (13.38) that

$$\mu_{\text{opt}} \approx \frac{1}{\sigma_v}\left(\frac{\text{tr}[\mathbf{R}_\omega]}{\text{tr}[\mathbf{R}_u]}\right)^{1/2} \tag{13.39}$$

and

$$\mathcal{M}_{\min} \approx \frac{1}{\sigma_v}\left(\text{tr}[\mathbf{R}_u]\text{tr}[\mathbf{R}_\omega]\right)^{1/2}. \tag{13.40}$$

Equations (13.34) and (13.39) indicate that, in general, optimization of the two figures of merit, the mean-square deviation and the misadjustment, leads to different values for the optimum setting of the step-size parameter μ. This should not be surprising since the two figures of merit emphasize different aspects of the tracking problem. Still, however the choice is made, it is presumed that the optimum μ satisfies the condition for convergence of the LMS algorithm in the mean-square sense, which should be in accord with Section 6.4.

13.6 TRACKING PERFORMANCE OF THE RLS ALGORITHM

Consider next the RLS algorithm used to implement the adaptive filter in the system model of Fig. 13.2. From Eqs. (10.22) and (10.25) of Chapter 10, we recall that the corresponding update equation for the weight vector of the adaptive FIR filter may be written in the form

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \boldsymbol{\Phi}^{-1}(n)\mathbf{u}(n)\xi^*(n), \tag{13.41}$$

where (ignoring the regularization term)

$$\boldsymbol{\Phi}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)\mathbf{u}^{\text{H}}(i) \tag{13.42}$$

is the correlation matrix of the input vector $\mathbf{u}(n)$ and

$$\xi(n) = d(n) - \hat{\mathbf{w}}^{\text{H}}(n-1)\mathbf{u}(n) \tag{13.43}$$

is the a priori estimation error. In Eq. (13.42), λ is the *exponential weighting factor* that lies in the range $0 < \lambda \leq 1$.

To accommodate the slight change in the notation for the weight vector in Eq. (13.41) compared to that in Eq. (13.21), we modify the first-order Markov model of Eq. (13.1) and the desired response $d(n)$ of Eq. (13.2) as follows, respectively:

$$\mathbf{w}_o(n) = a\mathbf{w}_o(n - 1) + \boldsymbol{\omega}(n) \tag{13.44}$$

and

$$d(n) = \mathbf{w}_o^{\mathrm{H}}(n - 1)\mathbf{u}(n) + \nu(n). \tag{13.45}$$

Accordingly, using Eqs. (13.41), (13.44), and (13.45), we may express the update equation for the weight-error vector in the RLS algorithm as shown by

$$\begin{aligned}\boldsymbol{\varepsilon}(n) = {} & [\mathbf{I} - \boldsymbol{\Phi}^{-1}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]\boldsymbol{\varepsilon}(n - 1) - \boldsymbol{\Phi}^{-1}(n)\mathbf{u}(n)\nu^*(n) \\ & - (1 - a)\mathbf{w}_o(n - 1) + \boldsymbol{\omega}(n),\end{aligned} \tag{13.46}$$

where $\mathbf{I}$ is the identity matrix. The linear stochastic difference equation (13.46) provides a complete description of the RLS algorithm embedded in the system model of Fig. 13.2, bearing in mind the aforementioned minor change in notation. As with the LMS algorithm, we assume that the model parameter a is close to one, so that we may ignore the term $(1 - a)\mathbf{w}_o(n - 1)$. That is, the process equation is described essentially by a random-walk model, for which Eq. (13.46) reduces to

$$\boldsymbol{\varepsilon}(n) = [\mathbf{I} - \boldsymbol{\Phi}^{-1}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]\boldsymbol{\varepsilon}(n - 1) - \boldsymbol{\Phi}^{-1}(n)\mathbf{u}(n)\nu^*(n) + \boldsymbol{\omega}(n). \tag{13.47}$$

Before proceeding further, it is instructive to find an approximation for the inverse matrix $\boldsymbol{\Phi}^{-1}(n)$ that makes the tracking analysis of the RLS algorithm mathematically trackable in a meaningful manner. To do so, we first take the expectation of both sides of Eq. (13.42), obtaining

$$\begin{aligned}\mathbb{E}[\boldsymbol{\Phi}(n)] &= \sum_{i=1}^{n} \lambda^{n-i}\mathbb{E}[\mathbf{u}(i)\mathbf{u}^{\mathrm{H}}(i)] \\ &= \sum_{i=1}^{n} \lambda^{n-i}\mathbf{R}_u \\ &= \mathbf{R}_u(1 + \lambda + \lambda^2 + \cdots + \lambda^{n-1}),\end{aligned} \tag{13.48}$$

where $\mathbf{R}_u$ is the ensemble-average correlation matrix of the input vector $\mathbf{u}(n)$. The series inside the parentheses on the right-hand side of Eq. (13.48) represents a geometric series with the following description: a first term equal to unity, geometric ratio equal to λ, and length equal to n. We now assume that n is large enough for the convergence mode of the algorithm to be essentially over. Under this condition, we may treat the geometric series to be essentially of infinite length, and so use the formula for the sum of such a series to rewrite Eq. (13.48) in the compact form

$$\mathbb{E}[\boldsymbol{\Phi}(n)] = \frac{\mathbf{R}_u}{1 - \lambda} \qquad \text{for } n \text{ large}. \tag{13.49}$$

Equation (13.49) defines the expectation (ensemble average) of $\mathbf{\Phi}(n)$, on the basis of which we may express $\mathbf{\Phi}(n)$ itself as follows (Eleftheriou & Falconer, 1986):

$$\mathbf{\Phi}(n) = \frac{\mathbf{R}_u}{1 - \lambda} + \widetilde{\mathbf{\Phi}}(n) \qquad \text{for } n \text{ large,} \tag{13.50}$$

where $\widetilde{\mathbf{\Phi}}(n)$ is a Hermitian *perturbation matrix* whose individual entries are represented by zero-mean random variables that are statistically independent from the input vector $\mathbf{u}(n)$. Assuming a slow adaptive process (i.e., the exponential weighting factor λ is close to unity), we may view the $\mathbf{\Phi}(n)$ in Eq. (13.50) as a *quasi-deterministic* matrix, in the sense that for large n we have[2]

$$\mathbb{E}[\|\widetilde{\mathbf{\Phi}}(n)\|^2] \ll \mathbb{E}[\|\mathbf{\Phi}(n)\|^2]$$

where $\|\cdot\|$ denotes matrix norm. Under this condition, we may go one step further by ignoring the perturbation matrix $\widetilde{\mathbf{\Phi}}(n)$, and thereby approximate the correlation matrix $\mathbf{\Phi}(n)$ as

$$\mathbf{\Phi}(n) \approx \frac{\mathbf{R}_u}{1 - \lambda} \qquad \text{for } n \text{ large.} \tag{13.51}$$

This approximation is crucial to the tracking analysis of the RLS algorithm presented herein. In a corresponding way to Eq. (13.51), we may express the inverse matrix $\mathbf{\Phi}^{-1}(n)$ as

$$\mathbf{\Phi}^{-1}(n) \approx (1 - \lambda)\mathbf{R}_u^{-1} \qquad \text{for } n \text{ large,} \tag{13.52}$$

where $\mathbf{R}_u^{-1}$ is the inverse of the ensemble-average correlation matrix $\mathbf{R}_u$.

Returning to Eq. (13.47) and using the approximation of Eq. (13.52) for $\mathbf{\Phi}^{-1}(n)$, we may now write

$$\begin{aligned}\boldsymbol{\varepsilon}(n) \approx\ & [\mathbf{I} - (1 - \lambda)\mathbf{R}_u^{-1}\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]\boldsymbol{\varepsilon}(n - 1) \\ & - (1 - \lambda)\mathbf{R}_u^{-1}\mathbf{u}(n)\nu^*(n) + \boldsymbol{\omega}(n) \qquad \text{for } n \text{ large.}\end{aligned} \tag{13.53}$$

Typically, the exponential weighting factor λ is close to unity, so that $1 - \lambda$ is small. Then, invoking the *direct-averaging method* of Section 6.4, we may state that $\boldsymbol{\varepsilon}(n)$ is "close" to the solution of the new stochastic difference equation in $\boldsymbol{\varepsilon}_0(n)$:

$$\boldsymbol{\varepsilon}_0(n) \approx \lambda\boldsymbol{\varepsilon}_0(n - 1) - (1 - \lambda)\mathbf{R}_u^{-1}\mathbf{u}(n)\nu^*(n) + \boldsymbol{\omega}(n) \qquad \text{for } n \text{ large.} \tag{13.54}$$

Equation (13.54) for the RLS algorithm has a form that is dramatically different from that of Eq. (13.25) for the LMS algorithm, which, of course, is to be expected.

Evaluating the correlation matrix of $\boldsymbol{\varepsilon}_0(n)$ in Eq. (13.54) and invoking the assumptions of Section 13.2, we obtain

[2]A completely general proof that the correlation matrix $\mathbf{\Phi}(n)$ is quasi-deterministic is yet to be presented in the literature. The issue was apparently first discussed in Eleftheriou and Falconer (1986), who used heuristic arguments. It was also discussed in Macchi and Bershad (1991), who presented a proof for the case of a nonstationary signal, namely, a noisy chirped sinusoid. This signal includes the commonly encountered example of a pure sinusoid in additive white Gaussian noise as a special case, which validates the proof for a stationary environment, too. However, a limitation of the proof presented by Macchi and Bershad is that it hinges on the unrealistic assumption that successive input vectors are statistically independent.

$$\mathbf{K}_0(n) \approx \lambda^2\mathbf{K}_0(n-1) + (1-\lambda)^2\sigma_\nu^2\mathbf{R}_u^{-1} + \mathbf{R}_\omega \qquad \text{for } n \text{ large.} \tag{13.55}$$

For a *steady-state solution* of the difference equation (13.55), for which n is large, we may legitimately set $\mathbf{K}_0(n-1) = \mathbf{K}_0(n)$. Under this condition, Eq. (13.55) simplifies as

$$(1-\lambda^2)\mathbf{K}_0(n) \approx (1-\lambda)^2\sigma_\nu^2\mathbf{R}_u^{-1} + \mathbf{R}_\omega \qquad \text{for } n \text{ large.} \tag{13.56}$$

For λ close to unity, we may use the approximation

$$\begin{aligned} 1-\lambda^2 &= (1-\lambda)(1+\lambda) \\ &\approx 2(1-\lambda). \end{aligned} \tag{13.57}$$

Accordingly, we may further simplify the correlation matrix for the RLS algorithm as

$$\mathbf{K}_0(n) \approx \frac{1-\lambda}{2}\sigma_\nu^2\mathbf{R}_u^{-1} + \frac{1}{2(1-\lambda)}\mathbf{R}_\omega \qquad \text{for } n \text{ large.} \tag{13.58}$$

This is the equation for evaluating the tracking capability of the RLS algorithm for the system identification problem described in Fig. 13.2, subject to the condition that a is close to unity.

Mean-Square Deviation of the RLS Algorithm

Applying the formula of Eq. (13.10) to Eq. (13.58), we readily find that the mean-square deviation of the RLS algorithm is defined by

$$\mathcal{D}(n) \approx \frac{1-\lambda}{2}\sigma_\nu^2\mathrm{tr}[\mathbf{R}_u^{-1}] + \frac{1}{2(1-\lambda)}\mathrm{tr}[\mathbf{R}_\omega] \qquad \text{for } n \text{ large.} \tag{13.59}$$

The first term, $(1-\lambda)\sigma_\nu^2\mathrm{tr}[\mathbf{R}_u^{-1}]/2$, is the estimation variance due to the measurement noise $\nu(n)$. The second term, $\mathrm{tr}[\mathbf{R}_\omega]/2(1-\lambda)$, is the lag variance due to the process noise vector $\boldsymbol{\omega}(n)$. These two contributions vary in proportion to $1-\lambda$ and $(1-\lambda)^{-1}$, respectively. The optimum value of the forgetting factor, λ_{opt}, occurs when these two contributions are equal. Thus, from Eq. (13.59), we readily find that

$$\lambda_{\text{opt}} \approx 1 - \frac{1}{\sigma_\nu}\left(\frac{\mathrm{tr}[\mathbf{R}_\omega]}{\mathrm{tr}[\mathbf{R}_u^{-1}]}\right)^{1/2}. \tag{13.60}$$

Correspondingly, the minimum mean-square deviation of the RLS algorithm is given by

$$\mathcal{D}_{\min} \approx \sigma_\nu\mathrm{tr}\left([\mathbf{R}_u^{-1}]\mathrm{tr}[\mathbf{R}_\omega]\right)^{1/2}. \tag{13.61}$$

Misadjustment of the RLS Algorithm

Multiplying both sides of Eq. (13.58) by the correlation matrix $\mathbf{R}_u$, we get

$$\mathbf{R}_u\mathbf{K}_0(n) \approx \frac{1-\lambda}{2}\sigma_\nu^2\mathbf{I} + \frac{1}{2(1-\lambda)}\mathbf{R}_u\mathbf{R}_\omega \qquad \text{for } n \text{ large.} \tag{13.62}$$

The identity matrix $\mathbf{I}$ is of size M by M, where M is the number of taps in the adaptive FIR filter. Hence, taking the trace of the two sides of Eq. (13.62) yields

$$\mathrm{tr}[\mathbf{R}_u\mathbf{K}_0(n)] \approx \frac{1-\lambda}{2}\sigma_v^2 M + \frac{1}{2(1-\lambda)}\mathrm{tr}[\mathbf{R}_u\mathbf{R}_\omega] \qquad \text{for } n \text{ large.} \tag{13.63}$$

Finally, using Eq. (13.19), we readily find that the misadjustment of the RLS algorithm is given by

$$\mathcal{M}(n) \approx \frac{1-\lambda}{2}M + \frac{1}{2\sigma_v^2(1-\lambda)}\mathrm{tr}[\mathbf{R}_u\mathbf{R}_\omega] \qquad \text{for } n \text{ large.} \tag{13.64}$$

The first term on the right-hand side of Eq. (13.64) represents the noise misadjustment of the RLS algorithm due to the measurement noise $\nu(n)$. This term varies *linearly* with $1-\lambda$; note also that it depends on the number of taps, M, in the adaptive FIR filter. The second term represents the lag misadjustment of the RLS algorithm due to the process noise vector $\boldsymbol{\omega}(n)$; this term varies *inversely* with $1-\lambda$. The optimum value of the forgetting factor, λ_{opt}, occurs when these two contributions are equal. We thus find from Eq. (13.64) that

$$\lambda_{\text{opt}} \approx 1 - \frac{1}{\sigma_\nu}\left(\frac{1}{M}\mathrm{tr}[\mathbf{R}_u\mathbf{R}_\omega]\right)^{1/2}. \tag{13.65}$$

Correspondingly, the minimum value of the misadjustment produced by the RLS algorithm is given by

$$\mathcal{M}_{\min} \approx \frac{1}{\sigma_\nu}(M\,\mathrm{tr}[\mathbf{R}_u\mathbf{R}_\omega])^{1/2}. \tag{13.66}$$

Here also, we find that the two criteria—minimum misadjustment and minimum mean-square deviation—lead to different values for λ_{opt}. For these values to be meaningful, they are subject to the assumption that λ_{opt} is positive and just less than unity.

We now have all the tools we need to make a quantitative comparison between the LMS and RLS algorithms in the context of the system model depicted in Fig. 13.2.

13.7 COMPARISON OF THE TRACKING PERFORMANCE OF LMS AND RLS ALGORITHMS

Given the fact that the LMS and RLS algorithms are formulated in entirely different ways, it is only natural to find that they exhibit not only different convergence properties but also different tracking properties. The difference in their tracking behavior may be traced back to the stochastic difference equations (13.25) and (13.54). In the RLS algorithm, the input vector $\mathbf{u}(n)$ is premultiplied by the inverse matrix $\mathbf{R}^{-1}$, wherein lies the fundamental difference between it and the LMS algorithm. Moreover, comparing Eqs. (13.25) and (13.54), on which the tracking analysis presented in the previous two sections is based, we see that, in a loose sense, $1-\lambda$ in the RLS algorithm plays a role analogous to that of μ in the LMS algorithm. In making this analogy, however, we should try to be more precise. In particular, the exponential weighting factor λ is dimensionless,

whereas the step-size parameter μ has the inverse dimension of power. To correct for this dimensional discrepancy, we do the following:

- For the LMS algorithm, we define the *normalized step-size parameter*

$$\mu_{\text{norm}} = \mu\sigma_u^2, \tag{13.67}$$

  where σ_u^2 is the variance of the zero-mean tap input $u(n)$.
- For the RLS algorithm, we define the *forgetting rate*

$$\beta = 1 - \lambda. \tag{13.68}$$

Moving to the main issue of interest, we may use Eqs. (13.35) and (13.61), on the one hand, and Eqs. (13.40) and (13.66), on the other, to formulate a corresponding pair of ratios for comparing the "optimum" tracking performance of the LMS and RLS algorithms for the system identification problem at hand; one ratio is based on the mean-square deviation, and the other is based on the misadjustment, as the figure of merit. Specifically, we may write

$$\frac{\mathcal{D}_{\text{min}}^{\text{LMS}}}{\mathcal{D}_{\text{min}}^{\text{RLS}}} \approx \left(\frac{M\,\text{tr}[\mathbf{R}_u^{-1}\mathbf{R}_\omega]}{\text{tr}[\mathbf{R}_u^{-1}]\,\text{tr}[\mathbf{R}_\omega]}\right)^{1/2} \tag{13.69}$$

and

$$\frac{\mathcal{M}_{\text{min}}^{\text{LMS}}}{\mathcal{M}_{\text{min}}^{\text{RLS}}} \approx \left(\frac{\text{tr}[\mathbf{R}_u]\,\text{tr}[\mathbf{R}_\omega]}{M\,\text{tr}[\mathbf{R}_u\mathbf{R}_\omega]}\right)^{1/2}, \tag{13.70}$$

where $\mathbf{R}_u$ is the correlation matrix of the input vector $\mathbf{u}(n)$, $\mathbf{R}_\omega$ is the correlation matrix of the process noise vector $\boldsymbol{\omega}$, and M is the number of taps in the adaptive FIR filter of Fig. 13.2. Clearly, whatever comparison we make between the LMS and RLS algorithms on the basis of Eqs. (13.69) and (13.70), the result depends on the environmental conditions that are prevalent and, in particular, on how the correlation matrices $\mathbf{R}_\omega$ and $\mathbf{R}_u$ are defined. In what follows, we consider three different hypothetical examples.[3]

EXAMPLE 1: $\mathbf{R}_\omega = \sigma_\omega^2\mathbf{I}$

Consider first the case of the process noise vector $\boldsymbol{\omega}(n)$ in the first-order Markov model of Eq. (13.1) originating from a white-noise source of zero mean and variance σ_ω^2. We may express the correlation matrix of $\boldsymbol{\omega}(n)$ as

$$\mathbf{R}_\omega = \sigma_\omega^2\mathbf{I}, \tag{13.71}$$

where $\mathbf{I}$ is the M-by-M identity matrix. Then, using Eq. (13.71) in Eqs. (13.69) and (13.70), we get the respective results (after cancelling common terms)

$$\mathcal{D}_{\text{min}}^{\text{LMS}} \approx \mathcal{D}_{\text{min}}^{\text{RLS}}, \quad \mathbf{R}_\omega = \sigma_\omega^2\mathbf{I} \tag{13.72}$$

and

$$\mathcal{M}_{\text{min}}^{\text{LMS}} \approx \mathcal{M}_{\text{min}}^{\text{RLS}}, \quad \mathbf{R}_\omega = \sigma_\omega^2\mathbf{I}. \tag{13.73}$$

[3]Example 1 is discussed in Widrow and Walach (1984) and Eleftheriou and Falconer (1986); examples 2 and 3 are discussed in Benveniste et al. (1987) and Slock and Kailath (1993), respectively.

Accordingly, we may state that the LMS and RLS algorithms produce essentially the same minimum levels of misadjustment and mean-square deviation for the case of a process noise vector $\boldsymbol{\omega}(n)$ drawn from a white-noise source, irrespective of the statistics of the input vector.

EXAMPLE 2: $\mathbf{R}_\omega = c_1\mathbf{R}_u$

Consider next the example when the correlation matrix $\mathbf{R}_\omega$ of the process noise vector $\boldsymbol{\omega}(n)$ in the first-order Markov model of Eq. (13.1) equals a constant c_1 times the correlation matrix $\mathbf{R}_u$ of the input vector $\mathbf{u}(n)$. The scaling factor c_1 is introduced here for two reasons:

1. To account for the fact that the process noise vector $\boldsymbol{\omega}(n)$ and the input vector $\mathbf{u}(n)$ are ordinarily measured in different units.
2. To ensure that the optimum μ for the LMS algorithm in Eq. (13.34) or Eq. (13.39), and the optimum λ for the RLS algorithm in Eq. (13.60) or Eq. (13.65), assume meaningful values.

Thus, putting $\mathbf{R}_\omega = c_1\mathbf{R}_u$ in Eqs. (13.69) and (13.70) and cancelling the scaling factor c_1, we get the two comparative yardsticks listed under $\mathbf{R}_\omega = c_1\mathbf{R}_u$ in Table 13.1. Before commenting on these results, it is instructive to go on and consider the complementary example described next.

EXAMPLE 3: $\mathbf{R}_\omega = c_2\mathbf{R}_u^{-1}$

In this final example, the correlation matrix $\mathbf{R}_\omega$ of the process noise vector $\boldsymbol{\omega}(n)$ is equal to a constant c_2 times the inverse of the correlation matrix $\mathbf{R}$ of the input vector $\mathbf{u}(n)$. The scaling factor c_2 is used here for exactly the same reasons given in Example 2. Thus, putting $\mathbf{R}_\omega = c_2\mathbf{R}_u^{-1}$ in Eqs. (13.69) and (13.70) and again cancelling the scaling factor c_2, we get the remaining two comparative yardsticks listed under $\mathbf{R}_\omega = c_2\mathbf{R}_u^{-1}$ in Table 13.1.

Remarks pertaining to Examples 2 and 3

The 2-by-2 array of entries shown in the table exhibits a useful property, namely, that of *reciprocal symmetry*. The significance of this property may be substantiated in theoretical terms by recognizing that the pair of cross-diagonal terms in this 2-by-2 array lend themselves to the application of the *Cauchy-Schwartz inequality*, for which the reader is referred to the end-of-chapter Problem 7. The findings of this problem, resulting from application of this inequality, are summarized in the following pair of equations:

TABLE 13.1 Comparative Yardsticks for LMS and RLS Algorithms for Examples 2 and 3

	$\mathbf{R}_\omega = c_1\mathbf{R}_u$	$\mathbf{R}_\omega = c_2\mathbf{R}_u^{-1}$
$\dfrac{\mathcal{D}_{\min}^{\text{LMS}}}{\mathcal{D}_{\min}^{\text{RLS}}}$	$\dfrac{M}{(\text{tr}[\mathbf{R}_u^{-1}]\,\text{tr}[\mathbf{R}_u])^{1/2}}$	$\dfrac{(M\,\text{tr}[\mathbf{R}_u^{-2}])^{1/2}}{\text{tr}[\mathbf{R}_u^{-1}]}$
$\dfrac{\mathcal{M}_{\min}^{\text{LMS}}}{\mathcal{M}_{\min}^{\text{RLS}}}$	$\dfrac{\text{tr}[\mathbf{R}_u]}{(M\,\text{tr}[\mathbf{R}_u^2])^{1/2}}$	$\dfrac{1}{M}(\text{tr}[\mathbf{R}_u]\,\text{tr}[\mathbf{R}_u^{-1}])^{1/2}$

$$\mathcal{M}_{\min}^{\text{LMS}} \leq \mathcal{M}_{\min}^{\text{RLS}} \quad \text{for } \mathbf{R}_\omega = c_1\mathbf{R}_u \tag{13.74}$$

and

$$\mathcal{D}_{\min}^{\text{RLS}} \leq \mathcal{D}_{\min}^{\text{LMS}} \quad \text{for } \mathbf{R}_\omega = c_2\mathbf{R}_u^{-1}. \tag{13.75}$$

In words, the property of reciprocal symmetry, embodied in Eqs. (13.74) and (13.75), may be expressed as follows: If, for $\mathbf{R}_\omega = c_1\mathbf{R}_u$, the minimum mean-square deviation $\mathcal{D}_{\min}$ produced by the LMS algorithm is smaller than the corresponding value produced by the RLS algorithm, then for $\mathbf{R}_\omega = c_2\mathbf{R}_u^{-1}$, the minimum misadjustment $\mathcal{M}_{\min}$ produced by the RLS algorithm is smaller than the corresponding value produced by the LMS algorithm.

To illustrate the validity of this statement, consider the special case of an adaptive filter with $M = 2$, for which the 2-by-2 correlation matrix of the input vector $\mathbf{u}(n)$ is denoted by

$$\mathbf{R}_u = \begin{bmatrix} r_{11} & r_{21} \\ r_{21} & r_{22} \end{bmatrix}, \quad r_{12} = r_{21}.$$

For this specification of $\mathbf{R}_u$, the 2-by-2 array of Table 13.1 takes on the particular form presented in Table 13.2. Next, recognizing that, since any 2-by-2 correlation matrix satisfies the condition

$$(r_{11} - r_{22})^2 + (2r_{21})^2 \geq 0,$$

we see that Table 13.2 leads us to make the following statements encompassing all four entries of the array:

1. For $\mathbf{R}_\omega = c_1\mathbf{R}_u$, the LMS algorithm performs better than the RLS algorithm, in that it yields smaller values for both $\mathcal{D}_{\min}$ and $\mathcal{M}_{\min}$.
2. For $\mathbf{R}_\omega = c_2\mathbf{R}_u^{-1}$, the RLS algorithm performs better than the LMS algorithm, in that it yields smaller values for both $\mathcal{D}_{\min}$ and $\mathcal{M}_{\min}$.

Examples 2 and 3 clearly illustrate that neither the LMS algorithm nor the RLS algorithm has a complete monopoly over good tracking behavior, which, in itself, is an insightful theoretical point-of-view to take away from Examples 2 and 3.

From a practical perspective, however, we do typically find that one of these two algorithms, LMS or RLS, is the preferred choice for tracking a nonstationary environment, bearing in mind the environmental conditions that are to be resolved. Such issues are addressed in Sections 13.8 to 13.11 that go beyond traditional adaptive filtering algorithms, LMS and RLS.

TABLE 13.2 Comparative Yardsticks for LMS and RLS Algorithms for Examples 2 and 3, Assuming That $M = 2$

	$\mathbf{R}_\omega = c_1\mathbf{R}_u$	$\mathbf{R}_\omega = c_2\mathbf{R}_u^{-1}$
$\dfrac{\mathcal{D}_{\min}^{\text{LMS}}}{\mathcal{D}_{\min}^{\text{RLS}}}$	$\dfrac{2\sqrt{r_{11}r_{22} - r_{21}^2}}{r_{11} + r_{22}}$	$\dfrac{\sqrt{2(r_{11}^2 + 2r_{21}^2 + r_{22}^2)}}{r_{11} + r_{22}}$
$\dfrac{\mathcal{M}_{\min}^{\text{LMS}}}{\mathcal{M}_{\min}^{\text{RLS}}}$	$\dfrac{r_{11} + r_{22}}{\sqrt{2(r_{11}^2 + 2r_{21}^2 + r_{22}^2)}}$	$\dfrac{r_{11} + r_{22}}{2\sqrt{r_{11}r_{22} - r_{21}^2}}$

13.8 TUNING OF ADAPTATION PARAMETERS

In light of the material covered in the preceding six sections on a time-varying system identification problem, we may say that Issue 1 raised in Section 13.1 on traditional LMS and RLS algorithms has been addressed, except for the matter of how to choose their adaptation parameters. The most straightforward approach is to tune them *manually*, which is feasible in a stationary environment. However, in a nonstationary environment, the more logical procedure is to tune the adaptation parameters *automatically*, so as to closely match the continually varying state of the nonstationary environment. This task is the essence of Issue 2 in Section 13.1.

Basically, to modify the LMS or RLS algorithm so that the respective adaptation parameter, μ or $1 - \lambda$, is tuned in an automatic manner, we propose to expand the algorithm by including a second level of adaptation.

The need for this expansion is satisfied by means of a *learning-within-learning scheme*, a block diagram of which is depicted in Fig. 13.3. This new scheme embodies two separate control mechanisms that work together side-by-side. Specifically, we have:

- *Primary control mechanism*, which is driven by the estimation error, $e(n)$; the objective here is to automatically control the adjustments applied to the tap-weights of the FIR filter in the algorithm in the traditional manner.
- *Secondary control mechanism*, which is also driven by the estimation error, $e(n)$. This time, however, the objective is to automatically apply appropriate adjustments to the adaptation parameter embedded in the primary control mechanism.

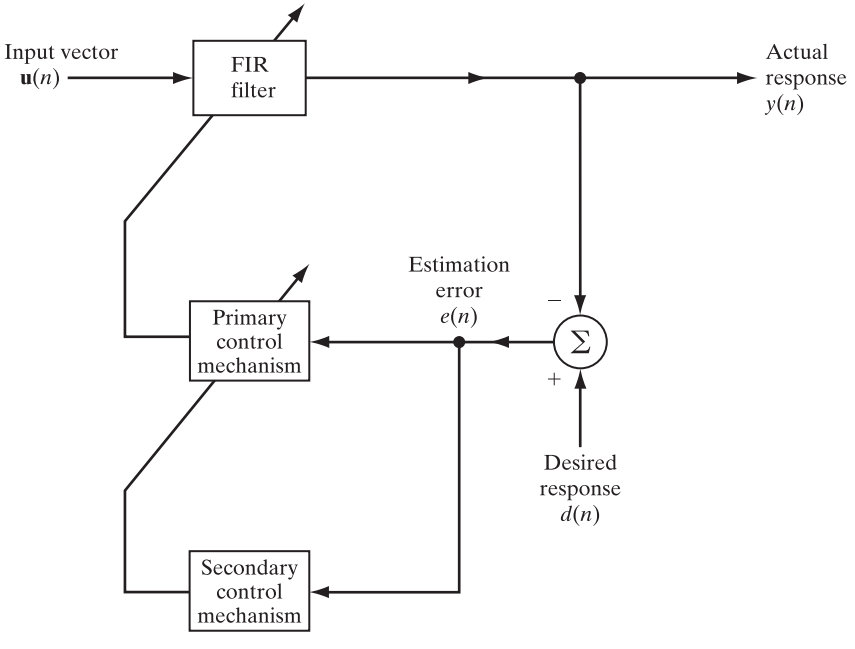


FIGURE 13.3 Block diagram of supervised learning-within-learning scheme.

There are two approaches on how to implement the learning-within-learning scheme of Fig. 13.3:

1. *Scaler approach.* In this first approach, the adaptation parameter takes the form of a *time-varying scaler*. For example, in the LMS algorithm, we use $\mu(n+1)$ for the step-size parameter μ in place of the traditional fixed-value μ, as shown by

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu(n+1)(\mathbf{u}(n)e^*(n)).$$

2. *Vector approach.* In this second approach, the adaptation parameter takes the form of a *time-varying vector*, each element of which is correlated with some feature of the input data vector $\mathbf{u}(n)$. Again, considering the LMS algorithm, we now write

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \boldsymbol{\mu}(n+1)\text{o}(\mathbf{u}(n)e^*(n)),$$

where the operator o denotes the *element-wise section product*, involving $\boldsymbol{\mu}(n+1)$ and $\mathbf{u}(n)$. Note that according to this formula, dimensionality of the step-size parameter vector $\boldsymbol{\mu}(n+1)$ is the same as that of the input vector $\mathbf{u}(n)$, or, equivalently, the tap-weight vector $\hat{\mathbf{w}}(n)$, which is intuitively satisfying.

The difference between these two approaches may be summed up as follows: In the scaler approach, the error-performance surface is of a quadratic form, with its minimum point (i.e., optimum Wiener solution) varying with time. On the other hand, the vector approach has an advantage over the scaler one in that it can deal with an error-performance surface that is more complex, with different curvatures in different directions.

In a way, there is nothing to stop us from extending the vector approach and applying it to the RLS algorithm, too. Unfortunately, with complexity of the RLS algorithm following a square law, the resulting complexity of the expanded RLS algorithm based on the scheme of Fig. 13.3 could become computationally unmanageable, particularly so if dimensionality of the FIR filter is high. For this reason, when we have to deal with *big data* (i.e., data that are continually growing in volume), simplicity in computational complexity is of the essence, hence the compulsion to opt for the LMS algorithm as the basis for algorithmic modification.

To summarize the discussion presented in this section, we may identify two procedures for implementing the learning-within learning scheme of Fig. 13.3:

Procedure 1: The adaptation parameter in the primary control mechanism is automatically tuned, but the adaptation parameter in the secondary control mechanism is manually tuned.

Procedure 2: The adaptation parameters in the primary and secondary control mechanisms are both automatically tuned, thereby becoming tuning-free.

Clearly, Procedure 2 is much more difficult than Procedure 1.

Having settled on the idea of expanding the LMS algorithm, a related issue that needs to be addressed is the following:

How do we mechanize the learning-within-learning scheme of Fig. 13.3, so that adaptive-parametric training is performed automatically throughout the entire scheme in a rigorous manner analytically?

In a way, we may view this question as the *ultimate in automatically tuned linear adaptive filtering*. Needless to say, it presents a challenge by requiring not only operating in an unknown nonstationary environment but also ruling out manual tuning altogether.

With the exposé presented in this section, we have set the stage for addressing the issues to be covered in the next two sections. To be specific, Section 13.9 presents a detailed treatment of an algorithm known as the incremental delta-bar-delta (IDBD) algorithm, and Section 13.10 describes a new procedure called the Autostep method. With both the IDBD algorithm and the Autostep method employing a pair of adaptation parameters, henceforth the following terminology is adopted:

- The step-size parameter, μ, is retained for the primary control mechanism in Fig. 13.3.
- The *meta*-learning-rate parameter, κ, is adopted for the secondary control mechanism in the figure.

The term "meta-learning" is used in the literature on neural networks and learning machines to describe "learning-how-to-learn" the adaptation parameters (i.e., learning rate) of an algorithm based on the scheme in Fig. 13.3.

13.9 INCREMENTAL DELTA-BAR-DELTA (IDBD) ALGORITHM

The *incremental delta-bar-delta (IDBD) algorithm* is an incremental meta-learning algorithm.[4] Basically, mathematical composition of the algorithm consists of two parts:

- The first part pertains to the LMS algorithm as its basis, with a slight modification that involves the use of a time-varying step-size parameter, $\mu_i(n+1)$, where $i = 0, 1, \ldots, M-1$, and where each of the M tap weights in the FIR filter is tuned automatically in accordance with the vector approach.
- The second part is an addendum made up of three equations, two of which involve the use of memory-related parameters and the third of which is a meta-learning-rate parameter that is tuned manually.

[4]In an intuitive sense, the IDBD algorithm, devised by Sutton (1992), is the same as the *delta-bar-delta (DBD) algorithm* originally devised by Jacobs (1988). The "I" in IDBD stands for "incremental," which means that the adjustments applied to the tap weights in the FIR filter are relatively small, in accordance with the underlying principle of stochastic gradient descent.

The IDBD algorithm differs from its DBD counterpart, however, in the following two ways:

1. First, the DBD algorithm was originally developed by Jacobs for the supervised training of neural networks on a batch-by-batch basis. On the other hand, the IDBD algorithm devised by Sutton deals with supervised training of a stochastic gradient descent algorithm that progresses in a sequential manner, with each data point being used only once.
2. Second, the DBD algorithm has three free parameters, whereas the IDBD algorithm has a single free parameter, namely, the meta-learning (step-size) parameter.

The LMS Part of IDBD

To proceed, let $\{u(n-i), d(n)\}, i=0,1,\ldots,M-1$, denote the data set used for supervised training of the LMS part of the IDBD algorithm for $n=1,2,\ldots$. As usual, M denotes the number of tap weights in the FIR filter. Following the traditional form of the LMS algorithm, we write

$$e(n) = d(n) - \sum_{i=0}^{M-1} \hat{w}_i^*(n)u(n-i) \tag{13.76}$$

and

$$\hat{w}_i(n+1) = \hat{w}_i(n) + \mu_i(n+1)u(n-i)e^*(n), \quad i = 0,1,\ldots,M-1, \tag{13.77}$$

where $e(n)$ is the estimation error. For reasons that will become apparent later, the time-varying step-size parameter μ_i is updated *before* updating the tap weights in Eq. (13.77). Note also that in Eq. (13.77), the step-size parameter varies with the tap weight in accordance with the vector approach described in Section 13.8.

The step-size parameter in the LMS algorithm plays a key role in the supervised learning process, which is why it is also commonly referred to in the neural networks literature as a "learning-rate parameter." In dealing with adaptation in a nonstationary environment, this parameter should be carefully distributed, such that the following desirable condition is attained (see Sutton, 1992):

> Inputs that are likely to be irrelevant should be given small learning rates (step sizes); on the other hand, inputs that are likely to be relevant, and therefore important, should be given relatively large learning rates.

With this desirable condition in mind, the time-varying step-size parameter $\mu_i(n)$ in the IDBD algorithm is defined as follows:

$$\mu_i(n) = \exp(\alpha_i(n)), \quad i = 0,1,\ldots,M-1, \tag{13.78}$$

where $\alpha_i(n)$ is its first one of two *adaptable memory parameters* of the IDBD algorithm. The exponential relationship in Eq. (13.78) offers two practical advantages:

1. It assures that the step-size parameter $\mu_i(n)$ is positive for all i and all n, which is how it should be for stochastic gradient descent.
2. It provides a simple mechanism for producing "geometric" steps in $\alpha_i(n)$, which are described by relatively slow additive updates in $\alpha_i(n)$ for all i.

The IDBD Addendum

The adaptive process responsible for updating the step-size parameter $\mu_i(n)$ from one adaptation cycle to the next is performed in the secondary control mechanism in Fig. 13.3. Specifically, the α_i in Eq. (13.78) is updated as follows (Sutton, 1992):

$$\alpha_i(n+1) = \alpha_i(n) + \kappa u(n-i)e^*(n)h_i(n), \quad i = 0,1,\ldots,M-1, \tag{13.79}$$

where κ is a positive constant, denoting the *meta-learning rate (step-size) parameter* in the IDBD algorithm.

In Eq. (13.79), we have also introduced a new parameter, $h_i(n)$, denoting the second adaptable memory parameter of the IDBD algorithm. This new parameter is updated as follows (Sutton, 1992):

$$h_i(n+1) = \big(1 - \mu_i(n+1)|u(n-i)|^2\big)^+ h_i(n) + \mu_i(n+1)u(n-i)e^*(n), \quad i = 0, 1, \ldots, M-1, \quad (13.80)$$

where we have introduced the following new notation for a positive bounding operation:

$$(x)^+ = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}. \quad (13.81)$$

The memory parameter h_i plays the role of a decaying trace of the cumulative sum of changes recently made in the ith tap weight w_i.

For an intuitive understanding of the IDBD algorithm, consider first the update formula of Eq. (13.79), in which the notion of *correlation* plays a significant role in how the incremental change to α_i materializes, putting aside the meta-step-size parameter κ. Specifically, we may view the multiple product term $u(n-i)e^*(n)h_i(n)$ as the correlation between the following two terms at each adaptation cycle n:

1. The product term $u(n-i)e^*(n)$, which represents the incremental change to the ith tap-weight $\hat{w}_i(n)$.
2. The term $h_i(n)$, which represents a trace of tap-weight changes in the recent past.

Accordingly, we may say that if this correlation is positive, then the cumulative effect of past adaptation cycles results in an increase in the first adaptable memory parameter α_i for $i = 0, 1, \ldots, M-1$. If, on the other hand, the correlation has a negative (opposite) effect, the end result is a decrease in α_i.

Turning next to an intuitive understanding of the second adaptable memory parameter h_i in Eq. (13.80), we may say the following for $i = 0, 1, \ldots, M-1$:

- The first term on the right-hand side of Eq. (13.80) is a decay term because the product term $\mu_i(n+1)|u(n-1)|^2$ is a small positive quantity or else is zero.
- The second term on the right-hand side of Eq. (13.80) changes the h_i by an amount equal to the incremental change made in the tap-weight $\hat{w}_i(n)$, in accordance with Eq. (13.77).

We therefore conclude that whatever change is made to the h_i in Eq. (13.80), it will be incrementally small at each adaptation cycle.

Putting it all together, we may intuitively state that the IDBD algorithm is another example of stochastic gradient descent in the first memory parameter α_i. This intuitive statement is confirmed next mathematically.

Derivation of the IDBD Addendum

In algorithmic terms, the IDBD addendum consists of Eqs. (13.79) and (13.80). We first derive Eq. (13.79) on the meta-step-size parameter, κ, by following a procedure similar

to that described for the LMS algorithm in Chapter 5. Specifically, in place of the update formula of Eq. (5.4), for the issue at hand, we write the following:

$$\alpha_i(n+1) = \alpha_i(n) - \frac{1}{2}\kappa \frac{\partial |e(n)|^2}{\partial \alpha_i}, \tag{13.82}$$

The second term involves κ, and the scaling factor 1/2 has been introduced for convenience of presentation. Note also that the partial derivative of the squared absolute value of the estimation error, $e(n)$, in Eq. (13.82) is written with respect to α_i without dependence on the adaptation cycle n; the rationale for doing this is to signify that this partial differentiation is carried out with respect to an infinitesimally small α_i for all n. To proceed further, we would like to bring partial differentiation of $|e(n)|^2$ with respect to $\hat{w}_i(n)$ into the evaluation of $\partial|e(n)|^2/\partial\alpha_i$. To this end, we use the *chain rule of calculus* to express this partial derivative in the following equivalent form:

$$\frac{\partial |e(n)|^2}{\partial \alpha_i} = \sum_{j=0}^{M-1} \frac{\partial |e(n)|^2}{\partial \hat{w}_j(n)} \frac{\partial \hat{w}_j(n)}{\partial \alpha_i}, \tag{13.83}$$

where the summation is intended to cover all M tap-weights in the FIR filter. Under the reasonable proposition that the primary effect of applying an incrementally small change to the α_i in Eq. (13.83) is centered on the $\hat{w}_i(n)$, that is,

$$\frac{\partial \hat{w}_j(n)}{\partial \alpha_i} \approx 0 \qquad \text{for all } i \neq j, \tag{13.84}$$

we may then approximate Eq. (13.83) as follows:

$$\frac{\partial |e(n)|^2}{\partial \alpha_i} \approx \frac{\partial |e(n)|^2}{\partial \hat{w}_i(n)} \frac{\partial \hat{w}_i(n)}{\partial \alpha_i}.$$

In Chapter 5, we previously showed that [see Eq. (5.3)]

$$\frac{\partial |e(n)|^2}{\partial \hat{w}_i(n)} \approx -2u(n-i)e^*(n),$$

where we have substituted the symbol i for k. Furthermore, introducing the definition

$$h_i(n) = \frac{\partial \hat{w}_i(n)}{\partial \alpha_i} \tag{13.85}$$

for the second adaptable memory parameter, we may now go on to approximate Eq. (13.83) as follows:

$$\frac{\partial |e(n)|^2}{\partial \alpha_i} \approx -2u(n-i)e^*(n)h_i(n). \tag{13.86}$$

Accordingly, substituting Eqs. (13.84) to (13.86) into Eq. (13.82), we obtain the update formula for the first adaptable memory parameter α_i previously presented in Eq. (13.79).

Next, we go on to derive the update formula for the second adaptable memory parameter h_i. With time-updating in mind, we use the defining equation (13.85), followed by Eq. (13.77), to write

$$\begin{aligned} h_i(n+1) &= \frac{\partial \hat{w}_i(n+1)}{\partial \alpha_i} \\ &= \frac{\partial}{\partial \alpha_i}(\hat{w}_i(n) + \mu_i(n+1)u(n-i)e^*(n)) \\ &= \frac{\partial \hat{w}_i(n)}{\partial \alpha_i} + u(n-i)\frac{\partial}{\partial \alpha_i}(\mu_i(n+1)e^*(n)) \\ &= h_i(n) + u(n-i)\frac{\partial}{\partial \alpha_i}(\mu_i(n+1)e^*(n)). \end{aligned} \tag{13.87}$$

Applying the *product rule of calculus* to the second term in the last line of Eq. (13.87), we write

$$\frac{\partial}{\partial \alpha_i}(\mu_i(n+1)e^*(n)) = e^*(n)\frac{\partial \mu_i(n+1)}{\partial \alpha_i} + \mu_i(n+1)\frac{\partial e^*(n)}{\partial \alpha_i}. \tag{13.88}$$

For the first partial derivative in the right-hand side of Eq. (13.88), we use the defining formula of Eq. (13.78) to write

$$\begin{aligned} \frac{\partial \mu_i(n+1)}{\partial \alpha_i} &= \frac{\partial}{\partial \alpha_i}\exp(\alpha_i(n+1)) \\ &= \exp(\alpha_i(n+1)) \\ &= \mu_i(n+1). \end{aligned} \tag{13.89}$$

For the second partial derivative in Eq. (13.88), we use Eq. (13.76) for the estimation error, $e(n)$, to write

$$\begin{aligned} \frac{\partial e^*(n)}{\partial \alpha_i} &= \frac{\partial}{\partial \alpha_i}\left(d^*(n) - \sum_{j=0}^{M-1}\hat{w}_j(n)u^*(n-j)\right) \\ &\approx -u^*(n-i)\frac{\partial \hat{w}_j(n)}{\partial \alpha_i}; \quad \text{as for Eq. (13.84), where } \frac{\partial \hat{w}_j(n)}{\partial \alpha_i} \approx 0 \text{ for all } j \neq i \\ &= -u^*(n-i)h_i(n). \end{aligned} \tag{13.90}$$

Thus, substituting Eqs. (13.89) and (13.90) into Eq. (13.88), we obtain

$$\frac{\partial}{\partial \alpha_i}(\mu_i(n+1)e^*(n)) = \mu_i(n+1)e^*(n) - \mu_i(n+1)u^*(n-i)h_i(n).$$

Using this partial derivative in Eq. (13.87) and collecting common terms, we obtain

$$h_i(n+1) = \left(1 - \mu_i(n+1)|u(n-i)|^2\right)h_i(n) + \mu_i(n+1)u(n-i)e^*(n)\,. \tag{13.91}$$

All that is left for us to do now is to apply the positive-bounding operation previously introduced in Eq. (13.81) to the large pair of parentheses on the right-hand side of Eq. (13.91). In so doing, the derivation of the update formula of Eq. (13.80) for the second memory parameter h_i is completed.

With the LMS part of the IDBD algorithm being of a stochastic gradient kind, and having proved that the IDBD addendum is likewise, we may therefore state that the IDBD algorithm, treated as a whole, is an example application of the method of stochastic gradient descent. Consequently, computational complexity of the IDBD algorithm does indeed scale *linearly* with respect to dimensionality of the FIR filter inside the algorithm.

Summary of the IDBD Algorithm

In Table 13.3, we present a summary of the IDBD algorithm, described in its complex form.[5] As with the Autostep method, presented in the next section that builds on the IDBD algorithm, the algorithm follows a strict order of algorithmic assignments. In particular, the update formula of Eq. (13.79) is formulated differently in Table 13.3 by making use of Eq. (13.78).

TABLE 13.3 Summary of the IDBD Algorithm

Definitions:

Training data: $\{\mathbf{u}(n-i), d(n)\}$ for $i = 0, 1, \ldots, M-1$ and $n = 0, 1, 2, \ldots$

$\mu_i(n)$: step-size parameter for tap-weight $\hat{w}_i(n)$

κ: meta-step-size parameter

Initialization: Set

$\kappa = 10^{-2}$

$h_i(0) = 0$

$\hat{w}_i(0) = 0$

$\mu_i(0) =$ manually tuned or $\dfrac{0.1}{\lambda_{\max}}$ if the largest eigenvalue of the correlation matrix of input vector $\mathbf{u}(n)$ is computable

Computation:

For each point in the training data: $\{\mathbf{u}(n-i), d(n)\}$ for $i = 0, 1, \ldots, M-1$ at adaptation cycle n, compute

$$e(n) = d(n) - \sum_{i=0}^{M-1} \hat{w}_i^*(n)u(n-i)$$

$$\alpha_i(n+1) = \log(\mu_i(n)) + \kappa u(n-i)e^*(n)h_i(n)$$

$$\mu_i(n+1) = \exp(\alpha_i(n+1))$$

$$\hat{w}_i(n+1) = \hat{w}_i(n) + \mu_i(n+1)u(n-i)e^*(n)$$

$$h_i(n+1) = \big(1 - \mu_i(n+1)|u(n-i)|^2\big)^+ h_i(n) + \mu_i(n+1)u(n-i)e^*(n)$$

[5]The real-valued version of the IDBD algorithm, first described in Sutton (1992), is a special case of its complex IDBD algorithm summarized in Table 13.3.

13.10 AUTOSTEP METHOD

The *Autostep method* is the result of a major endeavor to develop a tuning-free step-size adaptive procedure for the LMS algorithm (Mahmood, 2010). When the input data are high-dimensional and abundant, linear adaptive filtering algorithms rooted in the method of stochastic gradient descent, such as the LMS algorithm, are among the few choices available. Therefore, an obvious logical step is to mitigate their inherent drawbacks.

One of the most notorious problems encountered in the use of traditional gradient-descent methods is associated with their step-size parameters. The IDBD algorithm is a powerful solution to mitigate the step-size problem of the LMS algorithm. As explained in the previous section, the IDBD algorithm vectorizes the step-size parameter of the LMS algorithm, where each element of this vector matches a particular feature that characterizes the input data; as with the LMS algorithm, linearity across time as well as memory complexity are preserved by construction. By means of this adaptive procedure, the IDBD algorithm improves on the level of performance achievable by the LMS algorithm. Unfortunately, a potential drawback is that in employing the new meta-step-size parameter, κ, the IDBD algorithm may very well end up needing manual tuning of its own.

The next important issue to resolve is therefore to make the IDBD algorithm tuning-free; that is, the parameter κ is assigned a fixed value. To achieve this highly desirable objective is to do the following:

> Find the means whereby the large fluctuations in the pre-assigned value of the meta-step-size size parameter, κ, experienced in going from application of the IDBD algorithm to another, are stabilized.

Unfortunately, a mathematical approach to realize this objective is difficult. To get around this difficulty, we may follow an exhaustive experimental approach, covering a wide range of different applications.

To this end, Mahmood carried out a detailed experimental study on a large database made up of simulated data as well as variety of real-world data sets; the study is summarized as follows:

- For the simulated data, inputs were drawn from independent and identically distributed (i.i.d.) standard Gaussian distributions, with the target tap weights assuming Gaussian random walks. In this way, different varieties of simulation problems were generated by altering the variance of the input data as well as the variance of drifts in the target tap weights.
- For the real-world data, inputs were drawn from the sensorimotor records of a robot that had 56 different sensors, including light sensors, distance sensors, heat sensors, an accelerometer, a magnetometer, and so on. The input data collected from the different classes of sensors had different scales of statistics (i.e., means and variances). Three million data samples were collected by running the robot for several hours. The task of the IDBD algorithm was to use all the sensors so that, at a particular sample point, the algorithm would predict the output of one of the sensors at the next sample point. Six such tasks were generated by choosing the target sensors from a different class of sensors for each task.

With this extensive database at hand, it was discovered experimentally that the IDBD algorithm performed better than the LMS algorithm on each of the simulated and real-world problems. However, the IDBD algorithm required its meta-step-size parameter κ to be tuned for each specific problem. Moreover, the best-tuned values of the parameter κ were different by orders of magnitude from one problem to the next. This form of tuning dependence was also observed for all other well-known linear adaptive filtering algorithms that were part of the experimental study.

During the course of the study, it was also discovered that the meta-step-size parameter of the IDBD algorithm, κ, has units, representing an important property that makes its best value dependent on the training data (i.e., input vector applied to the IDBD algorithm and corresponding desired response). Before application, with the training data being naturally unobservable, the parameter κ of the IDBD algorithm had to be tuned to the specific problem under study, presenting another issue to be resolved. To this end, the first step in improving the IDBD algorithm was to normalize the parameter κ in such a way that it became unit-less. In effect, this normalization attempted to adjust the parameter, κ, based on the training data. The resulting method was substantially less dependent on the parameter κ than the original IDBD algorithm. However, the method was still not entirely tuning-free, needing the use of manual tuning from one problem to the next.

To make the IDBD algorithm even less sensitive to the parameter, κ, the next step was to impose a bound on permissible values of κ. Specifically, whenever overshooting occurred in the sample cost function, the step size was reduced such that there would be no overshooting. This modification of the IDBD algorithm did indeed make the estimation process even less sensitive to the meta-step-size parameter.

One last point of interest: In the experiments described in this section, a common range of the meta-step-size values of the Autostep method was discovered (around the value of 0.01), which, amazingly enough, seemed to "work best" for all the simulated and real-world problems. Moreover, the achieved performance was at worst comparable to and at best better than that of the IDBD algorithm. The insensitivity of the Autostep method to its fixed meta-step-size parameter, also denoted by κ, was finally tested on a new set of six real-world robot problems. These experiments confirmed practical utility of the Autostep method as a new linear adaptive filtering algorithm whose step size is tuning-free—a remarkable achievement.

To summarize, in adopting the two modifications of the IDBD algorithm just described, the net result was formulation of the so-called Autostep method, the name of which is made up of two words: "auto" for automatic and "step" for step-size parameter.

Rules Governing the Autostep Method

The steps involved in the underlying heuristic rules of the Autostep method, developed first in the thesis by Mahmood (2010) and then presented in the paper by Mahmood et al. (2012), are as follows (for real-valued data):

Step 1: Normalization. A *recursive normalizer*, denoted by $v_i(n)$, is introduced in the incremental adjustment of the multiple product term, $\kappa u_i(n)e(n)h_i(n)$ in the real-valued version of Eq. (13.79), as shown by

$$v_i(n+1) = v_i(n) + \gamma\mu_i(n)u_i^2(n)\big(|u_i(n)e(n)h_i(n)| - v_i(n)\big), \tag{13.92}$$

where the input is

$$u_i(n) = u(n - i), \quad i = 0, 1, \ldots, M - 1.$$

The new parameter, γ, called the *forgetting factor*, is defined as the reciprocal of the time-update scale parameter, τ. Correspondingly, Eq. (13.79) is modified as follows:

$$\alpha_i(n + 1) = \alpha_i(n) + \kappa\left(\frac{u_i(n)e(n)h_i(n)}{v_i(n + 1)}\right), \tag{13.93}$$

or

$$\alpha_i(n+1) = \alpha_i(n) \text{ if } v_i(n+1) = 0. \tag{13.94}$$

Thus, the new term $v_i\,(n+1)$ plays the role of a *normalizer* whenever it is nonzero.

Step 2: Modification of the Normalizer. To upper bound the update formula of Eq. (13.92), the normalizer is next modified as follows:

$$v_i(n + 1) = \max(|u_i(n)e(n)h_i(n)|, v_i(n) + \gamma\mu_i(n)u_i^2(n)(|u_i(n)e(n)h_i(n)| - v_i(n))). \tag{13.95}$$

The net effect of this second modification is to have the term $u_i(n)e(n)h_i(n)/v_i(n + 1)$ essentially upper bounded to unity. In so doing, the choice of the meta-step-size parameter κ is guarded against the occasional occurrence of abrupt increases in the value of the multiple product term $u_i(n)e(n)h_i(n)$.

Step 3: High-Step Detection. The last step in formulating the Autostep method involves two matters:

- **a.** *Detection* of high steps taken along the direction of tap-weight update in the FIR filter, which could arise in a sample realization of the error-performance surface.
- **b.** *Scaling* of the M-by-1 step-size vector $\boldsymbol{\mu}$ to smaller values to ensure that the updated tap-weight vector $\hat{\mathbf{w}}$ does not overshoot the minimum point along the update direction on the surface.

The detection of such large step sizes (and their reduction) is attained by using the following "if-then" rule:

$$\text{If } \sum_{j=0}^{M-1} \mu_j(n + 1)\mu_j^2(n) > 1, \text{ then } \mu_i(n + 1) \text{ is replaced with } \frac{\mu_i(n + 1)}{\sum_{j=0}^{M-1} \mu_j(n + 1)u_j^2(n)}. \tag{13.96}$$

All the rules just described can be used individually in the IDBD algorithm. However, it is when all the rules are used collectively together that the IDBD algorithm is at its most robust behavior, which has been demonstrated experimentally in various applications of the algorithm (Mahmood et al., 2012).

Summary of the Autostep Method

Summary of this Autostep method is presented in Table 13.4, based on the use of *real-valued data*. In this table, it is important to note that unlike the IDBD algorithm, the Autostep method does not require the notation $(\cdot)^+$, because it is guaranteed by the upper bound operation imposed in the step-size parameter μ, as shown by

$$\mu_i(n+1) \leftarrow \frac{\mu_i(n+1)}{B}, \tag{13.97}$$

where

$$B = \max\left(\sum_{i=0}^{M-1} \mu_i(n+1)u^2(n-i), 1\right). \tag{13.98}$$

Accordingly, the μ_i is always nonnegative. Note that, in formulating Table 13.4, we have also made use of Eq. (13.78).

TABLE 13.4 Summary of the Autostep Method

Initialization: Set

κ and γ as 10^{-2} and 10^{-4}, respectively

$h_i(0) = 0$

$v_i(0) = 0$

$\hat{w}_i(0) = 0$ or as desired

$\mu_i(0) = 0.1$ or $\dfrac{0.1}{\lambda_{\max}}$ if $\lambda_{\max}$, where $\lambda_{\max}$ is the largest eigenvalue of the correlation matrix of the input vector $u(n)$, if it is computable.

Computation:

For each sample point in the training data: $\{u(n-i), d(n)\}$ for $i = 0, 1, \ldots, M-1$ at adaptation cycle n, compute

$$e(n) = d(n) - \sum_{i=0}^{M-1} \hat{w}_i(n)u(n-i)$$

$$v_i(n+1) = \max(|e(n)u(n-i)h_i(n)|, v_i(n) + \gamma\mu_i(n)|u(n-i)|^2(|e(n)u(n-i)h_i(n)| - v_i(n)))$$

$$\alpha_i(n+1) = \log(\mu_i(n)) + \kappa\frac{e(n)u(n-i)h_i(n)}{v_i(n+1)}, \text{ or } \alpha_i(n+1) = \log(\mu_i(n)) \text{ if } v_i(n+1) = 0$$

$$\mu_i(n+1) = \exp(\alpha_i(n+1))$$

$$B = \max\left(\sum_{i=0}^{M-1} \mu_i(n+1)|u(n-i)|^2, 1\right)$$

$$\mu_i(n+1) = \frac{\mu_i(n+1)}{B}$$

$$\hat{w}_i(n+1) = \hat{w}_i(n) + \mu_i(n+1)e(n)u(n-i)$$

$$h_i(n+1) = \left(1 - \mu_i(n+1)|u(n-i)|^2\right)h_i(n) + \mu_i(n+1)e(n)u(n-i)$$

Note: Each line is for $i = 0, 1$, to $M-1$, except for the line that computes $e(n)$ and B.

The Meta-Step-Size-Parameter, κ

We started the second part of this chapter in Section 13.8 by stating the ultimate goal in linear adaptive filtering algorithm to be that of discovering an algorithm that is tuning free. That goal has indeed been realized with an experimentally oriented formulation of the Autostep method, summarized in Table 13.4.

Now, the alert reader, examining this table, may well raise the following question:

> What is the justification for fixing the meta-step-size parameter, κ, at the nominal value of 0.01?

Before responding to this question, it is instructive to explain the basic way in which the Autostep method differs from the IDBD algorithm: the parameter κ has to be manually tuned, varying drastically in value from one application of the IDBD algorithm to another.

In the Autostep method, on the other hand, the algorithm summarized in Table 13.4 has been formulated by performing a variety of computer experiments, such that it is essentially insensitive to the tuning of parameter κ. Moreover, it has been found, again experimentally, that the fixed value $\kappa = 0.01$ is a reliable choice.

In effect, setting $\kappa = 0.01$ has rendered the Autostep method assume a near-optimal performance without the need for manual tuning. To conclude this discussion, we may therefore go on to say the following (Mahmood, 2013):

> In operating in a nonstationary environment, when there is nothing known about a tracking application of interest or the requirement is to solve a large-scale tracking problem, the use of $\kappa = 0.01$ as the meta-step-size parameter in the Autostep method is a sound choice.

13.11 COMPUTER EXPERIMENT: MIXTURE OF STATIONARY AND NONSTATIONARY ENVIRONMENTAL DATA

This experiment is based on the system-identification problem described in Section 13.2. Referring to Fig. 13.1, the data generator is described by a first-order Markov process, namely, Eq. (13.1). Correspondingly, the desired response is described by the multiple linear regression model of Eq. (13.2).

For the experiment, dimensionality of the unknown weight vector $\mathbf{w}_o$ in Eqs. (13.1) and (13.2) is chosen to be $M = 20$, which is large enough to make the experiment reasonably challenging. For Eq. (13.2), we have the following:

> Measurement (input) convariance matrix, $\mathbf{R}_u = \mathbf{I}$, when $\mathbf{I}$ is the identity matrix.
>
> Measurement noise variance, $\sigma_v^2 = 1.0$.

Turning next to Eq. (13.1), we have

> Scalar constant, $a = 0.9998$.
>
> Process noise covariance matrix is described as follows:

$$\mathbf{R}_{\omega,\,ii} = \begin{cases} \sigma_\omega^2 & \text{for } i = 0, 1, \ldots, 4 \\ 0 & \text{for } i = 5, 6, \ldots, 19 \end{cases}$$

and

$$\mathbf{R}_{\omega,ij} = 0 \quad \text{for all } i \neq j.$$

Thus, the data generated by the regression model in Fig. 13.1 are made up of a mixture of nonstationary and stationary data.

Two different problems are studied in this experiment:

Problem 1: $\sigma_\omega^2 = 0.1$

With σ_ω^2 denoting the variance of the process noise $\boldsymbol{\omega}(n)$ in Eq. (13.1), the driving force for the process equation is relatively small. Therefore, Problem 1 addresses a mixture of slightly nonstationary and stationary data, which are generated by the first 5 and remaining 15 tap weights of the regression model, respectively.

Problem 2: $\sigma_\omega^2 = 10$

In this second problem, we have a mixture of highly nonstationary data alongside stationary data.

Other points of interest in the experiment are as follows:

1. A sequential data set consisting of 50,000 samples is generated for both Problems 1 and 2, in which the LMS, RLS, and IDBD algorithm, and the Autostep method are tested individually.
2. For all three algorithms and the Autostep method, the tunable adaptation parameter is varied from 10^{-12} to 1.0, with equal logarithmic intervals. Table 13.5 identifies the tunable parameters.
3. For the Autostep method, the time-scale parameter $\tau = 10^{+4}$.
4. For initialization, the estimated tap weights for all three algorithms and the Autostep method are set to zero.
5. The initial step-size parameter, $\mu_i(0)$, for both the IDBD algorithm and Autostep method is set to $\frac{1}{M} = 0.05$, where the model size $M = 20$, as pointed out previously.
6. The root mean-square error (RMSE) is used as the measure of performance, averaged over the last 25,000 samples, for which all three algorithms and the Autostep method would have relaxed to their respective steady-state responses.
7. For a final measure of performance, the RMSE is considered relative to the best LMS algorithm performance averaged over the two problems. To this end, let the RMSE of the IDBD algorithm with meta-step-size parameter κ for Problems 1 and 2

TABLE 13.5 Tunable Parameters

Algorithm or method	Tunable parameter
LMS	Step-size parameter, μ
RLS	$1 - \lambda$, where λ is the exponential weighting factor
IDBD	Meta-step-size parameter, κ
Autostep	Meta-step-size parameter, κ

be denoted by $\text{RMSE}_{\text{IDBD}(\kappa),1}$ and $\text{RMSE}_{\text{IDBD}(\kappa),2}$, respectively. Correspondingly, let $\text{RMSE}_{\text{LMSbest},1}$ and $\text{RMSE}_{\text{LMSbest},2}$ denote the best RMSE of the LMS algorithm (after tuning) for Problems 1 and 2, respectively. Then, our averaged measure of performance for the IDBD algorithm is defined by the following metric:

$$S(\kappa) = \frac{1}{2}\left(\frac{\text{RMSE}_{\text{IDBD}(\kappa),1}}{\text{RMSE}_{\text{LMSbest},1}} + \frac{\text{RMSE}_{\text{IDBD}(\kappa),2}}{\text{RMSE}_{\text{LMSbest},2}}\right). \tag{13.99}$$

In a similar manner, we may define the final measure of performance for the RLS algorithm and the Autostep method. According to Eq. (13.99), the metric $S = 1$ for the LMS algorithm.

Results of the Experiment

Table 13.6 presents the values of the best metric S for each of the three algorithms and the Autostep method. According to this table, the Autostep method comes out with the best averaged metric, S.

Figure 13.4 plots the averaged metric, S, versus the pertinent tunable parameter for each of the algorithms and the Autostep method. Each point in this figure represents how good each algorithm is for a particular value of its tunable adaptation parameter, compared to the best LMS algorithm performance. The results plotted in the figure show that the Autostep method outperforms the three algorithms, particularly the IDBD.

The results presented in Table 13.6 and Figure 13.4 are based on an average of the two different problems. In this context, we may wonder how each of the algorithms and the Autostep method performs on these two problems, addressed separately. With this objective in mind, parts (a) and (b) of Figure 13.5 plot the RMSE for each with

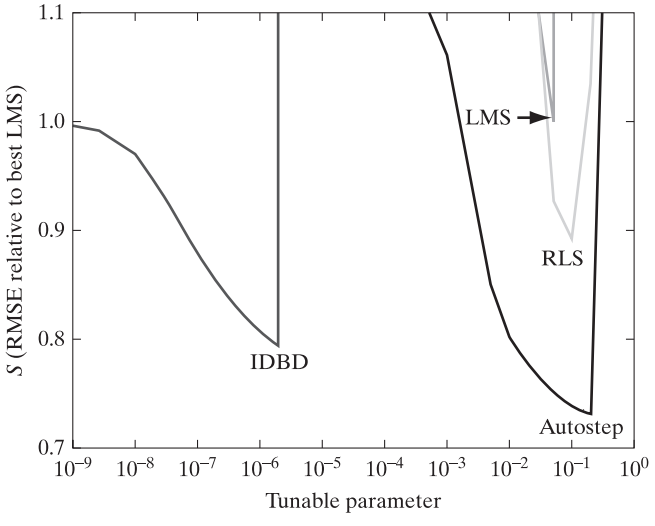


FIGURE 13.4 The averaged metric, S, plotted versus tunable parameters for the LMS, RLS, and IDBD algorithms and the Autostep method.

TABLE 13.6 Parameterization of LMS, RLS, IDBD, and Autostep

Algorithm or method	Adaptation parameter	Averaged metric, S
LMS	$\mu = 0.05$	1
RLS	$1 - \lambda = 0.1$	0.89
IDBD	$\alpha = 2 \times 10^{-6}$	0.79
Autostep	$\alpha = 0.2$	0.73

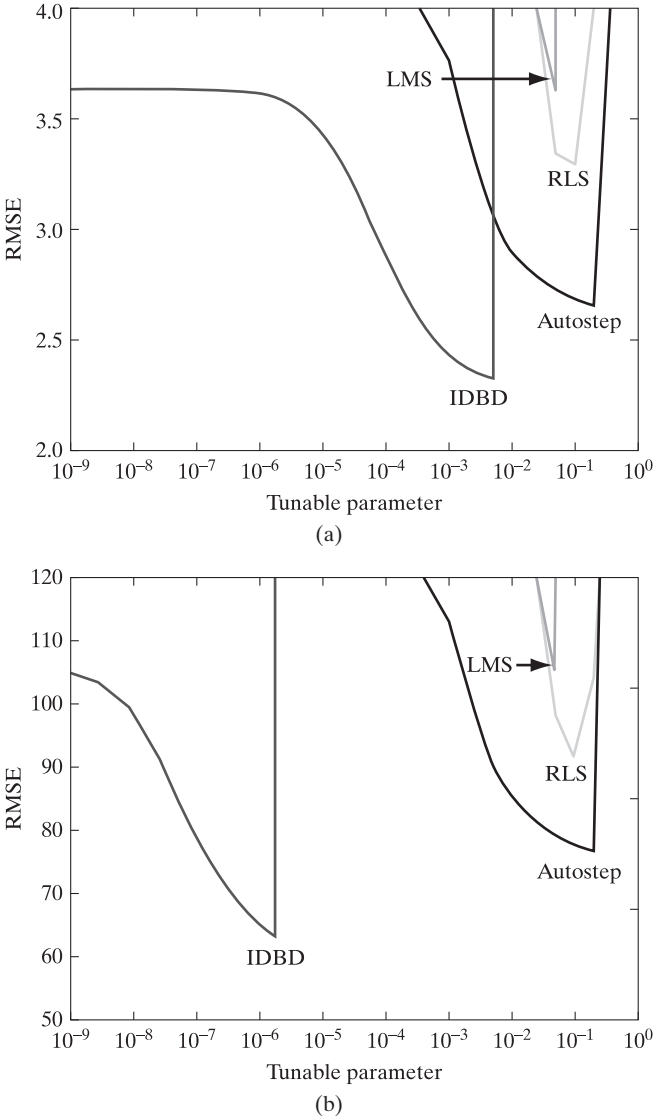


FIGURE 13.5 Experimental results for the LMS, RLS, and IDBD algorithms and the Autostep method for two different problems: (a) slightly stationary environmental data; (b) highly nonstationary environmental data.

parts (a) and (b) referring to Problems 1 and 2, respectively. Two observations may be made on the results shown in Figure 13.5:

1. The IDBD algorithm and the Autostep method outperform both the RLS and LMS algorithms.
2. In an overall sense, the recommended value for the meta-step-size parameter, κ, for the Autostep method is $\kappa = 0.01$ for both Problems 1 and 2, for which the RMSE is minimal. In direct contrast, the IDBD algorithm requires a vast range of recommended values for κ, changing from $\kappa = 10^{-3}$ for Problem 1 all the way down to $\kappa = 10^{-6}$ for Problem 2 so as to realize minimal RMSE for both problems.

   Simply stated:

   The IDBD algorithm requires manual tuning when we go from Problem 1 to Problem 2, whereas the Autostep method has the built-in capability to go from Problem 1 to Problem 2 without tuning.

What Have We Learned from This Experiment?

The two problems studied in this experiment are representative of a typical real-world problem. In problems of this kind, we do not know whether the given data are stationary or nonstationary. Ideally, for a stationary scenario, the step-size parameter in an adaptive filtering algorithm should get smaller as the algorithm relaxes to a steady-state response. On the other hand, for a nonstationary scenario, the step-size parameter should remain large to maintain tracking of the data.

The experiment demonstrates that having the capability to adapt the step-size parameter along different directions in response to statistical variations in environmental data does matter. To be specific, neither the LMS nor the RLS algorithm is able to tackle statistical variations when it is confronted with a mixture of stationary and nonstationary data. Moreover, confronted with such a situation in practice, the IDBD algorithm requires manual tuning of the step-size parameter, whereas the Autostep method does not.

13.12 SUMMARY AND DISCUSSION

In this chapter, we studied the important practical issue of how to deal with adaptive filtering of nonstationary data. To do so, an adaptive filter would have to *track* statistical variations in the environment responsible for generating the data, having reached a steady-state response (i.e., convergence process has been completed).

In the first part of the chapter, we compared the LMS and RLS algorithms in dealing with nonstationary data, using the system-identification problem to do the comparison. The conclusion to be drawn from that study is summarized as follows:

1. Neither the LMS nor the RLS algorithm has the built-in ability to monopolize good tracking behavior.
2. One or the other of these two adaptive filtering algorithms is the preferred choice for tracking a nonstationary environment, depending on the nonstationary behavior of the environment responsible for data generation.

3. Regardless of which one of these two algorithms is the preferred choice, the adaptation parameter needs to be manually tuned.

In the second part of the chapter, we described the IDBD algorithm and the Autostep method. The IDBD algorithm builds on the LMS algorithm by replacing the fixed step-size parameter of the LMS with a vectorized step-size parameter, such that each elemental step-size parameter correlates with a particular feature of the input data. The rationale behind this modification of the LMS algorithm is summed up as follows:

- Input data that are likely to be irrelevant should be given small step sizes.
- Input data that are likely to be relevant, and therefore important, should be given relatively large step sizes.

Typically, dimensionality of the step-size vector is chosen to be the same as that of the tap-weight vector in the IDBD algorithm.

The IDBD algorithm is an example of a learning-within-learning strategy: In catering to a step-size vector, we now have a new parameter, called the meta-step-size parameter. As it is with the traditional LMS algorithm, the meta-step-size parameter of the IDBD algorithm would have to be manually tuned. To get around the need for this manual tuning, we may look to the Autostep method that builds on the IDBD algorithm. To be specific, the Autostep method is formulated by deriving insights, and therefore rules, from many different experiments performed on the IDBD algorithm, and then integrating them into a composite algorithm. The computer experiments presented in Section 13.11 support practical applicability of the Autostep as a quasi-tuning-free method for addressing unknown nonstationary environments as well as solving large-scale tracking problems.

In a related context, we may conclude the chapter with the following question on the ultimate in adaptive filtering:

> How do we automatically tune an adaptive filtering algorithm, preferably in a rigorous manner, by building on:
>
> - the method of stochastic gradient descent, to be assured of a linear law of computational complexity, and
> - the learning-within-learning strategy, exemplified by the IDBD algorithm?

Presently, the Autostep method is the only known adaptive filtering method that has addressed this question of practical importance, albeit in a heuristic manner.

PROBLEMS

1. Find the quantization error that arises in computing the updated inverse-correlation matrix $\mathbf{P}(n)$. How are we justified in making this assertion?
2. Explain in detail the evaluation process for the tracking capability of the RLS algorithm for the system identification problem described in Fig. 13.2, subject to the condition that the model parameter a is close to unity.

3. The weight-error vector $\boldsymbol{\varepsilon}(n)$ may be expressed as the sum of the weight vector noise $\boldsymbol{\varepsilon}_1(n)$ and weight vector lag $\boldsymbol{\varepsilon}_2(n)$. Show that

$$\mathbb{E}[\boldsymbol{\varepsilon}_1^{\mathrm{H}}(n)\boldsymbol{\varepsilon}_2(n)] = \mathbb{E}[\boldsymbol{\varepsilon}_2^{\mathrm{H}}(n)\boldsymbol{\varepsilon}_1(n)] = 0.$$

Using the assumptions of Section 13.2, show that

$$\mathbb{E}[\|\boldsymbol{\varepsilon}(n)\|^2] = \mathbb{E}[\|\boldsymbol{\varepsilon}_1(n)\|^2] + \mathbb{E}[\|\boldsymbol{\varepsilon}_2(n)\|^2].$$

4. Continuing with Problem 3, use the assumptions of Section 13.2 to show that

$$\mathbb{E}[\boldsymbol{\varepsilon}_1^{\mathrm{H}}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}_1(n)] = \mathrm{tr}[\mathbf{R}_u\mathbf{K}_1(n)],$$
$$\mathbb{E}[\boldsymbol{\varepsilon}_2^{\mathrm{H}}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}_2(n)] = \mathrm{tr}[\mathbf{R}_u\mathbf{K}_2(n)],$$

and

$$\mathbb{E}[\boldsymbol{\varepsilon}_1^{\mathrm{H}}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}_2(n)] = \mathbb{E}[\boldsymbol{\varepsilon}_2^{\mathrm{H}}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\varepsilon}_1(n)] = 0,$$

where $\mathbf{u}(n)$ is the input vector, assumed to be of zero mean; $\mathbf{R}_u$ is the correlation matrix of $\mathbf{u}(n)$; and $\mathbf{K}_1(n)$ and $\mathbf{K}_2(n)$ are the correlation matrices of $\boldsymbol{\varepsilon}_1(n)$ and $\boldsymbol{\varepsilon}_2(n)$, respectively. How is the correlation matrix $\mathbf{K}(n)$ of $\boldsymbol{\varepsilon}(n)$ related to $\mathbf{K}_1(n)$ and $\mathbf{K}_2(n)$?

5. Derive the mean-square deviation of the LMS algorithm and show the optimum value of the step-size parameter for which the mean square deviation attains its minimum value.

6. Evaluate the maladjustment of the LMS algorithm for the system identification scenario described in Fig. 13.2.

7. Describe the IDBD algorithm and the autostep method.

8. List the various types of approaches to implement the learning-within-learning scheme of Fig. 13.3.

9. Consider a real-valued nonstationary problem where, at each adaptation cycle n, a single input $u(n)$ and corresponding actual response $y(n)$ are observed. The actual response (output) is evaluated as follows:

$$y(n) = w(n)u(n) + \nu(n),$$

where $w(n)$ is the target weight and $\nu(n)$ is drawn from a white-noise process of zero mean and unit variance. The target weight alternates between −1 and +1 every 10 adaptation cycles. The input $u(n) = 10$ is always maintained.

(a) Use the IDBD algorithm to learn the output from the input, under the following conditions:
- **(i)** $w(0) = 0$,
- **(ii)** step-size parameter, $\mu(0) = 0$, and
- **(iii)** meta-step-size parameter, $\kappa = 0.001$.

Convince yourself that the IDBD algorithm learns the output well; to do this, plot the estimated weight, $\hat{w}(n)$, over adaptation cycle n, wherein, after the transient response is over, $\hat{w}(n)$ should resemble the target weight, $w(n)$, approximately (i.e., alternating between −1 and +1 every 10 adaptation cycles).

(b) Change the problem description minimally, keeping the IDBD parameter unchanged, so that the IDBD algorithm *diverges*. For example, you may increase the value of the input by 10-fold or increase the variances of the white noise by 100-fold. Hence, respond to the following two questions for each example case:
- **(i)** Why does the IDBD algorithm diverge?
- **(ii)** What is the minimal change required in IDBD parameters (e.g., the initial step-size parameter, $\mu(0)$, or the meta-step-size-parameter, κ) so that the divergence is prevented?

10. Conduct the experiment described in Problem 9 for the Autostep method. This time, set

initial condition, $\hat{w}(0) = 0$,
meta-step-size parameter, $\kappa = 0.01$,
initial value of step-size parameter, $\mu(0) = 0.1$,
forgetting factor, $\gamma = 0.0001$.

(a) Is it possible for the Autostep method to diverge at all?
(b) How do the characteristics of the Autostep method differ from those of the IDBD algorithm in this problem?

Justify your answers.

11. Suppose that the step-size parameter of the Autostep method is initialized to a positive number.
(a) Is it *theoretically* possible for the step-size parameter to go to the value zero, ever?
(b) Is such an event *practically* possible on a computer due to finite-precision effects, leading to underflow?
(c) If the answer to part (b) is yes, what measure can be taken to prevent the step-size parameter from going to the value zero on a computer?
(d) Can a similar problem arise in the other Autostep parameters, namely $\mathbf{w}$, h, and γ, due to overflow or underflow?

Justify your answers.

To clarify the answer to part (d) of the problem, the following two points are noteworthy:

(i) *Arithmetic underflow* refers to a condition in a computer program which can occur when the net result of a floating-point operation is smaller in magnitude (i.e., closer to zero) than the smallest value representable as a normal floating-point number in the target data of interest.
(ii) *Arithmetic overflow* refers to a condition in a computer program, which can occur when a calculation produces a result greater in magnitude than what a given register or storage location in the program can store or represent.

Computer Experiments

Note for Problems 12, 13, and 14

The input data for all three problems are stationary. The unknown weight vector $\mathbf{w}$ is therefore fixed, and the process noise $\boldsymbol{\omega}(n)$ is zero.

12. *Stationary symmetric input data.* The input covariance matrix $\mathbf{R}_u$ for this experiment is described as follows:

$$\mathbf{R}_{u,ii} = 1 \quad \text{for all } i = 0, 1, \ldots, 19$$

and

$$\mathbf{R}_{u,ij} = 0 \quad \text{for all } i \neq j.$$

In other words, all the inputs in this experiment have the same variance, with the result that we have a *spherical cost function* to be minimized.
(a) Generate 25,000 sequential samples, and test each of the LMS, RLS, and IDBD algorithms, and the Autostep method on the data separately.
(b) For each of the algorithms and the Autostep method, vary the tunable step-size parameter from 10^{-12} to 1 with equal logarithmic intervals.

(c) For the Autostep method, set the time-scale parameter $\tau = 10^4$, and the initial step-size, $\mu(0)$, equal to $(1/M) = 0.05$, the latter being the same for the IDBD algorithm.
(d) As usual, set the initial values of the estimated tap weights to zero.
(e) Use the RMSE as the measure of performance, averaged over the entire set of 25,000 samples.

Since in this experiment the data are stationary, we only consider the transient response of each algorithm and the Autostep method. With this objective in mind, do the following:

(i) Tabulate the best RMSE for each of the four algorithms.
(ii) Plot the RMSE versus the tunable step-size parameter pertinent to each algorithm.

Finally, summarize your findings by describing how the LMS, RLS, and IDBD algorithms, and the Autostep method compare with each other.

13. *Stationary asymmetric input data.* The specifications of this second experiment are the same as those of Problem 12. However, assume that the input covariance matrix $\mathbf{R}_u$ is such that

$$\mathbf{R}_{u,ii} = \begin{cases} 100 & \text{for } i = 0, 1, \ldots, 4 \\ 1 & \text{for } i = 5, 6, \ldots, 19 \end{cases}$$

and

$$\mathbf{R}_{u,ij} = 0 \quad \text{for all } i \neq j.$$

Accordingly, the different inputs have different variances, thereby resulting in a cost function that is *elliptical*. Also, set the initial step-size parameter, $\mu(0)$, for the IDBD algorithm and the Autostep method to $1/(100 \times M) = 0.0005$.

Incorporating the changes described above, repeat all the other steps, tabulation of the best RMSE and plotting the RMSE versus the tunable parameter, as described in Problem 12.

14. *Stationary, correlated input data.* The specifications of this third experiment are the same as those of Problem 12, except for the following modification: The input covariance matrix, $\mathbf{R}_u$, is a positive definitive matrix, generated randomly such that we have

$$\mathbf{R}_{u,ii} = 1, \quad \text{for } i = 0, 1, \ldots, 19 \text{ and}$$
$$\mathbf{R}_{u,ij} = \mathbf{R}_{\omega,ji} \in [-1, +1] \quad \text{for all } i \neq j.$$

In this case, all the input data have the same variance but they are *correlated*, thereby producing a somewhat *rotated elliptical cost function*. Otherwise, all the other specifications are exactly the same as those described in Problem 12.

Incorporating the changes described above, repeat all the other steps, tabulation of the best RMSE and plotting the RMSE versus the tunable parameter, as described in Problem 12.

15. *Nonstationary input data.* In this fourth and last computer experiment, dealing with a nonstationary environment, everything is the same as that described in Problem 12, but with the following changes:

input covariance matrix, $\mathbf{R}_u = \mathbf{I}$,
measurement noise variance, $\sigma_v^2 = 1.0$,
scaler parameter, $a = 0.9998$.

Process noise covariance matrix $\mathbf{R}_\omega$ is such that we have

$$\mathbf{R}_{\omega,ii} = \begin{cases} 100 & \text{for } i = 0, 1, \ldots, 4 \\ 1 & \text{for } i = 5, 6, \ldots, 19 \end{cases}$$

and

$$\mathbf{R}_{\omega,ij} = 0 \quad \text{for all } i \neq j.$$

Thus, in this experiment, we find that all the elements are nonstationary, but five of the target tap weights have rates of nonstationarity different from those of the remaining 15 tap weights. Otherwise, the experiment follows the same specifications described in Problem 12.

Incorporating the changes described above, repeat all the other steps, tabulation of the best RMSE and plotting of the RMSE versus the tunable parameter, as discussed in Problem 12.

CHAPTER 14

Kalman Filters

In this chapter, we continue the study of tracking in a nonstationary environment, which was a focus of attention in the preceding chapter. This time, however, we broaden the scope of the study by developing the underlying ideas of *Kalman filters* (Kalman, 1960).

To pave the ground for this development, we begin by introducing the notion of *state* to provide the basis for a mathematical exposition of Kalman filtering. This notion plays a key role in formulating the *state-space model*, which embodies the following pair of equations:

- *System equation*, which describes evolution of the state across time.
- *Measurement equation*, which describes the dependence of measurements on the state.

In a sense, we may view this pair of equations as a generalization of the first-order Markov model and the multiple linear regression model, described by Eqs. (13.1) and (13.2) in Chapter 13, respectively. Simply put, the state-space model is central to the formulation of the Kalman filter.

Another novel feature of the Kalman filter is that its solution is computed *recursively*, applying without modification to stationary as well as nonstationary environments. In particular, each updated estimate of the state is computed from the previous estimate and the new input data, so only the previous estimate requires storage. In addition to eliminating the need for storing the entire past measured data, the Kalman filter is computationally more efficient than computing the estimate directly from all of those past data at each adaptation cycle of the filtering process, as it is with the Wiener filter, discussed in Chapter 2.

The study of Kalman filtering provides the mathematical basis for estimating the unknown "hidden" state of a linear time-varying system in a Gaussian environment, given a set of measurements, whereby the state estimation is performed in a recursive manner. This recursive computation makes Kalman filtering well suited for data processing on a computer, just as the least-mean-square (LMS) and recursive least-squares (RLS) algorithms are. Most importantly, the Kalman filter together with its variants and extensions provides an indispensable tool for solving target-tracking problems in signal processing and control.

However, insofar as this chapter is concerned, primary interest in Kalman filtering is motivated by the fact that it provides a unifying framework for the family of RLS adaptive filtering algorithms:

- The conventional RLS algorithm, discussed in Chapter 10.
- The square-root RLS algorithms, to be discussed in Chapter 15.
- The order-recursive RLS algorithms, to be discussed in Chapter 16.

We begin the study of Kalman filtering by solving the recursive minimum mean-square estimation problem for the simple example of a scalar random variable.

14.1 RECURSIVE MINIMUM MEAN-SQUARE ESTIMATION FOR SCALAR RANDOM VARIABLES

Let us assume that, on the basis of a complete set of measured random variables $y(1)$, $y(2), \ldots, y(n-1)$, starting with the first measurement at adaptation cycle 1 and extending up to and including adaptation cycle $n-1$, we have found the minimum mean-square estimate $\hat{x}(n-1|\mathcal{Y}_{n-1})$ of a related zero-mean random variable $x(n-1)$. We are assuming that the measurement at (or before) $n=0$ is zero. The space spanned by the measurement $y(1), \ldots, y(n-1)$ is denoted by $\mathcal{Y}_{n-1}$. Suppose that we now have an additional measurement $y(n)$ at adaptation cycle n, and the requirement is to compute an *updated* estimate $\hat{x}(n|\mathcal{Y}_n)$ of the related random variable $x(n)$, where $\mathcal{Y}_n$ denotes the space spanned by $y(1), \ldots, y(n)$. We may do this computation by storing the *past* measurement, $y(1), y(2), \ldots, y(n-1)$ and then redoing the whole problem with the available data $y(1), y(2), \ldots, y(n-1), y(n)$, including the new measurement. Computationally, however, it is much more efficient to use a *recursive estimation procedure*, in that we *store* the previous estimate $\hat{x}(n-1|\mathcal{Y}_{n-1})$ and exploit it to compute the updated estimate $\hat{x}(n|\mathcal{Y}_n)$ in light of the new measurement $y(n)$. There are several ways of developing the algorithm to do this recursive estimation. We will use the notion of *innovations* (Kailath, 1968, 1970), the origin of which may be traced back to Kolmogorov (1939).

We define the forward prediction error

$$f_{n-1}(n) = y(n) - \hat{y}(n|\mathcal{Y}_{n-1}), \quad n = 1, 2, \ldots, \tag{14.1}$$

where $\hat{y}(n|\mathcal{Y}_{n-1})$ is the *one-step prediction* of the measured random variable $y(n)$ at adaptation cycle n, using *all* past measurements available up to and including adaptation cycle $n-1$. The past measurements used in this estimation are $y(1), y(2), \ldots, y(n-1)$, so the order of the prediction equals $n-1$. We may view $f_{n-1}(n)$ as the output of a forward prediction-error filter of order $n-1$ and with the filter input fed by the time series $y(1), y(2), \ldots, y(n)$. Note that the *prediction order* $n-1$ *increases linearly with* n. According to the principle of orthogonality, the prediction error $f_{n-1}(n)$ is orthogonal to all past measurements $y(1), y(2), \ldots, y(n-1)$ and may therefore be regarded as a *measure* of the new information in the random variable $y(n)$ measured at adaptation cycle n—hence the name "innovation." The fact is that the measurement $y(n)$ does not itself convey completely new information, since the predictable part, $\hat{y}(n|\mathcal{Y}_{n-1})$, is already completely determined by the past measurements $y(1), y(2), \ldots, y(n-1)$. Rather, the part of the measurements $y(n)$ that is really new is contained in the forward prediction error $f_{n-1}(n)$. We may therefore refer to this prediction error as the *innovation* and, for simplicity of notation, write

$$\alpha(n) = f_{n-1}(n), \quad n = 1, 2, \ldots. \tag{14.2}$$

The innovation $\alpha(n)$ has several important properties, described next.

Property 1. *The innovation $\alpha(n)$ associated with the measured random variable $y(n)$ is orthogonal to the past measurements $y(1), y(2), \ldots, y(n-1)$, as shown by*

$$\mathbb{E}[\alpha(n)y^*(k)] = 0, \quad 1 \le k \le n-1, \tag{14.3}$$

where the asterisk denotes *complex conjugation.* This is simply a restatement of the principle of orthogonality.

Property 2. *The innovations $\alpha(1), \alpha(2), \ldots, \alpha(n)$ are orthogonal to each other, as shown by*

$$\mathbb{E}[\alpha(n)\alpha^*(k)] = 0, \quad 1 \le k \le n-1. \tag{14.4}$$

This is a restatement of the fact that [see part (e) of Problem 20, Chapter 3]

$$\mathbb{E}[f_{n-1}(n)f_{k-1}^*(k)] = 0, \quad 1 \le k \le n-1.$$

Equation (14.4), in effect, states that the innovation process $\alpha(n)$, described by Eqs. (14.1) and (14.2), is *white*.

Property 3. *There is a one-to-one correspondence between the measured data $\{y(1), y(2), \ldots, y(n)\}$ and the innovations $\{\alpha(1), \alpha(2), \ldots, \alpha(n)\}$, in that the one sequence may be obtained from the other by means of a causal and causally invertible filter without any loss of information. We may thus write*

$$\{y(1), y(2), \ldots, y(n)\} \rightleftharpoons \{\alpha(1), \alpha(2), \ldots, \alpha(n)\}. \tag{14.5}$$

To prove this property, we use a form of the Gram–Schmidt orthogonalization procedure (described in Chapter 3). The procedure assumes that the measurements $y(1), y(2), \ldots, y(n)$ are linearly independent in an algebraic sense. We first put

$$\alpha(1) = y(1), \tag{14.6}$$

where it is assumed that $\hat{y}(1|\mathcal{Y}_0)$ is zero. Next, we put

$$\alpha(2) = y(2) + a_{1,1}y(1). \tag{14.7}$$

The coefficient $a_{1,1}$ is chosen such that the innovations $\alpha(1)$ and $\alpha(2)$ are orthogonal, as shown by

$$\mathbb{E}[\alpha(2)\alpha^*(1)] = 0. \tag{14.8}$$

This requirement is satisfied by choosing

$$a_{1,1} = -\frac{\mathbb{E}[y(2)y^*(1)]}{\mathbb{E}[y(1)y^*(1)]}. \tag{14.9}$$

Except for the minus sign, $a_{1,1}$ is a partial correlation coefficient in that it equals the cross-correlation between the measurements $y(2)$ and $y(1)$, normalized with respect to the mean-square value of $y(1)$.

Next, we put

$$\alpha(3) = y(3) + a_{2,1}y(2) + a_{2,2}y(1), \tag{14.10}$$

where the coefficients $a_{2,1}$ and $a_{2,2}$ are chosen such that $\alpha(3)$ is orthogonal to both $\alpha(1)$ and $\alpha(2)$, and so on. Thus, in general, we may express the transformation of the measurements $y(1), y(2), \ldots, y(n)$ into the innovations $\alpha(1), \alpha(2), \ldots, \alpha(n)$ by writing

$$\begin{bmatrix} \alpha(1) \\ \alpha(2) \\ \vdots \\ \alpha(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ a_{1,1} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,n-1} & a_{n-1,n-2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(n) \end{bmatrix}. \tag{14.11}$$

The nonzero elements of row k of the *lower triangular transformation matrix* on the right-hand side of Eq. (14.11) are deliberately denoted as $a_{k-1,k-1}, a_{k-1,k-2} \ldots, 1$, where $k = 1, 2, \ldots, n$. These elements represent the coefficients of a *forward prediction-error filter* of order $k-1$. Note that $a_{k,0} = 1$ for all k. Accordingly, given the measurements $y(1), y(2), \ldots, y(n)$, we may compute the innovations $\alpha(1), \alpha(2), \ldots, a(n)$. There is no loss of information in the course of this transformation, since we may recover the original measurements $y(1), y(2), \ldots, y(n)$ from the innovations $\alpha(1), \alpha(2), \ldots, \alpha(n)$. This we do by premultiplying both sides of Eq. (14.11) by the inverse of the lower triangular transformation matrix. This matrix is nonsingular, since its determinant is unity for all n. The transformation is therefore reversible.

Using Eq. (14.5), we may thus write

$$\hat{x}(n|\mathcal{Y}_n) = \text{minimum mean-square estimate of } x(n), \text{ given the measurements } y(1), y(2), \ldots, y(n)$$

or, equivalently,

$$\hat{x}(n|\mathcal{Y}_n) = \text{minimum mean-square estimate of } x(n), \text{ given the innovations } \alpha(1), \alpha(2), \ldots, \alpha(n).$$

We define the estimate $\hat{x}(n|\mathcal{Y}_n)$ as a linear combination of the innovations $\alpha(1)$, $\alpha(2), \ldots, \alpha(n)$:

$$\hat{x}(n|\mathcal{Y}_n) = \sum_{k=1}^{n} b_k \alpha(k). \tag{14.12}$$

Note that the b_k are to be determined. With the innovations $\alpha(1), \alpha(2), \ldots, \alpha(n)$ orthogonal to each other and the b_k chosen to minimize the mean-square value of the estimation error $x(n) - \hat{x}(n|\mathcal{Y}_n)$, we find that

$$b_k = \frac{\mathbb{E}[x(n)\alpha^*(k)]}{\mathbb{E}[\alpha(k)\alpha^*(k)]}, \qquad 1 \le k \le n. \tag{14.13}$$

Isolating the term corresponding to $k = n$, we rewrite Eq. (14.12) in the form

$$\hat{x}(n|\mathcal{Y}_n) = \sum_{k=0}^{n-1} b_k \alpha(k) + b_n \alpha(n), \tag{14.14}$$

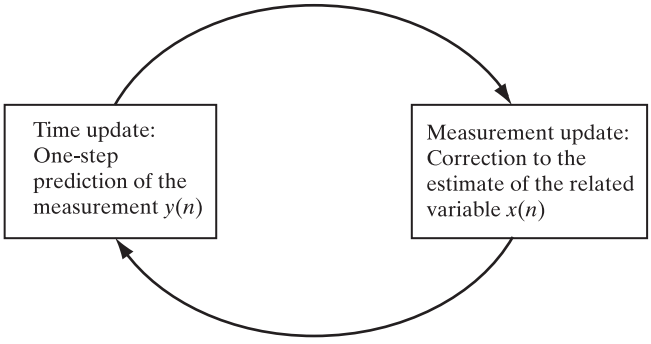


FIGURE 14.1 Graphical description of the solution to recursive minimum mean-square estimation portrayed as a predictor–corrector.

where

$$b_n = \frac{\mathbb{E}[x(n)\alpha^*(n)]}{\mathbb{E}[\alpha(n)\alpha^*(n)]}. \tag{14.15}$$

However, by definition, the summation term on the right-hand side of Eq. (14.14) equals the previous estimate, $\hat{x}(n-1|\mathcal{Y}_{n-1})$. We may thus express the recursive estimation algorithm that we are seeking as

$$\hat{x}(n|\mathcal{Y}_n) = \hat{x}(n-1|\mathcal{Y}_{n-1}) + b_n\alpha(n), \tag{14.16}$$

where b_n is defined by Eq. (14.15). Thus, by adding a *correction term* $b_n\alpha(n)$ to the previous estimate $\hat{x}(n-1|\mathcal{Y}_{n-1})$, with the correction proportional to the innovation $\alpha(n)$, we get the updated estimate $\hat{x}(n|\mathcal{Y}_n)$.

Equations (14.1), (14.3), (14.15), and (14.16) show that the underlying structure of a recursive minimum mean-square-error estimator is in the form of a *predictor–corrector*, as depicted in Fig. 14.1. This structure consists of two basic steps:

1. The use of measurements to compute a forward prediction error termed "innovation."
2. The use of the innovation to update (i.e., correct) the minimum mean-square estimate of a random variable related linearly to the measurements.

Equipped with this simple and yet powerful structure, shown in Fig. 14.1, we are now ready to study the more general Kalman filtering problem.

14.2 STATEMENT OF THE KALMAN FILTERING PROBLEM

Consider a *linear, discrete-time dynamic model* described by the signal-flow graph shown in Fig. 14.2. The time-domain description of the system presented here offers the following advantages (Gelb, 1974):

- Mathematical and notational convenience.
- A close relationship to physical reality.
- A useful basis for accounting for the statistical behavior of the original system.

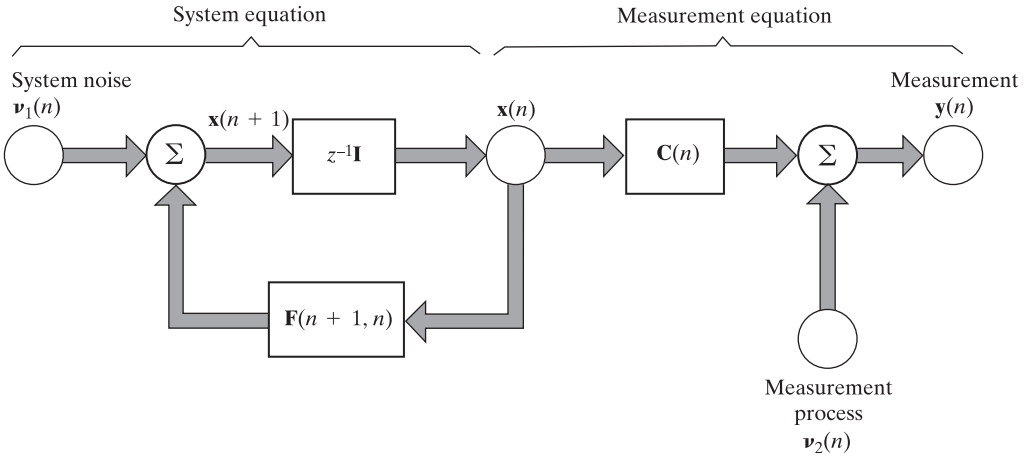


FIGURE 14.2 Signal-flow graph representation of a linear, discrete-time dynamic model.

The concept of *state* is fundamental to this formulation. The *state vector*, or simply *state*, denoted by $\mathbf{x}(n)$ in Fig. 14.2, is defined as the minimal set of data that is sufficient to uniquely describe the unforced dynamic behavior of the system. In other words, the state comprises the fewest data on the past behavior of the system that are needed to predict its future behavior. Typically, the state $\mathbf{x}(n)$, assumed to be of dimension M, is unknown. To estimate it, we use a set of observed data, denoted by the vector $\mathbf{y}(n)$ in the figure. The *measurement vector*, or simply *measurement*, $\mathbf{y}(n)$ is assumed to be of dimension N, which is typically different from the dimension M.

In mathematical terms, the signal-flow graph of Fig. 14.2 embodies the following pair of equations:

1. *System equation*,

$$\mathbf{x}(n+1) = \mathbf{F}(n+1, n)\mathbf{x}(n) + \boldsymbol{\nu}_1(n). \tag{14.17}$$

In this equation, the M-by-1 vector $\boldsymbol{\nu}_1(n)$ represents *system noise*, modeled as a zero-mean, white-noise process whose correlation matrix is defined by

$$\mathbb{E}[\boldsymbol{\nu}_1(n)\boldsymbol{\nu}_1^{\mathrm{H}}(k)] = \begin{cases} \mathbf{Q}_1(n), & n = k \\ \mathbf{O}, & n \neq k \end{cases}, \tag{14.18}$$

where the superscript H denotes *Hermitian transposition* (i.e., transposition combined with complex conjugation). *The system equation (14.7), also referred to as the process equation, models an unknown physical stochastic phenomenon described by the state* $\mathbf{x}(n)$ *as the output of a linear dynamic model excited by the white noise* $\boldsymbol{\nu}_1(n)$, *as depicted in the left-hand portion of Fig. 14.2*. The linear dynamic model is uniquely characterized by the feedback connection of two units: the *transition matrix*, denoted by $\mathbf{F}(n+1, n)$, and the *memory unit*, denoted by $z^{-1}\mathbf{I}$, where

z^{-1} is the unit delay and $\mathbf{I}$ is the M-by-M identity matrix. The transition matrix $\mathbf{F}(n+1, n)$ indicates a transition of the system from adaptation cycle n to $n+1$; it has the following properties:

(i) *Product rule:*

$$\mathbf{F}(n, m)\mathbf{F}(m, l) = \mathbf{F}(n, l),$$

where l, m, and n are integers.

(ii) *Inverse rule:*

$$\mathbf{F}^{-1}(n, m) = \mathbf{F}(m, n),$$

where m and n are integers.

From these two rules, we readily see that

$$\mathbf{F}(n, n) = \mathbf{I}.$$

2. *Measurement equation*, which describes the measurement vector as

$$\mathbf{y}(n) = \mathbf{C}(n)\mathbf{x}(n) + \boldsymbol{\nu}_2(n), \tag{14.19}$$

where $\mathbf{C}(n)$ is a known N-by-M *measurement matrix*. The N-by-1 vector $\boldsymbol{\nu}_2(n)$ is called *measurement noise*, modeled as a zero-mean, white-noise process whose correlation matrix is

$$\mathbb{E}[\boldsymbol{\nu}_2(n)\boldsymbol{\nu}_2^{\mathrm{H}}(k)] = \begin{cases} \mathbf{Q}_2(n), & n = k \\ \mathbf{O}, & n \neq k \end{cases}. \tag{14.20}$$

The measurement equation (14.19) relates the measurable output of the system $\mathbf{y}(n)$ *to the state* $\mathbf{x}(n)$, *as depicted in the right-hand portion of Fig. 14.2.*

It is assumed that $\mathbf{x}(0)$, the initial value of the state, is uncorrelated with both $\boldsymbol{\nu}_1(n)$ and $\boldsymbol{\nu}_2(n)$ for $n \geq 0$. The noise vectors $\boldsymbol{\nu}_1(n)$ and $\boldsymbol{\nu}_2(n)$ are statistically independent, so we may write

$$\mathbb{E}[\boldsymbol{\nu}_1(n)\boldsymbol{\nu}_2^{\mathrm{H}}(k)] = \mathbf{O} \qquad \text{for all } n \text{ and } k. \tag{14.21}$$

The Kalman filtering problem, namely, the problem of jointly solving the system and measurement equations for the unknown state in an optimal manner, may now be formally stated as follows:

> Use the entire measured data, consisting of the measurements $\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)$, to find, for each $n \geq 1$, the minimum mean-square estimate of the state $\mathbf{x}(i)$.

The problem is called *filtering* if $i = n$, *prediction* if $i > n$, and *smoothing* if $1 \leq i < n$. In this chapter, we shall be concerned only with filtering and prediction, which are closely related.

14.3 THE INNOVATIONS PROCESS

To solve the Kalman filtering problem, we will use an approach based on the innovations process.[1] Building on the notion of innovation introduced in Section 14.1, let the vector $\hat{\mathbf{y}}(n|\mathcal{Y}_{n-1})$ denote the minimum mean-square estimate of the measurement $\mathbf{y}(n)$ at adaptation cycle n, given all the past values of the measurements starting at adaptation cycle $n=1$ and extending up to and including adaptation cycle $n-1$. These past values are represented by the measurements $\mathbf{y}(1), \mathbf{y}(n), \ldots, \mathbf{y}(n-1)$, which span the vector space $\mathcal{Y}_{n-1}$. In light of Eqs. (14.1) and (14.2), we define the *innovations process* associated with $\mathbf{y}(n)$ as

$$\boldsymbol{\alpha}(n) = \mathbf{y}(n) - \hat{\mathbf{y}}(n|\mathcal{Y}_{n-1}), \qquad n = 1, 2, \ldots. \tag{14.22}$$

The M-by-1 vector $\boldsymbol{\alpha}(n)$ represents the new information in the measurement $\mathbf{y}(n)$.

Generalizing the results of Eqs. (14.3), (14.4), and (14.5), we find that the innovations process $\boldsymbol{\alpha}(n)$ has the following properties:

1. The innovations process $\boldsymbol{\alpha}(n)$, associated with the measurement $\mathbf{y}(n)$ at adaptation cycle n, is orthogonal to all past measurements $\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n-1)$, as shown by

$$\mathbb{E}[\boldsymbol{\alpha}(n)\mathbf{y}^{\mathrm{H}}(k)] = \mathbf{O}, \qquad 1 \le k \le n-1. \tag{14.23}$$

2. The innovations process consists of a sequence of vector random variables that are orthogonal to each other, as shown by

$$\mathbb{E}[\boldsymbol{\alpha}(n)\boldsymbol{\alpha}^{\mathrm{H}}(k)] = \mathbf{O}, \qquad 1 \le k \le n-1. \tag{14.24}$$

3. There is a one-to-one correspondence between the sequence of vector random variables $\{\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)\}$, representing the measured data, and the sequence of vector random variables $\{\boldsymbol{\alpha}(1), \boldsymbol{\alpha}(2), \ldots, \boldsymbol{\alpha}(n)\}$, representing the innovations process, in that the one sequence may be obtained from the other by means of linear stable operators without loss of information. Thus, we may state that

$$\{\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)\} \Longleftrightarrow \{\boldsymbol{\alpha}(1), \boldsymbol{\alpha}(2), \ldots, \boldsymbol{\alpha}(n)\}. \tag{14.25}$$

To form the sequence of vector random variables defining the innovations process, we may use a Gram–Schmidt orthogonalization procedure similar to that described

[1]The original derivation of the Kalman filter presented in the 1960 classic paper written by Kalman himself was based on the *orthogonal projection theorem*. For the case of scalar random variables, this theorem may be stated as follows (Doob, 1953; Kalman, 1960):

Let $x(n)$ and $y(n)$ denote scalar random processes, both with zero mean; that is,

$$\mathbb{E}[x(n)] = \mathbb{E}[y(n)] = 0 \qquad \text{for all } n.$$

Suppose we are given the measured random variables $y(1), y(2), \ldots, y(n)$, and suppose also that either of the following two conditions holds:

(i) The random processes $x(n)$ and $y(n)$ are Gaussian.

(ii) The optimal estimate is restricted to be a linear function of the measured random variables, and the cost function is defined as the mean-square value of the difference between $x(n)$ and its estimate.

Then the optimal estimate of $x(n)$ for the given measurements $y(1), y(2), \ldots, y(n)$ is the orthogonal projection of $x(n)$ on the linear space $\mathcal{Y}(n)$ spanned by these measurements.

In contrast to Kalman's approach, the derivation of the Kalman filter presented herein follows the *innovations approach* due to Kailath (1968, 1970).

in Section 14.1, except that the procedure is now formulated in terms of vectors and matrices (see Problem 1).

Correlation Matrix of the Innovations Process

To determine the correlation matrix of the innovations process $\boldsymbol{\alpha}(n)$, we first solve the state equation (14.17) recursively to obtain

$$\mathbf{x}(k) = \mathbf{F}(k, 0)\mathbf{x}(0) + \sum_{i=1}^{k-1} \mathbf{F}(k, i+1)\boldsymbol{\nu}_1(i), \tag{14.26}$$

where we have used the product rule governing the transition matrix and made two assumptions:

1. The initial value of the state is $\mathbf{x}(0)$.
2. The measured data, and therefore the noise vector $\boldsymbol{\nu}_1(n)$, are zero for $n \leq 0$.

Equation (14.26) indicates that $\mathbf{x}(k)$ is a linear combination of $\mathbf{x}(0)$ and $\boldsymbol{\nu}_1(1), \boldsymbol{\nu}_1(2), \ldots, \boldsymbol{\nu}_1(k-1)$.

By hypothesis, the measurement noise vector $\boldsymbol{\nu}_2(n)$ is uncorrelated with both the initial state vector $\mathbf{x}(0)$ and the system noise vector $\boldsymbol{\nu}_1(n)$. Accordingly, premultiplying both sides of Eq. (14.26) by $\boldsymbol{\nu}_2^{\mathrm{H}}(n)$ and taking expectations, we have

$$\mathbb{E}[\mathbf{x}(k)\boldsymbol{\nu}_2^{\mathrm{H}}(n)] = \mathbf{O}, \qquad k, n \leq 0. \tag{14.27}$$

Similarly, from the measurement equation (14.19), we have

$$\mathbb{E}[\mathbf{y}(k)\boldsymbol{\nu}_2^{\mathrm{H}}(n)] = \mathbf{O}, \qquad 0 \leq k \leq n-1. \tag{14.28}$$

Moreover, we may write

$$\mathbb{E}[\mathbf{y}(k)\boldsymbol{\nu}_1^{\mathrm{H}}(n)] = \mathbf{O}, \qquad 0 \leq k \leq n. \tag{14.29}$$

Given the past measurements $\mathbf{y}(1), \ldots, \mathbf{y}(n-1)$ that span the space $\mathcal{Y}_{n-1}$, we also find from the measurement equation (14.19) that the minimum mean-square estimate of the present measurement vector $\mathbf{y}(n)$:

$$\hat{\mathbf{y}}(n|\mathcal{Y}_{n-1}) = \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \hat{\boldsymbol{\nu}}_2(n|\mathcal{Y}_{n-1}).$$

However, the estimate $\hat{\boldsymbol{\nu}}_2(n|\mathcal{Y}_{n-1})$ of the measurement noise vector is zero, because $\boldsymbol{\nu}_2(n)$ is orthogonal to the past measurements $\mathbf{y}(1), \ldots, \mathbf{y}(n-1)$. [See Eq. (14.28).] Hence, we may simply write

$$\hat{\mathbf{y}}(n|\mathcal{Y}_{n-1}) = \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}). \tag{14.30}$$

Therefore, using Eqs. (14.22) and (14.30), we may express the innovations process in the form

$$\boldsymbol{\alpha}(n) = \mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}). \tag{14.31}$$

Substituting the measurement equation (14.19) into Eq. (14.31), we get

$$\boldsymbol{\alpha}(n) = \mathbf{C}(n)\boldsymbol{\varepsilon}(n, n-1) + \boldsymbol{\nu}_2(n), \tag{14.32}$$

where $\boldsymbol{\varepsilon}(n, n-1)$ is the *predicted state-error vector* at adaptation cycle n, using data up to adaptation cycle $n-1$. That is, $\boldsymbol{\varepsilon}(n, n-1)$ is the difference between the state $\mathbf{x}(n)$ and its one-step prediction $\hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1})$, or

$$\boldsymbol{\varepsilon}(n, n-1) = \mathbf{x}(n) - \hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1}). \tag{14.33}$$

Note that $\boldsymbol{\varepsilon}(n, n-1)$ is orthogonal to both the system noise vector $\boldsymbol{\nu}_1(n)$ and the measurement noise vector $\boldsymbol{\nu}_2(n)$. (See Problem 2.)

The correlation matrix of the innovations process $\boldsymbol{\alpha}(n)$ is defined by

$$\mathbf{R}(n) = \mathbb{E}[\boldsymbol{\alpha}(n)\boldsymbol{\alpha}^{\mathrm{H}}(n)]. \tag{14.34}$$

Therefore, substituting Eq. (14.32) into Eq. (14.34), expanding the pertinent terms, and then using the fact that the vectors $\boldsymbol{\varepsilon}(n, n-1)$ and $\boldsymbol{\nu}_2(n)$ are orthogonal, we obtain

$$\mathbf{R}(n) = \mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^{\mathrm{H}}(n) + \mathbf{Q}_2(n), \tag{14.35}$$

where $\mathbf{Q}_2(n)$ is the correlation matrix of the measurement noise vector $\boldsymbol{\nu}_2(n)$. The M-by-M matrix $\mathbf{K}(n, n-1)$ is called the *predicted state-error correlation matrix*, defined by

$$\mathbf{K}(n, n-1) = \mathbb{E}[\boldsymbol{\varepsilon}(n, n-1)\boldsymbol{\varepsilon}^{\mathrm{H}}(n, n-1)], \tag{14.36}$$

where $\boldsymbol{\varepsilon}(n, n-1)$ is the predicted state-error vector. The matrix $\mathbf{K}(n, n-1)$ may be viewed as the statistical description of the error incurred in computing the predicted estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$, as in Eq. (14.33).

14.4 ESTIMATION OF THE STATE USING THE INNOVATIONS PROCESS

Consider next the problem of deriving the minimum mean-square estimate of the state $\mathbf{x}(i)$ from the innovations process. From the discussion presented in Section 14.1, we find that this estimate may be expressed as a linear combination of the sequence of innovations processes $\boldsymbol{\alpha}(1), \boldsymbol{\alpha}(2), \ldots, \boldsymbol{\alpha}(n)$ [see Eq. (14.12) for comparison], or

$$\hat{\mathbf{x}}(i \mid \mathcal{Y}_n) = \sum_{k=1}^{n} \mathbf{B}_i(k)\boldsymbol{\alpha}(k), \tag{14.37}$$

where $\{\mathbf{B}_i(k)\}_{k=1}^{n}$ is a set of M-by-N matrices to be determined. According to the principle of orthogonality, the predicted state-error vector is orthogonal to the innovations process, as shown by

$$\begin{aligned} \mathbb{E}[\boldsymbol{\varepsilon}(i, n)\boldsymbol{\alpha}^{\mathrm{H}}(m)] &= \mathbb{E}\{[\mathbf{x}(i) - \hat{\mathbf{x}}(i \mid \mathcal{Y}_n)]\boldsymbol{\alpha}^{\mathrm{H}}(m)\} \\ &= \mathbf{O}, \qquad m = 1, 2, \ldots, n. \end{aligned} \tag{14.38}$$

Substituting Eq. (14.37) into Eq. (14.38) and using the orthogonality property of the innovations process, namely, Eq. (14.24), we get

$$\begin{aligned} \mathbb{E}[\mathbf{x}(i)\boldsymbol{\alpha}^{\mathrm{H}}(m)] &= \mathbf{B}_i(m)\mathbb{E}[\boldsymbol{\alpha}(m)\boldsymbol{\alpha}^{\mathrm{H}}(m)] \\ &= \mathbf{B}_i(m)\mathbf{R}(m). \end{aligned} \tag{14.39}$$

Hence, postmultiplying both sides of Eq. (14.39) by the inverse matrix $\mathbf{R}^{-1}(m)$, we find that $\mathbf{B}_i(m)$ is given by

$$\mathbf{B}_i(m) = \mathbb{E}[\mathbf{x}(i)\boldsymbol{\alpha}^{\mathrm{H}}(m)]\mathbf{R}^{-1}(m). \tag{14.40}$$

Finally, substituting Eq. (14.40) into Eq. (14.37), we get the minimum mean-square-error estimate

$$\begin{aligned}\hat{\mathbf{x}}(i \mid \mathcal{Y}_n) &= \sum_{k=1}^{n} \mathbb{E}[\mathbf{x}(i)\boldsymbol{\alpha}^{\mathrm{H}}(k)]\mathbf{R}^{-1}(k)\boldsymbol{\alpha}(k) \\ &= \sum_{k=1}^{n-1} \mathbb{E}[\mathbf{x}(i)\boldsymbol{\alpha}^{\mathrm{H}}(k)]\mathbf{R}^{-1}(k)\boldsymbol{\alpha}(k) \\ &\quad + \mathbb{E}[\mathbf{x}(i)\boldsymbol{\alpha}^{\mathrm{H}}(n)]\mathbf{R}^{-1}(n)\boldsymbol{\alpha}(n).\end{aligned}$$

For $i = n + 1$, we may therefore write

$$\begin{aligned}\hat{\mathbf{x}}(n+1 \mid \mathcal{Y}_n) &= \sum_{k=1}^{n-1} \mathbb{E}[\mathbf{x}(n+1)\boldsymbol{\alpha}^{\mathrm{H}}(k)]\mathbf{R}^{-1}(k)\boldsymbol{\alpha}(k) \\ &\quad + \mathbb{E}[\mathbf{x}(n+1)\boldsymbol{\alpha}^{\mathrm{H}}(n)]\mathbf{R}^{-1}(n)\boldsymbol{\alpha}(n).\end{aligned} \tag{14.41}$$

However, the state $\mathbf{x}(n+1)$ at adaptation cycle $n+1$ is related to the state $\mathbf{x}(n)$ at adaptation cycle n by Eq. (14.17). Therefore, using this relation, we may write, for $0 \le k \le n$,

$$\begin{aligned}\mathbb{E}[\mathbf{x}(n+1)\boldsymbol{\alpha}^{\mathrm{H}}(k)] &= \mathbb{E}\{\mathbf{F}[(n+1, n)\mathbf{x}(n) + \boldsymbol{\nu}_1(n)]\boldsymbol{\alpha}^{\mathrm{H}}(k)\} \\ &= \mathbf{F}(n+1, n)\mathbb{E}[\mathbf{x}(n)\boldsymbol{\alpha}^{\mathrm{H}}(k)],\end{aligned} \tag{14.42}$$

where we have made use of the fact that $\boldsymbol{\alpha}(k)$ depends only on the measurements $\mathbf{y}(1), \ldots, \mathbf{y}(k)$ and, therefore, from Eq. (14.29), we see that $\boldsymbol{\nu}_1(n)$ and $\boldsymbol{\alpha}(k)$ are orthogonal for $0 \le k \le n$. Thus, using Eq. (14.42) and the formula for $\hat{\mathbf{x}}(i \mid \mathcal{Y}_n)$ with $i = n$, we may rewrite the summation term on the right-hand side of Eq. (14.41) as

$$\begin{aligned}\sum_{k=1}^{n-1} \mathbb{E}[\mathbf{x}(n+1)\boldsymbol{\alpha}^{\mathrm{H}}(k)]\mathbf{R}^{-1}(k)\boldsymbol{\alpha}(k) &= \mathbf{F}(n+1, n)\sum_{k=1}^{n-1} \mathbb{E}[\mathbf{x}(n)\boldsymbol{\alpha}^{\mathrm{H}}(k)]\mathbf{R}^{-1}(k)\boldsymbol{\alpha}(k) \\ &= \mathbf{F}(n+1, n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})\end{aligned} \tag{14.43}$$

where in the last line of Eq. (14.43) we made use of the first line of the equation at the top of the page.

Kalman Gain

To proceed further, define the M-by-N matrix

$$\mathbf{G}(n) = \mathbb{E}[\mathbf{x}(n+1)\boldsymbol{\alpha}^{\mathrm{H}}(n)]\mathbf{R}^{-1}(n), \tag{14.44}$$

where $\mathbb{E}[\mathbf{x}(n+1)\boldsymbol{\alpha}^{\mathrm{H}}(n)]$ is the cross-correlation matrix between the state vector $\mathbf{x}(n+1)$ and the innovations process $\boldsymbol{\alpha}(n)$. Then, using this definition and the result of Eq. (14.43), we may rewrite Eq. (14.41) simply as

$$\hat{\mathbf{x}}(n+1 \mid \mathcal{Y}_n) = \mathbf{F}(n+1, n)\hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1}) + \mathbf{G}(n)\boldsymbol{\alpha}(n). \tag{14.45}$$

Equation (10.45) is of fundamental significance. It shows that we may compute the minimum mean-square estimate $\hat{\mathbf{x}}(n+1 \mid \mathcal{Y}_n)$ of the state of a linear dynamic model by adding to the previous estimate $\hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1})$, which is premultiplied by the transition matrix $\mathbf{F}(n+1, n)$, a correction term equal to $\mathbf{G}(n)\boldsymbol{\alpha}(n)$. The correction term equals the innovations process $\boldsymbol{\alpha}(n)$ premultiplied by the matrix $\mathbf{G}(n)$. Accordingly, and in recognition of the pioneering work by Kalman, the matrix $\mathbf{G}(n)$ is called the *Kalman gain*.

There now remains only the problem of expressing the Kalman gain $\mathbf{G}(n)$ in a form convenient for computation. To do this, we first use Eqs. (14.32) and (14.42) to express the expectation of the product of $\mathbf{x}(n+1)$ and $\boldsymbol{\alpha}^{\mathrm{H}}(n)$ as

$$\begin{aligned}\mathbb{E}[\mathbf{x}(n+1)\boldsymbol{\alpha}^{\mathrm{H}}(n)] &= \mathbf{F}(n+1,n)\mathbb{E}[\mathbf{x}(n)\boldsymbol{\alpha}^{\mathrm{H}}(n)] \\ &= \mathbf{F}(n+1,n)\mathbb{E}[\mathbf{x}(n)(\mathbf{C}(n)\boldsymbol{\varepsilon}(n,n-1)+\boldsymbol{\nu}_2(n))^{\mathrm{H}}] \\ &= \mathbf{F}(n+1,n)\mathbb{E}[\mathbf{x}(n)\boldsymbol{\varepsilon}^{\mathrm{H}}(n,n-1)]\mathbf{C}^{\mathrm{H}}(n), \end{aligned} \tag{14.46}$$

where we have used the fact that the state $\mathbf{x}(n)$ and noise vector $\boldsymbol{\nu}_2(n)$ are uncorrelated. [See Eq. (14.27).] Next we note that the predicted state-error vector $\boldsymbol{\varepsilon}(n,n-1)$ is orthogonal to the estimate $\hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1})$. Therefore, the expectation of the product of $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$ and $\boldsymbol{\varepsilon}^{\mathrm{H}}(n,n-1)$ is zero, so Eq. (14.46) is unchanged by replacing the multiplying factor $\mathbf{x}(n)$ by the predicted state-error vector $\boldsymbol{\varepsilon}(n,n-1)$ as follows:

$$\mathbb{E}[\mathbf{x}(n+1)\boldsymbol{\alpha}^{\mathrm{H}}(n)] = \mathbf{F}(n+1,n)\mathbb{E}[\boldsymbol{\varepsilon}(n,n-1)\boldsymbol{\varepsilon}^{\mathrm{H}}(n,n-1)]\mathbf{C}^{\mathrm{H}}(n). \tag{14.47}$$

From Eq. (14.36), we see that the expectation on the right-hand side of Eq. (14.47) equals the predicted state-error correlation matrix. Hence, we may rewrite Eq. (14.47) as

$$\mathbb{E}[\mathbf{x}(n+1)\boldsymbol{\alpha}^{\mathrm{H}}(n)] = \mathbf{F}(n+1,n)\mathbf{K}(n,n-1)\mathbf{C}^{\mathrm{H}}(n). \tag{14.48}$$

We may now redefine the Kalman gain. Substituting Eq. (14.48) into Eq. (14.44), we get

$$\mathbf{G}(n) = \mathbf{F}(n+1,n)\mathbf{K}(n,n-1)\mathbf{C}^{\mathrm{H}}(n)\mathbf{R}^{-1}(n), \tag{14.49}$$

where the correlation matrix $\mathbf{R}(n)$ is itself defined in Eq. (14.35).

The block diagram of Fig. 14.3 shows the signal-flow graph representation of Eq. (14.49) for computing the Kalman gain $\mathbf{G}(n)$. Having computed $\mathbf{G}(n)$, we may then use Eq. (14.45) to update the one-step prediction—that is, to compute $\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$, given its old value $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$, as illustrated in Fig. 14.4, in which we have also used Eq. (14.31) for the innovations process $\boldsymbol{\alpha}(n)$.

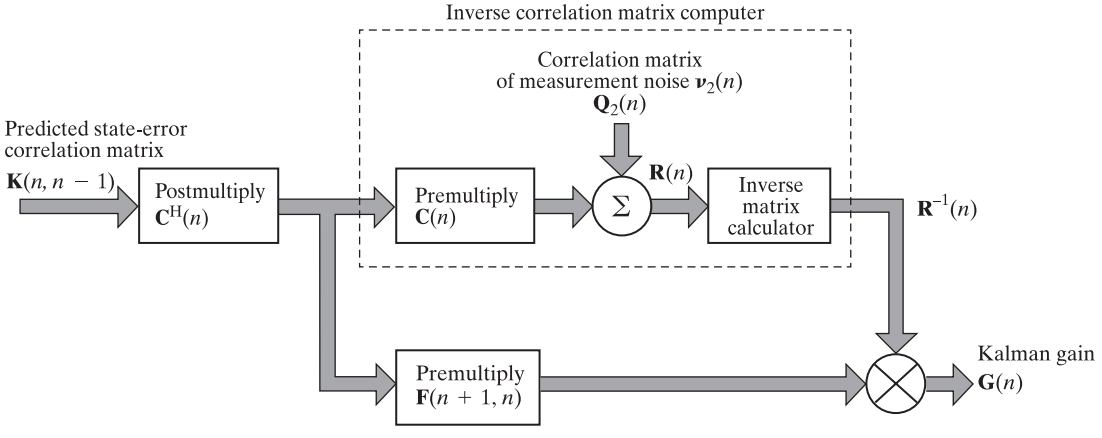


FIGURE 14.3 Kalman gain computer.

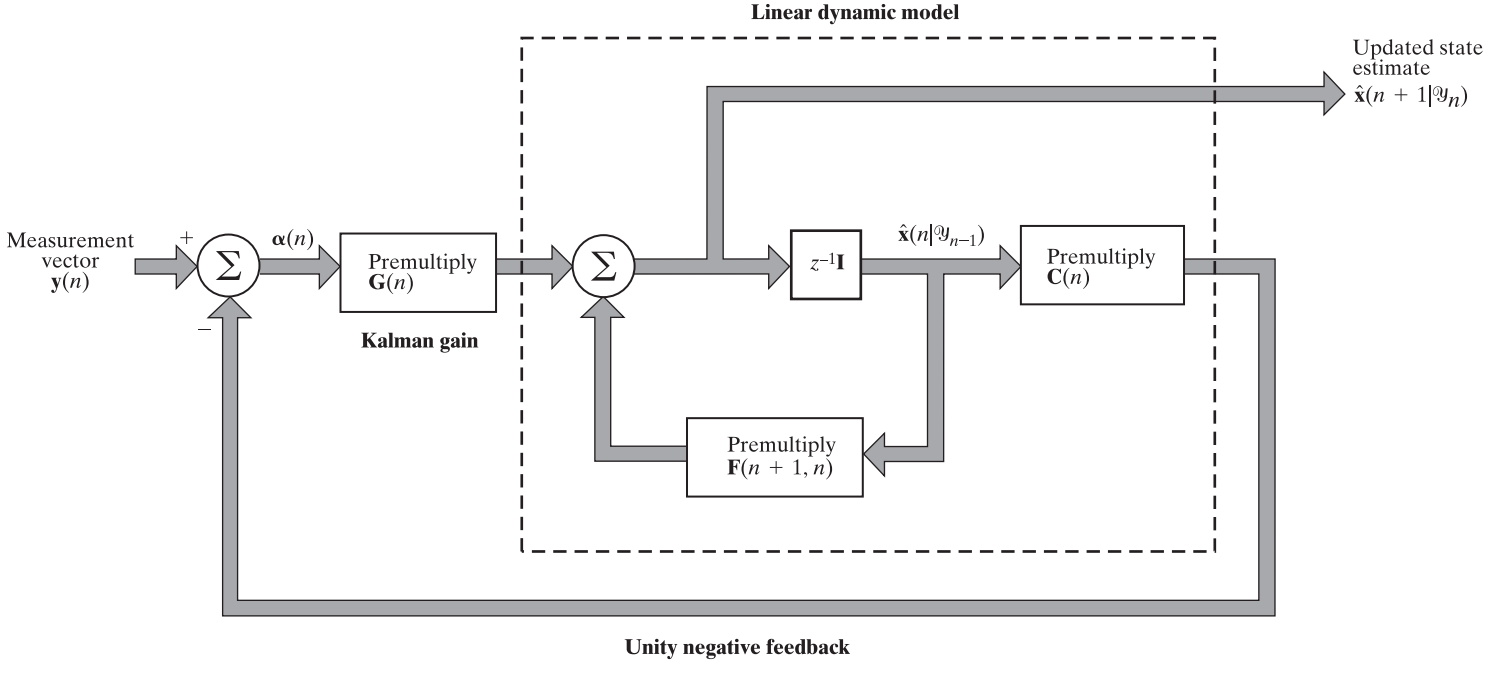


FIGURE 14.4 One-step state predictor; given the old estimate $\hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1})$ and the measurement $\mathbf{y}(n)$, the predictor computes the new state $\hat{\mathbf{x}}(n+1 \mid \mathcal{Y}_n)$.

Riccati Equation

As it stands, Eq. (14.49) is not particularly useful for computing the Kalman gain $\mathbf{G}(n)$, since it requires that the predicted state-error correlation matrix $\mathbf{K}(n, n-1)$ be known. To overcome this difficulty, we derive a formula for the recursive computation of $\mathbf{K}(n, n-1)$.

The predicted state-error vector $\boldsymbol{\varepsilon}(n+1, n)$ equals the difference between the state $\mathbf{x}(n+1)$ and the one-step prediction $\hat{\mathbf{x}}(n+1 \,|\, \mathcal{Y}_n)$ [see Eq. (14.33)]:

$$\boldsymbol{\varepsilon}(n+1, n) = \mathbf{x}(n+1) - \hat{\mathbf{x}}(n+1 \,|\, \mathcal{Y}_n). \tag{14.50}$$

Substituting Eqs. (14.17) and (14.45) into Eq. (14.50), and using Eq. (14.31) for the innovations process $\boldsymbol{\alpha}(n)$, we get

$$\begin{aligned}\boldsymbol{\varepsilon}(n+1, n) = \mathbf{F}(n+1, n)[\mathbf{x}(n) - \hat{\mathbf{x}}(n \,|\, \mathcal{Y}_{n-1})] \\ - \mathbf{G}(n)[\mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n \,|\, \mathcal{Y}_{n-1})] + \boldsymbol{\nu}_1(n).\end{aligned} \tag{14.51}$$

Next, using the measurement equation (14.19) to eliminate $\mathbf{y}(n)$ in Eq. (14.51), we get the following difference equation for the recursive computation of the predicted state-error vector:

$$\begin{aligned}\boldsymbol{\varepsilon}(n+1, n) = [\mathbf{F}(n+1, n) - \mathbf{G}(n)\mathbf{C}(n)]\boldsymbol{\varepsilon}(n, n-1) \\ + \boldsymbol{\nu}_1(n) - \mathbf{G}(n)\boldsymbol{\nu}_2(n).\end{aligned} \tag{14.52}$$

The correlation matrix of the predicted state-error vector $\boldsymbol{\varepsilon}(n+1, n)$ equals [see Eq. (14.36)]

$$\mathbf{K}(n+1, n) = \mathbb{E}[\boldsymbol{\varepsilon}(n+1, n)\boldsymbol{\varepsilon}^{\mathrm{H}}(n+1, n)]. \tag{14.53}$$

Substituting Eq. (14.52) into Eq. (14.53) and recognizing that the error vector $\boldsymbol{\varepsilon}(n, n-1)$ and the noise vectors $\boldsymbol{\nu}_1(n)$ and $\boldsymbol{\nu}_2(n)$ are mutually uncorrelated, we may express the predicted state-error correlation matrix as

$$\begin{aligned}\mathbf{K}(n+1, n) = [\mathbf{F}(n+1, n) - \mathbf{G}(n)\mathbf{C}(n)]\mathbf{K}(n, n-1)[\mathbf{F}(n+1, n) - \mathbf{G}(n)\mathbf{C}(n)]^{\mathrm{H}} \\ + \mathbf{Q}_1(n) + \mathbf{G}(n)\mathbf{Q}_2(n)\mathbf{G}^{\mathrm{H}}(n),\end{aligned} \tag{14.54}$$

where $\mathbf{Q}_1(n)$ and $\mathbf{Q}_2(n)$ are the correlation matrices of $\boldsymbol{\nu}_1(n)$ and $\boldsymbol{\nu}_2(n)$, respectively. By expanding the right-hand side of Eq. (14.54) and then using Eqs. (14.49) and (14.35) for the Kalman gain, we get the *Riccati difference equation*[2] for the recursive computation of the predicted state-error correlation matrix:

$$\mathbf{K}(n+1, n) = \mathbf{F}(n+1, n)\mathbf{K}(n)\mathbf{F}^{\mathrm{H}}(n+1, n) + \mathbf{Q}_1(n). \tag{14.55}$$

The new M-by-M matrix $\mathbf{K}(n)$ introduced in Eq. (14.55) is described by the recursion

$$\mathbf{K}(n) = \mathbf{K}(n, n-1) - \mathbf{F}(n, n+1)\mathbf{G}(n)\mathbf{C}(n)\mathbf{K}(n, n-1), \tag{14.56}$$

where, in accordance with the inverse rule governing $\mathbf{F}(n+1, n)$, we have

$$\mathbf{F}(n+1, n)\mathbf{F}(n, n+1) = \mathbf{I},$$

[2]The Riccati difference equation is named in honor of Count Jacopo Francisco Riccati. The equation is of particular importance in control theory.

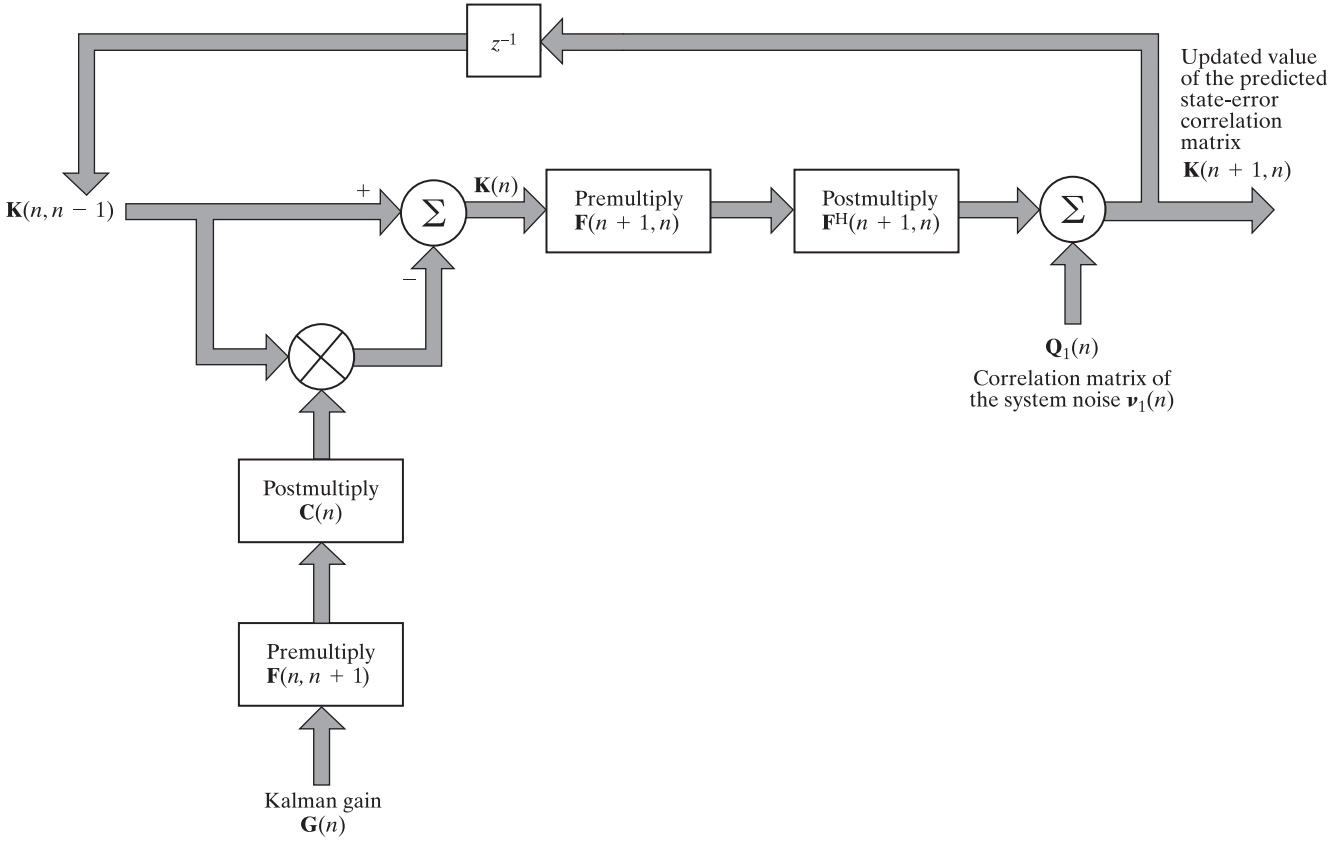


FIGURE 14.5 Riccati equation solver for propagating the predicted state-error correlation matrix.

which follows from the product and inverse rules governing the transition matrix. [The mathematical significance of the matrix $\mathbf{K}(n)$ in Eq. (14.56) will be explained later in Section 14.5.]

Figure 14.5 is a signal-flow graph representation of Eqs. (14.56) and (14.55), in that order. This diagram may be viewed as a representation of the *Riccati equation solver* in that, given $\mathbf{K}(n, n-1)$, it computes the updated value $\mathbf{K}(n+1, n)$.

Summarizing: the combined use of Eqs. (14.49), (14.35), (14.31), (14.45), (14.56), and (14.55), in that order, defines Kalman's one-step prediction algorithm.

14.5 FILTERING

The next signal-processing operation we wish to consider is that of filtering. In particular, we wish to compute the *filtered estimate* $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ by using the one-step prediction algorithm described previously.

We first note that the state $\mathbf{x}(n)$ and the system noise $\boldsymbol{\nu}_1(n)$ are independent of each other. Hence, from the state equation (14.17), we find that the minimum mean-square estimate of the state $\mathbf{x}(n+1)$ at adaptation cycle $n+1$, given the measured data up to and including adaptation cycle n [i.e., given $\mathbf{y}(1), \ldots, \mathbf{y}(n)$], equals

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \mathbf{F}(n+1, n)\hat{\mathbf{x}}(n|\mathcal{Y}_n) + \hat{\boldsymbol{\nu}}_1(n|\mathcal{Y}_n). \tag{14.57}$$

Since the noise $\boldsymbol{\nu}_1(n)$ is independent of the measurements $\mathbf{y}(1), \ldots, \mathbf{y}(n)$, it follows that the corresponding minimum mean-square estimate $\hat{\boldsymbol{\nu}}_1(n|\mathcal{Y}_n)$ is zero. Accordingly, Eq. (14.57) simplifies to

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \mathbf{F}(n+1, n)\hat{\mathbf{x}}(n|\mathcal{Y}_n). \tag{14.58}$$

To find the filtered estimate $\mathbf{x}(n|\mathcal{Y}_n)$, we premultiply both sides of Eq. (14.58) by the transition matrix $\mathbf{F}(n, n+1)$. By so doing, we obtain

$$\hat{\mathbf{x}}(n|\mathcal{Y}_n) = \mathbf{F}(n, n+1)\hat{\mathbf{x}}(n+1|\mathcal{Y}_n). \tag{14.59}$$

This equation shows that knowing the solution to the one-step prediction problem [i.e., the minimum mean-square estimate $\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$], we may determine the corresponding filtered estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ simply by multiplying $\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$ by the transition matrix $\mathbf{F}(n, n+1)$. Note that in arriving at Eq. (14.59), we used the inverse rule that governs the transition matrix.

Filtered Estimation Error and Conversion Factor

In the filtering framework, it is natural that we define a *filtered estimation error vector* in terms of the filtered estimate of the state as follows:

$$\mathbf{e}(n) = \mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_n). \tag{14.60}$$

This definition is similar to that of Eq. (14.31) for the innovations vector $\boldsymbol{\alpha}(n)$, except that we have substituted the filtered estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ for the predicted estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$. Using Eqs. (14.45) and (14.59) in (14.60), we get

$$\begin{aligned}\mathbf{e}(n) &= \mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) - \mathbf{C}(n)\mathbf{F}(n, n+1)\mathbf{G}(n)\boldsymbol{\alpha}(n)\\ &= \boldsymbol{\alpha}(n) - \mathbf{C}(n)\mathbf{F}(n, n+1)\mathbf{G}(n)\boldsymbol{\alpha}(n)\\ &= [\mathbf{I} - \mathbf{C}(n)\mathbf{F}(n, n+1)\mathbf{G}(n)]\boldsymbol{\alpha}(n).\end{aligned} \tag{14.61}$$

The matrix-valued quantity inside the square brackets in Eq. (14.61) is called the *conversion factor*: it provides a formula for converting the innovations vector $\boldsymbol{\alpha}(n)$ into the filtered estimation error vector $\mathbf{e}(n)$. Using Eq. (14.49) to eliminate the Kalman gain $\mathbf{G}(n)$ from this definition and canceling common terms, we may rewrite Eq. (14.61) in the equivalent form

$$\mathbf{e}(n) = \mathbf{Q}_2(n)\mathbf{R}^{-1}(n)\boldsymbol{\alpha}(n), \tag{14.62}$$

where $\mathbf{Q}_2(n)$ is the correlation matrix of the measurement noise process $\mathbf{v}_2(n)$ and the matrix $\mathbf{R}(n)$ is itself defined in Eq. (14.35) as the correlation matrix of the innovations process $\boldsymbol{\alpha}(n)$. Thus, except for a premultiplication by $\mathbf{Q}_2(n)$, Eq. (14.62) shows that the inverse matrix $\mathbf{R}^{-1}(n)$ plays the role of a conversion factor in the Kalman filter theory. Indeed, for the special case of $\mathbf{Q}_2(n)$ equal to the identity matrix, the inverse matrix $\mathbf{R}^{-1}$ is exactly the conversion factor defined herein.

Filtered State-Error Correlation Matrix

Earlier, we introduced the M-by-M matrix $\mathbf{K}(n)$ in the formulation of the Riccati difference equation (14.55). We conclude our present discussion of Kalman filter theory by showing that this matrix equals the correlation matrix of the error inherent in the filtered estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$.

Define the *filtered state-error vector* $\boldsymbol{\varepsilon}(n)$ as the difference between the state $\mathbf{x}(n)$ and the filtered estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$; that is,

$$\boldsymbol{\varepsilon}(n) = \mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathcal{Y}_n). \tag{14.63}$$

Substituting Eqs. (14.45) and (14.59) into Eq. (14.63) and recognizing that the product $\mathbf{F}(n, n+1)\,\mathbf{F}(n+1, n)$ equals the identity matrix, we get

$$\begin{aligned}\boldsymbol{\varepsilon}(n) &= \mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) - \mathbf{F}(n, n+1)\mathbf{G}(n)\boldsymbol{\alpha}(n)\\ &= \boldsymbol{\varepsilon}(n, n-1) - \mathbf{F}(n, n+1)\mathbf{G}(n)\boldsymbol{\alpha}(n),\end{aligned} \tag{14.64}$$

where $\boldsymbol{\varepsilon}(n, n-1)$ is the predicted state-error vector at adaptation cycle n, using data up to adaptation cycle $n-1$, and $\boldsymbol{\alpha}(n)$ is the innovations process.

By definition, the correlation matrix of $\boldsymbol{\varepsilon}(n)$ equals the expectation $\mathbb{E}[\boldsymbol{\varepsilon}(n)\boldsymbol{\varepsilon}^{\mathrm{H}}(n)]$. Hence, using Eq. (14.64) in this definition, we may express this expectation as follows:

$$\begin{aligned}\mathbb{E}[\boldsymbol{\varepsilon}(n)\boldsymbol{\varepsilon}^{\mathrm{H}}(n)] &= \mathbb{E}[\boldsymbol{\varepsilon}(n, n-1)\boldsymbol{\varepsilon}^{\mathrm{H}}(n, n-1)]\\ &\quad + \mathbf{F}(n, n+1)\mathbf{G}(n)\mathbb{E}[\boldsymbol{\alpha}(n)\boldsymbol{\alpha}^{\mathrm{H}}(n)]\mathbf{G}^{\mathrm{H}}(n)\mathbf{F}^{\mathrm{H}}(n, n+1)\\ &\quad - \mathbb{E}[\boldsymbol{\varepsilon}(n, n-1)\boldsymbol{\alpha}^{\mathrm{H}}(n)]\mathbf{G}^{\mathrm{H}}(n)\mathbf{F}^{\mathrm{H}}(n, n+1)\\ &\quad - \mathbf{F}(n+1, n)\mathbf{G}(n)\mathbb{E}[\boldsymbol{\alpha}(n)\boldsymbol{\varepsilon}^{\mathrm{H}}(n, n-1)].\end{aligned} \tag{14.65}$$

Examining the right-hand side of Eq. (14.65), we find that the four expectations contained in it may be interpreted individually as follows:

1. The first expectation equals the predicted state-error correlation matrix

$$\mathbf{K}(n, n-1) = \mathbb{E}[\boldsymbol{\varepsilon}(n, n-1)\boldsymbol{\varepsilon}^{\mathrm{H}}(n, n-1)].$$

2. The expectation in the second term equals the correlation matrix of the innovations process $\boldsymbol{\alpha}(n)$

$$\mathbf{R}(n) = \mathbb{E}[\boldsymbol{\alpha}(n)\boldsymbol{\alpha}^{\mathrm{H}}(n)].$$

3. The expectation in the third term may be expressed as

$$\begin{aligned}\mathbb{E}[\boldsymbol{\varepsilon}(n, n-1)\boldsymbol{\alpha}^{\mathrm{H}}(n)] &= \mathbb{E}[(\mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}))\boldsymbol{\alpha}^{\mathrm{H}}(n)] \\ &= \mathbb{E}[\mathbf{x}(n)\boldsymbol{\alpha}^{\mathrm{H}}(n)],\end{aligned}$$

where we have used the fact that the estimate $\hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1})$ is orthogonal to the innovations process $\boldsymbol{\alpha}(n)$. Next, from Eq. (14.42), we see, by putting $k = n$ and then premultiplying both sides by $\mathbf{F}^{-1}(n+1, n) = \mathbf{F}(n, n+1)$, that

$$\begin{aligned}\mathbb{E}[\mathbf{x}(n)\boldsymbol{\alpha}^{\mathrm{H}}(n)] &= \mathbf{F}(n, n+1)\mathbb{E}[\mathbf{x}(n+1)\boldsymbol{\alpha}^{\mathrm{H}}(n)] \\ &= \mathbf{F}(n, n+1)\mathbf{G}(n)\mathbf{R}(n),\end{aligned}$$

where, in the last line, we have made use of Eq. (14.44). Hence,

$$\mathbb{E}[\boldsymbol{\varepsilon}(n, n-1)\boldsymbol{\alpha}^{\mathrm{H}}(n)] = \mathbf{F}(n, n+1)\mathbf{G}(n)\mathbf{R}(n).$$

Similarly, we may express the expectation in the fourth term as

$$\mathbb{E}[\boldsymbol{\alpha}(n)\boldsymbol{\varepsilon}^{\mathrm{H}}(n, n-1)] = \mathbf{R}(n)\mathbf{G}^{\mathrm{H}}(n)\mathbf{F}^{\mathrm{H}}(n, n-1).$$

We may now combine these results in Eq. (14.65) and so obtain

$$\mathbb{E}[\boldsymbol{\varepsilon}(n)\boldsymbol{\varepsilon}^{\mathrm{H}}(n)] = \mathbf{K}(n, n-1) - \mathbf{F}(n, n+1)\mathbf{G}(n)\mathbf{R}(n)\mathbf{G}^{\mathrm{H}}(n)\mathbf{F}^{\mathrm{H}}(n, n+1). \quad (14.66)$$

We may further simplify this result by noting that [see Eq. (14.49)]

$$\mathbf{G}(n)\mathbf{R}(n) = \mathbf{F}(n+1, n)\mathbf{K}(n, n-1)\mathbf{C}^{\mathrm{H}}(n). \quad (14.67)$$

Accordingly, using Eqs. (14.66) and (14.67), and applying the inverse rule for the transition matrix, we get

$$\mathbb{E}[\boldsymbol{\varepsilon}(n)\boldsymbol{\varepsilon}^{\mathrm{H}}(n)] = \mathbf{K}(n, n-1) - \mathbf{K}(n, n-1)\mathbf{C}^{\mathrm{H}}(n)\mathbf{G}^{\mathrm{H}}(n)\mathbf{F}^{\mathrm{H}}(n, n+1). \quad (14.68)$$

Equivalently, using the Hermitian property of $\mathbb{E}[\boldsymbol{\varepsilon}(n)\boldsymbol{\varepsilon}^{\mathrm{H}}(n)]$ and that of $\mathbf{K}(n, n-1)$, we may write

$$\mathbb{E}[\boldsymbol{\varepsilon}(n)\boldsymbol{\varepsilon}^{\mathrm{H}}(n)] = \mathbf{K}(n, n-1) - \mathbf{F}(n, n+1)\mathbf{G}(n)\mathbf{C}(n)\mathbf{K}(n, n-1). \quad (14.69)$$

Comparing Eq. (14.69) with Eq. (14.56), we readily see that

$$\mathbb{E}[\boldsymbol{\varepsilon}(n)\boldsymbol{\varepsilon}^{\mathrm{H}}(n)] = \mathbf{K}(n). \quad (14.70)$$

This shows that the matrix $\mathbf{K}(n)$ used in the Riccati difference equation (14.55) is in fact the *filtered state-error correlation matrix*. The matrix $\mathbf{K}(n)$ is used as the statistical description of the error in the filtered estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$.

14.6 INITIAL CONDITIONS

To operate the one-step prediction and filtering algorithms described in Sections 14.4 and 14.5, we obviously need to specify the *initial conditions*. We now address this issue.

The initial state of the system equation (14.17) is not known precisely. Rather, it is usually described by its mean and correlation matrix. In the absence of any measured data at adaptation cycle $n = 0$, we may choose the *initial predicted estimate* as

$$\hat{\mathbf{x}}(1|\mathcal{Y}_0) = \mathbb{E}[\mathbf{x}(1)] \quad (14.71)$$

and its *correlation matrix*

$$\begin{aligned}\mathbf{K}(1,0) &= \mathbb{E}[(\mathbf{x}(1) - \mathbb{E}[\mathbf{x}(1)])(\mathbf{x}(1) - \mathbb{E}[\mathbf{x}(1)])^{\mathrm{H}}] \\ &= \mathbf{\Pi}_0. \end{aligned} \tag{14.72}$$

This choice for the initial conditions is not only intuitively satisfying, but also has the advantage of yielding a filtered estimate of the state $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ that is *unbiased*. (See Problem 7.) Assuming that the state vector $\mathbf{x}(n)$ has *zero mean*, we may simplify Eqs. (14.71) and (14.72) by setting

$$\hat{\mathbf{x}}(1|\mathcal{Y}_0) = \mathbf{0}$$

and

$$\mathbf{K}(1,0) = \mathbb{E}[\mathbf{x}(1)\mathbf{x}^{\mathrm{H}}(1)] = \mathbf{\Pi}_0.$$

14.7 SUMMARY OF THE KALMAN FILTER

Table 14.1 presents a summary of the variables and parameters used to formulate the solution to the Kalman filtering problem.[3] The input of the filter is the measurement $\mathbf{y}(n)$, picked from the vector space $\mathcal{Y}_n$, and the output is the filtered estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$

TABLE 14.1 Summary of the Kalman Variables and Parameters

Variable	Definition	Dimension
$\mathbf{x}(n)$	State at adaptation cycle n	M by 1
$\mathbf{y}(n)$	Measurement at adaptation cycle n	N by 1
$\mathbf{F}(n+1,n)$	Transition matrix from adaptation cycle n to adaptation cycle $n+1$	M by M
$\mathbf{C}(n)$	Measurement matrix at adaptation cycle n	N by M
$\mathbf{Q}_1(n)$	Correlation matrix of system noise $\boldsymbol{\nu}_1(n)$	M by M
$\mathbf{Q}_2(n)$	Correlation matrix of measurement noise $\boldsymbol{\nu}_2(n)$	N by N
$\hat{\mathbf{x}}(n\|\mathcal{Y}_{n-1})$	Predicted estimate of the state at adaptation cycle n given the measurements $\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n-1)$	M by 1
$\hat{\mathbf{x}}(n\|\mathcal{Y}_n)$	Filtered estimate of the state at adaptation cycle n, given the measurements $\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)$	M by 1
$\mathbf{G}(n)$	Kalman gain at adaptation cycle n	M by N
$\boldsymbol{\alpha}(n)$	Innovations vector at adaptation cycle n	N by 1
$\mathbf{R}(n)$	Correlation matrix of the innovations vector $\boldsymbol{\alpha}(n)$	N by N
$\mathbf{K}(n,n-1)$	Correlation matrix of the error in $\hat{\mathbf{x}}(n\|\mathcal{Y}_{n-1})$	M by M
$\mathbf{K}(n)$	Correlation matrix of the error in $\hat{\mathbf{x}}(n\|\mathcal{Y}_n)$	M by M

[3]In formulating the Kalman filter whose variables and parameters are summarized in Table 14.1, we have used $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$ and $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ to distinguish between the a priori [before including the measurement $\mathbf{y}(n)$] and a posteriori [after including the measurement $\mathbf{y}(n)$] estimates of the state $\mathbf{x}(n)$, respectively. In the literature on Kalman filters, the alternative notations $\mathbf{x}_n^-$ and $\hat{\mathbf{x}}_n$ are often used for the a priori and a posteriori estimates of the state, respectively. By the same token, $\mathbf{K}_n^-$ and $\mathbf{K}_n$ are often used in place of $\mathbf{K}(n, n-1)$ and $\mathbf{K}(n)$, respectively.

TABLE 14.2 Summary of the Kalman Filter Based on One-Step Prediction

Input vector process:

Measurements $= \{\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)\}$

Known parameters:

Transition matrix $= \mathbf{F}(n+1, n)$

Measurement matrix $= \mathbf{C}(n)$

Correlation matrix of system noise $= \mathbf{Q}_1(n)$

Correlation matrix of measurement noise $= \mathbf{Q}_2(n)$

Computation: $n = 1, 2, 3, \ldots$

$$\mathbf{G}(n) = \mathbf{F}(n+1, n)\mathbf{K}(n, n-1)\mathbf{C}^{\mathrm{H}}(n)[\mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^{\mathrm{H}}(n) + \mathbf{Q}_2(n)]^{-1}$$

$$\boldsymbol{\alpha}(n) = \mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$$

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \mathbf{F}(n+1, n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{G}(n)\boldsymbol{\alpha}(n)$$

$$\mathbf{K}(n) = \mathbf{K}(n, n-1) - \mathbf{F}(n, n+1)\mathbf{G}(n)\mathbf{C}(n)\mathbf{K}(n, n-1)$$

$$\mathbf{K}(n+1, n) = \mathbf{F}(n+1, n)\mathbf{K}(n)\mathbf{F}^{\mathrm{H}}(n+1, n) + \mathbf{Q}_1(n)$$

Initial conditions:

$$\hat{\mathbf{x}}(1 \mid \mathcal{Y}_0) = \mathbb{E}[\mathbf{x}(1)]$$

$$\mathbf{K}(1, 0) = \mathbb{E}[(\mathbf{x}(1) - \mathbb{E}[\mathbf{x}(1)])(\mathbf{x}(1) - \mathbb{E}[\mathbf{x}(1)])^{\mathrm{H}}] = \boldsymbol{\Pi}_0$$

of the state. In Table 14.2, we present a summary of the Kalman filter (including initial conditions) based on the one-step prediction algorithm.

A block diagram representation of the Kalman filter is given in Fig. 14.6, which is based on three functional blocks:

1. The one-step predictor, described in Fig. 14.4.
2. The Kalman gain computer, described in Fig. 14.3.
3. The Riccati equation solver, described in Fig. 14.5.

14.8 OPTIMALITY CRITERIA FOR KALMAN FILTERING

The core of Kalman filtering summarized in Table 14.1 may be traced to Section 14.4, where the recursive formula for estimating the predicted state in the state-space model was derived in Eq. (14.45). That derivation was accomplished by exploiting the innovations process, without having to invoke an optimality criterion. However, optimality is an issue of mathematical importance, hence the need to understand how it features in deriving the Kalman filter.

In this context, it is often said that Kalman filtering is a by-product of the celebrated *Bayesian estimation paradigm*. From such a viewpoint, there are two matters that need attention:

1. The manner in which the probability density function (pdf) of the state $\mathbf{x}(n)$ is *conditioned* on the entire past history of the measurements—namely, $\mathcal{Y}_n$.
2. The way in which the sequence of measurements in $\mathcal{Y}_n$ is propagated across time, from n to $n+1$.

Once these two matters are explicitly formulated, the stage is set for optimal estimation of the Kalman filter (Ho & Lee, 1964; Maybeck, 1979). In Problem 8, the

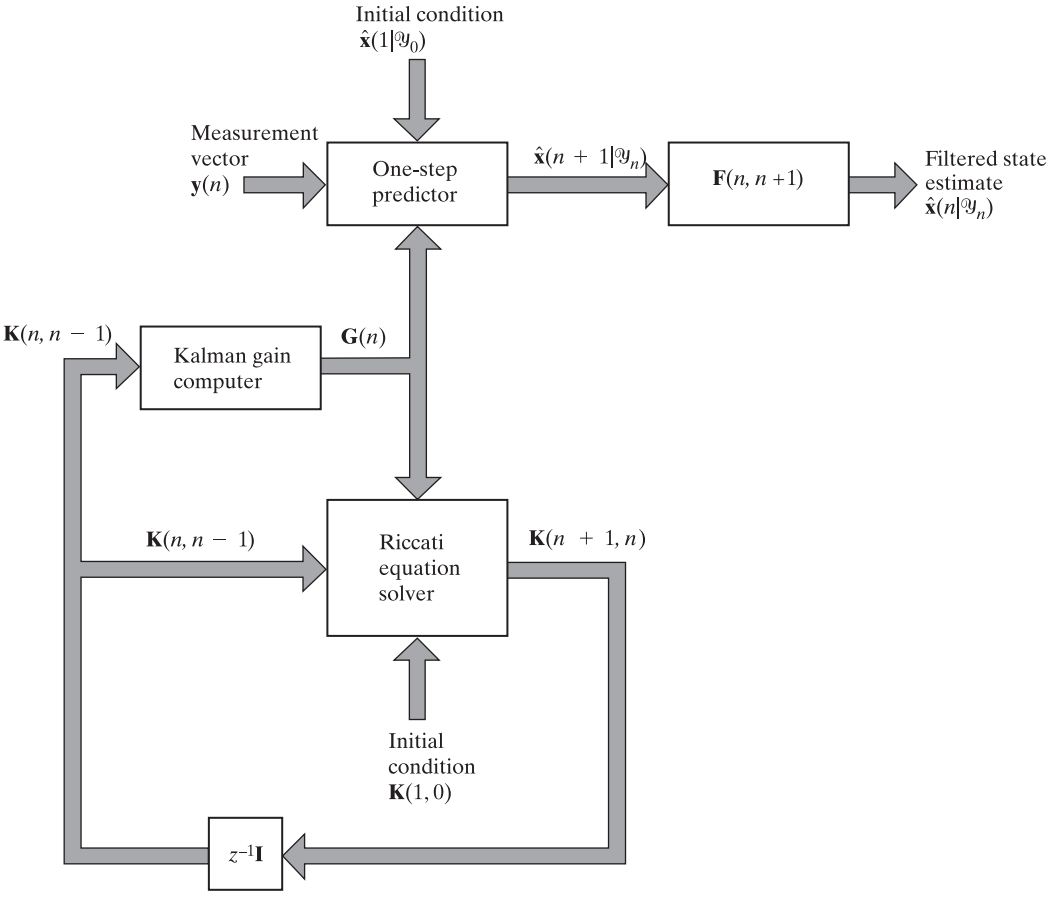


FIGURE 14.6 Block diagram of the Kalman filter based on one-step prediction.

Bayesian approach for deriving the Kalman filter is addressed under the assumption that the system noise $\boldsymbol{\nu}_1(n)$ and the measurement noise $\boldsymbol{\nu}_2(n)$ are both zero-mean Gaussian processes. To be more precise, solution of the *maximum a posteriori probability (MAP) criterion* yields the Kalman filter of Table 14.1.

Another optimality criterion suitable for solving the Kalman filtering problem is the *minimum-mean-square-error (MMSE) criterion*. Let $\hat{\mathbf{x}}_{\text{est}}(n)$ denote the estimate of the unknown state $\mathbf{x}(n)$ at time n given the sequence of measurements $\mathcal{Y}_n$. It turns out that the particular estimate that minimizes the squared Euclidean norm of the estimation error vector (i.e., the difference between the vectors $\mathbf{x}(n)$ and $\hat{\mathbf{x}}_{\text{est}}(n|\mathcal{Y}_n)$) is the *conditional mean estimator* $\hat{\mathbf{x}}_{\text{est}}(n|\mathcal{Y}_n)$. (The conditional mean estimator is discussed in Appendix D.) In this second approach, the Kalman filter is viewed as an elaboration of the conditional mean estimator (Van Trees, 1968). The MMSE criterion plays a role similar to the *least-squares-error (LSE) criterion*, which is the very criterion used in Chapter 10 to derive the recursive least-squares (RLS) algorithm. We therefore anticipate the existence of a close relationship between Kalman filtering and the RLS algorithm, which is addressed in the next section.

14.9 KALMAN FILTER AS THE UNIFYING BASIS FOR RLS ALGORITHMS

As mentioned in the introductory remarks to this chapter, a primary reason for our interest in Kalman filter theory is that it provides a unifying framework for the derivation of those linear adaptive filtering algorithms which constitute the family of RLS algorithms.[4] The key question is: Given a Kalman filter or one of its variants based on a stochastic model, how do we derive the corresponding version of the RLS algorithm based on a deterministic model?

To address this fundamental question, we clearly need to formulate *the state-space description for the underlying dynamics of RLS algorithms*. Consider first the special case of an RLS algorithm with an exponential weighting factor $\lambda = 1$. From Chapter 10, we recall that the RLS algorithm is a linear estimator of the multiple regression model of Fig. 10.4, which is reproduced here in the form shown in Fig. 14.7. According to that model, the reference signal or desired response $d(n)$ is related to the input vector $\mathbf{u}(n)$ as

$$d(n) = \mathbf{w}_o^{\mathrm{H}}\mathbf{u}(n) + e_o(n), \tag{14.73}$$

where $\mathbf{w}_o$ is the unknown parameter vector of the model and $e_o(n)$ is the measurement error modeled as white noise. With λ assumed to equal unity, the transition matrix of the state-space model is equal to the identity matrix, a relationship that is intuitively obvious. Moreover, the underlying dynamics of the RLS algorithm are unforced, which means that the system noise is zero. Hence, using the Kalman filter notations adopted in this chapter, we may postulate the following pair of equations for the state-space model of RLS algorithms with $\lambda = 1$:

$$\mathbf{x}(n+1) = \mathbf{x}(n); \tag{14.74}$$

$$y(n) = \mathbf{C}(n)\mathbf{x}(n) + \nu(n). \tag{14.75}$$

In Eq. (14.75), the measurement noise $\nu(n)$ is white with zero mean. A logical choice for $\mathbf{x}(n)$ is the parameter vector $\mathbf{w}_o$. Then, taking the complex conjugate of both sides of Eq. (14.73) and comparing it with the measurement equation (14.75), we deduce the following identities:

$$\left.\begin{aligned} \mathbf{x}(n) &= \mathbf{w}_o, \\ y(n) &= d^*(n), \\ \mathbf{C}(n) &= \mathbf{u}^{\mathrm{H}}(n), \\ \nu(n) &= e_o^*(n), \end{aligned}\right\} \text{ for } \lambda = 1. \tag{14.76}$$

Consider next the more general case of an RLS algorithm with its exponential weighting factor lying in the interval $0 < \lambda \leq 1$. This time, we write

$$\mathbf{x}(n+1) = \mathbf{F}(n+1, n)\mathbf{x}(n) \tag{14.77}$$

[4]The application of Kalman filter theory to adaptive filtering was apparently first reported in the literature by Lawrence and Kaufman (1971). This was followed by Godard (1974), who, in an approach different from that of Lawrence and Kaufman, formulated the adaptive filtering problem (using a tap-delay-line structure) as the estimation of a state vector in Gaussian noise, which represents a classical Kalman filtering problem. Godard's paper prompted many other investigators to explore the application of Kalman filter theory to adaptive filtering problems.

However, we had to await the paper by Sayed and Kailath (1994) to discover how indeed the Riccati-based Kalman filtering algorithm and its variants could be correctly framed into one-to-one correspondence with all the known algorithms in the RLS family.

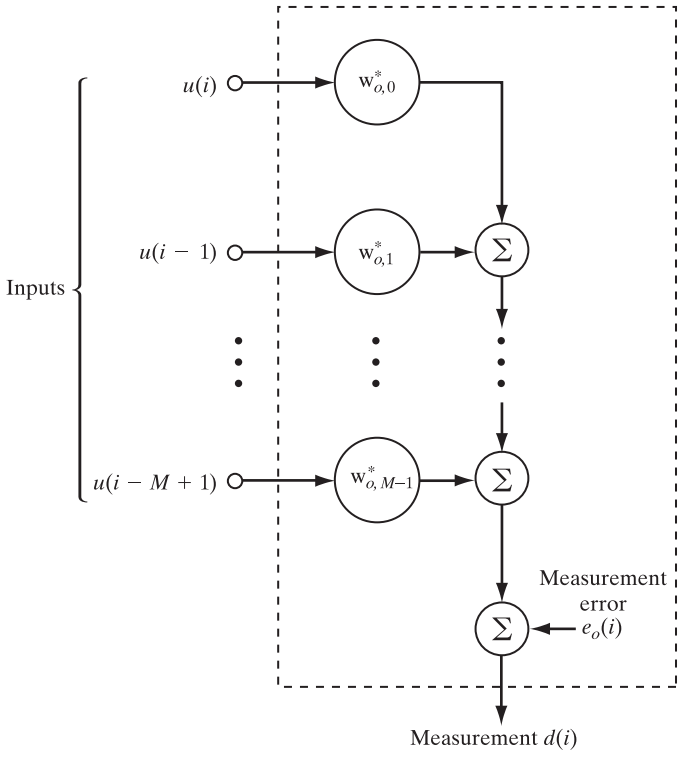


FIGURE 14.7 Multiple linear regression model.

and

$$y(n) = \mathbf{u}^{\mathrm{H}}(n)\mathbf{x}(n) + \nu(n), \tag{14.78}$$

where the only identity retained from those of Eq. (14.76) is $\mathbf{C}(n) = \mathbf{u}^{\mathrm{H}}(n)$, which is justified by the fact that $\mathbf{u}(n)$ is independent of λ. To find the right transition matrix $\mathbf{F}(n+1, n)$ and the correspondence between the measurement equation (14.78) for the Kalman filter and the multiple regression model of Eq. (14.73) for the RLS algorithm with $0 < \lambda \leq 1$, we proceed as follows:

1. From Eq. (14.55), we find that putting $\mathbf{Q}_1(n) = \mathbf{0}$ for an unforced dynamic model yields

$$\mathbf{K}(n+1, n) = \mathbf{F}(n+1, n)\mathbf{K}(n)\mathbf{F}^{\mathrm{H}}(n+1, n). \tag{14.79}$$

Moreover, as will be shown presently, the transition matrix $\mathbf{F}(n+1, n)$ characterizing the unforced dynamic model that we are seeking is of such a form that except for a scaling factor, Eq. (14.79) reduces to

$$\mathbf{K}(n+1, n) = \mathbf{K}(n).$$

Accordingly, henceforth we work on the basis that the predicted state-error correlation matrix $\mathbf{K}(n+1, n)$ and the filtered state-error correlation matrix $\mathbf{K}(n)$ have the same dependence on time n.

2. Assuming that the measurement noise $v(n)$ has unit variance, and using Eqs. (14.35) and (14.49), the formula for the Kalman gain takes the form of a vector defined as

$$\mathbf{g}(n) = \frac{\mathbf{F}(n+1,n)\mathbf{K}(n-1)\mathbf{u}(n)}{1 + \mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1)\mathbf{u}(n)}. \tag{14.80}$$

From Eq. (10.18) of Chapter 10, the gain vector of the RLS algorithm is defined by

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1 + \lambda^{-1}\mathbf{u}^{\mathrm{H}}(n)\mathbf{P}(n-1)\mathbf{u}(n)}. \tag{14.81}$$

The denominators in Eqs. (14.80) and (14.81) are identical if we set

$$\mathbf{K}(n-1) = \lambda^{-1}\mathbf{P}(n-1), \tag{14.82}$$

in light of which we immediately deduce the identity

$$\mathbf{g}(n) = \mathbf{F}(n+1,n)\mathbf{k}(n). \tag{14.83}$$

3. From Eq. (14.45) for the Kalman filter, we have

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \mathbf{F}(n+1,n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{g}(n)\alpha(n), \tag{14.84}$$

where the innovation $\alpha(n)$ is now scalar. Correspondingly, from Eq. (10.25) of Chapter 10 on the RLS algorithm, we have

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n), \tag{14.85}$$

where $\xi(n)$ is the a priori estimation error. Let

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \varphi(n)\hat{\mathbf{w}}(n), \tag{14.86}$$

where $\varphi(n)$ is a scalar function to be determined. Substituting Eqs. (14.83) and (14.86) into Eq. (14.84) and then comparing the resulting equation with Eq. (14.85), we deduce the following two identities:

$$\mathbf{F}(n+1,n)\varphi^{-1}(n)\varphi(n-1) = \mathbf{I} \tag{14.87}$$

and

$$\mathbf{F}(n+1,n)\varphi^{-1}(n)\alpha(n) = \xi^*(n), \tag{14.88}$$

where $\mathbf{I}$ is the identity matrix. Equation (14.87) is satisfied by setting

$$\mathbf{F}(n+1,n) = \lambda^{-1/2}\mathbf{I} \tag{14.89}$$

and

$$\varphi(n) = \lambda^{-(n+1)/2}, \tag{14.90}$$

in which case Eqs. (14.88), (14.86), and (14.83), respectively, reduce to

$$\alpha(n) = \lambda^{-n/2}\xi^*(n), \tag{14.91}$$

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \lambda^{-(n+1)/2}\hat{\mathbf{w}}(n), \tag{14.92}$$

and

$$\mathbf{g}(n) = \lambda^{-1/2}\mathbf{k}(n). \tag{14.93}$$

Also, we now see that, except for the scaling factor λ, the use of Eq. (14.89) in Eq. (14.79) yields $\mathbf{K}(n+1, n) = \mathbf{K}(n)$, confirming the correct basis for step 1.

4. From Eq. (14.31) for the Kalman filter, we have

$$\begin{aligned}\alpha(n) &= y(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) \\ &= y(n) - \lambda^{-n/2}\mathbf{u}^{\mathrm{H}}(n)\hat{\mathbf{w}}(n-1).\end{aligned} \tag{14.94}$$

Correspondingly, from Eq. (10.26) of Chapter 10 on the RLS algorithm, we have

$$\xi(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n-1)\mathbf{u}(n). \tag{14.95}$$

Substituting Eq. (14.91) into Eq. (14.94) and then comparing the result with Eq. (14.95), we deduce the following identity:

$$y(n) = \lambda^{-n/2}d^*(n). \tag{14.96}$$

5. Substituting Eq. (14.96) into the measurement equation (14.78) for the Kalman filter and then comparing the resulting equation with the multiple regression model of Eq. (14.73) for the RLS algorithm, we deduce the identities

$$\mathbf{x}(n) = \lambda^{-n/2}\mathbf{w}_o \tag{14.97}$$

and

$$\nu(n) = \lambda^{-n/2}e_o^*(n). \tag{14.98}$$

6. Using Eqs. (14.35) and (14.63), we may define a scalar *conversion factor* for the Kalman filter described by Eqs. (14.77) and (14.78) as

$$\begin{aligned}r^{-1}(n) &= \frac{e(n)}{\alpha(n)} \\ &= \frac{1}{1 + \mathbf{C}(n)\mathbf{K}(n-1)\mathbf{C}^{\mathrm{H}}(n)} \\ &= \frac{1}{1 + \lambda^{-1}\mathbf{u}^{\mathrm{H}}(n)\mathbf{P}(n-1)\mathbf{u}(n)},\end{aligned} \tag{14.99}$$

where $e(n)$ is the filtered estimation error and $\alpha(n)$ is the innovation for the Kalman filter. The RLS algorithm is characterized by a conversion factor of its own, denoted by $\gamma(n)$ and defined by Eqs. (10.42) and (10.18) of Chapter 10. [Note that the $e(n)$ for the RLS algorithm is not to be confused with the $e(n)$ for the Kalman filter.] Thus, from those equations, we have

$$\gamma(n) = \frac{1}{1 + \lambda^{-1}\mathbf{u}^{\mathrm{H}}(n)\mathbf{P}(n-1)\mathbf{u}(n)}. \tag{14.100}$$

Comparing Eqs. (14.99) and (14.100), we immediately deduce

$$r^{-1}(n) = \gamma(n). \tag{14.101}$$

Thus, on the basis of the identities defined by Eqs. (14.97), (14.96), (14.98), (14.92), (14.82), (14.93), (14.91), and (14.101), we may set up the correspondences between the *stochastic* Kalman variables and the *deterministic* RLS variables summarized in Table 14.3. The left half of the table pertains to Kalman variables and their descriptions

TABLE 14.3 Summary of Correspondences Between Kalman Variables and RLS Variables

Kalman	(Unforced dynamic model of Fig. 14.8)	RLS	(Multiple regression model of Fig. 14.7)
Description	Variable	Variable	Description
Initial value of state	$\mathbf{x}(0)$	$\mathbf{w}_o$	Parameter vector
State	$\mathbf{x}(n)$	$\lambda^{-n/2}\mathbf{w}_o$	Exponentially weighted parameter vector
Measurement	$y(n)$	$\lambda^{-n/2}d^*(n)$	Desired response
Measurement noise	$v(n)$	$\lambda^{-n/2}e_o^*(n)$	Measurement error
One-step prediction of state vector	$\hat{\mathbf{x}}(n+1\,\vert\,\mathcal{Y}_n)$	$\lambda^{-(n+1)/2}\hat{\mathbf{w}}(n)$	Estimate of parameter vector
Correlation matrix of error in state prediction	$\mathbf{K}(n)$	$\lambda^{-1}\mathbf{P}(n)$	Inverse of correlation matrix of input vector
Kalman gain	$\mathbf{g}(n)$	$\lambda^{-1/2}\mathbf{k}(n)$	Gain vector
Innovation	$\alpha(n)$	$\lambda^{-n/2}\xi^*(n)$	A priori estimation error
Conversion factor	$r^{-1}(n)$	$\gamma(n)$	Conversion factor
Initial conditions	$\hat{\mathbf{x}}(1\,\vert\,\mathcal{Y}_0) = \mathbf{0}$ $\mathbf{K}(0)$	$\hat{\mathbf{w}}(0) = \mathbf{0}$ $\lambda^{-1}\mathbf{P}(0)$	Initial conditions

and the right half pertains to the RLS variables. In making up the descriptions for the RLS variables, however, we have ignored (for the sake of simplicity) references to the operations of complex conjugation and multiplication by powers of the exponential weighting vector λ. The only place where the term "exponentially weighted" is used is in the second entry in the table, and this is done merely to distinguish that entry from the first one.

We conclude our discussion of the basic relationship between the Kalman filter and the RLS algorithm by postulating the *unforced state-space model*,

$$\mathbf{x}(n+1) = \lambda^{-1/2}\mathbf{x}(n), \qquad 0 < \lambda \leq 1, \tag{14.102}$$

$$y(n) = \mathbf{u}^{\mathrm{H}}(n)\mathbf{x}(n) + v(n), \tag{14.103}$$

where $\mathbf{x}(n)$ is the state, $\mathbf{u}^{\mathrm{H}}(n)$ is the measurement and $v(n)$ is the measurement noise modeled as white noise with zero mean and unit variance. The exponential weighting factor λ and input vector $\mathbf{u}(n)$ refer to the RLS algorithm. The system equation (14.102) follows from Eqs. (14.77) and (14.89). The measurement equation (14.103) is a repeat of Eq. (14.78), presented here merely for the sake of completeness. Starting with the initial condition $\mathbf{x}(0)$, we readily find from Eq. (14.102) that

$$\mathbf{x}(n) = \lambda^{-n/2}\mathbf{x}(0).$$

Clearly, the relationship between $\mathbf{x}(0)$ and the parameter vector $\mathbf{w}_o$ of the multiple regression model of Fig. 14.7 is independent of the exponential weight factor λ. Hence, from the first line of Eq. (14.76), we may set

$$\mathbf{x}(0) = \mathbf{w}_o.$$

Thus, for an arbitrary λ, we may define the state $\mathbf{x}(n)$ in terms of $\mathbf{w}_o$ as

$$\mathbf{x}(n) = \lambda^{-n/2}\mathbf{w}_o. \tag{14.104}$$

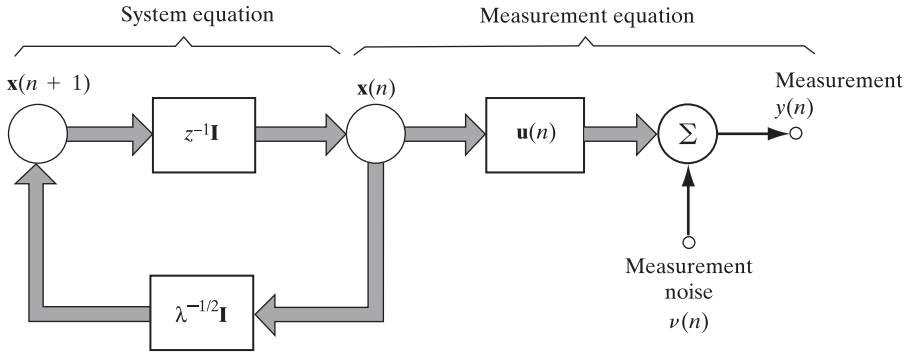


FIGURE 14.8 State-space model of RLS algorithm for an exponential weighting factor in the interval $0 < \lambda \leq 1$.

The unforced linear dynamic model of Eqs. (14.102) and (14.103) is represented by the signal-flow graph of Fig. 14.8, which generalizes the multiple regression model of Fig. 14.7 by including a dynamical component for the evolution of the state. Clearly, the state-space model of Fig. 14.8 is more powerful than the multiple regression model of Fig. 14.7. Note, however, that with the exponential weighting factor λ confined to the interval $0 < \lambda \leq 1$, the state $\mathbf{x}(n)$ remains constant at $\mathbf{w}_o$ or else grows exponentially with adaptation cycle n. (For further discussion of this latter issue, see Problem 12.)

With the unforced dynamic model of Fig. 14.8 and the accompanying Table 14.3 representing the frame of reference for what follows—namely, Sections 14.10 and 14.11—we will respectively study two *adaptive filtering algorithms* that are variants of the Kalman filter:

1. *Covariance filtering algorithm,* which propagates the filtered covariance matrix, $\mathbf{K}(n)$.
2. *Information filtering algorithm,* which propagates the inverse covariance matrix, $\mathbf{K}^{-1}(n)$.

14.10 COVARIANCE FILTERING ALGORITHM

Table 14.4 is a simplified version of Table 14.2, the latter pertaining to the Kalman filter in its most generic form. In actual fact, Table 14.4 is obtained from Table 14.2 by building on Table 14.3, which summarizes the correspondence between the Kalman filter and the RLS algorithm for the unforced dynamic model of Fig. 14.8. One of the other points to note in Table 14.4: The vector $\mathbf{g}(n)$ is the counterpart to the Kalman gain in Table 14.2. Henceforth, we refer to $\mathbf{g}(n)$ as the *gain vector*, which is synonymous with the RLS algorithm.

Divergence Phenomenon

The troublesome aspect of the covariance filtering algorithm[5] is the Riccati equation, represented by the last line under *Computation* in Table 14.4. Specifically, therein we have

$$\mathbf{K}(n) = \lambda^{-1}\mathbf{K}(n-1) - \lambda^{-1/2}\mathbf{g}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1), \tag{14.105}$$

[5]In Kalman filtering, the serious numerical difficulties of this celebrated algorithm are well-documented in the literature (Kaminski et al., 1971; Bierman & Thornton, 1977).

TABLE 14.4 Summary of the Covariance (Kalman) Filtering Algorithm for the Special Unforced Dynamic Model of Fig. 14.8

Input scalar process:

Measurements = $y(1), y(2), \ldots, y(n)$

Known parameters:

Transition matrix = $\mathbf{F}(n+1, n) = \lambda^{-1/2}\mathbf{I}$, $\mathbf{I}$ = identity matrix

Measurement matrix = $\mathbf{C}(n) = \mathbf{u}^{\mathrm{H}}(n)$

Variance of measurement noise $v(n) = \sigma_v^2 = 1$

Initial conditions:

$$\hat{\mathbf{x}}(1|\mathcal{Y}_0) = \mathbb{E}[\mathbf{x}(1)]$$
$$\mathbf{K}(1,0) = \mathbb{E}[(\mathbf{x}(1) - \mathbb{E}[\mathbf{x}(1)])(\mathbf{x}(1) - \mathbb{E}[\mathbf{x}(1)])^{\mathrm{H}}] = \Pi_0$$

Computation: $n = 1, 2, 3, \ldots$

$$\mathbf{g}(n) = \frac{\lambda^{-1/2}\mathbf{K}(n-1)\mathbf{u}(n)}{\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1)\mathbf{u}(n) + 1}$$
$$\alpha(n) = y(n) - \mathbf{u}^{\mathrm{H}}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$$
$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \lambda^{-1/2}\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{g}(n)\alpha(n)$$
$$\mathbf{K}(n) = \lambda^{-1}\mathbf{K}(n-1) - \lambda^{-1/2}\mathbf{g}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1)$$

which is defined as the difference between two nonnegative definite matrices. Hence, unless the numerical accuracy employed at every step of the algorithm is high enough, the filtered covariance matrix, $\mathbf{K}(n)$, resulting from this computation may *not* be nonnegative definite. Such a situation is clearly unacceptable, because $\mathbf{K}(n)$ represents a correlation matrix. The unstable behavior of the covariance filtering algorithm, which results from numerical inaccuracies due to the use of finite wordlength arithmetic, is called the *divergence phenomenon.*

A simple way of overcoming the divergence phenomenon is to artifically add white Gaussian noise to the (noiseless) system equation (14.102) of the unforced dynamic model in Fig. 14.8. The variance of this additive noise is chosen to be just large enough to ensure that the matrix $\mathbf{K}(n)$ is nonnegative for all n.

Square-Root Filtering

A more refined method of overcoming the divergence phenomenon is to use numerically stable unitary transformations at every adaptation cycle of the Kalman filtering algorithm (Potter, 1963; Kaminski et al., 1971; Morf & Kailath, 1975). In particular, the matrix $\mathbf{K}(n)$ is propagated in a square-root form by using the *Cholesky factorization*

$$\mathbf{K}(n) = \mathbf{K}^{1/2}(n)\mathbf{K}^{\mathrm{H}/2}(n), \tag{14.106}$$

where $\mathbf{K}^{1/2}(n)$ is reserved for a lower triangular matrix and $\mathbf{K}^{\mathrm{H}/2}$ is its Hermitian transpose. In linear algebra, the Cholesky factor $\mathbf{K}^{1/2}(n)$ is commonly referred to as the *square root* of the matrix $\mathbf{K}(n)$. Accordingly, any variant of the Kalman filtering algorithm based on the Cholesky factorization is referred to as *square-root filtering*. (The Cholesky factorization was also discussed in Section 3.7 in the context of linear prediction.) The important point to note here is that the matrix product $\mathbf{K}^{1/2}(n)\mathbf{K}^{\mathrm{H}/2}(n)$ is much less

likely to become indefinite, because the product of any square matrix and its Hermitian transpose is always positive definite. Indeed, even in the presence of round-off errors, the numerical conditioning of the Cholesky factor $\mathbf{K}^{1/2}(n)$ is generally much better than that of $\mathbf{K}(n)$ itself.

14.11 INFORMATION FILTERING ALGORITHM

As mentioned previously, the information filtering algorithm is the second variant of the Kalman filter. This new adaptive filtering algorithm, traced back to Fraser (1967), distinguishes itself from the covariance filtering algorithm of Section 14.10 in the following respect: It propagates the inverse covariance matrix, $\mathbf{K}^{-1}(n)$, in place of the filtered covariance matrix, $\mathbf{K}(n)$. By comparison with the covariance filtering algorithm, the information filtering algorithm is not as well-recognized in the literature, nor is it as widely used. However, it does have some unique and useful characteristics of its own, which are discussed in the latter part of the section.

Derivation of the Information Filtering Algorithm

To derive the information filtering algorithm, we may proceed in three steps as follows:

Step 1. Starting with the Riccati Eq. (14.105), we have

$$\lambda^{1/2}\mathbf{K}(n) = \lambda^{-1/2}\mathbf{K}(n-1) - \mathbf{g}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1). \tag{14.107}$$

Next, from the first line of the algorithm in Table 14.4, the gain vector for the unforced dynamic model of Fig. 14.8 is defined by

$$\mathbf{g}(n) = \frac{\lambda^{-1/2}\mathbf{K}(n-1)\mathbf{u}(n)}{\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1)\mathbf{u}(n) + 1}. \tag{14.108}$$

Cross-multiplying and rearranging terms, we may rewrite Eq. (14.108) in the new form:

$$\mathbf{g}(n) = [\lambda^{-1/2}\mathbf{K}(n-1) - (\mathbf{g}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1)]\mathbf{u}(n). \tag{14.109}$$

The right-hand side of Eq. (14.107) shares the same expression inside the square brackets in Eq. (14.109). We may therefore simply define the gain vector as:

$$\mathbf{g}(n) = \lambda^{1/2}\mathbf{K}(n)\mathbf{u}(n). \tag{14.110}$$

Next, eliminating $\mathbf{g}(n)$ between Eqs. (14.107) and (14.110) and multiplying the result by $\lambda^{1/2}$, we get

$$\mathbf{K}(n-1) = \lambda\mathbf{K}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1) + \lambda\mathbf{K}(n). \tag{14.111}$$

Finally for step 1, premultiplying Eq. (14.111) by the inverse matrix $\mathbf{K}^{-1}(n)$ and postmultiplying it by $\mathbf{K}^{-1}(n-1)$, we get the first time update of the information-filtering algorithm:

$$\mathbf{K}^{-1}(n) = \lambda\mathbf{K}^{-1}(n-1) + \lambda\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n). \tag{14.112}$$

Step 2. From the second and third lines of the algorithm summarized in Table 14.4, we have, respectively,

$$\alpha(n) = y(n) - \mathbf{u}^{\mathrm{H}}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) \tag{14.113}$$

and

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \lambda^{-1/2}\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{g}(n)\alpha(n). \tag{14.114}$$

Therefore, substituting Eq. (14.110) into Eq. (14.114), we get

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \lambda^{-1/2}\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \lambda^{1/2}\mathbf{K}(n)\mathbf{u}(n)\alpha(n). \tag{14.115}$$

Next, eliminating $\alpha(n)$ between Eqs. (14.113) and (14.115) yields

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = [\lambda^{-1/2}\mathbf{I} - \lambda^{1/2}\mathbf{K}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \lambda^{1/2}\mathbf{K}(n)\mathbf{u}(n)y(n). \tag{14.116}$$

But, from Eq. (14.111), we readily deduce the following relation:

$$\lambda^{-1/2}\mathbf{I} - \lambda^{1/2}\mathbf{K}(n)\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n) = \lambda^{1/2}\mathbf{K}(n)\mathbf{K}^{-1}(n-1). \tag{14.117}$$

Accordingly, we may simplify Eq. (14.116) to

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \lambda^{1/2}\mathbf{K}(n)\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \lambda^{1/2}\mathbf{K}(n)\mathbf{u}(n)y(n).$$

Premultiplying this equation by the inverse matrix $\mathbf{K}^{-1}(n)$, we get the second time update of the information-filtering algorithm:

$$\mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \lambda^{1/2}[\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{u}(n)y(n)]. \tag{14.118}$$

Step 3. Finally, the updated value of the state's estimate is computed by combining the result of step 2 with the matrix inverse of step 1, as shown by:

$$\begin{aligned}\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) &= \mathbf{K}(n)(\mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)) \\ &= [\mathbf{K}^{-1}(n)]^{-1}(\mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)).\end{aligned} \tag{14.119}$$

Equations (14.112), (14.118), and (14.119), in that order, constitute the information-filtering algorithm for the unforced dynamic model of Eqs. (14.102) and (14.103). A summary of the algorithm is presented in Table 14.5.

Startup Procedure for State Estimation

Given the initial conditions described in Table 14.5, it is quite likely that the inverse covariance matrix, $\mathbf{K}^{-1}(0)$, is singular. In such a situation, a unique estimate of the full state $\mathbf{x}(n)$, using Eq. (14.119), cannot be made unless we develop a viable startup procedure (Maybeck, 1979). To this end, we introduce the matrix

$$\mathbf{\Phi}(n) = \mathbf{K}^{-1}(n). \tag{14.120}$$

Then, according to Eq. (14.111)—that is, the first line in the algorithm summarized in Table 14.5—we may write

$$\mathbf{\Phi}(n) = \lambda[\mathbf{\Phi}(n-1) + \mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]. \tag{14.121}$$

Thus, starting from the initial condition, $\mathbf{\Phi}(0) = \mathbf{K}^{-1}(0)$, the time update of Eq. (14.121) is carried out step-by-step. Once the matrix $\mathbf{\Phi}(n)$, and therefore $\mathbf{K}^{-1}(n)$, becomes nonsingular, the information filtering algorithm as described in Table 14.5, including the final step of computing the state estimate, can proceed forward with no difficulty.

TABLE 14.5 Summary of the Information-Filtering Algorithm for the Special Unforced Dynamic Model of Fig. 14.8

Input scalar process:

Measurements = $y(1), y(2), \ldots, y(n)$

Known parameters:

Transition matrix = $\mathbf{F}(n+1, n) = \lambda^{-1/2}\mathbf{I}$, $\mathbf{I}$ = identity matrix

Measurement matrix = $\mathbf{C}(n) = \mathbf{u}^{\mathrm{H}}(n)$

Variance of measurement noise $v(n) = \sigma_v^2 = 1$

Initial conditions:

$$\hat{\mathbf{x}}(1|\mathcal{Y}_0) = \mathbb{E}[\mathbf{x}(1)]$$
$$\mathbf{K}(1, 0) = \mathbb{E}[(\mathbf{x}(1) - \mathbb{E}[\mathbf{x}(1)])(\mathbf{x}(1) - \mathbb{E}[\mathbf{x}(1)])^{\mathrm{H}}] = \Pi_0$$

Computation: $n = 1, 2, 3 \ldots$

$$\mathbf{K}^{-1}(n) = \lambda[\mathbf{K}^{-1}(n-1) + \mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]$$
$$\mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \lambda^{1/2}[\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{u}(n)y(n)]$$
$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = [\mathbf{K}^{-1}(n)]^{-1}\mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$$

If, however, the state estimate, $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$, is not required until all the measurements have been completely processed, then we may simplify matters by setting the inverse covariance matrix

$$\mathbf{K}^{-1}(0) = \mathbf{0}$$

and proceed accordingly.

Unique Characteristics of the Information Filtering Algorithm

Although the two adaptive filtering algorithms summarized in Tables 14.4 and 14.5 are, indeed, different in their computational routines, they are algebraicly equivalent and therefore require the same number of algebraic operations (i.e., multiplications and divisions). However, the information filtering algorithm has some unique characteristics of its own:

1. The information filtering algorithm has numerical properties that are entirely different from those of the covariance filtering algorithm (Kaminski et al., 1971). To be specific, the covariance filtering algorithm is prone to instability when the algorithm is implemented digitally, the source of which is the Riccati equation. In direct contrast, the information filtering algorithm does not involve the Riccati equation, with the result that it is *digitally stable*.
2. The information filtering algorithm allows the use of a *startup procedure* when the initial condition $\mathbf{K}^{-1}(0)$ is singular (Maybeck, 1979), as described in the previous subsection.
3. The information filtering algorithm can be directly exploited in the design of a *fixed-interval smoother* (Fraser, 1967; Maybeck, 1982). Specifically, the smoother is a combination of two optimum linear filters, one of which operates from the beginning of the input data interval in a *forward* manner and the other from the end of the data interval in a *backward* manner.
4. The information filtering algorithm permits the interpretation of the filtering process in information-theoretic terms, which, as discussed next, is highly insightful.

Fisher Information in the Information Filtering Algorithm

To address the inherent information-theoretic aspect of this algorithm, recall from Chapter 9 that in the course of elaborating on Property 4 of the least-squares estimator, we formally stated that the estimate $\hat{\mathbf{w}}$ achieves the Cramér-Rao lower bound for unbiased estimators. More specifically, therein we said that the inverse covariance matrix of the estimate $\hat{\mathbf{w}}$ is in fact the *expected Fisher information matrix* for the method of least squares. To be precise, using Eqs. (9.63) and (9.66), we defined this matrix, denoted by **J,** as follows:

$$\begin{aligned}\mathbf{J} &= \mathbb{E}\left[\left(\frac{\partial l}{\partial \mathbf{w}}\right)\left(\frac{\partial l}{\partial \mathbf{w}^{\mathrm{H}}}\right)\right] \\ &= [\mathrm{cov}(\hat{\mathbf{w}})]^{-1},\end{aligned}$$

where l is the log-likelihood function of the estimation error vector

$$\boldsymbol{\varepsilon}_o = \mathbf{w}_o - \hat{\mathbf{w}}$$

and where $\mathbf{w}_o$ is the unknown parameter vector of the multiple linear regression model of Fig. 9.1.

In basic terms, Fig. 9.1 is one and the same as Fig. 14.7. In a corresponding way, we may therefore make the statement:

> The inverse covariance matrix, $\mathbf{K}^{-1}$, used in the information filtering algorithm summarized in Table 14.5, is the expected Fisher information matrix.

Indeed, it is on account of this statement that the information filtering algorithm justifiably bears its name.

Moreover, in light of Frieden's book (2004), we may go on to say that in the context of the information filtering algorithm, *Fisher information* plays two key roles:

1. Fisher information provides a *measure* of how capable the information filtering algorithm is in estimating an element of the unknown state; in a generic sense, this role represents the cornerstone of statistics.[6]
2. Fisher information provides a measure of the state of *disorder* experienced by the algorithm, should it ever occur.

One last comment is in order: Fisher information is of a *localized* kind, which, therefore, distinguishes it from the notion of entropy in Shannon's information theory, which is *global*.

14.12 SUMMARY AND DISCUSSION

The Kalman filter is a linear, discrete-time, finite-dimensional system endowed with a recursive structure that makes a digital computer well suited for its implementation. A key property of the Kalman filter is that it is the *minimum mean-square (variance) estimator of the state* of a linear dynamic model, which follows from a stochastic state-space model.

[6]In Cavanaugh and Shumway (1996), a recursive algorithm is derived for computing the expected Fisher information matrix from state-state parameters using real-valued data; the algorithm is also supported by the use of simulations.

In the context of the family of linear adaptive filtering algorithms rooted in *deterministic least-squares estimation,* Kalman filter theory is of profound theoretic as well as practical importance for two reasons:

1. The underlying dynamics of recursive least-squares (RLS) algorithms, characterized by an exponential weighting factor λ in the interval $0 < \lambda \leq 1$, are described by the unforced dynamic model of Fig. 14.8. This model provides the basis for constructing the list of the correspondences between Kalman variables and RLS variables, summarized in Table 14.3, which was constructed by a careful comparison of the Kalman filtering algorithm of Table 14.2 and the RLS filtering algorithm of Table 10.1.
2. The literature on Kalman filters is extensive. Hence, we may exploit it to derive variants of RLS algorithms, namely, square-root RLS algorithms, as will be demonstrated in Chapter 15. Although these variants have been known for a long time, it is in Kalman filter theory that we have the mathematical basis for their derivations in a unified manner.

Numerous commentators have said that many of the problems in signal processing and control theory are mathematically equivalent. With the recognition that Kalman filter theory is rooted in the control literature, the link between Kalman filters and linear adaptive filters established in this chapter is further testimony to the relevance of that mathematical equivalence.

PROBLEMS

1. The Gram–Schmidt orthogonalization procedure enables the set of measurement vectors $y(1)$, $y(2), \ldots, y(n)$ to be transformed into the set of innovations $\alpha(1), \alpha(2), \ldots, \alpha(n)$ without loss of information, and vice versa. Illustrate this procedure for $n = 3$.
2. Show that the predicted state-error vector $\epsilon(n, n-1)$ is orthogonal to both $v_1(n)$ and $v_2(n)$.
3. Consider a set of scalar measurements $y(n)$ of zero mean that is transformed into the corresponding set of innovations $\alpha(n)$ of zero mean and variance $\sigma_\alpha^2(n)$. Let the estimate of the state vector $\mathbf{x}(i)$, given this set of data, be expressed as

$$\hat{\mathbf{x}}(i|\mathcal{Y}_n) = \sum_{k=1}^{n} \mathbf{b}_i(k)\alpha(k),$$

where $\mathcal{Y}_n$ is the space spanned by $y(1), \ldots, y(n)$ and $\mathbf{b}_i(k), k = 1, 2, \ldots, n$, is a set of vectors to be determined. The requirement is to choose the $\mathbf{b}_i(k)$ so as to minimize the expected value of the squared norm of the estimated state-error vector

$$\boldsymbol{\varepsilon}(i|\mathcal{Y}_n) = \mathbf{x}(i) - \hat{\mathbf{x}}(i|\mathcal{Y}_n).$$

Show that this minimization yields the result

$$\hat{\mathbf{x}}(i|\mathcal{Y}_n) = \sum_{k=1}^{n} \mathbb{E}[\mathbf{x}(i)\phi^*(k)]\phi(k),$$

where

$$\phi(k) = \frac{\alpha(k)}{\sigma_\alpha(k)}$$

is the normalized innovation. This result may be viewed as a special case of Eqs. (14.37) and (14.40).

4. The Kalman gain $\mathbf{G}(n)$, defined in Eq. (14.49), involves the inverse matrix $\mathbf{R}^{-1}(n)$. The matrix $\mathbf{R}(n)$ is itself defined in Eq. (14.35), reproduced here for convenience:

$$\mathbf{R}(n) = \mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^{\mathrm{H}}(n) + \mathbf{Q}_2(n).$$

The matrix $\mathbf{C}(n)$ is nonnegative definite, but not necessarily nonsingular.

(a) Why is $\mathbf{R}(n)$ nonnegative definite?

(b) What prior condition would you impose on the matrix $\mathbf{Q}_2(n)$ to ensure that the inverse matrix $\mathbf{R}^{-1}(n)$ exists?

5. In many cases, the predicted state-error correlation matrix $\mathbf{K}(n+1, n)$ converges to the steady-state value $\mathbf{K}$ as the number of adaptation cycles, n, approaches infinity. Show that the limiting value $\mathbf{K}$ satisfies the *algebraic Riccati equation*

$$\mathbf{K}\mathbf{C}^{\mathrm{H}}(\mathbf{C}\mathbf{K}\mathbf{C}^{\mathrm{H}} + \mathbf{Q}_2)^{-1}\mathbf{C}\mathbf{K} - \mathbf{Q}_1 = \mathbf{O},$$

where it is assumed that the state transition matrix equals the identity matrix and the matrices $\mathbf{C}$, $\mathbf{Q}_1$, and $\mathbf{Q}_2$ are the limiting values of $\mathbf{C}(n)$, $\mathbf{Q}_1(n)$, and $\mathbf{Q}_2(n)$, respectively.

6. A second-order tracking system is described by the following state-space equations:

$$x(n+1) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} x(n) + v(n)$$

$$y(n) = [1 \;\; 0]x(n) + v_2(n)$$

The system noise $v(n)$ is white with zero mean and correlation matrix equal to the identity matrix. Likewise, the measurement noise $v_2(n)$ is white with zero mean and unit variance. Using Table 14.2, formulate the recursions for computing the Kalman filter.

7. Using the initial conditions described in Eqs. (14.71) and (14.72), show that the resulting filtered estimate $\hat{\mathbf{x}}(n \mid \mathcal{Y}_n)$ produced by the Kalman filter is unbiased; that is, show that

$$\mathbb{E}[\hat{\mathbf{x}}(n) \mid \mathcal{Y}(n)] = \mathbf{x}(n).$$

8. Derivation of the Kalman filter presented in Section 14.4 is based on the notion of minimum mean-square error estimation. In this problem, we explore another derivation of the Kalman filter based on the *maximum a posteriori probability (MAP) criterion*. For this derivation, it is assumed that the system noise $\boldsymbol{\nu}_1(n)$ and measurement noise $\boldsymbol{\nu}_2(n)$ are both zero-mean Gaussian processes with correlation matrices $\mathbf{Q}_1(n)$ and $\mathbf{Q}_2(n)$, respectively. Let $f_X(\mathbf{x}(n)|\mathcal{Y}_n)$ denote the conditional probability density function (pdf) of $\mathbf{x}(n)$, given that $\mathcal{Y}_n$ denotes the set of measurements $\mathbf{y}(1), \ldots, \mathbf{y}(n)$. The MAP estimate of $\mathbf{x}(n)$, denoted by $\hat{\mathbf{x}}_{\mathrm{MAP}}(n)$, is defined as that particular value of $\mathbf{x}(n)$ that maximizes $f_X(\mathbf{x}(n) \mid \mathcal{Y}_n)$ or, equivalently, the logarithm of $f_X(\mathbf{x}(n) \mid \mathcal{Y}_n)$.

This evaluation requires that we solve for the condition

$$\left.\frac{\partial \ln f_X(\mathbf{x}(n) \mid \mathcal{Y}_n)}{\partial \mathbf{x}(n)}\right|_{\mathbf{x}(n)=\hat{\mathbf{x}}_{\mathbf{MAP}}(n)} = \mathbf{0} \tag{1}$$

and show that

$$\left.\frac{\partial^2 \ln f_X(\mathbf{x}(n)\,|\,\mathcal{Y}_n)}{\partial^2 \mathbf{x}(n)}\right|_{\mathbf{x}(n)=\hat{\mathbf{x}}_{\mathbf{MAP}}(n)} < \mathbf{0}. \tag{2}$$

To proceed, do the following:

(a) Using Bayes' rule, we may express $f_X(\mathbf{x}(n)\,|\,\mathcal{Y}_n)$ as follows:

$$f_X(\mathbf{x}(n)\,|\,\mathcal{Y}_n) = \frac{f_{XY}(\mathbf{x}(n), \mathcal{Y}_n)}{f_Y(\mathcal{Y}_n)},$$

which, in light of the definition of a joint pdf, may also be expressed as

$$f_X(\mathbf{x}(n)\,|\,\mathcal{Y}_n) = \frac{f_{XY}(\mathbf{x}(n), \mathbf{y}(n), \mathcal{Y}_{n-1})}{f_Y(\mathbf{y}(n), \mathcal{Y}_{n-1})}.$$

Hence, show that.

$$f_X(\mathbf{x}(n)\,|\,\mathcal{Y}_n) = \frac{f_Y(\mathbf{y}(n)\,|\,\mathbf{x}(n))f_X(\mathbf{x}(n)\,|\,\mathcal{Y}_{n-1})}{f_Y(\mathbf{y}(n), \mathcal{Y}_{n-1})}.$$

(b) Using the Gaussian characterizations of the system noise $\boldsymbol{\nu}_1(n)$ and measurement $\boldsymbol{\nu}_2(n)$, derive expressions for $f_Y(\mathbf{y}(n)\,|\,\mathbf{x}(n))$ and $f_X(\mathbf{x}(n)\,|\,\mathcal{Y}_{n-1})$. Next, recognizing that $f_Y(\mathbf{y}(n)\,|\,\mathcal{Y}_{n-1})$ may be treated as a constant since it does not depend on the state $\mathbf{x}(n)$, formulate the expression for $f_X(\mathbf{x}(n)\,|\,\mathcal{Y}_n)$.

(c) Using the results of part (b) in Eq. (1) of the problem, followed by the matrix inversion lemma discussed in Chapter 10, derive the formula for $\hat{\mathbf{x}}_{\mathrm{MAP}}(n)$ and show that it is exactly the same as the Kalman filter derived in Section 14.4.

(d) Finally, using Eq. (2) of the problem, show that the MAP estimate $\hat{\mathbf{x}}_{\mathrm{MAP}}(n)$ derived in part (c) does indeed satisfy this equation.

9. Consider a linear dynamic model described by the noiseless state-space model

$$\begin{aligned}\mathbf{x}(n+1) &= \mathbf{F}\mathbf{x}(n),\\ \mathbf{y}(n) &= \mathbf{C}\mathbf{x}(n),\end{aligned}$$

where $\mathbf{x}(n)$ is the state, $\mathbf{y}(n)$ is the measurement, $\mathbf{F}$ is the transition matrix, and $\mathbf{C}$ is the measurement matrix.

(a) Show that

$$\begin{aligned}\hat{\mathbf{x}}(n\,|\,\mathcal{Y}_n) &= \mathbf{F}(\mathbf{I} - \mathbf{G}(n)\mathbf{C})\hat{\mathbf{x}}(n\,|\,\mathcal{Y}_{n-1}) + \mathbf{C}\mathbf{G}(n)\mathbf{y}(n)\\ \boldsymbol{\alpha}(n) &= \mathbf{y}(n) - \mathbf{C}\hat{\mathbf{x}}(n\,|\,\mathcal{Y}_{n-1}),\end{aligned}$$

where $\mathbf{G}(n)$ is the Kalman gain and $\boldsymbol{\alpha}(n)$ denotes the innovations. How is $\mathbf{G}(n)$ defined?

(b) Using the results of part (a), justify the statement that the Kalman filter is a whitening filter in that it produces a "white" estimation error in response to $\mathbf{y}(n)$.

10. In this problem, we explore another procedure for establishing some of the correspondences presented in Table 14.3.

(a) Starting with the multiple regression model of Fig. 14.7, set up the system of linear simultaneous equations that define the desired response $d(i)$ in terms of the input vector $\mathbf{u}(i)$ of an RLS algorithm for $i = 0, 1, \ldots, n$.

(b) Starting with the unforced dynamic model of Fig. 14.8, set up the corresponding system of linear simultaneous equations that define the measurement $y(i)$ in terms of the state $\mathbf{x}(i)$ of a Kalman filter for $i = 0, 1, \ldots, n$.

(c) Using the results of parts (a) and (b), derive the following identities between the RLS algorithm and Kalman filter:

$$\begin{aligned}\mathbf{x}(0) &= \mathbf{w}_o;\\ y(n) &= \lambda^{-n/2} d^*(n);\\ \nu(n) &= \lambda^{-n/2} e_o^*(n).\end{aligned}$$

11. The system equation (14.102) for an RLS algorithm describes the evolution of the state as

$$\mathbf{x}(n+1) = \lambda^{-1/2}\mathbf{x}(n).$$

With the exponential weighting factor λ in the interval $0 < \lambda \leq 1$, the Euclidean norm of the state $\mathbf{x}(n)$ grows unboundedly with adaptation cycle n for $\lambda < 1$. Yet the RLS algorithm operates satisfactorily despite this seemingly abnormal behavior. Why?

12. The last two entries in Table 14.3 pertain to one-to-one correspondence between the initial conditions of the Kalman variables and those of the RLS variables. Justify these two entries.

13. What is a predicted state error? Illustrate its importance and derive the predicted state error correlation matrix.

14. Describe the covariance (Kalman) filtering algorithm for the special unforced dynamic model in Fig. 14.8.

15. Describe the basic relationship between the Kalman filter and the RLS algorithm by postulating the unforced state-space model.

CHAPTER 15

Square-Root Adaptive Filtering Algorithms

One of the problems encountered in applying the recursive least-squares (RLS) algorithm of Chapter 10 is that of numerical instability, which can arise because of the way in which the Riccati difference equation is formulated. This same problem is also known to arise in the classical Kalman filtering algorithm for exactly the same reason. In Chapter 14, we pointed out that the instability (divergence) problem encountered in a Kalman filter can be ameliorated by using a square-root variant of the filter. At that point in the discussion, we deferred a detailed treatment of square-root Kalman filtering until we would be ready for it. In this chapter, we take up a full discussion of this issue, beginning in the next section. The solution to the square-root Kalman filtering problem sets the stage for deriving the corresponding variants of the RLS algorithm in light of the one-to-one correspondences that exist between the Kalman variables and the RLS variables, established in the previous chapter.

15.1 SQUARE-ROOT KALMAN FILTERS

The recursions in a Kalman filter of the covariance type propagate the matrix $\mathbf{K}(n)$, which denotes the correlation matrix of the error in the filtered state estimate; this propagation takes place via the Riccati difference equation. The recursions in a root-square Kalman filter, on the other hand, propagate a lower triangular matrix $\mathbf{K}^{1/2}(n)$, defined as the *square root* of $\mathbf{K}(n)$. The relation between $\mathbf{K}(n)$ and $\mathbf{K}^{1/2}(n)$ is defined by

$$\mathbf{K}(n) = \mathbf{K}^{1/2}(n)\mathbf{K}^{\mathrm{H}/2}(n), \tag{15.1}$$

where the upper triangular matrix $\mathbf{K}^{\mathrm{H}/2}(n)$ is the Hermitian transpose of $\mathbf{K}^{1/2}(n)$. Unlike the situation that may exist with the covariance Kalman filter, the nonnegative definite character of $\mathbf{K}(n)$ as a correlation matrix is preserved by virtue of the fact that the product of any square matrix and its Hermitian transpose is always a nonnegative definite matrix.

In this section, we derive the square-root forms of the covariance and information processing variants of the Kalman filter. However, with the covariance and information filtering algorithms in mind, it is natural that we focus our attention on the special unforced dynamic model developed in Section 14.8. In this model,

$$\mathbf{x}(n+1) = \lambda^{-1/2}\mathbf{x}(n) \tag{15.2}$$

and

$$y(n) = \mathbf{u}^{\mathrm{H}}(n)\mathbf{x}(n) + \nu(n), \tag{15.3}$$

where $\mathbf{x}(n)$ is the state vector, the row vector $\mathbf{u}^{\mathrm{H}}(n)$ is the measurement matrix, the scalar $y(n)$ is an observable or reference signal, the scalar $\nu(n)$ is a white-noise process of zero mean and unit variance, and the positive real scalar λ is a constant of the model. Equations (15.2) and (15.3) are the same as Eqs. (14.102) and (14.103), respectively; they have been reproduced here for convenience of presentation. Before proceeding with derivations of square-root covariance and information filtering algorithms, we digress briefly to consider a lemma in matrix algebra that is pivotal to our present discussion.

Matrix Factorization Lemma

Given any two N-by-M matrices $\mathbf{A}$ and $\mathbf{B}$ with dimension $N \leq M$, the *matrix factorization lemma* states that (Stewart, 1973; Sayed & Kailath, 1994; Golub & Van Loan, 1996)

$$\mathbf{A}\mathbf{A}^{\mathrm{H}} = \mathbf{B}\mathbf{B}^{\mathrm{H}} \tag{15.4}$$

if, and only if, there exists a unitary matrix $\boldsymbol{\Theta}$ such that

$$\mathbf{B} = \mathbf{A}\boldsymbol{\Theta}. \tag{15.5}$$

Assuming that the condition (15.5) holds, we readily find that

$$\mathbf{B}\mathbf{B}^{\mathrm{H}} = \mathbf{A}\boldsymbol{\Theta}\boldsymbol{\Theta}^{\mathrm{H}}\mathbf{A}^{\mathrm{H}}. \tag{15.6}$$

From the definition of a unitary matrix, we have

$$\boldsymbol{\Theta}\boldsymbol{\Theta}^{\mathrm{H}} = \mathbf{I}, \tag{15.7}$$

where $\mathbf{I}$ is the identity matrix. Hence, Eq. (15.6) reduces immediately to Eq. (15.4).

Conversely, the equality described in Eq. (15.4) implies that the matrices $\mathbf{A}$ and $\mathbf{B}$ must be related. We may prove the converse implication of the matrix factorization lemma by invoking the singular-value decomposition theorem, according to which the matrix $\mathbf{A}$ may be factored as (see Section 9.11):

$$\mathbf{A} = \mathbf{U}_A\boldsymbol{\Sigma}_A\mathbf{V}_A^{\mathrm{H}}, \tag{15.8}$$

where $\mathbf{U}_A$ and $\mathbf{V}_A$ are N-by-N and M-by-M unitary matrices, respectively, and $\boldsymbol{\Sigma}_A$ is an N-by-M matrix defined by the singular values of matrix $\mathbf{A}$. Similarly, the second matrix $\mathbf{B}$ may be factored as

$$\mathbf{B} = \mathbf{U}_B\boldsymbol{\Sigma}_B\mathbf{V}_B^{\mathrm{H}}. \tag{15.9}$$

The identity $\mathbf{A}\mathbf{A}^{\mathrm{H}} = \mathbf{B}\mathbf{B}^{\mathrm{H}}$ implies that we have

$$\mathbf{U}_A = \mathbf{U}_B \tag{15.10}$$

and

$$\boldsymbol{\Sigma}_A = \boldsymbol{\Sigma}_B. \tag{15.11}$$

Now, let

$$\boldsymbol{\Theta} = \mathbf{V}_A\mathbf{V}_B^{\mathrm{H}}. \tag{15.12}$$

Then, using Eqs. (15.8) and (15.12) to evaluate the matrix product $\mathbf{A}\boldsymbol{\Theta}$, we obtain a result equal to matrix $\mathbf{B}$ by virtue of Eqs. (15.9) to (15.11); this result is precisely the converse implication of the matrix factorization lemma.

Square-Root Covariance Filtering Algorithm

Returning to the issue of square-root Kalman filtering, we note that the Riccati difference equation for the covariance filtering algorithm may be expressed as follows (by combining the first and final lines of the algorithm summarized in Table 14.4):

$$\mathbf{K}(n) = \lambda^{-1}\mathbf{K}(n-1) - \lambda^{-1}\mathbf{K}(n-1)\mathbf{u}(n)r^{-1}(n)\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1). \quad (15.13)$$

The scalar $r(n)$ is the variance of the filtered estimation error and is defined by

$$r(n) = \mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1)\mathbf{u}(n) + 1. \quad (15.14)$$

The following four distinct matrix terms constitute the right-hand side of the Riccati equation (15.13):

1. Scalar: $\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1)\mathbf{u}(n) + 1$.
2. 1-by-M vector: $\lambda^{-1/2}\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1)$.
3. M-by-1 vector: $\lambda^{-1/2}\mathbf{K}(n-1)\mathbf{u}(n)$.
4. M-by-M matrix: $\lambda^{-1}\mathbf{K}(n-1)$.

Keeping in mind the dimensional compatibility of these four terms, we may arrange these four terms in the form of a block matrix that contains the complete information on $\mathbf{K}(n)$:

$$\mathbf{H}(n) = \begin{bmatrix} \mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1)\mathbf{u}(n) + 1 & \lambda^{-1/2}\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1) \\ \lambda^{-1/2}\mathbf{K}(n-1)\mathbf{u}(n) & \lambda^{-1}\mathbf{K}(n-1) \end{bmatrix}. \quad (15.15)$$

Expressing the correlation matrix $\mathbf{K}(n-1)$ in its factored form yields

$$\mathbf{K}(n-1) = \mathbf{K}^{1/2}(n-1)\mathbf{K}^{\mathrm{H}/2}(n-1). \quad (15.16)$$

Recognizing that the matrix $\mathbf{H}(n)$ is a nonnegative-definite matrix, we may use Cholesky factorization to express Eq. (15.15) as

$$\mathbf{H}(n) = \begin{bmatrix} 1 & \mathbf{u}^{\mathrm{H}}(n)\mathbf{K}^{1/2}(n-1) \\ \mathbf{0} & \lambda^{-1/2}\mathbf{K}^{1/2}(n-1) \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{K}^{\mathrm{H}/2}(n-1)\mathbf{u}(n) & \lambda^{-1/2}\mathbf{K}^{\mathrm{H}/2}(n-1) \end{bmatrix}, \quad (15.17)$$

where $\mathbf{0}$ is the null vector and the superscript T denotes *transposition*.

The matrix product on the right-hand side of Eq. (15.17) may be interpreted as the product of matrix $\mathbf{A}$, say, and its Hermitian transpose $\mathbf{A}^{\mathrm{H}}$. The stage is therefore set for invoking the matrix factorization lemma, according to which we may write

$$\underbrace{\begin{bmatrix} 1 & \mathbf{u}^{\mathrm{H}}(n)\mathbf{K}^{1/2}(n-1) \\ \mathbf{0} & \lambda^{-1/2}\mathbf{K}^{1/2}(n-1) \end{bmatrix}}_{\mathbf{A}} \boldsymbol{\Theta}(n) = \underbrace{\begin{bmatrix} b_{11}(n) & \mathbf{0}^{\mathrm{T}} \\ \mathbf{b}_{21}(n) & \mathbf{B}_{22}(n) \end{bmatrix}}_{\mathbf{B}}, \quad (15.18)$$

where $\mathbf{\Theta}(n)$ is a *unitary rotation* and the scalar $b_{11}(n)$, the vector $\mathbf{b}_{21}(n)$, and the matrix $\mathbf{B}_{22}(n)$ denote the nonzero block elements of matrix $\mathbf{B}$. In Eq. (15.18), we may distinguish two arrays of numbers:

1. *Prearray*, $\mathbf{A}$, which is operated on by a unitary rotation.
2. *Postarray*, $\mathbf{B}$, which is characterized by a block zero entry resulting from the action of the unitary rotation. The postarray therefore has a "triangular" structure in a block sense.

To evaluate the unknown block elements b_{11}, $\mathbf{b}_{21}$, and $\mathbf{B}_{22}$ of the postarray, we proceed by squaring both sides of Eq. (15.18). Then, recognizing that $\mathbf{\Theta}(n)$ is a unitary matrix and, therefore, $\mathbf{\Theta}(n)\mathbf{\Theta}^{\mathrm{H}}(n)$ equals the identity matrix for all n, we may write

$$\underbrace{\begin{bmatrix} 1 & \mathbf{u}^{\mathrm{H}}(n)\mathbf{K}^{1/2}(n-1) \\ \mathbf{0} & \lambda^{-1/2}\mathbf{K}^{1/2}(n-1) \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{K}^{\mathrm{H}/2}(n-1)\mathbf{u}(n) & \lambda^{-1/2}\mathbf{K}^{\mathrm{H}/2}(n-1) \end{bmatrix}}_{\mathbf{A}^{\mathrm{H}}} = \underbrace{\begin{bmatrix} b_{11}(n) & \mathbf{0}^{\mathrm{T}} \\ \mathbf{b}_{21}(n) & \mathbf{B}_{22}(n) \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} b_{11}^{*}(n) & \mathbf{b}_{21}^{\mathrm{H}}(n) \\ \mathbf{0} & \mathbf{B}_{22}^{\mathrm{H}}(n) \end{bmatrix}}_{\mathbf{B}^{\mathrm{H}}}, \tag{15.19}$$

where the asterisk denotes *complex conjugation*. Expanding the matrix products and then comparing the respective terms on both sides of Eq. (15.19), we get the following three identities:

$$|b_{11}(n)|^2 = \mathbf{u}^{\mathrm{H}}(n)\mathbf{K}(n-1)\mathbf{u}(n) + 1 = r(n); \tag{15.20}$$

$$\mathbf{b}_{21}(n)b_{11}^{*}(n) = \lambda^{-1/2}\mathbf{K}(n-1)\mathbf{u}(n); \tag{15.21}$$

$$\mathbf{b}_{21}(n)\mathbf{b}_{21}^{\mathrm{H}}(n) + \mathbf{B}_{22}(n)\mathbf{B}_{22}^{\mathrm{H}}(n) = \lambda^{-1}\mathbf{K}(n-1). \tag{15.22}$$

Equations (15.20) through (15.22) may be satisfied by choosing

$$b_{11}(n) = r^{1/2}(n), \tag{15.23}$$

$$\mathbf{b}_{21}(n) = \lambda^{-1/2}\mathbf{K}(n-1)\mathbf{u}(n)r^{-1/2}(n) = \mathbf{g}(n)r^{1/2}(n), \tag{15.24}$$

and

$$\mathbf{B}_{22}(n) = \mathbf{K}^{1/2}(n), \tag{15.25}$$

where, in the second line, $\mathbf{g}(n)$ denotes the gain vector, a definition of which is given in the first computation step of Table 14.4.

We may thus rewrite Eq. (15.18) as

$$\begin{bmatrix} 1 & \mathbf{u}^{\mathrm{H}}(n)\mathbf{K}^{1/2}(n-1) \\ \mathbf{0} & \lambda^{-1/2}\mathbf{K}^{1/2}(n-1) \end{bmatrix}\mathbf{\Theta}(n) = \begin{bmatrix} r^{1/2}(n) & \mathbf{0}^{\mathrm{T}} \\ \mathbf{g}(n)r^{1/2}(n) & \mathbf{K}^{1/2}(n) \end{bmatrix}. \tag{15.26}$$

The block elements of the prearray and postarray in Eq. (15.26) deserve close scrutiny, as they reveal properties of their own:

- The block elements $\lambda^{-1/2}\mathbf{K}^{1/2}(n-1)$ and $\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}^{1/2}(n-1)$ of the prearray uniquely characterize the constitution of the quantities contained in the right-hand side

of the Riccati difference equation (15.13), except for $r(n)$. Correspondingly, the block element $\mathbf{K}^{1/2}(n)$ of the postarray provides the quantity needed to update the prearray and, therefore, initiate the next adaptation cycle of the algorithm.

- Inclusion of the block elements 1 and $\mathbf{0}$ in the prearray induces the generation of two block elements in the postarray, namely, $r^{1/2}(n)$ and $\mathbf{g}(n)r^{1/2}(n)$. These elements make it possible to calculate two useful variables: the gain vector $\mathbf{g}(n)$ and the variance $r(n)$ of the filtered estimation error. The variance is obtained simply by squaring the scalar entry $r^{1/2}(n)$. The gain vector is obtained equally simply by dividing the block entry $\mathbf{g}(n)r^{1/2}(n)$ by $r^{1/2}(n)$.

Building on the latter result, we may readily time update of the state estimate as

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \lambda^{-1/2}\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{g}(n)\alpha(n), \tag{15.27}$$

where

$$\alpha(n) = y(n) - \mathbf{u}^{\mathrm{H}}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) \tag{15.28}$$

is the *innovation*.

Equations (15.28) and (15.27) are respectively taken from the second and third computation steps of Table 14.4. Part 1 of Table 15.1 presents a summary of the computations performed in the square-root covariance filtering algorithm (Sayed & Kailath, 1994). The initialization of the algorithm proceeds in exactly the same way as for the conventional covariance filtering algorithm (see Table 14.4).

Square-Root Information Filtering Algorithm

Consider next the square-root implementation of information filtering algorithm, which propagates the inverse matrix $\mathbf{K}^{-1}(n)$ rather than $\mathbf{K}(n)$ itself. This form of propagation is useful particularly when there exist large initial uncertainties (i.e., the initial covariance matrix $\mathbf{K}(0)$ is singular). A summary of the information-filtering algorithm is

TABLE 15.1 Summary of the Computations Performed in Square-Root Covariance Filtering Algorithm

1. *Square-root covariance filtering algorithm*:

$$\begin{bmatrix} 1 & \mathbf{u}^{\mathrm{H}}(n)\mathbf{K}^{1/2}(n-1) \\ \mathbf{0} & \lambda^{-1/2}\mathbf{K}^{1/2}(n-1) \end{bmatrix}\boldsymbol{\Theta}(n) = \begin{bmatrix} r^{1/2}(n) & \mathbf{0}^{\mathrm{T}} \\ \mathbf{g}(n)r^{1/2}(n) & \mathbf{K}^{1/2}(n) \end{bmatrix}$$

$$\mathbf{g}(n) = (\mathbf{g}(n)r^{1/2}(n))(r^{1/2}(n))^{-1}$$

$$\alpha(n) = y(n) - \mathbf{u}^{\mathrm{H}}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$$

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \lambda^{-1/2}\,\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{g}(n)\alpha(n)$$

2. *Square-root information filtering algorithm*:

$$\begin{bmatrix} \lambda^{1/2}\mathbf{K}^{-\mathrm{H}/2}(n-1) & \lambda^{1/2}\mathbf{u}(n) \\ \hat{\mathbf{x}}^{\mathrm{H}}(n|\mathcal{Y}_{n-1})\mathbf{K}^{-\mathrm{H}/2}(n-1) & y^*(n) \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}\boldsymbol{\Theta}(n) = \begin{bmatrix} \mathbf{K}^{-\mathrm{H}/2}(n) & \mathbf{0} \\ \hat{\mathbf{x}}^{\mathrm{H}}(n+1|\mathcal{Y}_n)\mathbf{K}^{-\mathrm{H}/2}(n) & r^{-1/2}(n)\alpha^*(n) \\ \lambda^{1/2}\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}^{1/2}(n) & r^{-1/2}(n) \end{bmatrix}$$

$$\hat{\mathbf{x}}^{\mathrm{H}}(n+1|\mathcal{Y}_n) = (\hat{\mathbf{x}}^{\mathrm{H}}(n+1|\mathcal{Y}_n)\mathbf{K}^{-\mathrm{H}/2}(n))(\mathbf{K}^{-\mathrm{H}/2}(n))^{-1}$$

Note: In both algorithms, $\boldsymbol{\Theta}(n)$ is a unitary rotation that produces a block zero entry in the top row of the postarray.

presented in Table 14.5. The first two time updates of the algorithm are reproduced here for convenience of presentation:

$$\mathbf{K}^{-1}(n) = \lambda \mathbf{K}^{-1}(n-1) + \lambda \mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n); \tag{15.29}$$

$$\mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \lambda^{1/2}\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \lambda^{1/2}\mathbf{u}(n)y(n). \tag{15.30}$$

Let the inverse matrix $\mathbf{K}^{-1}(n)$ be expressed in its factored form:

$$\mathbf{K}^{-1}(n) = \mathbf{K}^{-\mathrm{H}/2}(n)\mathbf{K}^{-1/2}(n). \tag{15.31}$$

For reasons that will become apparent presently, we find it more convenient to express Eqs. (15.29) and (15.30) in their Hermitian transposed forms, in which case we may express the four quantities on the right-hand sides of these two equations in their individual factored forms as follows:

$$\begin{aligned}
\lambda \mathbf{K}^{-\mathrm{H}}(n-1) &= (\lambda^{1/2}\mathbf{K}^{-\mathrm{H}/2}(n-1))(\lambda^{1/2}\mathbf{K}^{-1/2}(n-1));\\
\lambda \mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n) &= (\lambda^{1/2}\mathbf{u}(n))(\lambda^{1/2}\mathbf{u}^{\mathrm{H}}(n));\\
\lambda^{1/2}\hat{\mathbf{x}}^{\mathrm{H}}(n|\mathcal{Y}_{n-1})\mathbf{K}^{-\mathrm{H}}(n-1) &= (\hat{\mathbf{x}}^{\mathrm{H}}(n|\mathcal{Y}_{n-1})\mathbf{K}^{-\mathrm{H}/2}(n-1))(\lambda^{1/2}\mathbf{K}^{-1/2}(n-1));\\
\lambda^{1/2}y^*(n)\mathbf{u}^{\mathrm{H}}(n) &= (y^*(n))(\lambda^{1/2}\mathbf{u}(n)).
\end{aligned}$$

We may now identify four distinct factors as the block elements of the prearray, which are paired in the following manner:

- $\lambda^{1/2}\mathbf{K}^{-\mathrm{H}/2}(n-1)$ and $\lambda^{1/2}\mathbf{u}(n)$, which are of dimensions M by M and M by 1, respectively.
- $\hat{\mathbf{x}}^{\mathrm{H}}(n|\mathcal{Y}_{n-1})\mathbf{K}^{-\mathrm{H}/2}(n-1)$ and $y^*(n)$, which are of dimensions 1 by M and 1 by 1, respectively.

Insofar as the prearray is concerned, the first two factors are naturally compatible by virtue of being a matrix and a vector, respectively. The compatibility of the last pair of factors as row vectors is the reason for working with the Hermitian transposed forms of Eqs. (15.29) and (15.30). We may thus construct the following prearray:

$$\begin{bmatrix}
\lambda^{1/2}\mathbf{K}^{-\mathrm{H}/2}(n-1) & \lambda^{1/2}\mathbf{u}(n)\\
\hat{\mathbf{x}}^{\mathrm{H}}(n|\mathcal{Y}_{n-1})\mathbf{K}^{-\mathrm{H}/2}(n-1) & y^*(n)\\
\mathbf{0}^{\mathrm{T}} & 1
\end{bmatrix}.$$

The last row, made up of a block of M zeros followed by a unity term, has been added in order to make room for the generation of other filtering variables in the postarray (Morf & Kailath, 1975; Sayed & Kailath, 1994). Suppose next that we choose a unitary rotation $\mathbf{\Theta}(n)$ that transforms the prearray so as to produce a block zero in the second entry of the postarray's top block row, as shown by

$$\begin{bmatrix}
\lambda^{1/2}\mathbf{K}^{-\mathrm{H}/2}(n-1) & \lambda^{1/2}\mathbf{u}(n)\\
\hat{\mathbf{x}}^{\mathrm{H}}(n|\mathcal{Y}_{n-1})\mathbf{K}^{-\mathrm{H}/2}(n-1) & y^*(n)\\
\mathbf{0}^{\mathrm{T}} & 1
\end{bmatrix}\mathbf{\Theta}(n) = \begin{bmatrix}
\mathbf{B}_{11}^{\mathrm{H}}(n) & \mathbf{0}\\
\mathbf{b}_{21}^{\mathrm{H}}(n) & b_{22}^*(n)\\
\mathbf{b}_{31}^{\mathrm{H}}(n) & b_{32}^*(n)
\end{bmatrix}. \tag{15.32}$$

In effect, the vector $\lambda^{1/2}\mathbf{u}(n)$ in the prearray is annihilated by the transformation. By proceeding in a manner similar to that described for the square-root covariance filtering algorithm [i.e., by squaring both sides of Eq. (15.32) and then comparing respective

terms on both sides of the resulting equality], we may choose the block elements of the postarray as follows (see Problem 1):

$$\mathbf{B}_{11}^{\mathrm{H}}(n) = \mathbf{K}^{-\mathrm{H}/2}(n); \tag{15.33}$$

$$\mathbf{b}_{21}^{\mathrm{H}}(n) = \hat{\mathbf{x}}^{\mathrm{H}}(n+1\,|\,\mathcal{Y}_n)\mathbf{K}^{-\mathrm{H}/2}(n); \tag{15.34}$$

$$\mathbf{b}_{31}^{\mathrm{H}}(n) = \lambda^{1/2}\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}^{1/2}(n); \tag{15.35}$$

$$b_{22}^{*}(n) = r^{-1/2}(n)\alpha^{*}(n); \tag{15.36}$$

$$b_{32}^{*}(n) = r^{-1/2}(n). \tag{15.37}$$

Accordingly, we may rewrite Eq. (15.32) in the desired form

$$\begin{bmatrix} \lambda^{1/2}\mathbf{K}^{-\mathrm{H}/2}(n-1) & \lambda^{1/2}\mathbf{u}(n) \\ \hat{\mathbf{x}}^{\mathrm{H}}(n\,|\,\mathcal{Y}_{n-1})\mathbf{K}^{-\mathrm{H}/2}(n-1) & y^{*}(n) \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \boldsymbol{\Theta}(n) = \begin{bmatrix} \mathbf{K}^{-\mathrm{H}/2}(n) & \mathbf{0} \\ \hat{\mathbf{x}}^{\mathrm{H}}(n+1\,|\,\mathcal{Y}_n)\mathbf{K}^{-\mathrm{H}/2}(n) & r^{-1/2}(n)\alpha^{*}(n) \\ \lambda^{1/2}\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}^{1/2}(n) & r^{-1/2}(n) \end{bmatrix}. \tag{15.38}$$

The block elements of the postarray provide two sets of useful results:

1. *Updated block elements of the prearray*:
   - The updated square root $\mathbf{K}^{-\mathrm{H}/2}(n)$ is given by $\mathbf{B}_{11}^{\mathrm{H}}(n)$.
   - The updated matrix product $\hat{\mathbf{x}}^{\mathrm{H}}(n+1\,|\,\mathcal{Y}_n)\mathbf{K}^{-\mathrm{H}/2}(n)$ is given by $\mathbf{b}_{21}^{\mathrm{H}}(n)$.
2. *Two other filtering variables*:
   - The conversion factor, $r^{-1}(n)$, is obtained by squaring $b_{32}(n)$, which is real.
   - The innovation, $\alpha(n)$, is obtained by dividing $b_{22}(n)$ by $b_{32}(n)$.

The updated state estimate $\hat{\mathbf{x}}(n+1\,|\,\mathcal{Y}_n)$ is computed from the upper triangular system of Eq. (15.34), where $\mathbf{K}^{-\mathrm{H}/2}(n)$ is known by virtue of Eq. (15.33). Specifically, the individual elements of $\hat{\mathbf{x}}(n+1\,|\,\mathcal{Y}_n)$ are computed by using the *method of back substitution* that exploits the upper triangular structure of the square root $\mathbf{K}^{-\mathrm{H}/2}(n)$.

Part 2 of Table 15.1 presents a summary of the square-root information-filtering algorithm; initialization of the algorithm proceeds in the same way described in Table 14.5.

The square-root covariance filter and the square-root information filter summarized in Table 15.1 share a common feature: The number of operations (multiplications and additions) needed to proceed from one adaptation cycle of the algorithm to the next, in both cases, is $O(M^2)$, where O is order and M is the state dimension.

15.2 BUILDING SQUARE-ROOT ADAPTIVE FILTERS ON THE TWO KALMAN FILTER VARIANTS

The two square-root variants of the Kalman filter described in the previous section provide the general framework for the derivation of known square-root adaptive filtering algorithms for exponentially weighted RLS estimation. We say so in light of the one-to-one correspondences that exist between the Kalman variables and RLS variables, as demonstrated in Chapter 14. (See Table 14.3.)

Two important square-root adaptive filtering algorithms for RLS estimation are known as the *QR-decomposition-based RLS (QRD-RLS) algorithm* and *inverse*

QRD-RLS algorithm. The reason for this terminology is that derivation of these algorithms has traditionally relied, in one form or another, on the use of an orthogonal triangularization process known in matrix algebra as *QR-decomposition*. The motivation for using QR-decomposition in adaptive filtering is to exploit its good numerical properties.

The QR-decomposition of a matrix $\mathbf{A}(n)$ may be written as (Stewart, 1973; Golub & Van Loan, 1996)

$$\mathbf{Q}(n)\mathbf{A}(n) = \begin{bmatrix} \mathbf{R}(n) \\ \mathbf{O} \end{bmatrix}, \tag{15.39}$$

where $\mathbf{Q}(n)$ is a unitary matrix, $\mathbf{R}(n)$ is an upper triangular matrix, and $\mathbf{O}$ is the null matrix. The pervasive use of the symbols $\mathbf{Q}$ and $\mathbf{R}$ in such a transformation has led, in the course of time, to the common use of "QR-decomposition." By the same token, adaptive RLS filtering algorithms based on QR-decomposition, in a broad sense, became known as "QRD-RLS algorithms." Traditionally, the QRD-RLS algorithms for exponentially weighted RLS estimation were derived starting from the prewindowed version of a data matrix, which was then triangularized by applying the QR-decomposition. In light of the material presented in Section 15.2, these adaptive filtering algorithms can indeed be deduced directly from the two square-root variants of the Kalman filter, thereby achieving two highly desirable objectives:

1. The unified treatment of QRD-RLS adaptive filtering algorithms for exponentially weighted RLS estimation.
2. Consolidating the bridge between the deterministic RLS estimation theory and the stochastic Kalman filter theory.

In the remainder of this chapter, we follow this insightful approach in deriving different QRD-RLS adaptive filtering algorithms. However, the order in which these algorithms are considered follows the traditional development of RLS estimation theory rather than the order of square-root Kalman filters summarized in Table 15.1.

15.3 QRD-RLS ALGORITHM

The *QRD-RLS algorithm*, or, more precisely, the *QR-decomposition-based RLS algorithm*, derives its name from the fact that the computation of the least-squares weight vector in a finite-duration impulse response (FIR) filter implementation of the adaptive filtering algorithm is accomplished by working directly with the incoming data matrix via the QR-decomposition rather than working with the (time-average) correlation matrix of the input data, as in the traditional RLS algorithm (Gentleman & Kung, 1981; McWhirter, 1983; Haykin, 1991). Accordingly, the QRD-RLS algorithm is numerically more stable than the traditional RLS algorithm.

Assuming the use of prewindowing on the input data, the *data matrix* is defined by

$$\begin{aligned} \mathbf{A}^{\mathrm{H}}(n) &= [\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(M), \ldots, \mathbf{u}(n)] \\ &= \begin{bmatrix} u(1) & u(2) & \cdots & u(M) & \cdots & u(n) \\ 0 & u(1) & \cdots & u(M-1) & \cdots & u(n-1) \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & u(1) & \cdots & u(n-M+1) \end{bmatrix}, \end{aligned} \tag{15.40}$$

where M is the number of FIR filter coefficients (i.e., filter order). Correspondingly, the correlation matrix of the input data is defined by

$$\begin{aligned}\boldsymbol{\Phi}(n) &= \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)\mathbf{u}^{\mathrm{H}}(i) \\ &= \mathbf{A}^{\mathrm{H}}(n)\boldsymbol{\Lambda}(n)\mathbf{A}(n).\end{aligned} \tag{15.41}$$

The matrix

$$\boldsymbol{\Lambda}(n) = \operatorname{diag}[\lambda^{n-1}, \lambda^{n-2}, \ldots, 1] \tag{15.42}$$

is called the *exponential weighting matrix*; λ is the *exponential weighting factor*. Equation (15.41) represents a generalization of Eq. (9.45) used in the method of least squares.

From Chapter 10, we recall that the matrix $\mathbf{P}(n)$, used in deriving the RLS algorithm, is defined as the inverse of the time-average correlation matrix $\boldsymbol{\Phi}(n)$, as shown by [see Eq. (10.17)]

$$\mathbf{P}(n) = \boldsymbol{\Phi}^{-1}(n). \tag{15.43}$$

From Table 14.3 of Chapter 14, we also note the following correspondences between the Kalman variables and RLS variables:

Kalman Variable	RLS Variable	Description
$\mathbf{K}^{-1}(n)$	$\lambda\mathbf{P}^{-1}(n) = \lambda\boldsymbol{\Phi}(n)$	Correlation matrix
$r^{-1}(n)$	$\gamma(n)$	Conversion factor
$\mathbf{g}(n)$	$\lambda^{-1/2}\mathbf{k}(n)$	Gain vector
$\alpha(n)$	$\lambda^{-n/2}\xi^{*}(n)$	A priori estimation error
$y(n)$	$\lambda^{-n/2}d^{*}(n)$	Desired response (observable)
$\hat{\mathbf{x}}(n\,\vert\,\mathcal{Y}_{n-1})$	$\lambda^{-n/2}\hat{\mathbf{w}}(n-1)$	Estimate of tap-weight vector

Before proceeding to formulate the QRD-RLS algorithm in light of these correspondences, we find it convenient to make a change of notation. According to the normal equations, the least-squares estimate of the tap-weight vector $\hat{\mathbf{w}}(n)$ is defined by [see Eq. (9.35)]

$$\boldsymbol{\Phi}(n)\hat{\mathbf{w}}(n) = \mathbf{z}(n), \tag{15.44}$$

where $\mathbf{z}(n)$ is the time-average cross-correlation vector between the desired response $d(n)$ and input data vector $\mathbf{u}(n)$. Let $\boldsymbol{\Phi}(n)$ be expressed in its factored form:

$$\boldsymbol{\Phi}(n) = \boldsymbol{\Phi}^{1/2}(n)\boldsymbol{\Phi}^{\mathrm{H}/2}(n). \tag{15.45}$$

Then, premultiplying both sides of Eq. (15.44) by the inverse square root $\boldsymbol{\Phi}^{-1/2}(n)$, we may introduce a new vector variable defined by

$$\mathbf{p}(n) = \boldsymbol{\Phi}^{\mathrm{H}/2}(n)\hat{\mathbf{w}}(n) = \boldsymbol{\Phi}^{-1/2}(n)\mathbf{z}(n). \tag{15.46}$$

Thus, by propagating $\boldsymbol{\Phi}^{1/2}(n)$ and $\mathbf{p}(n)$, we may view the QRD-RLS algorithm as the square-root information-filtering algorithm in Kalman filter theory, Chapter 14.

We are now ready to formulate the QRD-RLS algorithm for linear adaptive filtering. Specifically, we may translate Eq. (15.38) pertaining to the square-root information-filtering

algorithm into the corresponding prearray-to-postarray transformation for the QRD-RLS algorithm as follows (after the cancellation of common terms):

$$\begin{bmatrix} \lambda^{1/2}\boldsymbol{\Phi}^{1/2}(n-1) & \mathbf{u}(n) \\ \lambda^{1/2}\mathbf{p}^{\mathrm{H}}(n-1) & d(n) \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \boldsymbol{\Theta}(n) = \begin{bmatrix} \boldsymbol{\Phi}^{1/2}(n) & \mathbf{0} \\ \mathbf{p}^{\mathrm{H}}(n) & \xi(n)\gamma^{1/2}(n) \\ \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\Phi}^{-\mathrm{H}/2}(n) & \gamma^{1/2}(n) \end{bmatrix}. \tag{15.47}$$

Basically, $\boldsymbol{\Theta}(n)$ is any unitary rotation that operates on the elements of the input data vector $\mathbf{u}(n)$ in the prearray, *annihilating* them one by one so as to produce a block zero entry in the top block row of the postarray. Naturally, the lower triangular structure of the square root of the correlation matrix, namely, $\boldsymbol{\Phi}^{1/2}$, is preserved in its exact form before and after the transformation. This is indeed the very essence of the QR-decomposition for RLS estimation—hence the name "QRD-RLS algorithm."

Having computed the updated block values $\boldsymbol{\Phi}^{1/2}(n)$ and $\mathbf{p}^{\mathrm{H}}(n)$, we may then solve for the least-squares weight vector $\hat{\mathbf{w}}(n)$ by using the formula [see Eq. (15.46)]

$$\hat{\mathbf{w}}^{\mathrm{H}}(n) = \mathbf{p}^{\mathrm{H}}(n)\boldsymbol{\Phi}^{-1/2}(n). \tag{15.48}$$

The computation of this solution is accomplished with the *method of back substitution* that exploits the lower triangular structure of $\boldsymbol{\Phi}^{1/2}(n)$. Note, however, that this computation is feasible only for adaptation cycle $n > M$, for which the data matrix $\mathbf{A}(n)$, and therefore $\boldsymbol{\Phi}^{1/2}(n)$, is of full column rank and therefore nonsingular.

To initialize the QRD-RLS algorithm, we may set $\boldsymbol{\Phi}^{1/2}(0) = \delta^{1/2}\mathbf{I}$ and $\mathbf{p}(0) = \mathbf{0}$, where δ is the regularization parameter. The *exact initialization* of the QRD-RLS algorithm occupies the period $0 \le n \le M$ for which the a posteriori estimation error $e(n)$ is zero. At adaptation cycle $n = M$, the initialization is completed, whereafter $e(n)$ may assume a nonzero value.

A summary of the QRD-RLS algorithm is presented in Table 15.2, including details of the initialization and other matters of interest.

TABLE 15.2 Summary of the QRD-RLS Algorithm for Exponentially Weighted RLS Estimation

Inputs:
- data matrix: $\mathbf{A}^{\mathrm{H}}(n) = [\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(n)]$
- desired response: $\mathbf{d}^{\mathrm{H}}(n) = [d(1)\ d(2), \ldots, d(n)]$

Prescribed parameters:
- exponential weighting factor = λ
- regularization parameter = δ
- unitary rotation = $\boldsymbol{\Theta}(n)$

Initial conditions:
- $\boldsymbol{\Phi}^{1/2}(0) = \delta^{1/2}\mathbf{I}$
- $\mathbf{p}(0) = \mathbf{0}$

Computation:
For $n = 1, 2, \ldots,$ compute

$$\begin{bmatrix} \lambda^{1/2}\boldsymbol{\Phi}^{1/2}(n-1) & \mathbf{u}(n) \\ \lambda^{1/2}\mathbf{p}^{\mathrm{H}}(n-1) & d(n) \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \boldsymbol{\Theta}(n) = \begin{bmatrix} \boldsymbol{\Phi}^{1/2}(n) & \mathbf{0} \\ \mathbf{p}^{\mathrm{H}}(n) & \xi(n)\gamma^{1/2}(n) \\ \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\Phi}^{-\mathrm{H}/2}(n) & \gamma^{1/2}(n) \end{bmatrix}$$

$$\hat{\mathbf{w}}^{\mathrm{H}}(n) = \mathbf{p}^{\mathrm{H}}(n)\boldsymbol{\Phi}^{-1/2}(n)$$

Note: $\boldsymbol{\Theta}(n)$ is a unitary rotation that operates on the prearray to produce a block zero entry in the top block row of the postarray.

Systolic Array Implementation of the QRD-RLS Algorithm

Thus far, we have not focused on the particulars of the unitary rotation $\boldsymbol{\Theta}(n)$, other than to require that it be chosen to produce a block zero entry in the top block row of the postarray. A unitary matrix that befits this requirement is the transformation based on the *Givens rotation*, detailed discussion of which is presented in Appendix G.

The use of Givens rotations lends itself to a *parallel implementation* in the form of a *systolic array* (Kung & Leiserson, 1978).[1] A systolic array consists of an array of individual *processing cells* arranged as a regular structure. Each cell in the array is provided with local memory of its own and is connected only to its nearest neighbors. The array is designed such that regular streams of data are clocked through it in a highly rhythmic fashion, much like the pumping action of the human heart—hence the name "systolic" (Kung, 1982). The important point to note here is that systolic arrays are well suited for implementing complex signal-processing algorithms such as the QRD-RLS algorithm, particularly when the requirement is to operate in *real time* and at *high data bandwidths*.

Figure 15.1 shows an efficient systolic array structure for implementing the QRD-RLS algorithm (McWhirter, 1983); the structure applies to the example of a weight vector $\hat{\mathbf{w}}(n)$ with three elements (i.e., $M = 3$). The $(M+1)$-by-$(M+1)$ unitary matrix $\boldsymbol{\Theta}$ in Eq. (15.47) is implemented as a sequence of M Givens rotations, each of which is configured to annihilate a particular element of the M-by-1 vector $\mathbf{u}(n)$ in the prearray. We may thus write

$$\boldsymbol{\Theta} = \prod_{k=1}^{M} \boldsymbol{\Theta}_k, \tag{15.49}$$

[1]The first systolic implementation of the QRD-RLS algorithm was published by Gentleman and Kung (1981), who used a systolic array structure consisting of two distinct sections: a *triangular systolic array* and a *linear systolic array*. The Gentleman–Kung array differs from the McWhirter array of Fig. 15.1 in the following respects:

- The transformation of Eq. (15.47), constituting the mathematical basis of the QRD-RLS algorithm, is implemented in two stages. In stage 1, the last rows of the prearray and postarray are deleted, with a corresponding reduction in the dimensions of the unitary matrix $\boldsymbol{\Theta}(n)$; thus, we now write

$$\begin{bmatrix} \lambda^{1/2}\boldsymbol{\Phi}^{1/2}(n-1) & \mathbf{u}(n) \\ \lambda^{1/2}\mathbf{p}^{\mathrm{H}}(n-1) & d(n) \end{bmatrix} \boldsymbol{\Theta}(n) = \begin{bmatrix} \boldsymbol{\Phi}^{1/2}(n) & \mathbf{0} \\ \mathbf{p}^{\mathrm{H}}(n) & \xi(n)\gamma^{1/2}(n) \end{bmatrix},$$

  which is implemented by the triangular section of the Gentleman–Kung array. In stage 1 of the computation, the updated lower triangular matrix $\boldsymbol{\Phi}^{1/2}(n)$ and row vector $\mathbf{p}^{\mathrm{H}}(n)$ are produced. Once the entire orthogonal triangularization process is completed, the data flow stops, and the stored values of $\boldsymbol{\Phi}^{1/2}(n)$ and $\mathbf{p}^{\mathrm{H}}(n)$ are clocked out for stage 2 of the computation to proceed by the linear systolic section. In particular, a form of backward substitution is used by that section to compute the filter weights $\hat{w}_{M-1}(n), \ldots, \hat{w}_2(n), \hat{w}_1(n)$. In contrast, the McWhirter array consists of a single triangular array designed to compute the a posteriori estimation error $e(n)$.
- In structural terms, the Gentleman–Kung array uses internal and boundary cells, whereas the McWhirter array uses an additional cell, namely, the final processing cell to compute the a posteriori estimation error.

Practically speaking, we may say the following: If the requirement is to compute the a posteriori estimation error and the least-squares filter weights are needed only on an occasional basis, then the recommended systolic procedure is to use the McWhirter array. If, on the other hand, obtaining the least-squares filter weights is the primary requirement, then the systolic version of the inverse QRD-RLS algorithm described in Section 15.5 is the preferred method over the Gentleman–Kung array.

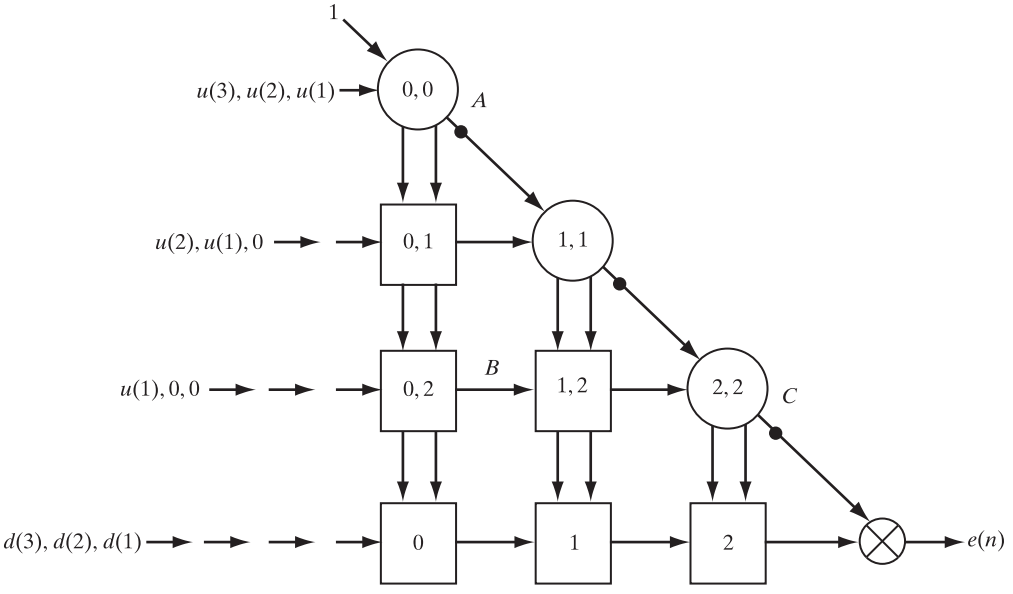


FIGURE 15.1 Systolic array implementation of the QRD-RLS algorithm. The dots along the diagonal of the array represent storage elements. The processing delay, which is a consequence of the temporal skew imposed on the input data, may be incorporated within the associated boundary cells.

where $\mathbf{\Theta}_k$ consists of a unitary matrix except for four strategic elements located at the points where the pair of rows k and $M+1$ intersects the pair of columns k and $M+1$. These four elements, denoted by θ_{kk}, $\theta_{M+1,\,k}$, $\theta_{k,\,M+1}$, and $\theta_{M+1,\,M+1}$, are defined as

$$\left.\begin{aligned} \theta_{kk} &= \theta_{M+1,\,M+1} = c_k \\ \theta_{M+1,\,k} &= s_k^* \\ \theta_{k,\,M+1} &= -s_k \end{aligned}\right\}, \tag{15.50}$$

where $k = 1, 2, \ldots, M$. The *cosine parameter* c_k is real, whereas the *sine parameter* s_k is complex. The choice of c_k and s_k is constrained by the relation

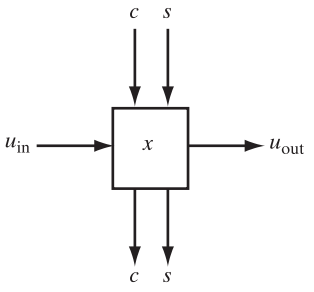
$$c_k^2 + |s_k|^2 = 1 \qquad \text{for all } k. \tag{15.51}$$

A transformation of the form defined in Eq. (15.50) is called the *Givens rotation*.

The systolic structure of Fig. 15.1 is configured with two points in mind. First, data flow through the structure from left to right, consistent with all other adaptive filters considered in previous chapters. Second, the systolic array operates directly on the input data that are represented by successive values of the input signal vector $\mathbf{u}(n)$ and the desired response $d(n)$.

The systolic array is controlled by a single *clock* and consists of three types of processing cells arranged in the form of a triangular section:

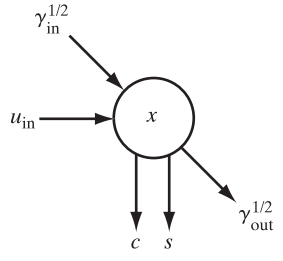
1. *Internal cells*, which are depicted as squares. The internal cells perform only additions and multiplications, as described in Fig. 15.2(a).



$$u_{out} \leftarrow cu_{in} - s^* \lambda^{1/2} x$$
$$x \leftarrow su_{in} + c\lambda^{1/2} x$$

Initialization:
At $n = 0$, set
$x = 0$
$c = 1$
$s = 0$

(a)



If $u_{in} = 0$, then

$c \leftarrow 1$

$s \leftarrow 0$, and

$\gamma_{out}^{1/2} \leftarrow \gamma_{in}^{1/2}$

$x \leftarrow \lambda^{1/2} x$

Otherwise,

$$x' \leftarrow \sqrt{\lambda x^2 + |u_{in}|^2}$$

$$c \leftarrow \frac{\lambda^{1/2} x}{x'}$$

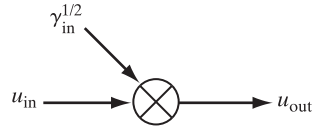
$$s \leftarrow \frac{u_{in}}{x'}$$

$x \leftarrow x'$

$$\gamma_{out}^{1/2} \leftarrow c\gamma_{in}^{1/2}$$

Initialization:
At $n = 0$, set
$x = 0$
$c = 1$
$s = 0$
$\gamma_{in} = 1$

(b)



$$u_{out} \leftarrow \gamma_{in}^{1/2} u_{in}$$

(c)

FIGURE 15.2 Cells for systolic array of Fig. 15.1: (a) internal cell; (b) boundary cell; (c) final cell. *Note*: The stored value x is initialized to be zero (i.e., real). For the boundary cell, it always remains real. Hence, the formulas for the rotation parameters c and s computed by the boundary cell can be simplified considerably, as shown in part (a). Note also that in parts (a) and (b), the values x stored in the array are elements of the lower triangular matrix $\mathbf{R}^H$; hence, $r^* = x$ for all elements of the array.

2. *Boundary cells*, which are depicted as large circles. The boundary cells are considerably more complex than the internal cells, in that they compute square roots and reciprocals, as described in Fig. 15.2(b).
3. *Final processing cell*, which is depicted by a small circle. The final cell produces an output simply by multiplying its two inputs, as described in Fig. 15.2(c).

Each processing cell of the triangular section excited by elements of the input vector $\mathbf{u}(n)$ stores a particular element of the lower triangular matrix $\mathbf{\Phi}^{1/2}(n)$, depending on the location of the cell under consideration. The function of each column of processing cells in the triangular section is to *rotate* one column of the stored triangular matrix with a vector of data received from the left in such a way that the leading element of the received input vector $\mathbf{u}(n)$ is *annihilated*. The reduced data vector is then passed to the right on the next column of processing cells. The boundary cell in each column of the triangular section computes the pertinent rotation parameters and then passes them downward on the next clock cycle. The internal cells subsequently apply the same rotation to all other elements of the input vector $\mathbf{u}(n)$. Since a delay of one clock cycle per cycle is incurred in passing the rotation parameters downward along a column, it is necessary that the input vector $\mathbf{u}(n)$ enter the array in a *skewed order*, as illustrated in Fig. 15.1 for the case of $M = 3$; likewise, for the desired response $d(n)$. This arrangement of the input data ensures that, as each column vector $\mathbf{u}(n)$ of the data matrix $\mathbf{A}^{\mathrm{H}}(n)$ propagates through the array, it interacts with the previously stored triangular matrix $\mathbf{\Phi}^{1/2}(n-1)$ and thereby undergoes the sequence of Givens rotations denoted by $\mathbf{\Theta}(n)$, as required. Accordingly, all the elements of the column vector $\mathbf{u}(n)$ are annihilated, one by one, and an updated lower triangular matrix $\mathbf{\Phi}^{1/2}(n)$ is produced and stored in the process, ready for the next sequence of operations.

The systolic array operates in a highly pipelined manner, whereby, as (time-skewed) input data vectors enter the array from the left, we find that, in effect, each such vector defines a processing wave front that moves across the array. Consequently, on any particular clock cycle, elements of the pertinent lower triangular matrix $\mathbf{\Phi}^{1/2}(n)$ exist only along the pertinent wave front.

As the orthogonal triangularization is being performed by the systolic section labeled *ABC* in Fig. 15.1, the row vector $\mathbf{p}^{\mathrm{H}}(n)$ is computed by the appended bottom row of internal cells, the last one of which produces the output $\xi(n)\gamma^{1/2}(n)$. (See Problem 7.)

From Fig. 15.2(b), we note that the kth boundary cell in the systolic array structure of Fig. 15.1 performs the computation

$$\gamma_{\text{out},\,k}^{1/2} = c_k(n)\gamma_{\text{in},\,k}^{1/2}(n), \quad k = 1, 2, \ldots, M, \tag{15.52}$$

where $c_k(n)$ is the cosine parameter of that cell. Accordingly, with a set of M boundary cells connected together as in Fig. 15.1, the output of the last boundary cell produced in response to a unit input applied to the first boundary cell may be expressed as

$$\begin{aligned}\gamma^{1/2}(n) &= \gamma_{\text{out},\,M}^{1/2}(n)\Big|_{\gamma_{\text{in},1}^{1/2}(n)=1} \\ &= \prod_{k=1}^{M} c_k(n).\end{aligned} \tag{15.53}$$

With inputs equal to $\gamma^{1/2}(n)$ and $\xi(n)\gamma^{1/2}(n)$, the final processing cell in Fig. 15.1 produces an output equal to the a posteriori estimation error $e(n)$, in accordance with the relation [see the first line of Eq. (10.42) of Chapter 10]

$$\begin{aligned}e(n) &= \xi(n)\gamma(n) \\ &= (\xi(n)\gamma^{1/2}(n))(\gamma^{1/2}(n)).\end{aligned} \tag{15.54}$$

As the time-skewed input data vectors enter the systolic array of the figure, we find that updated estimation errors are produced at the output of the array at the rate of one every clock cycle. The estimation error produced on a given clock cycle corresponds, of course, to the particular element of the desired response vector $d(n)$ that entered the array M clock cycles previously.

It is noteworthy that the a priori estimation error $\xi(n)$ may be obtained by *dividing* the output that emerges from the last cell in the appended (bottom) row of internal cells by the output from the last boundary cell. Also, the conversion factor $\gamma(n)$ may be obtained simply by squaring the output that emerges from the last boundary cell.

Figure 15.3 summarizes, in a diagrammatic fashion, the flow of signals in the systolic array of Fig. 15.1. To simplify the presentation, we have used $\mathbf{R}^{\mathrm{H}}(n)$ in place of $\mathbf{\Phi}^{1/2}(n)$ in Fig. 15.3. The figure includes the external inputs $\mathbf{u}(n)$ and $d(n)$, the resulting transformations in the internal states of the triangular section and appended row of internal cells, the respective outputs of these two sections, and the overall output of the complete processor.

A distinctive feature of the systolic structure shown in Fig. 15.1 is that computation of the a posteriori estimation error bypasses the need for computing the weight vector $\hat{\mathbf{w}}(n)$. However, if the weight vector is required, then it can be computed by a *serial weight flushing* that is performed nonsystolically (Ward et al., 1986; Shepherd & McWhirter, 1993). To explain the method, let $\mathbf{u}(n)$ denote the input vector and $d(n)$ denote the desired response, both at adaptation cycle n. Given that the weight vector at this adaptation cycle is $\hat{\mathbf{w}}(n)$, the corresponding a posteriori estimation error is

$$e(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n). \tag{15.55}$$

Suppose that the state of the array is *frozen* at adaptation cycle n_+, immediately *after* the systolic computation at adaptation cycle n is completed. That is, suppose that any update of stored values in the array is suppressed, but the processor is permitted to function normally in all other respects. At adaptation cycle n_+, we also set the desired response $d(n)$ equal to zero. We now define an input vector that consists of a string of zeros, except for the ith element, which is set equal to unity, as shown by

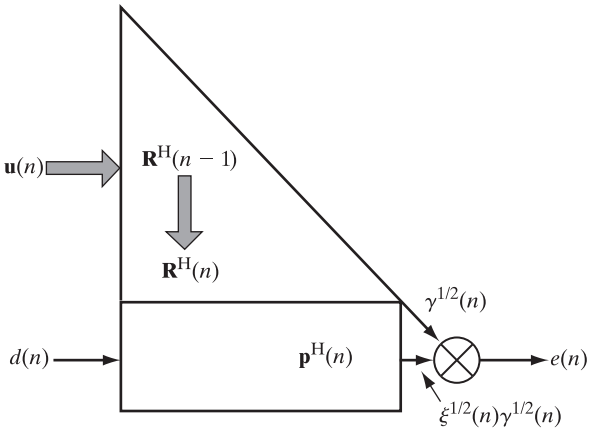


FIGURE 15.3 Diagrammatic representation of the flow of signals in the systolic array of Fig. 15.1.

$$\mathbf{u}^{\mathrm{H}}(n_+) = [0 \ldots 010 \ldots 0]. \quad \begin{matrix} \uparrow \\ i\text{th element} \end{matrix} \tag{15.56}$$

Then, setting $d(n) = 0$ and substituting Eq. (15.56) into Eq. (15.55), we get

$$e(n_+) = -\hat{w}_i^*(n_+). \tag{15.57}$$

In other words, except for a trivial sign change, we may compute the ith element of the M-by-1 weight vector $\mathbf{w}^{\mathrm{H}}(n)$ by freezing the state of the processor at adaptation cycle n, setting the desired response equal to zero, and feeding the processor with an input vector in which the ith element is unity and the remaining $M-1$ elements are all zero. The essence of all of this is that the Hermitian-transposed weight vector $\hat{\mathbf{w}}^{\mathrm{H}}(n)$ may be viewed as the *impulse response* of the *nonadaptive* (i.e., frozen) form of the systolic array processor, in the sense that it can be generated as the system output produced by inputting an $(M-1)$-by-$(M-1)$ identity matrix to the main triangular array and a zero vector to the bottom row of the array in Fig. 15.1 (Shepherd & McWhirter, 1993). To "flush" the entire M-by-1 weight vector $\hat{\mathbf{w}}^{\mathrm{H}}(n)$ out of the systolic processor, the procedure is therefore simply to halt the update of all stored values and input a data matrix that consists of a unit diagonal matrix (i.e., an identity matrix) of dimension M.

15.4 ADAPTIVE BEAMFORMING

From the discussions of adaptive beamforming presented in previous chapters, we recall that the objective of this spatial form of adaptive filtering is to modify the individual outputs of an array of sensors so as to produce an overall far-field pattern that optimizes, in some statistical sense, the reception of a target signal along a direction of interest. As with any adaptive filter, such an optimization is achieved by suitable modifications of a set of weights built into the construction of the array. However, unlike other adaptive filtering applications, adaptive beamforming does *not* require explicit knowledge of the weights. This suggests a possible area of application for the QRD-RLS algorithm implemented in the form of a systolic array, particularly the structure described in Fig. 15.1.

In this section, we revisit the *minimum variance distortionless response (MVDR) beamformer*, previously discussed in Chapters 2, 6, and 9. The key question, of course, is how to formulate the QRD-RLS algorithm, and therefore the triangular systolic array of Fig. 15.1, so as to perform the MVDR beamforming task.

The MVDR Problem

Consider a linear array of M uniformly spaced sensors whose outputs are individually weighted and then summed to produce the beamformer output

$$e(i) = \sum_{l=1}^{M} w_l^*(n) u_l(i), \tag{15.58}$$

where $u_l(i)$ is the output of sensor l at adaptation cycle i and $w_l(n)$ is the associated (complex) weight. To simplify the mathematical presentation, we consider the simple case of a single look direction. Let $s_1(\theta), s_2(\theta), \ldots, s_M(\theta)$ be the elements of a prescribed *steering vector* $\mathbf{s}(\theta)$; the *electrical angle* θ is determined by the look direction of interest. In particular, the element $s_l(\theta)$ is the output of sensor l of the array under the condition that there is no signal other than that due to a source of interest. We may thus state the MVDR problem as follows:

Minimize the cost function

$$\mathscr{E}(n) = \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2 \tag{15.59}$$

subject to the constraint

$$\sum_{l=1}^{M} w_l^*(n) s_l(\theta) = 1 \qquad \text{for all } n. \tag{15.60}$$

Using matrix notation, we may redefine the cost function of Eq. (15.59) as

$$\mathscr{E}(n) = \boldsymbol{\varepsilon}^{\mathrm{H}}(n) \boldsymbol{\Lambda}(n) \boldsymbol{\varepsilon}(n), \tag{15.61}$$

where $\boldsymbol{\Lambda}(n)$ is the exponential weighting matrix and $\boldsymbol{\varepsilon}(n)$ is the vector of constrained beamformer outputs. According to Eq. (15.58), the beamformer output vector $\boldsymbol{\varepsilon}(n)$ is related to the data matrix $\mathbf{A}(n)$ by

$$\begin{aligned} \boldsymbol{\varepsilon}(n) &= [e(1), e(2), \ldots, e(n)]^{\mathrm{H}} \\ &= \mathbf{A}(n)\mathbf{w}(n), \end{aligned} \tag{15.62}$$

where $\mathbf{w}(n)$ is the weight vector and $\mathbf{A}(n)$ is defined in terms of the *snapshots* $\mathbf{u}(1)$, $\mathbf{u}(2), \ldots, \mathbf{u}(n)$ by

$$\begin{aligned} \mathbf{A}^{\mathrm{H}}(n) &= [\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(n)] \\ &= \begin{bmatrix} u_1(1) & u_1(2) & \cdots & u_1(n) \\ u_2(1) & u_2(2) & \cdots & u_2(n) \\ \vdots & \vdots & & \vdots \\ u_M(1) & u_M(2) & \cdots & u_M(n) \end{bmatrix}. \end{aligned} \tag{15.63}$$

We may now restate the MVDR problem in matrix terms as follows:

Given the data matrix $\mathbf{A}(n)$ and the exponential weighting matrix $\boldsymbol{\Lambda}(n)$, minimize the cost function

$$\mathscr{E}(n) = \|\boldsymbol{\Lambda}^{1/2}(n)\mathbf{A}(n)\mathbf{w}(n)\|^2 \tag{15.64}$$

with respect to the weight vector $\mathbf{w}(n)$, subject to the constraint

$$\mathbf{w}^{\mathrm{H}}(n)\mathbf{s}(\theta) = 1 \qquad \text{for all } n,$$

where $\mathbf{s}(\theta)$ is the steering vector for a prescribed electrical angle θ.

The solution to this constrained optimization problem is described by the MVDR formula [see Eq. (9.91)]

$$\hat{\mathbf{w}}(n) = \frac{\mathbf{\Phi}^{-1}(n)\mathbf{s}(\theta)}{\mathbf{s}^{\mathrm{H}}(\theta)\mathbf{\Phi}^{-1}(n)\mathbf{s}(\theta)}, \tag{15.65}$$

where

$$\mathbf{\Phi}(n) = \mathbf{A}^{\mathrm{H}}(n)\mathbf{\Lambda}(n)\mathbf{A}(n) \tag{15.66}$$

is the M-by-M correlation matrix of the exponentially weighted sensor outputs averaged over n snapshots.

Systolic MVDR Beamformer

Let the correlation matrix $\mathbf{\Phi}(n)$ be expressed in its factored form:

$$\mathbf{\Phi}(n) = \mathbf{\Phi}^{1/2}(n)\mathbf{\Phi}^{\mathrm{H}/2}(n). \tag{15.67}$$

Correspondingly, we may rewrite Eq. (15.65) as

$$\hat{\mathbf{w}}(n) = \frac{\mathbf{\Phi}^{-\mathrm{H}/2}(n)\mathbf{\Phi}^{-1/2}(n)\mathbf{s}(\theta)}{\mathbf{s}^{\mathrm{H}}(\theta)\mathbf{\Phi}^{-\mathrm{H}/2}(n)\mathbf{\Phi}^{-1/2}(n)\mathbf{s}(\theta)}. \tag{15.68}$$

To simplify matters, we define the auxiliary vector:

$$\mathbf{a}(n) = \mathbf{\Phi}^{-1/2}(n)\mathbf{s}(\theta). \tag{15.69}$$

We now note that the denominator of Eq. (15.68) is a real-valued scalar equal to the squared Euclidean norm of the auxiliary vector $\mathbf{a}(n)$. The numerator is equal to the Hermitian-transposed inverse square root $\mathbf{\Phi}^{-\mathrm{H}/2}(n)$ postmultiplied by the auxiliary vector $\mathbf{a}(n)$. We may thus simplify Eq. (15.68) to

$$\hat{\mathbf{w}}(n) = \frac{\mathbf{\Phi}^{-\mathrm{H}/2}(n)\mathbf{a}(n)}{\|\mathbf{a}(n)\|^2}. \tag{15.70}$$

The MVDR beamformer output, or, in adaptive filtering terminology, the a posteriori estimation error produced at adaptation cycle n in response to the snapshot $\mathbf{u}(n)$, is given by

$$\begin{aligned} e(n) &= \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n) \\ &= \frac{\mathbf{a}^{\mathrm{H}}(n)\mathbf{\Phi}^{-1/2}(n)\mathbf{u}(n)}{\|\mathbf{a}(n)\|^2}. \end{aligned} \tag{15.71}$$

Let

$$e'(n) = \mathbf{a}^{\mathrm{H}}(n)\mathbf{\Phi}^{-1/2}(n)\mathbf{u}(n) \tag{15.72}$$

denote a new estimation error. We may then reduce Eq. (15.71) to

$$e(n) = \frac{e'(n)}{\|\mathbf{a}(n)\|^2}. \tag{15.73}$$

This equation shows that the MVDR beamformer output $e(n)$ is uniquely defined by two quantities: $e'(n)$ and $\mathbf{a}(n)$.

At this point in the discussion, we find it informative to recall the formula for the a posteriori estimation error $e(n)$ actually computed by the QRD-RLS algorithm. By definition,

$$e(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n), \tag{15.74}$$

where $d(n)$ is the desired response, $\hat{\mathbf{w}}(n)$ is the least-squares weight vector, and $\mathbf{u}(n)$ is the input data vector. Substituting Eq. (15.48) into Eq. (15.74) yields

$$e(n) = d(n) - \mathbf{p}^{\mathrm{H}}(n)\boldsymbol{\Phi}^{-1/2}(n)\mathbf{u}(n). \tag{15.75}$$

Thus, comparing Eqs. (15.72) and (15.75), we readily deduce the correspondences between the QRD-RLS adaptive filtering and MVDR beamforming variables listed in Table 15.3. The correspondences listed herein merely indicate the "similarity of roles," and not equivalence, between the variables listed under MVDR beamforming and those listed under QRD-RLS adaptive filtering.

TABLE 15.3 Correspondences between the QRD-RLS Adaptive Filtering and MVDR Beamforming Variables

QRD-RLS adaptive filtering	MVDR beamforming	Description
$e(n)$	$-e'(n)$	Estimation error
$d(n)$	0	Desired response
$\mathbf{p}(n)$	$\mathbf{a}(n)$	Auxiliary vector
$\mathbf{u}(n)$	$\mathbf{u}(n)$	Snapshot

The stage is now set for a recasting of the QRD-RLS algorithm to suit the MVDR beamforming problem. First, we apply the correspondences of Table 15.3 to the prearray in Eq. (15.47) for the QRD-RLS algorithm, thereby formulating the prearray for the MVDR beamformer as

$$\begin{bmatrix} \lambda^{1/2}\boldsymbol{\Phi}^{1/2}(n-1) & \mathbf{u}(n) \\ \lambda^{1/2}\mathbf{a}^{\mathrm{H}}(n-1) & 0 \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}.$$

Next, we determine the postarray that goes with this prearray by proceeding in the same manner as that described in Section 15.3. We may thus write

$$\begin{bmatrix} \lambda^{1/2}\boldsymbol{\Phi}^{1/2}(n-1) & \mathbf{u}(n) \\ \lambda^{1/2}\mathbf{a}^{\mathrm{H}}(n-1) & 0 \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \boldsymbol{\Theta}(n) = \begin{bmatrix} \boldsymbol{\Phi}^{1/2}(n) & \mathbf{0} \\ \mathbf{a}^{\mathrm{H}}(n) & -e'(n)\gamma^{-1/2}(n) \\ \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\Phi}^{-\mathrm{H}/2}(n) & \gamma^{1/2}(n) \end{bmatrix}. \tag{15.76}$$

We now see that the two quantities of interest in the MVDR problem may be obtained from the postarray of Eq. (15.76) as follows:

- The updated auxiliary vector $\mathbf{a}(n)$ is read directly from the second row of the postarray.

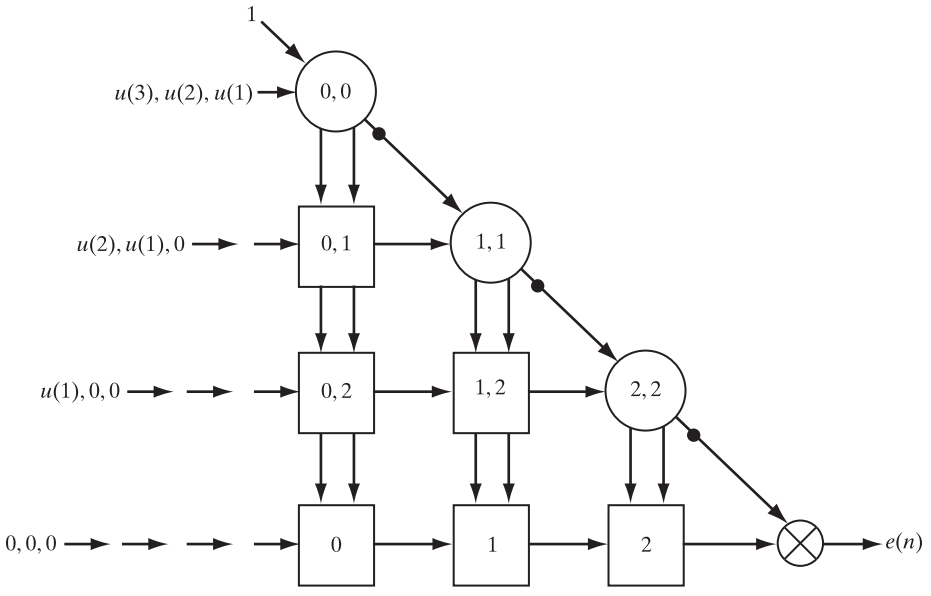


FIGURE 15.4 Systolic array for solving the MVDR beamforming problem.

- The estimation error is given by

$$e'(n) = (e'(n)\gamma^{-1/2}(n))(\gamma^{1/2}(n)), \tag{15.77}$$

where $e'(n)\gamma^{-1/2}(n)$ and $\gamma^{1/2}(n)$ are read directly from the nonzero entries of the second column of the postarray.

Finally, we may implement the MVDR beamformer by using the systolic array structure shown in Fig. 15.4, which is basically the same as that of Fig. 15.1 except for some minor changes (McWhirter & Shepherd, 1989). Specifically, $d(n)$ is set equal to zero for all n. With this change in place, we note the following from Fig. 15.4:

- The auxiliary vector $\mathbf{a}(n)$ is generated and stored in the bottom row of cells.
- The output of the final processing cell is identically equal to $-e'(n)$.

With a continuing sequence of snapshots $\mathbf{u}(n), \mathbf{u}(n+1), \ldots$ applied to the systolic array processor in Fig. 15.4, a corresponding sequence of estimation errors $e(n), e(n+1), \ldots$ is generated by the MVDR beamformer in accordance with Eq. (15.73).

Computer Experiment

We now illustrate the performance of the systolic array implementation of an adaptive MVDR beamformer by considering a linear array of five uniformly spaced sensors. The spacing d between adjacent elements equals one-half of the received wavelength. The array operates in an environment that consists of a target signal and a single interference, which originate from uncorrelated sources. The exponential weighting factor $\lambda = 1$.

The aims of the experiment are twofold:

1. To examine evolution of the adapted spatial response (pattern) of the beamformer across time.
2. To evaluate the effect of varying the interference-to-target ratio on the interference-nulling capability of the beamformer.

The directions of the target and source of interference are as follows:

Excitation	Actual angle of incidence, φ, measured with respect to normal to the array (radians)
Target	$\sin^{-1}(0.2)$
Interference	0

The steering vector is defined by

$$\mathbf{s}^{\mathrm{T}}(\theta) = [1, e^{-\mathrm{j}\theta}, e^{-\mathrm{j}2\theta}, e^{-\mathrm{j}3\theta}, e^{-\mathrm{j}4\theta}], \tag{15.78}$$

where the electrical angle

$$\theta = \pi \sin \varphi, \tag{15.79}$$

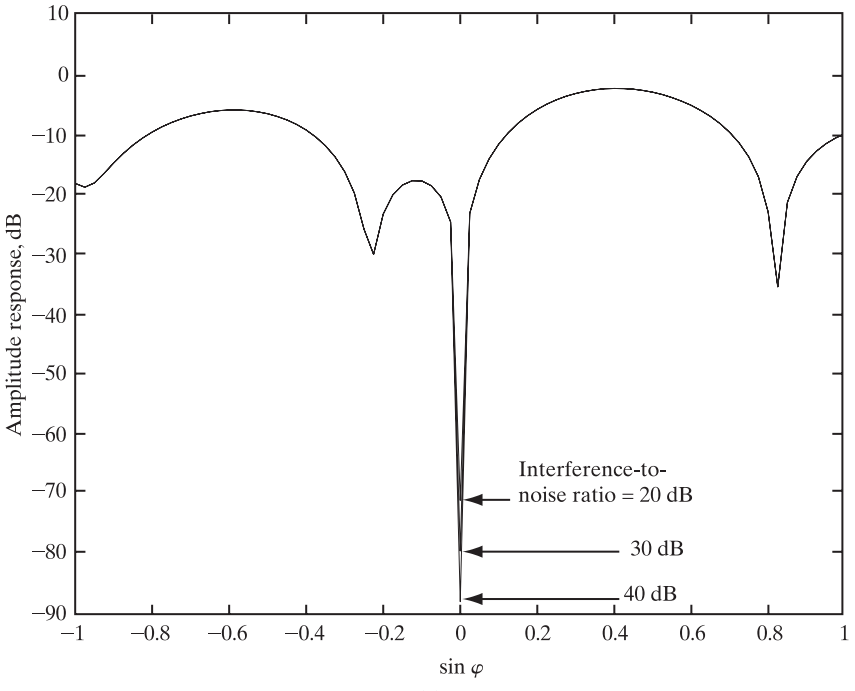
in which φ is the actual angle of incidence.

The data set used for the experiment consists of three components: a target signal, elemental receiver noise, and an interfering signal. The target signal and the interfering signal originate in the far field of the array antenna and are therefore represented by plane waves impinging on the array along their respective directions. Let these directions be denoted by angles φ_1 and φ_2, measured (in radians) with respect to the normal to the array antenna. The elemental signals of the array antenna are thus expressed in baseband form as

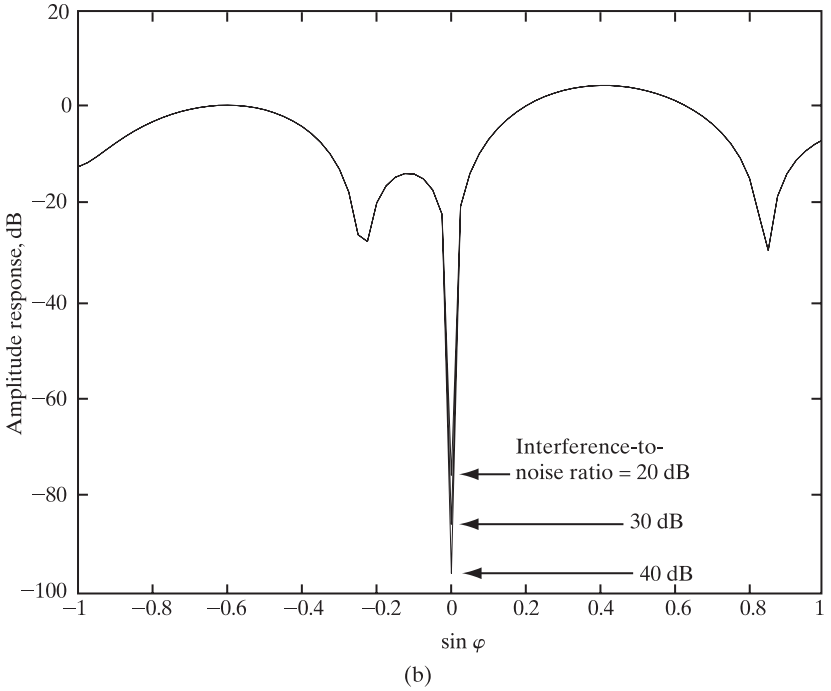
$$u(n) = A_0 \exp(\mathrm{j}n\theta_0) + A_1 \exp(\mathrm{j}n\theta_1 + \mathrm{j}\psi) + \nu(n), \quad n = 0, 1, 2, 3, 4, \tag{15.80}$$

where A_0 is the amplitude of the target signal and A_1 is the amplitude of the interfering signal. The electrical angles θ_0 and θ_1 are related to the individual angles of arrival φ_0 and φ_1, respectively, by Eq. (15.79). Since the target and interfering signals are uncorrelated, the phase difference ψ associated with the second component in Eq. (15.80) is a random variable uniformly distributed over the interval $(0, 2\pi]$. Lastly, the additive receiver noise $\nu(n)$ is a complex-valued Gaussian random variable with zero mean and unit variance. The target-to-noise ratio is held constant at 10 dB; the interference-to-noise ratio is variable, assuming the values 40, 30, and 20 dB.

Figure 15.5 shows the effects of varying the target-to-interference ratio and the number of snapshots (excluding those needed for initialization) on the adapted response of the beamformer. The amplitude response, expressed in decibels, is obtained by computing $20 \log_{10}|e(n)e^{\mathrm{j}n\theta}|$, where multiplication by the exponential factor $e^{\mathrm{j}n\theta}$ provides a means of spatially sampling the beamformer output $e(n)$. The results are presented in three parts, corresponding to 20, 100, and 200 snapshots; each part includes the three different values of interference-to-noise ratio, namely, 40, 30, and 20 dB.

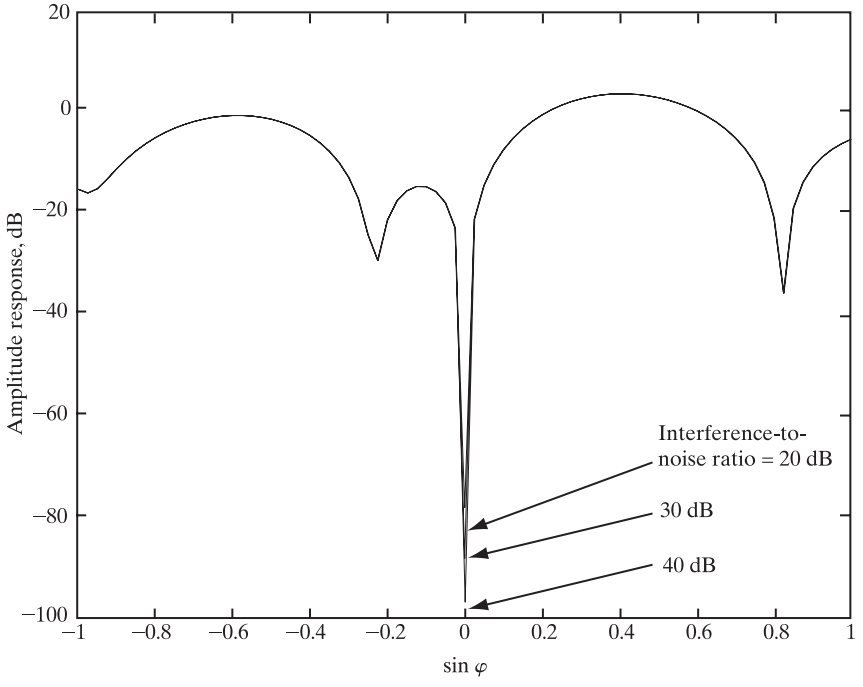


(a)



(b)

FIGURE 15.5 Results of the computer experiment on the spatial response of the systolic MVDR beamformer for varying interference-to-noise ratio and different number of snapshots: (a) $n = 20$; (b) $n = 100$; and (c) $n = 200$. Part (c) of the figure is shown on the next page.



(c)

FIGURE 15.5 (*continued*)

On the basis of the results, presented in Fig. 15.5, we may make the following observations:

- The response of the beamformer along the target is held fixed at a value of unity (i.e., 0 dB) under all conditions, as required.
- With as few as 20 snapshots, excluding initialization, the beamformer exhibits a reasonably effective nulling capability, which continually improves as the beamformer processes more snapshots.
- The response of the beamformer is relatively insensitive to variations in the interference-to-target ratio.

15.5 INVERSE QRD-RLS ALGORITHM

We now come to our last square-root adaptive filtering algorithm, known as the *inverse QRD-RLS algorithm*.[2] The algorithm derives its name from the fact that, instead of operating on the correlation matrix $\mathbf{\Phi}(n)$ as in the conventional QRD-RLS algorithm, the new

[2]The important virtue of the inverse QRD-RLS algorithm is that it permits the computation of a least-squares filter's weights systolically. As such, it is a *better alternative* to the *extended* QRD-RLS *algorithm*, the original motivation for which was to eliminate the need for using back substitution via the linear section in the Gentleman–Kung array. However, in historical terms, the invention of the extended QRD-RLS algorithm (Hudson & Shepherd, 1989) preceded that of the inverse QRD-RLS algorithm (Alexander & Ghirnikar, 1993). A derivation of the extended QRD-RLS algorithm is presented as Problem 3.

algorithm operates on the inverse of $\mathbf{\Phi}(n)$ (Pan & Plemmons, 1989; Alexander & Ghirnikar, 1993). More specifically, in square-root terms, the inverse QRD-RLS algorithm propagates $\mathbf{P}^{1/2}(n) = \mathbf{\Phi}^{-1/2}(n)$. In light of the fact that the Kalman parameter $\mathbf{K}(n)$ corresponds to the weighted RLS parameter $\lambda^{-1}\mathbf{P}(n)$, we see that the inverse QRD-RLS algorithm is basically a reformulation of the square-root covariance filtering algorithm.

Referring to Eq. (15.26), which pertains to the square-root covariance filtering algorithm, we readily see that (after canceling the common term $\lambda^{-1/2}$ from the bottom rows) the corresponding prearray-to-postarray transformation for the inverse QRD-RLS algorithm may be written as

$$\begin{bmatrix} 1 & \lambda^{-1/2}\mathbf{u}^{\mathrm{H}}(n)\mathbf{P}^{1/2}(n-1) \\ \mathbf{0} & \lambda^{-1/2}\mathbf{P}^{1/2}(n-1) \end{bmatrix} \mathbf{\Theta}(n) = \begin{bmatrix} \gamma^{-1/2}(n) & \mathbf{0}^{\mathrm{T}} \\ \mathbf{k}(n)\gamma^{-1/2}(n) & \mathbf{P}^{1/2}(n) \end{bmatrix}, \tag{15.81}$$

where $\mathbf{\Theta}(n)$ is a unitary rotation that operates on the block entry $\lambda^{-1/2}\mathbf{u}^{\mathrm{H}}(n)\mathbf{P}^{1/2}(n-1)$ in the prearray by annihilating its elements, one by one, so as to produce a block zero entry in the first row of the postarray. The gain vector $\mathbf{k}(n)$ of the RLS algorithm is readily obtained from the entries in the first column of the postarray in Eq. (15.81) by writing

$$\mathbf{k}(n) = (\mathbf{k}(n)\gamma^{-1/2}(n))(\gamma^{-1/2}(n))^{-1}. \tag{15.82}$$

Hence, the least-squares weight vector may be updated in accordance with the recursion

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n), \tag{15.83}$$

where the a priori estimation error is defined in the usual way:

$$\xi(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n-1)\mathbf{u}(n). \tag{15.84}$$

A summary of the inverse QRD-RLS algorithm, including initial conditions, is presented in Table 15.4.

TABLE 15.4 Summary of the Inverse QRD-RLS Algorithm

Inputs:
- data matrix: $\mathbf{A}^{\mathrm{H}}(n) = \{\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(n)\}$
- desired response vector: $\mathbf{d}^{\mathrm{H}}(n) = \{d(1), d(2), \ldots, d(n)\}$

Prescribed parameters:
- exponential weighting factor $= \lambda$
- regularization parameter $= \delta$

Initial conditions:

$$\mathbf{P}^{1/2}(0) = \delta^{-1/2}\mathbf{I}$$
$$\hat{\mathbf{w}}(0) = \mathbf{0}$$

Computations:

For $n = 1, 2, \ldots$, compute

$$\begin{bmatrix} 1 & \lambda^{-1/2}\mathbf{u}^{\mathrm{H}}(n)\mathbf{P}^{1/2}(n-1) \\ \mathbf{0} & \lambda^{-1/2}\mathbf{P}^{1/2}(n-1) \end{bmatrix} \mathbf{\Theta}(n) = \begin{bmatrix} \gamma^{-1/2}(n) & \mathbf{0}^{\mathrm{T}} \\ \mathbf{k}(n)\gamma^{-1/2}(n) & \mathbf{P}^{1/2}(n) \end{bmatrix},$$

where $\mathbf{\Theta}(n)$ is a unitary rotation that produces a zero entry in the first row of the postarray.

$$\mathbf{k}(n) = (\mathbf{k}(n)\gamma^{-1/2}(n))(\gamma^{-1/2}(n))^{-1}$$
$$\xi(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n-1)\mathbf{u}(n)$$
$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n)$$

Here we note that, since the square root $\mathbf{\Phi}^{1/2}(n)$ is lower triangular in accordance with Eq. (15.45), its inverse matrix $\mathbf{\Phi}^{-1/2}(n) = \mathbf{P}^{1/2}(n)$ is upper triangular.

The inverse QRD-RLS algorithm differs from the conventional QRD-RLS algorithm in a fundamental way, namely, the input data vector $\mathbf{u}(n)$ does not appear by itself as a block entry in the prearray of the algorithm; rather, it is multiplied by $\lambda^{-1/2}\mathbf{P}^{1/2}(n-1)$. Hence, the input data vector $\mathbf{u}(n)$ has to be preprocessed prior to performing the rotations described in Eq. (15.81). The *preprocessor* to do this simply computes the inner product of $\mathbf{u}(n)$, with each of the columns of the square-root matrix $\mathbf{P}^{1/2}(n-1)$ scaled by $\lambda^{-1/2}$. The preprocessor can be structured to take advantage of the upper triangular form of $\mathbf{P}^{1/2}(n-1)$.

The systolic processing in the inverse QRD-RLS algorithm lends itself to implementation in the form of two sections connected together as illustrated in Fig. 15.6:

- The *triangular systolic array* operates on the preprocessed input vector $\lambda^{-1/2}\mathbf{P}^{H/2}(n-1)\mathbf{u}(n)$ in accordance with the Hermitian transposed form of Eq. (15.81). Nonzero elements of the updated matrix $\mathbf{P}^{H/2}(n)$ are stored in the internal cells of the systolic array. The two other products of the systolic computation are $\gamma^{-1/2}(n)$ and $\gamma^{-1/2}(n)\ \mathbf{k}^{H}(n)$.
- The *linear section*, which is appended to the triangular section, operates on the latter two products of the systolic computation to produce the elements of the updated weight vector $\hat{\mathbf{w}}(n)$ in accordance with Eqs. (15.82), (15.84), and (15.83), in that order.

The combination of these two sections can be designed to operate in a completely parallel fashion (Alexander & Ghirnikar, 1993).

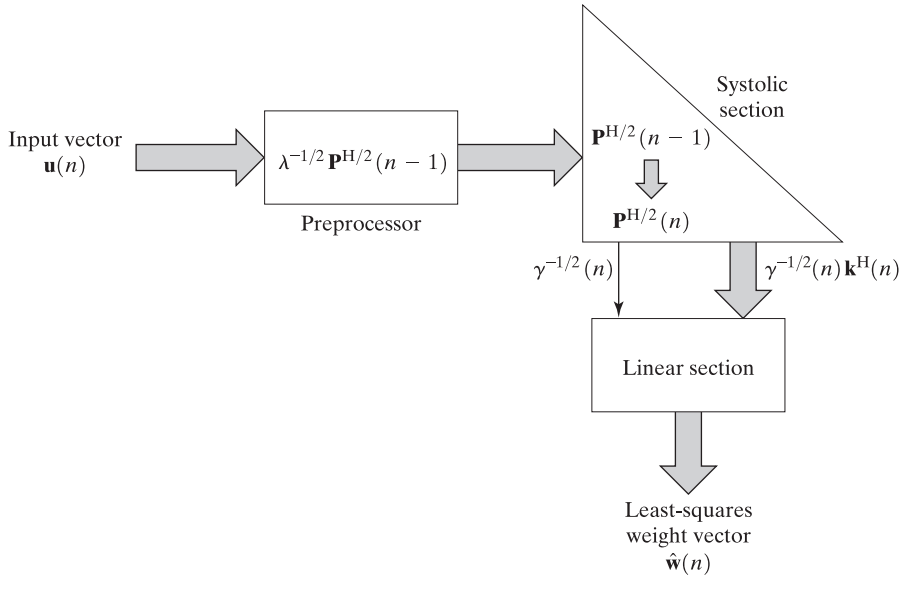


FIGURE 15.6 Block diagram of the inverse QRD-RLS algorithm.

15.6 FINITE-PRECISION EFFECTS

In Chapter 12, we discussed the impact of finite-precision effects on the least-mean-square (LMS) and RLS algorithms. In this section, we build on the material presented in that chapter by discussing the impact of finite-precision effects on the performance of square-root adaptive filtering algorithms—the focus of attention in this chapter.

QRD-RLS Algorithm

It is generally agreed that QR-decomposition is one of the best numerical procedures for solving the RLS estimation problem because of two important properties:

1. QR-decomposition operates on the input data directly.
2. QR-decomposition involves the use of only numerically well-behaved unitary rotations (e.g., Givens rotations).

In particular, the QRD-RLS algorithm propagates the square root of the correlation matrix $\Phi(n)$ rather than $\Phi(n)$ itself. Hence, the condition number of $\Phi^{1/2}(n)$ equals the square root of the condition number of $\Phi(n)$. This results in a significant reduction in the dynamic range of data handled by QR-decomposition–based algorithms and, in turn, a more accurate computation than the traditional RLS algorithm that propagates $\Phi(n)$. Moreover, the finite-precision form of the QRD-RLS algorithm is *stable in a bounded-input, bounded-output (BIBO) sense* (Leung & Haykin, 1989). However, it must be stressed that the BIBO stability of the QRD-RLS algorithm does not guarantee that the various quantities computed by the algorithm remain meaningful in any sense when the algorithm is operating in a finite-precision environment (Yang & Böhme, 1992). In particular, a unitary rotation (e.g., a sequence of Givens rotations) is used to annihilate a certain vector in the prearray and then operate on other related entries in the prearray. A perturbation in internal computations may produce a corresponding perturbation in rotation angles, which introduces yet another source of numerical error in the rotated entries of the postarray. These errors, in turn, produce further perturbations of their own in subsequent computations of the rotation angles, and the process goes on. The net result is that we have a *complicated parametric feedback system*, and it is not entirely clear whether this feedback system is in fact numerically stable.

Yang and Böhme (1992) presented experimental results that demonstrate the numerical stability of the QRD-RLS algorithm for $\lambda < 1$; they used the algorithm to perform adaptive prediction of an autoregressive (AR) process. All the computer simulations reported in that paper were performed on a personal computer (PC), using floating-point arithmetic. To observe finite-precision effects in a reasonable simulation time, the effective number of mantissa bits in the floating-point representation was reduced by truncating the mantissa at a predefined position without affecting the exponent. In the experiments reported by Yang and Böhme, the mantissa length took on the values 52, 12, and 5 bits; the resulting changes in wordlength were found to have only a minor effects on the convergence behavior of the algorithm. In addition, Yang and Böhme showed that the QRD-RLS algorithm diverges when $\lambda = 1$.

The numerical stability of the QRD-RLS algorithm for $\lambda < 1$ was also demonstrated experimentally by Ward et al. (1986) in the context of adaptive beamforming. In particular, they showed that, for the same number of bits of arithmetic precision, the QRD-RLS algorithm offers a significantly better performance than the sample matrix inversion algorithm described in Reed et al. (1974).

Inverse QRD-RLS Algorithm

The inverse QRD-RLS algorithm propagates $\mathbf{P}^{1/2}(n)$—the square root of the inverse correlation matrix $\mathbf{P}(n) = \boldsymbol{\Phi}^{-1}(n)$. Although the inverse QRD-RLS algorithm differs from the QRD-RLS algorithm that propagates $\boldsymbol{\Phi}^{1/2}(n)$, the two algorithms do share a common feature: They both avoid the propagation of the Hermitian inverse of their respective matrix quantities. Accordingly, the inverse QRD-RLS algorithm is able to exploit the good numerical properties of the QR-decomposition in a manner similar to that of the QRD-RLS algorithm.

For $\lambda < 1$, the propagation of a single error in the inverse QRD-RLS algorithm (and, for that matter, in the QRD-RLS algorithm) is *exponentially stable*. The rationale for this statement follows the numerical stability analysis of the RLS algorithm presented in Section 12.3. However, for $\lambda = 1$, the single-error propagation is *not* contractive. It may therefore be conjectured that the accumulation of quantization errors can cause the inverse QRD-RLS algorithm to be numerically divergent; this phenomenon has been confirmed experimentally, using computer simulations.[3] A similar remark applies to the QRD-RLS algorithm itself, for which experimental validation is presented in Yang and Böhme (1992).

In sum, regarding the requirement to operate a square-root adaptive filtering algorithm in a finite-precision environment, we may state that[4]

- If only the estimation error $e(n)$ is required, the QRD-RLS algorithm is the preferred choice.
- If the weight vector $\hat{\mathbf{w}}(n)$ is required, the inverse QRD-RLS algorithms is a good candidate.

15.7 SUMMARY AND DISCUSSION

In this chapter, we discussed the derivations of two square-root adaptive filtering algorithms for exponentially weighted recursive least-squares (RLS) estimation in a unified manner. The algorithms are known as the QRD-RLS algorithm and the inverse QRD-RLS algorithm. These two algorithms were derived by exploiting their one-to-one correspondences with the square-root information filtering and the square-root covariance filtering algorithms, respectively, which represent square-root variants of

[3]Yang private communication, 1995.

[4]The summarizing comments presented herein are based on Yang private communication, 1995.

the celebrated Kalman filter. The QRD-RLS algorithm and the inverse QRD-RLS algorithm propagate a *single* square root each, namely, $\mathbf{\Phi}^{1/2}(n)$ and $\mathbf{P}^{1/2}(n) = \mathbf{\Phi}^{-1/2}(n)$, respectively.

A common feature of the QRD-RLS algorithm and the inverse QRD-RLS algorithm is that, in varying degrees, they lend themselves to parallel implementation in the form of systolic arrays. Naturally, the actual details of the systolic array implementations depend on which algorithm is being considered. In particular, there are some basic differences that should be carefully noted. The QRD-RLS algorithm operates directly on the input data. On the other hand, in the inverse QRD-RLS algorithm, the input data vector $\mathbf{u}(n)$ is multiplied by the scaled square-root matrix $\lambda^{-1/2}\mathbf{P}^{H/2}(n-1)$ before it is processed by the systolic array. This adds computational complexity to the parallel implementation of the inverse QRD-RLS algorithm.

The parallel implementation of the inverse QRD-RLS algorithm permits a direct computation of the least-squares weight vector in an efficient manner. Accordingly, this square-root adaptive filtering algorithm is well suited for applications such as system identification, spectrum estimation, and adaptive equalization, wherein knowledge of the weight vector is a desirable requirement. In contrast, the QRD-RLS algorithm computes the a posteriori estimation error $e(n)$, which basically restricts its application to areas such as adaptive beamforming and acoustic echo cancellation, in which it is *not* necessary to have explicit knowledge of the weight vector.

Finally, the point that needs to be stressed is that both the QRD-RLS and inverse QRD-RLS algorithms preserve the desirable convergence properties of the traditional RLS algorithm, namely, a fast rate of convergence and insensitivity to variations in the eigenvalue spread of the correlation matrix of incoming data.

PROBLEMS

1. For the prearray-to-postarray transformation described in Eq. (15.32) for the square-root information filter, illustrate the summary of the square root information-filtering algorithm.

2. In this problem, we revisit the square-root information filtering algorithm. Specifically, the term $\nu(n)$ in the state-space model of Eqs. (15.2) and (15.3) is assumed to be a random variable of zero mean and variance $Q(n)$. Show that the square-root information filtering algorithm may now be formulated as

$$\mathbf{K}^{-1}(n) = \lambda(\mathbf{K}^{-1}(n-1) + Q^{-1}(n)\mathbf{u}(n)\mathbf{u}^{H}(n)),$$

$$\mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \lambda^{1/2}(\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + Q^{-1}(n)\mathbf{u}(n)y(n)),$$

which includes Eqs. (15.29) and (15.30) as a special case.

3. **(a)** Starting with the prearray

$$\begin{bmatrix} \lambda^{1/2}\mathbf{K}^{-H/2}(n-1) & \lambda^{1/2}\mathbf{u}(n) \\ \hat{\mathbf{x}}^{H}(n|\mathcal{Y}_{n-1})\mathbf{K}^{-H/2}(n-1) & y^{*}(n) \\ \mathbf{0}^{T} & 1 \\ \lambda^{-1/2}\mathbf{K}^{1/2}(n-1) & \mathbf{0} \end{bmatrix},$$

which is the expanded version of the prearray in Eq. (15.38), show that the *extended square-root information filtering algorithm* is described by

$$\begin{bmatrix} \lambda^{1/2}\mathbf{K}^{-\mathrm{H}/2}(n-1) & \lambda^{1/2}\mathbf{u}(n) \\ \hat{\mathbf{x}}^{\mathrm{H}}(n|\mathcal{Y}_{n-1})\mathbf{K}^{-\mathrm{H}/2}(n-1) & y^*(n) \\ \mathbf{0}^{\mathrm{T}} & 1 \\ \lambda^{-1/2}\mathbf{K}^{1/2}(n-1) & \mathbf{0} \end{bmatrix} \boldsymbol{\Theta}(n)$$

$$= \begin{bmatrix} \mathbf{K}^{-\mathrm{H}/2}(n) & \mathbf{0} \\ \hat{\mathbf{x}}^{\mathrm{H}}(n+1|\mathcal{Y}_n)\mathbf{K}^{-\mathrm{H}/2}(n) & r^{-1/2}(n)\alpha^*(n) \\ \lambda^{1/2}\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}^{1/2}(n) & r^{-1/2}(n) \\ \mathbf{K}^{1/2}(n) & -\mathbf{g}(n)r^{1/2}(n) \end{bmatrix}.$$

Hence, formulate an expression for the updated state estimate $\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$.

(b) Using the results of part (a), show that the *extended QRD-RLS algorithm* is described by the following recursions:

For $n = 1, 2, \ldots$, compute

$$\begin{bmatrix} \lambda^{1/2}\boldsymbol{\Phi}^{1/2}(n-1) & \mathbf{u}(n) \\ \lambda^{1/2}\mathbf{p}^{\mathrm{H}}(n-1) & d(n) \\ \mathbf{0}^{\mathrm{T}} & 1 \\ \lambda^{-1/2}\boldsymbol{\Phi}^{-\mathrm{H}/2}(n-1) & \mathbf{0} \end{bmatrix} \boldsymbol{\Theta}(n) = \begin{bmatrix} \boldsymbol{\Phi}^{1/2}(n) & \mathbf{0} \\ \mathbf{p}^{\mathrm{H}}(n) & \xi(n)\gamma^{1/2}(n) \\ \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\Phi}^{-\mathrm{H}/2}(n) & \gamma^{1/2}(n) \\ \boldsymbol{\Phi}^{-\mathrm{H}/2}(n) & -\mathbf{k}(n)\gamma^{-1/2}(n) \end{bmatrix},$$

where $\boldsymbol{\Theta}(n)$ is a unitary rotation that operates on the prearray to produce a block zero entry in the top block row of the postarray. How is the least-squares weight vector computed by the algorithm?

4. The extended QRD-RLS algorithm is defined by the following recursions (see Problem 3):

For $n = 1, 2, \ldots$, compute

$$\begin{bmatrix} \lambda^{1/2}\boldsymbol{\Phi}^{1/2}(n-1) & \mathbf{u}(n) \\ \lambda^{1/2}\mathbf{p}^{\mathrm{H}}(n-1) & d(n) \\ \mathbf{0}^{\mathrm{T}} & 1 \\ \lambda^{-1/2}\boldsymbol{\Phi}^{-\mathrm{H}/2}(n-1) & \mathbf{0} \end{bmatrix} \boldsymbol{\Theta}(n) = \begin{bmatrix} \boldsymbol{\Phi}^{1/2}(n) & \mathbf{0} \\ \mathbf{p}^{\mathrm{H}}(n) & \xi(n)\gamma^{1/2}(n) \\ \mathbf{u}^{\mathrm{H}}(n)\boldsymbol{\Phi}^{-\mathrm{H}/2}(n) & \gamma^{1/2}(n) \\ \boldsymbol{\Phi}^{-\mathrm{H}/2}(n) & -\mathbf{k}(n)\gamma^{-1/2}(n) \end{bmatrix}$$

and

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + (\mathbf{k}(n)\gamma^{-1/2}(n))(\xi(n)\gamma^{-1/2}(n))^*,$$

where $\boldsymbol{\Theta}(n)$ is a unitary rotation that operates on the prearray to produce a block zero entry in the top block row of the postarray.

Let

$$\mathbf{X}(n) = \boldsymbol{\Phi}^{-\mathrm{H}/2}(n) + \boldsymbol{\eta}_x(n)$$

denote the quantized version of the Hermitian inverse matrix $\boldsymbol{\Phi}^{-\mathrm{H}/2}(n)$. Show that error propagation due to $\boldsymbol{\eta}_x(n)$ in the extended QRD-RLS algorithm at time step $n-1$ is not necessarily stable, in that local errors tend to grow unboundedly.

5. Let the n-by-n unitary matrix $\mathbf{Q}(n)$ involved in the QR-decomposition of the data matrix $\mathbf{A}(n)$ be partitioned as follows:

$$\mathbf{Q}(n) = \begin{bmatrix} \mathbf{Q}_1(n) \\ \mathbf{Q}_2(n) \end{bmatrix}.$$

Here, $\mathbf{Q}_1(n)$ has the same number of rows as the upper triangular matrix $\mathbf{R}(n)$ in the QR-decomposition of $\mathbf{A}(n)$. Suppose that the exponential weighting factor $\lambda = 1$.

According to the method of least squares presented in Chapter 9, the projection operator is

$$\mathbf{P}(n) = \mathbf{A}(n)(\mathbf{A}^{\mathrm{H}}(n)\mathbf{A}(n))^{-1}\mathbf{A}^{\mathrm{H}}(n).$$

Show that, for the problem at hand,

$$\mathbf{P}(n) = \mathbf{Q}_1^{\mathrm{H}}(n)\mathbf{Q}_1(n).$$

How is the result modified for the case where $0 < \lambda \leq 1$?

6. Explain the way in which the systolic array structure of Fig. 15.1 may be used to operate as a prediction-error filter.

7. In describing the systolic implementation of the QRD-RLS algorithm in Section 15.3, compute the ith element of the M-by-1 weight vector $\mathbf{w}^{\mathrm{H}}(\mathrm{n})$ by freezing the state of the processor at adaptation cycle n, setting the desired response equal to zero, and feeding the processor with an input vector in which the ith element is unity and the remaining $M-1$ elements are all zero.

8. Figure P15.1 is a block diagram representation of an MVDR beamforming algorithm. The triangular array in part (a) of the figure is frozen at adaptation cycle n, and the steering vector $\mathbf{s}(\theta)$ is input into the array. The stored $\mathbf{R}^{\mathrm{H}}(n)$ of the array and its output $\mathbf{a}^{\mathrm{H}}(n)$ are applied to a linear systolic section in part (b) of the figure.

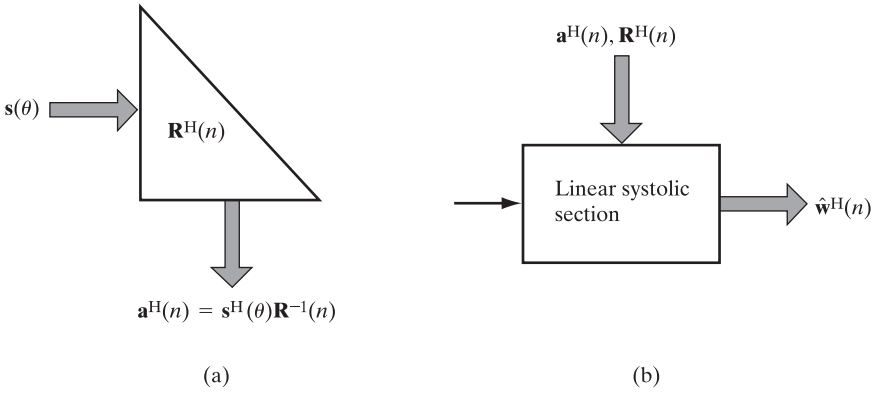


FIGURE P15.1

(a) Show that the output of the triangular array is

$$\mathbf{a}^{\mathrm{H}}(n) = \mathbf{s}^{\mathrm{H}}(\theta)\mathbf{R}^{-1}(n).$$

(b) Using the method of back substitution, show that the linear systolic array produces the Hermitian transposed weight vector $\mathbf{w}^{\mathrm{H}}(n)$ as its output.

9. By using the systolic array structure shown in Fig. 15.4, explain how to implement a MVDR beamformer.

10. Illustrate the condition used, as well as the procedure, for exact initialization for a QRD-RLS algorithm.

Computer Experiments

11. In this problem, revisit the computer experiment on MVDR beamforming described in Section 15.4, which involves *uncorrelated* sources of target signal and interference. The target signal-to-noise ratio is 10 dB, and the interference-to-noise ratio is 40 dB. As before, the angle-of-arrival for the interference is

$$\varphi_{\text{interf}} = \sin^{-1}(0).$$

This time, however, investigate what happens to the spatial response of the beamformer as the target moves closer to the interference. Thus, plot the spatial response of the beamformer for an increasing number of adaptation cycles for each of the following angles-of-arrival for the target:

(i) $\varphi_{\text{target}} = \sin^{-1}(-0.15)$,
(ii) $\varphi_{\text{target}} = \sin^{-1}(-0.10)$,
(iii) $\varphi_{\text{target}} = \sin^{-1}(-0.05)$.

Comment on your results.

12. The sources of target signal and interference in the computer experiment of Problem 11 are uncorrelated with each other. In this problem, we study what happens to the spatial response of the MVDR beamformer when these two sources are *correlated*. That is, the elemental signals of the array antenna, defined in baseband form, are now as follows:

$$u(n) = A_0 \exp(\mathrm{j}n\theta_0) + A_1 \exp(\mathrm{j}n\theta_1 + \mathrm{j}\psi_1) + \nu(n), \qquad n = 0, 1, 2, 3, 4,$$

where A_0 and A_1 are, respectively, the amplitudes of the target signal and interference, θ_0 and θ_1 are their respective electrical angles, and ψ_1 is some fixed phase shift.

Repeat the computer experiment of Problem 11 for the new input $u(n)$.

CHAPTER 16

Order-Recursive Adaptive Filtering Algorithm

In this chapter, we develop another class of linear adaptive filters whose designs are based on algorithms that involve both *order-update and time-update recursions*.[1] Except for Chapter 5, these adaptive filters distinguish themselves from those studied in previous chapters by virtue of order updates, which are made possible by exploiting the *time-shifting property* of uniformly sampled temporal data. In structural terms, order updates lead to a *computationally efficient, modular, latticelike structure* whereby certain information gathered from previous computations for filter order $m - 1$ is carried over to the updated filter order m. The net result is the realization of an adaptive filter whose computational complexity is *linear* in filter order (length) m.

Recall that the gradient adaptive lattice (GAL) algorithm, covered in Chapter 5, is also a computationally efficient adaptive lattice-like filtering algorithm in its own way; it is rooted in the method of stochastic gradient descent, just like the celebrated least-mean-square (LMS) algorithm. On the other hand, the corresponding order-recursive adaptive filtering algorithms, occupying this whole chapter, belong to the class of *least-squares estimators* known for their exact algebraic formulations; hence, they are referred to as *order-recursive least-squares lattice (LSL) adaptive filtering algorithms*.

Another point to note: Each lattice module of the GAL algorithm has a single complex-valued reflection coefficient. In direct contrast, each module of an order-recursive LSL adaptive filtering algorithm has a different pair of complex-valued reflection coefficients, which, compared to the GAL algorithm, results in a much more sophisticated algebraic formulation.

[1]The family of order-recursive adaptive filtering algorithms that is discussed in this chapter is part of a larger class of adaptive filtering algorithms known collectively as *fast algorithms*. In the context of recursive least-squares (RLS) estimation, an algorithm is said to be fast if its *computational complexity increases linearly with the number of adjustable parameters*. A fast algorithm is therefore similar to the least-mean-square (LMS) algorithm in its computational requirement, but more demanding in coding terms.

The family of fast algorithms also includes the so-called *fast transversal filter (FTF),* which involves the combined use of four transversal (i.e., finite-duration impulse response) filters; these filters account for forward and backward prediction errors, gain-vector computation, and joint-process estimation. The FTF algorithm is mathematically elegant; unfortunately, the algorithm has a tendency to become *unstable* when it is implemented in finite-precision arithmetic.

The bulk of the material covered in this chapter addresses the many mathematical aspects of a *numerically robust* order-recursive algorithm, which involves the combined use of a posteriori as well as a priori estimation errors that are related by a conversion factor. In the latter part of the chapter, we finish the discussion by describing a simplified version of this algorithm that requires the use of a priori estimation errors. However, the algebraic simplification is achieved at the cost of numerical robustness, which is yet another example of the no-free-lunch theorem.

16.1 ORDER-RECURSIVE ADAPTIVE FILTERS USING LEAST-SQUARES ESTIMATION: AN OVERVIEW

In Chapter 14, we established the correspondence between Kalman filters and RLS algorithms. In this chapter, we use that correspondence to formulate the most basic *state-space model* of lattice filtering, which, in turn, makes it possible to exploit the relevant aspects of Kalman filter theory in developing the algorithmic formulation of order-recursive adaptive filters using exact least-squares estimation. As already remarked, least-squares order-recursive adaptive filtering algorithms involve both forward and backward predictions, much like the GAL filtering algorithm. However, unlike their GAL counterpart that uses approximate estimation with a single reflection coefficient, the order-recursive adaptive filtering algorithms discussed in this chapter use exact least-squares estimation with a pair of reflection coefficients, one for the forward prediction and the other for the backward prediction.

There are two types of order-recursive filters using least-squares estimation:

1. *QR-decomposition-based least-squares lattice (QRD-LSL) adaptive filters*, which rely on the use of *unitary rotations* for QR-decomposition. The purpose of a unitary rotation is to produce a postarray in which an entry in the prearray is annihilated.
2. *Recursive least-squares lattice (LSL) adaptive filters*, which are obtained by squaring the arrays in the QRD-LSL algorithm. The effect of squaring is to remove the unitary rotations from the algorithm, thereby simplifying the implementation. However, this simplification is attained at the cost of reduced numerical accuracy and possible instability when a recursive LSL algorithm is implemented in finite-precision arithmetic.

It is thus apparent that the QRD-LSL algorithm is the basic order-recursive adaptive filtering algorithm—hence the order in which the algorithmic derivations are presented in the rest of the chapter.

The computations involved in the QRD-LSL algorithm address the following items:

- Adaptive forward and backward linear predictors, which are characterized by independent parameter vectors of their own.
- A conversion factor, which provides the connecting link between different sets of a priori and a posteriori estimation errors.
- An LSL predictor, each stage of which is characterized by a pair of reflection coefficients.

- Angle normalization, which makes the formulation of the lattice predictor invariant to the choice of a priori and a posteriori errors.
- First-order state-space models for lattice filtering, the formulation of which paves the way for deriving the QRD-LSL algorithm.

In addressing these items, we develop the LSL version of the Levinson–Durbin recursion, which was discussed previously in Chapter 3. We also develop a time-update recursion for a certain cross-correlation function denoted by $\Delta_{m-1}(n)$, which holds the key to the QRD-LSL algorithm's going forward in time.

16.2 ADAPTIVE FORWARD LINEAR PREDICTION

Consider a forward linear predictor of order m, depicted in Fig. 16.1(a) for operation at adaptation cycle n. The tap-weight vector $\hat{\mathbf{w}}_{f,m}(n)$ of this predictor is optimized in the least-squares sense over the entire observation interval $1 \le i \le n$. Let

$$f_m(i) = u(i) - \hat{\mathbf{w}}_{f,m}^{\mathrm{H}}(n)\mathbf{u}_m(i-1), \qquad i = 1, 2, \ldots, n, \tag{16.1}$$

where the superscript **H** indicates Hermitian transposition (i.e., transposition combined with complex conjugation), denote the forward prediction error produced by the predictor at adaptation cycle i in response to the tap-input vector $\mathbf{u}_m(i-1)$ of size m. According to this definition, $u(i)$ plays the role of the desired response for forward linear prediction. The compositions of input vector $\mathbf{u}_m(i-1)$ and weight vector $\hat{\mathbf{w}}_m(n)$ are respectively as follows:

$$\begin{aligned}\mathbf{u}_m(i-1) &= [u(i-1), u(i-2), \ldots, u(i-m)]^{\mathrm{T}};\\ \hat{\mathbf{w}}_{f,m}(n) &= [w_{f,m,1}(n), w_{f,m,2}(n), \ldots, w_{f,m,m}(n)]^{\mathrm{T}},\end{aligned}$$

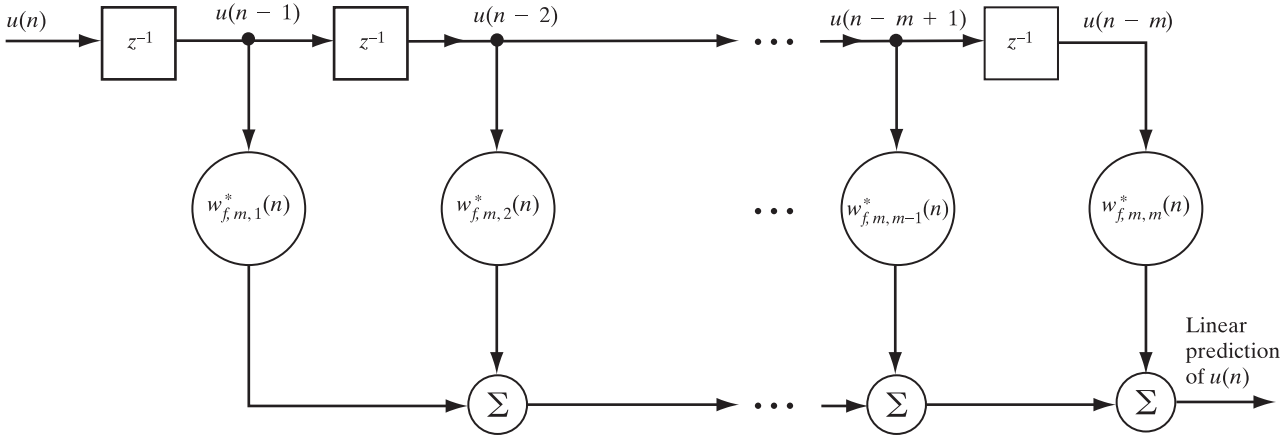
where the superscript T denotes transposition. We refer to $f_m(i)$ as the *forward a posteriori prediction error*, since its computation is based on the current value of the forward predictor's tap-weight vector, $\hat{\mathbf{w}}_{f,m}(n)$. Correspondingly, we may define the *forward a priori prediction error* as

$$\eta_m(i) = u(i) - \hat{\mathbf{w}}_{f,m}^{\mathrm{H}}(n-1)\mathbf{u}_m(i-1), \qquad i = 1, 2, \ldots, n, \tag{16.2}$$

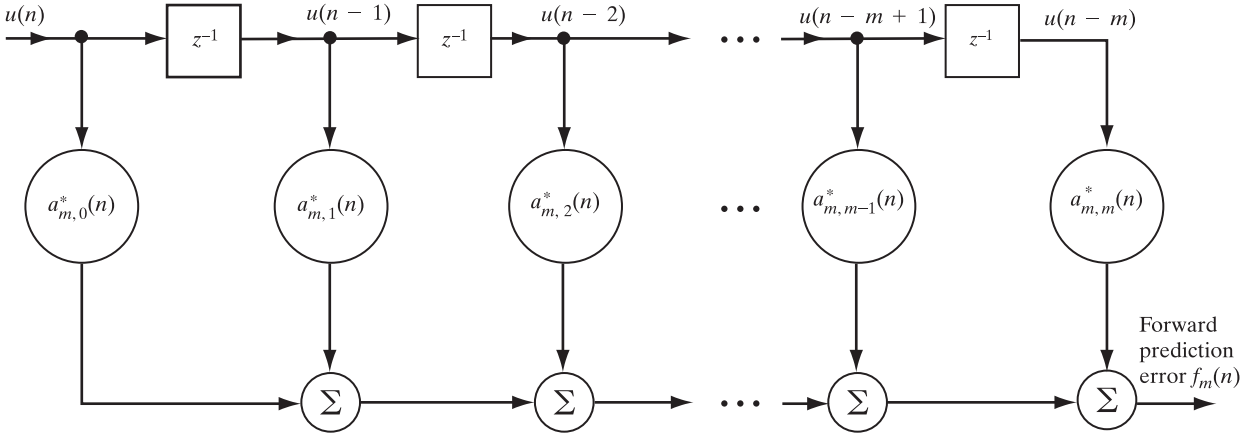
the computation of which is based on the past value of the forward predictor's tap-weight vector $\hat{\mathbf{w}}_{f,m}(n-1)$. In effect, $\eta_m(i)$ represents a form of innovation.

Table 16.1 lists the correspondences between the various quantities characterizing linear estimation in general and those characterizing forward linear prediction in particular, with the RLS algorithm in mind. With the aid of this table, it is a straightforward matter to modify the RLS algorithm developed in Sections 10.3 and 10.4 to write the recursions for adaptive forward linear prediction. Specifically, we deduce the following recursion for updating the tap-weight vector of the forward predictor:

$$\hat{\mathbf{w}}_{f,m}(n) = \hat{\mathbf{w}}_{f,m}(n-1) + \mathbf{k}_m(n-1)\eta_m^*(n), \tag{16.3}$$



(a)



(b)

FIGURE 16.1 (a) Forward predictor of order m; (b) corresponding predictor-error filter.

TABLE 16.1 Summary of Correspondences Between Linear Estimation, Forward Prediction, and Backward Prediction

Quantity	Linear estimation (general)	Forward linear prediction of order m	Backward linear prediction of order m
Tap-input vector	$\mathbf{u}(n)$	$\mathbf{u}_m(n-1)$	$\mathbf{u}_m(n)$
Desired response	$d(n)$	$u(n)$	$u(n-m)$
Tap-weight vector	$\hat{\mathbf{w}}(n)$	$\hat{\mathbf{w}}_{f,m}(n)$	$\hat{\mathbf{w}}_{b,m}(n)$
A posteriori estimation error	$e(n)$	$f_m(n)$	$b_m(n)$
A priori estimation error	$\xi(n)$	$\eta_m(n)$	$\beta_m(n)$
Gain vector	$\mathbf{k}(n)$	$\mathbf{k}_m(n-1)$	$\mathbf{k}_m(n)$
Minimum value of sum of weighted error squares	$\mathscr{E}_{\min}(n)$	$\mathscr{F}_m(n)$	$\mathscr{B}_m(n)$

where the asterisk denotes *complex conjugation.* Here, $\eta_m(n)$ is the forward a priori prediction error defined in Eq. (16.2) for $i = n$, and $\mathbf{k}_m(n-1)$ is the past value of the *gain vector* defined by

$$\mathbf{k}_m(n-1) = \boldsymbol{\Phi}_m^{-1}(n-1)\mathbf{u}_m(n-1). \tag{16.4}$$

The matrix $\boldsymbol{\Phi}_m^{-1}(n-1)$ is the inverse of the correlation matrix of the input data, defined by

$$\boldsymbol{\Phi}_m(n-1) = \sum_{i=1}^{n-1} \lambda^{n-1-i}\mathbf{u}_m(i)\mathbf{u}_m^{\mathrm{H}}(i). \tag{16.5}$$

The adaptive forward linear prediction problem just described is in terms of a predictor characterized by the tap-weight vector $\hat{\mathbf{w}}_{f,m}(n)$. Equivalently, we may describe the problem by specifying a *forward prediction-error filter*, as depicted in Fig. 16.1(b). Let $\mathbf{a}_m(n)$ denote the $(m+1)$-by-1 tap-weight vector of the prediction-error filter of order m. This tap-weight vector is related to that of the forward predictor in Fig. 16.1(a) by

$$\mathbf{a}_m(n) = \begin{bmatrix} 1 \\ -\hat{\mathbf{w}}_{f,m}(n) \end{bmatrix}, \tag{16.6}$$

where the first element of $\mathbf{a}_m(n)$, namely, $a_{m,0}(n)$, is unity. Then we may respectively redefine the forward a posteriori prediction error and the forward a priori prediction error as

$$f_m(i) = \mathbf{a}_m^{\mathrm{H}}(n)\mathbf{u}_{m+1}(i), \qquad i = 1, 2, \ldots, n, \tag{16.7}$$

and

$$\eta_m(i) = \mathbf{a}_m^{\mathrm{H}}(n-1)\mathbf{u}_{m+1}(i), \qquad i = 1, 2, \ldots, n, \tag{16.8}$$

where the input vector $\mathbf{u}_{m+1}(i)$ of size $m+1$ is partitioned in the following way:

$$\mathbf{u}_{m+1}(i) = \begin{bmatrix} u(i) \\ \mathbf{u}_m(i-1) \end{bmatrix}.$$

TABLE 16.2 Principle of Orthogonality Summarized for Linear Estimation, Forward Prediction, and Backward Prediction

Linear estimation (general)	Forward linear prediction of order m	Backward linear prediction of order m
$\sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)e^*(i) = \mathbf{0}$	$\sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}_m(i-1)f_m^*(i) = \mathbf{0}$	$\sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}_m(i)b_m^*(i) = \mathbf{0}$

The tap-weight vector $\hat{\mathbf{w}}_{f,m}(n)$ satisfies the *principle of orthogonality for forward linear prediction*, which states that

$$\sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}_m(i-1)f_m^*(i) = \mathbf{0}. \tag{16.9}$$

Table 16.2 summarizes this principle for linear least-squares estimation, forward prediction, and backward prediction. The first entry of the table is an exponentially weighted extension of Eq. (9.15) derived in Chapter 9 on the method of least squares. The second entry follows directly from the first one by using the correspondences of Table 16.1.

The tap-weight vector $\hat{\mathbf{w}}_{f,m}(n)$ may also be viewed as the solution obtained by minimizing the sum of weighted forward a posteriori prediction-error squares for $1 \le i \le n$; that is, we minimize

$$\mathcal{F}_m(n) = \sum_{i=1}^{n} \lambda^{n-i}|f_m(i)|^2. \tag{16.10}$$

Equivalently, the tap-weight vector $\mathbf{a}_m(n)$ of the prediction-error filter is the solution to the same minimization problem, subject to the constraint that the first element of $\mathbf{a}_m(n)$ equals unity, in accordance with Eq. (16.6).

Finally, using the definition of Eq. (16.1) in Eq. (16.10), followed by the application of the recursion of Eq. (16.3) and the orthogonality condition of Eq. (16.9), we get the following recursion for updating the minimum value of the sum of weighted forward prediction-error squares (i.e., the forward prediction-error energy):

$$\mathcal{F}_m(n) = \lambda\mathcal{F}_m(n-1) + \eta_m(n)f_m^*(n). \tag{16.11}$$

In Eq. (16.11), the product term $\eta_m(n)f_m^*(n)$ is real valued for all m and n.

16.3 ADAPTIVE BACKWARD LINEAR PREDICTION

Consider next the *backward linear predictor of order m*, depicted in Fig. 16.2(a) for operation at adaptation cycle n. The tap-weight vector $\hat{\mathbf{w}}_{b,m}(n)$ of this predictor is optimized in the least-squares sense over the entire observation interval $1 \le i \le n$. Let

$$b_m(i) = u(i-m) - \hat{\mathbf{w}}_{b,m}^{\mathrm{H}}(n)\mathbf{u}_m(i), \qquad i = 1, 2, \ldots, n \tag{16.12}$$

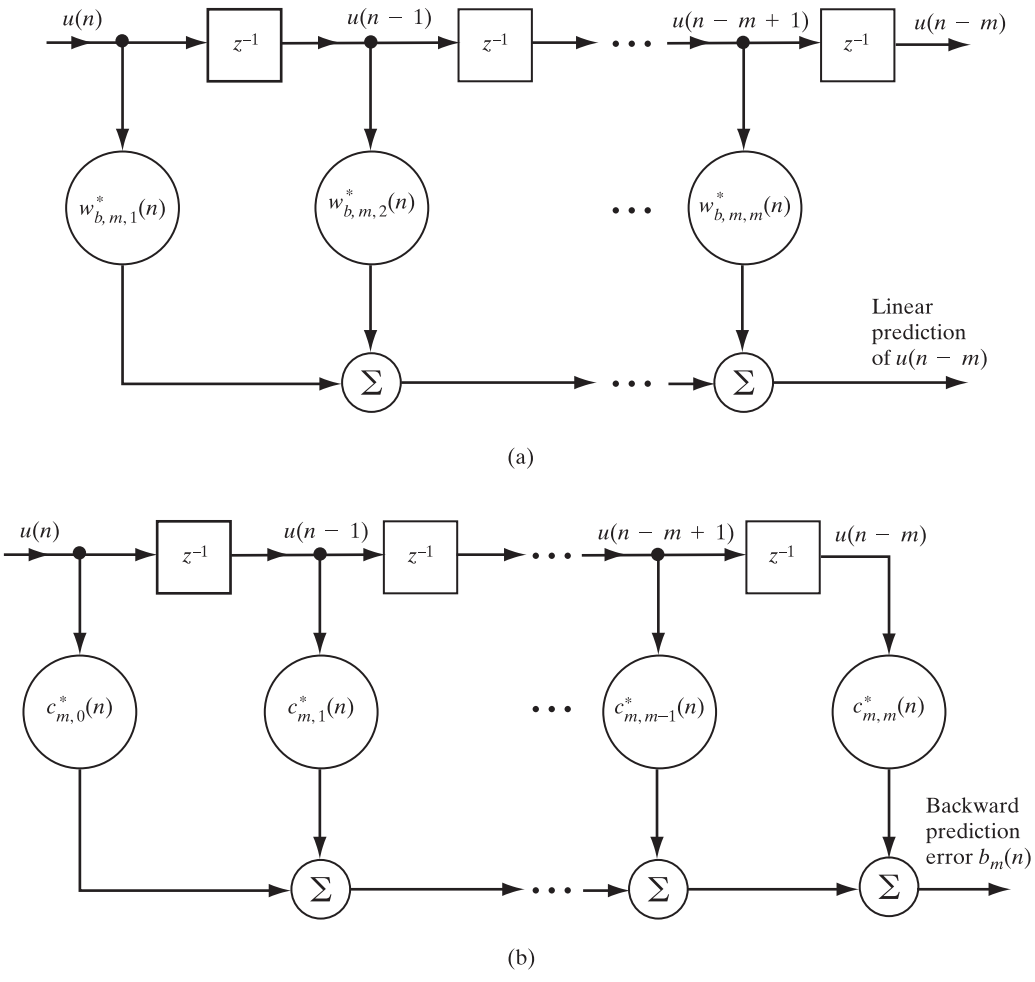


FIGURE 16.2 (a) Backward predictor of order m; (b) corresponding backward prediction-error filter.

denote the backward prediction error produced by the predictor at adaptation cycle i in response to the tap-input vector $\mathbf{u}_m(i)$ of size m. Then, according to this definition, $u(i-m)$ plays the role of the desired response for backward linear prediction, and we have

$$\mathbf{u}_m(i) = [u(i), u(i-1), \ldots, u(i-m+1)]^{\mathrm{T}}$$

and

$$\hat{\mathbf{w}}_{b,m}(n) = [\hat{w}_{b,m,1}(n), \hat{w}_{b,m,2}(n), \ldots, \hat{w}_{b,m,m}(n)]^{\mathrm{T}}.$$

We refer to $b_m(i)$ as the *backward a posteriori prediction error*, since its computation is based on the current value of the backward predictor's tap-weight vector, $\hat{\mathbf{w}}_{b,m}(n)$. Correspondingly, we may define the *backward a priori prediction error* as

$$\beta_m(i) = u(i-m) - \hat{\mathbf{w}}_{b,m}^{\mathrm{H}}(n-1)\mathbf{u}_m(i), \qquad i = 1, 2, \ldots, n, \tag{16.13}$$

the computation of which is based on the past value of the backward predictor's tap-weight vector, $\hat{\mathbf{w}}_{b,m}(n-1)$.

Table 16.1 also lists the correspondences between the quantities characterizing linear estimation in general and those characterizing backward linear prediction in particular. To write the recursions for adaptive backward linear prediction, we may again modify the RLS algorithm developed in Sections 10.3 and 10.4 in light of these correspondences. Thus, we deduce the following recursion for updating the tap-weight vector of the backward predictor:

$$\hat{\mathbf{w}}_{b,m}(n) = \hat{\mathbf{w}}_{b,m}(n-1) + \mathbf{k}_m(n)\beta_m^*(n). \tag{16.14}$$

Here, $\beta_m(n)$ is the backward a priori prediction error defined in Eq. (16.13) for $i = n$, and

$$\mathbf{k}_m(n) = \boldsymbol{\Phi}_m^{-1}(n)\mathbf{u}_m(n) \tag{16.15}$$

is the current value of the *gain vector*, in which the matrix $\boldsymbol{\Phi}_m^{-1}(n)$ is the inverse of the correlation matrix (ignoring the regularization term)

$$\boldsymbol{\Phi}_m(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}_m(i)\mathbf{u}_m^{\mathrm{H}}(i) \tag{16.16}$$

of the input data.

The description of the backward linear prediction problem just presented is in terms of a backward predictor characterized by the tap-weight vector $\hat{\mathbf{w}}_{b,m}(n)$. Equivalently, we may describe the problem in terms of a *backward prediction-error filter*, as depicted in Fig. 16.2(b). Let the prediction-error filter of order m be characterized by a tap-weight vector $\mathbf{c}_m(n)$, which is related to that of the backward predictor in Fig. 16.1(a) by the formula

$$\mathbf{c}_m(n) = \begin{bmatrix} -\hat{\mathbf{w}}_{b,m}(n) \\ 1 \end{bmatrix}, \tag{16.17}$$

where the last element of $\mathbf{c}_m(n)$, namely, $c_{m,m}(n)$, is unity. Thus, with an input vector $\mathbf{u}_{m+1}(i)$ of size $m+1$, the backward a posteriori prediction error and the backward a priori prediction error may be rewritten, respectively, as

$$b_m(i) = \mathbf{c}_m^{\mathrm{H}}(n)\mathbf{u}_{m+1}(i), \qquad i = 1, 2, \ldots, n \tag{16.18}$$

and

$$\beta_m(i) = \mathbf{c}_m^{\mathrm{H}}(n-1)\mathbf{u}_{m+1}(i), \qquad i = 1, 2, \ldots, n. \tag{16.19}$$

In this case, the input vector $\mathbf{u}_{m+1}(i)$ is partitioned in the following way:

$$\mathbf{u}_{m+1}(i) = \begin{bmatrix} \mathbf{u}_m(i) \\ u(i-m) \end{bmatrix}.$$

The tap-weight vector $\hat{\mathbf{w}}_{b,m}(n)$ satisfies the *principle of orthogonality for backward linear prediction*, which states that

$$\sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}_m(i)b_m^*(i) = \mathbf{0}. \tag{16.20}$$

This equation, the last entry in Table 16.2, follows directly from the first entry pertaining to linear least-squares estimation by using the correspondences of Table 16.1.

The tap-weight vector $\hat{\mathbf{w}}_{b,m}(n)$ may also be viewed as the solution obtained by minimizing the sum of weighted backward a posteriori prediction-error squares

$$\mathcal{B}_m(n) = \sum_{i=1}^{n} \lambda^{n-i} |b_m(i)|^2 \qquad \text{for } 1 \le i \le n. \tag{16.21}$$

Equivalently, the tap-weight vector $\mathbf{c}_m(n)$ of the backward prediction-error filter is the solution to the same minimization problem, subject to the constraint that the last element of $\mathbf{c}_m(n)$ equal unity, in accordance with Eq. (16.17).

Using the definition of Eq. (16.12) in Eq. (16.21), followed by the application of the recursion of Eq. (16.14) and the orthogonality condition of Eq. (16.20), we get the following recursion for updating the minimum value of the sum of weighted backward prediction-error squares (i.e., backward prediction-error energy):

$$\mathcal{B}_m(n) = \lambda \mathcal{B}_m(n-1) + \beta_m(n) b_m^*(n). \tag{16.22}$$

In this equation, the product term $\beta_m(n)b_m^*(n)$ is real valued for all m and n.

In closing this discussion of the RLS prediction problem, it is of interest to note that in the case of backward prediction, the input vector $\mathbf{u}_{m+1}(n)$ is partitioned with the desired response $u(n-m)$ as the *last* entry. On the other hand, in the case of forward linear prediction, the input vector $\mathbf{u}_{m+1}(n)$ is partitioned with the desired response $u(n)$ as the *leading* entry. Note also that the update recursion for the tap-weight vector $\hat{\mathbf{w}}_{b,m}(n)$ of the backward linear predictor in Eq. (16.14) requires knowledge of the current value $\mathbf{k}_m(n)$ of the gain vector. On the other hand, the update recursion for the tap-weight vector $\hat{\mathbf{w}}_{f,m}(n)$ of the forward linear predictor in Eq. (16.3) requires knowledge of the old value $\mathbf{k}_m(n-1)$ of the gain vector.

16.4 CONVERSION FACTOR

The definition of the m-by-1 vector

$$\mathbf{k}_m(n) = \boldsymbol{\Phi}_m^{-1}(n)\mathbf{u}_m(n)$$

may also be viewed as the solution of a special case of the normal equations for least-squares estimation. To be specific, the gain vector $\mathbf{k}_m(n)$ defines the tap-weight vector of an FIR filter that contains m taps and that operates on the input data $u(1), u(2), \ldots, u(n)$ to produce the least-squares estimate of a special desired response defined by

$$d(i) = \begin{cases} 1, & i = n \\ 0, & i = 1, 2, \ldots, n-1 \end{cases}. \tag{16.23}$$

The n-by-1 vector whose elements equal the $d(i)$ of Eq. (16.23) is called the *first coordinate vector*. This vector has the property that its inner product with any time-dependent vector reproduces the upper or "most recent" element of that vector.

Substituting Eq. (16.23) into Eq. (10.9), we find that the m-by-1 cross-correlation vector $\mathbf{z}_m(n)$ between the m tap inputs of the FIR filter and the desired response equals

$\mathbf{u}_m(n)$. This therefore confirms the gain vector $\mathbf{k}_m(n)$ as the special solution of the normal equations that arises when the desired response is defined by Eq. (16.23).

For the problem described here, we define the *estimation error*

$$\begin{aligned}\gamma_m(n) &= 1 - \mathbf{k}_m^{\mathrm{H}}(n)\mathbf{u}_m(n) \\ &= 1 - \mathbf{u}_m^{\mathrm{H}}(n)\boldsymbol{\Phi}_m^{-1}(n)\mathbf{u}_m(n).\end{aligned} \tag{16.24}$$

The estimation error $\gamma_m(n)$ represents the output of an FIR filter whose tap-weight vector equals the gain vector $\mathbf{k}_m(n)$ and that is excited by the tap-input vector $\mathbf{u}_m(n)$, as depicted in Fig. 16.3. Since the filter output has the structure of a Hermitian form, it follows that the estimation error $\gamma_m(n)$ is a real-valued scalar. Moreover, $\gamma_m(n)$ has the important property that it is bounded by zero and unity; that is,

$$0 < \gamma_m(n) \leq 1. \tag{16.25}$$

This property is readily proved by substituting the recursion of Eq. (10.16) for the inverse matrix $\boldsymbol{\Phi}_m^{-1}(n)$ into Eq. (16.24) and then simplifying to obtain the result

$$\gamma_m(n) = \frac{1}{1 + \lambda^{-1}\mathbf{u}_m^{\mathrm{H}}(n)\boldsymbol{\Phi}_m^{-1}(n-1)\mathbf{u}_m(n)}. \tag{16.26}$$

The Hermitian form $\mathbf{u}_m^{\mathrm{H}}(n)\boldsymbol{\Phi}_m^{-1}(n-1)\mathbf{u}_m(n) \geq 0$ and $0 < \lambda \leq 1$; consequently, the estimation error $\gamma_m(n)$ is bounded as in Eq. (16.25).

It is noteworthy that $\gamma_m(n)$ also equals the sum of weighted error squares resulting from the use of the FIR filter in Fig. 16.3, whose tap-weight vector equals the gain vector $\mathbf{k}_m(n)$, to obtain the least-squares estimate of the first coordinate vector. (See Problem 1.)

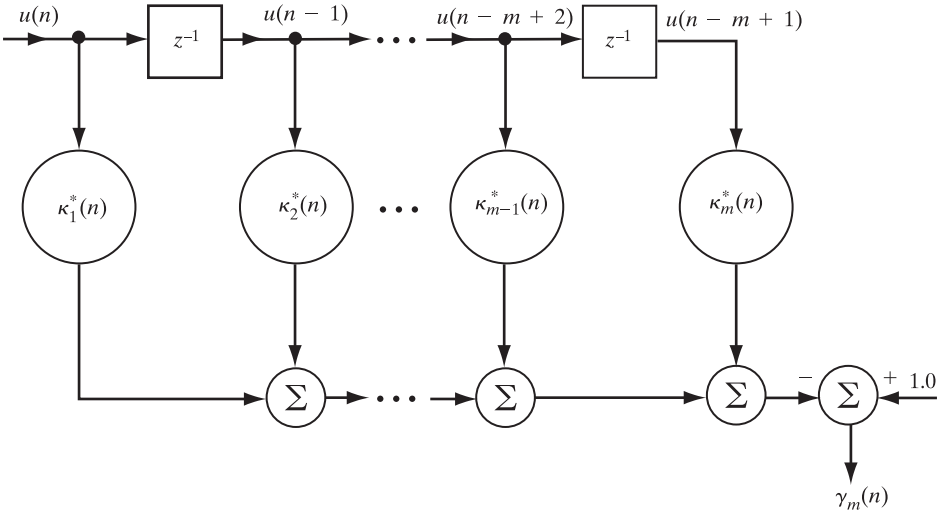


FIGURE 16.3 FIR filter for defining the estimation error $\gamma_m(n)$.

Other Useful Interpretations of $\gamma_m(n)$

Depending on the approach taken, the parameter $\gamma_m(n)$ may be given three other entirely different interpretations:

1. The parameter $\gamma_m(n)$ may be viewed as a *likelihood variable* (Lee et al., 1981). This interpretation follows from a statistical formulation of the tap-input vector in terms of its log-likelihood function, under the assumption that the tap inputs have a joint Gaussian distribution. (See Problem 14.)
2. The parameter $\gamma_m(n)$ may be interpreted as an *angle variable* (Lee et al., 1981; Carayannis et al., 1983). This interpretation follows from Eq. (16.24). In particular, following the discussion presented in Section 15.3, we may express the (positive) square root of $\gamma_m(n)$ as

$$\gamma_m^{1/2}(n) = \cos\phi_m(n),$$

   where $\phi_m(n)$ represents the angle of a plane (Givens) rotation. [See Eq. (15.53).]
3. Finally, the parameter $\gamma_m(n)$ may be interpreted as a *conversion factor* (Carayannis et al., 1983). According to this interpretation, the availability of $\gamma_m(n)$ helps us determine the value of an a posteriori estimation error, given the value of the corresponding a priori estimation error.

It is this third interpretation that we pursue here. Indeed, it is because of that interpretation that we have adopted the terminology "conversion factor" as a description for $\gamma_m(n)$.

Three Kinds of Estimation Error

In linear least-squares estimation theory, there are three kinds of estimation error to be considered: the ordinary estimation error (involved in the estimation of some desired response), the forward prediction error, and the backward prediction error. Correspondingly, $\gamma_m(n)$ has three useful interpretations as a conversion factor:

1. For RLS estimation,

$$\gamma_m(n) = \frac{e_m(n)}{\xi_m(n)}, \tag{16.27}$$

   where $e_m(n)$ is the a posteriori estimation error and $\xi_m(n)$ is the a priori estimation error. Equation (16.27) states that, given the a priori estimation error $\xi_m(n)$, we may determine the corresponding value of the a posteriori estimation error $e_m(n)$ by multiplying $\xi_m(n)$ by $\gamma_m(n)$. We may therefore view $\xi_m(n)$ as a tentative value of the estimation error $e_m(n)$ and $\gamma_m(n)$ as the multiplicative correction.
2. For adaptive forward linear prediction,

$$\gamma_m(n-1) = \frac{f_m(n)}{\eta_m(n)}. \tag{16.28}$$

Equation (16.28) states that, given the forward a priori prediction error $\eta_m(n)$, we may compute the forward a posteriori prediction error $f_m(n)$ by multiplying $\eta_m(n)$ by the delayed estimation error $\gamma_m(n-1)$. We may therefore view $\eta_m(n)$ as a tentative value for the forward a posteriori prediction error $f_m(n)$ and $\gamma_m(n-1)$ as the multiplicative correction.

3. Lastly, for adaptive backward linear prediction,

$$\gamma_m(n) = \frac{b_m(n)}{\beta_m(n)}. \tag{16.29}$$

Equation (16.29) states that, given the backward a priori prediction error $\beta_m(n)$, we may compute the backward a posteriori prediction error $b_m(n)$ by multiplying $\beta_m(n)$ by the estimation error $\gamma_m(n)$. We may therefore view $\beta_m(n)$ as a tentative value for the backward prediction error $b_m(n)$ and $\gamma_m(n)$ as the multiplicative correction.

For proofs of these three points, the reader is referred to Problem 6.

The foregoing discussion teaches that the unique role of the variable $\gamma_m(n)$: It is the *common* factor (either in its regular or delayed form) in the conversion of an a priori estimation error into the corresponding a posteriori estimation error, be it in the context of ordinary estimation, forward prediction, or backward prediction. Accordingly, we may refer to $\gamma_m(n)$ as a *conversion factor*. Indeed, it is remarkable that, through the use of this conversion factor, we are able to compute the a posteriori errors $e_m(n)$, $f_m(n)$, and $b_m(n)$ at adaptation cycle n before the tap-weight vectors of the pertinent filters that produce them have actually been computed (Carayannis et al., 1983).

16.5 LEAST-SQUARES LATTICE (LSL) PREDICTOR

Returning to the time-shifting property of the input data, we note from the partitioned vector

$$\mathbf{u}_{m+1}(n) = \begin{bmatrix} \mathbf{u}_m(n) \\ u(n-m) \end{bmatrix}$$

that the input vector $\mathbf{u}_m(n)$ for a backward linear predictor of order $m-1$ and the input vector $\mathbf{u}_{m+1}(n)$ for a backward linear predictor of order m have exactly the same first $m-1$ entries. Likewise, we note from the partitioned vector

$$\mathbf{u}_{m+1}(n) = \begin{bmatrix} u(n) \\ \mathbf{u}_m(n-1) \end{bmatrix}$$

that the input vector $\mathbf{u}_m(n-1)$ for a forward linear predictor of order $m-1$ and the input vector $\mathbf{u}_{m+1}(n)$ for a forward linear predictor of order m have exactly the same $m-1$ last entries. These observations prompt us to raise the following fundamental question:

In the course of increasing the prediction order from, say, $m-1$ to m, is it possible to carry over information gathered from previous computations pertaining to the prediction order $m-1$?

The answer to this question is an emphatic yes, and it is embodied in a modular filtering structure known as the *least-squares lattice (LSL) predictor*.

To derive this important filtering structure and its algorithmic design, we propose to proceed as follows: In this section, we use the *principle of orthogonality* to derive the basic equations that characterize the LSL predictor. Then, under the unifying umbrella of *Kalman filter theory*, we derive various algorithms for the design of the LSL predictor in subsequent sections of the chapter.

To begin with, consider the situation depicted in Fig. 16.4, involving a pair of forward and backward prediction-error filters of order $m - 1$. Both filters are fed by the same input vector $\mathbf{u}_m(i)$. The forward prediction-error filter, characterized by the tap-weight vector $\mathbf{a}_{m-1}(n)$, produces $f_{m-1}(i)$ at its output. The backward prediction-error filter, characterized by the tap-weight vector $\mathbf{c}_{m-1}(n)$, produces $b_{m-1}(i)$ at its output. The input data $u(i)$ occupy the observation interval $1 \le i \le n$. The next issue we wish to address may be stated as follows:

> Given the forward prediction error $f_{m-1}(i)$ and the backward prediction error $b_{m-1}(i)$, determine their order-updated values $f_m(i)$ and $b_m(i)$, respectively, in a computationally efficient manner.

By "computationally efficient," we mean the following: The input vector in the figure is enlarged by adding the past sample $u(i - m)$, and the prediction order is thereby increased by one, yet the computations involved in evaluating $f_{m-1}(i)$ and $b_{m-1}(i)$ remain completely intact.

Forward Linear Prediction

The forward prediction error $f_{m-1}(i)$ is determined by the tap inputs $u(i), u(i-1), \ldots, u(i-m+1)$. The order-updated forward prediction error $f_m(i)$ requires knowledge of the additional tap input (i.e., past sample) $u(i-m)$. The backward prediction error $b_{m-1}(i)$ is determined by the same tap inputs as those involved in $f_{m-1}(i)$. If, therefore, we were to delay $b_{m-1}(i)$ by one adaptation cycle, the additional past sample $u(i-m)$ needed for computing $f_m(i)$ would be found in the composition of the *delayed* backward prediction

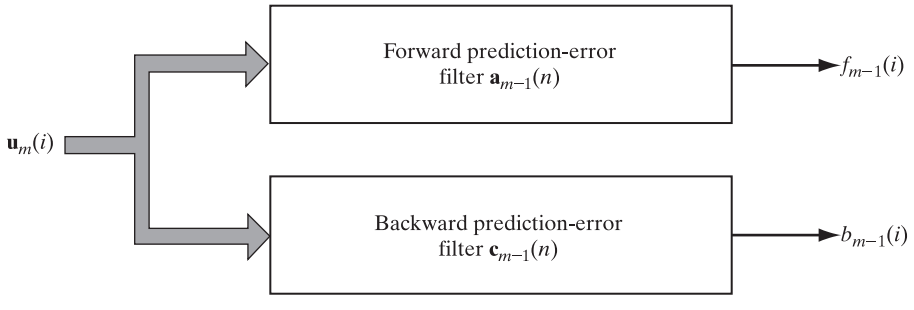


FIGURE 16.4 Setting the stage for formulating the LSL predictor.

error $b_{m-1}(i-1)$. Thus, treating $b_{m-1}(i-1)$ as the input to a *one-tap least-squares filter*, $f_{m-1}(i)$ as the desired response, and $f_m(i)$ as the residual resulting from the least-squares estimation, we may write (see Fig. 16.5(a))

$$f_m(i) = f_{m-1}(i) + \kappa_{f,m}^*(n) b_{m-1}(i-1), \qquad i = 1, 2, \ldots, n, \tag{16.30}$$

where $\kappa_{f,m}(n)$ is the filter's scalar coefficient. Note that, in accordance with the method of least squares, the coefficient $\kappa_{f,m}(n)$ is maintained constant throughout the observation interval extending from $i = 1$ to $i = n$. The format of Eq. (16.30) is similar to that of the corresponding order update derived in Chapter 3 for a lattice predictor operating on stationary inputs. However, the formula for $\kappa_{f,m}(n)$ is different. For the determination of that coefficient, we turn to the principle of orthogonality summarized in Table 16.2 for the three basic forms of linear least-squares estimation. According to this principle, the estimation error (i.e., the residual) produced by a linear least-squares filter in response to a set of inputs is orthogonal to each of those inputs (in a time-averaged sense) over the entire observation interval of interest. Thus, adapting the second entry in Table 16.2 to the forward prediction framework depicted in Fig. 16.5(a) with $b_{m-1}(i-1)$ as the input and $f_m(i)$ viewed as the residual, we may write

$$\sum_{i=1}^{n} \lambda^{n-i} b_{m-1}(i-1) f_m^*(i) = 0. \tag{16.31}$$

Substituting Eq. (16.30) into Eq. (16.31) and then solving for $\kappa_{f,m}(n)$, we obtain

$$\kappa_{f,m}(n) = -\frac{\sum_{i=1}^{n} \lambda^{n-i} b_{m-1}(i-1) f_{m-1}^*(i)}{\sum_{i=1}^{n} \lambda^{n-i} |b_{m-1}(i-1)|^2}. \tag{16.32}$$

The denominator of this formula is the sum of weighted backward prediction-error squares for order $m-1$; that is,

$$\begin{aligned} \mathcal{B}_{m-1}(n-1) &= \sum_{i=1}^{n-1} \lambda^{n-1-i} |b_{m-1}(i)|^2 \\ &= \sum_{i=1}^{n} \lambda^{n-i} |b_{m-1}(i-1)|^2, \end{aligned} \tag{16.33}$$

where, in the last line, we have used the fact that

$$b_{m-1}(0) = 0 \quad \text{for all } m \geq 1$$

by virtue of prewindowing the input data. For the numerator of Eq. (16.32), we introduce the definition of an *exponentially weighted cross-correlation* between forward and backward prediction errors:

$$\Delta_{m-1}(n) = \sum_{i=1}^{n} \lambda^{n-i} b_{m-1}(i-1) f_{m-1}^*(i). \tag{16.34}$$

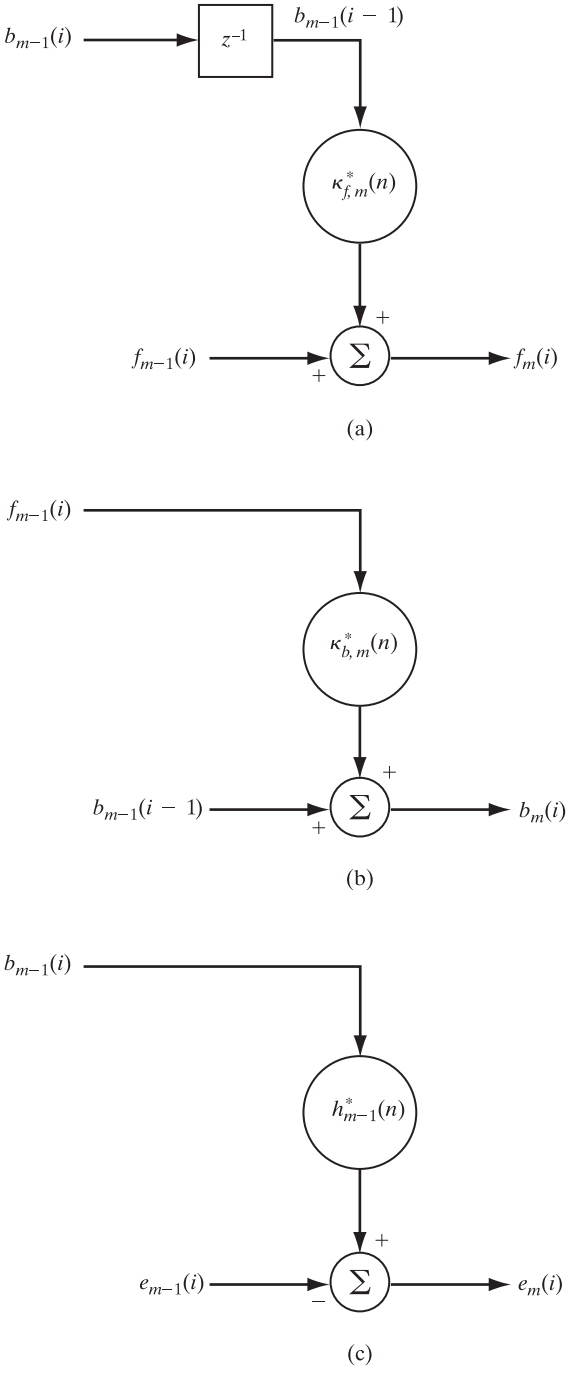


FIGURE 16.5 Single-coefficient linear combiners for (a) forward prediction, (b) backward prediction, and (c) joint-process estimation.

Using the definitions of Eqs. (16.33) and (16.34) in Eq. (16.32), we see that the formula for the scalar coefficient $\kappa_{f,m}(n)$ takes on the compact form

$$\kappa_{f,m}(n) = -\frac{\Delta_{m-1}(n)}{\mathcal{B}_{m-1}(n-1)}. \tag{16.35}$$

Linear Backward Prediction

Consider next the issue of computing the order-updated backward prediction error $b_m(i)$. As before, we may work with the forward prediction error $f_{m-1}(i)$ and delayed backward prediction error $b_{m-1}(i-1)$, except for the fact that their filtering roles are now interchanged. Specifically, we have a one-tap least-squares filter with $f_{m-1}(i)$ acting as the input, $b_{m-1}(i-1)$ as the desired response, and $b_m(i)$ as the residual of the filtering process, as depicted in Fig. 16.5(b); hence, we may write

$$b_m(i) = b_{m-1}(i-1) + \kappa_{b,m}^*(n) f_{m-1}(i), \qquad i = 1, 2, \ldots, n, \tag{16.36}$$

where $\kappa_{b,m}(n)$ is the filter's scalar coefficient. Here again, in accordance with the method of least squares, the coefficient $\kappa_{b,m}(n)$ is maintained constant throughout the observation interval extending from $i = 1$ to $i = n$. The format of Eq. (16.36) is also similar to the corresponding order update derived in Chapter 3 for a lattice predictor operating on stationary inputs. However, the formula for $\kappa_{b,m}(n)$ is different. In particular, we determine $\kappa_{b,m}(n)$ by applying the principle of orthogonality to the input $f_{m-1}(i)$ and residual $b_m(i)$ in the backward prediction problem postulated in Eq. (16.36). Thus, adapting the third entry in Table 16.2 to the backward prediction framework depicted in Fig. 16.5(b), we may write

$$\sum_{i=1}^{n} \lambda^{n-i} f_{m-1}(i) b_m^*(i) = 0. \tag{16.37}$$

Substituting Eq. (16.36) into Eq. (16.37) and then solving for $\kappa_{b,m}(n)$, we get

$$\kappa_{b,m}(n) = -\frac{\sum_{i=1}^{n} \lambda^{n-i} f_{m-1}(i) b_{m-1}^*(i-1)}{\sum_{i=1}^{n} \lambda^{n-i} |f_{m-1}(i)|^2}. \tag{16.38}$$

The numerator of this formula is the complex conjugate of the quantity $\Delta_{m-1}(n)$ defined in Eq. (16.34). The denominator is recognized as the sum of weighted forward prediction-error squares for order $m-1$:

$$\mathcal{F}_{m-1}(n) = \sum_{i=1}^{n} \lambda^{n-i} |f_{m-1}(i)|^2. \tag{16.39}$$

Accordingly, we may recast Eq. (16.38) into the compact form

$$\kappa_{b,m}(n) = -\frac{\Delta_{m-1}^*(n)}{\mathcal{F}_{m-1}(n)}. \tag{16.40}$$

Projection Theory

The results described in Eqs. (16.30) and (16.36) are basic to the LSL predictor. For their physical interpretation, we define the n-by-1 prediction-error vectors

$$\mathbf{f}_m(n) = [f_m(1), f_m(2), \ldots, f_m(n)]^{\mathrm{T}},$$
$$\mathbf{b}_m(n) = [b_m(1), b_m(2), \ldots, b_m(n)]^{\mathrm{T}},$$

and

$$\mathbf{b}_m(n-1) = [0, b_m(1), \ldots, b_m(n-1)]^{\mathrm{T}},$$

where the prediction order $m = 0, 1, 2, \ldots, M$. Then, on the basis of Eqs. (16.30) and (16.36) we may make the following two statements in the terminology of *projection theory*:

1. The result of projecting the vector $\mathbf{b}_{m-1}(n-1)$ onto $\mathbf{f}_{m-1}(n)$ is represented by the residual vector $\mathbf{f}_m(n)$; the *forward reflection coefficient* $\kappa_{f,m}(n)$ is the parameter needed to do this projection.
2. The result of projecting the vector $\mathbf{f}_{m-1}(n)$ onto $\mathbf{b}_{m-1}(n-1)$ is represented by the residual vector $\mathbf{b}_m(n)$; the *backward reflection coefficient* $\kappa_{b,m}(n)$ is the parameter needed to do this second projection.

To put a finishing touch on this part of the discussion, we now expand Eqs. (16.30) and (16.36) for the end of the observation interval $i = n$. We thus have the pair of interrelated order-update recursions, first for forward linear prediction,

$$f_m(n) = f_{m-1}(n) + \kappa^*_{f,m}(n) b_{m-1}(n-1) \tag{16.41}$$

and, second for backward linear prediction,

$$b_m(n) = b_{m-1}(n-1) + \kappa^*_{b,m}(n) f_{m-1}(n), \tag{16.42}$$

where $m = 1, 2, \ldots, M$, and M is the *final prediction order*. When $m = 0$, no prediction is being performed on the input data; this corresponds to the *initial conditions* described by

$$f_0(n) = b_0(n) = u(n), \tag{16.43}$$

where $u(n)$ is the input datum at adaptation cycle n. Thus, as we vary the prediction order m from zero all the way up to the final value M, we get the M-stage LSL predictor of Fig. 16.6. An important feature of the LSL predictor described herein is its *modular* structure, which implies that the computational complexity scales *linearly* with the prediction order.

LSL Version of the Levinson–Durbin Recursion

The forward prediction error $f_m(n)$ and backward prediction error $b_m(n)$ are defined by Eqs. (16.7) and (16.18), reproduced here for $i = n$ as

$$f_m(n) = \mathbf{a}_m^{\mathrm{H}}(n)\mathbf{u}_{m+1}(n)$$

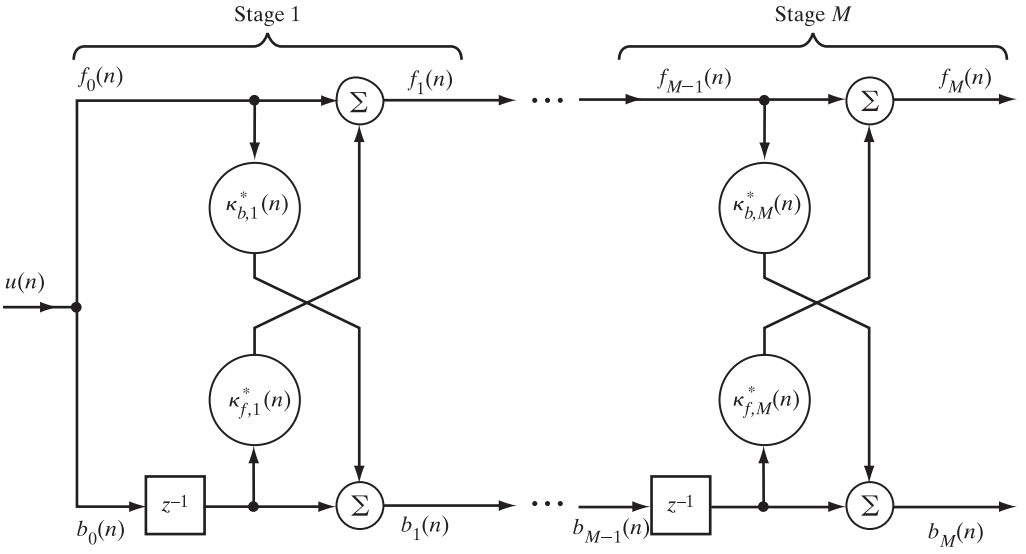


FIGURE 16.6 M-stage lattice predictor.

and

$$b_m(n) = \mathbf{c}_m^{\mathrm{H}}(n)\mathbf{u}_{m+1}(n),$$

respectively, where $\mathbf{a}_m(n)$ and $\mathbf{c}_m(n)$ are the tap-weight vectors of the corresponding forward and backward prediction-error filters, respectively. The forward prediction error $f_{m-1}(n)$ and delayed backward prediction error $b_{m-1}(n-1)$, pertaining to a lower prediction order, are defined as follows:

$$\begin{aligned} f_{m-1}(n) &= \mathbf{a}_{m-1}^{\mathrm{H}}(n)\mathbf{u}_m(n) \\ &= \begin{bmatrix} \mathbf{a}_{m-1}(n) \\ 0 \end{bmatrix}^{\mathrm{H}} \begin{bmatrix} \mathbf{u}_m(n) \\ u(n-m) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{a}_{m-1}(n) \\ 0 \end{bmatrix}^{\mathrm{H}} \mathbf{u}_{m+1}(n); \\ b_{m-1}(n-1) &= \mathbf{c}_{m-1}^{\mathrm{H}}(n-1)\mathbf{u}_m(n-1) \\ &= \begin{bmatrix} 0 \\ \mathbf{c}_{m-1}(n-1) \end{bmatrix}^{\mathrm{H}} \begin{bmatrix} u(n) \\ \mathbf{u}_m(n-1) \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ \mathbf{c}_{m-1}(n-1) \end{bmatrix}^{\mathrm{H}} \mathbf{u}_{m+1}(n). \end{aligned}$$

The two prediction errors just defined share a common input vector, namely, $\mathbf{u}_{m+1}(n)$. Therefore, substituting their defining equations into Eqs. (16.41) and (16.42) and then comparing terms on the two sides of the resultants, we deduce the pair of order updates

$$\mathbf{a}_m(n) = \begin{bmatrix} \mathbf{a}_{m-1}(n) \\ 0 \end{bmatrix} + \kappa_{f,m}(n) \begin{bmatrix} 0 \\ \mathbf{c}_{m-1}(n-1) \end{bmatrix} \tag{16.44}$$

and

$$\mathbf{c}_m(n) = \begin{bmatrix} 0 \\ \mathbf{c}_{m-1}(n) \end{bmatrix} + \kappa_{b,m}(n) \begin{bmatrix} \mathbf{a}_{m-1}(n) \\ 0 \end{bmatrix}, \tag{16.45}$$

where $m = 1, 2, \ldots, M$. Equations (16.44) and (16.45) may be viewed as the least-squares version of the *Levinson–Durbin recursion* discussed in Chapter 3 for stationary inputs. Recognizing that, by definition, the last element of $\mathbf{c}_{m-1}(n-1)$ and the first element of $\mathbf{a}_{m-1}(n)$ equal unity, we readily find from Eqs. (16.44) and (16.45) that

$$\kappa_{f,m}(n) = a_{m,m}(n) \tag{16.46}$$

and

$$\kappa_{b,m}(n) = c_{m,0}(n), \tag{16.47}$$

where $a_{m,m}(n)$ is the last element of the vector $\mathbf{a}_m(n)$ and $c_{m,0}(n)$ is the first element of the vector $\mathbf{c}_m(n)$. Thus, unlike the situation described in Chapter 3 for a stationary environment, we generally find that in an LSL predictor,

$$\kappa_{f,m}(n) \neq \kappa_{b,m}^*(n).$$

In any event, the order updates of Eqs. (16.44) and (16.45) reveal a remarkable property of an LSL predictor of final order M:

> In an implicit sense, such a predictor embodies a chain of forward prediction-error filters of order $1, 2, \ldots, M$ and a chain of backward prediction-error filters of order $1, 2, \ldots, M$, all in one modular structure, as shown in Fig. 16.6.

Time-Update Recursion for $\Delta_{m-1}(n)$

From Eqs. (16.35) and (16.40), we see that the reflection coefficients $\kappa_{f,m}(n)$ and $\kappa_{b,m}(n)$ of the LSL predictor are uniquely determined by three quantities: $\Delta_{m-1}(n)$, $\mathcal{F}_{m-1}(n)$, and $\mathcal{B}_{m-1}(n-1)$. Equations (16.11) and (16.22) provide the means to time-update the last of these two quantities. We need the corresponding time update for computing $\Delta_{m-1}(n)$ for recursive computation of the two reflection coefficients.

To proceed with the derivation of this remaining recursion, we recall the following two equations from Section 16.2 on forward linear prediction, with $m-1$ written in place of m:

$$f_{m-1}(i) = u(i) - \hat{\mathbf{w}}_{f,m-1}^{\mathrm{H}}(n)\mathbf{u}_{m-1}(i-1), \qquad i = 1, 2, \ldots, n,$$

and

$$\hat{\mathbf{w}}_{f,m-1}(n) = \hat{\mathbf{w}}_{f,m-1}(n-1) + \mathbf{k}_{m-1}(n-1)\eta_{m-1}^*(n).$$

Substituting these two equations into Eq. (16.34), we get (after rearranging terms)

$$\begin{aligned} \Delta_{m-1}(n) = {} & \sum_{i=1}^{n} \lambda^{n-i}[u(i) - \hat{\mathbf{w}}_{f,m-1}^{\mathrm{H}}(n-1)\mathbf{u}_{m-1}(i-1)]^* b_{m-1}(i-1) \\ & - \eta_{m-1}(n)\mathbf{k}_{m-1}^{\mathrm{T}}(n-1) \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}_{m-1}^*(i-1)b_{m-1}(i-1). \end{aligned}$$

This equation simplifies as follows:

- The second term on the right-hand side of the equation is zero by virtue of the principle of orthogonalization for backward linear prediction, which states that

$$\sum_{i=1}^{n} \lambda^{n-i} \mathbf{u}_{m-1}(i-1) b_{m-1}^{*}(i-1) = \mathbf{0}.$$

  [See Eq. (16.20) and the third entry in Table 16.2.]
- The expression inside the square brackets of the first term on the right-hand side of the equation is, by definition, the a priori forward prediction error:

$$\eta_{m-1}(i) = u(i) - \hat{\mathbf{w}}_{f,m-1}^{\mathrm{H}}(n-1)\mathbf{u}_{m-1}(i-1), \qquad i = 1, 2, \ldots, n.$$

  [See Eq. (16.2).]

Accordingly, we may redefine $\Delta_{m-1}(n)$ simply as

$$\Delta_{m-1}(n) = \sum_{i=1}^{n} \lambda^{n-i} \eta_{m-1}^{*}(i) b_{m-1}(i-1). \tag{16.48}$$

Next, we isolate from the summation the term $\eta_{m-1}^{*}(n) b_{m-1}(n-1)$ corresponding to $i = n$, and thus write

$$\begin{aligned}\Delta_{m-1}(n) &= \sum_{i=1}^{n-1} \lambda^{n-i} \eta_{m-1}^{*}(i) b_{m-1}(i-1) + \eta_{m-1}^{*}(n) b_{m-1}(n-1) \\ &= \lambda \sum_{i=1}^{n-1} \lambda^{n-1-i} \eta_{m-1}^{*}(i) b_{m-1}(i-1) + \eta_{m-1}^{*}(n) b_{m-1}(n-1).\end{aligned}$$

The summation term on the right-hand side of this equation is recognized to be simply the past value $\Delta_{m-1}(n-1)$. Hence, we may finally write

$$\Delta_{m-1}(n) = \lambda \Delta_{m-1}(n-1) + \eta_{m-1}^{*}(n) b_{m-1}(n-1), \tag{16.49}$$

which is the desired recursion. Note that Eq. (16.49) is similar to Eq. (16.11) for $\mathcal{F}_m(n)$ and, likewise, Eq. (16.22) for $\mathcal{B}_m(n)$ in that in each of these three updates, the correction term involves the product of a posteriori and a priori prediction errors.

Exact Decoupling Property of the LSL Predictor

Another important property of an LSL predictor consisting of m stages is that the backward prediction errors $b_0(n), b_1(n), \ldots, b_m(n)$ produced at the various stages of the predictor are *uncorrelated* with (orthogonal to) each other in a time-averaged sense at all adaptation cycles. In other words, the LSL predictor transforms a correlated input data sequence $\{u(n), u(n-1), \ldots, u(n-m)\}$ into the new sequence of uncorrelated backward prediction errors (i.e., innovations):

$$\{u(n), u(n-1), \ldots, u(n-m)\} \rightleftharpoons \{b_0(n), b_1(n), \ldots, b_m(n)\}. \tag{16.50}$$

The transformation shown here is *reciprocal*, which means that the LSL predictor preserves the full information content of the input data.

Now, consider a backward prediction-error filter of order m. Let the $(m+1)$-by-1 tap-weight vector of the filter, optimized in the least-squares sense over adaptation cycles $1 \le i \le n$, be denoted by $\mathbf{c}_m(n)$. In expanded form,

$$\mathbf{c}_m(n) = [c_{m,m}(n), c_{m,m-1}(n), \ldots, 1]^{\mathrm{T}}.$$

Let $b_m(i)$ denote the backward a posteriori prediction error produced at the output of the filter in response to the $(m+1)$-by-1 input vector $\mathbf{u}_{m+1}(i)$, which, in expanded form, is

$$\mathbf{u}_{m+1}(i) = [u(i), u(i-1), \ldots, u(i-m)]^{\mathrm{T}}, \qquad i > m.$$

We may thus express the error $b_m(i)$ as

$$\begin{aligned} b_m(i) &= \mathbf{c}_m^{\mathrm{H}}(n)\mathbf{u}_{m+1}(i) \\ &= \sum_{k=0}^{m} c_{m,k}^{*}(n)u(i-m+k), \qquad \begin{matrix} m < i \le n \\ m = 0, 1, 2, \ldots \end{matrix} \end{aligned} \tag{16.51}$$

Let

$$\mathbf{b}_{m+1}(i) = [b_0(i), b_1(i), \ldots, b_m(i)]^{\mathrm{T}}, \qquad \begin{matrix} m < i \le n \\ m = 0, 1, 2, \ldots \end{matrix}$$

be an $(m+1)$-by-1 backward a posteriori prediction-error vector. Substituting Eq. (16.51) into the vector $\mathbf{b}_{m+1}(i)$, we may express the transformation of the input data into the corresponding set of backward a posteriori prediction errors as

$$\mathbf{b}_{m+1}(i) = \mathbf{L}_m(n)\mathbf{u}_{m+1}(i), \tag{16.52}$$

where the $(m+1)$-by-$(m+1)$ *transformation matrix*

$$\mathbf{L}_m(n) = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ c_{1,1}^{*}(n) & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ c_{m,m}^{*}(n) & c_{m,m-1}^{*} & \cdots & 1 \end{bmatrix} \tag{16.53}$$

is a lower triangular matrix. Note that the subscript m in the symbol $\mathbf{L}_m(n)$ refers to the highest order of backward prediction-error filter involved in the matrix constitution. Note also the following points:

- The nonzero elements of row l of matrix $\mathbf{L}_m(n)$ are defined by the tap weights of a backward prediction-error filter of order $(l-1)$.
- The diagonal elements of matrix $\mathbf{L}_m(n)$ equal unity; this follows from the fact that the last tap weight of every backward prediction-error filter equals unity.
- The determinant of matrix $\mathbf{L}_m(n)$ equals unity for all m; hence, the inverse matrix $\mathbf{L}_m^{-1}(n)$ exists, and the reciprocal nature of the transformation in Eq. (16.50) is confirmed.

By definition, the correlation between the backward prediction errors pertaining to different orders—k and m, say—is given by the exponentially weighted time average

$$\begin{aligned}\phi_{km}(n) &= \sum_{i=1}^{n} \lambda^{n-i} b_k(i) b_m^*(i) \\ &= \sum_{i=1}^{n} \lambda^{n-i} \mathbf{c}_k^{\mathrm{H}}(n) \mathbf{u}_k(i) b_m^*(i) \\ &= \mathbf{c}_k^{\mathrm{H}}(n) \sum_{i=1}^{n} \lambda^{n-i} \mathbf{u}_k(i) b_m^*(i),\end{aligned} \tag{16.54}$$

where $\mathbf{c}_k(n)$ is the tap-weight vector of the backward prediction-error filter of order k that is responsible for generating the error $b_k(n)$. Without loss of generality, we may assume that $m > k$. Then, recognizing that the elements of the input vector $\mathbf{u}_k(n)$ are involved in generating the backward prediction represented by the error $b_m(n)$, optimized in the least-squares sense, we readily deduce from the principle of orthogonality that the correlation $\phi_{km}(n)$ is zero for $m > k$. In other words, for $m \neq k$, the backward prediction errors $b_k(n)$ and $b_m(n)$ are uncorrelated with each other in a time-averaged sense.

This remarkable property makes the LSL predictor an ideal device for *exact least-squares joint-process estimation*. Specifically, we may exploit the sequence of backward prediction errors produced by the lattice structure of Fig. 16.6 to perform the least-squares estimation of a desired response in the order-recursive manner depicted in Fig. 16.7. In particular, for order (stage) m, we may write

$$e_m(n) = e_{m-1}(n) - h_{m-1}^*(n) b_{m-1}(n), \qquad m = 1, 2, \ldots, M+1. \tag{16.55}$$

For the *initial condition* of joint-process estimation, we have

$$e_0(n) = d(n). \tag{16.56}$$

The parameters $h_{m-1}(n)$, $m = 1, 2, \ldots, M+1$ are called *joint-process estimation* or *regression coefficients*. Thus, the least-squares estimation of the desired response $d(n)$ may proceed on a stage-by-stage basis, jointly with the linear prediction process.

Equation (16.55) represents a *single-order linear combiner*, as depicted in Fig. 16.5(c), where we have purposely used the adaptation cycle i in place of n for the estimation variables in order to be consistent with the notations used in parts (a) and (b) of the figure. The point to note here is that $b_{m-1}(i)$ may be viewed as the input and $e_{m-1}(i)$ as the desired response for $1 \leq i \leq n$.

16.6 ANGLE-NORMALIZED ESTIMATION ERRORS

The formulation of the LSL predictor presented in the previous section was based on the forward a posteriori prediction error $f_m(n)$ and the backward a posteriori prediction error $b_m(n)$. The resulting order-recursive relations of Eqs. (16.41) and (16.42) are defined in terms of the current value of the forward reflection coefficient $\kappa_{f,m}(n)$ and

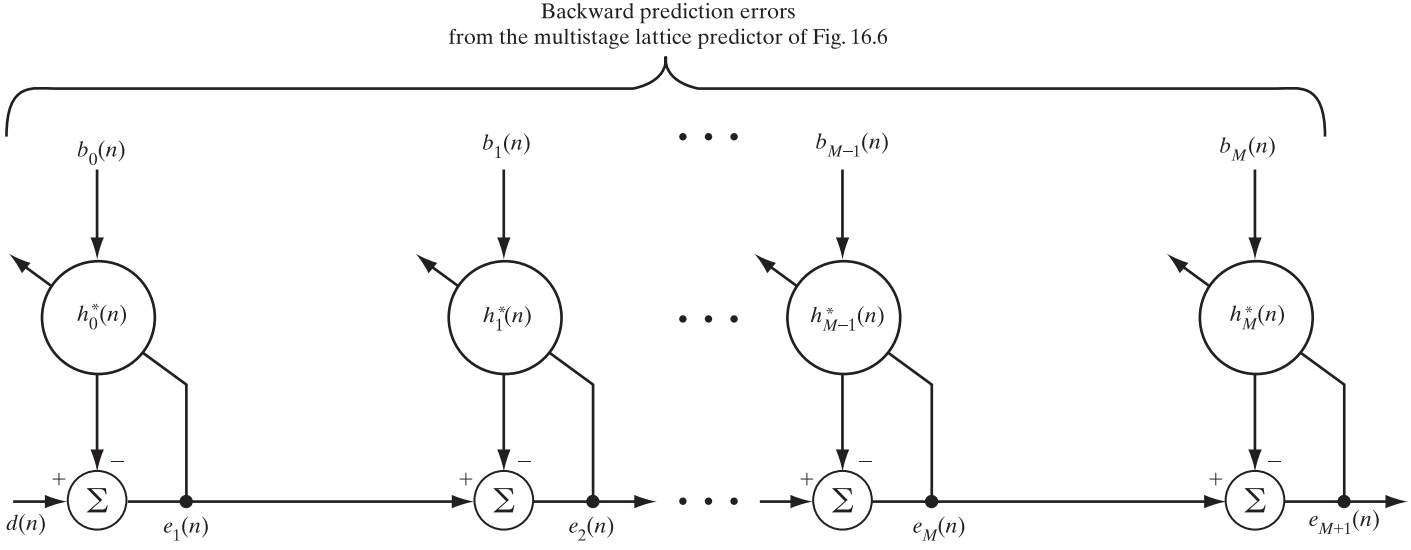


FIGURE 16.7 Least-squares estimation using a sequence of backward prediction errors.

the current value of the backward reflection coefficient $\kappa_{b,m}(n)$. We may equally well formulate the LSL predictor in terms of the forward a priori prediction error $\eta_m(n)$ and the backward a priori prediction error $\beta_m(n)$. In the latter case, the order-recursive relations are defined in terms of the past value of the forward reflection coeffcient $\kappa_{f,m}(n-1)$ and the past value of the backward reflection coefficient $\kappa_{b,m}(n-1)$, as shown here:

$$\eta_m(n) = \eta_{m-1}(n) + \kappa^*_{f,m}(n-1)\beta_{m-1}(n-1); \tag{16.57}$$

$$\beta_m(n) = \beta_{m-1}(n-1) + \kappa^*_{b,m}(n-1)\eta_{m-1}(n). \tag{16.58}$$

From a developmental point of view, it would be highly desirable to formulate the LSL prediction problem in a way that is invariant to the choice of a posteriori or a priori prediction errors. This objective is attained by introducing the notion of *angle-normalized estimation errors*. In this regard, with three different forms of estimation in mind, we introduce the following set of angle-normalized estimation errors for a LSL predictor of order m:

- *The angle-normalized forward prediction error*, defined by

$$\varepsilon_{f,m}(n) = \gamma_m^{1/2}(n-1)\eta_m(n) = \frac{f_m(n)}{\gamma_m^{1/2}(n-1)}, \tag{16.59}$$

  where $\gamma_m(n-1)$ is the past value of the conversion factor.
- *The angle-normalized backward prediction error*, defined by

$$\varepsilon_{b,m}(n) = \gamma_m^{1/2}(n)\beta_m(n) = \frac{b_m(n)}{\gamma_m^{1/2}(n)}, \tag{16.60}$$

  where $\gamma_m(n)$ is the current value of the conversion factor.
- *The angle-normalized joint-process estimation error*, defined by

$$\varepsilon_m(n) = \gamma_m^{1/2}(n)\xi_m(n) = \frac{e_m(n)}{\gamma_m^{1/2}(n)}, \tag{16.61}$$

  where $e_m(n)$ and $\xi_m(n)$ are respectively the a posteriori and a priori values of the joint-process estimation error.

Accordingly, we may reformulate the three single-coefficient linear combiners of Fig. 16.5 for forward prediction, backward prediction, and joint-process estimation in terms of angle-normalized variables, as shown in Fig. 16.8.

The term "angle" used in the definitions presented herein refers to an interpretation of the conversion factor as the cosine of an angle. (See Section 16.4 for more details.) In any event, the important point to note here is that in basing the algorithmic development of LSL filtering on angle-normalized estimation errors, we no longer have to distinguish between the a posteriori and a priori versions of the different estimation errors, hence the generality of the approach.

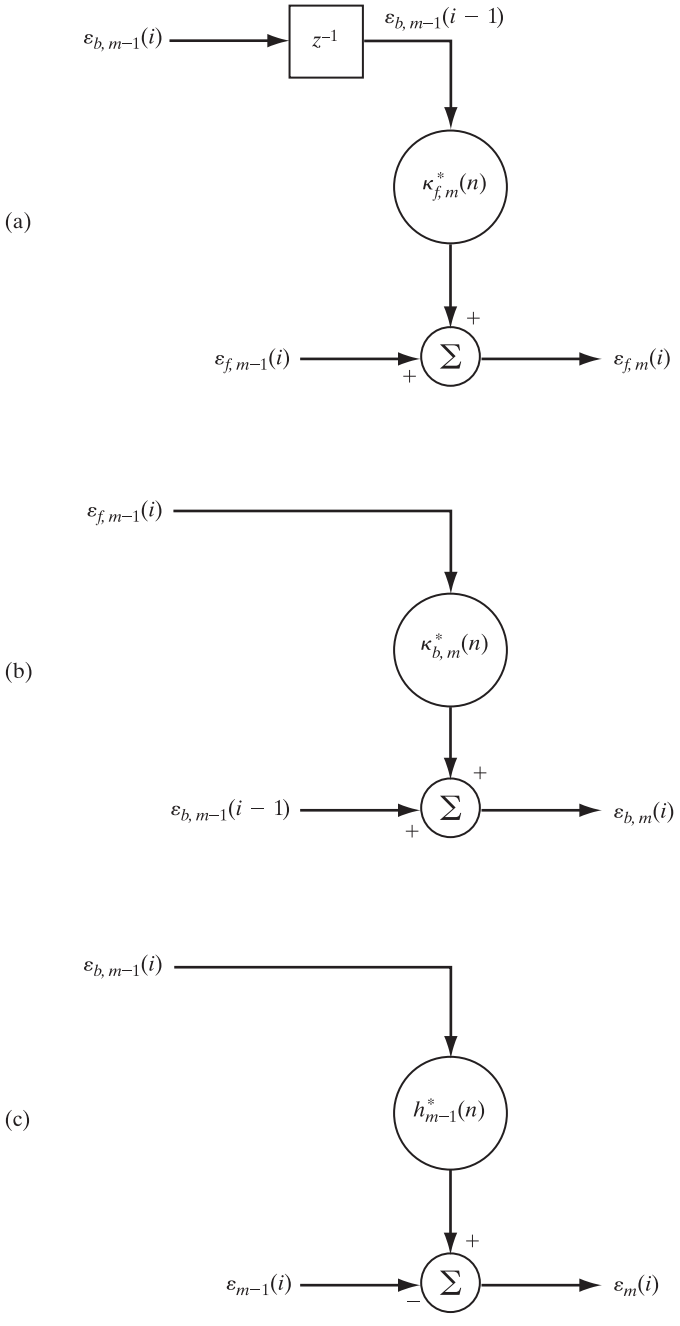


FIGURE 16.8 Angle-normalized versions of single-coefficient linear combiners for (a) forward prediction, (b) backward prediction, and (c) joint-process estimation.

16.7 FIRST-ORDER STATE-SPACE MODELS FOR LATTICE FILTERING

With the background material presented in Sections 16.2 through 16.6 at our disposal, we are now ready to embark on the derivation of algorithms for the design of order-recursive adaptive filters based on least-squares estimation. The approach used here builds on the one-to-one correspondences between RLS variables and Kalman variables. To carry out the derivation, we clearly need to formulate state-space representations for least-squares prediction using a lattice structure and its extension for joint-process estimation. For reasons explained in the previous section, we wish to formulate the state-space models in terms of angle-normalized estimation errors.

Consider, then, the following three n-by-1 vectors of angle-normalized prediction errors of order $m-1$:

$$\left.\begin{aligned}
\boldsymbol{\varepsilon}_{f,m-1}(n) &= \begin{bmatrix} \varepsilon_{f,m-1}(1) \\ \varepsilon_{f,m-1}(2) \\ \vdots \\ \varepsilon_{f,m-1}(n) \end{bmatrix}; \\
\boldsymbol{\varepsilon}_{b,m-1}(n-1) &= \begin{bmatrix} 0 \\ \varepsilon_{b,m-1}(1) \\ \vdots \\ \varepsilon_{b,m-1}(n-1) \end{bmatrix}; \\
\boldsymbol{\varepsilon}_{m-1}(n) &= \begin{bmatrix} e_{m-1}(1) \\ e_{m-1}(2) \\ \vdots \\ e_{m-1}(n) \end{bmatrix}.
\end{aligned}\right\} \tag{16.62}$$

To initialize the LSL predictor, we typically set

$$\mathcal{F}_{m-1}(0) = \mathcal{B}_{m-1}(-1) = \delta$$

and

$$\Delta_{m-1}(0) = 0.$$

The regularization parameter δ is usually small enough to have a negligible effect on $\mathcal{F}_{m-1}(n)$ and $\mathcal{B}_{m-1}(n-1)$ for increasing n, so it may be ignored. Accordingly, we may draw the following conclusions from Eqs. (16.10), (16.21), and (16.34), respectively:

1. The sum of weighted forward prediction-error squares, $\mathcal{F}_{m-1}(n)$, is equal to the exponentially weighted squared norm of the corresponding angle-normalized vector $\boldsymbol{\varepsilon}_{f,m-1}(n)$; that is,

$$\mathcal{F}_{m-1}(n) = \boldsymbol{\varepsilon}_{f,m-1}^{\mathrm{H}}(n)\boldsymbol{\Lambda}(n)\boldsymbol{\varepsilon}_{f,m-1}(n), \tag{16.63}$$

where

$$\boldsymbol{\Lambda}(n) = \mathrm{diag}[\lambda^{n-1}, \lambda^{n-2}, \ldots, 1]$$

is the n-by-n *exponential weighting matrix*.

2. The sum of weighted backward prediction-error squares, $\mathcal{B}_{m-1}(n-1)$, is equal to the exponentially weighted squared norm of the corresponding angle-normalized vector $\boldsymbol{\varepsilon}_{b,m-1}(n-1)$; that is,

$$\mathcal{B}_{m-1}(n-1) = \boldsymbol{\varepsilon}_{b,m-1}^{\mathrm{H}}(n-1)\boldsymbol{\Lambda}(n-1)\boldsymbol{\varepsilon}_{b,m-1}(n-1). \tag{16.64}$$

3. The parameter $\Delta_{m-1}^{*}(n)$, representing a form of exponentially weighted cross-correlation, is equal to the exponentially weighted inner product of the angle-normalized vectors $\boldsymbol{\varepsilon}_{b,m-1}(n-1)$ and $\boldsymbol{\varepsilon}_{f,m-1}(n)$; that is,

$$\Delta_{m-1}^{*}(n) = \boldsymbol{\varepsilon}_{b,m-1}^{\mathrm{H}}(n-1)\boldsymbol{\Lambda}(n)\boldsymbol{\varepsilon}_{f,m-1}(n). \tag{16.65}$$

These new definitions mean that the complex-conjugated forward reflection coefficient

$$\kappa_{f,m}^{*}(n) = -\frac{\Delta_{m-1}^{*}(n)}{\mathcal{B}_{m-1}(n-1)}$$

is also given by

$$\kappa_{f,m}^{*}(n) = -\frac{\boldsymbol{\varepsilon}_{b,m-1}^{\mathrm{H}}(n-1)\boldsymbol{\Lambda}(n)\boldsymbol{\varepsilon}_{f,m-1}(n)}{\boldsymbol{\varepsilon}_{b,m}^{\mathrm{H}}(n-1)\boldsymbol{\Lambda}(n-1)\boldsymbol{\varepsilon}_{b,m-1}(n-1)}. \tag{16.66}$$

In other words, the scalar $\kappa_{f,m}^{*}(n)$ can be interpreted as the coefficient that we need in order to project $\boldsymbol{\varepsilon}_{b,m-1}(n-1)$ onto $\boldsymbol{\varepsilon}_{f,m-1}(n)$. This suggests that we may replace the problem of projecting the a posteriori prediction vector $\mathbf{b}_{m-1}(n-1)$ onto the a posteriori prediction vector $\mathbf{f}_{m-1}(n)$ by the equivalent problem of projecting the angle-normalized prediction vector $\boldsymbol{\varepsilon}_{b,m-1}(n-1)$ onto the angle-normalized prediction vector $\boldsymbol{\varepsilon}_{f,m-1}(n)$. A similar conclusion follows for the problem of projecting $\mathbf{f}_{m-1}(n)$ onto $\mathbf{b}_{m-1}(n-1)$ and also for the joint-process estimation problem.

Accordingly, referring to the three signal-flow graphs of Fig. 16.8, we may formulate a combination of *three first-order state-space models* for stage m of the least-squares lattice filtering problem, based on the following three projections:

1. For forward linear prediction, $\boldsymbol{\varepsilon}_{b,m-1}(n-1)$ is projected onto $\boldsymbol{\varepsilon}_{f,m-1}(n)$.
2. For backward linear prediction, $\boldsymbol{\varepsilon}_{f,m-1}(n)$ is projected onto $\boldsymbol{\varepsilon}_{b,m-1}(n-1)$.
3. For joint-process linear estimation, $\boldsymbol{\varepsilon}_{b,m-1}(n)$ is projected onto $\boldsymbol{\varepsilon}_{m-1}(n)$.

Thus, bearing in mind the one-to-one correspondences between the Kalman variables and RLS variables that were established in Chapter 14, we may break up the state-space characterization of stage m of the LSL filtering problem into three parts:

1. *Forward prediction*: Here,

$$x_1(n+1) = \lambda^{-1/2}x_1(n) \tag{16.67}$$

and

$$y_1(n) = \varepsilon_{b,m-1}^{*}(n-1)x_1(n) + \nu_1(n), \tag{16.68}$$

where $x_1(n)$ is the state variable and the reference signal (observation) is

$$y_1(n) = \lambda^{-n/2}\varepsilon_{f,m-1}^{*}(n). \tag{16.69}$$

The scalar measurement noise $\nu_1(n)$ is a random variable with zero mean and unit variance.

2. *Backward prediction*: Here,

$$x_2(n+1) = \lambda^{-1/2} x_2(n) \tag{16.70}$$

and

$$y_2(n) = \varepsilon^*_{f,m-1}(n) x_2(n) + \nu_2(n), \tag{16.71}$$

where $x_2(n)$ is the second state variable and the second reference signal (observation) is

$$y_2(n) = \lambda^{-n/2} \varepsilon^*_{b,m-1}(n). \tag{16.72}$$

As with $\nu_1(n)$, the scalar measurement noise $\nu_2(n)$ is a random variable with zero mean and unit variance.

3. *Joint-process estimation*: Here,

$$x_3(n+1) = \lambda^{-1/2} x_3(n) \tag{16.73}$$

and

$$y_3(n) = \varepsilon^*_{b,m-1}(n) x_3(n) + \nu_3(n), \tag{16.74}$$

where $x_3(n)$ is the third and final state variable and the corresponding reference signal (observation) is

$$y_3(n) = \lambda^{-n/2} \varepsilon^*_{m-1}(n). \tag{16.75}$$

As before, the scalar measurement noise $\nu_3(n)$ is a random variable with zero mean and unit variance.

The noise variables $\nu_1(n)$, $\nu_2(n)$, and $\nu_3(n)$ are all independent of each other.

On the basis of the state-space models just described, we may formulate the list of one-to-one correspondences, shown in Table 16.3, between Kalman variables and three sets of LSL variables, assuming a prediction order of $m-1$. The three sets of LSL variables separately refer to forward prediction, backward prediction, and joint-process estimation. The first three lines of the table follow readily from the state-space models of Eqs. (16.67) through (16.75), together with Table 14.3 of Chapter 14 listing the one-to-one correspondences between Kalman variables and RLS variables. To verify the remaining three lines of correspondences, we may proceed as follows for the case of forward linear prediction:

1. From Table 14.3 of Chapter 14, we recall the following correspondence between the filtered state-error correlation matrix in Kalman filter theory and the inverse of the correlation matrix of the input vector in RLS theory:

$$\mathbf{K}(n-1) \leftrightarrow \lambda^{-1}\mathbf{P}(n-1) = \lambda^{-1}\mathbf{\Phi}^{-1}(n-1).$$

(Hereafter, double-headed arrows signify one-to-one correspondences.) For the problem at hand, depicted in Fig. 16.8(a), we see that the input applied to the forward reflection coefficient is $\varepsilon_{b,m-1}(i-1)$, after the unit delay. For forward prediction of order $m-1$, we may therefore write

TABLE 16.3 Summary of One-to-One Correspondences Between Kalman Variables and Angle-Normalized LSL Variables in Stage m of the Lattice Predictor

Kalman variable	LSL variable/parameter		
	Forward prediction	Backward prediction	Joint-process estimation
$y(n)$	$\lambda^{-n/2}\varepsilon_{f,m-1}^{*}(n)$	$\lambda^{-n/2}\varepsilon_{b,m-1}^{*}(n-1)$	$\lambda^{-n/2}\varepsilon_{m-1}^{*}(n)$
$\mathbf{u}^{\mathrm{H}}(n)$	$\varepsilon_{b,m-1}^{*}(n-1)$	$\varepsilon_{f,m-1}^{*}(n)$	$\varepsilon_{b,m-1}^{*}(n)$
$\hat{\mathbf{x}}(n\,\vert\,\mathcal{Y}_{n-1})$	$-\lambda^{-n/2}\boldsymbol{\kappa}_{f,m}(n-1)$	$-\lambda^{-n/2}\boldsymbol{\kappa}_{b,m}(n-1)$	$\lambda^{-n/2}\boldsymbol{\kappa}_{m-1}(n-1)$
$\mathbf{K}(n-1)$	$\lambda^{-1}\mathcal{B}_{m-1}^{-1}(n-2)$	$\lambda^{-1}\mathcal{F}_{m-1}^{-1}(n-1)$	$\lambda^{-1}\mathcal{B}_{m-1}^{-1}(n-1)$
$\mathbf{g}(n)$	$\lambda^{-1/2}\mathcal{B}_{m-1}^{-1}(n-1)\boldsymbol{\varepsilon}_{b,m-1}(n-1)$	$\lambda^{-1/2}\mathcal{F}_{m-1}^{-1}(n)\boldsymbol{\varepsilon}_{f,m-1}(n)$	$\lambda^{-1/2}\mathcal{B}_{m-1}^{-1}(n)\varepsilon_{b,m-1}(n)$
$r(n)$	$\dfrac{\gamma_{m-1}(n-1)}{\gamma_m(n-1)}$	$\dfrac{\gamma_{m-1}(n-1)}{\gamma_m(n)}$	$\dfrac{\gamma_{m-1}(n)}{\gamma_m(n)}$

$$\mathbf{K}(n-1) \leftrightarrow \lambda^{-1}\left(\sum_{i=1}^{n-2}\lambda^{n-i+2}\,|\,\varepsilon_{b,m-1}(i-2)\,|^2\right)^{-1} = \lambda^{-1}\mathcal{B}_{m-1}^{-1}(n-2). \tag{16.76}$$

2. From Table 14.3 of Chapter 14, we also recall the following correspondence between the gain vector in Kalman filter theory and the gain vector in RLS theory:

$$\mathbf{g}(n) \leftrightarrow \lambda^{-1/2}\mathbf{k}(n).$$

Adapting the definition of the gain vector $\mathbf{k}(n)$ in Eq. (10.22) of Chapter 10 for the problem of forward prediction of order $m-1$, we may write

$$\mathbf{g}(n) \leftrightarrow \lambda^{-1/2}\mathcal{B}_{m-1}^{-1}(n-1)\varepsilon_{b,m-1}(n-1). \tag{16.77}$$

3. Thus far, the derivations of the entries defined by Eqs. (16.76) and (16.77) have been relatively straightforward. However, the last entry in Table 16.3 requires more subtle considerations. From Section 14.8, we recall that the conversion factor $r^{-1}(n)$ in Kalman filter theory is equal to the ratio of a posteriori estimation error $e(n)$ to the a priori estimation error $\alpha(n)$. In contrast, the angle-normalized single-coefficient models of Fig. 16.8 do *not* distinguish between a priori and a posteriori estimation errors. To get around this difficulty, we proceed as follows: First, the a priori estimation error or innovation in Kalman filter theory is defined by

$$\alpha(n) = y(n) - \mathbf{u}^{\mathrm{H}}(n)\hat{\mathbf{x}}(n\,|\,\mathcal{Y}_{n-1}). \tag{16.78}$$

From the first three lines of Table 16.3, we have the following one-to-one correspondences for forward prediction:

$$y(n) \leftrightarrow \lambda^{-n/2}\varepsilon_{f,m}^{*}(n);$$
$$\mathbf{u}^{\mathrm{H}}(n) \leftrightarrow \varepsilon_{b,m-1}^{*}(n-1);$$
$$\hat{\mathbf{x}}(n\,|\,\mathcal{Y}_{n-1}) \leftrightarrow -\lambda^{-n/2}\boldsymbol{\kappa}_{f,m}(n-1).$$

Therefore, substituting these correspondences into the right-hand side of Eq. (16.78), we get

$$\alpha(n) \leftrightarrow \lambda^{-n/2}(\varepsilon^*_{f,m-1}(n) + \kappa_{f,m}(n-1)\varepsilon^*_{b,m-1}(n-1)).$$

Next, using Eqs. (16.59) and (16.60), we may equivalently write

$$\alpha(n) \leftrightarrow \lambda^{-n/2}\gamma^{1/2}_{m-1}(n-1)(\eta^*_{m-1}(n) + \kappa_{f,m}(n-1)\beta^*_{m-1}(n-1)).$$

Thus, in light of Eq. (16.57), we conclude that, for forward prediction of order $m-1$,

$$\alpha(n) \leftrightarrow \lambda^{-n/2}\gamma^{1/2}_{m-1}(n-1)\eta^*_m(n). \tag{16.79}$$

Next, the filtered estimation error in Kalman filter theory is defined by

$$e(n) = y(n) - \mathbf{u}^{\mathrm{H}}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_n), \tag{16.80}$$

where the filtered state estimate is itself defined by

$$\hat{\mathbf{x}}(n\,|\mathcal{Y}_n) = \mathbf{F}(n, n+1)\hat{\mathbf{x}}(n+1\,|\mathcal{Y}_n).$$

For the problem at hand, the transition matrix is [see Eq. (14.89)]

$$\mathbf{F}(n+1, n) = \lambda^{-1/2},$$

and from the inverse rule governing the transition matrix,

$$\mathbf{F}(n, n+1) = \lambda^{1/2}.$$

Thus, from the third row of Table 16.3,

$$\hat{\mathbf{x}}(n+1\,|\mathcal{Y}_n) \leftrightarrow -\lambda^{-(n+1)/2}\kappa_{f,m}(n).$$

It follows, as expected, that

$$\hat{\mathbf{x}}(n\,|\mathcal{Y}_n) \leftrightarrow -\lambda^{-n/2}\kappa_{f,m}(n).$$

We may therefore use Eq. (16.80) to write, for forward linear prediction,

$$e(n) \leftrightarrow \lambda^{-n/2}\big(\varepsilon^*_{f,m-1}(n) + \kappa_{f,m}(n)\varepsilon^*_{b,m-1}(n-1)\big).$$

Again, using the relations of Eqs. (16.79) and (16.60), we may equivalently write

$$e(n) \leftrightarrow \lambda^{-n/2}\gamma^{-1/2}_{m-1}(n-1)\big(f^*_{m-1}(n) + \kappa_{f,m}(n)b^*_{m-1}(n-1)\big).$$

Thus, in light of Eq. (16.41), we conclude that, for forward prediction of order $m-1$,

$$e(n) \leftrightarrow \lambda^{-n/2}\gamma^{-n/2}_{m-1}(n-1)f^*_m(n). \tag{16.81}$$

Finally, we use Eqs. (16.79) and (16.81) to write the following one-to-one correspondence for forward prediction of order $m-1$:

$$r(n) \leftrightarrow \frac{\gamma_{m-1}(n-1)}{\gamma_m(n-1)}. \tag{16.82}$$

Thus, Eqs. (16.76), (16.77), and (16.82) provide the basis for the last three lines of correspondences between the Kalman and LSL variables for forward prediction listed in Table 16.3. By proceeding in a manner similar to that just described, we may fill in the

remaining correspondences pertaining to backward prediction and joint-process estimation. (We leave this as an exercise for the reader; see Problem 11.)

16.8 QR-DECOMPOSITION–BASED LEAST-SQUARES LATTICE (QRD-LSL) FILTERS

Equipped with the state-space models for LSL filtering described in Section 16.7 and square-root information filtering developed in Chapter 15, we are at long last positioned to get on with the derivation of our first and most important order-recursive adaptive filtering algorithm. The writings of arrays for the algorithm and their expansions are presented in three parts, dealing with adaptive forward prediction, adaptive backward prediction, and adaptive joint-process estimation, in that order. The basic tool for the writings presented herein is the prearray-to-postarray transformation that defines the square-root information filtering algorithm of Eq. (15.38), which is reproduced here for convenience of presentation:

$$\underbrace{\begin{bmatrix} \lambda^{1/2}\mathbf{K}^{-\mathrm{H}/2}(n-1) & \lambda^{1/2}\mathbf{u}(n) \\ \hat{\mathbf{x}}^{\mathrm{H}}(n\,|\,\mathcal{Y}_{n-1})\mathbf{K}^{-\mathrm{H}/2}(n-1) & y^*(n) \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}}_{\text{Prearray}} \boldsymbol{\Theta}(n)$$

$$= \underbrace{\begin{bmatrix} \mathbf{K}^{-\mathrm{H}/2}(n) & \mathbf{0} \\ \hat{\mathbf{x}}^{\mathrm{H}}(n+1\,|\,\mathcal{Y}_n)\mathbf{K}^{-\mathrm{H}/2}(n) & r^{-1/2}(n)\alpha^*(n) \\ \lambda^{1/2}\mathbf{u}^{\mathrm{H}}(n)\mathbf{K}^{1/2}(n) & r^{-1/2}(n) \end{bmatrix}}_{\text{Postarray}}, \tag{16.83}$$

where $\boldsymbol{\Theta}(n)$ is a *unitary rotation* configured to annihilate the first block entry, $\lambda^{1/2}\mathbf{u}(n)$, in the second column of the prearray.

Arrays for Adaptive Forward Prediction

Adapting Eq. (16.83) to suit the forward prediction model described by the state-space equations (16.67) through (16.69), with the aid of Table 16.3 listing the one-to-one correspondences between the Kalman variables and LSL variables (for forward prediction), we may write the following arrays for stage m of the LSL forward predictor:

$$\underbrace{\begin{bmatrix} \lambda^{1/2}\mathcal{B}_{m-1}^{1/2}(n-2) & \varepsilon_{b,\,m-1}(n-1) \\ \lambda^{1/2}p_{f,\,m-1}^*(n-1) & \varepsilon_{f,\,m-1}(n) \\ 0 & \gamma_{m-1}^{1/2}(n-1) \end{bmatrix}}_{\text{Prearray}} \boldsymbol{\Theta}_{b,\,m-1}(n-1) \tag{16.84}$$

$$= \underbrace{\begin{bmatrix} \mathcal{B}_{m-1}^{1/2}(n-1) & 0 \\ p_{f,\,m-1}^*(n) & \varepsilon_{f,\,m}(n) \\ b_{m-1}^*(n-1)\mathcal{B}_{m-1}^{-1/2}(n-1) & \gamma_m^{1/2}(n-1) \end{bmatrix}}_{\text{Postarray}}.$$

Note, however, that in writing Eq. (16.84), we have done two things: First, the common factors $\lambda^{1/2}$ and $\lambda^{-n/2}$ have been cancelled from the pre- and postarrays in the first and

second rows, respectively. Second, we have multiplied the pre- and postarrays in the third row by $\gamma_{m-1}^{1/2}(n-1)$. The motivation for so doing is to simplify the products of the transformation.

The scalar quantities $\mathcal{B}_{m-1}(n-1)$ and $p_{f,m-1}(n)$ appearing in the postarray are respectively defined as follows:

1. The real-valued quantity

$$\begin{aligned}\mathcal{B}_{m-1}(n-1) &= \sum_{i=1}^{n-1} \lambda^{n-1-i} \varepsilon_{b,m-1}(i-1)\varepsilon_{b,m-1}^{*}(i-1) \\ &= \lambda \mathcal{B}_{m-1}(n-2) + \varepsilon_{b,m-1}(n-1)\varepsilon_{b,m-1}^{*}(n-1)\end{aligned} \tag{16.85}$$

is the autocorrelation of the delayed, angle-normalized backward prediction error $\varepsilon_{b,m-1}(n-1)$ for a lag of zero. The quantity $\mathcal{B}_{m-1}(n-1)$ may also be interpreted as the minimum value of the sum of weighted backward a posteriori prediction-error squares, which is defined in accordance with RLS theory as follows [see Eq. (16.22)]:

$$\mathcal{B}_{m-1}(n-1) = \lambda \mathcal{B}_{m-1}(n-2) + \beta_{m-1}(n-1)b_{m-1}^{*}(n-1).$$

Note that the product term $\beta_{m-1}(n-1)b_{m-1}^{*}(n-1)$ is always real; that is,

$$\beta_{m-1}(n-1)b_{m-1}^{*}(n-1) = \beta_{m-1}^{*}(n-1)b_{m-1}(n-1).$$

2. Except for the factor $\mathcal{B}_{m-1}^{-1/2}(n-1)$, the complex-valued quantity

$$p_{f,m-1}(n) = \frac{\Delta_{m-1}(n)}{\mathcal{B}_{m-1}^{1/2}(n-1)}, \tag{16.86}$$

where

$$\begin{aligned}\Delta_{m-1}(n) &= \sum_{i=1}^{n} \lambda^{n-i} \varepsilon_{b,m-1}(i-1)\varepsilon_{f,m-1}^{*}(i) \\ &= \lambda \Delta_{m-1}(n-1) + \varepsilon_{b,m-1}(n-1)\varepsilon_{f,m-1}^{*}(n),\end{aligned} \tag{16.87}$$

is the cross-correlation between the angle-normalized forward and backward prediction errors. Indeed, $p_{f,m-1}(n)$ is related to the forward reflection coefficient $\kappa_{f,m}(n)$ for prediction order m by the formula (Haykin, 1991)

$$\begin{aligned}\kappa_{f,m}(n) &= -\frac{\Delta_{m-1}(n)}{\mathcal{B}_{m-1}(n-1)} \\ &= -\frac{p_{f,m-1}(n)}{\mathcal{B}_{m-1}^{1/2}(n-1)}.\end{aligned} \tag{16.88}$$

The 2-by-2 matrix $\boldsymbol{\Theta}_{b,m-1}(n-1)$ in Eq. (16.84) is a unitary rotation that reduces the $(1,2)$ entry in the postarray to zero; that is, the matrix annihilates the entry $\varepsilon_{b,m-1}(n-1)$ in the prearray. This requirement is readily satisfied by using the Givens rotation

$$\boldsymbol{\Theta}_{b,m-1}(n-1) = \begin{bmatrix} c_{b,m-1}(n-1) & -s_{b,m-1}(n-1) \\ s_{b,m-1}^{*}(n-1) & c_{b,m-1}(n-1) \end{bmatrix}, \tag{16.89}$$

where the cosine and sine parameters are themselves defined by

$$c_{b,\,m-1}(n-1) = \frac{\lambda^{1/2}\mathcal{B}_{m-1}^{1/2}(n-2)}{\mathcal{B}_{m-1}^{1/2}(n-1)} \tag{16.90}$$

and

$$s_{b,\,m-1}(n-1) = \frac{\boldsymbol{\varepsilon}_{b,\,m-1}(n-1)}{\mathcal{B}_{m-1}^{1/2}(n-1)}, \tag{16.91}$$

respectively. Thus, using Eq. (16.89) in Eq. (16.84), we get the following update relations in addition to that of Eq. (16.85):

$$p_{f,\,m-1}^{*}(n) = c_{b,\,m-1}(n-1)\lambda^{1/2}p_{f,\,m-1}^{*}(n-1) + s_{b,\,m-1}^{*}(n-1)\varepsilon_{f,\,m-1}(n); \tag{16.92}$$

$$\varepsilon_{f,\,m}(n) = c_{b,\,m-1}(n-1)\varepsilon_{f,\,m-1}(n) - s_{b,\,m-1}(n-1)\lambda^{1/2}p_{f,\,m-1}^{*}(n-1); \tag{16.93}$$

$$\gamma_{m}^{1/2}(n-1) = c_{b,\,m-1}(n-1)\gamma_{m-1}^{1/2}(n-1). \tag{16.94}$$

Equations (16.85) and (16.90) through (16.94) constitute the set of relations for a square-root information-filtering solution to the adaptive forward linear problem in order-recursive LSL filtering theory.

Arrays for Adaptive Backward Prediction

Consider next the adaptive backward prediction model described by the state-space equations (16.70) through (16.72). With the aid of the one-to-one correspondences between the Kalman variables and LSL variables (for backward prediction) listed in Table 16.3, we may use Eq. (16.83) to write the following array for stage m of the least-squares lattice predictor:

$$\underbrace{\begin{bmatrix} \lambda^{1/2}\mathcal{F}_{m-1}^{1/2}(n-1) & \varepsilon_{f,\,m-1}(n) \\ \lambda^{1/2}p_{b,\,m-1}^{*}(n-1) & \varepsilon_{b,\,m-1}(n-1) \\ 0 & \gamma_{m-1}^{1/2}(n-1) \end{bmatrix}}_{\text{Prearray}} \boldsymbol{\Theta}_{f,\,m-1}(n) = \underbrace{\begin{bmatrix} \mathcal{F}_{m-1}^{1/2}(n) & 0 \\ p_{b,\,m-1}^{*}(n) & \varepsilon_{b,\,m}(n) \\ f_{m-1}^{*}(n)\mathcal{F}_{m-1}^{-1/2}(n) & \gamma_{m}^{1/2}(n) \end{bmatrix}}_{\text{Postarray}}. \tag{16.95}$$

Here, again, in writing Eq. (16.95), we have done the following things: First, the common factors $\lambda^{1/2}$ and $\lambda^{-n/2}$ have been cancelled from the pre- and postarrays in the first and second rows, respectively. Second, we have multiplied the pre- and postarrays in the third row by $\gamma_{m-1}^{1/2}(n-1)$.

The two new scalar quantities $\mathcal{F}_{m-1}(n)$ and $p_{b,\,m-1}(n)$ appearing in the postarray of Eq. (16.95) are respectively defined as follows:

1. The real-valued quantity $\mathcal{F}_{m-1}(n)$ is the autocorrelation of the angle-normalized forward prediction error $\varepsilon_{f,\,m-1}(n)$ for a lag of zero; that is,

$$\begin{aligned} \mathcal{F}_{m-1}(n) &= \sum_{i=1}^{n} \lambda^{n-i}\varepsilon_{f,\,m-1}(i)\varepsilon_{f,\,m-1}^{*}(i) \\ &= \lambda\mathcal{F}_{m-1}(n-1) + \varepsilon_{f,\,m-1}(n)\varepsilon_{f,\,m-1}^{*}(n). \end{aligned} \tag{16.96}$$

$\mathcal{F}_{m-1}(n)$ may also be interpreted as the minimum value of the sum of forward prediction-error squares, which is defined, in accordance with RLS theory, as [see Eq. (16.11)]

$$\mathcal{F}_{m-1}(n) = \lambda\mathcal{F}_{m-1}(n-1) + \eta_{m-1}(n)f^*_{m-1}(n). \tag{16.97}$$

In a manner similar to adaptive forward prediction, the product $\eta_{m-1}(n)f^*_{m-1}(n)$ is always real valued.

2. Except for the factor $\mathcal{F}^{-1/2}_{m-1}(n)$, the complex-valued quantity

$$p_{b,\,m-1}(n) = \frac{\Delta^*_{m-1}(n)}{\mathcal{F}^{1/2}_{m-1}(n)} \tag{16.98}$$

is the complex conjugate of the cross-correlation between the angle-normalized forward and backward prediction errors. [The said cross-correlation is defined in Eq. (16.87).] The quantity $p_{b,\,m-1}(n)$ is also related to the backward reflection coefficient for prediction order m by the formula (Haykin, 1991)

$$\begin{aligned}\kappa_{b,\,m}(n) &= -\frac{\Delta^*_{m-1}(n)}{\mathcal{F}_{m-1}(n)} \\ &= -\frac{p_{b,\,m-1}(n)}{\mathcal{F}^{1/2}_{m-1}(n)}.\end{aligned} \tag{16.99}$$

The 2-by-2 transformation matrix $\boldsymbol{\Theta}_{f,\,m-1}(n)$ is a unitary rotation that reduces the (1,2) entry of the postarray in Eq. (16.95) to zero; that is, the matrix annihilates the entry $\varepsilon_{f,\,m-1}(n)$ in the prearray of this same equation. This requirement may be satisfied by using the Givens rotation

$$\boldsymbol{\Theta}_{f,\,m-1}(n) = \begin{bmatrix} c_{f,\,m-1}(n) & -s_{f,\,m-1}(n) \\ s^*_{f,\,m-1}(n) & c_{f,\,m-1}(n) \end{bmatrix}, \tag{16.100}$$

where the cosine and sine parameters are themselves defined by

$$c_{f,\,m-1}(n) = \frac{\lambda^{1/2}\mathcal{F}^{1/2}_{m-1}(n-1)}{\mathcal{F}^{1/2}_{m-1}(n)} \tag{16.101}$$

and

$$s_{f,\,m-1}(n) = \frac{\varepsilon_{f,\,m-1}(n)}{\mathcal{F}^{1/2}_{m-1}(n)}, \tag{16.102}$$

respectively. Thus, substituting Eq. (16.100) into Eq. (16.95), we readily deduce the following recursions:

$$p^*_{b,\,m-1}(n) = c_{f,\,m-1}(n)\lambda^{1/2}p^*_{b,\,m-1}(n-1) + s^*_{f,\,m-1}(n)\varepsilon_{b,\,m-1}(n-1); \tag{16.103}$$

$$\varepsilon_{b,\,m}(n) = c_{f,\,m-1}(n)\varepsilon_{b,\,m-1}(n-1) - s_{f,\,m-1}(n)\lambda^{1/2}p^*_{b,\,m-1}(n-1); \tag{16.104}$$

$$\gamma^{1/2}_m(n) = c_{f,\,m-1}(n)\gamma^{1/2}_{m-1}(n-1). \tag{16.105}$$

Equations (16.96) and (16.101) through (16.105) constitute the set of recursions for a square-root information-filtering solution to the adaptive backward problem in order-recursive LSL filtering theory.

Array for Joint-Process Estimation

Finally, consider the joint-process estimation problem described by the state-space equations (16.73) through (16.75), pertaining to stage m of the LSL filtering process. With the aid of the one-to-one correspondences between the Kalman variables and LSL variables (for joint-process estimation) listed in Table 16.3, we may use Eq. (16.83) to write

$$\underbrace{\begin{bmatrix} \lambda^{1/2}\mathcal{B}_{m-1}^{1/2}(n-1) & \varepsilon_{b,m-1}(n) \\ \lambda^{1/2}p_{m-1}^{*}(n-1) & \varepsilon_{m-1}(n) \\ 0 & \gamma_{m-1}^{1/2}(n) \end{bmatrix}}_{\text{Prearray}} \boldsymbol{\Theta}_{b,m-1}(n) = \underbrace{\begin{bmatrix} \mathcal{B}_{m-1}^{1/2}(n) & 0 \\ p_{m-1}^{*}(n) & \varepsilon_{m}(n) \\ b_{m-1}^{*}(n)\mathcal{B}_{m-1}^{-1/2}(n) & \gamma_{m}^{1/2}(n) \end{bmatrix}}_{\text{Postarray}}. \tag{16.106}$$

In writing Eq. (16.106), we have done certain things to the pre- and postarrays, which are the same as those done for the corresponding equations pertaining to the adaptive forward prediction and adaptive backward prediction.

In Eq. (16.106), there is only one new quantity that we have to define, namely, $p_{m-1}(n)$, for the postarray. Except for the factor $\mathcal{B}_{m-1}^{-1/2}(n)$, this new quantity is the cross-correlation between the angle-normalized backward prediction error and angle-normalized joint-estimation error; that is,

$$p_{m-1}(n) = \frac{1}{\mathcal{B}_{m-1}^{1/2}(n)} \sum_{i=1}^{n} \lambda^{n-i} \varepsilon_{b,m-1}(i)\varepsilon_{m-1}^{*}(i). \tag{16.107}$$

The joint-estimation (regression) parameter $h_{m-1}(n)$ for prediction order $m-1$ is correspondingly defined by (Haykin, 1991)

$$h_{m-1}(n) = \frac{p_{m-1}(n)}{\mathcal{B}_{m-1}^{1/2}(n)}. \tag{16.108}$$

The 2-by-2 transformation matrix $\boldsymbol{\Theta}_{b,m-1}(n)$ is a unitary rotation designed to annihilate the entry $\varepsilon_{b,m-1}(n)$ in the prearray of Eq. (16.106). To do this, we may use the same Givens rotation as that in Eq. (16.109), except for a shift in time by one adaptation cycle. That is,

$$\boldsymbol{\Theta}_{b,m-1}(n) = \begin{bmatrix} c_{b,m-1}(n) & -s_{b,m-1}(n) \\ s_{b,m-1}^{*}(n) & c_{b,m-1}(n) \end{bmatrix}, \tag{16.109}$$

where

$$c_{b,m-1}(n) = \frac{\lambda^{1/2}\mathcal{B}_{m-1}^{1/2}(n-1)}{\mathcal{B}_{m-1}^{1/2}(n)} \tag{16.110}$$

and

$$s_{b,m-1}(n) = \frac{\varepsilon_{b,m-1}(n)}{\mathcal{B}_{m-1}^{1/2}(n)}. \tag{16.111}$$

Substituting Eq. (16.109) into Eq. (16.106), we get the desired pair of recursions:

$$p^*_{m-1}(n) = c_{b,m-1}(n)\lambda^{1/2}p^*_{m-1}(n-1) + s^*_{b,m-1}(n)\varepsilon_{m-1}(n); \tag{16.112}$$

$$\varepsilon_m(n) = c_{b,m-1}(n)\varepsilon_{m-1}(n) - s_{b,m-1}(n)\lambda^{1/2}p^*_{m-1}(n-1). \tag{16.113}$$

Equations (16.85), (16.90), and (16.91) with adaptation cycle $n-1$ replaced by n, together with Eqs. (16.112) and (16.113), constitute the set of recursions for a square-root information-filtering solution to the joint-process estimation problem in order-recursive LSL filtering theory.

Summary of the QRD-LSL Algorithm Using a Posteriori and a Priori Estimation Errors

Table 16.4 presents a summary of the angle-normalized QRD-LSL algorithm,[2] based on the arrays of Eqs. (16.84), (16.95), and (16.106). Note that the forward and backward predictions are performed for $m = 1, 2, \ldots, M$, whereas those for the joint-process estimation are performed for $m = 1, 2, \ldots, M+1$, where M is the final prediction order. That is, the joint-process estimation involves one final set of computations pertaining to $m = M+1$. Note also that in the second and third arrays of the table, we have omitted the particular rows that involve updating the conversion factor. This requirement is taken care of by the first array.

Table 16.4 includes the *initialization* procedure, which is of a *soft-constraint* form. This form of initialization is consistent with that adopted for the conventional RLS

[2]The use of arrays for deriving the QRD-LSL algorithm was first described in the paper by Sayed and Kailath (1994).

The idea of a fast QR-decomposition–based algorithm for RLS estimation was first presented by Cioffi (1988). A detailed derivation of the algorithm is described in Cioffi (1990). In the latter paper, Cioffi presents a geometric approach to the derivation that is reminiscent of his earlier work on fast FIR filters. The algorithm derived by Cioffi is of a Kalman, or matrix-oriented, type. Several other authors have presented seemingly simple algebraic derivations and other versions of the QRD-fast RLS algorithm (Bellanger, 1988a, b; Proudler et al., 1988, 1989; Regalia & Bellanger, 1991). The paper by Proudler et al. (1989) is of particular interest in that it develops a novel implementation of the QRD-RLS algorithm by using a lattice structure. A similar fast algorithm was derived independently by Ling (1989), using the modified Gram–Schmidt orthogonalization procedure. The connection between the modified Gram–Schmidt orthogonalization and QR-decomposition is discussed in Shepherd and McWhirter (1993).

Haykin (1991) presented a development of the QRD-LSL algorithm that is based on a hybridization of ideas due to Proudler et al. (1989) and Regalia and Bellanger (1991). Specifically, the development followed Proudler et al. in deriving QR-decomposition-based solutions to forward and backward linear prediction problems and followed Regalia and Bellanger in solving the joint-process estimation problem. In such an approach, the complications in the Proudler et al. procedure, which uses forward linear prediction errors for joint-process estimation, are avoided. The structure of the QRD-LSL algorithm derived by Haykin follows a philosophy directly analogous to that described for conventional LSL algorithms.

Rontogiannis and Theodoridis (1998a, b) proposed a unified approach for deriving QRD-fast algorithms. The starting point of the derivation is the inverse Cholesky factor of the time-average correlation matrix of the input data; the inverse Cholesky factor is derived using a QR-decomposition. Rontogiannis and Theodoridis actually propose two algorithms: One is a fixed-order QR-decomposition-based scheme, and the other is an order-recursive, latticelike algorithm based on Givens rotations.

TABLE 16.4 Summary of the QRD-LSL Algorithm

1. *Unitary (Givens) rotations:*

$$\boldsymbol{\Theta}_{b,m}(n) = \begin{bmatrix} c_{b,m}(n) & -s_{b,m}(n) \\ s^*_{b,m}(n) & c_{b,m}(n) \end{bmatrix}$$

where

$$c_{b,m}(n) = \frac{\lambda^{1/2}\mathcal{B}^{1/2}_m(n-1)}{\mathcal{B}^{1/2}_m(n)}$$

and

$$s_{b,m}(n) = \frac{\boldsymbol{\varepsilon}_{b,m}(n)}{\mathcal{B}^{1/2}_m(n)}.$$

$$\boldsymbol{\Theta}_{f,m}(n) = \begin{bmatrix} c_{f,m}(n) & -s_{f,m}(n) \\ s^*_{f,m}(n) & c_{f,m}(n) \end{bmatrix}$$

where

$$c_{f,m}(n) = \frac{\lambda^{1/2}\mathcal{F}^{1/2}_m(n-1)}{\mathcal{F}^{1/2}_m(n)}$$

and

$$s_{f,m}(n) = \frac{\boldsymbol{\varepsilon}_{f,m}(n)}{\mathcal{F}^{1/2}_m(n)}.$$

2. *Computations:*

(a) *Predictions:* For each adaptation cycle $n = 1, 2, \ldots$, perform the following computations and repeat for each prediction order $m = 1, 2, \ldots, M$, where M is the final prediction order:

$$\begin{bmatrix} \lambda^{1/2}\mathcal{B}^{1/2}_{m-1}(n-2) & \varepsilon_{b,m-1}(n-1) \\ \lambda^{1/2}p^*_{f,m-1}(n-1) & \varepsilon_{f,m-1}(n) \\ 0 & \gamma^{1/2}_{m-1}(n-1) \end{bmatrix} \boldsymbol{\Theta}_{b,m-1}(n-1) = \begin{bmatrix} \mathcal{B}^{1/2}_{m-1}(n-1) & 0 \\ p^*_{f,m-1}(n) & \boldsymbol{\varepsilon}_{f,m}(n) \\ b^*_{m-1}(n-1)\mathcal{B}^{-1/2}_{m-1}(n-1) & \gamma^{1/2}_m(n-1) \end{bmatrix}$$

$$\begin{bmatrix} \lambda^{1/2}\mathcal{F}^{1/2}_{m-1}(n-1) & \varepsilon_{f,m-1}(n) \\ \lambda^{1/2}p^*_{b,m-1}(n-1) & \varepsilon_{b,m-1}(n-1) \end{bmatrix} \boldsymbol{\Theta}_{f,m-1}(n) = \begin{bmatrix} \mathcal{F}^{1/2}_{m-1}(n) & 0 \\ p^*_{b,m-1}(n) & \boldsymbol{\varepsilon}_{b,m}(n) \end{bmatrix}$$

(b) *Filtering:* For each adaptation cycle $n = 1, 2, \ldots$, perform the following computations and repeat for each prediction order $m = 1, 2, \ldots, M+1$, where M is the final prediction order:

$$\begin{bmatrix} \lambda^{1/2}\mathcal{B}^{1/2}_{m-1}(n-1) & \varepsilon_{b,m-1}(n) \\ \lambda^{1/2}p^*_{m-1}(n-1) & \varepsilon_{m-1}(n) \end{bmatrix} \boldsymbol{\Theta}_{b,m-1}(n) = \begin{bmatrix} \mathcal{B}^{1/2}_{m-1}(n) & 0 \\ p^*_{m-1}(n) & \varepsilon_m(n) \end{bmatrix}$$

3. *Initialization:*

(a) *Auxiliary parameter initialization:* For order $m = 1, 2, \ldots, M$, set

$$p_{f,m-1}(0) = p_{b,m-1}(0) = 0$$

and for order $m = 1, 2, \ldots, M+1$, set

$$p_{m-1}(0) = 0$$

(b) *Soft-constraint initialization:* For order $m = 0, 1, \ldots, M$, set

$$\mathcal{B}_m(-1) = \mathcal{B}_m(0) = \delta$$
$$\mathcal{F}_m(0) = \delta$$

where δ is a small positive constant.

(c) *Data initialization:* For $n = 1, 2, \ldots$, compute

$$\varepsilon_{f,0}(n) = \varepsilon_{b,0}(n) = u(n)$$
$$\varepsilon_0(n) = d(n)$$
$$\gamma_0(n) = 1$$

where $u(n)$ is the input and $d(n)$ is the desired response at adaptation cycle n.

algorithm, as described in Chapter 10. The algorithm proceeds with a set of *initial values* determined by the input datum $u(n)$ and desired response $d(n)$, namely,

$$\varepsilon_{f,0}(n) = \varepsilon_{b,0}(n) = u(n)$$

and

$$\varepsilon_0(n) = d(n).$$

The initial value for the conversion factor is chosen to be

$$\gamma_0(n) = 1.$$

16.9 FUNDAMENTAL PROPERTIES OF THE QRD-LSL FILTER

The QRD-LSL algorithm summarized in Table 16.4 is so called in recognition of three facts. First, the unitary transformations in Eqs. (16.84), (16.95), and (16.106), in which the (1, 2) entry of each postarray is reduced to zero, are all examples of the *QR-decomposition.* Second, the algorithm is rooted in *recursive least-squares (RLS) estimation*, supported by Kalman filter theory. Third, the computations performed by the algorithm proceed on a stage-by-stage fashion, with each stage having the form of a *lattice*. By virtue of these facts, the QRD-LSL algorithm is endowed with a highly desirable set of operational and implementational characteristics:

- *Good numerical properties*, which are inherited from the QR-decomposition part of the algorithm.
- *Good convergence properties* (i.e., a fast rate of convergence and insensitivity to variations in the eigenvalue spread of the underlying correlation matrix of the input data), which are due to the RLS nature of the algorithm.
- A *high level of computational efficiency*, which results from the modular, lattice-like structure of the prediction process.

The unique combination of these characteristics makes the QRD-LSL algorithm a powerful adaptive filtering algorithm.

The latticelike structure of the QRD-LSL algorithm, using a sequence of Givens rotations, is clearly illustrated by the multistage signal-flow graph of Fig. 16.9. In particular, stage m of the predictor section of the algorithm involves the computations of the angle-normalized prediction errors $\varepsilon_{f,m}(n)$ and $\varepsilon_{b,m}(n)$, where the prediction order $m = 0, 1, 2, \ldots, M$. By the same token, the filtering section of the algorithm involves the computation of the angle-normalized joint-process estimation error $\varepsilon_m(n)$, where $m = 0, 1, 2, \ldots, M + 1$. The details of these computations are depicted in the signal-flow graphs shown separately in Fig. 16.10, which further emphasize the inherent lattice nature of the QRD-LSL algorithm.

The boxes labeled $z^{-1}\mathbf{I}$ in Fig. 16.9 signify *storage*, which is needed to accommodate the fact that the Givens rotations involved in computing the backward prediction errors are delayed by one adaptation cycle with respect to those involved in the joint-process

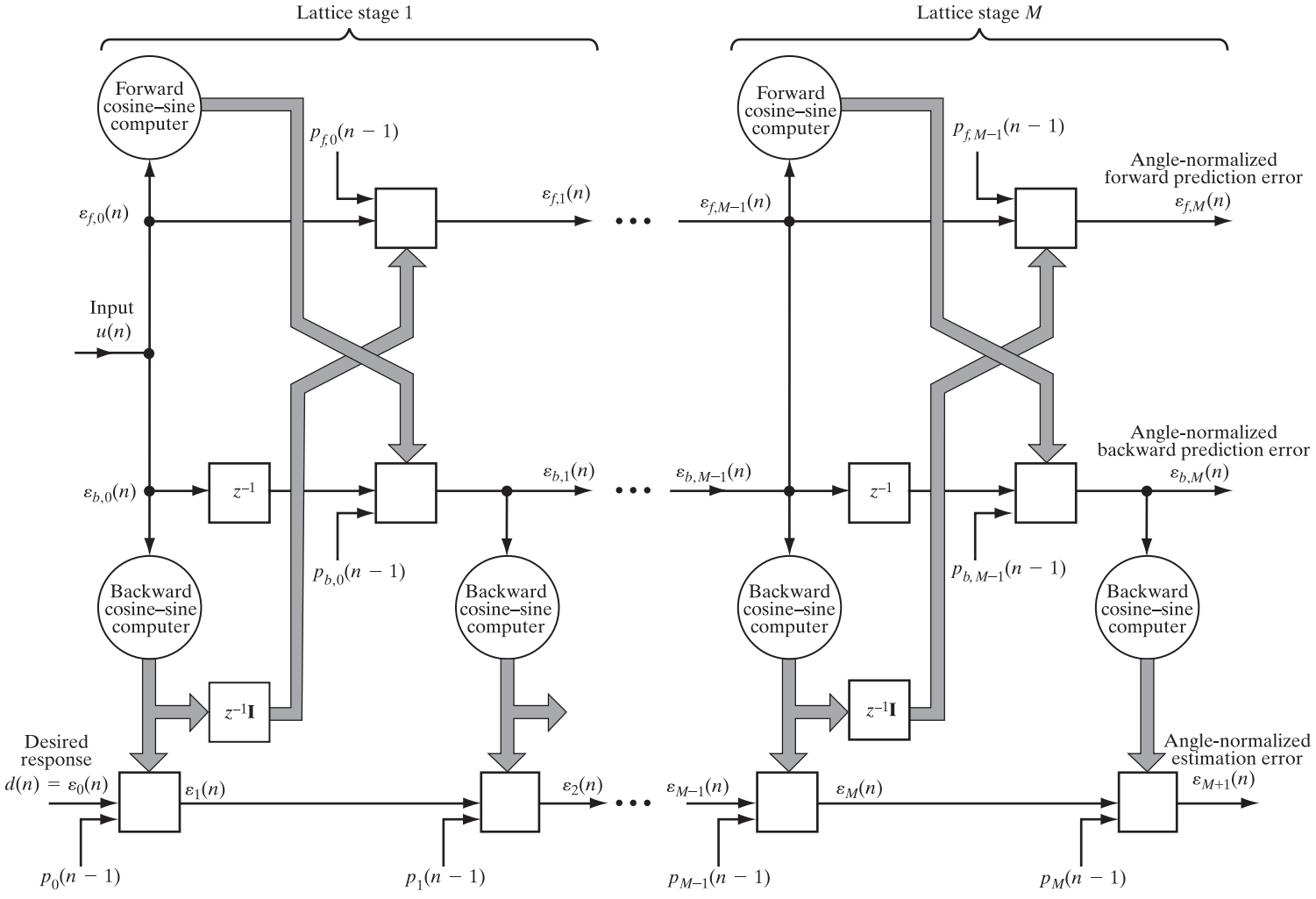


FIGURE 16.9 Signal-flow graph of the QRD-LSL algorithm. For the unlabeled processing units, see the signal-flow graphs of Fig. 16.10.

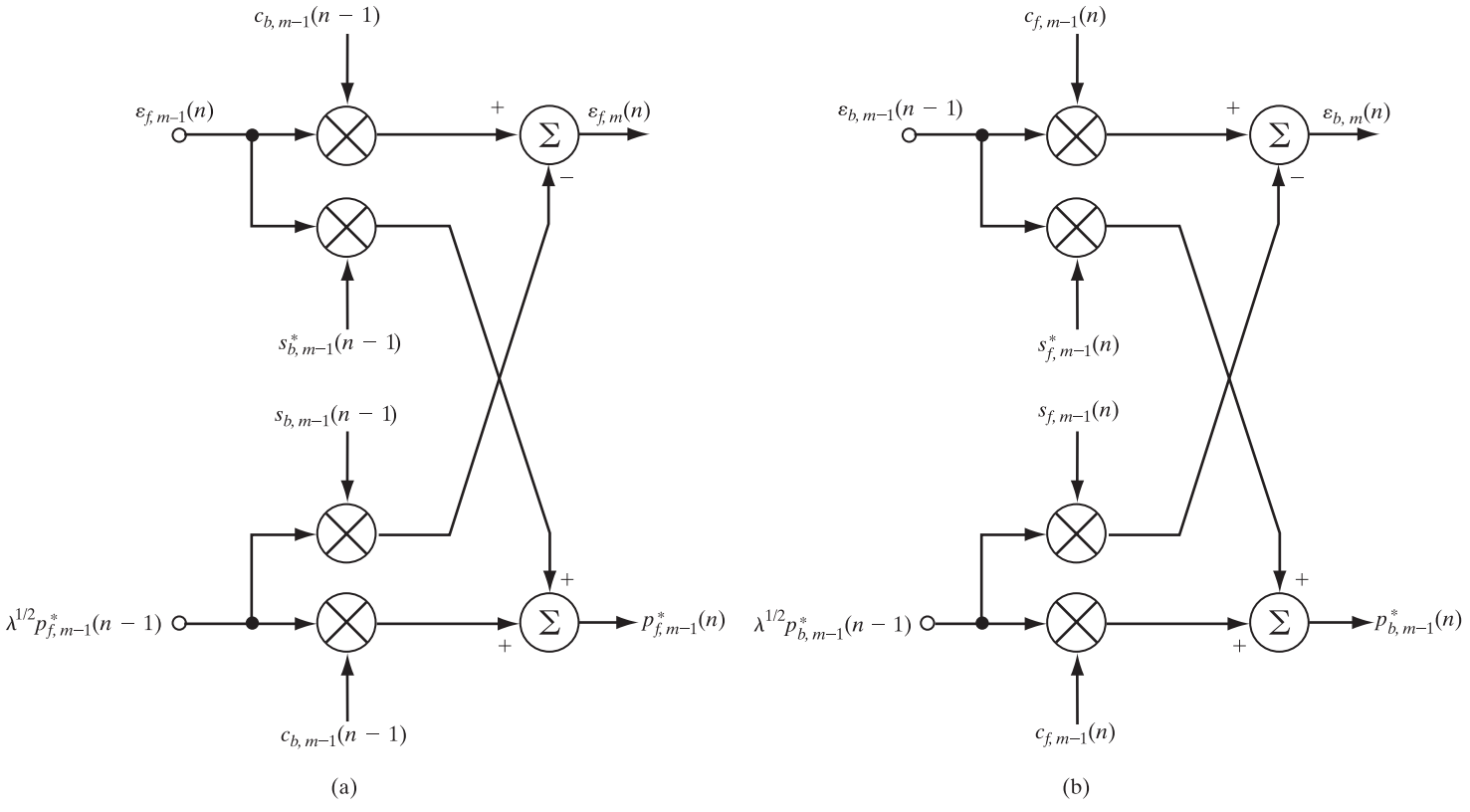
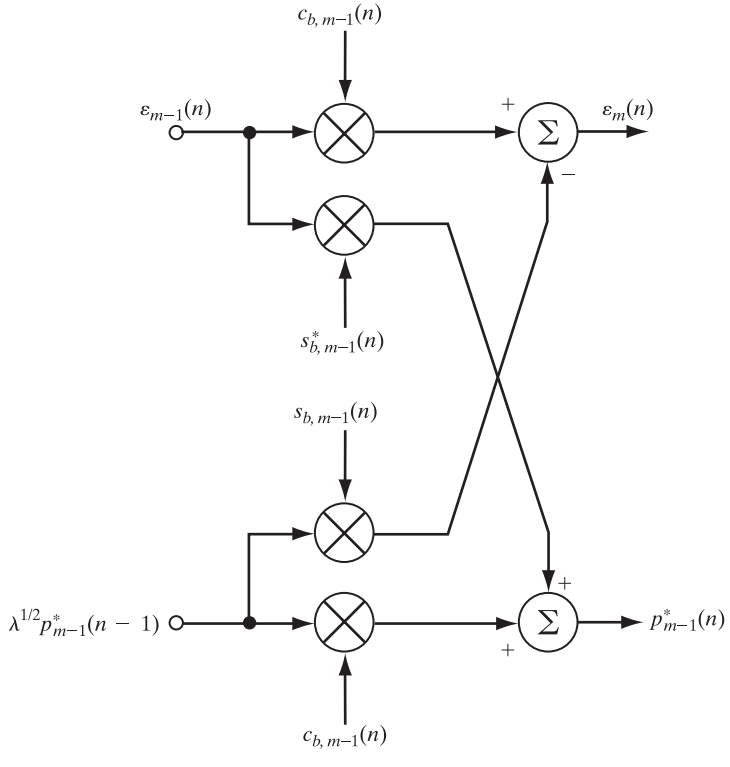


FIGURE 16.10 Signal-flow graphs for computing normalized variables in the QRD-LSL algorithm: (a) angle-normalized forward prediction error $\varepsilon_{f,\,m}(n)$; (b) angle-normalized backward prediction error $\varepsilon_{b,\,m}(n)$. Part (c) of the figure is shown on the next page.



(c)

FIGURE 16.10 (*continued*) (c) Angle-normalized joint-process estimation error $\varepsilon_m(n)$.

estimation process. Note, however, that the joint-process estimation process involves one last Givens rotation, all by itself.

From the signal-flow graph of Fig. 16.9, we clearly see that the total number of Givens rotations needed for the computation of $\varepsilon_{M+1}(n)$ is $2M+1$, which increases *linearly* with the final prediction order M. However, the price paid for the high level of computational efficiency of the fast algorithm described herein, compared with the conventional RLS algorithm described in Chapter 10, is that of having to write a more elaborate set of algorithmic instructions, which could limit its practical applications.

Examining the signal-flow graphs of Figs. 16.9 and 16.10, we may identify two different sets of recursions in the formulation of the QRD-LSL algorithm:

1. *Order updates (recursions).* At each stage of the algorithm, *order updates* are performed on the angle-normalized estimation errors. Specifically, a series of M order updates applied to the initial values $\varepsilon_{f,o}(n)$ and $\varepsilon_{b,o}(n)$ yields the final values $\varepsilon_{f,M}(n)$ and $\varepsilon_{b,M}(n)$, respectively, where M is the final prediction order. To compute the final value of the angle-normalized joint-process estimation error $\varepsilon_{M+1}(n)$, another series of $M+1$ order updates is applied to the initial value $\varepsilon_0(n)$. This latter set of updates includes the use of the sequence of angle-normalized backward prediction errors $\varepsilon_{b,0}(n), \varepsilon_{b,1}(n), \ldots, \varepsilon_{b,M}(n)$. The final order update pertains to the computation of the square root of the conversion factor $\gamma_{M+1}^{1/2}(n)$, which involves the application of $M+1$ order updates to the initial value $\gamma_0^{1/2}(n)$. The availability of the final values $\varepsilon_{M+1}(n)$ makes it possible to compute the final value $e_{M+1}(n)$ of the joint-process estimation error.
2. *Time updates (recursions).* The computations of $\varepsilon_{f,m}(n)$ and $\varepsilon_{b,m}(n)$ as outputs of predictor stage m of the algorithm involve the use of the auxiliary parameters $p_{f,m-1}(n)$ and $p_{b,m-1}(n)$, respectively, for $m = 1, 2, \ldots, M$. Similarly, the computation of $\varepsilon_m(n)$ involves the auxiliary parameter $p_{m-1}(n)$ for $m = 1, 2, \ldots, M+1$. The computations of these three auxiliary parameters themselves have the following common features:
   - They are all governed by *first-order difference equations*.
   - The coefficients of the equation are *time varying*. For exponential weighting (i.e., $\lambda \leq 1$), the coefficients are bounded in absolute value by unity. Hence, the solution of the equation converges.
   - The term playing the role of excitation is represented by some form of an estimation error.
   - In the prewindowing method, all three auxiliary parameters are set equal to zero for $n \leq 0$.

   Consequently, the auxiliary parameters $p_{f,m-1}(n)$ and $p_{b,m-1}(n)$ for $m = 1, 2, \ldots, M$, and $p_m(n)$ for $m = 0, 1, \ldots, M$, may be computed recursively in time.

Finally, and perhaps most importantly, the angle-normalized QRD-LSL algorithm summarized in Table 16.4 plays a central role in the derivation of the whole family of recursive LSL algorithms. We say this because all other existing recursive LSL algorithms

using a posteriori estimation errors or a priori estimation errors (or combinations thereof) may be viewed as rewritings of the QRD-LSL algorithm. The validity of this statement will be demonstrated in Sections 16.11 and 16.12 dealing with two different recursive LSL algorithms.

16.10 COMPUTER EXPERIMENT ON ADAPTIVE EQUALIZATION

In this computer experiment, we study the use of the QRD-LSL algorithm for the *adaptive equalization* of a linear channel that produces unknown distortion. The parameters of the channel are the same as those used to study the RLS algorithm in Section 10.9 for a similar application. The results of the experiment should therefore help us to make an assessment of the performance of this order-recursive algorithm, compared with the traditional RLS algorithm.

The parameters of the QRD-LSL algorithm studied here are identical to those used for the RLS algorithm in Section 10.9:

Exponential weighting factor:	$\lambda = 1$
Prediction order:	$M = 10$
Number of equalizer taps:	$M + 1 = 11$
Regularization parameter:	$\delta = 0.004$

The computer simulations were run for four different values of the channel parameter W defined in Eq. (6.112), namely, $W = 2.9, 3.1, 3.3$, and 3.5. These values of W correspond to the following eigenvalue spreads of the underlying correlation matrix $\mathbf{R}$ of the channel output (equalizer input): $\chi(\mathbf{R}) = 6.78, 11.124, 21.713$, and 46.822, respectively. The signal-to-noise ratio measured at the channel output was 30 dB. (For more details of the experimental setup, the reader is referred to Sections 6.8 and 10.9.)

Learning Curves

Figure 16.11 presents the superposition of learning curves of the QRD-LSL algorithm for the four said values of the channel parameter $W = 2.9, 3.1, 3.3$, and 3.5. Each learning curve was obtained by ensemble-averaging the squared value of the final a priori estimation error (i.e., the innovation) $\xi_{M+1}(n)$ over 200 independent Monte Carlo runs of the experiment for a final prediction order $M = 10$. To compute $\xi_{M+1}(n)$, we use Eq. (16.61) for $m = M + 1$ and thus write

$$\xi_{M+1}(n) = \frac{\varepsilon_{M+1}(n)}{\gamma_{M+1}^{1/2}(n)},$$

where $\varepsilon_{M+1}(n)$ is the final value of the angle-normalized joint-process estimation error and γ_{M+1} is the associated conversion factor.

For each eigenvalue spread, the learning curve of the QRD-LSL algorithm follows a path practically identical to that of the RLS algorithm once the initialization is

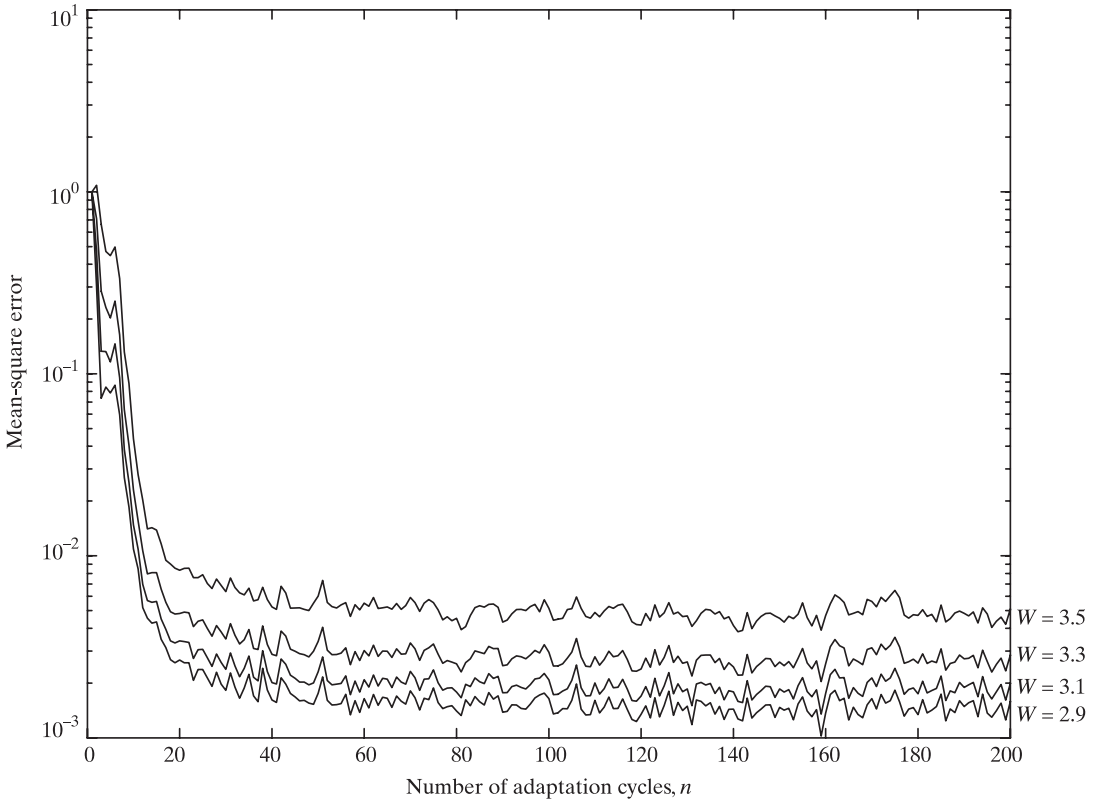


FIGURE 16.11 Learning curves of the QRD-LSL algorithm for the adaptive equalization experiment.

completed. This relationship is readily confirmed by comparing the plots of Fig. 16.11 with those of Fig. 10.6. (In both cases, double-precision arithmetic was used so that finite-precision effects are negligible.)

Note also that in computing the plots of Fig. 16.11, the transients inherent to the evolution of the conversion factor $\gamma_{M+1}(n)$ were removed from the computation of the innovation $\xi_{M+1}(n)$ during the initialization period.

Conversion Factor

In Fig. 16.12, we show the superposition of four ensemble-averaged plots of the conversion factor $\gamma_{M+1}(n)$ (for the final stage) versus the number of adaptation cycles, n, corresponding to the four different values of the eigenvalue spread $\chi(\mathbf{R})$ specified on the previous page. The curves plotted in the figure were obtained by ensemble-averaging $\gamma_{M+1}(n)$ over 200 independent Monte Carlo runs of the experiment. It is noteworthy that after the initial transients have died out, the time variation of this

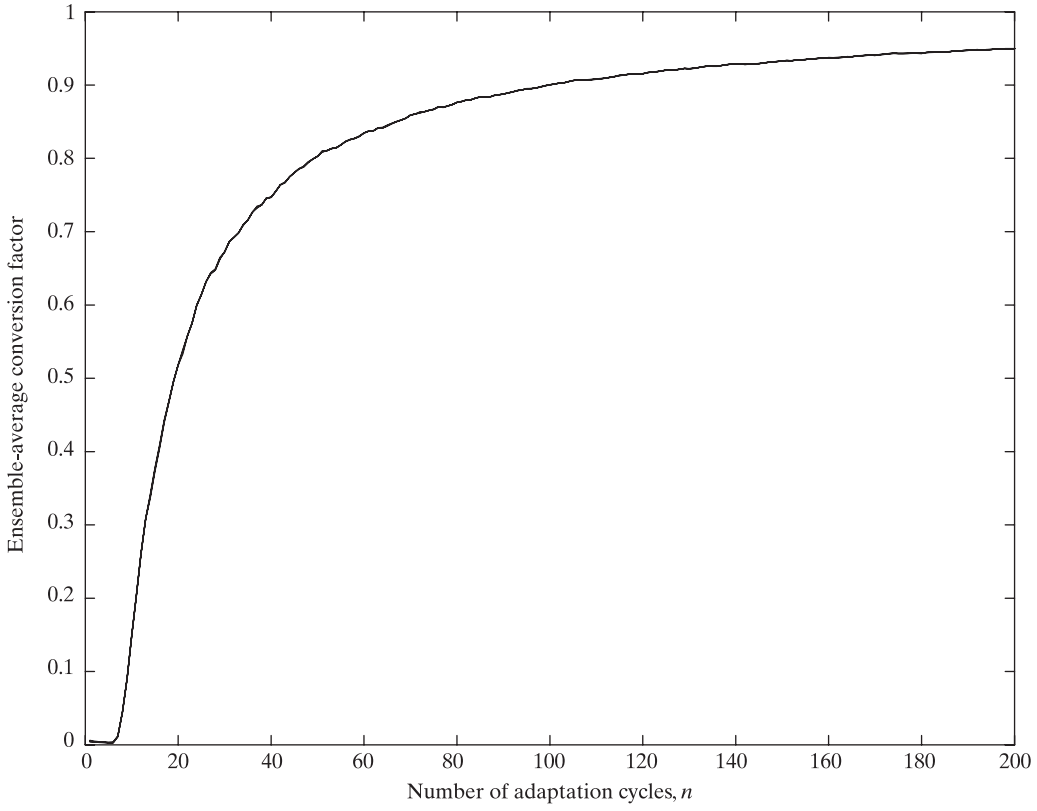


FIGURE 16.12 Ensemble-average conversion factor $\gamma_{M+1}(n)$ for varying eigenvalue spread.

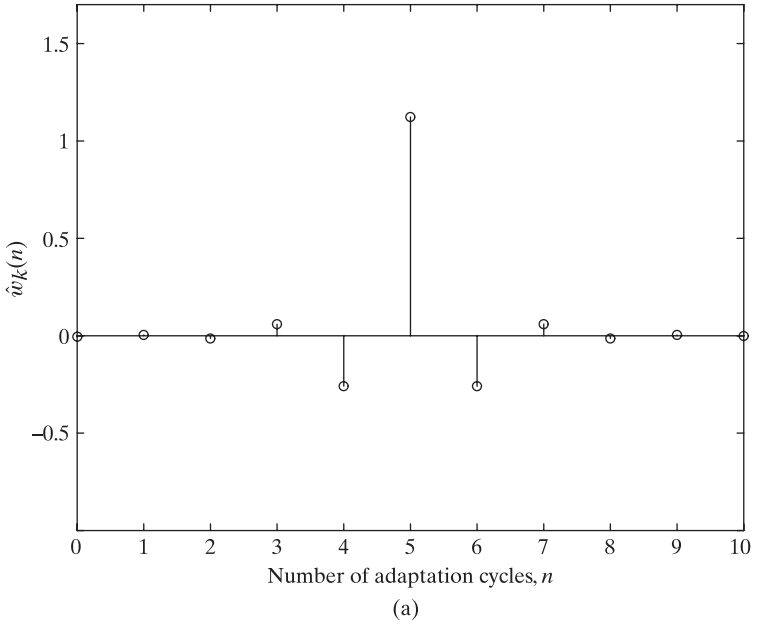
ensemble-average conversion factor $\mathbb{E}[\gamma_m(n)]$ follows an *inverse* law, as evidenced by the approximation

$$\mathbb{E}[\gamma_m(n)] \approx 1 - \frac{m}{n} \qquad \text{for } m = 1, 2, \ldots, M+1 \text{ and } n \geq m.$$

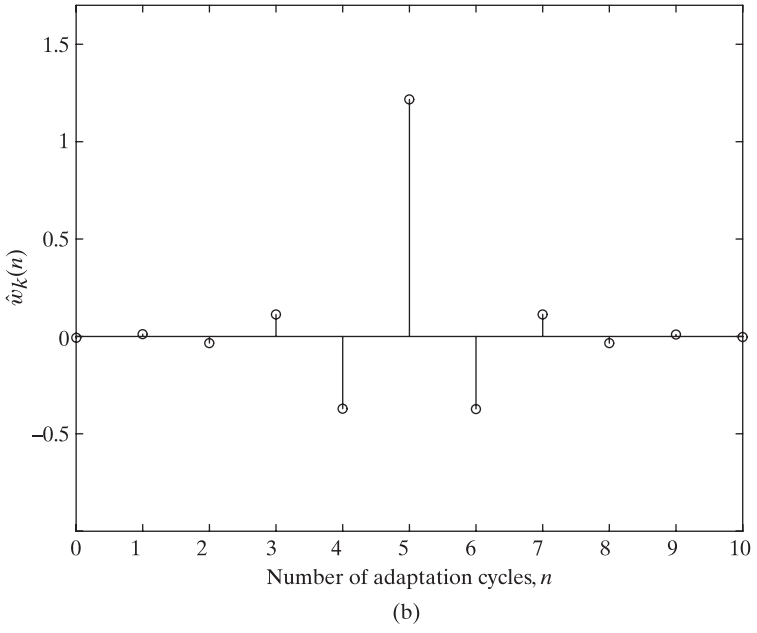
This equation provides a good fit to the experimentally computed curve shown in Fig. 16.12, particularly for n large compared with the predictor order $m = M + 1$. The reader is invited to check the validity of this fit. Note also that the experimental plots of the conversion factor $\gamma_{M+1}(n)$ are insensitive to variations in the eigenvalue spread of the correlation matrix of the equalizer input for $n \geq 10$.

Impulse Response

In Fig. 16.13, we have plotted the ensemble-average impulse response of the adaptive equalizer after $n = 500$ adaptation cycles for each of the four eigenvalue spreads. As before, the ensemble averaging was performed over 200 independent Monte Carlo runs of the experiment. The results for the QRD-LSL algorithm are, for all practical purposes, indistinguishable from the corresponding theoretic values of the channel's impulse response.

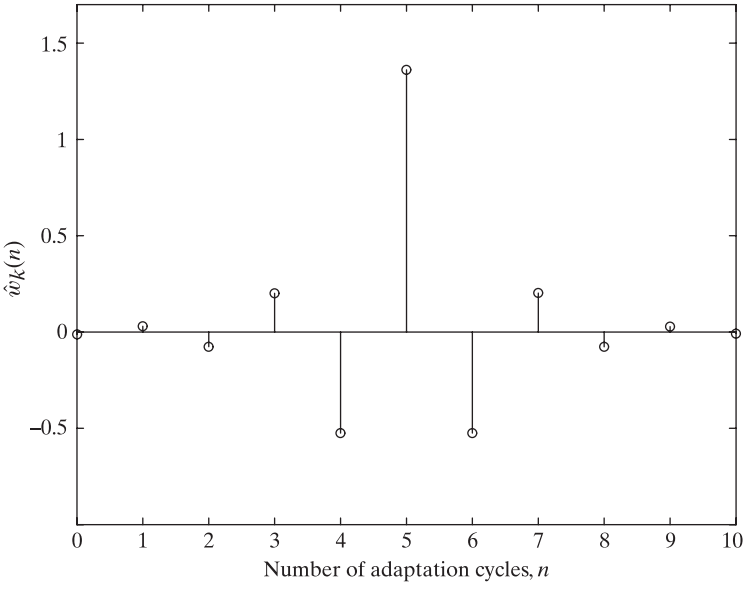


(a)

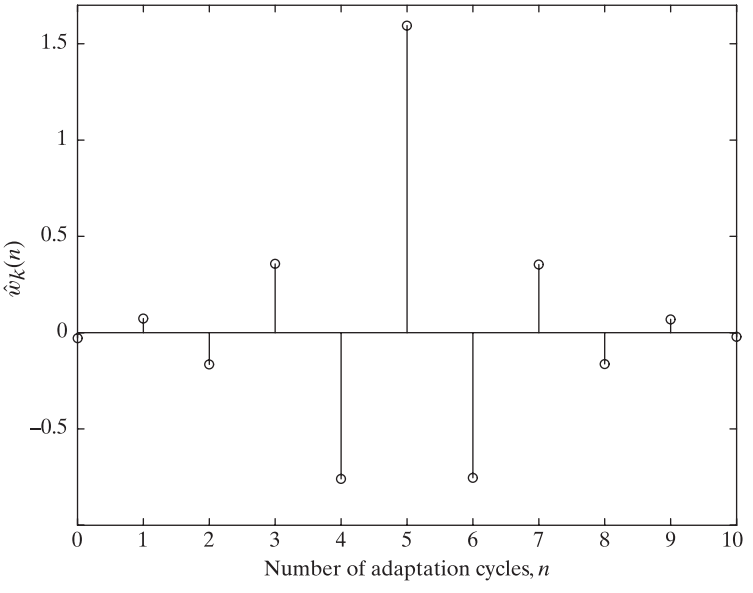


(b)

FIGURE 16.13 Ensemble-average impulse response of the adaptive equalizer for varying eigenvalue spread: (a) $W = 2.9$, $\chi(\mathbf{R}) = 6.0782$; (b) $W = 3.1$, $\chi(\mathbf{R}) = 11.1238$, (c) $W = 3.3$, $\chi(\mathbf{R}) = 21.7132$; (d) $W = 3.5$, $\chi(\mathbf{R}) = 46.8216$. Parts (c) and (d) of the figure are presented on the next page.



(c)



(d)

FIGURE 16.13 (*continued*)

16.11 RECURSIVE (LSL) FILTERS USING A POSTERIORI ESTIMATION ERRORS

In a generic sense, the family of recursive LSL algorithms may be divided into two subgroups: those that involve the use of unitary rotations and those that do not. A well-known algorithm that belongs to the latter subgroup is the traditional *recursive LSL algorithm using a posteriori estimation errors* (Morf, 1974; Morf & Lee, 1978; Lee et al., 1981). The estimation errors used in this algorithm are represented by the a posteriori forward prediction error $f_m(n)$, the a posteriori backward prediction error $b_m(n)$, and the a posteriori joint-process estimation error $e_m(n)$, where $m = 0, 1, 2, \ldots, M$, as shown in the signal-flow graphs of Figs. 16.9 and 16.10.

We may derive the recursive LSL algorithm using a posteriori estimation errors (and for that matter, other recursive LSL algorithms) from the QRD-LSL algorithm of Table 16.4 by using a simple two-step procedure:

- The three arrays of the QRD-LSL algorithm, dealing with adaptive forward prediction, adaptive backward prediction, and adaptive joint-process estimation, are squared; the effects of unitary rotations are thereby completely removed from the algorithm.
- Certain terms (depending on the algorithm of interest) on both sides of the resultant arrays are retained and then compared.

Examples of this procedure were presented in Chapter 15, on square-root adaptive filters.

Applying the foregoing procedure to the three arrays of Table 16.4 that describe the angle-normalized QRD-LSL algorithm, and expressing the results in terms of the a posteriori estimation errors, we get the following three sets of recursions:

1. *Adaptive forward prediction*:

$$\mathcal{B}_{m-1}(n-1) = \lambda \mathcal{B}_{m-1}(n-2) + \frac{|b_{m-1}(n-1)|^2}{\gamma_{m-1}(n-1)}; \tag{16.114}$$

$$\Delta_{m-1}(n) = \lambda \Delta_{m-1}(n-1) + \frac{b_{m-1}(n-1) f^*_{m-1}(n)}{\gamma_{m-1}(n-1)}; \tag{16.115}$$

$$\kappa_{f,m} = -\frac{\Delta_{m-1}(n)}{\mathcal{B}_{m-1}(n-1)}; \tag{16.116}$$

$$f_m(n) = f_{m-1}(n) + \kappa^*_{f,m}(n) b_{m-1}(n-1); \tag{16.117}$$

$$\gamma_m(n-1) = \gamma_{m-1}(n-1) - \frac{|b_{m-1}(n-1)|^2}{\mathcal{B}_{m-1}(n-1)}. \tag{16.118}$$

Note that in the second line we have made use of the relation between $\Delta_{m-1}(n)$ and $p_{f,m-1}(n)$ given in Eq. (16.86), and in the third line we have made use of the definition of the forward reflection coefficient $\kappa_{f,m}(n)$ given in Eq. (16.88).

2. *Adaptive backward prediction*:

$$\mathcal{F}_{m-1}(n) = \lambda \mathcal{F}_{m-1}(n-1) + \frac{|f_{m-1}(n)|^2}{\gamma_{m-1}(n-1)}; \tag{16.119}$$

$$\kappa_{b,m}(n) = -\frac{\Delta^*_{m-1}(n)}{\mathcal{F}_{m-1}(n)}; \tag{16.120}$$

$$b_m(n) = b_{m-1}(n-1) + \kappa^*_{b,m}(n) f_{m-1}(n). \tag{16.121}$$

Here, in the second line we have made use of the relation between $\Delta_{m-1}(n)$ and $p_{b,m-1}(n)$ given in Eq. (16.98) and also the definition of the backward reflection coefficient $\kappa_{b,m}(n)$ given in Eq. (16.99).

3. *Adaptive joint-process estimation*:

$$\pi_{m-1}(n) = \lambda \pi_{m-1}(n-1) + \frac{b_{m-1}(n) e^*_{m-1}(n)}{\gamma_{m-1}(n)}; \tag{16.122}$$

$$h_{m-1}(n) = \frac{\pi_{m-1}(n)}{\mathcal{B}_{m-1}(n)}; \tag{16.123}$$

$$e_m(n) = e_{m-1}(n) - h^*_{m-1}(n) b_{m-1}(n). \tag{16.124}$$

Here, $\pi_{m-1}(n)$ is defined in terms of $p_{m-1}(n)$ by

$$\pi_{m-1}(n) = \mathcal{B}^{1/2}_{m-1}(n) p_{m-1}(n) \tag{16.125}$$

and the joint-process regression coefficient $h_{m-1}(n)$ is defined in terms of $p_{m-1}(n)$ by

$$h_{m-1}(n) = \frac{p_{m-1}(n)}{\mathcal{B}^{1/2}_{m-1}(n)}. \tag{16.126}$$

Summary of the Recursive LSL Algorithm Using A Posteriori Estimation Errors

The complete list of order- and time-update recursions constituting the recursive LSL algorithm (based on a posteriori estimation errors) is presented in Table 16.5. Since the LSL algorithm summarized in the table involves division by updated parameters at some of the steps, care must be taken to ensure that these values are not allowed to become too small. Unless a high-precision computer is used, selection of the regularization parameter δ [determining the initial values $\mathcal{F}_0(0)$ and $\mathcal{B}_0(0)$] may have a severe effect on the initial transient performance of the LSL algorithm. Friedlander (1982) suggests using some form of *thresholding* wherein, if the divisor (in any computation of an LSL algorithm) is less than a preassigned threshold, the corresponding term involving that divisor is set to zero. This remark also applies to other versions of the recursive LSL algorithm (e.g., those summarized later in Table 16.6).

TABLE 16.5 Summary of the Recursive LSL Algorithm Using A Posteriori Estimation Errors

Predictions:
For $n = 1, 2, 3, \ldots$, compute the various order updates in the sequence $m = 1, 2, \ldots, M$, where M is the final order of the LSL predictor:

$$\begin{aligned}
\Delta_{m-1}(n) &= \lambda\Delta_{m-1}(n-1) + \frac{b_{m-1}(n-1)f^*_{m-1}(n)}{\gamma_{m-1}(n-1)} \\
\mathcal{B}_{m-1}(n-1) &= \lambda\mathcal{B}_{m-1}(n-2) + \frac{|b_{m-1}(n-1)|^2}{\gamma_{m-1}(n-1)} \\
\mathcal{F}_{m-1}(n) &= \lambda\mathcal{F}_{m-1}(n-1) + \frac{|f_{m-1}(n)|^2}{\gamma_{m-1}(n-1)} \\
\kappa_{f,m}(n) &= -\frac{\Delta_{m-1}(n)}{\mathcal{B}_{m-1}(n-1)} \\
\kappa_{b,m}(n) &= -\frac{\Delta^*_{m-1}(n)}{\mathcal{F}_{m-1}(n)} \\
f_m(n) &= f_{m-1}(n) + \kappa^*_{f,m}(n)b_{m-1}(n-1) \\
b_m(n) &= b_{m-1}(n-1) + \kappa^*_{b,m}(n)f_{m-1}(n) \\
\gamma_m(n-1) &= \gamma_{m-1}(n-1) - \frac{|b_{m-1}(n-1)|^2}{\mathcal{B}_{m-1}(n-1)}
\end{aligned}$$

Filtering:
For $n = 1, 2, 3, \ldots$, compute the various order updates in the sequence $m = 1, 2, \ldots, M+1$:

$$\begin{aligned}
\pi_{m-1}(n) &= \lambda\pi_{m-1}(n-1) + \frac{b_{m-1}(n)e^*_{m-1}(n)}{\gamma_{m-1}(n)} \\
h_{m-1}(n) &= \frac{\pi_{m-1}(n)}{\mathcal{B}_{m-1}(n)} \\
e_m(n) &= e_{m-1}(n) - h^*_{m-1}(n)b_{m-1}(n)
\end{aligned}$$

Initialization:

1. To initialize the algorithm, at adaptation cycle $n = 0$, set

$$\begin{aligned}
\Delta_{m-1}(0) &= 0 \\
\mathcal{F}_{m-1}(0) &= \delta \qquad \delta = \text{small positive constant} \\
\mathcal{B}_{m-1}(-1) &= \delta \\
\gamma_0(0) &= 1
\end{aligned}$$

2. At each instant $n \geq 1$, generate the various zeroth-order variables as follows:

$$\begin{aligned}
f_0(n) &= b_0(n) = u(n) \\
\mathcal{F}_0(n) &= \mathcal{B}_0(n) = \lambda\mathcal{F}_0(n-1) + |u(n)|^2 \\
\gamma_0(n-1) &= 1
\end{aligned}$$

3. For joint-process estimation, at adaptation cycle $n = 0$, initialize the algorithm by setting

$$\pi_{m-1}(0) = 0$$

At each adaptation cycle $n \geq 1$, generate the zeroth-order variable

$$e_0(n) = d(n)$$

Note: For prewindowed data, the input $u(n)$ and the desired response $d(n)$ are both zero for $n \leq 0$.

Initialization of the Recursive LSL Algorithm

To initialize the recursive LSL algorithm using a posteriori estimation errors, we start with the elementary case of zero prediction order, for which [see Eq. (16.43)]

$$f_0(n) = b_0(n) = u(n),$$

where $u(n)$ is the lattice predictor input at adaptation cycle n.

The remaining initial values pertain to the sums of weighted a posteriori prediction-error squares for zero prediction order. Specifically, setting $m = 0$ in Eq. (16.11) yields

$$\mathcal{F}_0(n) = \lambda \mathcal{F}_0(n-1) + |u(n)|^2. \tag{16.127}$$

Similarly, setting $m = 0$ in Eq. (16.22) yields

$$\mathcal{B}_0(n) = \lambda \mathcal{B}_0(n-1) + |u(n)|^2. \tag{16.128}$$

With the conversion factor $\gamma_m(n-1)$ bounded by zero and unity, a logical choice for the zeroth-order value of that parameter is

$$\gamma_0(n-1) = 1. \tag{16.129}$$

We complete the initialization of the algorithm for forward and backward prediction by applying the conditions

$$\Delta_{m-1}(0) = 0 \tag{16.130}$$

and

$$\mathcal{F}_{m-1}(0) = \mathcal{B}_{m-1}(-1) = \delta \tag{16.131}$$

at adaptation cycle $n = 0$. The small regularization parameter δ is used to ensure nonsingularity of the correlation matrix $\mathbf{\Phi}_m(n)$.

Turning finally to the initialization of the joint-estimation process, we see (for zero-prediction order) that

$$e_0(n) = d(n),$$

where $d(n)$ is the desired response. Thus, to initiate this part of the computation, we generate $e_0(n)$ for each adaptation cycle n. To complete the initialization of the recursive LSL algorithm for joint-process estimation, at adaptation cycle $n = 0$ we set

$$\pi_{m-1}(0) = 0 \quad \text{for } m = 1, 2, \ldots, M+1.$$

Table 16.5 includes the initialization of the recursive LSL algorithm as just described.

16.12 RECURSIVE LSL FILTERS USING A PRIORI ESTIMATION ERRORS WITH ERROR FEEDBACK

In this section, we derive another recursive LSL algorithm, which differs from that of Section 16.11 in two respects. First, it is based on a priori estimation errors; second, the reflection and joint-process estimation coefficients are all derived directly. The latter

difference has significant practical implications when the use of finite-precision arithmetic is the preferred method of implementation.

The algorithm is called the *recursive LSL algorithm using a priori estimation errors with error feedback* (Ling & Proakis, 1984a; Ling et al., 1985). This algorithm can be derived in several ways, the two most common of which are:

1. Apply the squaring procedure to an extended form of the QRD-LSL algorithm.
2. Apply Kalman filter theory in conjunction with Table 14.4 on correspondences between Kalman variables and angle-normalized LSL variables.

We will follow Procedure 2, as it is insightful and fairly straightforward. Discussion of Procedure 1 is presented in Haykin (1996).

To proceed with the derivation of the algorithm, we first recall the state-update equation for the Kalman filter, which is reproduced here:

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \lambda^{-1/2}\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{g}(n)\alpha(n). \tag{16.132}$$

(See the third line, under computation, in Table 14.4.) Next, we note the following correspondeces between Kalman filter and LSL filter variables for the case of forward prediction of order $m-1$:

$$\begin{aligned}\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) &\leftrightarrow -\lambda^{-n/2}\kappa_{f,m}(n-1);\\ \mathbf{g}(n) &\leftrightarrow \lambda^{-1/2}\mathcal{B}_{m-1}^{-1}(n-1)\varepsilon_{b,m-1}(n-1);\\ \alpha(n) &\leftrightarrow \lambda^{-n/2}\gamma_{m-1}^{1/2}(n-1)\eta_m^*(n).\end{aligned}$$

[See line 3 of Table 16.3 and Eqs. (16.77) and (16.79).] Substituting these correspondeces into Eq. (16.132), using Eq. (16.60), and cancelling common terms, we get

$$\kappa_{f,m}(n) = \kappa_{f,m}(n-1) - \left(\frac{\gamma_{m-1}(n-1)\beta_{m-1}(n-1)}{\mathcal{B}_{m-1}(n-1)}\right)\eta_m^*(n), \quad m = 1, 2, \ldots, M, \tag{16.133}$$

which now permits the recursive computation of the forward reflection coefficient directly in terms of the order-updated forward a priori prediction error $\eta_m(n)$ and the delayed a priori backward prediction error $\beta_{m-1}(n-1)$. From Eq. (16.57), we find that $\eta_m(n)$ depends on $\kappa_{f,m}(n-1)$. We see therefore that the second term on the right-hand side of Eq. (16.133) applies *error feedback* to the computation of $\kappa_{f,m}(n)$.

Proceeding in a similar manner, we may show that the corresponding recursion for computing the backward reflection coefficient for prediction order $m-1$ is as follows:

$$\kappa_{b,m}(n) = \kappa_{b,m}(n-1) - \left(\frac{\gamma_{m-1}(n-1)\eta_{m-1}(n)}{\mathcal{F}_{m-1}(n)}\right)\beta_m^*(n), \quad m = 1, 2, \ldots, M. \tag{16.134}$$

Likewise, we may show that the recursion for computing the corresponding joint-process estimation coefficient is as follows:

$$h_{m-1}(n) = h_{m-1}(n-1) + \left(\frac{\gamma_{m-1}(n)\beta_{m-1}(n)}{\mathcal{B}_{m-1}(n)}\right)\xi_m^*(n), \quad m = 1, 2, \ldots, M+1. \tag{16.135}$$

The quantities inside the outer parentheses pertaining to the second terms on the right-hand sides of Eqs. (16.133) through (16.135) act as gain factors for the recursions for forward prediction, backward prediction, and joint-process estimation, respectively. [The derivations of Eqs. (16.134) and (16.135) are presented as Problem 17.]

Summary of the Recursive LSL Algorithm Using A Priori Estimation Errors with Error Feedback

Table 16.6 summarizes the computations involved in the forward and backward prediction parts of the recursive LSL algorithm using estimation errors with error feedback. The table also includes the computations involved in the joint-process estimation part of the algorithm. The basis for the entries summarized therein is as follows:

1. Predictions:
   - Lines 1 and 2 follow from the combined use of Eqs. (16.11), (16.22), (16.28), and (16.29).
   - Lines 3 and 4 are repeats of Eqs. (16.57) and (16.58), respectively.
   - Lines 5 and 6 are repeats of Eqs. (16.133) and (16.134), respectively.
   - Finally, line 7 follows from the combined use of Eqs. (16.29) and (16.118).
2. Filtering:
   - Line 1 follows from the combined use of Eqs. (16.27), (16.29), and (16.55).
   - Line 2 is a repeat of Eq. (16.135).
3. Initialization: The initializing conditions are essentially the same as those in Table 16.5.

Comparison of the First and Second Recursive LSL Algorithms

Figure 16.14 presents a signal-flow graph of the second recursive LSL algorithm, emphasizing that order updating of the variables of interest (i.e., a priori forward prediction, backward prediction, and joint-process estimation errors) at adaptation cycle n requires knowledge of the forward reflection coefficients, backward reflection coefficients, and regression coefficients at the previous adaptation cycle $n - 1$.

An important difference between the two recursive LSL algorithms summarized in Tables 16.5 and 16.6 is the way in which the reflection coefficients and regression coefficients are updated. In the case of Table 16.5, the updating is performed *indirectly*. First, we compute the cross-correlation between forward and delayed backward prediction errors and the cross-correlation between backward prediction errors and joint-process estimation errors. Next, we compute the sum of weighted forward prediction-error squares and the sum of weighted backward-error squares. Finally, we compute the reflection and regression coefficients by dividing a cross-correlation by a sum of weighted prediction-error squares. By contrast, in Table 16.6, the updating of the reflection and regression coefficients is performed *directly*. The differences between indirect and direct forms of updating, as described herein, have an important bearing on the numerical behavior of these recursive LSL algorithms; this issue is discussed in Section 16.14.

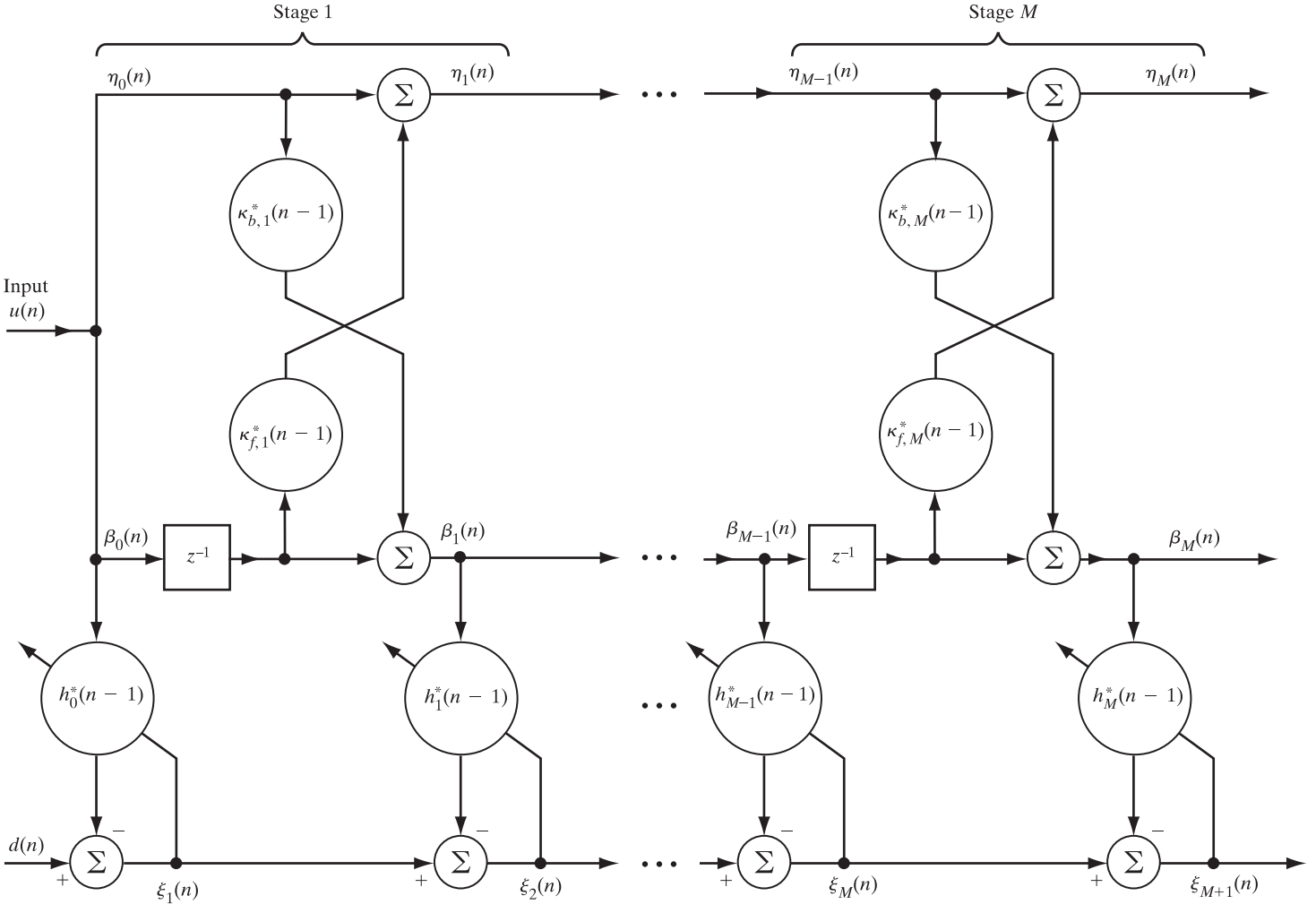


FIGURE 16.14 Joint-process estimator using the recursive LSL algorithm based on a priori estimation errors. The figure displays error feedback acting on the regression coefficients. The corresponding actions of error feedback on the forward and backward reflection coefficients are not shown so as to keep the figure simple.

Relationship Between the Recursive LSL Algorithm with Error Feedback and the GAL Algorithm

One last comment is in order—the GAL algorithm of Table 5.2 may be viewed as a simplified form of the recursive LSL algorithm of Table 16.6 under the following special set of conditions (see Problem 19):

TABLE 16.6 Summary of the Recursive LSL Algorithm Using A Priori Estimation Errors with Error Feedback

Predictions:
For $n = 1, 2, 3, \ldots$, compute the various order updates in the sequence $m = 1, 2, \ldots, M$, where M is the final order of the least-squares predictor:

$$\mathcal{F}_{m-1}(n) = \lambda \mathcal{F}_{m-1}(n-1) + \gamma_{m-1}(n-1)|\eta_{m-1}(n)|^2$$

$$\mathcal{B}_{m-1}(n-1) = \lambda \mathcal{B}_{m-1}(n-2) + \gamma_{m-1}(n-1)|\beta_{m-1}(n-1)|^2$$

$$\eta_m(n) = \eta_{m-1}(n) + \kappa_{f,m}^*(n-1)\beta_{m-1}(n-1)$$

$$\beta_m(n) = \beta_{m-1}(n-1) + \kappa_{b,m}^*(n-1)\eta_{m-1}(n)$$

$$\kappa_{f,m}(n) = \kappa_{f,m}(n-1) - \frac{\gamma_{m-1}(n-1)\beta_{m-1}(n-1)}{\mathcal{B}_{m-1}(n-1)}\eta_m^*(n)$$

$$\kappa_{b,m}(n) = \kappa_{b,m}(n-1) - \frac{\gamma_{m-1}(n-1)\eta_{m-1}(n)}{\mathcal{F}_{m-1}(n)}\beta_m^*(n)$$

$$\gamma_m(n-1) = \gamma_{m-1}(n-1) - \frac{\gamma_{m-1}^2(n-1)|\beta_{m-1}(n-1)|^2}{\mathcal{B}_{m-1}(n-1)}$$

Filtering:
For $n = 1, 2, 3, \ldots$, compute the various order updates in the sequence $m = 1, 2, \ldots, M+1$:

$$\xi_m(n) = \xi_{m-1}(n) - h_{m-1}^*(n-1)\beta_{m-1}(n)$$

$$h_{m-1}(n) = h_{m-1}(n-1) + \frac{\gamma_{m-1}(n)\beta_{m-1}(n)}{\mathcal{B}_{m-1}(n)}\xi_m^*(n)$$

Initialization:

1. To initialize the algorithm, at adaptation cycle $n = 0$, set

$$\mathcal{F}_{m-1}(0) = \delta, \qquad \delta = \text{small positive constant}$$
$$\mathcal{B}_{m-1}(-1) = \delta$$
$$\kappa_{f,m}(0) = \kappa_{b,m}(0) = 0$$
$$\gamma_0(0) = 1$$

2. For each adaptation cycle $n \geq 1$, generate the zeroth-order variables:

$$\eta_0(n) = \beta_0(n) = u(n)$$
$$\mathcal{F}_0(n) = \mathcal{B}_0(n) = \lambda \mathcal{F}_0(n-1) + |u(n)|^2$$
$$\gamma_0(n-1) = 1$$

3. For joint-process estimation, at adaptation cycle $n = 0$, set

$$h_{m-1}(0) = 0$$

At each adaptation cycle $n \geq 1$, generate the zeroth-order variable

$$\xi_0(n) = d(n)$$

1. For all m and n, the backward reflection coefficient $\kappa_{b,m}(n)$ is set equal to the complex conjugate of the forward reflection coefficient $\kappa_{f,m}(n)$.
2. For all m and n, the conversion factor $\gamma_m(n)$ is set equal to unity.
3. At adaptation cycle n, the adjustment to the reflection coefficient κ_m in the GAL algorithm is related in a special way to the corresponding adjustments in $\kappa_{f,m}(n)$ and $\kappa_{b,m}(n)$.

16.13 RELATION BETWEEN RECURSIVE LSL AND RLS ALGORITHMS

In solving the joint-process estimation problem with an order-recursive adaptive filter, we have shown how the least-squares predictor can be expanded to include the estimation of a desired response. The solution to this problem encompasses the computation of a set of regression coefficients $\{h_0(n), h_1(n), \ldots, h_M(n)\}$ that is fed with a corresponding set of inputs represented by the backward prediction errors $\{b_0(n), b_1, \ldots, b_M(n)\}$. [See Fig. 16.7.] Recognizing that there is a one-to-one correspondence between this set of backward prediction errors and the set of tap inputs $\{u(n), u(n-1), \ldots, (n-M)\}$, as shown in Eq. (16.50), we expect to find a corresponding relationship between the sequence of LSL regression coefficients and the set of least-squares tap weights $\{\hat{w}_0(n), \hat{w}_1(n), \ldots, \hat{w}_M(n)\}$ in the RLS algorithm. The purpose of this section is to derive this relationship formally.

Toward that end, consider the conventional tapped-delay-line or FIR-filter structure shown in Fig. 16.15. For order m, the tap inputs $u(n)$, $u(n-1)$, $\ldots$, $u(n-m)$ are derived directly from the input $u(n)$ and the tap weights $\hat{w}_0(n)$, $\hat{w}_1(n)$, $\ldots$, $\hat{w}_m(n)$ are used to form respective scalar inner products. From Chapter 9, we recall that the

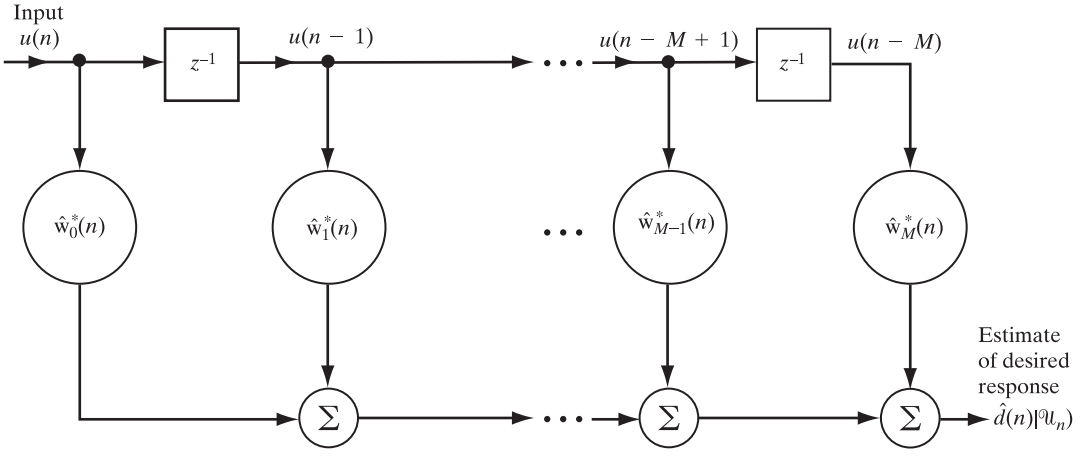


FIGURE 16.15 Conventional FIR fiter for estimating the desired response using the RLS algorithm.

least-squares solution for the $(m+1)$-by-1 tap vector $\hat{\mathbf{w}}_m(n)$ consisting of the elements $\hat{w}_0(n), \hat{w}_1(n), \ldots, \hat{w}_m(n)$ is defined by

$$\boldsymbol{\Phi}_{m+1}(n)\hat{\mathbf{w}}_m(n) = \mathbf{z}_{m+1}(n), \tag{16.136}$$

where $\boldsymbol{\Phi}_{m+1}(n)$ is the $(m+1)$-by-$(m+1)$ correlation matrix of the tap inputs and $\mathbf{z}_{m+1}(n)$ is the corresponding $(m+1)$-by-1 cross-correlation vector between the tap inputs and the desired response. We modify Eq. (16.136) in two steps:

1. We premultiply both sides of the equation by a $(m+1)$-by-$(m+1)$ lower triangular transformation matrix $\mathbf{L}_m(n)$.
2. We interject the $(m+1)$-by-$(m+1)$ identity matrix $\mathbf{I} = \mathbf{L}_m^{\mathrm{H}}(n)\mathbf{L}_m^{-\mathrm{H}}(n)$ between the matrix $\boldsymbol{\Phi}_{m+1}(n)$ and the vector $\hat{\mathbf{w}}_m(n)$ on the left-hand side of the equation.

The matrix $\mathbf{L}_m(n)$ is defined in terms of the tap weights of backward prediction-error filters of orders $0, 1, 2, \ldots, m$, as in Eq. (16.53). The symbol $\mathbf{L}_m^{-\mathrm{H}}(n)$ denotes the Hermitian transpose of the inverse matrix $\mathbf{L}_m^{-1}(n)$. We may thus write

$$\mathbf{L}_m(n)\boldsymbol{\Phi}_{m+1}(n)\mathbf{L}_m^{\mathrm{H}}(n)\mathbf{L}_m^{-\mathrm{H}}(n)\hat{\mathbf{w}}_m(n) = \mathbf{L}_m(n)\mathbf{z}_{m+1}(n). \tag{16.137}$$

Now, let

$$\mathbf{D}_{m+1}(n) = \mathbf{L}_m(n)\boldsymbol{\Phi}_{m+1}(n)\mathbf{L}_m^{\mathrm{H}}(n) \tag{16.138}$$

in Eq. (16.137). Then, using the formula for the augmented normal equations for backward linear prediction, we may show that the product $\boldsymbol{\Phi}_{m+1}(n)\mathbf{L}_m^{\mathrm{H}}(n)$ consists of a lower triangular matrix whose diagonal elements equal the various sums of weighted backward a posteriori prediction-error squares—that is, $\mathcal{B}_0(n), \mathcal{B}_1(n), \ldots, \mathcal{B}_m(n)$. (See Problem 13.) The matrix $\mathbf{L}_m(n)$ is, by definition, a lower triangular matrix whose diagonal elements are all equal to unity. Hence, the product of $\mathbf{L}_m(n)$ and $\boldsymbol{\Phi}_{m+1}(n)\mathbf{L}_m^{\mathrm{H}}(n)$ is a lower triangular matrix. We also know that $\mathbf{L}_m^{\mathrm{H}}(n)$ is an upper triangular matrix, and so is the matrix product $\mathbf{L}_m(n)\boldsymbol{\Phi}_{m+1}(n)$. Hence, the product of $\mathbf{L}_m(n)\boldsymbol{\Phi}_{m+1}(n)$ and $\mathbf{L}_m^{\mathrm{H}}(n)$ is an upper triangular matrix. In other words, the matrix $\mathbf{D}_{m+1}(n)$ is both *upper* and *lower* triangular, which can only be satisfied if $\mathbf{D}_{m+1}(n)$ is *diagonal*. Accordingly, we may write

$$\begin{aligned}\mathbf{D}_{m+1}(n) &= \mathbf{L}_m(n)\boldsymbol{\Phi}_{m+1}(n)\mathbf{L}_m^{\mathrm{H}}(n)\\ &= \mathrm{diag}[\mathcal{B}_0(n), \mathcal{B}_1(n), \ldots, \mathcal{B}_m(n)].\end{aligned} \tag{16.139}$$

Equation (16.139) is further proof that the backward a posteriori prediction errors $b_0(n), b_1(n), \ldots, b_m(n)$ produced by the various stages of the LSL predictor are uncorrelated (in a time-average sense) at all adaptation cycles.

The product $\mathbf{L}_m(n)\mathbf{z}_{m+1}(n)$ on the right-hand side of Eq. (16.137) equals the cross-correlation vector between the backward prediction errors and the desired response; let this cross-correlation vector be defined by

$$\mathbf{t}_{m-1}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{b}_{m+1}(i)d^*(i) \tag{16.140}$$

where $d(i)$ is the desired response. Substituting Eq. (16.52) into Eq. (16.140), we thus get

$$\begin{aligned}\mathbf{t}_{m+1}(n) &= \sum_{i=1}^{n} \lambda^{n-i}\mathbf{L}_m(n)\mathbf{u}_{m+1}(i)d^*(i)\\ &= \mathbf{L}_m(n)\sum_{i=1}^{n}\lambda^{n-i}\mathbf{u}_{m+1}(i)d^*(i)\\ &= \mathbf{L}_m(n)\mathbf{z}_{m+1}(n),\end{aligned} \tag{16.141}$$

which is the result we have been seeking. Accordingly, the combined use of Eqs. (16.138) and (16.141) in Eq. (16.137) yields the *transformed RLS solution*

$$\mathbf{D}_{m+1}(n)\mathbf{L}_m^{-\mathrm{H}}(n)\hat{\mathbf{w}}_m(n) = \mathbf{t}_{m+1}(n). \tag{16.142}$$

Thus far, we have considered how the application of the lower triangular matrix $\mathbf{L}_m(n)$ transforms the RLS solution for the tap-weight vector of the conventional FIR structure shown in Fig. 16.15. We next consider the linear combiner of Fig. 16.7 involving the regression coefficient vector

$$\mathbf{h}_m(n) = [h_0(n), h_1(n), \ldots, h_m(n)]^{\mathrm{T}}. \tag{16.143}$$

The vector $\mathbf{h}_m(n)$ may be viewed as the solution that minimizes the cost function

$$\sum_{i=1}^{n}\lambda^{n-i}|\, d(i) - \mathbf{b}_{m+1}^{\mathrm{T}}(i)\mathbf{h}_m^*(n)\,|^2,$$

where $\mathbf{h}_m(n)$ is held constant for $1 \le i \le n$. The solution to the RLS problem is

$$\mathbf{D}_{m+1}(n)\mathbf{h}_m(n) = \mathbf{t}_{m+1}(n), \tag{16.144}$$

where $\mathbf{D}_{m+1}(n)$ is the $(m+1)$-by-$(m+1)$ correlation matrix of the backward a posteriori prediction errors used as inputs to the regression coefficients and $\mathbf{t}_{m+1}(n)$ is the $(m+1)$-by-1 cross-correlation vector between these inputs and the desired response.

By comparing the transformed RLS solution of Eq. (16.142) with the RLS solution of Eq. (16.144), we immediately see that the connecting *link* between these two solutions is provided by the lower triangular matrix $\mathbf{L}_m(n)$ of Eq. (16.53), as shown by

$$\mathbf{h}_m(n) = \mathbf{L}_m^{-\mathrm{H}}(n)\hat{\mathbf{w}}_m(n) \tag{16.145}$$

or, equivalently,

$$\hat{\mathbf{w}}_m(n) = \mathbf{L}_m^{\mathrm{H}}(n)\mathbf{h}_m(n). \tag{16.146}$$

On the basis of these two equations, we conclude the discussion of least-squares estimation by stating that linear adaptive filtering problems such as system identification and

channel equalization can be solved by using one of two equivalent structures, each with its own advantages:

- A *least-squares FIR estimator*, which offers structural *simplicity* as shown in Fig. 16.15.
- A *least-squares joint-process estimator*, which offers structural *modularity*. Specifically, it consists of the multistage lattice predictor of Fig. 16.6 that feeds the linear combiner of Fig. 16.7.

16.14 FINITE-PRECISION EFFECTS

In this section, we continue our discussion concerning the impact of finite-precision effects on adaptive filtering algorithms that began in Chapter 12, where attention was focused on the least-mean-square (LMS) and RLS algorithms, and continued in Section 15.6, which examined the case of square-root adaptive filtering algorithms. Here, we complete the study of finite-precision effects by discussing their impact on order-recursive adaptive filtering algorithms. In such algorithms (and, for that matter, in all fast RLS algorithms known to date), the particular section responsible for joint-process estimation is subordinate to the section responsible for performing the forward and backward linear predictions. Accordingly, the numerical stability of order-recursive LSL adaptive filtering algorithms is critically dependent on how the prediction section performs its computations.

QRD-LSL Algorithm

In the *QR-decomposition–based least-squares lattice (QRD-LSL) algorithm* summarized in Table 16.4, the prediction section consists of M lattice stages, where M is the final prediction order. Each stage of the prediction section uses QR-decomposition in the form of Givens rotations to perform its computations. The net result is that the sequence of input data $u(n), u(n-1), \ldots, u(n-M)$ is transformed into a corresponding sequence of angle-normalized backward prediction errors $\varepsilon_{b,0}(n), \varepsilon_{b,1}(n), \ldots, \varepsilon_{b,M}(n)$. Given this latter sequence, the joint-process estimation section also uses QR-decomposition, on a stage-by-stage basis, to perform its own computations; the final product is a least-squares estimate of some desired response $d(n)$. In other words, all the computations throughout the algorithm are performed using QR-decomposition.

From a numerical point of view, the QRD-LSL algorithm has some desirable numerical properties:

1. The sines and cosines involved in applying the Givens rotations are all numerically *well behaved*.
2. The algorithm is *numerically consistent* in that, from one adaptation cycle to the next, each section of the algorithm propagates the *minimum* possible number of parameters needed for satisfactory operation; that is, the propagation of related

parameters is avoided. Table 16.4 shows that the parameters propagated by the three sections of the algorithm are as follows:

Section	Parameters propagated
Forward prediction	$\mathcal{B}_{m-1}^{1/2}(n-2)$, $p_{f,m-1}(n-1)$, $\gamma_{m-1}^{1/2}(n-1)$
Backward prediction	$\mathcal{F}_{m-1}^{1/2}(n-1)$, $p_{b,m-1}(n-1)$
Joint-process estimation	$p_{m-1}(n-1)$

3. The auxiliary parameters $p_{f,m-1}(n-1)$, $p_{b,m-1}(n-1)$, and $p_{m-1}(n-1)$, which are involved in the order updating of the angle-normalized estimation errors $\varepsilon_{f,m-1}(n)$, $\varepsilon_{b,m-1}(n)$, and $\varepsilon_m(n)$, respectively, are all computed *directly*. That is, *local error feedback* is involved in the time-update recursions used to compute each of these auxiliary parameters. This form of feedback is another factor in assuring numerical stability of the algorithm.

Among the various pieces of experimental evidence for numerical stability of the QRD-LSL algorithm are computer simulation studies reported by Ling (1989), Yang and Böhme (1992), McWhirter and Proudler (1993), and Levin and Cowan (1994), all of which have demonstrated the numerical robustness of variants of the QRD-LSL algorithm. Of particular interest is the paper by Levin and Cowan (1994), in which the performance of eight different adaptive filtering algorithms of the RLS family was evaluated in a finite-precision environment. The results presented therein demonstrate the superior performance of algorithms belonging to the square-root information-filtering domain (exemplified by the QRD-LSL algorithm) over those belonging to the covariance-filtering domain, the latter being exemplified by the traditional RLS algorithm. Moreover, of the eight algorithms considered in the study, the QRD-LSL algorithm appeared to be the least affected by numerical imprecision.

Further evidence for the numerical robustness of the QRD-LSL is presented by Capman et al. (1995), who describe an acoustic echo canceller that combines a multirate scheme with a variant of the QRD-LSL algorithm. Simulation results presented therein demonstrate the numerical robustness of this solution to the echo cancellation problem.

Recursive LSL Algorithms

Turning next to *recursive least-squares lattice (LSL) algorithms,* recall that these algorithms are special cases of the QRD-LSL algorithm. Indeed, they are derived by squaring the arrays of the QRD-LSL algorithm and then comparing terms. Recognizing that, in the context of the numerical behavior of an algorithm, squaring has an effect opposite to that of taking the square root, we may therefore state that the performance of recursive LSL

algorithms in a finite-precision environment is always *inferior* to that of the QRD-LSL algorithm from which they are derived.

A recursive LSL algorithm provides a "fast" solution to the recursive least-squares estimation problem by employing a multistage lattice predictor for transforming the input data into a corresponding sequence of backward prediction errors. This transformation may be viewed as a form of the classical Gram–Schmidt orthogonalization procedure, which is known to be *numerically inaccurate* (Stewart, 1973). Correspondingly, a conventional form of the recursive LSL algorithm (be it based on a posteriori or a priori prediction errors) has poor numerical behavior. The key to a practical method of overcoming the numerical accuracy problem in a recursive LSL algorithm is to update the forward and backward reflection coefficients *directly,* rather than first computing the individual sums of weighted forward and backward prediction errors and their cross-correlations and then taking ratios of the appropriate quantities (as in a conventional LSL algorithm). This is precisely what is done in a recursive LSL algorithm *with error feedback* (Ling & Proakis, 1984a), exemplified by the algorithm summarized in Table 16.6. For a prescribed fixed-point representation, a recursive LSL algorithm with error feedback works with much more accurate values of the forward and backward reflection coefficients, which are the key parameters in any recursive LSL algorithm. The direct computation of the forward and backward reflection coefficients therefore has the overall effect of preserving the positive definiteness of the underlying inverse correlation matrix of the input data, despite the presence of quantization errors due to finite-precision effects. Therefore, insofar as numerical performance is concerned, recursive LSL algorithms with error feedback are preferred to their conventional forms.[3]

16.15 SUMMARY AND DISCUSSION

In this chapter, we further consolidated the close relationship between Kalman filter theory and the family of adaptive linear filters that is rooted in least-squares estimation. In particular, we demonstrated how the square-root information filtering algorithm, which is a variant of the Kalman filter, can be used to derive the QR-decomposition-based least-squares lattice (QRD-LSL) algorithm. Simply put, the QRD-LSL algorithm represents the most fundamental form of an order-recursive adaptive filtering algorithm. Other order-recursive adaptive filtering algorithms, such as the recursive LSL algorithm using a posteriori estimation errors and the recursive LSL algorithm using a priori estimation errors with error feedback, are in fact rewritings of the QRD-LSL algorithm in one form or another.

[3]North et al. (1993) present computer simulations (using floating-point arithmetic) that compare the numerical behavior of a 32-bit directly updated recursive LSL algorithm (i.e., an algorithm with error feedback) with a 32-bit indirectly updated recursive LSL algorithm. Their study involved adaptive interference cancellation. In the recursive LSL algorithm with indirect updating, it was found that after about 10^5 adaptation cycles the accumulation of numerical errors resulted in a degradation of approximately 20 dB in interference cancellation, compared with the directly updated recursive LSL algorithm.

The QRD-LSL algorithm combines highly desirable features of RLS estimation, QR-decomposition, and a lattice structure. Accordingly, it offers a unique set of computational and implementational advantages:

- The QRD-LSL algorithm has a *fast rate of convergence*, which is inherent in RLS estimation.
- The QRD-LSL algorithm can be implemented using a sequence of Givens rotations, which represent a form of QR-decomposition. Moreover, the good numerical properties of the QR-decomposition mean that the QRD-LSL algorithm is *numerically stable*.
- The QRD-LSL algorithm offers a *high level of computational efficiency*, in that its complexity is on the order of M, where M is the final prediction order (i.e., the number of available degrees of freedom).
- The lattice structure of the QRD-LSL algorithm is *modular* in nature, which means that the prediction order can be increased without having to recalculate all previous values. This property is particularly useful when there is no prior knowledge as to what the final value of the prediction order should be.
- Another implication of the modular structure of the QRD-LSL algorithm is that it lends itself to the use of *very large-scale integration (VLSI)* technology for its hardware implementation. Of course, the use of this sophisticated technology can be justified only if the application of interest calls for the use of VLSI chips in large numbers.
- The QRD-LSL algorithm includes an *integral set of desired variables and parameters* that are useful to have in signal-processing applications. Specifically, the algorithm offers the following three sets of useful by-products:
  - Angle-normalized forward and backward prediction errors.
  - Auxiliary parameters that can be used for the *indirect* computation of the forward and backward reflection coefficients and the regression coefficients (i.e., tap weights).

The recursive LSL algorithms enjoy many of the properties of the QRD-LSL algorithm, namely, fast convergence, modularity, and an integral set of useful parameters and variables for signal-processing applications. However, the numerical properties of recursive LSL algorithms depend on whether error feedback is included in their composition; this issue was discussed in Chapter 12.

The order-recursive adaptive filters considered in this chapter have a computational advantage over the square-root adaptive filters examined in the previous chapter: With the order-recursive filters, the computational cost increases linearly with the number of adjustable parameters, whereas with the square-root filters, the cost of computation increases as the square of the number of adjustable parameters. However, the use of order-recursive adaptive filters is limited to *temporal* signal-processing applications that permit the exploitation of the time-shifting property of the input data. In contrast, square-root adaptive filters can be used for both *temporal and spatial* signal-processing applications.

PROBLEMS

1. Derive the estimation error $\gamma_m(n)$ that represents the output of an FIR filter whose tap-weight vector equals the gain vector $\mathbf{k}_m(n)$, and is excited by the tap-input vector $\mathbf{u}_m(n)$ as depicted in Fig. 16.3. [*Hint*: Since the filter output has the structure of a Hermitian form, it follows that the estimation error $\gamma_m(n)$ is a real-valued scalar. Moreover, $\gamma_m(n)$ is bounded by zero and unity.]

2. Let $\mathbf{\Phi}_m(n)$ denote the time-average correlation matrix of the tap-input vector $\mathbf{u}_m(n)$ at adaptation cycle n, and likewise for $\mathbf{\Phi}_m(n-1)$. Show that the conversion factor

$$\gamma_m(n) = \lambda \frac{\det[\mathbf{\Phi}_m(n-1)]}{\det[\mathbf{\Phi}_m(n)]},$$

where λ is the exponential weighting factor. [*Hint*: Use the identity

$$\det(\mathbf{I}_1 + \mathbf{AB}) = \det(\mathbf{I}_2 + \mathbf{BA}),$$

where $\mathbf{I}_1$ and $\mathbf{I}_2$ are identity matrices of appropriate dimensions and $\mathbf{A}$ and $\mathbf{B}$ are matrices of compatible dimensions.]

3. **(a)** Show that the inverse of the correlation matrix $\mathbf{\Phi}_{m-1}(n)$ may be expressed as

$$\mathbf{\Phi}_{m+1}^{-1}(n) = \begin{bmatrix} 0 & \mathbf{0}_m^{\mathrm{T}} \\ \mathbf{0}_m & \mathbf{\Phi}_m^{-1}(n-1) \end{bmatrix} + \frac{1}{\mathscr{F}_m(n)} \mathbf{a}_m(n)\mathbf{a}_m^{\mathrm{H}}(n),$$

where $\mathbf{0}_m$ is the M-by-1 null vector, $\mathbf{0}_m^{\mathrm{T}}$ is its transpose, $\mathscr{F}_m(n)$ is the minimum sum of weighted forward prediction-error squares, and $\mathbf{a}_m(n)$ is the tap-weight vector of a forward prediction-error filter. Both $\mathbf{a}_m(n)$ and $\mathscr{F}_m(n)$ refer to prediction order m.

(b) Show that the inverse of $\mathbf{\Phi}_{m+1}(n)$ may also be expressed as

$$\mathbf{\Phi}_{m+1}^{-1}(n) = \begin{bmatrix} \mathbf{\Phi}_m^{-1}(n) & \mathbf{0}_m \\ \mathbf{0}_m^{\mathrm{T}} & 0 \end{bmatrix} + \frac{1}{\mathscr{B}_m(n)} \mathbf{c}_m(n)\mathbf{c}_m^{\mathrm{H}}(n),$$

where $\mathscr{B}_m(n)$ is the minimum sum of weighted backward a posteriori prediction-error squares and $\mathbf{c}_m(n)$ is the tap-weight vector of the backward prediction-error filter. Both $\mathscr{B}_m(n)$ and $\mathbf{c}_m(n)$ refer to prediction order m.

4. Derive the following update formulas for the conversion factor:

$$\gamma_{m+1}(n) = \gamma_m(n-1) - \frac{|f_m(n)|^2}{\mathscr{F}_m(n)};$$

$$\gamma_{m+1}(n) = \gamma_m(n) - \frac{|b_m(n)|^2}{\mathscr{B}_m(n)};$$

$$\gamma_{m+1}(n) = \lambda \frac{\mathscr{F}_m(n-1)}{\mathscr{F}_m(n)} \gamma_m(n-1);$$

$$\gamma_{m+1}(n) = \lambda \frac{\mathscr{B}_m(n-1)}{\mathscr{B}_m(n)} \gamma_m(n).$$

5. **(a)** Starting with Eq. (16.10) and using the orthogonality condition of Eq. (16.9), derive the time-update equation (16.11) for computing the forward prediction-error energy.

(b) Starting with Eq. (16.21) and using the orthogonality condition of Eq. (16.20), derive the time-update equation (16.22) for computing the backward prediction-error energy.

6. Referring back to the three kinds of estimation error discussed in Section 16.4 on the conversion factor, justify the following equations:
(a) Eq. (16.27) for RLS estimation.
(b) Eq. (16.28) for adaptive forward linear prediction.
(c) Eq. (16.29) for adaptive backward linear prediction.

7. Equation (16.49) presents one way of updating the exponentially weighted cross-correlation $\Delta_{m-1}(n)$. Equivalently, we may use the update equation

$$\Delta_{m-1}(n) = \lambda\Delta_{m-1}(n-1) + f^*_{m-1}(n)\beta_{m-1}(n-1).$$

Hence, we may deduce the equivalence

$$f^*_{m-1}(n)\beta_{m-1}(n-1) = \eta^*_{m-1}(n)b_{m-1}(n).$$

(a) Starting with the definition of Eq. (16.34) and following a procedure similar to that used in Section 16.5, derive the second recursion presented in this problem for $\Delta_{m-1}(n)$.
(b) Justify the given equivalence involving the a priori and a posteriori forms of forward and backward prediction errors using the interpretations of the conversion factor presented in Section 16.4.

8. In this problem, we explore another (albeit much more complicated) procedure for deriving the time-update equation (16.49) for computing $\Delta_{m-1}(n)$.
(a) Starting with the augmented normal equations

$$\mathbf{\Phi}_{m+1}(n)\mathbf{a}_m(n) = \begin{bmatrix} \mathcal{F}_m(n) \\ \mathbf{0}_m \end{bmatrix}$$

for forward linear prediction and using the expansion

$$\mathbf{\Phi}_{m+1}(n) = \left[\begin{array}{cc} \mathbf{\Phi}_m(n) & \boldsymbol{\phi}_2(n) \\ \boldsymbol{\phi}_2^{\mathrm{H}}(n) & \mathcal{U}_2(n) \end{array}\right],$$

show that

$$\mathbf{\Phi}_{m+1}(n)\begin{bmatrix} \mathbf{a}_{m-1}(n) \\ 0 \end{bmatrix} = \begin{bmatrix} \mathcal{F}_{m-1}(n) \\ \mathbf{0}_{m-1} \\ \Delta_{m-1}(n) \end{bmatrix},$$

where

$$\Delta_{m-1}(n) = \boldsymbol{\phi}_2^{\mathrm{H}}(n)\mathbf{a}_{m-1}(n).$$

(b) Show that this new definition of $\Delta_{m-1}(n)$ is equivalent to that of Eq. (16.34).
(c) The augmented normal equations may be written in the equivalent form

$$\mathbf{\Phi}_{m+1}(n)\mathbf{c}_m(n) = \begin{bmatrix} \mathbf{0}_m \\ \mathcal{B}_m(n) \end{bmatrix},$$

where $\mathbf{\Phi}_{m+1}(n)$ is now expanded as

$$\mathbf{\Phi}_{m+1}(n) = \left[\begin{array}{c|c} \mathcal{U}_1(n) & \boldsymbol{\phi}_1^{\mathrm{H}}(n) \\ \hline \boldsymbol{\phi}_1(n) & \mathbf{\Phi}_m(n-1) \end{array}\right].$$

On this second basis, show that we may also write

$$\mathbf{\Phi}_{m+1}(n)\begin{bmatrix} 0 \\ \mathbf{c}_{m-1}(n-1) \end{bmatrix} = \begin{bmatrix} \Delta'_{m-1}(n) \\ \mathbf{0}_{m-1} \\ \mathcal{B}_{m-1}(n-1) \end{bmatrix}.$$

(d) Next, using the results of parts (a) and (c), show that the parameters $\Delta_{m-1}(n)$ and $\Delta'_{m-1}(n)$ are the complex conjugate of one another; that is,

$$\Delta'_{m-1}(n) = \Delta^*_{m-1}(n).$$

(e) Recognizing that the leading element of the vector $\mathbf{a}_{m-1}(n-1)$ equals unity, we may express $\Delta_{m-1}(n)$ as

$$\Delta_{m-1}(n) = [\Delta_{m-1}(n), \mathbf{0}^{\mathrm{T}}, \mathcal{B}_{m-1}(n-1)]\begin{bmatrix} \mathbf{a}_{m-1}(n-1) \\ 0 \end{bmatrix}.$$

Hence, using the recursion

$$\mathbf{\Phi}_{m+1}(n) = \lambda\mathbf{\Phi}_{m+1}(n-1) + \mathbf{u}_{m+1}(n)\mathbf{u}^{\mathrm{H}}_{m+1}(n),$$

show that

$$\begin{aligned}\Delta_{m-1}(n) = {} & \lambda[0, \mathbf{c}^{\mathrm{H}}_{m-1}(n-1)\mathbf{\Phi}_{m+1}(n-1)]\begin{bmatrix} \mathbf{a}_{m-1}(n-1) \\ 0 \end{bmatrix} \\ & + [0, \mathbf{c}^{\mathrm{H}}_{m-1}(n-1)\mathbf{u}_{m+1}(n)\ \mathbf{u}^{\mathrm{H}}_{m+1}(n)]\begin{bmatrix} \mathbf{a}_{m-1}(n-1) \\ 0 \end{bmatrix}.\end{aligned}$$

(f) Finally, using the definitions for the forward a priori prediction error $\eta_{m-1}(n)$ and the backward a posteriori prediction error $b_{m-1}(n-1)$, derive the desired recursion

$$\Delta_{m-1}(n) = \lambda\Delta_{m-1}(n-1) + \eta^*_{m-1}(n)b_{m-1}(n-1).$$

9. Using the results obtained in parts (a) and (c) of Problem 7, derive the following order-update recursions involving the sums of forward and backward prediction-error squares, respectively:

$$\mathcal{F}_m(n) = \mathcal{F}_{m-1}(n) - \frac{|\Delta_{m-1}(n)|^2}{\mathcal{B}_{m-1}(n-1)};$$

$$\mathcal{B}_m(n) = \mathcal{B}_{m-1}(n-1) - \frac{|\Delta_{m-1}(n)|^2}{\mathcal{F}_{m-1}(n)}.$$

10. In this problem, we show how the various quantities of the fast prediction equations relate to each other, and we demonstrate the parametric redundancy that they contain.[4]

(a) By combining parts (a) and (b) from Problem 3, show that

$$\begin{bmatrix} \boldsymbol{\Phi}_m^{-1}(n) & \mathbf{0}_m \\ \mathbf{0}_m^{\mathrm{T}} & 0 \end{bmatrix} - \begin{bmatrix} 0 & \mathbf{0}_m^{\mathrm{T}} \\ \mathbf{0}_m & \boldsymbol{\Phi}_m^{-1}(n-1) \end{bmatrix} = \frac{\mathbf{a}_m(n)\mathbf{a}_m^{\mathrm{H}}(n)}{\mathcal{F}_m(n)} - \frac{\mathbf{c}_m(n)\mathbf{c}_m^{\mathrm{H}}(n)}{\mathcal{B}_m(n)}.$$

(b) Using the recursive Eqs. (10.16) and (10.18) in Chapter 10, plus Eq. (16.26) in this chapter, show that the time update for $\boldsymbol{\Phi}_m^{-1}$ may be expressed as follows:

$$\boldsymbol{\Phi}_m^{-1}(n) = \lambda^{-1}\boldsymbol{\Phi}_m^{-1}(n-1) - \frac{\mathbf{k}_m(n)\mathbf{k}_m^{\mathrm{H}}(n)}{\gamma_m(n)},$$

where $\mathbf{k}_m(n)$ is the gain vector and $\gamma_m(n)$ is the conversion factor.

(c) By eliminating $\boldsymbol{\Phi}_m^{-1}(n-1)$ from the preceding two expressions, show that all the variables may be reconciled as

$$\begin{bmatrix} \boldsymbol{\Phi}_m^{-1}(n) & \mathbf{0}_m \\ \mathbf{0}_m^{\mathrm{T}} & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & \mathbf{0}_m^{\mathrm{T}} \\ \mathbf{0}_m & \boldsymbol{\Phi}_m^{-1}(n) \end{bmatrix} = \frac{\mathbf{a}_m(n)\mathbf{a}_m^{\mathrm{H}}(n)}{\mathcal{F}_m(n)} + \lambda \begin{bmatrix} 0 \\ \mathbf{k}_m(n) \end{bmatrix} \frac{[0, \mathbf{k}_m^{\mathrm{H}}(n)]}{\gamma_m(n)} - \frac{\mathbf{c}_m(n)\mathbf{c}_m^{\mathrm{H}}(n)}{\mathcal{B}_m(n)},$$

in which all the variables have a common adaptation cycle n and a common order index m. The left-hand side of this equation is called a *displacement residue* of $\boldsymbol{\Phi}_m^{-1}(n)$, and the right-hand side, being the sum and difference of three vector dyads, has rank not exceeding three. In matrix theory, $\boldsymbol{\Phi}_m^{-1}(n)$ is said to have a *displacement rank* three as a result of the relevant structure of the data matrix pertaining to the *prewindowing method*, that is the third numbered item in Section 9.2. Note that this structure holds irrespective of the sequence $u(n)$ that builds the data matrix.

(d) Suppose we multiply the result of part (c) from the left by the row vector $[1, z\sqrt{\lambda}, \ldots, (z/\sqrt{\lambda})^m]$ and from the right by the column vector $[1, w\sqrt{\lambda}, \ldots, (w/\sqrt{\lambda})^m]^{\mathrm{H}}$, where z and w are two complex variables. Show that the result of part (c) is equivalent to the two-variable polynomial equation

$$(1 - zw^*)P(z, w^*) = A(z)A^*(w) + K(z)K^*(w) - C(z)C^*(w) \qquad \text{for all } z, w,$$

provided that we make the correspondences

$$P(z, w^*) = [1, z/\sqrt{\lambda}, \ldots, (z/\sqrt{\lambda})^{m-1}]\boldsymbol{\Phi}_m^{-1}(n) \begin{bmatrix} 1 \\ w^*/\sqrt{\lambda} \\ \vdots \\ (w^*/\sqrt{\lambda})^{M-1} \end{bmatrix},$$

$$A(z) = [1, z/\sqrt{\lambda}, \ldots, (z/\sqrt{\lambda})^m] \frac{\mathbf{a}_m(n)}{\sqrt{\mathcal{F}_m(n)}},$$

$$K(z) = [1, z/\sqrt{\lambda}, \ldots, (z/\sqrt{\lambda})^m] \sqrt{\frac{\lambda}{\gamma_m(n)}} \begin{bmatrix} 0 \\ \mathbf{k}_m(n) \end{bmatrix},$$

and

$$C(z) = [1, z/\sqrt{\lambda}, \ldots, (z/\sqrt{\lambda})^m] \frac{\mathbf{c}_m(n)}{\sqrt{\mathcal{B}_m(n)}}.$$

Similarly, $A^*(w) = [A(w)]^*$, and so on.

[4]This problem was originally formulated by P. Regalia, private communication, 1995.

(e) Set $z = w = e^{j\omega}$ in the result of part (d) to show that

$$|A(e^{j\omega})|^2 + |K(e^{j\omega})|^2 = |C(e^{j\omega})|^2 \qquad \text{for all } \omega;$$

that is, the three polynomials $A(z)$, $K(z)$, and $C(z)$ are power complementary along the unit circle $|z| = 1$.

(f) Show that, because $\mathbf{\Phi}_m^{-1}(n)$ is positive definite, the following system of inequalities necessarily results:

$$|A(z)|^2 + |K(z)|^2 - |C(z)|^2 = \begin{cases} < 0, & |z| > 1 \\ = 0 & |z| = 1. \\ \geq 0, & |z| < 1 \end{cases}$$

[*Hint*: Set $w^* = z^*$ in part (d), and note that if $\mathbf{\Phi}_m^{-1}(n)$ is positive definite, the inequality

$$P(z, z^*) > 0 \qquad \text{for all } z$$

must result. Note also that the center equality is equivalent to the result of part (e).]

(g) Finally, deduce from the first inequality of part (f) that $C(z)$ must be devoid of zeros in $|z| > 1$ and, hence, that given $A(z)$ and $K(z)$, the polynomial $C(z)$ is uniquely determined from part (e) via spectral factorization. This shows that, once the forward prediction and gain quantities are known, the backward prediction variables contribute nothing further to the solution and so are theoretically redundant.

11. In the text, we derived the entries under forward prediction in Table 16.3 summarizing the one-to-one correspondences between Kalman variables and angle-normalized LSL variables in stage m of a lattice predictor. Complete the table by deriving the entries listed therein under both backward prediction and joint-process estimation.

12. Justify the following relationships:

(a) Joint-process estimation errors:

$$|\varepsilon_m(n)| = \sqrt{|e_m(n)| \cdot |\xi_m(n)|};$$
$$\text{ang}[\varepsilon_m(n)] = \text{ang}[e_m(n)] + \text{ang}[\xi_m(n)].$$

(b) Backward prediction errors:

$$|\varepsilon_{b,m}(n)| = \sqrt{|b_m(n)| \cdot |\beta_m(n)|};$$
$$\text{ang}[\varepsilon_{b,m}(n)] = \text{ang}[b_m(n)] + \text{ang}[\beta_m(n)].$$

(c) Forward prediction errors:

$$|\varepsilon_{f,m}(n)| = \sqrt{|f_m(n)| \cdot |\eta_m(n)|};$$
$$\text{ang}[\varepsilon_{f,m}(n)] = \text{ang}[f_m(n)] + \text{ang}[\eta_m(n)].$$

13. The correlation matrix $\mathbf{\Phi}_{m+1}(n)$ is postmultiplied by the Hermitian transpose of the lower triangular matrix $\mathbf{L}_m(n)$ defined by Eq. (16.53). Show that the product $\mathbf{\Phi}_{m+1}(n)\mathbf{L}_m^{\text{H}}(n)$ consists of a lower triangular matrix whose diagonal elements equal the various sums of weighted backward prediction-error squares $\mathcal{B}_0(n), \mathcal{B}_1(n), \ldots, \mathcal{B}_m(n)$. Hence, show that the product $\mathbf{L}_m(n)\mathbf{\Phi}_{m-1}(n)\mathbf{L}_m^{\text{H}}(n)$ is a diagonal matrix, given by

$$\mathbf{D}_{m+1}(n) = \text{diag}[\mathcal{B}_0(n), \mathcal{B}_1(n), \ldots, \mathcal{B}_m(n)].$$

14. Consider the case where the input samples $u(n)$, $u(n-1)$, . . . , $u(n-M)$ have a *joint Gaussian distribution with zero mean*. Assume that, within a scaling factor, the ensemble-average correlation matrix $\mathbf{R}_{M+1}$ of the input signal is equal to its time-average correlation matrix $\mathbf{\Phi}_{M+1}(n)$ for adaptation cycle $n \geq M$. Show that the log-likelihood function for this input includes a term equal to the parameter $\gamma_M(n)$ associated with the recursive LSL algorithm. [For this reason, the parameter $\gamma_M(n)$ is sometimes referred to as a *likelihood variable*.]

15. Let $\hat{d}(n\,|\,\mathcal{U}_{n-m+1})$ denote the least-squares estimate of the desired response $d(n)$, given the inputs $u(n-m+1), \ldots, u(n)$ that span the space $\mathcal{U}_{n-m+1}$. Similarly, let $\hat{d}(n\,|\,\mathcal{U}_{n-m})$ denote the least-squares estimate of the desired response, given the inputs $u(n-m), u(n-m+1), \ldots, u(n)$ that span the space $\mathcal{U}_{n-m}$. In effect, the latter estimate exploits an additional piece of information represented by the input $u(n-m)$. Show that this new information is represented by the corresponding backward prediction error $b_m(n)$. Show also that the two estimates are related by the recursion

$$\hat{d}(n\,|\,\mathcal{U}_{n-m}) = \hat{d}(n\,|\,\mathcal{U}_{n-m+1}) + h_m^*(n)b_m(n),$$

where $h_m(n)$ denotes the pertinent regression coefficient in the joint-process estimator. Compare this result with that of Section 14.1 dealing with the concept of innovations.

16. In QRD-LSL filters, list the set of relations for a square-root information-filtering solution to the adaptive forward linear problem in an LSL sense.

17. Starting from the Kalman filter equation (16.132) and using Table 16.3 and Eqs. (16.77) and (16.79), derive the update equations (16.134) and (16.135) for the backward reflection coefficient $\hat{\kappa}_{b,m}(n)$ and joint-process estimation coefficient $\hat{h}_{m-1}(n)$, which pertain to the recursive LSL algorithm with error feedback.

18. In this problem, we explore the derivation of the GAL as a special form of the recursive LSL algorithm using a priori estimation with error feedback.

Starting with the algorithm summarized in Table 16.6, do the following:

(i) For all m and n, put

$$\kappa_{b,m}(n) = \kappa_{f,m}^*(n)$$

and

$$\gamma_m(n) = 1.$$

(ii) Under this special set of conditions, explore the ways in which the recursive LSL algorithm of Table 16.6 reduces to the GAL algorithm of Table 5.2.

19. In Section 16.12, we discussed a modification of the a priori error LSL algorithm by using a form of error feedback. In this problem, we consider the corresponding modified version of the a posteriori LSL algorithm. Show that

$$\kappa_{f,m}(n) = \frac{\gamma_m(n-1)}{\gamma_{m-1}(n-1)}\left[\kappa_{f,m}(n-1) - \frac{1}{\lambda}\frac{b_{m-1}(n-1)f_{m-1}^*(n)}{\mathcal{B}_{m-1}(n-2)\gamma_{m-1}(n-1)}\right]$$

and

$$\kappa_{b,m} = \frac{\gamma_m(n)}{\gamma_{m-1}(n-1)}\left[\kappa_{b,m}(n-1) - \frac{1}{\lambda}\frac{f_{m-1}(n)b_{m-1}^*(n-1)}{\mathcal{F}_{m-1}(n-1)\gamma_{m-1}(n-1)}\right].$$

20. The following table is a summary of the *normalized LSL algorithm*:

$$\overline{\Delta}_{m-1}(n) = \overline{\Delta}_{m-1}(n-1)[1 - |\bar{f}_{m-1}(n)|^2]^{1/2}[1 - |\bar{b}_{m-1}(n-1)|^2]^{1/2} + \bar{b}_{m-1}(n-1)\bar{f}^*_{m-1}(n)$$

$$\bar{b}_m(n) = \frac{\bar{b}_{m-1}(n-1) - \overline{\Delta}_{m-1}(n)\bar{f}_{m-1}(n)}{[1 - |\overline{\Delta}_{m-1}(n)|^2]^{1/2}[1 - |\bar{f}_{m-1}(n)|^2]^{1/2}}$$

$$\bar{f}_m(n) = \frac{\bar{f}_{m-1}(n) - \overline{\Delta}^*_{m-1}(n)\bar{b}_{m-1}(n-1)}{[1 - |\overline{\Delta}_{m-1}(n)|^2]^{1/2}[1 - |\bar{b}_{m-1}(n-1)|^2]^{1/2}}$$

The normalized parameters are defined by

$$\bar{f}_m(n) = \frac{f_m(n)}{\mathcal{F}_m^{1/2}(n)\gamma_m^{1/2}(n-1)},$$

$$\bar{b}_m(n) = \frac{b_m(n)}{\mathcal{B}_m^{1/2}(n)\gamma_m^{1/2}(n)},$$

and

$$\overline{\Delta}_m(n) = \frac{\Delta_m(n)}{\mathcal{F}_m^{1/2}(n)\mathcal{B}_m^{1/2}(n-1)}.$$

Derive the recursions defining the normalized LSL algorithm.

CHAPTER 17

Blind Deconvolution

Deconvolution is a signal-processing operation that, ideally, unravels the effects of convolution performed by a linear time-invariant system operating on an input signal. More specifically, in ordinary deconvolution, the output signal and the system are both known, and the requirement is to reconstruct what the input signal must have been. In *blind deconvolution*, or in more precise terms, *unsupervised deconvolution*, only the output signal is known (i.e., both the system and the input signal are unknown), and the requirement is to find the input signal, hence the system itself. Clearly, blind deconvolution is a more difficult signal-processing task than ordinary deconvolution. In return, blind deconvolution fills a void in signal-processing applications, where the need for blind adaptation is a desirable requirement.

To pave the way for a study of the blind deconvolution problem, we begin by discussing the theoretic implications and practical importance of blind deconvolution.

17.1 OVERVIEW OF BLIND DECONVOLUTION

Consider an *unknown* linear time-invariant system $\mathscr{L}$ with input $x(n)$, as depicted in Fig. 17.1. The input data (information-bearing) sequence $x(n)$ is assumed to consist of *independently and identically distributed (i.i.d.) symbols*; the only thing known about the input is its probability distribution. The problem is:

> Restore $x(n)$, or equivalently, *identify the inverse* $\mathscr{L}^{-1}$ *of the system* $\mathscr{L}$, *given the observed sequence* $u(n)$ *at the system output*.

If the system $\mathscr{L}$ is *minimum phase* (i.e., the transfer function of the system has all of its poles and zeros confined to the interior of the unit circle in the z-plane), then not only is $\mathscr{L}$ stable, so is the inverse system $\mathscr{L}^{-1}$. In that case, we may view the input sequence $x(n)$ as the "innovation" of the system output $u(n)$, and the inverse system $\mathscr{L}^{-1}$ is just a

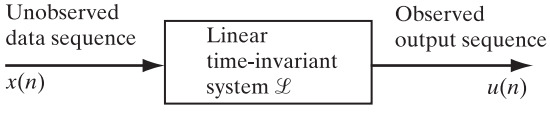


FIGURE 17.1 Setting the stage for blind deconvolution.

whitening filter. With them the blind deconvolution problem is solved. These observations follow from the study of linear prediction presented in Chapter 3.

In many practical situations, however, the system $\mathscr{L}$ may *not* be minimum phase. A system is said to be *non-minimum-phase* if its transfer function has any of its zeros located outside the unit circle in the z-plane; exponential stability of the system dictates that the poles must be located inside the unit circle. Two practical examples of a non-minimum-phase system are a telephone channel and a fading radio channel. In such situations, the restoration of the input sequence $x(n)$, given the channel output, is a difficult problem.

Typically, adaptive equalizers used in digital communications require an initial *training period*, during which a *known* data sequence is transmitted. A replica of this sequence is made available at the receiver in proper synchronism with the transmitter, thereby making it possible for adjustments to be made to the equalizer coefficients in accordance with the adaptive filtering algorithm employed in the equalizer design. When the training is completed, the equalizer is switched to its *decision-directed mode*, and normal data transmission may then commence. These modes of operation of an adaptive equalizer were discussed in Chapter 6 on the LMS algorithm.

However, in some practical situations, it would be highly desirable for a receiver to be able to achieve complete adaptation *without* access to a desired response. For example, in a *multipoint data network* involving a *control unit* connected to several pieces of *data terminal equipment* (DTE), we have a "master–slave" type of situation, in that a DTE is permitted to transmit only when its modem is polled by the modem of the control unit. A problem peculiar to these networks is that of retraining the receiver of a DTE that is unable to recognize data and polling messages, either due to severe variations in channel characteristics or simply because that particular receiver was not powered on during the initial synchronization of the network. Clearly, in a large or heavily loaded multipoint network, data throughput is increased and the burden of monitoring the network is eased if some form of *blind equalization* is built into the receiver design (Godard, 1980).

In wireless communications, the use of blind equalization offers the following advantages over traditional (i.e., supervised) adaptative equalization techniques:

- Unsupervised (self-organized) learning does away with the need for an externally supplied desired response at the receiver.
- Spectral efficiency is improved by avoiding the waste of time in transmission over the channel, thus preserving channel bandwidth.

In reflection seismology, the traditional method of removing the source waveform from a seismogram is to use *linear-predictive deconvolution* (which was discussed in Chapter 6). The method of predictive deconvolution is derived from four fundamental assumptions (Gray, 1979):

1. The reflectivity series is *white*. This assumption is, however, often violated by reflection seismograms, as the reflectivities result from a differential process applied to acoustic impedances. In many sedimentary basins, there are thin beds that cause the reflectivity series to be correlated in sign.
2. The source signal is *minimum phase*, in that its z-transform has all of its zeros confined to the interior of the unit circle in the z-plane; here, it is presumed that the source

signal is in discrete-time form. This assumption is valid for several explosive sources (e.g., dynamite), but it is only approximate for more complicated sources, such as those used in marine exploration.

3. The reflectivity series and noise are *statistically independent* and *stationary in time*. The stationarity assumption, however, is violated because of spherical divergence and attenuation of seismic waves. To cope with nonstationarity of the data, we may use adaptive deconvolution, but such a method often destroys primary events of interest.
4. The *minimum mean-square-error criterion* is used to solve the linear prediction problem. This criterion is appropriate only when the prediction errors (the reflectivity series and noise) have a Gaussian distribution. Statistical tests performed on reflectivity series, however, show that their kurtosis is much higher than that expected from a Gaussian distribution. The *skewness* and *kurtosis* of a distribution function are respectively defined as

$$\gamma_1 = \frac{\mu_3}{\sigma^3} \tag{17.1}$$

and

$$\gamma_2 = \frac{\mu_4}{\sigma^4} - 3, \tag{17.2}$$

where σ^2 is the variance of the distribution and μ_3 and μ_4 are its third- and fourth-order central moments, respectively. For a real-valued Gaussian distribution with zero mean, both γ_1 and γ_2 are zero.

The main point to note here is that valuable phase information contained in a reflection seismogram is ignored by the method of predictive deconvolution. This limitation is overcome by using *blind deconvolution* (Godfrey & Rocca, 1981).

Blind equalization in digital communications and blind deconvolution in reflection seismology are examples of a special kind of adaptive inverse filtering that operates in an *unsupervised* manner. Only the received signal and some additional information in the form of a *probabilistic source model* are provided. In the case of equalization for digital communications, the model describes the statistics of the transmitted data sequence. In the case of seismic deconvolution, the model describes the statistics of the earth's reflection coefficients.

Approaches to Blind Deconvolution

Expanding on the brief discussion on blind deconvolution just presented, the literature on blind deconvolution algorithms and their practical applications has grown steadily during the past three to four decades, so much so that several books have been written on the subject.[1]

[1]For books on the many facets of blind deconvolution, the reader is referred to the following list:

1. Haykin (1994): This is the first edited book on blind convolution.
2. Haykin (2000): This edited book covers two volumes, the first dealing with source separation and the second dealing with blind deconvolution; these two signal-processing operations are related.

(*continued*)

With the limited space we have in this single chapter of the book, however, the study of blind deconvolution will be confined to the following special, yet important, class of systems:

> Non-minimum-phase, linear time-invariant systems of a single-input, single-output (SISO) kind, for which the received signal (i.e., measurements) is non-Gaussian.

To be more precise, we focus on two types of blind deconvolution algorithms, one linear and the other nonlinear.

1. *Linear blind deconvolution.* Here, the received signal is purposely *oversampled* at an integer multiple of the symbol rate. Thereby, the received signal is converted into a multichannel signal. Given this new representation of the received signal, the task confronting us is how to extract second-order information about the system (i.e., channel) being considered so as to make up for the unavailability of the system input, or desired response. One interesting way of accomplishing this difficult task is to look to *cyclostationarity*, an inherent characteristic of modulated signals in communications, as the source of the desired second-order information.

   The remarkable point to note here is the fact that it is, indeed, possible to exploit the cyclostationarity property of the received signal for the purpose of blind system (i.e., closed) identification, despite the fact that we have no access to the system input (i.e., source signal). The solution is based on a block-oriented, subspace decomposition procedure (Tong et al., 1995); the price to be paid for this remarkable achievement is a computationally intensive identification procedure.[2]

3. Ding and Li (2001): This book tries to clarify the capabilities and limitations of blind equalization algorithms and addresses algorithms for single-input, multiple-output (SIMO) systems.
4. Tugnait (2003): This chapter of a book presents a tutorial exposition of various approaches to blind channel equalization of the single-input, single-output (SISO) type; it addresses multiple and limited channel bandwidth as two practical issues of interest as well as practical applications of blind equalization.
5. Campisi and Egiazarian (2007): This edited book explores various image deconvolution algorithms in different areas—namely, image restoration, microscopy, medical imaging, remote sensing, geophysical prospecting, and many other applications.
6. Pinchas (2012): This book describes mathematical aspects of a novel approach to blind channel equalization by exploiting the maximum entropy principle and comparing it to traditional approaches.

[2]Other ways of implementing linear blind deconvolution are summarized here:

- *Minimal noise subspace (Hua et al., 1997)*: This paper introduces the notion of a minimal noise subspace, which is computed from a set of channel pairs that form a *tree*. The tree exploits, with minimum redundancy, the diversity among the subchannels. It is claimed that this new procedure is computationally more efficient than the subspace method attributed to Moulines et al. (1995).
- *Linear prediction theory (Slock, 1994; Slock and Papadias, 1995; Mannerkoski & Taylor, 1999; Papadias & Slock, 1999)*: Unlike the subspace-decomposition procedure described in Section 17.3 on blind identification, the motivation for the prediction approach is blind equalization. The approach described in Mannerkoski and Taylor (1999) is based on the use of least-square lattice prediction (i.e., order-recursive adaptive filtering), the theory of which was discussed in Chapter 16.
- *Mutually referenced filters (Gesbert et al., 1997)*: In this approach, several filters are considered, the outputs of which act as training signals for each other during the adaptive procedure.

2. *Nonlinear blind deconvolution.* Linear filters are associated with second-order statistics (SOS), whereas nonlinear filters are associated with higher-order statistics (HOS). The HOS of the received signal are exploited in one of two ways:

   - *Explicit sense*, which is exemplified by the use of higher-order cumulants or their discrete Fourier transforms known as polyspectra. (See Chapter 1.) The property that polyspectra have of preserving phase information makes explicit HOS-based algorithms well suited for blind deconvolution (Pan & Nikias, 1988; Hatzinakos & Nikias, 1989, 1991).
   - *Implicit sense*, in which case the HOS of the received signal are exploited indirectly; the implicit HOS-based algorithms include Bussgang algorithms, so called because the deconvolved signal exhibits Bussgang statistics when the algorithm converges in the mean value (Bellini, 1986, 1988, 1994).

   Implicit HOS-based blind deconvolution algorithms are relatively simple to implement and are generally capable of delivering a good performance, as is evidenced by their actual use in digital communication systems. However, they suffer from two basic limitations: a potential for converging to a local minimum and sensitivity to timing jitter. In contrast, explicit HOS-based blind deconvolution algorithms overcome the local minimum problem by avoiding the need for minimizing a cost function; unfortunately, they are computationally much more complex. Perhaps the most serious limitation of both kinds of algorithm is their slow rate of convergence. To appreciate the reason for this poor behavior, we have to recognize that the time-average estimation of HOS requires a much larger sample size than is the case for SOS. According to Brillinger (1975), the sample size needed to estimate the nth-order statistics of a stochastic process, subject to prescribed values of estimation bias and variance, increases almost exponentially with order n. It is therefore not surprising that HOS-based blind deconvolution algorithms exhibit a slow rate of convergence compared with conventional adaptive filtering algorithms that rely on a training sequence for their operation. Thus, whereas a traditional adaptive filtering algorithm may require a few hundred adaptation cycles to converge, an HOS-based blind deconvolution algorithm may require several thousand adaptation cycles to converge. This slow rate of convergence is of no serious concern in some applications, such as seismic deconvolution. However, in a highly nonstationary environment, such as mobile digital communications, the algorithm may simply not have enough time to reach a steady state and may therefore be unable to track the statistical variations of the environment. Accordingly, these

Correspondingly, a constrained cost function in the form of a multidimensional mean-square error is devised, the minimization of which appears to provide a necessary and sufficient condition for equalization. The underlying theory of the procedure is linked to linear prediction.

- *Linear neural network (Fang & Chow, 1999)*: This approach uses a network (following an oversampler) that consists of two linear layers connected in cascade. The first layer whitens the multichannel output of the sampler. The second (output) layer is designed to optimize the estimate of the transmitted data symbol. The stochastic learning algorithms used to compute the adjustable weights of the two layers are applied separately.

All of the foregoing procedures share a common feature: They involve oversampling, as indicated in the oversampled channel model of Fig. 17.2.

blind deconvolution algorithms cannot be used in applications in which rapid acquisition is a system requirement.

Organization of the Chapter

The linear type of deconvolution algorithms will occupy Sections 17.2 and 17.3 of this chapter. The material presented therein is aimed at the identification of a communication channel by exploiting the SOS attributed to cyclostationarity of the channel output.

When we come to the nonlinear type of blind deconvolution algorithms, the material is organized as follows:

- Sections 17.4 through 17.6 are devoted to the family of Bussgang algorithms for blind channel equalization; this particular family has a long history that goes back to the classic article by Sato (1975a). Bussgang algorithms have been extensively applied to communication systems, in which the channel may change slowly over time.
- In Section 17.7 we study fractional equalizers, in which SOS-based and Bussgang algorithms are tied together; in a loose sense, we may view these blind deconvolution algorithms to be complementary.
- Section 17.8 is devoted to a novel approach for nonlinear blind deconvolution that is based on the maximum entropy principle (Pinchas & Bobrovsky, 2006; Pinchas 2012). Derivation of this algorithm for channel equalization is highly demanding in mathematical terms.

It is important to recognize that when we speak of nonlinear blind deconvolution algorithms, some form of nonlinearity is built into their structure so as to facilitate the self-organized computation of a desired response.

17.2 CHANNEL IDENTIFIABILITY USING CYCLOSTATIONARY STATISTICS

In HOS-based deconvolution algorithms, information about the unknown phase response of a non-minimum-phase channel is extracted by using the HOS of the channel output, which is sampled at the baud rate (i.e., the symbol rate). Alternatively, we may extract this phase information by exploiting another inherent characteristic of the channel output, namely, *cyclostationarity*. To explain this latter characteristic, we first write the received signal in a digital communications system in its most general baseband form as

$$u(t) = \sum_{k=-\infty}^{\infty} x_k h(t - kT) + \nu(t), \tag{17.3}$$

where a symbol x_k is transmitted every T seconds (i.e., $1/T$ is the baud rate) and t denotes continuous time; $h(t)$ is the overall impulse response of the channel (including transmit and receive filters), and $\nu(t)$ is the channel noise. (The channel noise ν used here should not be confused with the convolutional noise ν to be used later in the discussion of the Bussgang algorithm.) All the quantities described in Eq. (17.3) are complex valued. Under the assumption that the transmitted sequence x_k and the channel noise $v(t)$ are

both wide-sense stationary with zero mean, we may readily show that the received signal $u(t)$ also has zero mean and its autocorrelation function is *periodic* in the symbol duration T (see Problem 1):

$$\begin{aligned} r_u(t_1, t_2) &= \mathbb{E}[u(t_1)u^*(t_2)] \\ &= r_u(t_1 + T, t_2 + T). \end{aligned} \tag{17.4}$$

That is, the received signal $u(t)$ is *cyclostationary in the wide sense*. The asterisk denotes *complex conjugation.*

What makes the use of cyclostationarity particularly attractive as the basis of an approach to blind deconvolution is the fact that it uses only *second-order statistics*, thereby overcoming the "slow-to-converge" limitation of HOS-based algorithms.

Apparently, Gardner (1991) was the first to recognize that cyclostationary characteristics of modulated signals permit the recovery of a communication channel's amplitude and phase responses using SOS only. However, the idea of blind channel identification and equalization using cyclostationary statistics is attributed to Tong et al. (1995). Indeed, the ability to solve the difficult problem of blind deconvolution on the sole basis of SOS deserves to be viewed as a clever algorithmic achievement. The original idea proposed by Tong et al. relies on the use of *temporal diversity* (i.e., oversampling the received signal). Ordinarily, this operation is performed in a digital communications system for the specific purpose of timing and phase recovery. However, in the context of our present discussion, the use of oversampling leads to a *fractionally spaced equalizer* (FSE), which is so called because the equalization taps are spaced more closely than the reciprocal of the incoming symbol rate.

Among the many fractionally spaced blind channel identification or equalization techniques that have been proposed to date, we have picked a *subspace decomposition method* that bears a close relationship to the *mu*ltiple *si*gnal *c*lassification (MUSIC) algorithm originally proposed by Schmidt (1979) for estimating the angle of arrival of the signal. Thus, the material presented in the next section points to the fact that much can be gained from the extensive literature on statistical array signal processing in solving the blind deconvolution problem.

17.3 SUBSPACE DECOMPOSITION FOR FRACTIONALLY SPACED BLIND IDENTIFICATION

As previously mentioned, the subspace decomposition method begins by oversampling the received signal. Suppose, then, that the received signal $u(t)$ is oversampled by setting

$$t = \frac{iT}{L}, \tag{17.5}$$

where T is the symbol period and L is a positive integer. Then, Eq. (17.3) takes on the discrete form

$$u\left(\frac{iT}{L}\right) = \sum_{k=-\infty}^{\infty} x_k h\left(\frac{iT}{L} - kT\right) + \nu\left(\frac{iT}{L}\right). \tag{17.6}$$

Let

$$i = nL + l, \qquad l = 0, 1, \ldots, L - 1. \tag{17.7}$$

Accordingly, we may rewrite Eq. (17.6) as

$$u\left(nT + \frac{lT}{L}\right) = \sum_{k=-\infty}^{\infty} x_k h\left((n-k)T + \frac{lT}{L}\right) + \nu\left(nT + \frac{lT}{L}\right). \tag{17.8}$$

For convenience of presentation, let

$$h_n^{(l)} = h\left(nT + \frac{lT}{L}\right),$$

$$u_n^{(l)} = u\left(nT + \frac{lT}{L}\right),$$

and

$$\nu_n^{(l)} = \nu\left(nT + \frac{lT}{L}\right).$$

Correspondingly, we may define the oversampled channel output of Eq. (17.8) in the simplified form

$$u_n^{(l)} = \sum_{k=-\infty}^{\infty} x_k h_{n-k}^{(l)} + \nu_n^{(l)}, \qquad l = 0, 1, \ldots, L - 1. \tag{17.9}$$

To proceed further, we make two assumptions:

1. The channel is causal with a finite time support; that is,

$$h_k^{(l)} = 0 \qquad \text{for } k < 0 \text{ or } k > M \text{ and for all } l, \tag{17.10}$$

where M is the true channel order.

2. At time n, the receiver processes the channel output due to a transmitted signal vector that consists of $(M + N)$ symbols; that is,

$$\mathbf{x}_n = [x_n, x_{n-1}, \ldots, x_{n-M-N+1}]^{\mathrm{T}}. \tag{17.11}$$

Thus, at the receiving end, we find that each data *block* consists of NL samples. Depending on how these samples are grouped together, we may distinguish two equivalent matrix representations for the oversampled channel:

1. *A single-input, multiple-output (SIMO) model*, which consists of L *virtual channels* (subchannels) fed from a common input, as depicted in Fig. 17.2. Each virtual channel has a similar time support and a noise contribution of its own. Let the lth virtual channel be characterized with the following:
   - An $(M + 1)$-by-1 tap-weight (coefficient) vector

$$\mathbf{h}^{(l)} = [h_0^{(l)}, h_1^{(l)}, \ldots, h_M^{(l)}]^{\mathrm{T}}.$$

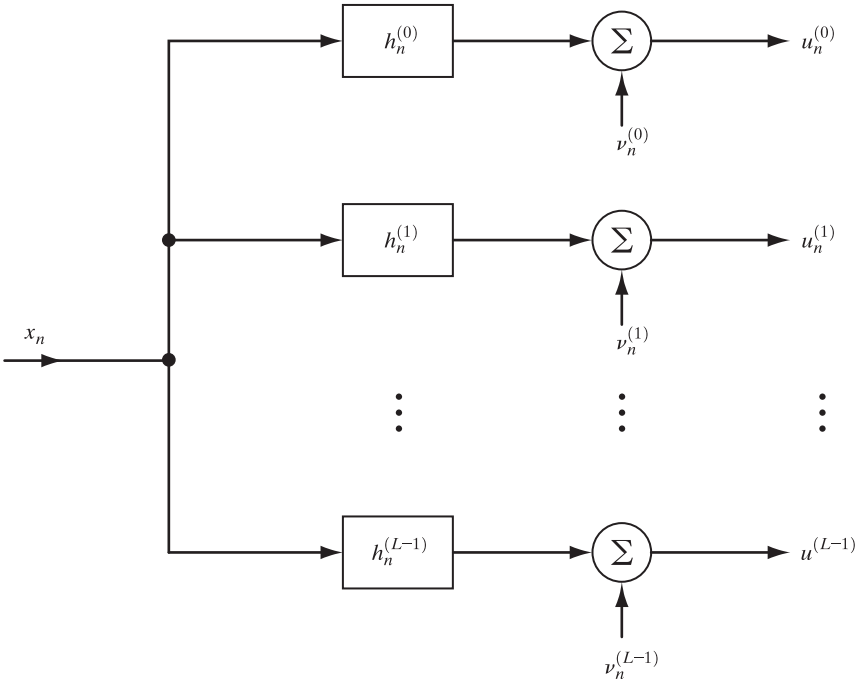


FIGURE 17.2 Representation of an oversampled channel as a single-input, multiple-output (SIMO) model.

- An N-by-1 received signal vector

$$\mathbf{u}_n^{(l)} = [u_n^{(l)}, u_{n-1}^{(l)}, \ldots, u_{n-N+1}^{(l)}]^{\mathrm{T}}.$$

- An N-by-1 noise vector

$$\boldsymbol{\nu}_n^{(l)} = [\nu_n^{(l)}, \nu_{n-1}^{(l)}, \ldots, \nu_{n-N+1}^{(l)}]^{\mathrm{T}}.$$

We may then represent Eq. (17.9), written for N successive received samples, in the compact form

$$\mathbf{u}_n^{(l)} = \mathbf{H}^{(l)}\mathbf{x}_n + \boldsymbol{\nu}_n^{(l)}, \qquad l = 0, 1, \ldots, L-1, \tag{17.12}$$

where the transmitted signal vector $\mathbf{x}_n$ is defined in Eq. (17.11). The N-by-$(M+N)$ matrix $\mathbf{H}^{(l)}$, termed the *filtering matrix*, has a Toeplitz structure, as shown by

$$\mathbf{H}^{(l)} = \begin{bmatrix} h_0^{(l)} & h_1^{(l)} & \cdots & h_M^{(l)} & 0 & \cdots & 0 \\ 0 & h_0^{(l)} & \cdots & h_{M-1}^{(l)} & h_M^{(l)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_0^{(l)} & h_1^{(l)} & \cdots & h_M^{(l)} \end{bmatrix}. \tag{17.13}$$

Finally, combining the set of L equations (17.12) into a single relation, we may write

$$\boldsymbol{u}_n = \mathcal{H}\mathbf{x}_n + \boldsymbol{v}_n, \tag{17.14}$$

where

$$\boldsymbol{u}_n = \begin{bmatrix} \mathbf{u}_n^{(0)} \\ \mathbf{u}_n^{(1)} \\ \vdots \\ \mathbf{u}_n^{(L-1)} \end{bmatrix}$$

is the LN-by-1 multichannel received signal vector,

$$\boldsymbol{v}_n = \begin{bmatrix} \boldsymbol{\nu}_n^{(0)} \\ \boldsymbol{\nu}_n^{(1)} \\ \vdots \\ \boldsymbol{\nu}_n^{(L-1)} \end{bmatrix}$$

is the LN-by-1 multichannel noise vector, and

$$\mathcal{H} = \begin{bmatrix} \mathbf{H}^{(0)} \\ \mathbf{H}^{(1)} \\ \vdots \\ \mathbf{H}^{(L-1)} \end{bmatrix} \tag{17.15}$$

is the LN-by-$(M+N)$ *channel convolutional matrix*, in which the individual matrix entries are defined in Eq. (17.13).

2. *Sylvester matrix representation*, wherein the L virtual channel coefficients having the same delay index are all grouped together. Specifically, we write

$$\mathbf{h}'_k = [h_k^{(0)}, h_k^{(1)}, \ldots, h_k^{(L-1)}]^{\mathrm{T}}, \qquad k = 0, 1, \ldots, M,$$

and, correspondingly, we define an L-by-1 received signal vector

$$\mathbf{u}'_n = [u_n^{(0)}, u_n^{(1)}, \ldots, u_n^{(L-1)}]^{\mathrm{T}}$$

and an L-by-1 noise vector

$$\boldsymbol{\nu}'_n = [\nu_n^{(0)}, \nu_n^{(1)}, \ldots, \nu_n^{(L-1)}]^{\mathrm{T}}.$$

Then, on this basis, we may use Eq. (17.9) to group the NL received samples as

$$\begin{aligned} \boldsymbol{u}'_n &= \begin{bmatrix} \mathbf{u}'_n \\ \mathbf{u}'_{n-1} \\ \vdots \\ \mathbf{u}'_{n-N+1} \end{bmatrix} \\ &= \mathcal{H}'\mathbf{x}_n + \boldsymbol{v}'_n, \end{aligned} \tag{17.16}$$

where the transmitted signal vector $\mathbf{x}_n$ is as previously defined in Eq. (17.11). The LN-by-1 noise vector $\boldsymbol{v}'_n$ is defined by

$$\boldsymbol{\nu}'_n = \begin{bmatrix} \boldsymbol{\nu}'_n \\ \boldsymbol{\nu}'_{n-1} \\ \vdots \\ \boldsymbol{\nu}'_{n-N+1} \end{bmatrix},$$

and the LN-by-$(M+N)$ *matrix* $\mathcal{H}'$ is defined by

$$\mathcal{H}' = \begin{bmatrix} \mathbf{h}'_0 & \mathbf{h}'_1 & \cdots & \mathbf{h}'_M & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{h}'_0 & \cdots & \mathbf{h}'_{M-1} & \mathbf{h}'_M & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{h}'_0 & \mathbf{h}'_1 & \cdots & \mathbf{h}'_M \end{bmatrix}. \tag{17.17}$$

The block-Toeplitz matrix $\mathcal{H}'$ is called a *Sylvester resultant matrix* (Rosenbrock, 1970; Tong et al., 1993)—hence the terminology used to refer to this second matrix representation of an oversampled channel.

It is of interest to note that a SIMO channel, as described herein, can also be obtained by using an array of L sensors or antennas.

Filtering-Matrix Rank Theorem

The matrices $\mathcal{H}$ and $\mathcal{H}'$, defined in Eqs. (17.15) and (17.17), respectively, differ primarily in the way in which their individual rows are arranged; they contain the same information about the channel but display it differently. Most importantly, the spaces spanned by the columns of $\mathcal{H}$ and $\mathcal{H}'$ are *canonically equivalent*. Therefore, from here on, we restrict the discussion to the SIMO model of Fig. 17.2.

The multichannel filtering matrix $\mathcal{H}$ plays a central role in the blind identification problem. In particular, the problem is solvable if and only if $\mathcal{H}$ is of *full column rank*. This requirement is covered by an important theorem attributed to Tong et al. (1993), which may be stated as follows:

> The LN-by-$(M+N)$ channel convolutional matrix $\mathcal{H}$ is of full column rank [i.e., rank $(\mathcal{H}) = M + N$], provided that the following three conditions are satisfied:
>
> **1.** The polynomials
>
> $$H^{(l)}(z) = \sum_{m=0}^{M} h_m^{(l)} z^{-m} \qquad \text{for } l = 0, 1, \ldots, L-1$$
>
> have *no* common zeros.
>
> **2.** At least one of the polynomials $H^{(l)}(z)$, $l = 0, 1, \ldots, L-1$, has the maximum possible degree M; this condition is commonly called the *channel disparity condition*.
>
> **3.** The size N of the received signal vector $\mathbf{u}_n^{(l)}$ for each virtual channel is greater than M.

Equipped with this theorem, hereafter referred to as the *filtering-matrix rank theorem*, we are ready to describe the subspace decomposition-based procedure for blind identification.

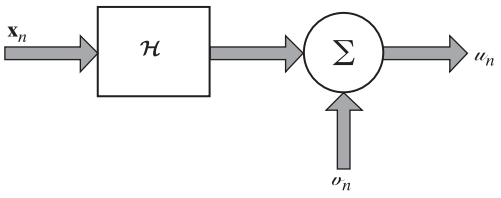


FIGURE 17.3 Matrix representation of an oversampled channel.

Blind Identification

The basic equation (17.14) provides a matrix description of an oversampled channel. A block diagram representation of this equation is shown in Fig. 17.3, which may be viewed as a condensed version of the SIMO model of Fig. 17.2. To proceed with a statistical characterization of the channel, we make the following assumptions:

- The transmitted signal vector $\mathbf{x}_n$ and the multichannel noise vector $\mathbf{v}_n$ originate from wide-sense stationary sources that are statistically independent.
- The transmitted signal vector $\mathbf{x}_n$ has zero mean and correlation matrix

$$\mathbf{R}_x = \mathbb{E}[\mathbf{x}_n \mathbf{x}_n^{\mathrm{H}}],$$

  where the superscript H denotes *Hermitian transposition* (i.e., transposition combined with complex conjugation). The $(M + N)$-by-$(M + N)$ matrix $\mathbf{R}_x$ has full column rank, but it is unknown.
- The additive channel noise is white, permitting us to characterize the noise vector $\boldsymbol{\nu}_n$ with zero mean and correlation matrix

$$\begin{aligned}\mathbf{R}_\nu &= \mathbb{E}[\boldsymbol{\nu}_n \boldsymbol{\nu}_n^{\mathrm{H}}] \\ &= \sigma^2 \mathbf{I}.\end{aligned}$$

  The noise variance σ^2 is known, and $\mathbf{I}$ is the N-by-N identity matrix.

Accordingly, the LN-by-1 received signal vector u_n has zero mean and a correlation matrix defined by

$$\begin{aligned}\mathbf{R} &= \mathbb{E}[u_n u_n^{\mathrm{H}}] \\ &= \mathbb{E}[(\mathcal{H}\mathbf{x}_n + v_n)(\mathcal{H}\mathbf{x}_n + v_n)^{\mathrm{H}}] \\ &= \mathbb{E}[\mathcal{H}\mathbf{x}_n \mathbf{x}_n^{\mathrm{H}} \mathcal{H}^{\mathrm{H}}] + \mathbb{E}[v_n v_n^{\mathrm{H}}] \\ &= \mathcal{H}\mathbf{R}_x \mathcal{H}^{\mathrm{H}} + \mathbf{R}_v,\end{aligned} \tag{17.18}$$

where

$$\begin{aligned}\mathbf{R}_v &= \mathbb{E}[v(n) v^{\mathrm{H}}(n)] \\ &= \sigma^2 \mathbf{I}_{LN},\end{aligned} \tag{17.19}$$

in which $\mathbf{I}_{LN}$ is the LN-by-LN identity matrix. It should be noted that the signal-induced term $\mathcal{H}\mathbf{R}_x\mathcal{H}^{\mathrm{H}}$ has rank $M + N$, which is less than the dimension LN of the correlation

matrix $\mathbf{R}$. The singularity of the signal-induced part of $\mathbf{R}$ is indeed the key feature exploited in all subspace estimation algorithms.

To gain some insight into the blind identification problem, we cast it in a geometric framework originally proposed by Schmidt (1979, 1981). First, we invoke the spectral theorem of Appendix E to describe the LN-by-LN correlation matrix

$$\mathbf{R} = \sum_{k=0}^{LN-1} \lambda_k \mathbf{q}_k \mathbf{q}_k^{\mathrm{H}} \tag{17.20}$$

in terms of its eigenvalues and associated eigenvectors, where the eigenvalues are arranged in decreasing order,

$$\lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_{LN-1}.$$

Next, we invoke the filtering-matrix rank theorem to divide these eigenvalues into two groups:

1. $\lambda_k > \sigma^2, \qquad k = 0, 1, \ldots, M + N - 1.$
2. $\lambda_k = \sigma^2, \qquad k = M + N, M + N + 1, \ldots, LN - 1.$

Correspondingly, the space spanned by the eigenvectors of matrix $\mathbf{R}$ is divided into two subspaces:

1. *Signal subspace* $\mathcal{S}$, spanned by the eigenvectors associated with the eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_{M+N-1}$; these eigenvectors are written as

$$\mathbf{s}_k = \mathbf{q}_k, \qquad k = 0, 1, \ldots, M + N - 1.$$

2. *Noise subspace* $\mathcal{N}$, spanned by the eigenvectors associated with the remaining eigenvalues, as shown by

$$\mathbf{g}_k = \mathbf{q}_k, \qquad k = 0, 1, \ldots, M + N - 1.$$

The noise subspace is the *orthogonal complement* of the signal subspace. By definition,

$$\mathbf{R}\mathbf{g}_k = \sigma^2 \mathbf{g}_k, \qquad k = 0, 1, \ldots, LN - M - N - 1. \tag{17.21}$$

Substituting Eq. (17.18) with $\mathbf{R}_\nu = \sigma^2 \mathbf{I}$ into Eq. (17.21) and then simplifying, we get

$$\mathcal{H}\mathbf{R}_x \mathcal{H}^{\mathrm{H}} \mathbf{g}_k = \mathbf{0}, \qquad k = 0, 1, \ldots, LN - M - N - 1.$$

Since both matrices $\mathcal{H}$ and $\mathbf{R}_x$ are of full column rank, it follows that

$$\mathcal{H}^{\mathrm{H}} \mathbf{g}_k = \mathbf{0}, \qquad k = 0, 1, \ldots, LN - M - N - 1. \tag{17.22}$$

Equation (17.22) provides the theoretic framework of the *subspace decomposition procedure for blind identification* described in Moulines et al. (1995). The procedure builds on two items:

1. Knowledge of the eigenvectors associated with the $LN - M - N$ smallest eigenvalues of the correlation matrix $\mathbf{R}$ of the received signal vector $\boldsymbol{u}_n$.
2. Orthogonality of the columns of the unknown multichannel filtering matrix $\mathcal{H}$ to the noise subspace $\mathcal{N}$.

In other words, the cyclostationary statistics of the received signal u_n, exemplified by the correlation matrix $\mathbf{R}$, are indeed sufficient for blind identification of the channel, to within a multiplicative constant.

Alternative Formulation of the Orthogonality Condition

From a computational point of view, we find it more convenient to work with an alternative formulation of the *orthogonality condition* described in Eq. (17.22). To begin with, we rewrite this condition in the equivalent scalar form

$$\|\mathcal{H}^{\mathrm{H}}\mathbf{g}_k\|^2 = \mathbf{g}_k^{\mathrm{H}}\mathcal{H}\,\mathcal{H}^{\mathrm{H}}\mathbf{g}_k = 0, \qquad k = 0, 1, \ldots, LN - M - N - 1. \tag{17.23}$$

Recognizing the partitioned structure of the channel convolutional matrix $\mathcal{H}$ displayed in Eq. (17.15), in a corresponding way we may partition the LN-by-1 eigenvector $\mathbf{g}_k$ as

$$\mathbf{g}_k = \begin{bmatrix} \mathbf{g}_k^{(0)} \\ \mathbf{g}_k^{(1)} \\ \vdots \\ \mathbf{g}_k^{(L-1)} \end{bmatrix}, \tag{17.24}$$

where $\mathbf{g}_k^{(l)}$, $l = 0, 1, \ldots, L - 1$, is an N-by-1 vector. Next, guided by the composition of matrix $\mathbf{H}^{(l)}$ given in Eq. (17.13), we formulate the $(M+1)$-by-$(M+N)$ matrix

$$\mathbf{G}_k^{(l)} = \begin{bmatrix} g_{k,0}^{(l)} & g_{k,1}^{(l)} & \cdots & g_{k,N-1}^{(l)} & 0 & \cdots & 0 \\ 0 & g_{k,0}^{(l)} & \cdots & g_{k,N-2}^{(l)} & g_{k,N-1}^{(l)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & g_{k,0}^{(l)} & g_{k,1}^{(l)} & \cdots & g_{k,N-1}^{(l)} \end{bmatrix}. \tag{17.25}$$

Finally, in light of Eq. (17.15) describing the channel convolutional matrix $\mathcal{H}$, we use the matrices defined in Eq. (17.25) for $l = 0, 1, \ldots, L - 1$ to set up the $L(M+1)$-by-$(M+N)$ matrix

$$\mathcal{G}_k = \begin{bmatrix} \mathbf{G}_k^{(0)} \\ \mathbf{G}_k^{(1)} \\ \vdots \\ \mathbf{G}_k^{(L-1)} \end{bmatrix}, \qquad k = 0, 1, \ldots, LN - M - N - 1. \tag{17.26}$$

Given the $\mathcal{G}_k$ just defined, we may show that (see Problem 3)

$$\mathbf{g}_k^{\mathrm{H}}\mathcal{H}\mathcal{H}^{\mathrm{H}}\mathbf{g}_k = \mathbf{h}^{\mathrm{H}}\mathcal{G}_k\mathcal{G}_k^{\mathrm{H}}\mathbf{h}, \tag{17.27}$$

where

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}^{(0)} \\ \mathbf{h}^{(1)} \\ \vdots \\ \mathbf{h}^{(L-1)} \end{bmatrix}$$

is an $L(M+1)$-by-1 vector defined in terms of the multichannel coefficients. Accordingly, we may reformulate the orthogonality condition of Eq. (17.23) in the equivalent form

$$\mathbf{h}^{\mathrm{H}}\mathcal{G}_k\mathcal{G}_k^{\mathrm{H}}\mathbf{h} = 0, \qquad k = 0, 1, \ldots, LN - M - N - 1, \tag{17.28}$$

which is the desired relation. In Eq. (17.28), the unknown multichannel coefficients feature in the simple form of vector $\mathbf{h}$, whereas in Eq. (17.23), they feature in the highly elaborate structure of matrix $\mathcal{H}$.

Estimation of the Channel Coefficients

In practice, we have to work with estimates of the eigenvectors $\mathbf{g}_k$. Let these estimates be denoted by $\hat{\mathbf{g}}_k$, $k = 0, 1, \ldots, LN - M - N - 1$. To derive a corresponding estimate of the multichannel coefficient vector $\mathbf{h}$, we use the orthogonality condition of Eq. (17.28) to define the cost function

$$\mathscr{E}(\mathbf{h}) = \mathbf{h}^{\mathrm{H}}\mathcal{Q}\mathbf{h}, \tag{17.29}$$

where

$$\mathcal{Q} = \sum_{k=0}^{LN-M-N-1} \hat{\mathcal{G}}_k \hat{\mathcal{G}}_k^{\mathrm{H}} \tag{17.30}$$

is an $L(M + 1)$-by-$L(M + 1)$ matrix. The estimated matrix $\hat{\mathcal{G}}_k$ is itself defined by Eqs. (17.26) and (17.25) with $\hat{\mathbf{g}}_k$ used in place of $\mathbf{g}_k$. In the ideal case of a true correlation matrix $\mathbf{R}$, the true multichannel coefficient vector $\mathbf{h}$ is uniquely defined (except for a multiplicative constant) by the condition $\mathscr{E}(\mathbf{h}) = 0$. Working with the matrix $\mathcal{Q}$ based on the estimate's $\hat{\mathbf{g}}_k$, a least-squares estimate of the vector $\mathbf{h}$ is computed by minimizing the cost function $\mathscr{E}(\mathbf{h})$. However, this minimization would have to be performed subject to a properly chosen constraint, so as to avoid the trivial solution $\mathbf{h} = \mathbf{0}$. Moulines et al. (1995) suggest two possible optimization criteria:

1. *Linear constraint.* Minimize the cost function $\mathscr{E}(\mathbf{h})$ subject to $\mathbf{c}^{\mathrm{H}}\mathbf{h} = \mathbf{1}$, where $\mathbf{c}$ is some $L(M+1)$-by-1 vector.
2. *Quadratic constraint.* Minimize the cost function $\mathscr{E}(\mathbf{h})$ subject to $\|\mathbf{h}\| = 1$.

The first criterion requires the prescription of an arbitrary vector $\mathbf{c}$, whereas the second criterion appears to be more natural, but computationally more demanding.

Practical Considerations

The subspace decomposition theory embodied by Eqs. (17.5) through (17.30) is rather idealized, in that its successful use rests on three assumptions:

1. The additive channel noise $\nu(t)$ is white with known variance σ^2.
2. The transfer functions of the virtual channels in the model of Fig. 17.2 have no common zeros.
3. The channel order M is known.

Assumption 1 is reasonable. To test for the validity of Assumption 2, we require exact knowledge of the channel order M, which is the essence of Assumption 3. Unfortunately, such information is not available in practice. We are therefore compelled to estimate the channel order by working with an oversampled data matrix.

The issue of determining the order of the model is complicated by another practical reality: Typically, the impulse response of a communication channel is long and decomposable into two parts:

1. *The significant part*, which lies in the middle and constitutes the dominant part of the impulse response.
2. *The tails*, which are made up of relatively small leading or trailing parts of the impulse response.

On this basis, we may define an *effective channel order* as the order that pertains to the significant part of the channel impulse response. The effective channel order is naturally smaller than the true channel order. In Liavas et al. (1999a), it is shown that, in a blind channel identification problem, only the significant part of the channel impulse response should be modeled. The reason for this restriction is that, in trying to model the tails of the channel impulse response as well, we may end up with an ill-conditioned overmodeled situation, which, for computational reasons, should be avoided. The influence of ignoring the tails of the channel impulse response is to introduce a *perturbation* that may be viewed as a form of colored noise.

Why Classical Information-Theoretic Criteria Do Not Work Well for Determining the Effective Channel Order

To estimate the unknown effective channel order M, we could resort to the classical information-theoretic criteria, namely, Akaike's information-theoretic criterion (AIC) and Rissanen's minimum-description-length (MDL) criterion. From the discussion presented in Chapter 1, we recall that the derivations of these criteria are based on two assumptions: The successive data vectors are i.i.d. zero-mean Gaussian random vectors, and the additive noise is white, Gaussian, and uncorrelated with the information-bearing signal at the channel output. Unfortunately, in certain respects, the validity of these two assumptions is questionable for the blind channel identification problem, for the following reasons:

- The correlation matrix $\mathbf{R}$ in Eq. (17.18) is constructed from data vectors that exhibit the time-shifting property, with the result that successive data vectors are *not* statistically independent, thereby invalidating the first assumption.
- The additive colored noise, accounting for the influence of the channel impulse response's tails that are ignored in determining the effective channel order, invalidates the second assumption.

Indeed, using measured real-life microwave radio channel data, Liavas et al. (1999b) have shown that the model-order estimates based on the AIC and MDL criteria are sensitive to variations in the signal-to-noise ratio (SNR) and the number of data samples used in the estimation. Specifically, we may say the following:

- For high SNR (SNR $>$ 30 dB) and many data samples ($N > 300$), these two information-theoretic criteria lead to overmodeling, which makes the resulting model-order estimates practically useless.
- For low SNR and few data samples, the AIC and MDL criteria provide useful estimates leading to sufficiently good blind identification results. However,

the estimates are highly sensitive to statistical variations, rendering the results unsatisfactory for practical use. Moreover, the sensitivity to statistical variations contributes to the creation of confusion concerning data classification into undermodeled or overmodeled cases, which is clearly undesirable.

Rank Detection Criterion

To overcome the limitations of the classical information-theoretic criteria, Liavas et al. (1999b) proposed a *rank-detection criterion* for determining the effective channel order M. This new criterion is rooted in numerical analysis, which is the natural course to take, recognizing that the SOS-based blind channel identification procedure described herein relies on subspace decomposition ideas.

Let

$$\hat{\mathbf{R}} = \frac{1}{K}\sum_{n=1}^{K} \boldsymbol{u}_n \boldsymbol{u}_n^{\mathrm{H}} \tag{17.31}$$

denote the LN-by-LN estimate of the correlation matrix $\mathbf{R}$, where $\boldsymbol{u}_n$ is the oversampled version of the received signal vector and K is the number of blocks of LN-by-1 data vectors used in the estimation. Assume that $\hat{\mathbf{R}}$ is the sum of two components: an "ideal" matrix $\mathbf{R}_{\text{ideal}}$ of rank p and a "perturbed" matrix $\mathbf{E}$; that is,

$$\hat{\mathbf{R}} = \mathbf{R}_{\text{ideal}} + \mathbf{E}. \tag{17.32}$$

The rank p is itself defined by

$$p = N + M. \tag{17.33}$$

The effective channel order M, and therefore the rank p, is to be determined. The "perturbation" matrix $\mathbf{E}$ is assumed to incorporate the combined influence of (1) the ignored tails of the channel impulse response; (2) the additive channel noise, be it white or colored; and (3) the numerical inaccuracies incurred in computing the estimate $\hat{\mathbf{R}}$. It is also assumed that $\mathbf{E}$ is small compared with $\mathbf{R}_{\text{ideal}}$.

The partitioning of the estimate $\hat{\mathbf{R}}$ into $\mathbf{R}_{\text{ideal}}$ and $\mathbf{E}$ is motivated by physical considerations, namely, the fact that the significant part of the channel impulse response determines the effective channel order M. In reality, however, we do not have access to the channel impulse response. Rather, we have only the estimate $\hat{\mathbf{R}}$ to decompose. Let the spectrally decomposed

$$\hat{\mathbf{R}} = \sum_{k=0}^{LN-1} \hat{\lambda}_k \hat{\mathbf{q}}_k \hat{\mathbf{q}}_k^{\mathrm{H}}. \tag{17.34}$$

Let $\hat{\mathcal{S}}$ denote the "estimated" signal subspace spanned by the eigenvectors associated with the p largest eigenvalues $\hat{\lambda}_0, \hat{\lambda}_1, \ldots, \hat{\lambda}_{p-1}$. Let $\hat{\mathcal{N}}$ denote the "estimated" noise subspace spanned by the eigenvectors associated with the remaining eigenvalues $\hat{\lambda}_p, \hat{\lambda}_{p+1}, \ldots, \hat{\lambda}_{LN-1}$. The noise subspace $\hat{\mathcal{N}}$ is orthogonal to the signal subspace $\hat{\mathcal{S}}$.

What we are seeking is an approximation of the "ideal" subspace $\mathcal{S}_{\text{ideal}}$, namely, the p-dimensional subspace spanned by the columns of the matrix $\mathbf{R}_{\text{ideal}}$. Specifically, we approximate $\mathcal{S}_{\text{ideal}}$ by $\hat{\mathcal{S}}$, computed from $\hat{\mathbf{R}}$. Invoking the assumption that $\mathbf{R}_{\text{ideal}}$ is

of rank p, we derive the following lower bound on the squared Euclidean norm of the perturbation matrix $\mathbf{E}$:

$$\|\mathbf{E}\|^2 \geq \hat{\lambda}_p. \tag{17.35}$$

Here, $\|\mathbf{E}\|$ is the matrix norm of $\mathbf{E}$. (See Appendix E for the definition of matrix norm.) In other words, the unknown "ideal" and the "estimated" signal subspaces are related through a perturbation whose size is equal to or greater than the eigenvalue $\hat{\lambda}_p$.

Since the perturbation matrix $\mathbf{E}$ is assumed to be small compared with the ideal matrix $\mathbf{R}_{\text{ideal}}$, we would like the estimated signal subspace $\hat{\mathcal{S}}$ to be "close" to the ideal signal subspace $\mathcal{S}_{\text{ideal}}$ in some sense to be determined. To that end, we note that although Eq. (17.35) is insufficient to calculate the distance between these two signal subspaces, it points to a way of examining the *sensitivity* of the estimated signal subspace $\hat{\mathcal{S}}$ with respect to a "small" perturbation. In what follows, this perturbation is denoted by an LN-by-LN matrix whose size (i.e., squared Euclidean norm) is defined by

$$\|\boldsymbol{\varepsilon}\|^2 \geq \hat{\lambda}_p, \tag{17.36}$$

which is the smallest size that the actual perturbation matrix $\mathbf{E}$ may have.

If, on the one hand, $\hat{\mathcal{S}}$ is insensitive to the perturbation $\boldsymbol{\varepsilon}$, then one is justified in believing that the estimated signal subspace $\hat{\mathcal{S}}$ is "close" to the ideal signal subspace $\mathcal{S}_{\text{ideal}}$. If, on the other hand, $\hat{\mathcal{S}}$ is sensitive to $\boldsymbol{\varepsilon}$, then there is reason to believe that these two signal subspaces are "far" from each other.

On the basis of these intuitively satisfying ideas, we now offer the following criterion for rank estimation (Liavas et al., 1999b):

> The estimate of the unknown rank of the ideal matrix $\mathbf{R}_{\text{ideal}}$ is that positive integer p for which the matrix
>
> $$\hat{\mathbf{S}} = \sum_{i=0}^{p-1} \hat{\lambda}_i \hat{\mathbf{q}}_i \hat{\mathbf{q}}_i^{\mathrm{H}} \tag{17.37}$$
>
> pertaining to the estimated signal subspace $\hat{\mathcal{S}}$ is the least sensitive, with respect to all perturbations $\boldsymbol{\varepsilon}$ of size $\|\boldsymbol{\varepsilon}\|^2 \geq \hat{\lambda}_p$, over all possible values of p.

Consider, then, a perturbation $\boldsymbol{\varepsilon}$ applied to the estimate $\hat{\mathbf{R}}$, yielding the new matrix

$$\widetilde{\mathbf{R}} = \hat{\mathbf{R}} + \boldsymbol{\varepsilon}, \tag{17.38}$$

whose spectral decomposition is

$$\widetilde{\mathbf{R}} = \sum_{i=0}^{LN-1} \widetilde{\lambda}_i \widetilde{\mathbf{q}}_i \widetilde{\mathbf{q}}_i^{\mathrm{H}}. \tag{17.39}$$

Correspondingly, we may define a "perturbed" signal subspace $\hat{\mathcal{S}}$ described by the matrix

$$\widetilde{\mathbf{S}} = \sum_{i=0}^{p-1} \widetilde{\lambda}_i \widetilde{\mathbf{q}}_i \widetilde{\mathbf{q}}_i^{\mathrm{H}}. \tag{17.40}$$

To proceed further, we make use of a distance measure between two linear subspaces of the same dimension, namely, the trigonometric sine of their *canonical angles*,

which is commonly used in numerical analysis. Specifically, we may state the following (Stewart & Sun, 1990):

> Let X and Y be two linear subspaces of the same dimension. Let the columns of matrix $\mathbf{X}_\perp$ form an orthonormal basis for the space $X_\perp$ (i.e., an orthogonal complement of the space X) and the columns of matrix $\mathbf{Y}$ form an orthonormal basis for the space Y. Then, the nonzero singular values of the matrix $\mathbf{X}_\perp^{\mathrm{H}}\mathbf{Y}$ are the sines of the nonzero canonical angles between the subspaces X and Y.

On this basis, we may now introduce the sought-after distance measure

$$\mathcal{T} = \|\sin \angle(\hat{\mathcal{S}}, \widetilde{\mathcal{S}})\|^2, \tag{17.41}$$

where $\angle(\hat{\mathcal{S}}, \widetilde{\mathcal{S}})$ is the angle subtended between the subspaces $\hat{\mathcal{S}}$ and $\widetilde{\mathcal{S}}$. It turns out that this distance measure is determined by the size of the perturbation $\boldsymbol{\varepsilon}$ and the separation between the eigenvalues associated with the perturbed signal subspace $\widetilde{\mathcal{S}}$ and the estimated noise subspace $\hat{\mathcal{N}}$ (Wedin, 1972). Building on this result, Liavas et al. (1999b) go on to derive the following upper bound on the distance measure $\mathcal{T}$:

$$\mathcal{T} \leq r(p) = \begin{cases} \dfrac{\hat{\lambda}_p}{\hat{\lambda}_{p-1} - 2\hat{\lambda}_p} & \text{if} \quad \hat{\lambda}_p \leq \dfrac{\hat{\lambda}_{p-1}}{3} \\ 1 & \text{otherwise} \end{cases}. \tag{17.42}$$

Equation (17.42) shows that the sensitivity of the estimated signal subspace $\hat{\mathcal{S}}$, with respect to a perturbation $\boldsymbol{\varepsilon}$ that satisfies Eq. (17.36), is governed essentially by the separation between the smallest eigenvalue $\hat{\lambda}_{p-1}$ of the estimated signal subspace $\hat{\mathcal{S}}$ and the largest eigenvalue $\hat{\lambda}_p$ of the estimated noise subspace $\hat{\mathcal{N}}$.

If $r(p) \ll 1$ (i.e., $\hat{\lambda}_p \gg \hat{\lambda}_{p-1}$), then the estimated signal subspace $\hat{\mathcal{S}}$ is *insensitive* to a perturbation $\boldsymbol{\varepsilon}$ of size $\hat{\lambda}_p$. Such a result would lend credence to the notion that the estimated signal subspace $\hat{\mathcal{S}}$ was close to the ideal signal subspace $\mathcal{S}_{\text{ideal}}$. If, by contrast, $r(p) \approx 1$ (i.e., $\hat{\lambda}_{p-1} \approx \hat{\lambda}_p$), then the estimated signal subspace $\hat{\mathcal{S}}$ is *sensitive* to the perturbation $\boldsymbol{\varepsilon}$, in which case it is doubtful that the two signal subspaces are close to each other.

We may thus formalize the rank-detection procedure by writing

$$\hat{p} = \arg\min_p r(p), \tag{17.43}$$

where $r(p)$ is defined in Eq. (17.42) and $\hat{p}$ is the estimate of the unknown rank of the ideal matrix $\mathbf{R}_{\text{ideal}}$.

Summary of the SOS-Based Subspace Decomposition Procedure

Table 17.1 summarizes the steps involved in applying the subspace decomposition procedure for blind channel identification. The table highlights the following distinguishing features of the procedure:

- Only SOS of the channel output are used in the computation.
- The procedure is batch (block) oriented.
- The procedure is computationally intensive.

TABLE 17.1 Summary of the Subspace Decomposition Procedure for Blind Channel Identification

1. Oversample the channel output $u(t)$ by setting

$$t = nT + \frac{lT}{L}, \qquad l = 0, 1, \ldots, L-1,$$

where T is the transmitted symbol period and L is the number of virtual channels. Hence, define

$$u_n^{(l)} = u\left(nT + \frac{lT}{L}\right),$$
$$\mathbf{u}_n^{(l)} = [u_n^{(l)}, u_{n-1}^{(l)}, \ldots, u_{n-N+1}^{(l)}]^{\mathrm{T}},$$

and

$$\boldsymbol{\mu}_n = \begin{bmatrix} \mathbf{u}_n^{(0)} \\ \mathbf{u}_n^{(1)} \\ \vdots \\ \mathbf{u}_n^{(L-1)} \end{bmatrix}.$$

2. Construct the W-by-K data matrix

$$\mathbf{A}^{\mathrm{H}} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K],$$

where $W = LN$ and K is the number of data blocks used in the blind channel identification. Typically, $W > K$, which means that the matrix $\mathbf{A}$ corresponds to an overdetermined system.

3. Compute the singular-value decomposition of the data matrix $\mathbf{A}$, viz.,

$$\mathbf{U}^{\mathrm{H}}\mathbf{A}\mathbf{V} = \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix},$$

where

$$\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_0, \sigma_1, \ldots, \sigma_{W-1})$$

and $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices whose columns are, respectively, the left and right singular vectors of the data matrix. The σ's, ordered as $\sigma_0 \geq \sigma_1 \geq \cdots \sigma_{w-1} > 0$, are the singular values of the data matrix. The kth singular value σ_k is the square root of the kth eigenvalue of the estimate of the correlation matrix

$$\hat{\mathbf{R}} = \mathbf{A}^{\mathrm{H}}\mathbf{A},$$

where we have ignored the scaling factor $1/K$. The columns of matrix $\mathbf{V}$ are the eigenvectors of $\hat{\mathbf{R}}$.

4. Use the rank detection criterion

$$r(p) = \begin{cases} \dfrac{\sigma_p^2}{\sigma_{p-1}^2 - 2\sigma_p^2} & \text{if} \quad \sigma_p^2 \leq \dfrac{\sigma_{p-1}^2}{3} \\ 1 & \text{otherwise} \end{cases}$$

to estimate the rank p of the ideal matrix $\mathbf{R}_{\text{ideal}}$:

$$\hat{p} = \arg\min_p r(p).$$

5. Given the "effective rank" $\hat{p}$ determined in step 4, divide the data space into

(a) the signal subspace $\hat{\mathcal{S}}$, for which

$$\hat{\mathbf{s}}_k = \mathbf{v}_k, \qquad k = 0, 1, \ldots, \hat{p} - 1,$$

(b) and the noise subspace $\hat{\mathcal{N}}$, for which

$$\hat{\mathbf{g}}_k = \mathbf{v}_{\hat{p}+k}, \qquad k = 0, 1, \ldots, W - \hat{p} - 1.$$

(*Continues on next page*)

TABLE 17.1 (*continued*)

Now, formulate the matrices

$$\hat{\mathbf{G}}_k^{(l)} = \begin{bmatrix} \hat{\mathbf{g}}_{k,0}^{(l)} & \hat{\mathbf{g}}_{k,1}^{(l)} & \cdots & \hat{\mathbf{g}}_{k,N-1}^{(l)} & 0 & \cdots & 0 \\ 0 & \hat{\mathbf{g}}_{k,0}^{(l)} & \cdots & \hat{\mathbf{g}}_{k,N-2}^{(l)} & \hat{\mathbf{g}}_{k,N-1}^{(l)} & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \hat{\mathbf{g}}_{k,0}^{(l)} & \hat{\mathbf{g}}_{k,1}^{(l)} & \cdots & \hat{\mathbf{g}}_{k,N-1}^{(l)} \end{bmatrix}$$

and

$$\hat{\mathcal{G}}_k = \begin{bmatrix} \hat{\mathbf{G}}_k^{(0)} \\ \hat{\mathbf{G}}_k^{(1)} \\ \vdots \\ \hat{\mathbf{G}}_k^{(L-1)} \end{bmatrix}, \qquad k = 0, 1, \ldots, W - \hat{p} - 1.$$

6. Finally, compute an estimate of the channel impulse response vector $\mathbf{h}$ by minimizing the cost function

$$\mathscr{E}(\mathbf{h}) = \mathbf{h}^{\mathrm{H}} \mathcal{Q} \mathbf{h},$$

subject to the constraint $\|\mathbf{h}\| = 1$ and where the matrix

$$\mathcal{Q} = \sum_{k=0}^{W-\hat{p}-1} \hat{\mathcal{G}}_k \hat{\mathcal{G}}_k^{\mathrm{H}}.$$

In the summary presented in Table 17.1, we have used the singular-value decomposition (SVD; see Chapter 9) to compute the eigenvalues and eigenvectors of the estimate $\hat{\mathbf{R}}$ of the correlation matrix. The SVD procedure applied to the data matrix $\mathbf{A}$ is less prone to numerical errors than the direct eigenanalysis of the time-average correlation matrix $\hat{\mathbf{R}}$.

Under step 6 of Table 17.1, note that minimizing $\mathscr{E}(\mathbf{h})$ subject to $\|\mathbf{h}\| = 1$ is equivalent to finding an *extremal eigenvector* of the matrix $\mathcal{Q}$. The remarks concerning the advantages of using the SVD of the data matrix $\mathbf{A}$ rather than the eigendecomposition of the matrix product $\mathbf{A}^{\mathrm{H}}\mathbf{A}$ apply to step 6 as well. In so doing, we can compute an extremal left singular vector of the matrix

$$[\hat{\mathbf{G}}_0, \hat{\mathbf{G}}_1, \ldots, \hat{\mathbf{G}}_{W-\hat{p}-1}]$$

rather than an extremal eigenvector of $\mathcal{Q}$.

17.4 BUSSGANG ALGORITHM FOR BLIND EQUALIZATION

Turning next to the second approach to blind deconvolution, consider the *baseband model* of a digital communications system, depicted in Fig. 17.4. The model consists of the cascade connection of a *linear communication channel* and a *blind equalizer*.

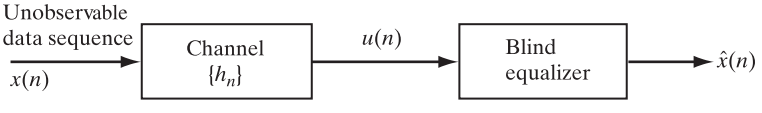


FIGURE 17.4 Noiseless cascade connection of an unknown channel and a blind equalizer.

As before, the channel includes the combined effects of a transmitting filter, a transmission medium, and a receiving filter. The channel is characterized by an *unknown* impulse response h_n that may be time varying, albeit slowly. The nature of h_n (i.e., whether it is real or complex valued) is determined by the type of modulation employed. To simplify the discussion, in this section we assume that the impulse response is real, which corresponds to the use of *multilevel pulse-amplitude modulation (M-ary PAM)*. We may thus express the sampled input–output relation of the channel by the *convolution sum*

$$u(n) = \sum_{k=-\infty}^{\infty} h_k x(n-k), \qquad n = 0, \pm 1, \pm 2, \ldots, \tag{17.44}$$

where $x(n)$ is the *data (message) sequence* applied to the channel input and $u(n)$ is the resulting *channel output*. For this introductory treatment of blind deconvolution, the effect of receiver noise in Fig. 17.4 is ignored. We are justified in ignoring the noise because degradation in the performance of data transmission (over a voice-grade telephone channel, say) is usually dominated by *intersymbol interference* due to channel dispersion. We further assume that

$$\sum_k h_k^2 = 1,$$

which implies the use of *automatic gain control* (AGC) that keeps the variance of the channel output $u(n)$ essentially constant. Also, in general, the channel is *noncausal*, which means that

$$h_n \neq 0 \qquad \text{for } n < 0.$$

The problem we wish to solve is the following:

> Given the received signal (i.e., channel output) $u(n)$, reconstruct the original data sequence $x(n)$ applied to the channel input.

Equivalently, we may restate the problem as follows:

> Design a blind equalizer that is the inverse of the unknown channel, with the channel input being unobservable and with no desired response available.

To solve the blind equalization problem, we need to prescribe a *probabilistic model* for the data sequence $x(n)$. For the problem at hand, we assume the following (Bellini, 1986, 1994):

1. The data sequence $x(n)$ is *white*; that is, the data symbols are *i.i.d. random variables* with zero mean and unit variance, as shown by

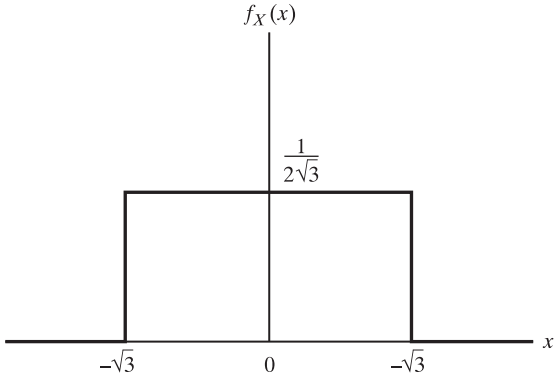
$$\mathbb{E}[x(n)] = 0 \qquad \text{for all } n \tag{17.45}$$

and

$$\mathbb{E}[x(n)x(k)] = \begin{cases} 1, & k = n \\ 0, & k \neq n \end{cases}, \tag{17.46}$$

where $\mathbb{E}$ is the statistical expectation operator.

FIGURE 17.5 Uniform distribution.



2. The *probability density function* of the data symbol $x(n)$ is *symmetric* and *uniform*; that is (see Fig. 17.5),

$$f_X(x) = \begin{cases} 1/2\sqrt{3}, & -\sqrt{3} \le x < \sqrt{3} \\ 0, & \text{otherwise} \end{cases}. \tag{17.47}$$

This distribution has the merit of being independent of the number M of amplitude levels employed in the modulation process.

Note that Eq. (17.45) and the first line of Eq. (17.46) follow from Eq. (17.47).

With the distribution of $x(n)$ assumed to be symmetric, as in Fig. 17.5, we find that the data sequence $-x(n)$ has the same law as $x(n)$. In other words, we cannot distinguish the desired inverse filter $\mathscr{L}^{-1}$ [corresponding to $x(n)$] from the opposite one, $-\mathscr{L}^{-1}$ [corresponding to $-x(n)$]. We may overcome this *sign ambiguity problem* by *initializing* the deconvolution algorithm such that there is a single nonzero tap weight with the desired algebraic sign. This can also be rendered innocuous by using *differential coding* of the source signal.

Iterative Deconvolution: The Objective

Let w_i denote the impulse response of the *ideal inverse filter*, which is related to the impulse response h_i of the channel via the formula

$$\sum_i w_i h_{l-i} = \delta_l, \tag{17.48}$$

where δ_l is the *Kronecker delta*:

$$\delta_l = \begin{cases} 1, & l = 0 \\ 0, & l \neq 0 \end{cases}. \tag{17.49}$$

An inverse filter defined in this way is "ideal" in the sense that it reconstructs the transmitted data sequence $x(n)$ *correctly*. To demonstrate this property, we first write

$$\sum_i w_i u(n - i) = \sum_i \sum_k w_i h_k x(n - i - k). \tag{17.50}$$

Making the change of index $k = l - i$ in Eq. (17.50) and interchanging the order of summation, we get

$$\sum_i w_i u(n - i) = \sum_l x(n - l) \sum_i w_i h_{l-i}. \tag{17.51}$$

Using Eq. (17.48) in Eq. (17.51) and then applying the definition of Eq. (17.49), we obtain

$$\begin{aligned} \sum_i w_i u(n - i) &= \sum_l \delta_l x(n - l) \\ &= x(n), \end{aligned} \tag{17.52}$$

which is the desired result.

For the situation described herein, the impulse response h_n is unknown. We cannot therefore use Eq. (17.48) to determine the inverse filter. Instead, we use an *iterative deconvolution procedure* to compute an *approximate inverse filter* characterized by the impulse response $\hat{w}_i(n)$, which denotes an estimate of $w(n)$. The index i refers to the *tap-weight number* in the finite-duration impulse response (FIR) *filter* realization of the approximate inverse filter, which is depicted in Fig. 17.6. The index n refers to the *adaptation cycle number*; each *adaptation cycle* corresponds to the transmission of a data symbol. The computation is performed iteratively in such a way that the convolution of the impulse response $\hat{w}(n)$ with the received signal $u(n)$ results in the complete or partial removal of the intersymbol interference (Bellini, 1986). Thus, at the nth adaptation cycle, we have the approximately *deconvolved sequence* or inverse *filter output* (two terms used interchangably)

$$y(n) = \sum_{i=-L}^{L} \hat{w}_i(n) u(n - i), \tag{17.53}$$

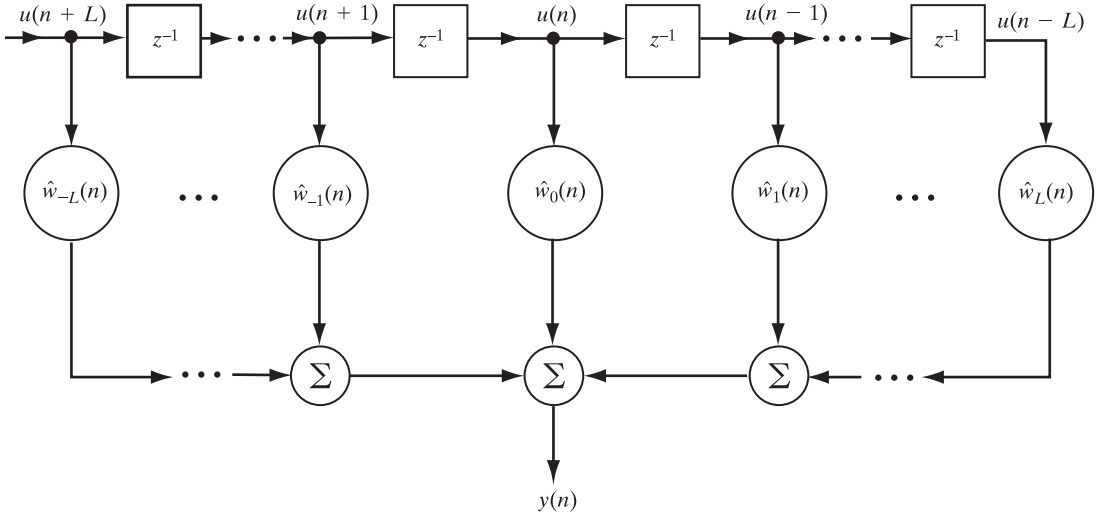


FIGURE 17.6 FIR filter realization of an approximate inverse filter; the use of real data is assumed.

where $2L + 1$ is the truncated *length* of the impulse response $\hat{w}_i(n)$. (See Fig. 17.6.) We assume that the FIR filter (equalizer) is symmetric about the midpoint $i = 0$, but this assumption is not required yet.

The convolution sum on the left-hand side of Eq. (17.52), pertaining to the ideal inverse filter, is *infinite* in extent, in that the index i ranges from $-\infty$ to ∞. In this case, we speak of a *doubly infinite filter (equalizer)*. On the other hand, the convolution sum on the right-hand side of Eq. (17.53), pertaining to the approximate inverse filter, is *finite* in extent, in that i extends from $-L$ to L. In the latter case, which is usually how it is in practice, we speak of a *finitely parameterized filter (equalizer)*. Clearly, we may rewrite Eq. (17.53) as

$$y(n) = \sum_i \hat{w}_i(n)u(n - i), \qquad \hat{w}_i(n) = 0 \text{ for } |i| > L,$$

or, equivalently,

$$y(n) = \sum_i w_i u(n - i) + \sum_i [\hat{w}_i(n) - w_i]u(n - i). \tag{17.54}$$

Let the mismatch between the actual and estimated impulse responses be defined by

$$\nu(n) = \sum_i [\hat{w}_i(n) - w_i]u(n - i), \qquad \hat{w}_i = 0 \text{ for } |i| > L. \tag{17.55}$$

Then, using the ideal result of Eq. (17.52) and the definition of Eq. (17.55), we may simplify Eq. (17.54) as follows:

$$y(n) = x(n) + \nu(n). \tag{17.56}$$

The term $\nu(n)$ is called the *convolutional noise*, which represents the *residual* intersymbol interference that results from the use of an approximate inverse filter. [As pointed out in Section 17.2, the convolutional noise ν in Eq. (17.56) should not be confused with the channel noise used in that section.]

The inverse filter output $y(n)$ is next applied to a *zero-memory nonlinear estimator*, producing the estimate $\hat{x}(n)$ for the data symbol $x(n)$. This operation is depicted in the block diagram of the blind equalizer of Fig. 17.7. We may thus write

$$\hat{x}(n) = g(y(n)), \tag{17.57}$$

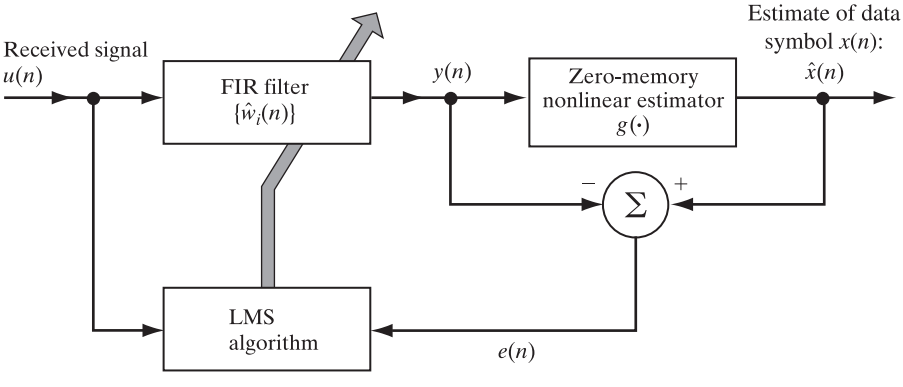


FIGURE 17.7 Block diagram of a blind equalizer.

where $g(\cdot)$ is some nonlinear function to be described. (Nonlinear estimation is discussed in the next subsection.)

Ordinarily, we find that the estimate $\hat{x}(n)$ at adaptation cycle n is not reliable enough. Nevertheless, we may use it in an *adaptive* scheme to obtain a "better" estimate at the next adaptation cycle, $n+1$. Indeed, we have a variety of *linear* adaptive filtering algorithms (discussed in previous chapters) at our disposal that we can use to perform this adaptive parameter estimation. In particular, a simple yet effective scheme is provided by the least-mean-square (LMS) algorithm discussed in Chapter 6. To apply it to the problem at hand, we note the following:

1. The ith tap input of the FIR filter at adaptation cycle (time) n is $u(n-i)$.
2. Viewing the nonlinear estimate $\hat{x}(n)$ as the desired response [since the transmitted data symbol $x(n)$ is unavailable to us], and recognizing that the corresponding FIR filter output is $y(n)$, we may express the *estimation error* for the iterative deconvolution procedure as
$$e(n) = \hat{x}(n) - y(n). \tag{17.58}$$
3. The ith tap weight $\hat{w}_i(n)$ at adaptation cycle n represents the "old" parameter estimate.

Accordingly, the updated value of the ith tap weight at adaptation cycle $n+1$ is computed as

$$\hat{w}_i(n+1) = \hat{w}_i(n) + \mu u(n-i)e(n), \qquad i = 0, \pm 1, \ldots, \pm L, \tag{17.59}$$

where μ is the *step-size parameter*. (Recall that for the situation being considered here, the data are all *real* valued.)

Equations (17.53), (17.57), (17.58), and (17.59) constitute the iterative deconvolution algorithm for the blind equalization of a real baseband channel (Bellini, 1986). As remarked earlier, each adaptation cycle of the algorithm corresponds to the transmission of a data symbol, the duration of which is assumed to be known at the receiver.

The idea of generating the estimation error $e(n)$, as detailed in Eqs. (17.57) and (17.58), is similar in philosophy to the decision-directed mode of operating an adaptive equalizer discussed in Chapter 6. (More will be said on this issue later in the section.)

Nonconvexity of the Cost Function

The ensemble-average cost function corresponding to the tap-weight update equation (17.59) is defined by

$$\begin{aligned} J(n) &= \mathbb{E}[e^2(n)] \\ &= \mathbb{E}[(\hat{x}(n) - y(n))^2] \\ &= \mathbb{E}[(g(y(n)) - y(n))^2], \end{aligned} \tag{17.60}$$

where $y(n)$ is defined by Eq. (17.53). In the LMS algorithm, the instantaneous cost function is a quadratic (convex) function of the tap weights: By contrast, the cost function $J(n)$ of Eq. (17.60) is an *ensemble average nonconvex* function of the tap weights, which means that, in general, the error-performance surface of the iterative deconvolution

procedure described here may have *local minima* in addition to *global minima*. More than one global minimum may exist, corresponding to data sequences that are equivalent under the chosen blind deconvolution criterion (e.g., ambiguity in the sign). The cost function $J(n)$ may be nonconvex because the estimate $\hat{x}(n)$, performing the role of an internally generated "desired response," is produced by passing the linear combiner output $y(n)$ through a zero-memory nonlinearity and also because $y(n)$ is itself a function of the tap weights.

It is noteworthy that practical experience indicates that most (if not all) blind deconvolution criteria are multimodal, thereby exhibiting multiple minima. This issue becomes a problem when the different minima yield different performance values. Which minimum is approached is strongly dependent on the initialization. In general, a procedure for finding an initialization that lies in the basin of attraction of the global minimum remains an open problem. However, some exceptions to the rule do occur, as explained later in Section 17.7. (The important issue of convergence is considered in greater detail later in the section.)

Statistical Properties of Convolutional Noise

The additive convolutional noise $\nu(n)$ is defined in Eq. (17.55). To develop a more refined formula for $\nu(n)$, we note that the tap input $u(n - i)$ involved in the summation on the right-hand side of that equation is given by [see Eq. (17.44)]

$$u(n - i) = \sum_k h_k x(n - i - k). \tag{17.61}$$

We may therefore rewrite Eq. (17.55) as a double summation:

$$\nu(n) = \sum_i \sum_k h_k[\hat{w}_i(n) - w_i]x(n - i - k). \tag{17.62}$$

Now, let

$$n - i - k = l.$$

Then, we may rewrite Eq. (17.62) as

$$\nu(n) = \sum_i x(l)\nabla(n - l), \tag{17.63}$$

where

$$\nabla(n) = \sum_k h_k[\hat{w}_{n-k}(n) - w_{n-k}]. \tag{17.64}$$

The sequence $\nabla(n)$ is a sequence of small numbers, representing the *residual impulse response* of the channel due to imperfect equalization. We imagine $\nabla(n)$ as a *long and oscillatory wave* that is convolved with the transmitted data sequence $x(n)$ to produce the convolutional noise sequence $\nu(n)$, as indicated in Eq. (17.63).

The definition of Eq. (17.63) is basic to the statistical characterization of the convolutional noise $\nu(n)$. The mean of $\nu(n)$ is zero, as shown by

$$\begin{aligned}\mathbb{E}[\nu(n)] &= \mathbb{E}\left[\sum_l x(l)\nabla(n-l)\right] \\ &= \sum_l \nabla(n-l)\mathbb{E}[x(l)] \\ &= 0,\end{aligned} \tag{17.65}$$

where, in the last line, we have made use of Eq. (17.45). Next, the autocorrelation function of $\nu(n)$ for a lag j is given by

$$\begin{aligned}\mathbb{E}[\nu(n)\nu(n-j)] &= \mathbb{E}\left[\sum_l x(l)\nabla(n-l)\sum_m x(m)\nabla(n-m-j)\right] \\ &= \sum_l\sum_m \nabla(n-l)\nabla(n-m-j)\mathbb{E}[x(l)x(m)] \\ &= \sum_l \nabla(n-l)\nabla(n-l-j),\end{aligned} \tag{17.66}$$

where, in the last line, this time we have made use of Eq. (17.46). Since $\nabla(n)$ is a long and oscillatory waveform, the sum on the right-hand side of Eq. (17.66) is nonzero only for $j = 0$, so that

$$\mathbb{E}[\nu(n)\nu(n-j)] = \begin{cases}\sigma^2(n), & j = 0 \\ 0, & j \neq 0\end{cases} \tag{17.67}$$

where

$$\sigma^2(n) = \sum_l \nabla^2(n-l). \tag{17.68}$$

On the basis of Eqs. (17.65) and (17.67), we may thus describe the convolutional noise $\nu(n)$ as a *zero-mean time-varying white-noise process with variance equal to* $\sigma^2(n)$, defined by Eq. (17.68).

According to the model of Eq. (17.63), the convolutional noise $\nu(n)$ is a weighted sum of i.i.d. variables representing different transmissions of data symbols. If, therefore, the residual impulse response $\nabla(n)$ is long enough, the *central-limit theorem* makes a *Gaussian* model for $\nu(n)$ to be plausible.

Having characterized the convolutional noise $\nu(n)$ by itself, all that remains for us to do is to evaluate the *cross-correlation* between $\nu(n)$ and the data sample $x(n)$. These two random variables are certainly *correlated* with each other, since $\nu(n)$ is the result of convolving the residual impulse response $\nabla(n)$ with $x(n)$, as shown in Eq. (17.63). However, the cross-correlation between $\nu(n)$ and $x(n)$ is negligible compared with the variance of $\nu(n)$. To demonstrate this, we write

$$\begin{aligned}\mathbb{E}[x(n)\nu(n-j)] &= \mathbb{E}\left[x(n)\sum_l x(l)\nabla(n-l-j)\right] \\ &= \sum_l \nabla(n-l-j)\mathbb{E}[x(n)x(l)] \\ &= \nabla(-j),\end{aligned} \tag{17.69}$$

where, in the last line, we have made use of Eq. (17.46). Here again, using the assumption that $\nabla(n)$ is a long and oscillatory waveform, we infer that the variance of $\nu(n)$ is large compared with the magnitude of the cross-correlation $\mathbb{E}[x(n)\nu(n-j)]$.

Since the data sequence $x(n)$ is white by assumption and the convolutional noise sequence $\nu(n)$ is approximately white by inference, and since these two sequences are essentially uncorrelated, it follows that their sum $y(n)$ is approximately white, too. On this basis, we may view statistically $x(n)$ and $\nu(n)$ to be essentially independent. We may thus model the convolutional noise $\nu(n)$ as an *additive, zero-mean, white Gaussian noise that is statistically independent of the data sequence* $x(n)$.

Because of the approximations made in deriving the model described herein for the convolutional noise, its use in an iterative deconvolution process yields a *suboptimal* estimator for the data sequence. In particular, given that the iterative deconvolution process is convergent, the intersymbol interference (ISI) during the latter stages of the process may be small enough for the model to be applicable. In the early stages of the iterative deconvolution process, however, the ISI is typically large, with the result that the data sequence and the convolutional noise are strongly correlated and the convolutional noise sequence is more uniform than Gaussian (Godfrey & Rocca, 1981).

Zero-Memory Nonlinear Estimation of the Original Data Sequence

We are now ready to consider the next important issue: estimating the data sequence $x(n)$, given the deconvolved sequence $y(n)$ at the FIR filter output. Specifically, we may formulate the estimation problem as follows:

> We are given a deconvolved sequence $y(n)$ that consists of the sum of two components (see Fig. 17.8):
>
> **1.** A uniformly distributed data symbol $x(n)$ with zero mean and unit variance.
> **2.** A white Gaussian noise $\nu(n)$ with zero mean and variance $\sigma^2(n)$, which is statistically independent of $x(n)$.
>
> The requirement is to derive a *Bayes estimate* of $x(n)$, optimized in a statistical sense.

Before proceeding with this classical estimation problem, two noteworthy observations are in order. First, the estimate is naturally a *conditional estimate* that depends on the optimization criterion. Second, although the estimate (in theory) is optimum in a mean-square-error sense, in the context of our present situation, it is *suboptimal*

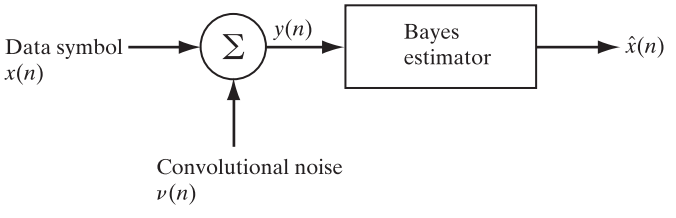


FIGURE 17.8 Estimation of the data symbol $x(n)$, given the observation $y(n)$.

by virtue of the approximations made in the development of the model described for the convolutional noise $\nu(n)$.

Putting aside this approximate form of the model, an optimization criterion of particular interest is that of minimizing the mean-square value of the error between the actual transmission $x(n)$ and the estimation $\hat{x}(n)$. The choice of this optimization criterion yields a *conditional mean estimator* that is both sensible and robust in a Bayesian sense.

For convenience of presentation, we will supress the dependence of random variables on the adaptation cycle n. Let $f_X(x|y)$ denote the *conditional probability density function* of the random variable X, given the observation sample, $Y = y$, at the deconvolver output. Correspondingly, the conditional mean estimate, denoted by $\hat{x}$, of the unknown sample $X = x$ at the deconvolver input is defined by the following conditional expectation:

$$\begin{aligned}\hat{x} &= \mathbb{E}[X|Y] \\ &= \int_{-\infty}^{\infty} x f_X(x|y)\,\mathrm{d}x.\end{aligned} \tag{17.70}$$

Now, according to *Bayes' rule* in probability theory, we have

$$f_X(x|y) = \frac{f_Y(y|x)f_X(x)}{f_Y(y)}, \tag{17.71}$$

where $f_Y(y|x)$ is the conditional probability density function of y, given x, and $f_X(x)$ and $f_Y(y)$ are the probability density functions of x and y, respectively. We may therefore rewrite Eq. (17.70) as

$$\hat{x} = \frac{1}{f_Y(y)} \int_{-\infty}^{\infty} x f_Y(y|x) f_X(x)\mathrm{d}x. \tag{17.72}$$

Let the deconvolved output y be a scaled version of the original input x, except for an additive noise term $v(n)$; that is,

$$y = c_0 x + \nu. \tag{17.73}$$

The scaling factor c_0 is slightly smaller than unity. This factor has been included in Eq. (17.73) so as to keep the mean-square value $\mathbb{E}[y^2]$ equal to unity. In accordance with the statistical model for the *convolutional noise* ν developed previously, x and ν are statistically independent. With ν modeled to have zero mean and variance σ^2, we readily see from Eq. (17.73) that the scaling factor is

$$c_0 = \sqrt{1 - \sigma^2}. \tag{17.74}$$

Furthermore, from Eq. (17.73), we find that

$$f_Y(y|x) = f_V(y - c_0 x). \tag{17.75}$$

Accordingly, the use of Eq. (17.75) in Eq. (17.72) yields

$$\hat{x} = \frac{1}{f_Y(y)} \int_{-\infty}^{\infty} x f_V(y - c_0 x) f_X(x)\,\mathrm{d}x. \tag{17.76}$$

The evaluation of $\hat{x}$ is straightforward but tedious. To proceed with it, we note the following:

1. The mathematical form of the estimate $\hat{x}(n)$ produced at the output of the Bayes (conditional mean) estimator depends on the probability density function of the original data symbol $x(n)$. For the analysis presented here, we assume that x is *uniformly distributed* with zero mean and unit variance, as depicted in Fig. 17.5; its probability density function is given in Eq. (17.47), which is reproduced here for convenience:

$$f_X(x) = \begin{cases} 1/2\sqrt{3}, & -\sqrt{3} \le x < \sqrt{3} \\ 0 & \text{otherwise} \end{cases}. \tag{17.77}$$

2. The convolutional noise ν is *Gaussian distributed* with zero mean and variance σ^2; its probability density function is therefore

$$f_V(\nu) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{\nu^2}{2\sigma^2}\right). \tag{17.78}$$

3. The filtered observation y is the sum of $c_0 x$ and ν; its probability density function is therefore equal to the convolution of the probability density function of x with that of ν:

$$f_Y(y) = \int_{-\infty}^{\infty} f_X(x) f_V(y - c_0 x)\, \mathrm{d}x. \tag{17.79}$$

Hence, using Eqs. (17.77) through (17.79) in Eq. (17.76), we get the desired result (Bellini, 1988)

$$\hat{x} = \frac{1}{c_0 y} - \frac{\sigma}{c_0} \frac{Z(y_1) - Z(y_2)}{Q(y_1) - Q(y_2)}, \tag{17.80}$$

where the variables

$$y_1 = \frac{1}{\sigma}(y + \sqrt{3}\, c_0)$$

and

$$y_2 = \frac{1}{\sigma}(y - \sqrt{3}\, c_0).$$

The function $Z(y)$ is the normalized Gaussian probability density function:

$$Z(y) = \frac{1}{\sqrt{2\pi}} e^{-y^2/2}. \tag{17.81}$$

The function $Q(y)$ is the corresponding probability distribution function:

$$Q(y) = \frac{1}{\sqrt{2\pi}} \int_y^{\infty} e^{-u^2/2}\, \mathrm{d}u. \tag{17.82}$$

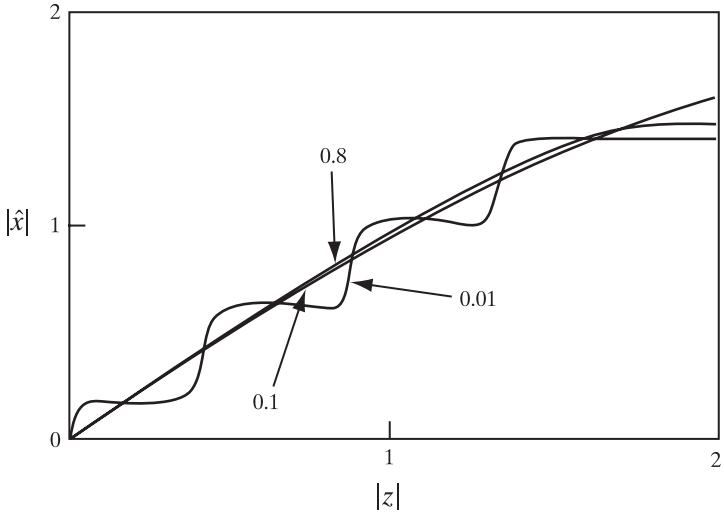


FIGURE 17.9 Nonlinear estimators of eight-level PAM data in Gaussian noise with $\hat{x} = g(z)$. The noise-to-(signal + noise) ratios are 0.01, 0.1, and 0.8. (From Bellini, 1986, with permission of the IEEE.)

A small *gain correction* to the nonlinear estimator of Eq. (17.80) is needed in order to achieve perfect equalization[3] when the iterative deconvolution algorithm [described by Eqs. (17.57) through (17.59)] converges eventually. Perfect equalization requires that $y = x$. Under the minimum mean-square-error condition, the estimation error is orthogonal to each of the tap inputs in the FIR filter realization of the approximate inverse filter. Putting all of these points together, we find that the following condition must hold (Bellini, 1986, 1988):

$$\mathbb{E}[\hat{x}g(\hat{x})] = 1. \tag{17.83}$$

In this equation, $g(\hat{x})$ is the nonlinear estimator $\hat{x} = g(y)$ with $y = \hat{x}$ for perfect equalization. (See Problem 6.)

Figure 17.9 shows the nonlinear estimate $\hat{x} = g(z)$ plotted against the $|z|$ for an eight-level PAM system (Bellini, 1986, 1988). The estimator is normalized in accordance with Eq. (17.83). In the figure, three widely different levels of convolutional noise are

[3]In general, for perfect equalization, we require that

$$y = (x - D)e^{i\phi},$$

where D is a constant delay and ϕ is a constant phase shift. This condition corresponds to an equalizer whose transfer function has unity magnitude and a linear phase response. We note that the input data sequence x_i is stationary and the channel is linear and time invariant. Hence, the observed sequence $y(n)$ at the channel output is also stationary, and its probability density function is invariant to the constant delay D. The constant phase shift ϕ is of no immediate consequence when the probability density function of the input sequence remains symmetric under rotation, which is indeed the case for the assumed density function given in Eq. (17.77). We may therefore simplify the condition for perfect equalization by requiring that $y = x$.

considered. Here, we note from Eq. (17.73) that the *distortion-to-(signal plus distortion) ratio* is given by

$$\begin{aligned}\frac{\mathbb{E}[(y - x)^2]}{\mathbb{E}[y^2]} &= (1 - c_0)^2 + \sigma^2 \\ &= 2(1 - c_0),\end{aligned} \tag{17.84}$$

where, in the last line, we have made use of Eq. (17.74). The curves presented in the figure correspond to three values of this ratio, namely, 0.01, 0.1, and 0.8. We observe the following from these curves:

1. When the convolutional noise is low, the blind equalization algorithm approaches a minimum mean-square-error criterion.
2. When the convolutional noise is high, the nonlinear estimator appears to be independent of the fine structure of the amplitude-modulated data. Indeed, different values of amplitude modulation result in only very small differences in gain, due to the normalization defined by Eq. (17.83). This suggests that the use of a uniform amplitude distribution for multilevel modulation systems is an adequate approximation.
3. The nonlinear estimator is robust with respect to variations in the variance of the convolutional noise.
4. The input–output characteristic plotted in Fig. 17.9 for a high noise-to-(signal plus noise) ratio (e.g., 0.8) is closely approximated by the hyperbolic tangent function (Haykin, 1996)

$$\hat{x} = a_1 \tanh\left(\frac{a_2 y}{2}\right), \tag{17.85}$$

where the first constant

$$a_1 = 1.945$$

and the second constant

$$a_2 = 1.25.$$

Note that the slope of this nonlinearity at the origin is directly proportional to a_2.

Convergence Considerations

For the iterative deconvolutional algorithm described by Eqs. (17.57) through (17.59) to converge in the mean value, we require the expected value of the tap weight $\hat{w}_i(n)$ to approach some constant as the number of adaptation cycles n approaches infinity. Correspondingly, we find that the *condition for convergence in the mean value* is described by

$$\mathbb{E}[u(n - i)y(n)] = \mathbb{E}[u(n - i)g(y(n))] \qquad \text{for large } n \text{ and for } i = 0, \pm 1, \ldots, \pm L.$$

Multiplying both sides of this equation by $\hat{w}_{i-k}$ and summing over i, we get

$$\mathbb{E}\left[y(n)\sum_{i=-L}^{L}\hat{w}_{i-k}(n)u(n - i)\right] = \mathbb{E}\left[g(y(n))\sum_{i=-L}^{L}\hat{w}_{i-k}(n)u(n - i)\right] \qquad \text{for large } n. \tag{17.86}$$

We next note from Eq. (17.53) that

$$\begin{aligned} y(n-k) &= \sum_{i=-L}^{L} \hat{w}_i(n-k)u(n-k-i) \\ &= \sum_{i=-L-k}^{L-k} \hat{w}_{i-k}(n)u(n-i) \qquad \text{for large } n. \end{aligned}$$

Provided that L is large enough for the FIR equalizer to achieve perfect equalization, we may approximately write

$$y(n-k) \approx \sum_{i=-L}^{L} \hat{w}_{i-k}(n)u(n-i) \qquad \text{for large } n \text{ and large } L. \tag{17.87}$$

Accordingly, we may use Eq. (17.87) to simplify Eq. (17.86) as follows:

$$\mathbb{E}[y(n)y(n-k)] \simeq \mathbb{E}[g(y(n))y(n-k)] \qquad \text{for large } n \text{ and large } L. \tag{17.88}$$

We now recognize the following property:

A stochastic process $y(n)$ is said to be a *Bussgang process* if it satisfies the condition

$$\mathbb{E}[y(n)y(n-k)] = \mathbb{E}[y(n)g(y(n-k))], \tag{17.89}$$

where the function g is a zero-memory nonlinearity.[4]

In other words, a Bussgang process has the property that its autocorrelation function is equal to the cross-correlation between the process and the output of a zero-memory nonlinearity produced by that process, with both correlations being measured for the same lag. Note that a Bussgang process satisfies Eq. (17.89) up to a multiplicative constant; in the case discussed here, the multiplicative constant is unity, by virtue of the assumption made in Eq. (17.83).

Returning to the issue at hand, we may state that the process $y(n)$ acting as the input to the zero-memory nonlinearity in Fig. 17.7 is *approximately* a Bussgang process, provided that L is large; the approximation becomes better as L is made larger. For this reason, the blind equalization algorithm described by Eqs. (17.57) through (17.59) is referred to as a *Bussgang algorithm* (Bellini, 1986, 1988).

In general, convergence of the Bussgang algorithm is not guaranteed. Indeed, the cost function of the Bussgang algorithm operating with a finite L is *nonconvex* and may therefore have false minima.

For the idealized case of a doubly infinite equalizer, however, a rough proof of convergence of the Bussgang algorithm may be sketched as follows (Bellini, 1988) [the proof relies on a theorem derived in Benveniste et al. (1980) that provides sufficient conditions for convergence][5]: Let the function $\psi(y)$ denote the dependence of the

[4]A number of stochastic processes belong to the class of Bussgang processes. Bussgang (1952) was the first to recognize that any correlated Gaussian process has the property described in Eq. (17.89). Subsequently, Barrett and Lampard (1955) extended Bussgang's result to all stochastic processes with exponentially decaying autocorrelation functions. These include an independent process, since its autocorrelation function consists of a delta function that may be viewed as an infinitely fast decaying exponential.

[5]Note that the function $\psi(y)$ to be defined in Eq. (17.90) is the negative of that defined in Benveniste et al. (1980).

estimation error in the LMS algorithm on the FIR filter output $y(n)$. According to our terminology, we have [see Eqs. (17.57) and (17.58)]

$$\psi(y) = g(y) - y. \tag{17.90}$$

The *Benveniste–Goursat–Ruget theorem* states that the convergence of the Bussgang algorithm is guaranteed if the probability distribution of the data sequence $x(n)$ is sub-Gaussian and the second derivative of $\psi(y)$ is negative on the interval $[0, \infty)$. In particular, we may state the following:

1. A probability distribution is said to be *sub-Gaussian* if the kurtosis γ_2 defined in Eq. (17.2), is negative (i.e., less than that of a Gaussian distribution). According to this equation, $\gamma_2 = 3$ for a Gaussian distribution. For example, a random variable x with the probability density function

$$f_X(x) = Ke^{-|x/\beta|^\nu}, \qquad K = \text{constant}, \tag{17.91}$$

is sub-Gaussian when $\nu > 2$. For the limiting case of $\nu = \infty$, the probability density function of Eq. (17.91) reduces to that of a uniformly distributed random variable. Also, by choosing $\beta = \sqrt{3}$, we have $\mathbb{E}[x^2] = 1$. Thus, the probabilistic model assumed in Eq. (17.47) satisfies the first part of the Benveniste–Goursat–Ruget theorem.

2. The second part of the theorem is also satisfied by the Bussgang algorithm, since

$$\frac{\partial^2 \psi}{\partial y^2} < 0 \qquad \text{for } 0 < y < \infty. \tag{17.92}$$

This is readily verified by examining the curves plotted in Fig. 17.9 or Eq. (17.85).

The Benveniste–Goursat–Ruget theorem exploited in this proof is based on the assumption of a doubly infinite equalizer. Unfortunately, this assumption breaks down in practice as we have to work with a finitely parameterized equalizer. To date, no zero-memory nonlinear function g is known that would result in global convergence of the blind equalizer in Fig. 17.7 to the inverse of the unknown channel (Verdú, 1984; Johnson, 1991). Global convergence of the Bussgang algorithm for an arbitrarily large, but finite, filter length remains an open problem. Nevertheless, there is practical evidence, supported by convergence analysis presented in Li and Ding (1995), for the conjecture that the Bussgang algorithm will converge to a desired global minimum if the FIR equalizer is long enough and initialized with a nonzero center tap (e.g., $\hat{w}_0(0) = 1$ in Fig. 17.6).

Decision-Directed Algorithm

When the Bussgang algorithm has converged, the blind equalizer should be switched smoothly to the *decision-directed mode* of operation. Then, minimum mean-square-error control of the tap weights of the FIR filter component in the equalizer is exercised, as in a traditional adaptive equalizer.

Figure 17.10 presents a block diagram of the equalizer operating in its decision-directed mode. The only difference between this mode of operation and that of blind equalization described in Figure 17.7 lies in the type of zero-memory nonlinearity

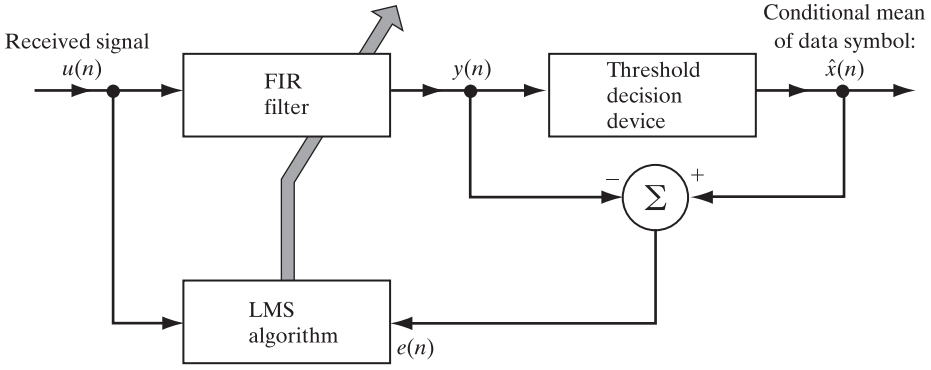


FIGURE 17.10 Block diagram of a decision-directed mode of operation.

employed. Specifically, the conditional mean estimation of the blind equalizer shown in Fig. 17.7 is replaced by a *threshold decision device*. Given the deconvolved sequence $y(n)$—that is, the equalized signal at the FIR filter output—the threshold device makes a *decision in favor of a particular value in the known alphabet of the transmitted data sequence that is closest to* $y(n)$. We may thus write

$$\hat{x}(n) = \text{dec}(y(n)). \tag{17.93}$$

For example, in the simple case of an *equiprobable binary data sequence*, the data level is

$$x(n) = \begin{cases} +1 & \text{for symbol 1} \\ -1 & \text{for symbol 0} \end{cases} \tag{17.94}$$

and the decision function is

$$\text{dec}(y(n)) = \text{sgn}(y(n)), \tag{17.95}$$

where sgn denotes the *signum function*. Thus, the estimate $\hat{x}(n)$ is said to equal +1 if $y(n)$ is positive and −1 if it is negative.

The equations that govern the operation of the decision-directed algorithm are the same as those of the Bussgang algorithm, except for the use of Eq. (17.93) in place of Eq. (17.57). Herein lies an important practical advantage of a blind equalizer that is based on the Bussgang algorithm and that incorporates the decision-directed algorithm:

> Its implementation is only slightly more complex than that of a traditional adaptive equalizer, yet it does not require the use of a training sequence.

Suppose that the following conditions are satisfied:

1. On completion of blind equalization, the eye pattern is open. (The *eye pattern* refers to the synchronized superposition of different realizations of the received signal, with each realization corresponding to the transmission of a data symbol; the pattern is so called because of its resemblance to the human eye.)
2. The step-size parameter μ used in the LMS implementation of the decision-directed algorithm is fixed (a common practice).

3. The sequence of observations at the channel output, denoted by the vector $\mathbf{u}(n)$, is *ergodic*, in the sense that

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \mathbf{u}(n)\mathbf{u}^{\mathrm{T}}(n) \to \mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{T}}(n)] \qquad \text{almost surely.} \tag{17.96}$$

Then, under these conditions, *the tap-weight vector in the decision-directed algorithm converges to the optimum (Wiener) solution in the mean-square-error sense* (Macchi & Eweda, 1984). This is a powerful result, making the decision-directed algorithm an important adjunct of the Bussgang algorithm for blind equalization in digital communications.

Annealing Process

From the discussion presented on the Bussgang algorithm thus far, we see the need for an *annealing process* whereby the blind equalizer is enabled to deal with varying levels of convolutional noise as the equalization progresses from one adaptation cycle to the next. The need for such a process can be discerned from examination of the plots shown in Fig. 17.9 for three highly different values of noise-to-(noise plus signal) ratio. First of all, however, we recognize that the hyperbolic tangent function $a_1 \tanh(a_2 y/2)$ serves as a good approximator to the zero-memory soft nonlinearity g in the blind equalizer of Fig. 17.7 when the noise level is high. The primary purpose of the annealing process is to vary the slope parameter a_2, while the scaling parameter a_1 is maintained constant. With this particular scenario in mind, we may now postulate a three-phase annealing process as described here:

1. *Initialization*. The scaling parameter a_1 and slope parameter a_2 are assigned prescribed values. As pointed out previously, for the example plots of Fig. 17.9, a good choice is $a_1 = 1.945$ and $a_2 = 1.25$.
2. *Convergence phase*. From Fig. 17.9, we observe that in moving from a noise-to-(signal plus noise) ratio of 0.8 down to 0.1, the slope parameter a_2 would need to increase by a very small amount. Yet, the blind equalizer would have to experience a large number of adaptation cycles before such a significant reduction in noise-to-(noise plus signal) ratio can be attained. Accordingly, during the *convergence phase* of the annealing process, which may occupy 1000 adaptation cycles or more, the slope parameter a_2 is increased very gradually until the blind equalizer reaches the neighborhood of a local or global minimum.
3. *Decision-directed phase*. Once the blind equalizer has converged and the eye pattern is open, the slope parameter a_2 is smoothly enlarged in such a way that in the course of a few adaptation cycles, the hyperbolic tangent is gradually changed to its limiting form: the signum function.

We may thus expand the structure of the blind equalizer of Fig. 17.7 by adding an *annealing controller*, as shown in Fig. 17.11. In effect, this new structure disposes with the need for the decision-directed mode of operation depicted in Fig. 17.10.

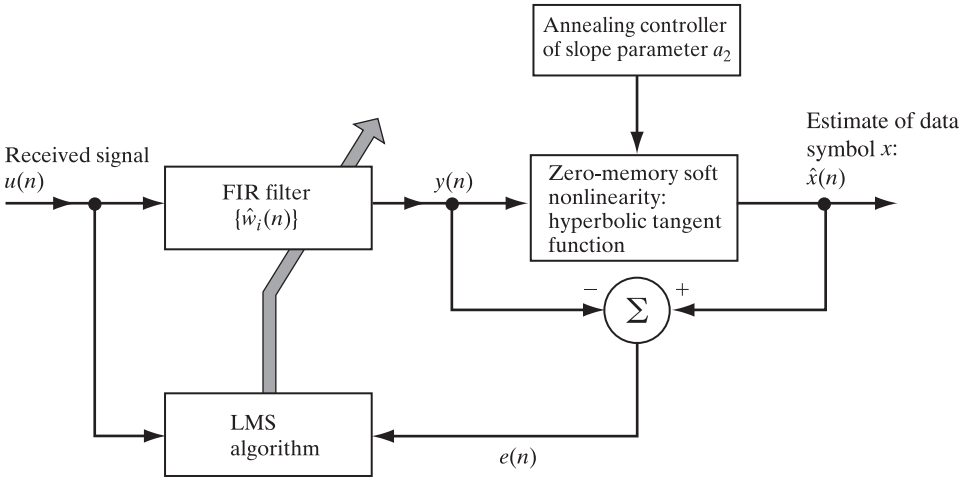


FIGURE 17.11 Block diagram of an expanded blind equalizer including an annealing controller.

17.5 EXTENSION OF THE BUSSGANG ALGORITHM TO COMPLEX BASEBAND CHANNELS

Thus far, we have discussed the use of Bussgang algorithms only for the blind equalization of M-ary PAM systems, which are characterized by a real baseband channel. In this section, we extend the use of this family of blind equalization algorithms to *quadrature-amplitude modulation (QAM)* systems that involve a hybrid combination of amplitude and phase modulations (Haykin, 2013).

In the case of a complex baseband model of the channel, the transmitted data sequence $x(n)$, the channel impulse response h_n, and the received signal $u(n)$ are all complex valued. We may thus write

$$x(n) = x_I(n) + \mathrm{j}x_Q(n), \tag{17.97}$$

$$h_n = h_{I,n} + \mathrm{j}h_{Q,n}, \tag{17.98}$$

and

$$u(n) = u_I(n) + \mathrm{j}u_Q(n), \tag{17.99}$$

where the subscripts I and Q refer to the *in-phase (real)* and *quadrature (imaginary) components*, respectively. Correspondingly, the conditional mean estimate of the complex datum $x(n)$, given the observation $y(n)$ at the FIR filter output, is written as

$$\begin{aligned}\hat{x}(n) &= \mathbb{E}[x(n)|y(n)] \\ &= \hat{x}_I(n) + \mathrm{j}\hat{x}_Q(n) \\ &= g(y_I(n)) + \mathrm{j}g(y_Q(n)),\end{aligned} \tag{17.100}$$

TABLE 17.2 Summary of Bussgang Algorithm for Blind Equalization of a Complex Baseband Channel

Initialization:

$$\hat{w}_i(0) = \begin{cases} 1, & i = 0 \\ 0, & i = \pm 1, \ldots, \pm L \end{cases}$$

Computation: $n = 1, 2, \ldots$

$$\begin{aligned} y(n) &= y_I(n) + \mathrm{j}y_Q(n) \\ &= \sum_{i=-L}^{L} \hat{w}_i^*(n)u(n-i) \\ \hat{x}(n) &= \hat{x}_I(n) + \mathrm{j}\hat{x}_Q(n) \\ &= g(y_I(n)) + \mathrm{j}g(y_Q(n)) \\ e(n) &= \hat{x}(n) - y(n) \\ \hat{w}_i(n+1) &= \hat{w}_i(n) + \mu u(n-i)e^*(n), \qquad i = 0, \pm 1, \ldots, \pm L \end{aligned}$$

where g denotes a zero-memory nonlinearity. Equation (17.100) states that the in-phase and quadrature components of the transmitted data sequence $x(n)$ may be estimated separately from the in-phase and quadrature components of the FIR filter output $y(n)$, respectively, Note, however, that the conditional mean $\mathbb{E}[x(n)|y(n)]$ can be expressed as in Eq. (17.100) only if the data transmitted in the in-phase and quadrature channels operate statistically independently of each other, which is usually the case.

Building on the complex LMS algorithm described in Chapters 5 and 6, Table 17.2 presents a summary of the Bussgang algorithm for a complex baseband channel; the table also includes a real baseband channel as a special case.

17.6 SPECIAL CASES OF THE BUSSGANG ALGORITHM

The Bussgang algorithm discussed in Sections 17.4 and 17.5 is of a general formulation, in that it includes a number of blind equalization algorithms as special cases. In this section, we consider two special cases of the Bussgang algorithm.

Sato Algorithm

The idea of blind equalization in M-ary PAM systems dates back to the pioneering work of Sato (1975b). The *Sato algorithm* consists of minimizing a *nonconvex* cost function

$$J(n) = \mathbb{E}[(\hat{x}(n) - y(n))]^2, \tag{17.101}$$

where $y(n)$ is the FIR filter output defined in Eq. (17.53) and $\hat{x}(n)$ is an estimate of the transmitted data symbol $x(n)$. This estimate is obtained by a zero-memory nonlinearity described by the formula

$$\hat{x}(n) = \alpha \operatorname{sgn}[y(n)]. \tag{17.102}$$

The constant

$$\alpha = \frac{\mathbb{E}[x^2(n)]}{\mathbb{E}[|x(n)|]} \tag{17.103}$$

sets the *gain* of the equalizer. It is apparent that the Sato algorithm is a special (nonoptimal) case of the Bussgang algorithm, with the nonlinear function $g(y)$ defined by

$$g(y) = \alpha \operatorname{sgn}(y), \tag{17.104}$$

where sgn denotes the signum function. The nonlinearity defined in Eq. (17.104) is similar to that in the decision-directed algorithm for binary PAM, except for the data-dependent gain factor α.

The Sato algorithm for blind equalization was introduced originally to deal with one-dimensional multilevel (M-ary PAM) signals, with the objective of being more robust than a decision-directed algorithm. Initially, the algorithm treats such a digital signal as a binary signal by estimating the most significant bit; the remaining bits of the signal are treated as additive noise insofar as the blind equalization process is concerned. The algorithm then uses the results of this preliminary step to modify the error signal obtained from a traditional decision-directed algorithm.

The Benveniste–Goursat–Ruget theorem for convergence holds for the Sato algorithm even though the nonlinear function ψ is not differentiable. According to this theorem, global convergence of the Sato algorithm can be achieved, provided that the probability density function of the transmitted data sequence can be approximated by a sub-Gaussian function such as the uniform distribution (Benveniste et al., 1980). However, global convergence of the Sato algorithm holds only for the limiting case of a doubly infinite equalizer.[6]

Godard Algorithm

Godard (1980) was the first to propose a family of *constant-modulus* blind equalization algorithms for use in two-dimentional digital communication systems (e.g., M-ary QAM signals).[7] The *Godard algorithm* minimizes a nonconvex cost function of the form

$$J(n) = \mathbb{E}[(|y(n)|^p - R_p)^2], \tag{17.105}$$

[6]Deviations of the Sato algorithm from the limiting condition of a doubly infinite equalizer are discussed in the following papers:

- In Mazo (1980), Verdú (1984), and Macchi and Eweda (1984), it is shown that the Sato algorithm exhibits local minima for discrete QAM input signals.
- In Ding and Li (1989), it is shown that, for finitely parameterized equalizers, the Sato algorithm may converge to local minima for sub-Gaussian inputs.

[7]The original motivation for the development of the Godard algorithm was to decouple channel equalization and carrier synchronization using M-ary QAM signals. The constant-modulus algorithm (CMA), a special case of the Godard algorithm, was so named by Treichler and Agee (1983), independently of Godard's 1980 paper, for constant envelope frequency-modulated signals. The CMA is probably the most widely investigated blind equalization algorithm and the one most widely used in practice.

The desire to reduce computational complexity of a blind equalizer has motivated the development of modified versions of CMA:

- *Signed-error version* of CMA, where the error term $e(n)$ is replaced by the signum function. Schnitter and Johnson (1999) exploit the judicious use of *dither* to modify the signed-error CMA, thereby resulting in an algorithm with robustness properties closely resembling those of the original CMA; the reduction in computational complexity, however, is attained at the cost of an increase in mean-square error. (See Problem 10 for more details.)
- *Region-based quantization* of the error term in CMA, which can be implemented using a look-up table in place of costly multipliers and adders (Endres et al., 2001).

where p is a positive integer and

$$R_p = \frac{\mathbb{E}[|x(n)|^{2p}]}{\mathbb{E}[|x(n)|^p]} \tag{17.106}$$

is a positive real constant. The Godard algorithm is designed to penalize deviations of the blind equalizer output $\hat{x}(n)$ from a constant modulus. The constant R_p is chosen in such a way that the gradient of the cost function $J(n)$ is zero when perfect equalization [i.e., $\hat{x}(n) = x(n)$] is attained.

The tap-weight vector of the equalizer is adapted in accordance with the vectorizal version of the LMS algorithm, namely

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \mathbf{u}(n) e^*(n), \tag{17.107}$$

where μ is the step-size parameter, $\mathbf{u}(n)$ is the tap-input vector, and

$$e(n) = y(n)|y(n)|^{p-2}(R_p - |y(n)|^p) \tag{17.108}$$

is the error signal (Godard, 1982). From the definition of the cost function $J(n)$ in Eq. (17.105) and from the definition of the error signal $e(n)$ in Eq. (17.108), we see that adaptation of the equalizer, according to the Godard algorithm, does not require carrier phase recovery. The algorithm therefore tends to converge slowly. However, it offers the advantage of decoupling the ISI equalization and carrier phase recovery problems from each other.

Two cases of the Godard algorithm are of specific interest:

Case 1: $p = 1$ The cost function of Eq. (17.105) for this case reduces to

$$J(n) = \mathbb{E}[(|y(n)| - R_1)^2], \tag{17.109}$$

where

$$R_1 = \frac{\mathbb{E}[|x(n)|^2]}{\mathbb{E}[|x(n)|]}. \tag{17.110}$$

Case 1 may be viewed as a modification of the Sato algorithm.

Case 2: $p = 2$ In this case, the cost function of Eq. (17.105) reduces to

$$J(n) = \mathbb{E}[(|y(n)|^2 - R_2)^2], \tag{17.111}$$

where

$$R_2 = \frac{\mathbb{E}[|x(n)|^4]}{\mathbb{E}[|x(n)|^2]}. \tag{17.112}$$

Case 2 is referred to in the literature as the *constant-modulus algorithm* (*CMA*).

The Godard algorithm is considered to be the most successful among the Bussgang family of blind equalization algorithms, as demonstrated by the comparative studies reported in Shynk et al. (1991) and Jablon (1992). In particular, we may say the following (Papadias, 1995):

- The Godard algorithm is more robust than other Bussgang algorithms with respect to the carrier phase offset. This important property of the algorithm is due to the fact that the cost function used for its derivation is based solely on the amplitude of the received signal.
- Under steady-state conditions, the Godard algorithm attains a mean-square error that is lower than that of other Bussgang algorithms.
- Last, but by no means least, the Godard algorithm is often able to equalize a dispersive channel, such that the eye pattern is opened up when it is initially closed for all practical purposes.

Summary of Special Forms of the Complex Bussgang Algorithm

The decision-directed, Sato, constant modulus, and Godard algorithms may be viewed as special cases of the complex Bussgang algorithm (Bellini, 1986). In particular, we may use Eqs. (17.93), (17.102), and (17.108) to set up the entries shown in Table 17.3 for the special forms of the zero-memory nonlinear function g pertaining to these algorithms (Hatzinakos, 1990). The entries for the decision-directed and Sato algorithms follow directly from the definition

$$\hat{x}(n) = g(y(n)).$$

In the case of the Godard algorithm, we note that

$$e(n) = \hat{x}(n) - y(n)$$

or, equivalently,

$$g(y(n)) = y(n) + e(n).$$

We may use the latter relation and Eq. (17.108) to derive the special forms of the complex Godard algorithm in Table 17.3.

TABLE 17.3 Special Cases of the Complex Bussgang Algorithm

Algorithm	Zero-memory nonlinear function g	Definitions
Decision-directed*	sgn	
Sato	α sgn	$\alpha = \dfrac{\mathbb{E}[x^2(n)]}{\mathbb{E}[\lvert x(n)\rvert]}$
Constant modulus	$y(n)(1 + R_2 - \lvert y(n)\rvert^2)$	$R_2 = \dfrac{\mathbb{E}[\lvert x(n)\rvert^4]}{\mathbb{E}[\lvert x(n)\rvert^2]}$
Godard	$\dfrac{y(n)}{\lvert y(n)\rvert}\,(\lvert y(n)\rvert + R_p\lvert y(n)\rvert^{p-1} - \lvert y(n)\rvert^{2p-1})$	$R_p = \dfrac{\mathbb{E}[\lvert x(n)\rvert^{2p}]}{\mathbb{E}[\lvert x(n)\rvert^{p}]}$

*The zero-memory nonlinear function sgn for the decision-directed algorithm applies if the input data are binary; for the general case of M-ary PAM, an M-ary slicer is required.

17.7 FRACTIONALLY SPACED BUSSGANG EQUALIZERS

In this section, we complete the study of traditional blind deconvolution algorithms by tying together the two major themes of the chapter: SOS-based algorithms and Bussgang algorithms relying on HOS. We do so by underscoring the relevance of cyclostationarity for SOS as well as Bussgang algorithms.

In light of the material presented in Section 17.2, we may say that the importance of oversampling, fractional spacing, and cyclostationarity in SOS-based deconvolution algorithms is now well established. The same channel disparity condition and filtering-matrix rank condition, which were discussed in Section 17.3, can also be exploited advantageously for the Bussgang algorithms discussed in Sections 17.4 and 17.5. This important result was brought to light in Li and Ding (1996), who showed that all minima of the Godard algorithm achieve perfect channel equalization in a SIMO channel setting (in combination with a multiple-input, single-output equalizer), under the same conditions as those in the filtering-matrix rank theorem with a minor adjustment to condition 3 of the theorem: The size N now represents the number of coefficients in each branch of the equalizer. This is in stark contrast to the situation involving a SISO channel (and, thus, a SISO equalizer), for which the multiple minima of the constant modulus criterion, in general, give disparate performance levels.

The clever proof presented in Li and Ding (1996) does not initially resemble that of the Benveniste–Goursat–Ruget theorem, although some points of tangency are worth bringing to light. This theorem assumes a doubly infinite equalizer, and on a minor technical point, assumes that the channel transfer function has no zeros on the unit circle. With $\{w_k\}$ denoting the equalizer tap weights, the *combined channel-equalizer response* is

$$\{c_k\} = \{h_k\} * \{w_k\}$$

where the asterisk denotes convolution. If $\{w_k\}$ is doubly infinite, then so is $\{c_k\}$; if the transfer function $H(z)$ of the channel has no zeros on the unit circle, then an arbitrary configuration of the combined response $\{c_k\}$ may be attained for an appropriate setting of the equalizer tap weights $\{w_k\}$. If, instead, $\{w_k\}$ is a finite-length sequence, then $W(z)$ is an FIR filter and any zeros of $H(z)$ remain in the combined transfer function; such a combined channel-equalizer response $\{c_k\}$ is clearly subject to constraints, namely, that its z-transform vanishes at the zeros of $H(z)$.

Now, using $\mathbb{E}[\psi^2(n)]$ as the cost function, where $\psi(n)$ is defined in Eq. (17.90), the stationary points in the combined channel-equalizer response space would be solutions to the system

$$\frac{\mathrm{d}\mathbb{E}[\psi^2(n)]}{\mathrm{d}c_k} = 0 \qquad \text{for all } k. \tag{17.113}$$

The minimum points of the cost function would be those stationary points for which the second derivative matrix (i.e., Hessian), whose (j, k) element, namely, $\partial^2\mathbb{E}[\psi^2(n)]/(\mathrm{d}c_j \mathrm{d}c_k)$, is positive (semi)-definite. For "correctly" chosen nonlinearities $\mathrm{g}(\cdot)$ and appropriate signal statistics, all solutions yield an ideal combined response, that is, a response having a sole nonzero term [e.g., Benveniste et al. (1980), Godard (1980)].

This analysis by itself, however, does not apply to the finite-equalizer case, since it ignores the constraint that the combined response contain the channel zeros as factors. In particular, the finite-length constraint on the equalizer may preclude any ideal combined response being attainable.

The doubly infinite equalizer case (combined with a channel having no zeros on the unit circle) provides an exceptional setting, because all configurations of the combined response are then attainable, including an ideal equalizer. The descriptions in the equalizer coefficient space and the combined channel-equalizer response space are then equivalent. Formally, we may show (Benveniste et al., 1980) for this case that any stationary point in the combined response space generates a stationary point (or possibly a stationary "manifold") in the equalizer space, and that the characterization of each stationary point as minimum or saddle point is likewise preserved. (In the case of a saddle point, the trajectories going to the *saddle point* are stable, whereas the trajectories coming from the saddle point are unstable.)

The favorable situation described herein carries over to the finite-length equalizer case, provided that we use a SIMO channel combined with a MISO equalizer. This particular combination yields a signal-flow diagram similar to that of Fig. 17.2, in which each channel output $u_n^{(l)}$ is now fed to an FIR filter with tap weights $\{w_k^{(l)}\}$, with the L filter outputs summed together. Denoting the tap-weight vector of the lth equalizer branch as

$$\mathbf{w}^{(l)} = \left[w_0^{(l)}, w_1^{(l)}, \ldots, w_{N-1}^{(l)}\right]^{\mathrm{T}}, \quad l = 0, 1, \ldots, L-1, \tag{17.114}$$

the combined channel-equalizer response involves the sum of the L branches as

$$\begin{aligned}\mathbf{c} &= \left[c_0, c_1, \ldots, c_{M+N-1}\right]^{\mathrm{T}} \\ &= \mathcal{H}^{\mathrm{T}}\mathbf{w},\end{aligned} \tag{17.115}$$

where

$$\mathcal{H} = \begin{bmatrix} \mathbf{H}^{(0)} \\ \mathbf{H}^{(1)} \\ \vdots \\ \mathbf{H}^{(L-1)} \end{bmatrix}$$

and

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}^{(0)} \\ \mathbf{w}^{(1)} \\ \vdots \\ \mathbf{w}^{(L-1)} \end{bmatrix}.$$

The new setting described in Eq. (17.115) involves the same channel convolution matrix $\mathcal{H}$ of Eq. (17.15). If the conditions of the filtering-matrix rank theorem are satisfied (where N is now the length of each equalizer branch), then $\mathcal{H}$ has full column rank $M+N$. In this case, an arbitrary configuration of the combined channel-equalizer response vector $\mathbf{c}$ (containing $M+N$ elements) can be reached for an appropriate choice of the equalizer coefficients. Calculating the minimum points in the equalizer space then reduces to

the (simpler) problem of calculating minimum points in the combined response space. If the remaining conditions of the Benveniste–Goursat–Ruget theorem are satisfied, then all minimum points yield ideal equalizers, in the absence of channel noise.[8] We thus see that cyclostationarity and the related concepts hold performance advantages for Bussgang algorithms as well, in addition to their special interest for SOS-based blind deconvolution algorithms. In particular, we may state that the *conditions for perfect blind equalizability* using fractionally spaced Bussgang algorithms are as follows (Li and Ding, 1996; Johnson et al., 2000):

1. The channel convolution matrix $\mathcal{H}$ has full column rank.
2. The channel is noise free.
3. The transmitted signal (i.e., channel input) consists of i.i.d. complex-valued symbols that are circularly symmetric.
4. The underlying probability distribution of the transmitted signal is sub-Gaussian, which means that its kurtosis is smaller than that of a Gaussian distribution.

Conditions 1 and 2 pertain to the channel-equalizer combination, while conditions 3 and 4 pertain to the transmitted signal. However, in a practical situation, the channel is always noisy. Provided that the channel noise is white and relatively small (i.e., the received SNR is high), then the minimum points of the blind equalizer will be perturbed but remain in the neighborhood of the ideal minimum points pertaining to the noiseless channel (Li and Ding, 1996).

Computer Experiment

In this computer experiment,[9] we consider a four-tap channel model and two-tap equalizer, both of which are $T/2$-spaced, where T is the symbol period. The 2-by-2 channel matrix used in the experiment is given by

$$\mathcal{H} = \begin{bmatrix} h_1 & h_3 \\ h_0 & h_2 \end{bmatrix} = \begin{bmatrix} -0.5 & 0.3 \\ 1.0 & 0.2 \end{bmatrix}.$$

[8]The convergence analysis of a stochastic gradient-descent adaptive algorithm amounts to studying the stable equilibria (i.e., local and global minima) of the algorithm. Since the channel convolutional matrix $\mathcal{H}$ may have a nontrivial null space, the equilibria of the algorithm are classified into:

- Channel-dependent equilibria (CDE).
- Algorithm-dependent equilibria (ADE).

The null space of $\mathcal{H}$ is affected by the channel diversity produced by oversampling. In Ding (1997), it is shown that the convergence behavior of fractionally spaced equalizers (FSEs) is determined by the ADE alone, so long as the channel convolutional matrix $\mathcal{H}$ has full column rank. This full rank condition is equivalent to requiring (1) that all subchannels have no common zeros and (2) that the size of $\mathcal{H}$ is large enough; these requirements are known as the *length-and-zero condition*. In particular, FSEs using such blind deconvolution algorithms as the Godard algorithm (Godard, 1980) and the Shalvi–Weinstein algorithm (Shalvi & Weinstein, 1990) are globally convergent to desired equilibria.

[9]This computer experiment is based on Johnson et al. (2000). This reference presents a highly detailed treatment of fractionally spaced Bussgang algorithms.

This matrix is invertible, which means that the even polynomial

$$H_e(z) = h_0 + h_2 z^{-1}$$

and the odd polynomial

$$H_o(z) = h_1 + h_3 z^{-1}$$

have different roots; hence, the transfer function $H(z)$ has no zeros on the unit circle.

Let the tap-weight vector of the blind equalizer be denoted by

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}.$$

Then in the general case of a noisy channel, the cost function of the fractionally spaced Bussgang equalizer is defined by (Johnson et al., 2000)

$$\begin{aligned} J(\mathbf{w}) = {} & \frac{1}{4}\sigma_s^4(\gamma_{2,s} - 3)\|\mathcal{H}^{\mathrm{T}}\mathbf{w}\|_4^4 + \frac{3}{4}\sigma_s^4\|\mathcal{H}^{\mathrm{T}}\mathbf{w}\|_2^4 \\ & + \frac{1}{4}\sigma_n^4(\gamma_{2,n} - 3)\|\mathbf{w}\|_4^4 + \frac{3}{4}\sigma_n^4\|\mathbf{w}\|_2^4 \\ & + \frac{3}{2}\sigma_s^2\sigma_n^2\|\mathcal{H}^{\mathrm{T}}\mathbf{w}\|_2^2 \cdot \|\mathbf{w}\|_2^2 - \frac{1}{2}\sigma_s^2\gamma_{2,s}(\sigma_s^2\|\mathcal{H}^{\mathrm{T}}\mathbf{w}\|^2 + \sigma_n^2\|\mathbf{w}\|_2^2) \\ & + \frac{1}{4}\sigma_s^4\gamma_{2,s}^2, \end{aligned} \tag{17.116}$$

where σ_s^2 and $\gamma_{2,s}$ are, respectively, the variance and kurtosis of the zero-mean transmitted signal and σ_n^2 and $\gamma_{2,n}$ are, respectively, the variance and kurtosis of the zero-mean additive white channel noise. The $\mathbb{L}^k$-norm of a vector $\mathbf{x}$ with components x_1 and x_2 is defined by

$$\|\mathbf{x}\|_k = (x_1^k + x_2^k)^{1/k}, \qquad k = 2, 4. \tag{17.117}$$

Figure 17.12 depicts a three-dimensional plot of the cost function $J(\mathbf{w})$ versus the tap weights w_0 and w_1. This figure is obtained by evaluating Eq. (17.116) for a transmitted signal in the form of binary phase-shift keying with i.i.d. symbols and kurtosis $\gamma_{2,s} = 1$, and assuming a noiseless channel. Figure 17.13 displays the corresponding contour plots of tap-weight w_1 versus tap-weight w_0 for varying cost function J. These two figures clearly show that all four minimum points of the cost function are indeed global minima, each of which therefore yields perfect equalization.

Figure 17.14 illustrates the effect of nonzero channel noise on the contour plots of the blind equalizer. This figure pertains to SNR of 20 dB, which is defined by

$$\text{SNR} = 10 \log_{10}\left(\frac{\sigma_s^2}{\sigma_n^2}\right) \text{dB}.$$

Comparing the contour plots of Fig. 17.13 with those of Fig. 17.14, we see that the presence of channel noise causes a perturbation in the shape of the error-performance surface, with the result that the minimum points of the cost function $J(\mathbf{w})$ are now local rather than global. In other words, in the presence of channel noise, the different minimum points of the cost function no longer result in the same level of equalizer performance.

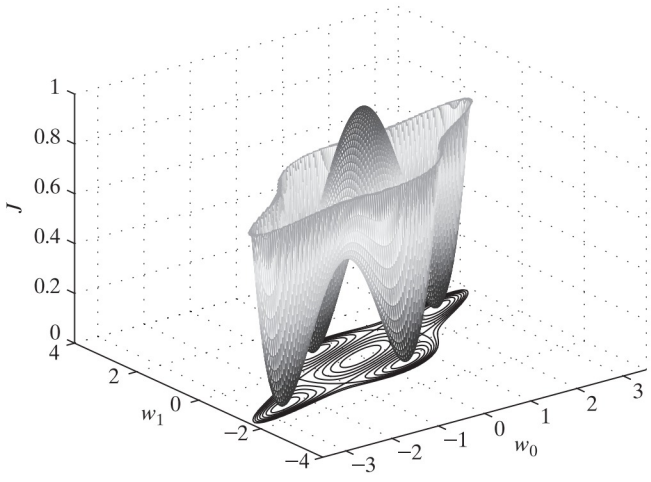


FIGURE 17.12 Three-dimensional plot of the cost function $J(w_0, w_1)$ versus the tap-weights w_0 and w_1, assuming a noise-free channel.

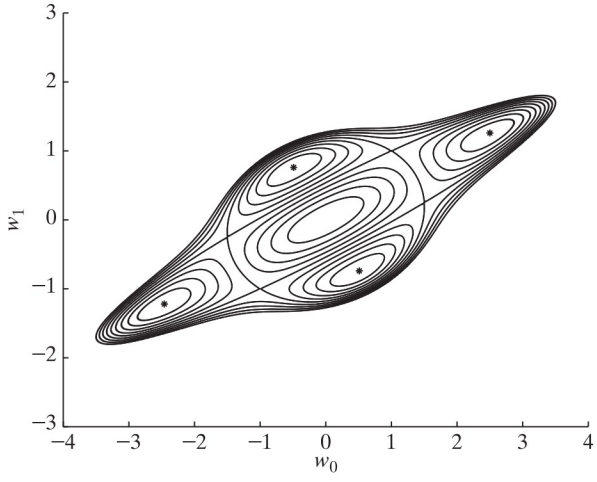


FIGURE 17.13 Contour plots for a noise-free channel, showing the tap-weight w_1 versus w_0, with each contour corresponding to a fixed value of the cost function $J(w_0, w_1)$.

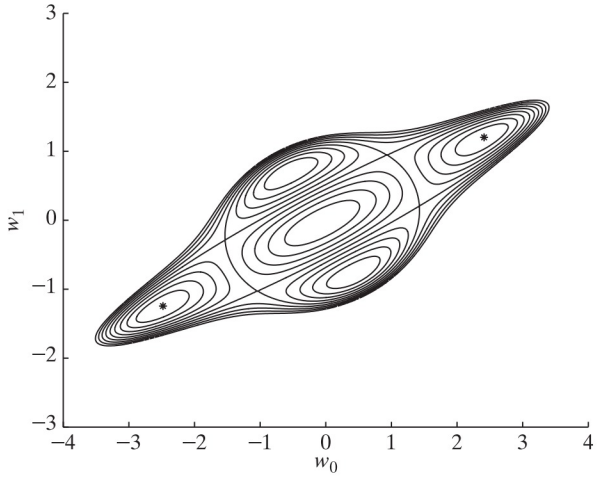


FIGURE 17.14 Contour plots for a noisy channel, showing the tap-weight w_1 versus w_0, with each contour corresponding to a fixed value of the cost function $J(w_0, w_1)$; the signal-to-noise ratio for the computations is 20 dB.

17.8 ESTIMATION OF UNKNOWN PROBABILITY DISTRIBUTION FUNCTION OF SIGNAL SOURCE

In deriving the Bussgang algorithm for blind deconvolution in Section 17.4, the underlying probabilistic distribution of the signal source was assumed to be a uniform distribution, which is the least informative of all possible probability distributions. Suppose, however, that we are given a set of cumulants expressing an incomplete but useful *probabilistic model* of the signal source. In such a scenario, we may raise the following basic question,

> Given such a model, how do we estimate the unknown probability distribution of the signal source?

Clearly, in pursuing such an approach, the blind deconvolution problem is more informative than the uniform distribution problem, and an algorithm exploiting an estimate of the unknown distribution is likely to be more statistically efficient than the Bussgang algorithm, hence the interest in the material presented in this section. In any event, our first task is to address the solution to the above-mentioned question.

Maximum Entropy Principle

Inspired by Shannon's information theory, Jaynes (1982) formulated a rationale for the broadly defined field of maximum entropy methods, which have found applications on several fronts.

To elaborate, the *maximum entropy principle*, hereafter referred to simply as the *MaxEnt principle*, is summarized as follows:

> When an inference is made about a probabilistic problem of interest on the basis of incomplete information about the problem, the inference should be drawn from the probability distribution that maximizes the entropy, subject to constraints imposed on the distribution for practical reasons.

The MaxEnt problem is therefore a constrained optimization problem, the solution of which naturally involves the *method of Lagrange multipliers*, discussed in Appendix C.

Before proceeding further, however, we need to define what we mean by *entropy*. Consider a continuous random variable X, whose sample value is denoted by x. According to Shannon's information theory,[10] the entropy of the random variable X is defined by the following formula:

$$H(X) = -\int_{-\infty}^{\infty} f_X(x) \log f_X(x) \mathrm{d}x \tag{17.118}$$

[10]In the classic paper "A Mathematical Theory of Communication," Shannon (1948) laid down the foundations of information theory as we know it today. The concept of entropy defined in Eq. (17.118) is inspired by the concept of entropy as understood in thermodynamics. It occupies an important place in information theory.

where $f_X(x)$ is the probability density function (pdf) of X. Note that $H(X)$ is not a function of X; rather, the argument X in $H(X)$ is simply the random variable for which the entropy is defined. It is on the basis of Eq. (17.118) that the MaxEnt principle is defined.

To proceed with the problem at hand, let $m_1, m_2, \ldots, m_k$ denote prior knowledge about the statistics of the random variable X. The ith moment of X is formally defined by the expectation

$$\begin{aligned} m_i &= \mathbb{E}[X^i] \\ &= \int_{-\infty}^{\infty} x^i f_X(x)\mathrm{d}x, \quad i = 1, 2, \ldots, K. \end{aligned} \tag{17.119}$$

With this prior knowledge, the stage is now set for invoking the MaxEnt principle to find an *estimate* of the pdf, $f_X(x)$, which is denoted by $\hat{f}_X(x)$. Specifically, using the method of Lagrange multipliers with the entropy of the random variable X defined in Eq. (17.118) as the constraint, we may follow the procedure described in Jumarie (1990) to express the desired estimate, $\hat{f}_X(x)$, as follows:

$$\hat{f}_X(x) = \exp\left(\sum_{i=1}^{K} \lambda_i x^i\right), \tag{17.120}$$

where λ_i is the ith Lagrange multiplier. Moreover, according to Jumarie, the formula for finding the Lagrange multipliers is defined by the summation

$$\sum_{k=1}^{K} k\lambda_k m_{k+i} = -m_i, \quad i = 1, 2, \ldots, K. \tag{17.121}$$

However, it should be noted that according to Eq. (17.121), we need $2K$ moments of the random variable X in order to calculate the desired K Lagrange multipliers.[11] However, statistical errors made in adopting the approximate pdf of Eq. (17.120) are in perfect accord with the MaxEnt principle.

Estimation of the Conditional Expectation

With the estimate $\hat{f}_X(x)$, defined in Eq. (17.120), we may now go on to consider the conditional mean expectation, $\mathbb{E}[X|Y]$, which plays a key role in addressing the zero-mean Bayesian estimation, discussed in Section 17.4. To this end, we use the definition of marginal distribution in probability theory to express the pdf of the blind equalizer's output in Figure 17.7; specifically, we write

$$f_Y(y) = \int_{-\infty}^{\infty} f_Y(y|x) f_X(x)\mathrm{d}x. \tag{17.122}$$

Then, substituting this equation into Eq. (17.71), we may redefine the conditional mean expectation as the ratio of two definite integrals, as shown by

[11] In an effort to simplify calculation of the Lagrange multipliers, Pinchas and Bobrovsky (2006) derived a new formula that invokes another assumption, over and above the assumptions made in deriving the Bussgang algorithm in Section 17.4—namely, the unknown pdf, $f_X(x)$, is assumed to be *even symmetric*.

$$\mathbb{E}[X|Y] = \frac{\displaystyle\int_{-\infty}^{\infty} x f_Y(y|x) f_X(x)\,\mathrm{d}x}{\displaystyle\int_{-\infty}^{\infty} f_Y(y|x) f_X(x)\,\mathrm{d}x}. \tag{17.123}$$

Note that the two integrals in Eq. (17.123) are similar, except for the presence of the sample value x in the integral of the numerator.

In Section 17.4, we assumed that the convolutional noise ν is Gaussian distributed with zero mean and variance σ^2; moreover, ν was defined in Eq. (17.73) to be

$$\nu = y - c_0 x,$$

where the scaling factor c_0 applied to the input x was chosen to be slightly smaller than unity.

On the other hand, in a new approach formulated by Pinchas and Bobrovsky (2006) to solve the blind deconvolution problem using the MaxEnt principle, c_0 is taken to be unity. In such a case, the Gaussian distribution of the convolutional noise ν in Eq. (17.78) is reformulated as follows:

$$f_Y(y|x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y-x)^2}{2\sigma^2}\right), \tag{17.124}$$

where σ^2 is the variance of the convolutional noise ν. Thus, substituting the approximate formula of Eq. (17.120) and the formula of Eq. (17.124) into Eq. (17.123), we obtain the desired approximate formula for the conditional mean expectation, $\mathbb{E}[X|Y]$.

In order to make this approximation analytically manageable, Pinchas and Bobrovsky introduced a new set of definitions under the assumption that the unknown pdf, $f_X(x)$, is even symmetric, as follows:

$$\mathbb{E}[X|Y] = \frac{\displaystyle\int_{-\infty}^{\infty} \alpha_1(x) \exp(-\psi(x)/\rho)\,\mathrm{d}x}{\displaystyle\int_{-\infty}^{\infty} \alpha_0(x) \exp(-\psi(x)/\rho)\,\mathrm{d}x}, \tag{17.125}$$

where, in accordance with Eq. (17.123) and the use of Eq. (17.121), if we set

$$\alpha_0(x) = \exp\left(\sum_{k=1}^{K} \lambda_k x^k\right), \tag{17.126}$$

then

$$\alpha_1(x) = x\alpha_0(x), \tag{17.127}$$

$$\psi(x) = (y-x)^2, \tag{17.128}$$

and

$$\rho = 2\sigma^2. \tag{17.129}$$

There is no approximation made in Eq. (17.125), provided that summation order K is infinitely large.

Hereafter, the functions $\alpha_0(x)$, $\alpha_1(x)$, and $\psi(x)$ are all assumed to be real continuous functions of the variable x. Under this assumption, we may view the definite

integrals in the numerator and denominator of Eq. (17.125) to be *Laplace integrals*, with the exponential function $\exp(-\psi(x)/\rho)$ playing the role of a *kernel*.

Laplace's Method for Local Analysis

Analytic evaluation of the Laplace integrals in Eq. (17.125) is a difficult proposition. To get around this difficulty, Pinchas and Bobrovsky (2006) settled for an approximate local analysis of the integrals. Using Laplace's method,[12] they derived series expansions of the two definite integrals in the numerator and denominator of Eq. (17.125). With local approximations in mind around the point $-\infty < x_0 < \infty$, the approximations involve the new terms $\alpha_j(x_0)$, $\alpha_j^{(2)}(x_0)/\psi(x_0)$, and $\alpha_j^{(4)}(x_0)/\psi(x_0)$, where, in terms of $\alpha_j(x_0)$, we have the following even-order partial derivatives:

$$\alpha_j^{(2)}(x_0) = \frac{\partial^2}{\partial x^2}\alpha_j(x)\bigg|_{x=x_0} \tag{17.130}$$

and

$$\alpha_j^{(4)}(x_0) = \frac{\partial^4}{\partial x^4}\alpha_j(x)\bigg|_{x=x_0}. \tag{17.131}$$

For $j = 0$, the partial derivative in Eq. (17.130) corresponds to the definite integral in the denominator of Eq. (17.125), and for $j = 1$, the partial derivative of Eq. (17.131) corresponds to the definite integral in the numerator of Eq. (17.125). Note also that the even orders of the partial derivatives, defined in Eqs. (17.130) and (17.131), result from the even symmetric assumption of the pdf, $f_X(x)$. According to Pinchas and Bobrovsky (2006), the numerical error in using Laplace's method for $K = 4$ is on the order of ρ^3; hence, for small ρ (i.e., large SNR), the error is small enough to be ignored for all practical purposes.

Blind Channel Equalization

Let $\hat{x}$ denote the *conditional-mean estimate* of the random value X, given Y. Then, in light of Eq. (17.70), the approximation procedure, just described, may be summed up as follows:

$$\mathbb{E}[X|Y] \approx \hat{x}.$$

With blind channel equalization as the task of interest, given the observation $Y = y$ as the decoded sequence in Fig. 17.7 and the corresponding estimate $X = \hat{x}$, Pinchas and Bobrovsky (2006) formulated a mathematically demanding algorithm for

[12]Consider the integral

$$I(t) = \int_{-\infty}^{\infty} \alpha(x) \exp(-t\psi(x))\mathrm{d}x,$$

where $\alpha(x)$ and $\psi(x)$ are both real continuous functions of x. *Laplace's method* rests on the following idea (Bender & Orszag, 1999):

> If the function $\psi(x)$ attains its minimum value at the point $x = x_0$, where $-\infty < x_0 < \infty$, and if $\alpha(x_0) \neq 0$, then only the immediate neighborhood of $x = x_0$ contributes to the full asymptotic series expansion of $I(t)$ for large t.

Note that t in this footnote corresponds to $1/\rho$ in Eq. (17.125).

blind channel equalization. For details, reference should be made to their 2006 paper, in which extensive Monte Carlo simulations are presented for five different channels, operating at different levels of intersymbol interference and for 16-QAM and 64-QAM, where 64 stands for the number of symbols used in the simulation. In the simulations, Pinchas and Bobrovsky compared their new blind channel equalization algorithm against several other corresponding algorithms, including the Godard algorithm (described previously in the chapter). The simulation results of the Pinchas–Bobrovsky algorithm consistently outperformed the old ones in terms of *statistical efficiency*, meaning that the steady-state mean-square error was smaller and the rate of convergence was faster.

From the perspective of blind equalization for practical applications, the rate of convergence of the Pinchas–Bobrovsky algorithm (based on the MaxEnt principle) appears to have an advantage over traditional Bussgang algorithms insofar as statistical efficiency is concerned. However, formulation of this new algorithm is mathematically demanding, which, in turn, could mean that the algorithm is more sensitive to finite-precision effects compared to the Godard algorithm, for example. This comparative evaluation is apparently missing from the paper (Pinchas and Bobrovsky, 2006).

17.9 SUMMARY AND DISCUSSION

Blind deconvolution is an example of *unsupervised learning*, in the sense that it identifies the inverse of an unknown linear time-invariant (possibly non-minimum-phase) system *without* having access to a training sequence (i.e., a desired response). This operation requires the identification of both the magnitude and phase of the system's transfer function. To identify the magnitude component, we only need second-order statistics (SOS) of the received signal (i.e., the system output). However, identifying the phase component is more difficult.

One class of deconvolution procedures, discussed in this chapter, relies on SOS and makes up for the lack of a desired response by exploiting cyclostationarity, which characterizes the output of a communication channel involving the use of modulation. In this chapter, we have shown that it is algorithmically feasible to identify an unknown linear channel solely on the basis of cyclostationary statistics of the received signal, as exemplified by the subspace decomposition procedure described in Section 17.3. This is indeed a significant algorithmic achievement in the blind equalization of a communication channel in signal-processing terms.

The SOS-based subspace decomposition procedure for blind channel identification relies on block processing for its implementation. It therefore offers rapid acquisition, which makes it particularly attractive for use in a highly nonstationary environment (e.g., wireless communications). However, this important advantage is gained at the expense of a highly intensive computational complexity, which, in light of the ever-expanding computer technology, may not be a serious practical issue in the long run.

The second class of blind deconvolution procedures discussed in this chapter, namely, Bussgang algorithms, exemplified by the Godard algorithm, relies on the use of higher-order statistics (HOS) in an implicit sense to make up for the absence of a desired response. This form of blind deconvolution mandates the use of nonlinear processing. Most importantly, the received signal must be non-Gaussian for the procedure to work.

The Bussgang algorithm performs blind equalization of a linear communication channel by subjecting the received signal to an iterative deconvolution process. When the algorithm has converged in the mean value, the deconvolved sequence assumes Bussgang statistics—hence the broadly defined name of the algorithm. The distinguishing features of the Bussgang algorithm are as follows:

- The minimization of a nonconvex cost function and, therefore, the likelihood of being trapped in a local minimum.
- A low computational complexity—slightly greater than that of a traditional adaptive equalizer (based on the LMS principle algorithm) having access to a training sequence.
- A relatively slow rate of convergence.

In a sense, the SOS-based subspace decomposition procedure and the implicit HOS-based family of Bussgang algorithms are complementary: Moreover, these two themes are tied together in the design of fractionally spaced Bussgang equalizers.

Another implicit HOS-based blind convolution algorithm discussed in the chapter is the new algorithm formulated by Pinchas and Bobrovsky (2006), using the MaxEnt principle. In mathematical terms, this new algorithm is more demanding than the Godard algorithm. However, computer simulations, reported in their 2006 paper, show that this new algorithm outperforms the Godard algorithm in terms of statistical efficiency.

We conclude this discussion on an important issue that deserves serious considerations: Specifically, in Chapter 11, we stressed the need for paying attention to robustness of an adaptive filtering algorithm with respect to unknown environmental disturbances (uncertainties). In a sense, robustness also includes the issue of sensitivity of the algorithm to finite-precision effects, discussed in Chapter 12. Regrettably, however, robustness is yet to feature in the signal-processing literature on blind deconvolution on a scale comparable to that on traditional LMS and RLS algorithms.

PROBLEMS

1. In this problem, we explore the possibility of extracting the phase response of an unknown channel using cyclostationary statistics.

(a) Using Eq. (17.3) for the received signal $u(t)$ at the output of a linear communication channel with impulse response h_n, and invoking the assumptions made in Section 17.3 on the transmitted signal x_k and channel noise $\nu(t)$, show that the autocorrelation function of $u(t)$ evaluated at the two time instants t_1 and t_2 is given by

$$\begin{aligned} r_u(t_1, t_2) &= \mathbb{E}[u(t_1)u^*(t_2)] \\ &= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} r_x(kT - lT)h(t_1 - kT)h^*(t_2 - lT) + \sigma_\nu^2 \delta(t_1 - t_2), \end{aligned}$$

where $r_x(kT)$ is the autocorrelation function of the transmitted signal for lag kT and σ_ν^2 is the noise variance. Demonstrate that $u(t)$ is cyclostationary in the wide sense.

(b) The cyclic autocorrelation function and spectral density of a cyclostationary process $u(t)$ are respectively defined by the continuous-time versions of Eqs. (1.163) and (1.167), viz.,

$$r_u^\alpha(\tau) = \frac{1}{T} \int_{-T/2}^{T/2} r_u\left(t + \frac{\tau}{2}, t - \frac{\tau}{2}\right) \exp(\mathrm{j}2\pi\alpha t)\mathrm{d}t$$

and

$$S_u^{\alpha}(\omega) = \int_{-\infty}^{\infty} r_u^{\alpha}(\tau) \exp(-\mathrm{j}2\pi f\tau)\mathrm{d}t, \qquad \omega = 2\pi f,$$

where

$$\alpha = \frac{k}{T}, \qquad k = 0, \pm 1, \pm 2, \ldots.$$

Let $\Psi_k(\omega)$ denote the phase response of $S_u^{k/T}(\omega)$ and $\Phi(\omega)$ denote the phase response of the channel. Show that

$$\Psi_k(\omega) = \Phi\left(\omega + \frac{k\pi}{T}\right) - \Phi\left(\omega - \frac{k\pi}{T}\right), \qquad k = 0, \pm 1, \pm 2, \ldots.$$

(c) Let $\psi_k(T)$ and $\phi(\tau)$ denote the inverse Fourier transforms of $\Psi_k(\omega)$ and $\Phi(\omega)$, respectively. Using the result of part (b), show that

$$\psi_k(\tau) = -2\mathrm{j}\phi(\tau)\sin\left(\frac{\pi k\tau}{T}\right), \qquad k = 0, \pm 1, \pm 2, \ldots.$$

What conclusions can you draw from this relation with regard to the possibility of extracting the phase response $\Phi(\omega)$ from $\psi_k(\tau)$?

2. Suppose that the channel convolutional matrix $\mathcal{H}$ of the SIMO model defined in Eq. (17.15) has been estimated by means of the subspace decomposition procedure described in Section 17.3.

Show that, in the noise-free case, perfect equalization of the channel is achieved by using a multichannel structure whose own filtering matrix is defined by the pseudoinverse of $\mathcal{H}$.

3. Using the definitions of channel convolutional matrix $\mathcal{H}$ in Eq. (17.15) and the eigenvectors $\mathbf{g}_k$ associated with the noise space $\mathcal{N}$, derive Eq. (17.27).

4. The use of linear prediction provides the basis for other procedures for blind identification. The basic idea behind these procedures resides in the *generalized Bezout identity* (Kailath, 1980). Define the L-by-1 polynomial vector

$$\mathbf{H}(z) = [H^{(0)}(z), H^{(1)}(z), \ldots, H^{(L-1)}(z)]^{\mathrm{T}},$$

where $H^{(l)}(z)$ is the transfer function of the lth virtual channel. Under the condition that $\mathbf{H}(z)$ is irreducible, the generalized Bezout identity states that there exists a 1-by-L polynomial vector

$$\mathbf{G}(z) = [G^{(0)}(z), G^{(1)}(z), \ldots, G^{(L-1)}(z)]^{\mathrm{T}}$$

such that

$$\mathbf{G}(z)\mathbf{H}(z) = 1;$$

that is,

$$\sum_{l=0}^{L-1} G^{(l)}(z)H^{(l)}(z) = 1.$$

The implication of this identity is that a set of moving-average processes described in terms of a white-noise process $\nu(n)$ by the operation $\mathbf{y}(n) = \mathbf{H}(z)[\nu(n)]$ may also be represented by an autoregressive process of finite order.

Consider the ideal case of a noiseless channel for which the received signal of the lth virtual channel is defined by

$$u_n^{(l)} = \sum_{m=0}^{M} h_m^{(l)} x_{n-m}, \qquad l = 0, 1, \ldots, L-1,$$

where x_n is the transmitted symbol and $h_n^{(l)}$ is the impulse response of the lth virtual channel. Using the generalized Bezout identity, show that

$$\sum_{l=0}^{L-1} G^{(l)}(z)[u_n^{(l)}] = x_n$$

and, thus, that x_n is reproduced exactly. [Note that $G^{(l)}(z)$ acts as an operator.] How would you interpret this result in light of linear prediction?

5. Refer to the block diagram of Fig. 17.7, using a zero-memory nonlinear function, g. Equation (17.80) defines the conditional mean estimate of the transmitted symbol x, assuming that the convolutional noise ν is additive, white, Gaussian, and statistically independent of x. Derive this formula.

6. For perfect equalization, we require that the equalizer output $y(n)$ be exactly equal to the transmitted symbol $x(n)$. Show that when the Bussgang algorithm has converged in the mean value and perfect equalization has been attained, the nonlinear estimator must satisfy the condition

$$\mathbb{E}[\hat{x}g(\hat{x})] = 1,$$

where $\hat{x}$ is the conditional mean estimate of x.

7. Equation (17.59) provides an adaptive method for finding the tap weights of the FIR filter in the Bussgang algorithm for performing iterative deconvolution. Develop an alternative method for doing this computation, assuming the availability of an overdetermined system of equations and the use of the method of least squares, which was discussed in Chapter 9.

8. A data stream consisting of i.i.d. symbols is applied to a binary phase-shift keying (PSK) system. The resulting modulated signal, $x(n)$, is applied to a linear channel of unknown impulse response. For blind equalization of the channel, the constant-modulus algorithm (i.e., the Godard algorithm with $p = 2$) is used.

(a) Plot the error signal $e(n)$ versus the decoded sequence $y(n)$.

(b) Assuming the use of the signed-error version of the CMA, plot the piece-wise approximation of $e(n)$.

(c) Formulate the CMA and its signed-error version.

You may refer to footnote 7 for a brief description of the signed-error CMA.

9. Repeat Problem 8 for the case of a transmitter using quadrature phase-shift keying (QPSK).

10. In this problem, we build on the material presented in Problem 8 for real-valued data by exploiting the *dithered signed-error* version of the CMA, which is hereafter referred to as DSE-CMA. (See footnote 7.) This new algorithm is described by the updated equation

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{u}(n)[\alpha \operatorname{sgn}(\nu(n))],$$

where

$$\nu(n) = e(n) + \alpha\varepsilon(n),$$
$$e(n) = y(n)(R_2 - y^2(n)),$$

and μ, α, and R_2 are all positive constants and $\varepsilon(n)$ is the added dither. The samples of the dither are i.i.d. over the interval $[-1, 1]$.

According to quantization theory, the operator $\alpha\ \mathrm{sgn}(\nu(n))$ has an effect equivalent to that of the two-level quantizer

$$Q(\nu(n)) = \begin{cases} \Delta/2 & \text{for} \quad \nu(n) \geq 0 \\ -\Delta/2 & \text{for} \quad \nu(n) < 0 \end{cases},$$

where $\Delta = 2\alpha$ and $\alpha \geq |e(n)|$ for the relevant parts of the decoded sequence $y(n)$.

(a) Show that the conditional expectation of $\nu(n)$, given $y(n)$, is equivalent to a hard-limited version of the error signal $e(n)$ in the traditional CMA:

$$\mathbb{E}[\nu(n)\,|y(n)] = \begin{cases} \alpha & \text{for} \quad \nu(n) > 0 \\ e(n) & \text{for} \quad |\nu(n)| \leq \alpha. \\ -\alpha & \text{for} \quad \nu(n) < -\alpha \end{cases}$$

(b) Plot the result obtained in part (a) for $\alpha = 2$.

11. The transmitter of a digital communication system uses QPSK, which is driven by a data stream consisting of i.i.d. symbols. The modulated signal is applied to a linear channel of unknown impulse response.

For blind equalization of the channel, it is proposed to use the *Shalvi–Weinstein equalizer,* whose design is based on the cost function

$$J = \mathbb{E}[|y(n)|^4] \quad \text{subject to } \mathbb{E}[|y(n)|^2] = \sigma_x^2,$$

where $y(n)$ is the equalizer output and σ_x^2 is the variance of the original data sequence.

(a) Show that the Shalvi–Weinstein criterion is essentially the same as the Godard criterion for $p = 2$.

(b) To be more precise, show that, in the absence of channel noise, the minima of the Shalvi–Weinstein cost function are the same as the minima of the Godard cost function. (*Hint*: Rewrite the tap-weight vector in polar form—that is, a unit-norm vector times a radial scale factor—and then optimize the Godard cost function with respect to the radial scale factor.)

12. Derive the cost function $J(\mathbf{w})$ defined in Eq. (17.116) that applies to a fractionally spaced Bussgang equalizer for a linear communication channel.

Epilogue

In this final chapter of the book, we address two ways of thinking about adaptive filtering:

1. Section 1 *looks back* to the material covered in previous chapters of the book. Specifically, this section revisits three pervasive issues that play key roles in linear adaptive-filtering applications: robustness, efficiency, and complexity.
2. Section 2 *looks forward* to an emerging new topic in nonlinear adaptive filtering. Although this topic has been discussed at some length in the literature, this section presents an exposé of a new class of *kernel-based adaptive filtering*, which builds on ideas in machine learning as well as linear adaptive filtering.

With the material covered in Chapter 2 being new to the book, it is longer than Section 1.

1. ROBUSTNESS, EFFICIENCY, AND COMPLEXITY

These three issues featured in all the chapters of the book; one, two, or all three of them were discussed in one chapter or another. However, it was in Chapters 11 and 13 where all these issues were considered side by side.

Robustness–Efficiency Trade-Off

From a practical point of view, it is highly desirable for an adaptive filtering algorithm to be *robust* in the presence of *unknown disturbances*. Another desirable property—namely, *statistical efficiency*—also deserves attention of its own because of its practical importance. The key questions to be recalled here are:

1. How do we measure robustness and efficiency individually?
2. Just as importantly, how do we decide to opt for one over the other?

Considering the issue of robustness first, suppose we have a linear adaptive filtering algorithm, viewed as an *estimator*, that is prone to the presence of unknown disturbances at its input. The estimator *maps* the disturbances at the input to *estimation errors* at the output. On this basis, we may introduce the notion of *energy gain*, defined as follows:

> The energy gain of an estimator is the ratio of the error energy at the output to the total disturbance energy at the input.

Since this ratio depends on disturbances that are unknown, we remove this dependence by picking the *largest energy gain measured over all conceivable disturbances,* which are likely to arise in practice. In so doing, we come up with the H^∞ *norm* of the estimator, paving the way to describe the *optimal* H^∞ *estimation problem* as follows:

> Find that particular causal estimator whose H^∞ norm has the smallest possible value under all conceivable disturbances.

The optimal H^∞ estimator that solves this problem is of a *minimax nature* in the following sense:

> Nature, acting as the opponent, has access to all conceivable unknown disturbances that can arise, thereby increasing the energy gain. On the other hand, the estimation-algorithmic designer has the privilege of being able to select that particular algorithm that minimizes the energy gain.

In the context of linear adaptive filtering, there are two basic algorithms to choose from, the least-mean-square (LMS) algorithm and the recursive least-squares (RLS) algorithm. The findings reported on these two algorithms in Chapter 11 on robustness are summarized as follows:

1. The LMS algorithm is H^∞ *optimal*, in that no other adaptive filtering algorithm achieves a maximum energy gain less than unity. In mathematical terms, we may therefore write

$$\gamma_{\text{LMS}}^2 \leq 1 \quad \text{for} \quad 0 < \mu < \min_{n=1,\ldots,N} \frac{1}{\|\mathbf{u}(n)\|^2} \quad \text{and any integer } N, \tag{1}$$

   where γ^2 denotes the maximum energy gain, μ is the step-size parameter, and $\mathbf{u}(n)$ is the input vector.
2. On the other hand, the maximum energy of the RLS algorithm is lower- and upper-bounded as follows:

$$(\sqrt{\bar{r}} - 1)^2 \leq \gamma_{\text{RLS}}^2 \leq (\sqrt{\bar{r}} + 1)^2, \tag{2}$$

   where $\bar{r}$ is a dimensionless quantity that is greater than one.

From this pair of equations, we readily see that when robustness is the issue of interest, the LMS algorithm outperforms the RLS algorithm. So, if robustness is the requirement for a linear adaptive filtering application of interest, the LMS algorithm is the method of choice.

Turning next to the issue of statistical efficiency, there are two possible measures for quantifying efficiency:

1. *Rate of convergence*, which is defined as the number of adaptation cycles needed for an adaptive filtering algorithm to effectively reach a "steady state."
2. *Misadjustment*, which is defined as the excess mean-square error of an adaptive filtering algorithm expressed as a percentage of the optimal mean-square error produced by the Wiener filter, which is viewed as a frame of reference for the algorithm.

In Chapter 11, we adopted the rate of convergence as the measure of statistical efficiency:

> Typically, it is here that the RLS algorithm outperforms the LMS algorithm by an order of magnitude.

Accordingly, when statistical efficiency measured in terms of the rate of convergence is the requirement, the RLS algorithm is the method of choice.

In light of the discussion so far, we may go on to say:

> There is no way of designing a linear adaptive filtering algorithm that is both robust and statistically efficient.

Rather, the designer needs to make a decision in favor of the LMS or RLS algorithm, or their respective extensions, depending on the application of interest. Saying it another way, there is *no free lunch*, in that one way or another, a trade-off is made between robustness and efficiency.

Complexity

Today, we live in a world that is increasingly defined by *big data* as well as by the need for *fast on-line data processing*. The term "big data" refers to an ensemble of data sets, which is so large and complex that this processing is an emerging challenge. To meet these challenging trends in the context of linear adaptive filtering, we are compelled to keep algorithmic complexity as low as possible. To satisfy this requirement, it is highly desirable to opt for an adaptive filtering algorithm for which the computational complexity increases *linearly* with respect to the number of adjustable tap weights (i.e., parameters) in the finite-duration impulse response (FIR) filter, particularly when the requirement calls for a large number of tap weights.

In such circumstances, the question to be resolved is:

> Which of the two adaptive filtering algorithms, the LMS and the RLS, is the method of choice when computational complexity or simplicity is a requirement?

From the discussions presented in Chapters 6 and 9, we recall that algorithmic complexity of the LMS follows a *linear law*, whereas that of the RLS follows a *square law*. Hence, the obvious answer to the question just raised is to opt for the LMS algorithm.

Given that the LMS algorithm distinguishes itself from the RLS algorithm in two important practical respects—namely, computational simplicity and robustness—it is not surprising that the LMS algorithm is the most widely used tool for linear adaptive filtering applications. It will remain so for years to come as well.

With computationally efficient processing of big data as the challenge, we may raise the next question:

> How do we build on the widely used LMS algorithm such that linear adaptive filtering power is enhanced significantly, including the elimination of manual tuning of the step-size parameter?

The answer to this increasingly important question is in two parts, which follow from Chapter 13 on adaptation in nonstationary environments:

1. The LMS algorithm is expanded by *vectorizing* its step-size parameter and introducing a new parameter called the *meta-step-size parameter*. The new result of

this expansion, exemplified by the *incremental delta-bar-delta (IDBD) algorithm* (Sutton, 1992), makes it possible for each element in the vector to adaptively match a particular feature of the input data.

2. For complete automatic adjustment of all the parameters in the IDBD algorithm, an effort is made to eliminate the manual tuning of the meta-step-size parameter. The result, which takes care of the second part of the question by building on the IDBD algorithm in a *heuristic* manner, is called the *Autostep method* (Mahmood, 2013).

Both the IDBD algorithm and the Autostep method are rooted in the method of stochastic gradient descent, which was discussed in Chapter 5. Hence, just like the LMS algorithm, they both follow a *linear law* for their computational complexity.

Summarizing Remarks

The Autostep method brings a new way of thinking to linear adaptive filtering in a twofold sense:

1. The Autostep method employs an *adaptation-within-adaptation mechanism*.
2. With manual tuning of the meta-step-size parameter in the IDBD algorithm eliminated, all the parameters in the Autostep method are now automatically adjustable.

We have referred to the Autostep method as a heuristic expansion of the IDBD algorithm, aimed at eliminating the meta-step-size parameter through clever formulation. The challenge is how to improve on the Autostep method in a rigorous mathematical way!

2. KERNEL-BASED NONLINEAR ADAPTIVE FILTERING

In much of the material covered in this book, we focused attention on *linear* adaptive filtering, in the sense that no nonlinear physical element was built into the filtering structure (except for Chapter 17 on blind deconvolution). However, there are practical situations in which the underlying physical mechanism generates training data (for supervised adaptation) that are inherently nonlinear. For example, in underwater communications, the communication channel is not only highly nonstationary but also nonlinear. In such situations, there may well be a need for nonlinear adaptive filtering. Nonlinear adaptive filtering could make a difference in other applications as well.

In this section,[1] we revisit the popular LMS algorithm, but this time nonlinearity is purposely built into the filtering structure. The practical utility of the LMS algorithm is thereby extended beyond its traditional adaptive signal-processing power. The new algorithm is called *kernel least mean square (KLMS)*, which is a member of a new class of broadly defined *kernel adaptive filters*, in which a kernel is responsible for nonlinearity. The motivation for using this kind of nonlinearity is twofold[2]:

1. Transform a nonlinear adaptive filtering problem into a linear one, so that we may explore the literature on linear adaptive filters, be they of the LMS or RLS variety.

[1] Much of the material presented in this section, up to and including the representer theorem, is adapted from Section 6.10 of the book (Haykin, 2009).

[2] Liu et al. (2010) present a detailed account of two families of kernel adaptive filters, those that exploit the LMS algorithm and its extensions and those that exploit the RLS algorithm and its extensions.

2. Structure the new nonlinear adaptive filtering algorithm, so as to operate iteratively in an on-line manner.

To be more precise, the mathematical formulation of kernel adaptive filtering algorithms is rooted in reproducing kernel Hilbert spaces. As such, we may refer to this new class of adaptive filtering algorithms as nonlinear in an overall sense but linear in *reproducing kernel Hilbert spaces*; it is in the latter context that linear adaptive filtering comes into play. To be consistent with the rest of the book, the formulation is carried out entirely in the complex domain.

In a historical context, the roots of reproducing kernel Hilbert spaces may be traced to a classic paper by Aronszajn (1950). However, it was in the machine-learning literature where this new way of thinking developed, particularly so in a class of machine-learning algorithms called *support vector machines* (Boser et al., 1992; Vapnik, 1998; Schölkopf & Smola, 2002).

Kernel adaptive filtering is in its early stages of development. Nevertheless, it offers a number of useful attributes (Theodoridis et al., 2011 and Theodoridis, 2012):

1. Use is made of *convex* cost functions, which lead to unique, well-characterized solutions.
2. Demands on computational and memory resources can be relatively moderate.
3. Unified mathematical treatment of different types of nonlinearities is provided.

As with every innovation, however, kernel adaptive filtering algorithms do have limitations of their own, which are discussed later in the section.

The Complex Hilbert Space

As already mentioned, the idea of a *reproducing kernel Hilbert space (RKHS)* plays a key role in the underlying mathematical theory of the *kernel least-mean-square (KLMS) filtering algorithm.* To set the stage for defining the RKHS, we must start with what a Hilbert space is. To simplify mathematical exposition throughout this section, we find it convenient for the sake of simplicity to use the following notation:

$$\mathbf{u}_n \equiv \mathbf{u}(n) \tag{3}$$

for the signal vector applied to the input of the KLMS that is the focus of attention herein; as before, n stands for discrete time.

Consider, then, a complex nonlinear function $\varphi(\mathbf{u}_n)$,[3] whose argument $\mathbf{u}_n$ lies in a *complex Euclidean space*. However, the complex continuous space, in which the function $\varphi(\mathbf{u}_n)$ resides, is entirely different. To define this new space, we require it to satisfy the following two distinct mathematical conditions:

1. The space is of infinite dimension.
2. Every Cauchy sequence[4] in the space is convergent.

[3]In reality, $\varphi(\mathbf{u}_n)$ is a *functional*, in that it represents a function of a function.

[4]Consider the sequence $\{\varphi(\mathbf{u}_n)\}_{n=1}^{M}$ for varying n. Such a sequence is said to be a *Cauchy sequence* if it satisfies the following requirement:

For any $\varepsilon > 0$, there is an integer M such that $|\varphi(\mathbf{u}_n) - \varphi(\mathbf{u}_i)| < \varepsilon$ for all $n, i > M$.

A complex continuous functional space, which satisfies both of these two conditions, is said to be a *complex Hilbert space*; henceforth, this space is denoted by $\mathcal{H}$, which is not to be confused with the same symbol used in Chapter 17 on blind deconvolution.

Properties of the Complex Hilbert Space

Properties of the complex Hilbert space include the following:

1. *Conjugate symmetry*. The *inner product* of a pair of nonlinear functions, $\varphi(\mathbf{u}_n)$ and $\varphi(\mathbf{u}_i)$ in $\mathcal{H}$, satisfies the property of conjugate symmetry, as shown by

$$\langle \varphi(\mathbf{u}_n), \varphi(\mathbf{u}_i) \rangle_{\mathcal{H}} = \langle \varphi(\mathbf{u}_i), \varphi(\mathbf{u}_n) \rangle^* \quad \text{for all } n \text{ and } i, \tag{4}$$

   where the asterisk denotes *complex conjugation*.

2. *Positive semidefiniteness*. The *norm* of a complex nonlinear function, $\varphi(\mathbf{u}_n)$, defined as the inner product of the function $\varphi(\mathbf{u}_n)$ with itself in $\mathcal{H}$, satisfies the property of positive semidefiniteness, defined by

$$\begin{aligned} \|\varphi(\mathbf{u}_n)\|_{\mathcal{H}}^2 &= \langle \varphi(\mathbf{u}_n), \varphi(\mathbf{u}_n) \rangle_{\mathcal{H}} \\ &\geq 0 \quad \text{for all } n. \end{aligned} \tag{5}$$

   The term $\|\varphi(\mathbf{u}_n)\|_{\mathcal{H}}^2$ is called the squared *norm* of the complex nonlinear function $\varphi(\mathbf{u}_n)$ in $\mathcal{H}$.

3. *Distribution (linearity).* Given a pair of constants, a and b, the distribution (linearity) of complex nonlinear functions in the complex Hilbert space is described as follows:

$$\langle (a\varphi(\mathbf{u}_n) + b\varphi(\mathbf{u}_m)), \varphi(\mathbf{u}_l) \rangle_{\mathcal{H}} = a\langle \varphi(\mathbf{u}_n), \varphi(\mathbf{u}_l) \rangle + b\langle (\varphi(\mathbf{u}_m), \varphi(\mathbf{u}_l)) \rangle_{\mathcal{H}}, \tag{6}$$

   which holds for all constants a and b as well as all indices n, m, and l.

The Notion of a Kernel

Now that we have defined a complex Hilbert space, the next issue to address is the notion of a kernel.

Just as an FIR filter is at the heart of linear adaptive filtering, so a kernel is at the heart of the new class of nonlinear adaptive filters that we have in mind. In linear adaptive filtering, we only have two spaces: an *input space*, where the incoming data lie, and an *output space*, where the actual response of the filter lies. On the other hand, kernel-based adaptive filters have three spaces: an input space, a feature space, and an output space. The *feature space* is naturally hidden, in that it is not directly reachable from the outside. The kernel lies in the feature space.

The input space of the KLMS algorithm is *wired directly* (i.e., in a weightless manner) to the feature space, and the feature space is *linearly connected* to the output space. It is in the latter setting that the LMS comes into play in the KLMS algorithm. In general, the input space is a complex Euclidean space as mentioned previously, whereas the feature space is a complex Hilbert space by design.

To move on, consider Fig. 1, where attention is focused on the *nonlinear mapping* from the input space to the feature space. Specifically, the input vectors, $\mathbf{u}_n$ and $\mathbf{u}_i$, are

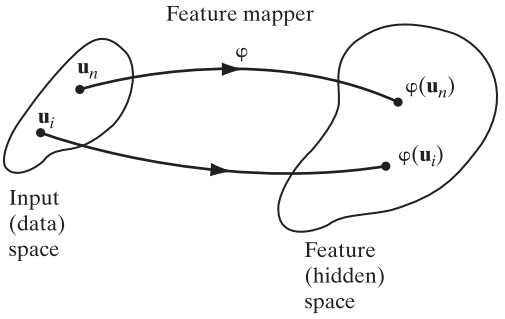


FIGURE 1 Illustration of the nonlinear mapping of vectors $\mathbf{u}_n$ and $\mathbf{u}_i$ in the input space onto the respectively transformed nonlinear functions $\varphi(\mathbf{u}_n)$ and $\varphi(\mathbf{u}_i)$ in the feature space.

mapped into the complex nonlinear functions, $\varphi(\mathbf{u}_n)$ and $\varphi(\mathbf{u}_i)$, respectively. With such a setting, we may make the following, intuitively satisfying statement:

> A kernel, denoted by $k(\mathbf{u}_n, \mathbf{u}_i)$, provides a measure of the similarity (matching) that may exist between the pair of complex nonlinear functions, $\varphi(\mathbf{u}_n)$ and $\varphi(\mathbf{u}_i)$, in the feature space.

Putting this statement in mathematical form in terms of inner products in the complex Hilbert space, we may formally define the kernel as follows:

$$k(\mathbf{u}_n, \mathbf{u}_i) = \langle(\varphi(\mathbf{u}_i), \varphi(\mathbf{u}_n))\rangle_{\mathcal{H}}. \tag{7}$$

In this defining equation, note carefully that the indices n and i in the kernel $k(\mathbf{u}_n, \mathbf{u}_i)$ have been reversed with respect to those in the inner product of the complex nonlinear functions $\varphi(\mathbf{u}_i)$ and $\varphi(\mathbf{u}_n)$ in $\mathcal{H}$. Another point that should also be noted: Earlier in this section, we required that the complex Hilbert space be of infinite dimension. It therefore follows that the *feature map*, φ, can take care of all kinds of kernels, be they of finite or infinite dimension, as well as real or complex.

In light of how the kernel $k(\mathbf{u}_n, \mathbf{u}_i)$ is defined in Eq. (7), we may propose another insightful way of describing it:

> The kernel $k(\mathbf{u}_n, \mathbf{u}_i)$ provides a measure of the similarity (matching) between the *images* produced in the feature space under a form of nonlinear embedding, denoted by the feature map, φ, which is applied to any pair of vectors, $\mathbf{u}_n$ and $\mathbf{u}_i$, that lie in the complex Euclidean space (i.e., input space).

In the machine-learning literature, the kernel defined mathematically in Eq. (7) is commonly referred to as a *Mercer kernel*. It is so named in recognition of the fact that the kernel $k(\mathbf{u}_n, \mathbf{u}_i)$ satisfies a theorem in *functional analysis* that was first described by Mercer (1909).[5] Henceforth, we will adopt the name *Mercer kernel*.

[5] *Mercer's theorem* (Mercer, 1909). This theorem was originally formulated for a real domain. To be consistent with the material presented herein, the theorem is stated in the complex domain as follows:

> Let $k(\mathbf{u}, \tilde{\mathbf{u}})$ be a continuous Hermitian symmetric kernel, where the vectors $\mathbf{u}$ and $\tilde{\mathbf{u}}$ reside in a complex Euclidean space, denoted by $\mathcal{U}$. This kernel can be expanded in the series
>
> $$k(\mathbf{u}, \tilde{\mathbf{u}}) = \sum_{i=1}^{\infty} \lambda_i q_i^*(\mathbf{u}) q_i((\tilde{\mathbf{u}}),$$
>
> **(continued)**

Properties of the Mercer Kernel

Properties of the Mercer kernel include the following:

1. *The Mercer kernel is a positive semidefinite function of its arguments.* To demonstrate this property, we first express the kernel $k(\mathbf{u}_n, \mathbf{u}_i)$ in the following equivalent form:

$$k(\mathbf{u}_n, \mathbf{u}_i) = \langle \mathrm{k}(\bullet, \mathbf{u}_i),\ k(\bullet, \mathbf{u}_n)\rangle_{\mathcal{H}}, \tag{8}$$

which is another way of describing an inner product in the complex Hilbert space, with the Mercer kernel viewed as a complex nonlinear function in its own right. Hence, given $n = 1, 2, \ldots, N$ and likewise for i, and given a corresponding vector of constants—namely, $\mathbf{c}_n = \{c_n\}_{n=1}^{N}$ and likewise for $\mathbf{c}_i$—we may go on to write the following:

$$\begin{aligned}
\sum_{n=1}^{N}\sum_{i=1}^{N} c_n^* k(\mathbf{u}_n, \mathbf{u}_i) c_i &= \sum_{i=1}^{n}\left(\sum_{n=1}^{N} c_n^* \langle k(\bullet, \mathbf{u}_i), k(\bullet, \mathbf{u}_n)\rangle_{\mathcal{H}}\right) \\
&= \sum_{i=1}^{N} c_i \left\langle k(\bullet, \mathbf{u}_i),\ \sum_{n=1}^{N} c_n k(\bullet, \mathbf{u}_n)\right\rangle_{\mathcal{H}} \\
&= \left\langle \sum_{i=1}^{N} c_i k(\bullet, \mathbf{u}_i),\ \sum_{n=1}^{N} c_n k(\bullet, \mathbf{u}_n)\right\rangle_{\mathcal{H}} \\
&= \left\| \sum_{n=1}^{N} c_n k(\bullet, \mathbf{u}_n)\right\|_{\mathcal{H}}^2 .
\end{aligned} \tag{9}$$

with nonnegative $\lambda_i \geq 0$ for all i. For this expansion to converge absolutely and uniformly, it is necessary and sufficient that the condition

$$\int_{\mathcal{U}}\int_{\mathcal{U}} k(\mathbf{u}, \tilde{\mathbf{u}})\psi^*((\mathbf{u})\psi(\tilde{\mathbf{u}})\mathrm{d}\mathbf{u}\mathrm{d}\tilde{\mathbf{u}} > 0$$

holds for all $\psi(\bullet)$, *normalized* such that

$$\int_{\mathcal{U}} |\psi(\mathbf{u})|^2 \mathrm{d}\mathbf{u} = 1.$$

The $q_i(\mathbf{u})$ are called the *eigenfunctions* of the expansion, and the corresponding λ_i are called the associated *eigenvalues*. The fact that all of the eigenvalues are nonnegative means that the kernel is *positive semidefinite*.

Mercer's theorem is remarkable in that it defines not only the necessary and sufficient condition for the series expansion of the kernel $k(\mathbf{u}, \tilde{\mathbf{u}})$ to be convergent but also provides a constructive procedure for the corresponding *feature map*. To demonstrate this latter point, consider the following *mapper*:

$$\boldsymbol{\phi}(\mathbf{u}) = [\ \sqrt{\lambda_1} q_1(\mathbf{u}), \sqrt{\lambda_2} q_2(\mathbf{u}), \ldots\ ]^{\mathrm{T}},$$

where the superscript T denotes transposition. We may then go on to express the kernel $k(\mathbf{u}, \tilde{\mathbf{u}})$ in its customary form, as shown by

$$\begin{aligned}
k(\mathbf{u}, \tilde{\mathbf{u}}) &= \sum_{i=1}^{\infty} \left(\sqrt{\lambda_i} q_i(\mathbf{u})\right)^* \cdot \left(\sqrt{\lambda_i} q_i(\tilde{\mathbf{u}})\right) \\
&= \sum_{i=1}^{\infty} \varphi_i^*(\mathbf{u}) \cdot \varphi_i(\tilde{\mathbf{u}}) \\
&= \langle \varphi_i(\mathbf{u}), \varphi_i(\tilde{\mathbf{u}})\rangle_{\mathcal{H}} .
\end{aligned}$$

In the first line of Eq. (9), we made use of Eq. (8) on the right-hand side. Then, in the second and third lines, we made use of the distribution property of $\mathcal{H}$ in Eq. (6). In the last line we made use of the positive semidefiniteness property of $\mathcal{H}$ in Eq. (5).

2. *The Mercer kernel is a conjugate-symmetric function.* This second property is justified as follows:

$$\begin{aligned} k(\mathbf{u}_n, \mathbf{u}_i) &= \langle k(\bullet, \mathbf{u}_i), k(\bullet, \mathbf{u}_n) \rangle_{\mathcal{H}} \\ &= \langle k(\bullet, \mathbf{u}_n), k(\bullet, \mathbf{u}_i) \rangle_{\mathcal{H}}^{*} \\ &= k^{*}(\mathbf{u}_i, \mathbf{u}_n). \end{aligned} \tag{10}$$

3. *The Mercer kernel, $k(\mathbf{u}_n, \mathbf{u}_i)$, is an element of an N-by-N positive semidefinite matrix.* To justify this third and last property of the Mercer kernel, we start by introducing an N-by-N matrix:

$$\begin{aligned} \mathbf{K} &= \{k(\mathbf{u}_n, \mathbf{u}_i)\}_{n,\, i=1}^{N} \\ &= \begin{bmatrix} k(\mathbf{u}_1, \mathbf{u}_1) & k(\mathbf{u}_1, \mathbf{u}_2) & \cdots & k(\mathbf{u}_1, \mathbf{u}_N) \\ k(\mathbf{u}_2, \mathbf{u}_1) & k(\mathbf{u}_2, \mathbf{u}_2) & \cdots & k(\mathbf{u}_2, \mathbf{u}_N) \\ \vdots & \vdots & & \vdots \\ k(\mathbf{u}_N, \mathbf{u}_1) & k(\mathbf{u}_N, \mathbf{u}_2) & \cdots & k(\mathbf{u}_N, \mathbf{u}_N) \end{bmatrix}. \end{aligned} \tag{11}$$

This matrix is a *positive semidefinite matrix*, in that for any complex N-by-1 vector, $\mathbf{c}$, the matrix $\mathbf{K}$ satisfies the following condition:

$$\mathbf{c}^{\mathrm{H}}\mathbf{K}\mathbf{c} \geq 0, \tag{12}$$

where the superscript H denotes *Hermitian matrix transposition*. The condition described in Eq. (12) follows readily from the definitions of matrix $\mathbf{K}$ and vector $\mathbf{c}$. Accordingly, we say that the matrix $\mathbf{K}$ is a *Hermitian-symmetric matrix*; that is, $\mathbf{K} = \mathbf{K}^{\mathrm{H}}$. Such a matrix is called the *kernel matrix*; it is also referred to as the *Gram matrix* in the machine-learning literature.

EXAMPLE 1 Gaussian Kernel

For an illustrative example, consider a *Gaussian kernel*[6] defined as follows:

$$k(\bullet, \mathbf{u}_n) = \exp\left(-\frac{1}{2\sigma^2}\|\bullet - \mathbf{u}_n\|^2\right), \tag{13}$$

where the data vector $\mathbf{u}_n$ defines the *center* of the Gaussian kernel and σ^2 provides a measure of its *width* (spread) for all n. Indeed, the feature space of the Gaussian kernel has infinite dimension, a point that follows readily from the defining equation (13).

---

[6]The Gaussian kernel is one of the many kernels described in the machine-learning literature (see, e.g., Schölkopf & Smola, 2002).

The Reproducing Property of Mercer Kernels

Consider next a function $f(\bullet)$ that is picked from the complex Hilbert space $\mathcal{H}$, in which $\bullet$ denotes an arbitrary nonlinear vector. This function may be expressed by the following series expansion in terms of the kernel $k(\bullet, \mathbf{u}_i)_{\mathcal{H}}$ as follows:

$$f(\bullet) = \sum_{i-1}^{l} a_i k(\bullet, \mathbf{u}_i), \tag{14}$$

where a_i is the coefficient of the expansion. The inner product of the function $f(\bullet)$ and the Mercer kernel $k(\bullet, \mathbf{u})$ in $\mathcal{H}$ are expressed as follows:

$$\begin{aligned} \langle f(\bullet), k(\bullet, \mathbf{u}) \rangle_{\mathcal{H}} &= \left\langle \left(\sum_{i=1}^{l} a_i k(\bullet, \mathbf{u}_i) \right), k(\bullet, \mathbf{u}) \right\rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{l} a_i \langle k(\bullet, \mathbf{u}_i), k(\bullet, \mathbf{u}) \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{l} a_i k(\mathbf{u}, \mathbf{u}_i) \\ &= f(\mathbf{u}), \end{aligned} \tag{15}$$

where in the second line we made use of Eq. (8). The final result obtained in Eq. (15) is the original function $f(\bullet)$ in Eq. (14), with the vector $\mathbf{u}$ assuming the role of its argument.

For obvious reasons, the property of the Mercer kernel described in Eq. (15) is known as the *reproducing property* of the Mercer kernel (Aronszajn, 1950). We may therefore say:

> The Mercer kernel is endowed with the *reproducing kernel Hilbert space* (RKHS) property.

Hereafter, we refer to this designation as the *RKHS property.*

The Kernel Trick

Previously, we mentioned that dimensionality of the feature space can be high, and possibly infinitely so, compared to the input space. Consequently, algorithmic optimization of the KLMS algorithm in the feature space can be computationally demanding. To overcome this difficulty, formulate the KLMS algorithm in terms of *images* of the input vectors; that is, follow the second intuitive way of describing the Mercer kernel introduced in words earlier. Mathematically speaking, we may employ the RKHS property—namely,

$$\begin{aligned} \langle \varphi(\mathbf{u}_i), \varphi(\mathbf{u}_n) \rangle_{\mathcal{H}} &= \langle k(\bullet, \mathbf{u}_i), k(\bullet, \mathbf{u}_n) \rangle_{\mathcal{H}} \\ &= k(\mathbf{u}_n, \mathbf{u}_i), \end{aligned} \tag{16}$$

which is a restatement of Eq. (7). In so doing, the costly computation of inner products in the high-dimensional feature space is replaced with a much less demanding computation of kernels in the original Euclidean input space, where both $\mathbf{u}_n$ and $\mathbf{u}_i$ reside.

To implement the property described in Eq. (16), we proceed in two steps:

Step 1. Formulate the KLMS algorithm in terms of complex continuous functions in the feature space.

Step 2. To optimize the algorithm, recast it by replacing the computation of inner products in the feature space with the computation of the Mercer kernel in the complex Euclidean input space.

The rule behind replacing step 1 with step 2 is commonly referred to as the *kernel trick* in the machine-learning literature.

The Representer Theorem

We finish off the preparatory material on kernel-based adaptive filtering, leading to the formulation of the RKHS, by demonstrating the analytic power of working in this new space. To proceed, define a space $\mathcal{H}$ viewed as the RKHS, which is induced by the Mercer kernel $k(\bullet, \mathbf{u})$. Given any complex nonlinear function $f(\bullet) \in \mathcal{H}$, we may decompose into the sum of two components, both of which lie in $\mathcal{H}$:

1. One component is contained in the *span* of the kernel functions, denoted by $k(\bullet, \mathbf{u}_1), k(\bullet, \mathbf{u}_2), \ldots, k(\bullet, \mathbf{u}_l)$. Denoting this component by the function of $f_{\|}(\bullet)$, we may use Eq. (15) to represent it by the series expansion:

$$f_{\|}(\bullet) = \sum_{i=1}^{l} a_i k(\bullet, \mathbf{u}_i)_{\mathcal{H}}. \tag{17}$$

2. The second component is *orthogonal* to the span of the kernel functions, which is denoted by $f_{\perp}(\bullet)$.

Thus, the original function $f(\bullet)$ is expressed as the summation:

$$\begin{aligned} f(\bullet) &= f_{\|}(\bullet) + f_{\perp}(\bullet) \\ &= \sum_{i=1}^{l} a_i k(\bullet, \mathbf{u}_i)_{\mathcal{H}} + f_{\perp}(\bullet). \end{aligned} \tag{18}$$

Next, consider a complex nonlinear function, denoted by $f(\mathbf{u}_n)$ in the RKHS. Building on Eq. (8), we may express this new function as follows:

$$\begin{aligned} f(\mathbf{u}_n) &= \langle f(\bullet), k(\bullet, \mathbf{u}_n) \rangle_{\mathcal{H}} \\ &= \left\langle \left(\sum_{i=1}^{l} a_i k(\bullet, \mathbf{u}_i) + f_{\perp}(\bullet) \right), k(\bullet, \mathbf{u}_n) \right\rangle_{\mathcal{H}} \\ &= \left\langle \left(\sum_{i=1}^{l} a_i k(\bullet, \mathbf{u}_i), k(\bullet, \mathbf{u}_n) \right) \right\rangle_{\mathcal{H}} + \langle f_{\perp}(\bullet), k(\bullet, \mathbf{u}_n) \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{l} a_i \langle k(\bullet, \mathbf{u}_i), k(\bullet, \mathbf{u}_n) \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{l} a_i k(\mathbf{u}_n, \mathbf{u}_i), \end{aligned} \tag{19}$$

where in the third line we made use of the distribution (linearity) of the complex Hilbert space in Eq. (7), in the fourth line we used the zero orthogonality of $f_{\perp}(\bullet)$ with respect to $f_{\perp}(\bullet)$, and finally, we made use of Eq. (8).

Equation (19) is a mathematical statement of the *representer theorem*,[7] which, in words, may be expressed as follows:

> Any complex nonlinear function in the RKHS can be represented by a linear combination of Mercer kernels.

As we shall see in the next subsection, the representer theorem occupies an important place in formulation of the KLMS algorithm.

The Kernel Least-Mean-Square (KLMS) Algorithm

At long last, we can derive the KLMS algorithm. Naturally, to do this derivation we build not only on the preparatory material developed so far in this section but also on the related material for the traditional LMS algorithm covered in Chapter 6.

With this objective in mind, we first refer back to Chapter 6 to reproduce the weight-update formula of Eq. (6.3) as follows:

$$\hat{\mathbf{w}}_{n+1} = \hat{\mathbf{w}}_n + \mu e_n^* \mathbf{u}_n,$$

where we have used the simplified notation defined in Eq. (3). Then, under the assumed zero initial condition, we find that repeated application of this update formula yields the following input–output relationship:

$$\begin{aligned} \hat{d}_n &= \hat{\mathbf{w}}_n^{\mathrm{H}} \mathbf{u}_n \\ &= \mu \sum_{i=1}^{n-1} e_i \mathbf{u}_i^{\mathrm{H}} \mathbf{u}_n, \end{aligned} \tag{20}$$

where $\hat{d}_n$ is the estimate of the desired response, d_n, produced at the output of the LMS algorithm in response to the input vector $\mathbf{u}_n$; the e_i denotes an error signal. Examination of Eq. (20) reveals two noteworthy points:

1. *Linearity of the input–output relationship*, which reaffirms the LMS as a linear filtering algorithm.
2. *Bypassing of the weight vector* $\hat{\mathbf{w}}_n$, which is of particular interest to the discussion at hand.

Referring back to Fig. 1, we see that for an input vector $\mathbf{u}_i$ in the complex Euclidean input space, its nonlinearly mapped version in the feature space is the complex function $\varphi(\mathbf{u}_i)$. Based on this observation, we may construct the *analogy* between the linear LMS algorithm and its nonlinear counterpart, the KLMS algorithm, as shown in Table 1. This table is particularly helpful in using what we know about the LMS algorithm and applying it to the KLMS algorithm. Indeed, we may look to this table as the linkage between linear adaptive filtering and kernel-based nonlinear filtering.

In light of Table 1, we may make the following statement:

> In place of the inner product $\mathbf{u}_i^{\mathrm{H}} \mathbf{u}_n$ in Eq. (20) for the complex Euclidean input space of the traditional LMS algorithm, we have the inner product $\langle \varphi(\mathbf{u}_i), \varphi(\mathbf{u}_n) \rangle_{\mathcal{H}}$ in the complex Hilbert space for the KLMS algorithm.

[7] The celebrated representer theorem was first described in the paper by Kimeldorf and Wahba (1971) for solving practical problems in statistical estimation based on squared-loss (cost) functions.

TABLE 1 Mathematical Analogy Between the LMS and KLMS Algorithms

Algorithm	Inputs	Complex inner products	Correlation (covariance) function
Traditional LMS	$\mathbf{u}_i, \mathbf{u}_n$	Complex Euclidean space: $\mathbf{u}_i^{\mathrm{H}}\mathbf{u}_n$	Correlation of tap-input vector: $\mathbf{R}$
KLMS	Transformed inputs: $\varphi(\mathbf{u}_i), \varphi(\mathbf{u}_n)$	Complex Hilbert space: $\langle \varphi(\mathbf{u}_i), \varphi(\mathbf{u}_n) \rangle_{\mathcal{H}}$	Kernel matrix (Gram): $\frac{1}{N}\mathbf{K}$, where N is the size of the training data set over which the matrix is defined.

Accordingly, recognizing that the feature space is *linearly* connected to the output space in the KLMS algorithm, we may build on Eq. (20) to formulate the *estimate* of the desired response d_n in the KLMS algorithm in response to the input $\mathbf{u}_n$, which we denote by $\hat{d}_n$, as shown by

$$\begin{aligned} \hat{d}_n &= \mu \sum_{i=1}^{n-1} e_i \langle \varphi(\mathbf{u}_i), \varphi(\mathbf{u}_n) \rangle_{\mathcal{H}} \\ &= \mu \sum_{i=1}^{n-1} e_i k(\mathbf{u}_n, \mathbf{u}_i), \end{aligned} \tag{21}$$

where in the second line we made use of Eq. (16).

In Eq. (21), we have an important formula for the estimate $\hat{d}_n$ because it bypasses the need for computing the weight vector in the KLMS algorithm. This formula is important for the following simple reason:

> In the KLMS algorithm, depending on how the Mercer kernel is chosen, it is possible for iterative computation of the weight vector to go on expanding indefinitely because the feature space of the algorithm may have an infinite dimension of its own.

The key question to be resolved therefore is:

> How do we overcome this computational difficulty?

The answer is to bypass the need for employing weights and build on the representer theorem for guidance.

To be specific, comparing Eq. (21) for the KLMS algorithm with Eq. (19) for the representer theorem, we may readily make the following observation by setting $l = n-1$ and $f(\mathbf{u}_n) = d_n$:

> The KLMS algorithm is a special case of the representer theorem.

Specifically, we may express Eq. (20) in the equivalent form:

$$\hat{d}_n = \sum_{i=1}^{n-1} a_i k(\mathbf{u}_n, \mathbf{u}_i), \tag{22}$$

which is produced in response to the input vector $\mathbf{u}_n$. Moreover, as a result of the comparison, we have

$$a_i = \mu e_i, \quad i = 1, 2, \ldots, n-1. \tag{23}$$

With d_n denoting the actual desired response, which becomes available at adaptation cycle n, we may express the corresponding *error signal* in the usual way:

$$e_n = d_n - \hat{d}_n. \tag{24}$$

Based on Eq. (22), we may now describe the topological composition of the KLMS algorithm in the form shown in Fig. 2. The interesting point to note here is the fact that the part of the figure labeled "linear combiner" plays a role in the KLMS algorithm similar to that of the linear combiner in the tradition LMS algorithm, with one basic difference:

> The linear combiner in the KLMS algorithm is made up of error signal–based parameters, whereas the linear combiner in the LMS algorithm is made up of the FIR filter's tap weights.

Comparison of KLMS Topology with Radial-Basis Function (RBF) Network

At first glance the KLMS topology in Fig. 2 may bring to mind the *radial-based function (RBF) network*, which is widely used in machine learning (Haykin, 2009). In reality, however, the KLMS algorithm differentiates itself from the RBF network in three ways:

1. The KLMS algorithm can accommodate a variety of Mercer kernels that are *not* RBFs.

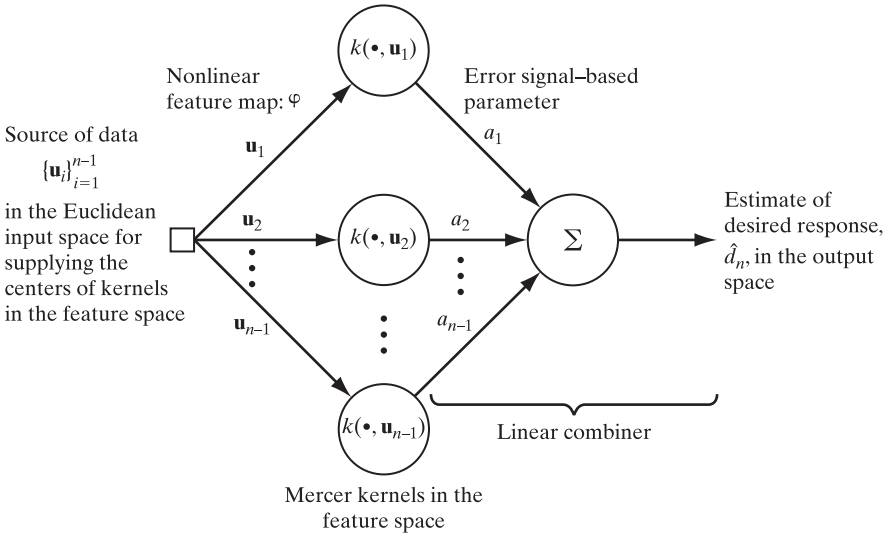


FIGURE 2 Topological diagram of the KLMS algorithm based on the representer theorem.

2. Unlike the RBF network with a fixed structure, the KLMS network of Fig. 2 is a network that *grows* with time.
3. The parameters that connect the feature space to the output space are not weights; rather, the a_i represent error signal–based parameters in accordance with Eq. (23).

Moreover, the KLMS algorithm is a *simple* kernel-based adaptive filtering algorithm for which computational complexity follows a *linear law*, just like the traditional LMS algorithm.

Formulation of the KLMS Algorithm

Table 2 presents the proposed outline of the KLMS algorithm in its complex form, for which we may use the Gaussian kernel, for example. The early part of the table—namely, *Preliminaries, Selections*, and *Initializations*—speaks for itself. Herein, we elaborate on the three computational steps that represent the core of the algorithm.[8]

In step 1 under *Computation* in Table 2, we build on the insightful use of Eq. (20) pertaining to the traditional LMS algorithm, thereby formulating Eq. (22) for the KLMS algorithm. This latter equation defines the estimate of the desired response, $\hat{d}_n$, produced by the KLMS algorithm in response to the input vector $\mathbf{u}_n$.

Step 2 under *Computation* follows the traditional terminology in adaptive filtering, wherein the error signal, e_n, is defined as the difference between the actual desired response, d_n, and its estimate, $\hat{d}_n$, as shown in Eq. (24).

Finally, step 3 under *Computation* provides the update for the parameter a_n in the representer theorem in light of the updated error signal, e_n, computed in step 2. In particular, use is made of Eq. (23) for $i = n$.

The description of these three steps is testimony to how simple the KLMS algorithm is in computational terms.

Properties Inherited from the LMS Algorithm

Regularization. Much of the commonality that exists between the traditional LMS and KLMS algorithms is attributed to the fact that they are both example applications of the method of stochastic gradient, discussed in Chapter 5. Accordingly, they are

[8]In Bouboulis and Theodoridis (2011) and Bouboulis et al. (2012), derivation of the KLMS algorithm follows an entirely different procedure from that described in this section.

To be specific, the Wirtinger calculus described in Appendix B is generalized. This generalization is achieved by incorporating the *Frechét differential* into the Wirtinger calculus, hence the reference to the generalization as the *Frechét–Wirtinger calculus*. The Frechét differential deals with functionals, which befits the fact that, as pointed out previously in footnote 3, the feature $\varphi(\mathbf{u}_n)$ is in reality a functional. With the Frechét–Wirtinger calculus as the mathematical tool at hand, the stage is set for deriving the KLMS algorithm for nonlinear filtering using the notion of a gradient in the method of stochastic gradient descent. As such, the mathematically rigorous deviation presented in the two above-mentioned papers is developed, first using principles in the complex RKHS and Frechét–Wirtinger calculus.

TABLE 2 The KLMS Algorithm in the Complex Domain

Preliminaries:

Training data: $\{\mathbf{u}_n, d_n\}_{n=1}^{N}$
Dictionary of the input vectors: $\mathcal{U} = \{\mathbf{u}_n\}$
Dictionary of parameters in the representer theorem: $a = \{a_n\}_{n=1}^{N-1}$

Selections:

1. Step-size parameter, μ, is selected in accordance with Eq. (25).
2. Mercer kernel and associated RKHS are selected.
3. The feature space is selected.

Initializations:

1. Error signal, $e_1 = d_1$, recognizing that e_0 for $n = 0$ is zero.
2. Representer theorem parameter, $a_1 = \mu d_1$.
3. First entry in the dictionary, $\mathcal{U}$, is the input vector $\mathbf{u}_1$.

Computation:

For $n = 2, 3, \ldots$, compute

Step 1. $\hat{d}_n = \sum_{i=1}^{n-1} a_i k(\mathbf{u}_n, \mathbf{u}_i)$

Step 2. $e_n = d_n - \hat{d}_n$

Step 3. $a_n = \mu e_n$

Step 4. Go back to step 1. Add the new input $\mathbf{u}_{n+1}$ to the dictionary $\mathcal{U}$, add a_n to the dictionary a, and repeat the computation.

both *model-independent*, which in turn means that just as the traditional LMS algorithm does not require regularization, so it is with the KLMS algorithm.

Statistical Learning Theory. Turning next to the algorithmic *learning curve*, which was discussed at length in Section 6.4 for the traditional LMS algorithm, we may summarize the findings reported there as follows:

> Under the assumption that the step-size parameter, μ, is small, the learning curve for the LMS algorithm is described mathematically in Eq. (6.98), with two useful points in mind:
>
> 1. Equation (6.98) does not include FIR's tap-weights in its formulation.
> 2. Rather, the formulation is based on eigendecomposition of the correlation matrix, $\mathbf{R}$, of the input vector, $\mathbf{u}_n$.

It follows, therefore, that the same formula in Eq. (6.98) applies equally well to the KLMS algorithm provided that, in accordance with the last column in Table 1, we make the following substitution:

> The scaled kernel matrix, $(1/N)\mathbf{K}$, is substituted for the correlation matrix, $\mathbf{R}$, in analyzing Eq. (6.98) to formulate the learning curve for the KLMS algorithm, where N is the size of the training data used to compute $\mathbf{K}$.

On this basis, if λ_{max} denotes the largest eigenvalue of the scaled kernel matrix, $(1/N)\mathbf{K}$, then to assure convergence of the learning curve of the KLMS algorithm, the step-size parameter, μ, must satisfy the following condition:

$$\mu < \frac{2}{\lambda_{max}}. \tag{25}$$

As it is with the LMS algorithm, the preferred choice for μ is to make it small compared to the reciprocal of λ_{max}.

Moreover, in accordance with Eqs. (6.104) and (6.105) in Section 6.6 on the efficiency of the LMS algorithm as well as Table 1, we may go on to characterize the statistical efficiency of the KLMS algorithm by the following two measures:

1. *Rate of convergence.* With *N natural modes* of the KLMS algorithm, each of which is characterized by an eigenvalue of its own, the *k*th *time constant* is defined by

$$\tau_{mse,\,k,\,KLMS} \approx \frac{1}{2\mu\lambda_k}, \quad k = 1, 2, \ldots, N. \tag{26}$$

   Correspondingly, the slowest time constant is given by

$$\tau_{mse,\,min,\,KLMS} \approx \frac{1}{2\mu\lambda_{min}}, \tag{27}$$

   where λ_{min} denotes the smallest eigenvalue of $(1/N)\mathbf{K}$. In effect, Eq. (27) defines the most pessimistic rate of convergence of the KLMS algorithm. Note also that the rate of convergence is inversely proportional to the step-size parameter, μ, which in turn means that the smaller we make μ, the slower the rate of convergence will be.

2. *Misadjustment.* This second measure of statistical efficiency is the excess mean-square error of the KLMS, expressed as a percentage of the optimal mean-square error produced by the Wiener filter. Specifically, we have

$$\mathcal{M}_{KLMS} \approx \frac{\mu}{2N}\text{tr}[\mathbf{K}], \tag{28}$$

   where tr[ ] is the *trace operator*. Here, we see that the misadjustment is directly proportional to the step-size parameter μ; the smaller we make μ, the smaller the misadjustment of the algorithm will be.

Robustness. The material described in Eq. (1) for robustness of the LMS algorithm applies equally well to the KLMS algorithm. Specifically, robustness of the KLMS algorithm is assured provided that its maximum energy gain satisfies the following condition:

$$\gamma_{KLMS}^2 \leq 1 \quad \text{for} \quad 0 < \mu < \min_{n=1,\ldots,N} \frac{1}{\|\varphi(\mathbf{u}_n)\|_{\mathcal{H}}^2} \quad \text{and any integer } N, \tag{29}$$

where $\|\varphi(\mathbf{u}_n)\|_{\mathcal{H}}^2$ is the squared *norm* of the complex nonlinear function $\varphi(\mathbf{u}_n)$, as defined in Eq. (5). This definition is reproduced here in expanded form:

$$\begin{aligned}\|\varphi(\mathbf{u}_n)\|_{\mathcal{H}}^2 &= \langle \varphi(\mathbf{u}_n), \varphi(\mathbf{u}_n)\rangle_{\mathcal{H}} \\ &= \int_{\mathcal{U}} |\varphi(\mathbf{u}_n)|^2 \mathrm{d}\mathbf{u}_n,\end{aligned} \tag{30}$$

where $\mathcal{U}$ denotes the complex Euclidean space in which the input vector $\mathbf{u}_n$ lies for all n.

To fulfill the requirement of Eq. (29), the integral in Eq. (30) would have to evaluated for $n = 1, 2, \ldots, N$ and then the smallest value picked as the permissible upper bound on μ. Evaluation of this integral needs careful consideration. For example, in the case of a Gaussian kernel, $\varphi(\mathbf{u}_n)$, we may have to use the *Monte Carlo integration theorem* (Press et al., 1988) for its approximate evaluation using computer simulations.

EXAMPLE 2 Comparison of the KLMS Versus the LMS Algorithm

In this example, we compare the KLMS versus the traditional LMS algorithm for predicting the monthly unemployment in the United States, consisting of 775 data points that extended from January 1, 1948, to July 1, 2012. The task is to perform one-step prediction, based on the 10 data points (i.e., $n = 10$), and progress in this manner across the complete data set. For the sake of comparison, the learning curve, computed in the manner just described, is repeated for both algorithms.

Figure 3 plots the learning curves for the LMS and KLMS algorithms, using the following specifications:

1. LMS: step-size parameter = 0.1.
2. KLMS: step-size parameter = 1.0.
   Gaussian kernel, with width (spread), $\sigma^2 = 500$.

Examination of the two learning curves in Fig. 3 leads to the following observations:

1. The KLMS algorithm has a rate of convergence that is an order of magnitude faster than that of the LMS algorithm.
2. The misadjustment produced by the KLMS algorithm is smaller than that produced by the LMS algorithm by approximately 20%.

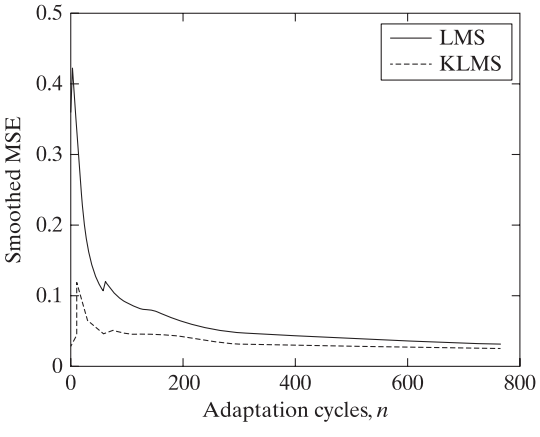


FIGURE 3 Smooth mean-square-error (MSE) curves for the KLMS and traditional LMS algorithms, used to predict the unemployment rate in the United States.

Limitations of the KLMS Algorithm

In the introductory material to this section, we highlighted the attributes of the KLMS algorithm for nonlinear adaptive filtering. Herein, we describe its two practical limitations (Theodoridis et al., 2011 and Theodoridis, 2012):

1. The KLMS algorithm shares the same disadvantages of other kernel methods used in machine learning:

   How do we pick the right Mercer kernel for each specific application encountered in practice?

   Unfortunately, we have yet to find a systematic procedure for how to address this question in practice.

2. Referring back to Eq. (21), it is apparent that the KLMS algorithm has a *memory-growing problem*; that is:

   The memory of the KLMS algorithm grows with increasing number of adaptation cycles.

   There are two ways of tackling the memory-growing problem:

1. *Sparsification.* In this approach, the computation performed in the KLMS algorithm is *sparsified.* To this end, the training data set is viewed as a *dictionary*, as indicated in Table 2, with computational resources being employed *only* when they are needed. Expansion of the memory is thereby limited under the proviso that a prescribed rule is satisfied (Liu et al., 2010).
2. *Quantization.* In this second approach, a quantization process is employed, wherein the size of the input data space or that of the feature space is compressed as needed, as described by Chen et al. (2012). In their paper, a *simple on-line vector quantization method* is adopted with the aim of exploiting the notion of *redundancy*, be that in the input space or the feature space.

Regardless of whether we adopt the sparsification or the quantization approach for mitigating the memory-growing problem, we cannot escape from the *no-free-lunch theorem*, which in the context of the practical issue at hand means that continued memory growth is curtailed by sacrificing optimality.

Summarizing Remarks

The KLMS algorithm is basically a nonlinear extension of the traditional LMS algorithm. In theoretic terms, this new approach to nonlinear adaptive filtering is not only elegant but also enriched by the extensive literature already developed on support vector machines and extensions thereof in machine learning.

Most importantly, experimental results (e.g., those presented in Example 2) indicate that the rate of convergence and misadjustment of the KLMS algorithm improve significantly compared to the traditional LMS algorithm, so much so that its statistical

efficiency is practically on par with that of the traditional RLS algorithm. Accordingly, we may say that when robustness and efficiency are jointly required, they can both be achieved at the expense of increased *system complexity*; however, the *desired linear law of computational complexity is maintained.* The KLMS algorithm may therefore be viewed as another potential candidate for adaptive processing of *big data*, provided that the memory-growing problem is tamed efficiently through sparsification, quantization, their joint deployment, or some other means. Needless to say, this provision is a challenging task and may remain so for some time.

APPENDIX A

Theory of Complex Variables

This appendix presents a brief review of the functional theory of complex variables. In the context of the material considered in the text, a complex variable of interest is the variable z associated with the z-transform. We begin the review by defining analytic functions of a complex variable and then derive the important theorems that make up the important subject of complex variables.[1]

A.1 CAUCHY–RIEMANN EQUATIONS

Consider a complex variable

$$z = x + jy,$$

where $x = \text{Re}[z]$ and $y = \text{Im}[z]$. We speak of the plane in which the complex variable z is represented as the *z-plane*. Let $f(z)$ denote a *function of the complex variable* z, written as

$$w = f(z) = u + jv.$$

The function $w = f(z)$ is *single valued* if there is only one value of w for each z in a given region of the z-plane. If more than one value of w corresponds to z, the function $w = f(z)$ is said to be *multiple valued*.

We say that a point $z = x + jy$ in the z-plane approaches a fixed point $z_0 = x_0 + jy_0$ if $x \to x_0$ and $y \to y_0$. Let $f(z)$ denote a single-valued function of z that is defined in some neighborhood of the point $z = z_0$. The *neighborhood* of z_0 is the set of all points in a sufficiently small circular region centered at z_0. Let

$$\lim_{z \to z_0} f(z) = w_0.$$

In particular, if $f(z_0) = w_0$, then the function $f(z)$ is said to be *continuous* at $z = z_0$.

Let $f(z)$ be written in terms of its real and imaginary parts as

$$f(z) = u(x, y) + jv(x, y).$$

[1]For a detailed treatment of the functional theory of complex variables, see Wylie and Barrett (1982).

Then, if $f(z)$ is continuous at $z_0 = x_0 + jy_0$, its real and imaginary parts, $u(x, y)$ and $v(x, y)$, respectively, are continuous functions at (x_0, y_0), and vice versa.

Let $w = f(z)$ be continuous at each point of some region of interest in the z-plane. The complex quantities w and z may then be represented on separate planes of their own, referred to as the w- and z-planes, respectively. In particular, a point (x, y) in the z-plane corresponds to a point (u, v) in the w-plane by virtue of the relationship $w = f(z)$.

Consider an incremental change Δz such that the point $z_0 + \Delta z$ may lie anywhere in the neighborhood of z_0, throughout which the function $f(z)$ is defined. We may then define the *derivative* of $f(z)$ with respect to z at $z = z_0$ as

$$f'(z_0) = \lim_{\Delta z \to 0} \frac{f(z_0 + \Delta z) - f(z_0)}{\Delta z}. \tag{A.1}$$

Clearly, for the derivative $f'(z_0)$ to have a unique value, the limit in Eq. (A.1) must be independent of the way in which Δz approaches zero.

For a function $f(z)$ to have a unique derivative at some point $z = x + jy$, it is necessary that its real and imaginary parts satisfy certain conditions. Let

$$w = f(z) = u(x, y) + jv(x, y).$$

With $\Delta w = \Delta u + j\Delta v$ and $\Delta z = \Delta x + j\Delta y$, we may write

$$\begin{aligned} f'(z) &= \lim_{\Delta z \to 0} \frac{\Delta w}{\Delta z} \\ &= \lim_{\substack{\Delta x \to 0 \\ \Delta y \to 0}} \frac{\Delta u + j\Delta v}{\Delta x + j\Delta y}. \end{aligned} \tag{A.2}$$

Suppose that we let $\Delta z \to 0$ by first letting $\Delta y \to 0$ and then $\Delta x \to 0$, in which case Δz is purely real. We then infer from Eq. (A.2) that

$$\begin{aligned} f'(z) &= \lim_{\Delta x \to 0} \frac{\Delta u}{\Delta x} + j\frac{\Delta v}{\Delta x} \\ &= \frac{\partial u}{\partial x} + j\frac{\partial v}{\partial x}. \end{aligned} \tag{A.3}$$

Suppose next that we let $\Delta z \to 0$ by first letting $\Delta x \to 0$ and then $\Delta y \to 0$, in which case Δz is purely imaginary. This time, we infer from Eq. (A.2) that

$$\begin{aligned} f'(z) &= \lim_{\Delta y \to 0} \frac{\Delta v}{\Delta y} - j\frac{\Delta u}{\Delta y} \\ &= \frac{\partial v}{\partial y} - j\frac{\partial u}{\partial y}. \end{aligned} \tag{A.4}$$

If the derivative $f'(z)$ is to exist, it is necessary that the two expressions in Eqs. (A.3) and (A.4) be one and the same. Hence, we require that

$$\frac{\partial u}{\partial x} + \mathrm{j}\frac{\partial v}{\partial x} = \frac{\partial v}{\partial y} - \mathrm{j}\frac{\partial u}{\partial y}.$$

Accordingly, equating real and imaginary parts, we get the following pair of relations:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}; \tag{A.5}$$

$$\frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y}. \tag{A.6}$$

Equations (A.5) and (A.6), known as the *Cauchy–Riemann equations*, were derived from a consideration of merely two of the infinitely many ways in which Δz can approach zero. For $\Delta w/\Delta z$ evaluated along these other paths also to approach $f'(z)$, we need only make the additional requirement that the partial derivatives in Eqs. (A.5) and (A.6) are continuous at the point (x, y). In other words, provided that the real part $u(x, y)$ and the imaginary part $v(x, y)$, together with their first partial derivatives, are continuous at the point (x, y), the Cauchy–Riemann equations are not only necessary but also sufficient for the existence of a derivative of the complex function $w = u(x, y) + \mathrm{j}v(x, y)$ at the point (x, y). Accordingly, the function is said to be complex-differential in a complex plane.

A function $f(z)$ is said to be *analytic*, or *holomorphic*, at some point $z = z_0$ in the z-plane if it has a derivative at $z = z_0$ and at every point in the neighborhood of z_0; the point z_0 is called a *regular point* of the function $f(z)$. If the function $f(z)$ is *not* analytic at a point z_0, but if every neighborhood of z_0 contains points at which $f(z)$ is analytic, the point z_0 is referred to as a *singular point* of $f(z)$.

A.2 CAUCHY'S INTEGRAL FORMULA

Let $f(z)$ be any continuous function of the complex variable z, analytic or otherwise. Let $\mathscr{C}$ be a sectionally smooth path joining the points $A = z_0$ and $B = z_n$ in the z-plane. Suppose that the path $\mathscr{C}$ is divided into n segments Δs_k by the points z_k, $k = 1, 2, \ldots, n - 1$, as illustrated in Fig. A.1. This figure also shows an arbitrary point ζ_k on segment Δs_k, depicted as an elementary arc of length Δz_k. Consider, then, the summation $\Sigma_{k=1}^{n} f(\zeta_k)\Delta z_k$. The *line integral* of $f(z)$ along the path $\mathscr{C}$ is defined by the limiting value of this summation as the number n of segments is allowed to increase indefinitely in such a way that Δz_k approaches zero. That is

$$\oint_{\mathscr{C}} f(z)\,\mathrm{d}z = \lim_{n\to\infty} \sum_{k=1}^{n} f(\zeta_k)\Delta z_k. \tag{A.7}$$

In the special case when the points A and B coincide and $\mathscr{C}$ is a closed curve, the integral in Eq. (A.7) is referred to as a *contour integral* and is written as $\oint_{\mathscr{C}} f(z)\,\mathrm{d}z$. Note that, according to the notation described herein, the contour $\mathscr{C}$ is traversed in the *counterclockwise direction*.

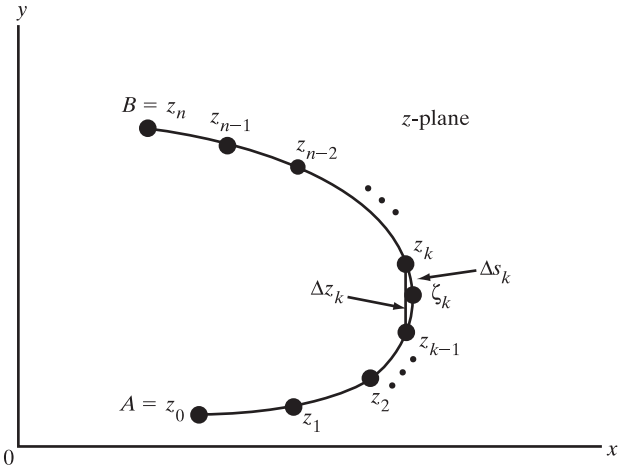


FIGURE A.1 Sectionally smooth path.

Let $f(z)$ be an analytic function in a given region $\mathcal{R}$, and let the derivative $f'(z)$ be continuous there. Then, the line integral $\oint_{\mathscr{C}} f(z)\,dz$ is independent of the path $\mathscr{C}$ that joins any pair of points in the region $\mathcal{R}$. If the path $\mathscr{C}$ is a closed curve, the value of this integral is zero. We thus have *Cauchy's integral theorem*, stated as follows:

> If a function $f(z)$ is analytic throughout a region $\mathcal{R}$, then the contour integral of $f(z)$ along any closed path $\mathscr{C}$ lying inside the region $\mathcal{R}$ is zero; that is,

$$\oint_{\mathscr{C}} f(z)dz = 0. \tag{A.8}$$

Cauchy's integral theorem is of cardinal importance in the study of analytic functions.

An important consequence of Cauchy's theorem is known as *Cauchy's integral formula*. Let $f(z)$ be analytic within and on the boundary $\mathscr{C}$ of a simple connected region. Let z_0 be any point in the interior of $\mathscr{C}$. Then, Cauchy's integral formula states that

$$f(z_0) = \frac{1}{2\pi j} \oint_{\mathscr{C}} \frac{f(z)}{z - z_0} dz, \tag{A.9}$$

where the contour integration around $\mathscr{C}$ is taken in the counterclockwise direction.

Cauchy's integral formula expresses the value of the analytic function $f(z)$ at an interior point z_0 of $\mathscr{C}$ in terms of its values on the boundary of $\mathscr{C}$. With this formula, it is a straightforward matter to express the derivative of $f(z)$ of all orders as

$$f^{(n)}(z_0) = \frac{n!}{2\pi j} \oint_{\mathscr{C}} \frac{f(z)}{(z - z_0)^{n+1}} dz, \tag{A.10}$$

where $f^{(n)}(z_0)$ is the nth derivative of $f(z)$ evaluated at $z = z_0$. Equation (A.10) is obtained by repeated differentiation of Eq. (A.9) with respect to z_0.

Cauchy's Inequality

Let the contour $\mathscr{C}$ consist of a circle of radius r and center z_0. Then, using Eq. (A.10) to evaluate the magnitude of $f^{(n)}(z_0)$, we may write

$$\begin{aligned} |f^{(n)}(z_0)| &= \frac{n!}{2\pi}\left|\oint_{\mathscr{C}} \frac{f(z)}{(z-z_0)^{n+1}}\mathrm{d}z\right| \\ &\leq \frac{n!}{2\pi}\oint_{\mathscr{C}} \frac{|f(z)|}{|z-z_0|^{n+1}}|\mathrm{d}z| \\ &\leq \frac{n!}{2\pi}\frac{M}{r^{n+1}}\oint_{\mathscr{C}}|\mathrm{d}z| \\ &= \frac{n!}{2\pi}\frac{M}{r^{n+1}}2\pi r \\ &= n!\frac{M}{r^n}, \end{aligned} \tag{A.11}$$

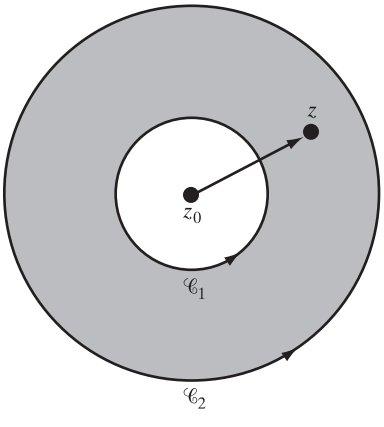
where M is the maximum value of $f(z)$ on $\mathscr{C}$. The inequality (A.11) is known as *Cauchy's inequality*.

A.3 LAURENT'S SERIES

Let the function $f(z)$ be analytic in the annular region of Fig. A.2, including the boundary of the region. The annular region consists of two concentric circles $\mathscr{C}_1$ and $\mathscr{C}_2$ whose common center is z_0. Let the point $z = z_0 + h$ be located inside the annular region, as depicted in the figure. According to *Laurent's series*,

$$f(z_0 + h) = \sum_{k=-\infty}^{\infty} a_k h^k, \tag{A.12}$$

FIGURE A.2 Annular region.



where the coefficients for varying k are given by

$$a_k = \begin{cases} \dfrac{1}{2\pi \mathrm{j}} \displaystyle\oint_{\mathscr{C}_2} \frac{f(z)\,\mathrm{d}z}{(z - z_0)^{k+1}}, & k = 0, 1, 2, \ldots \\ \dfrac{1}{2\pi \mathrm{j}} \displaystyle\oint_{\mathscr{C}_1} \frac{f(z)\,\mathrm{d}z}{(z - z_0)^{k+1}}, & k = -1, -2, \ldots \end{cases} . \tag{A.13}$$

Note that we may also express the Laurent expansion of $f(z)$ around the point z as

$$f(z) = \sum_{k=-\infty}^{\infty} a_k (z - z_0)^k. \tag{A.14}$$

When all the coefficients of negative index have the value zero, Eq. (A.14) reduces to *Taylor's series*:

$$f(z) = \sum_{k=0}^{\infty} a_k (z - z_0)^k. \tag{A.15}$$

In light of Eq. (A.10) and the first line of Eq. (A.13), we may define the coefficient

$$a_k = \frac{f^{(k)}(z_0)}{k!}, \qquad k = 0, 1, 2, \ldots . \tag{A.16}$$

Taylor's series provides the basis of Liouville's theorem, considered next.

Liouville's Theorem

Let a function $f(z)$ of the complex variable z be bounded and analytic for all values of z. Then, according to *Liouville's theorem*, $f(z)$ is simply a constant.

To prove this theorem, we first note that, since $f(z)$ is analytic everywhere inside the z-plane, we may use Taylor's series to expand $f(z)$ about the origin:

$$f(z) = \sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!} z^k. \tag{A.17}$$

The power series of Eq. (A.17) is convergent and therefore provides a valid representation of $f(z)$. Let contour $\mathscr{C}$ consist of a circle of radius r and origin as center. Then, invoking Cauchy's inequality (A.11), we may write

$$|f^{(k)}(0)| \leq \frac{k! M_c}{r^k}, \tag{A.18}$$

where M_c is the maximum value of $f(z)$ on $\mathscr{C}$. Correspondingly, the value of the kth coefficient in the power series expansion of Eq. (A.17) is bounded as

$$|a_k| = \frac{|f^{(k)}(0)|}{k!} \leq \frac{M_c}{r^k} \leq \frac{M}{r^k}, \tag{A.19}$$

where M is the bound on $|f(z)|$ for all values of z. Since, by hypothesis, M does exist, it follows from Eq. (A.19) that, for an arbitrarily large r,

$$a_k = \begin{cases} f(0), & k = 0 \\ 0, & k = 1, 2, \ldots \end{cases}. \tag{A.20}$$

Accordingly, Eq. (A.17) reduces to

$$f(z) = f(0) = \text{constant},$$

which proves Liouville's theorem.

A function $f(z)$ that is analytic for all values of z is said to be an *entire function*. Thus, Liouville's theorem may be restated as follows (Wylie & Barrett, 1982):

> An entire function, which is bounded for all values of z, is a constant.

A.4 SINGULARITIES AND RESIDUES

Let $z = z_0$ be a singular point of an analytic function $f(z)$. If the neighborhood of $z = z_0$ contains *no* other singular points of $f(z)$, the singularity at $z = z_0$ is said to be *isolated*. In the neighborhood of such a singularity, the function $f(z)$ may be represented by the Laurent series

$$\begin{aligned} f(z) &= \sum_{k=-\infty}^{\infty} a_k (z - z_0)^k \\ &= \sum_{k=0}^{\infty} a_k (z - z_0)^k + \sum_{k=-\infty}^{-1} a_k (z - z_0)^k \\ &= \sum_{k=0}^{\infty} a_k (z - z_0)^k + \sum_{k=1}^{\infty} \frac{a_{-k}}{(z - z_0)^k}. \end{aligned} \tag{A.21}$$

The particular coefficient a_{-1} in the Laurent expansion of $f(z)$ in the neighborhood of the isolated singularity at the point $z = z_0$ is called the *residue* of $f(z)$ at $z = a$. The residue plays an important role in the evaluation of integrals of analytic functions. In particular, putting $k = -1$ in Eq. (A.13), we get the following connection between the residue a_{-1} and the integral of the function $f(z)$:

$$a_{-1} = \frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}} f(z)\, \mathrm{d}z. \tag{A.22}$$

There are two nontrivial cases to be considered:

1. The Laurent expansion of $f(z)$ contains *infinitely* many terms with negative powers of $z - z_0$, as in Eq. (A.21). The point $z - z_0$ is then called an *essential singular point* of $f(z)$.

2. The Laurent expansion of $f(z)$ contains at most a *finite* number of terms m with negative powers of $z - z_0$, as given by

$$f(z) = \sum_{k=0}^{\infty} a_k (z - z_0)^k + \frac{a_{-1}}{z - z_0} + \frac{a_{-2}}{(z - z_0)^2} + \cdots + \frac{a_{-m}}{(z - z_0)^m}. \tag{A.23}$$

According to the latter representation, $f(z)$ is said to have a *pole of order m* at $z = z_0$. The *finite sum* of all the terms containing *negative powers* on the right-hand side of Eq. (A.23) is called the *principal part* of $f(z)$ at $z = z_0$.

Note that when the singularity at $z = z_0$ is a pole of order m, the residue of the pole may be determined by using the formula

$$a_{-1} = \frac{1}{(m-1)!} \frac{\mathrm{d}^{m-1}}{\mathrm{d}z^{m-1}} [(z - z_0)^m f(z)]_{z=z_0}. \tag{A.24}$$

In effect, by using this formula, we avoid the need to derive the Laurent series. For the special case when the order $m = 1$, the pole is said to be *simple*. Correspondingly, the formula of Eq. (A.24) for the residue of a simple pole reduces to

$$a_{-1} = \lim_{z \to z_0} (z - z_0) f(z). \tag{A.25}$$

A.5 CAUCHY'S RESIDUE THEOREM

Consider a closed contour $\mathscr{C}$ in the z-plane containing a number of isolated singularities of some function $f(z)$. Let $z_1, z_2, \ldots, z_n$ define the locations of these isolated singularities. Around each singular point of the function $f(z)$, we draw a circle small enough to ensure that it does not enclose the other singular points of $f(z)$, as depicted in Fig. A.3. The original contour $\mathscr{C}$, together with these small circles, constitutes the boundary of a

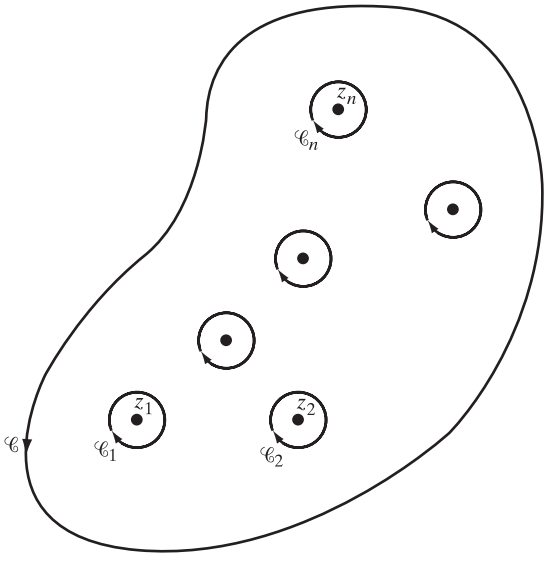


FIGURE A.3 Multiply connected region.

multiply connected region in which $f(z)$ is analytic everywhere and to which Cauchy's integral theorem may therefore be applied. Specifically, for the situation depicted in the figure, we may write

$$\frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}} f(z)\,\mathrm{d}z + \frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}_1} f(z)\,\mathrm{d}z + \cdots + \frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}_n} f(z)\,\mathrm{d}z = 0. \tag{A.26}$$

Note that the contour $\mathscr{C}$ is traversed in the *positive* sense (i.e., the counterclockwise direction), whereas the small circles are traversed in the *negative* sense (i.e., the clockwise direction).

Suppose we now *reverse* the direction along which the integral around each small circle in Fig. A.3 is taken. This operation has the effect of applying a minus sign to each of the integrals in Eq. (A.26) that involve the small circles $\mathscr{C}_1, \ldots, \mathscr{C}_n$. Accordingly, for the case when *all* the integrals around the original contour $\mathscr{C}$ and the small circles $\mathscr{C}_1, \ldots, \mathscr{C}_n$ are taken in the counterclockwise direction, we may rewrite Eq. (A.26) as

$$\frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}} f(z)\,\mathrm{d}z = \frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}_1} f(z)\,\mathrm{d}z + \cdots + \frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}_n} f(z)\,\mathrm{d}z. \tag{A.27}$$

By definition, the integrals on the right-hand side of Eq. (A.27) are the residues of the function $f(z)$, evaluated at the various isolated singularities of $f(z)$ within the contour $\mathscr{C}$. We may thus express the integral of $f(z)$ around the contour $\mathscr{C}$ simply as

$$\oint_{\mathscr{C}} f(z)\,\mathrm{d}z = 2\pi \mathrm{j} \sum_{k=1}^{n} \mathrm{Res}\,(f(z), z_k), \tag{A.28}$$

where $\mathrm{Res}\,(f(z), z_k)$ stands for the residue of the function $f(z)$ evaluated at the isolated singular point $z = z_k$. Equation (A.28) is called *Cauchy's residue theorem* and is extremely important in the theory of functions in general and in evaluating definite integrals in particular.

A.6 PRINCIPLE OF THE ARGUMENT

Consider a complex function $f(z)$, characterized as follows:

1. The function $f(z)$ is analytic in the interior of a closed contour $\mathscr{C}$ in the z-plane, except at a finite number of poles.
2. The function $f(z)$ has neither poles nor zeros on the contour $\mathscr{C}$. By a "zero," we mean a point in the z-plane at which $f(z) = 0$. In contrast, at a "pole," as defined previously, we have $f(z) = \infty$. Let N be the *number of zeros* and P be the *number of poles* of the function $f(z)$ in the interior of contour $\mathscr{C}$, where each zero or pole is counted according to its *multiplicity*.

We may then state the following theorem (Wylie & Barrett, 1982):

$$\frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}} \frac{f'(z)}{f(z)} \mathrm{d}z = N - P. \tag{A.29}$$

Here, as is usual, $f'(z)$ is the derivative of $f(z)$. We note that

$$\frac{\mathrm{d}}{\mathrm{d}z}\ln f(z) = \frac{f'(z)}{f(z)}\mathrm{d}z,$$

where ln denotes the natural logarithm. Hence,

$$\begin{aligned}\oint_{\mathscr{C}} \frac{f'(z)}{f(z)}\mathrm{d}z &= \ln f(z)\big|_{\mathscr{C}} \\ &= \ln|f(z)|_{\mathscr{C}} + \mathrm{j}\arg f(z)\big|_{\mathscr{C}},\end{aligned} \tag{A.30}$$

where $|f(z)|$ denotes the magnitude of $f(z)$ and $\arg f(z)$ denotes the argument of $f(z)$. The first term on the right-hand side of Eq. (A.30) is zero, since the logarithmic function $\ln f(z)$ is single valued and the contour $\mathscr{C}$ is closed. Hence,

$$\oint_{\mathscr{C}} \frac{f'(z)}{f(z)}\,\mathrm{d}z = \mathrm{j}\arg f(z)\big|_{\mathscr{C}}. \tag{A.31}$$

Thus, substituting Eq. (A.31) into Eq. (A.29), we get

$$N - P = \frac{1}{2\pi}\arg f(z)\big|_{\mathscr{C}}. \tag{A.32}$$

This result, which is a reformulation of the theorem described in Eq. (A.29), is called the *principle of the argument*.

For a geometric interpretation of the principle of the argument, let $\mathscr{C}$ be a closed contour in the z-plane, as in Fig. A.4(a). As z traverses the contour $\mathscr{C}$ in the counterclockwise direction, $w = f(z)$ traces out a contour $\mathscr{C}'$ of its own in the w-plane; for the purpose of illustration, $\mathscr{C}'$ is shown in Fig. A.4(b). Suppose now a line is drawn in the w-plane

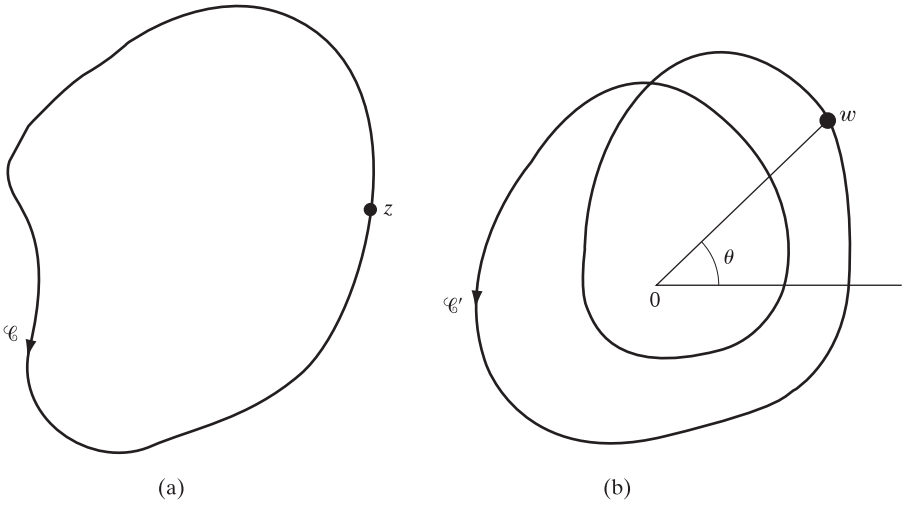


FIGURE A.4 (a) Contour $\mathscr{C}$ in the z-plane; (b) contour $\mathscr{C}'$ in the w-plane, where $w = f(z)$.

from the origin to the point $w = f(z)$, as depicted in Fig. A.4(b). Then, the angle θ that this line makes with a fixed direction (shown as the horizontal direction in the figure) is arg $f(z)$. The principle of the argument thus provides a description of the number of times the point $w = f(z)$ winds around the origin of the w-plane (i.e., the point $w = 0$) as the complex variable z traverses the contour $\mathscr{C}$ in a counterclockwise direction.

Rouché's Theorem

Let the function $f(z)$ be analytic on a closed contour $\mathscr{C}$ and in the interior of $\mathscr{C}$. Let $g(z)$ be a second function that, in addition to satisfying the same condition for analyticity as $f(z)$, fulfills the following condition on the contour $\mathscr{C}$:

$$|f(z)| > |g(z)|.$$

In other words, on the contour $\mathscr{C}$,

$$\left|\frac{g(z)}{f(z)}\right| < 1. \tag{A.33}$$

Define the function

$$F(z) = 1 + \frac{g(z)}{f(z)}, \tag{A.34}$$

which has no poles or zeros on $\mathscr{C}$. By the principle of the argument applied to $F(z)$, we have

$$N - P = \frac{1}{2\pi} \arg F(z)\big|_{\mathscr{C}}. \tag{A.35}$$

However, from Eq. (A.33), when z is on the contour $\mathscr{C}$, it follows that

$$|F(z) - 1| < 1. \tag{A.36}$$

In other words, the point $w = F(z)$ lies inside a circle with center at $w = 1$ and unit radius, as illustrated in Fig. A.5. Therefore,

$$|\arg F(z)| < \frac{\pi}{2} \qquad \text{for } z \text{ on } \mathscr{C}, \tag{A.37}$$

or, equivalently,

$$\arg F(z)\big|_{\mathscr{C}} = 0. \tag{A.38}$$

From Eq. (A.38), we infer that $N = P$, where both N and P refer to $f(z)$. From the definition of the function $F(z)$ given in Eq. (A.34), we note that the poles of $F(z)$ are the zeros of $f(z)$ and the zeros of $F(z)$ are the zeros of the sum $f(z) + g(z)$. Accordingly, the fact that $N = P$ means that $f(z) + g(z)$ and $f(z)$ have the same number of zeros. This result is known as *Rouché's theorem*, which may be formally stated as follows:

> Let $f(z)$ and $g(z)$ be analytic on a closed contour $\mathscr{C}$ and in the interior of $\mathscr{C}$. Let $|f(z)| > |g(z)|$ on $\mathscr{C}$. Then, $f(z)$ and $f(z) + g(z)$ have the same number of zeros inside contour $\mathscr{C}$.

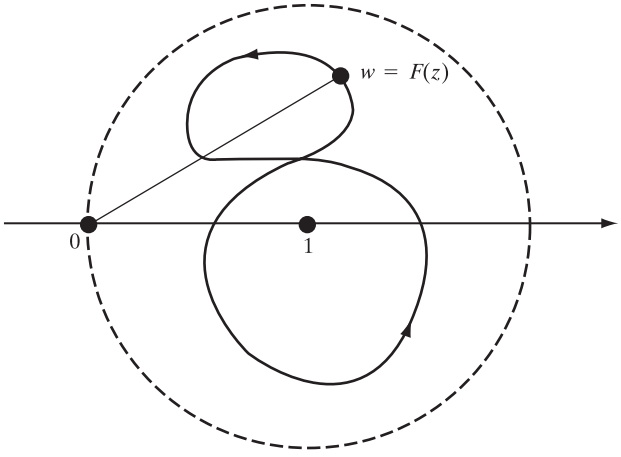


FIGURE A.5 Point $w = f(z)$ on a closed contour lying inside the unit circle.

EXAMPLE

Consider the contour depicted in Fig. A.6(a). This contour constitutes the boundary of a multiply connected region in the z-plane. Let $F(z)$ and $G(z)$ be two polynomials in z^{-1}, both of which are analytic on the contour and in the interior of it. Moreover, let $|F(z)| > |G(z)|$. Then, according to Rouché's theorem, both $F(z)$ and $F(z) + G(z)$ have the same number of zeros inside the contour described in Fig. A.6(a).

Suppose now that we let the radius R of the outside circle $\mathscr{C}$ in the figure approach infinity. Also, let the separation l between the two straight-line portions of the contour approach zero. Then, in the limit, the region enclosed by the contour will be made up of the entire area that lies *outside* the inner circle $\mathscr{C}_1$, as depicted in Fig. A.6(b). In other words, the polynomials $F(z)$ and $F(z) + G(z)$ have the same number of zeros outside the circle $\mathscr{C}_1$, under the conditions just described. [Note that the circle $\mathscr{C}_1$ is traversed in the clockwise direction (i.e., in a negative sense).]

A.7 INVERSION INTEGRAL FOR THE z-TRANSFORM

The material presented in Sections A.1 through A.6 is applicable to functions of a complex variable in general. In this section and the next, we consider the special case of a complex function defined as the z-transform of a sequence of samples taken in time.

Let $X(z)$ denote the z-transform of a sequence $x(n)$ that converges to an analytic function in the annular domain $R_1 < |z| < R_2$. By definition, $X(z)$ may be written as the Laurent series

$$X(z) = \sum_{m=-\infty}^{\infty} x(m)z^{-m}, \qquad R_1 < |z| < R_2, \tag{A.39}$$

where, for convenience of presentation, we have used m in place of n as the index of time. Let $\mathscr{C}$ be a closed contour that lies inside the *region of convergence* $R_1 < |z| < R_2$.

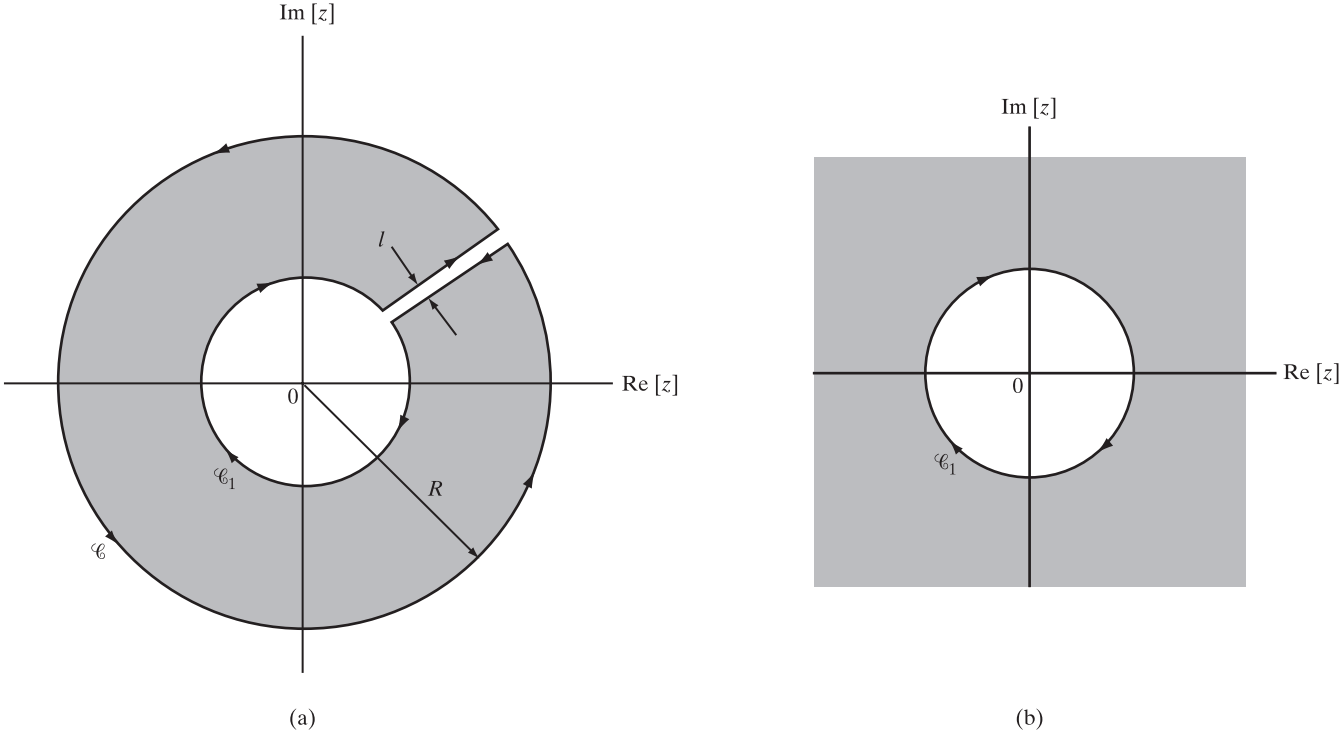


FIGURE A.6 (a) Multiply connected region; (b) limiting form of the region in (a) as $R \to \infty$ and $l \to 0$.

Then, multiplying both sides of Eq. (A.39) by z^{n-1}, integrating around the contour $\mathscr{C}$ in a counterclockwise direction, and interchanging the order of integration and summation, we get

$$\frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}} X(z) z^n \frac{\mathrm{d}z}{z} = \sum_{m=-\infty}^{\infty} x(m) \frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}} z^{n-m} \frac{\mathrm{d}z}{z}. \tag{A.40}$$

The interchange of integration and summation is justified here because the Laurent series that defines $X(z)$ converges uniformly on $\mathscr{C}$. Let

$$z = re^{\mathrm{j}\theta}, \qquad R_1 < r < R_2. \tag{A.41}$$

Then,

$$z^{n-m} = r^{n-m} e^{\mathrm{j}(n-m)\theta}$$

and

$$\frac{\mathrm{d}z}{z} = \mathrm{j}\, \mathrm{d}\theta.$$

Correspondingly, we may express the contour integral on the right-hand side of Eq. (A.40) as

$$\begin{aligned} \frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}} z^{n-m} \frac{\mathrm{d}z}{z} &= \frac{1}{2\pi} \int_0^{2\pi} r^{n-m} e^{\mathrm{j}(n-m)\theta}\, \mathrm{d}\theta \\ &= \begin{cases} 1, & m = n \\ 0, & m \neq n \end{cases}. \end{aligned} \tag{A.42}$$

Inserting Eq. (A.42) into Eq. (A.40), we get

$$x(n) = \frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}} X(z) z^n \frac{\mathrm{d}z}{z}. \tag{A.43}$$

Equation (A.43) is called the *inversion integral formula* for the z-transform.

A.8 PARSEVAL'S THEOREM

Let $X(z)$ denote the z-transform of the sequence $x(n)$ with the region of convergence $R_{1x} < |z| < R_{2x}$. Let $Y(z)$ denote the z-transform of a second sequence $y(n)$ with the region of convergence $R_{1y} < |z| < R_{2y}$. Then, *Parseval's theorem* states that

$$\sum_{n=-\infty}^{\infty} x(n) y^*(n) = \frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}} X(z) Y^*\left(\frac{1}{z^*}\right) \frac{\mathrm{d}z}{z}, \tag{A.44}$$

where $\mathscr{C}$ is a closed contour defined in the overlap of the regions of convergence of $X(z)$ and $Y(z)$, both of which are analytic, and the asterisk denotes *complex conjugation*. The function $Y^*(1/z^*)$ is obtained from the z-transform $Y(z)$ by using $1/z^*$ in place of z and then taking the complex conjugate of the resulting function. Note that $Y^*(1/z^*)$ is analytic, too.

To prove Parseval's theorem, we use the inversion integral of Eq. (A.43) to write

$$\sum_{n=-\infty}^{\infty} x(n)y^*(n) = \frac{1}{2\pi \mathrm{j}} \sum_{n=-\infty}^{\infty} y^*(n) \oint_c X(z) z^n \frac{\mathrm{d}z}{z} \tag{A.45}$$

$$= \frac{1}{2\pi \mathrm{j}} \oint_{\mathscr{C}} X(z) \sum_{n=-\infty}^{\infty} y^*(n) z^n \frac{\mathrm{d}z}{z}.$$

From the definition of the z-transform of $y(n)$, namely,

$$Y(z) = \sum_{n=-\infty}^{\infty} y(n) z^{-n},$$

we note that

$$Y^*\left(\frac{1}{z^*}\right) = \sum_{n=-\infty}^{\infty} y^*(n) z^n. \tag{A.46}$$

Hence, using Eq. (A.46) in Eq. (A.45), we get the result given in Eq. (A.44), and the proof of Parseval's theorem is completed.

APPENDIX B

Wirtinger Calculus for Computing Complex Gradients

In Chapter 2 on Wiener filters, we described a procedure for computing the complex gradient vector of a real cost function in terms of its filter coefficients. The procedure described therein was based on evaluating the partial derivative of the cost function of each filter coefficient in terms of its real and imaginary parts, treated separately. From an algebraic perspective, this procedure makes intuitive sense.

When, however, the requirement is for a simple and straightforward procedure to compute the complex gradient vector of a cost function, particularly with matrix analysis for mathematical compactness in mind, we need a new procedure that does not involve the evaluation of separate derivatives with respect to the real and imaginary parts of every filter coefficient in the cost function. To satisfy this requirement, we look to *Wirtinger calculus,* so named in honor of Wilhelm Wirtinger (1927). Compared to the procedure described in Chapter 2, Wirtinger calculus is more mathematically sophisticated.

B.1 WIRTINGER CALCULUS: SCALAR GRADIENTS

To begin the discussion, we have to recognize that a real cost function is not complex-differentiable in a complex plane. The reason is that it violates the *Cauchy–Riemann equations*—namely, Eqs. (A.5) and (A.6) derived in Appendix A on complex variable theory. Simply put, this violation is attributed to the fact that the imaginary part of the cost function is zero, by definition.

Wirtinger calculus[1] relaxes the rigid mathematical structure of the Cauchy–Riemann equations in a clever way:

> The cost function is treated as a real-differentiable function in a complex plane.

The end result is a simple, straightforward procedure for computing the complex gradient of a cost function.

[1]For a detailed treatment of Wirtinger calculus, the reader is referred to Adali and Li (2010), which includes an extended list of related references. Moreover, Adali and Li address Jacobians and Hessians (involving second-order derivatives).

The key point behind Wirtinger calculus is summed up in the following theorem (Adali & Li, 2010):

> Let f be a function of real variables, x and y, such that we may write
>
> $$f(z, z^*) = f(x, y), \tag{B.1}$$
>
> where z is a complex variable defined by
>
> $$z = x + \mathrm{j}y \tag{B.2}$$
>
> and z^* is the complex conjugate of z. The function f so defined is complex-differentiable with respect to z and z^*, which are viewed as a pair of independent constants.

The following two corollaries follow from this theorem:

1. The partial derivatives defined by

$$\frac{\partial f}{\partial z} = \frac{1}{2}\left(\frac{\partial f}{\partial x} - \mathrm{j}\frac{\partial f}{\partial y}\right) \tag{B.3}$$

and

$$\frac{\partial f}{\partial z^*} = \frac{1}{2}\left(\frac{\partial f}{\partial x} + \mathrm{j}\frac{\partial f}{\partial y}\right) \tag{B.4}$$

are both computable because z and z^* are viewed in $f(z, z^*)$ as independent constants.

2. The second corollary is in two parts:

2.1 A necessary and sufficient condition for the function f to have a stationary point in the complex plane is that

$$\frac{\partial f}{\partial z} = 0, \tag{B.5}$$

where z^* is formally treated as a constant.

2.2 Similarly,

$$\frac{\partial f}{\partial z^*} = 0 \tag{B.6}$$

is also a necessary and sufficient condition; this time, z is formally treated as a constant.

To summarize the procedure for computing the complex gradient of a real function f, we may proceed in one of two ways:

1. The given function f is expressed in the form $f(z, z^*)$, and the partial derivative $\partial f/\partial z^*$ is computed by formally treating z as a constant.

2. The partial derivative $\partial f/\partial z$ is computed by formally treating z^* as a constant.

These two partial derivatives are naturally related as follows:

$$\frac{\partial f}{\partial z^*} = \left(\frac{\partial f}{\partial z}\right)^*. \tag{B.7}$$

Henceforth, we proceed by computing $\partial f/\partial z^*$, as illustrated in the following example.

EXAMPLE 1: First-Order Predictor

Consider a first-order predictor defined by the scalar product

$$\hat{u}(n) = w^* u(n-1), \tag{B.8}$$

where w is an adjustable complex parameter and $\hat{u}(n)$ is an estimate of the current input $u(n)$ given the past input $u(n-1)$. The prediction error is defined by

$$\begin{aligned} e(n) &= u(n) - \hat{u}(n) \\ &= u(n) - w^* u(n-1). \end{aligned} \tag{B.9}$$

The mean-square value of the prediction error, representing the cost function, is defined by

$$\begin{aligned} J(w) &= \mathbb{E}[e(n)e^*(n)] \\ &= \mathbb{E}[(u(n) - w^* u(n-1))(u^*(n) - wu^*(n-1))], \end{aligned} \tag{B.10}$$

where $\mathbb{E}$ is the expectation operator. With Wirtinger calculus in mind, we redefine the symbol for the cost function in Eq. (B.10) in the new form

$$J(w, w^*) = \mathbb{E}[(u(n) - w^* u(n-1))(u^*(n) - wu^*(n-1))].$$

Hence, differentiating $J(w, w^*)$ with respect to w^* and formally treating w as a constant, we may go on to write

$$\begin{aligned} \frac{\partial J(w, w^*)}{\partial w^*} &= \frac{\partial}{\partial w^*}\mathbb{E}[(u(n) - w^* u(n-1))(u^*(n) - wu^*(n-1))] \\ &= \mathbb{E}\left[\frac{\partial}{\partial w^*}\{(u(n) - w^* u(n-1))(u^*(n) - wu^*(n-1))\}\right] \\ &= -\mathbb{E}[u(n-1)(u^*(n) - wu^*(n-1))]. \end{aligned} \tag{B.11}$$

Following the definition introduced in Eq. (2.42) for an autocorrelation function, we have

$$r(-1) = \mathbb{E}[u(n-1)u^*(n)]. \tag{B.12}$$

Moreover, assuming that the input is stationary, we also have

$$\sigma_u^2 = \mathbb{E}[|u(n-1)|^2], \tag{B.13}$$

which describes the variance of the input, assumed to have zero mean. Thus, substituting Eqs. (B.12) and (B.13) into Eq. (B.11), setting the partial derivative to zero, and solving for the optimum w, w_o, we write

$$w_o = \frac{r(-1)}{\sigma_u^2}, \tag{B.14}$$

which defines the optimum predictor of order one.

B.2 GENERALIZED WIRTINGER CALCULUS: GRADIENT VECTORS

Consider next the more general case of a cost function formulated around a *multidimensional finite-duration impulse response (FIR) filter,* which is characterized by a tap-weight vector $\mathbf{w}$. In such a scenario, we generalize Wirtinger calculus by defining the corresponding cost function as $J(\mathbf{w}, \mathbf{w}^H)$, where $\mathbf{w}^H$ is the Hermitian transpose of $\mathbf{w}$. This definition is justified because it is perfectly *consistent* with the definition of an inner product that has been used throughout the text. For a reminder, given a pair of vectors, $\mathbf{w}$ and $\mathbf{u}$, of the same dimensionality, their inner product is defined by $\mathbf{w}^H\mathbf{u}$ or $\mathbf{u}^H\mathbf{w}$. In effect, $\mathbf{w}^H$ for computing a gradient vector assumes the role of w^* for computing a scalar gradient. Except for this change, the underlying theory of Wirtinger calculus holds.

Thus, to generalize the procedure for computing the complex gradient of a cost function, we may proceed along one of two paths:

1. Differentiate $J(\mathbf{w}, \mathbf{w}^H)$ with respect to $\mathbf{w}^H$, formally treating $\mathbf{w}$ as a constant vector.
2. Differentiate $J(\mathbf{w}, \mathbf{w}^H)$ with respect to $\mathbf{w}$, formally treating $\mathbf{w}^H$ as a constant vector.

To be consistent with the choice made in Section B.1, we adopt the first path, as illustrated in the next example.

EXAMPLE 2: Wiener Filter

Referring to Chapter 2 on the Wiener filter, the generalized cost function is defined by

$$J(\mathbf{w}, \mathbf{w}^H) = \mathbb{E}\left[\underbrace{\big(d(n) - \mathbf{w}^H\mathbf{u}(n)\big)}_{\text{estimation error, } e(n)} \underbrace{\big(d^*(n) - \mathbf{u}^H(n)\mathbf{w}\big)}_{e^*(n)} \right], \tag{B.15}$$

where $\mathbf{u}(n)$ is the input vector and $d(n)$ is the desired response. Differentiating $J(\mathbf{w}, \mathbf{w}^H)$ with respect to $\mathbf{w}^H$ and formally treating $\mathbf{w}$ as a constant vector, we obtain the partial derivative

$$\begin{aligned} \frac{\partial J(\mathbf{w}, \mathbf{w}^H)}{\partial \mathbf{w}^H} &= \frac{\partial}{\partial \mathbf{w}^H}\mathbb{E}[(d(n) - \mathbf{w}^H\mathbf{u}(n))(d^*(n) - \mathbf{u}^H(n)\mathbf{w})] \\ &= \mathbb{E}\left[\frac{\partial}{\partial \mathbf{w}^H}\{(d(n) - \mathbf{w}^H\mathbf{u}(n))(d^*(n) - \mathbf{u}^H(n)\mathbf{w})\}\right] \\ &= -\mathbb{E}[\mathbf{u}(n)(d^*(n) - \mathbf{u}^H(n)\mathbf{w})]. \end{aligned} \tag{B.16}$$

At this point, we introduce the following definitions:

Correlation function of the input vector $\mathbf{u}(n)$:

$$\mathbf{R} = \mathbb{E}[\mathbf{u}(n)\mathbf{u}^H(n)], \tag{B.17}$$

and *cross-correlation vector of the input vector* $\mathbf{u}(n)$ *and desired response* $d(n)$:

$$\mathbf{p} = \mathbb{E}[\mathbf{u}(n)d^*(n)], \tag{B.18}$$

which follow from Eqs. (2.29) and (2.32), respectively.

Accordingly, we may recast Eq. (B.16) in the following form:

$$\frac{\partial J(\mathbf{w}, \mathbf{w}^{\mathrm{H}})}{\partial \mathbf{w}^{\mathrm{H}}} = -\mathbf{p} + \mathbf{R}\mathbf{w}. \tag{B.19}$$

Setting this partial gradient vector to zero and solving for the optimum tap-weight vector, $\mathbf{w}_o$, we obtain

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p}, \tag{B.20}$$

which is the very solution defined in Eq. (2.36).

The insightful observation to be made from this example is the straightforward manner in which the Wiener solution of Eq. (B.20) is derived. This trademark of Wirtinger calculus—namely, simplicity—applies equally well to the next example.

EXAMPLE 3: Log-Likelihood Function

For another example, consider the real log-likelihood function discussed in Chapter 9, reproduced here as

$$l(\mathbf{w}) = F - \frac{1}{\sigma^2}\boldsymbol{\varepsilon}^{\mathrm{H}}\boldsymbol{\varepsilon}, \tag{B.21}$$

where F is a constant, σ^2 is the variance of white noise representing measurement error in a multiple linear regression model, and the estimation error vector is defined by

$$\boldsymbol{\varepsilon} = \mathbf{d} - \mathbf{A}\mathbf{w}, \tag{B.22}$$

where $\mathbf{d}$ is the desired response vector, $\mathbf{A}$ is the data matrix, and $\mathbf{w}$ is the vector characterizing the parameterized regression model.

Following the generalized Wirtinger calculus, we introduce a new symbol for the objective function:

$$\begin{aligned} l(\mathbf{w}, \mathbf{w}^{\mathrm{H}}) &= F - \frac{1}{\sigma^2}(\mathbf{d} - \mathbf{A}\mathbf{w})^{\mathrm{H}}(\mathbf{d} - \mathbf{A}\mathbf{w}) \\ &= F - \frac{1}{\sigma^2}(\mathbf{d}^{\mathrm{H}} - \mathbf{w}^{\mathrm{H}}\mathbf{A}^{\mathrm{H}})(\mathbf{d} - \mathbf{A}\mathbf{w}). \end{aligned} \tag{B.23}$$

Differentiating $l(\mathbf{w}, \mathbf{w}^{\mathrm{H}})$ with respect to $\mathbf{w}^{\mathrm{H}}$ and formally treating $\mathbf{w}$ as a constant vector, we obtain

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}^{\mathrm{H}}} l(\mathbf{w}, \mathbf{w}^{\mathrm{H}}) &= \frac{\partial}{\partial \mathbf{w}^{\mathrm{H}}}\left[F - \frac{1}{\sigma^2}(\mathbf{d}^{\mathrm{H}} - \mathbf{w}^{\mathrm{H}}\mathbf{A}^{\mathrm{H}})(\mathbf{d} - \mathbf{A}\mathbf{w}) \right] \\ &= \frac{1}{\sigma^2}\mathbf{A}^{\mathrm{H}}(\mathbf{d} - \mathbf{A}\mathbf{w}). \end{aligned} \tag{B.24}$$

Using $\mathbf{w}_o$ again to denote the optimum $\mathbf{w}$ and, correspondingly, defining the optimum error vector, $\boldsymbol{\varepsilon}_o$, as

$$\boldsymbol{\varepsilon}_o = \mathbf{d} - \mathbf{A}\mathbf{w}_o,$$

we may rewrite Eq. (B.24) in the desired form:

$$\frac{\partial l}{\partial \mathbf{w}^{\mathrm{H}}} = \frac{1}{\sigma^2}\mathbf{A}^{\mathrm{H}}\boldsymbol{\varepsilon}_o, \tag{B.25}$$

which is exactly the result described in Eq. (9.70).

B.3 ANOTHER APPROACH TO COMPUTE GRADIENT VECTORS

In Examples 1 through 3, we proceeded along the following two steps:

1. Define the estimation error.
2. Apply the generalized Wirtinger calculus to the expected value of the estimation error multiplied by its complex conjugate; the product, in its intact form, represents the cost function.

Another way of approaching the problem is:

1. Define the cost function in its expanded form, using the expectation operator.
2. Apply the generalized Wirtinger calculus.

Typically, in this latter approach, we find inner product terms, such as $\mathbf{p}^{\mathrm{H}}\mathbf{w}$ and $\mathbf{w}^{\mathrm{H}}\mathbf{p}$, and quadratic terms, such as $\mathbf{w}^{\mathrm{H}}\mathbf{R}\mathbf{w}$, showing up in the cost function. Applying the generalized Wirtinger calculus to these terms individually yields the following useful results:

$$\frac{\partial}{\partial \mathbf{w}^{\mathrm{H}}}(\mathbf{p}^{\mathrm{H}}\mathbf{w}) = \mathbf{0}, \tag{B.26}$$

where $\mathbf{w}$ is formally treated as a constant vector;

$$\frac{\partial}{\partial \mathbf{w}^{\mathrm{H}}}(\mathbf{w}^{\mathrm{H}}\mathbf{p}) = \mathbf{p}; \tag{B.27}$$

and

$$\frac{\partial}{\partial \mathbf{w}^{\mathrm{H}}}(\mathbf{w}^{\mathrm{H}}\mathbf{R}\mathbf{w}) = \mathbf{R}\mathbf{w}, \tag{B.28}$$

where $\mathbf{w}$ is formally treated as a constant vector.

The final example illustrates the second procedure.

EXAMPLE 4: Wiener Filter Revisited

Referring to Eq. (2.50) in the chapter on Wiener filters, the cost function is defined by

$$J(\mathbf{w}) = \sigma_d^2 - \mathbf{w}^{\mathrm{H}}\mathbf{p} - \mathbf{p}^{\mathrm{H}}\mathbf{w} + \mathbf{w}^{\mathrm{H}}\mathbf{R}\mathbf{w}. \tag{B.29}$$

Following the generalized Wirtinger calculus, we write

$$J(\mathbf{w}, \mathbf{w}^{\mathrm{H}}) = \sigma_d^2 - \mathbf{w}^{\mathrm{H}}\mathbf{p} - \mathbf{p}^{\mathrm{H}}\mathbf{w} + \mathbf{w}^{\mathrm{H}}\mathbf{R}\mathbf{w}. \tag{B.30}$$

Differentiating $J(\mathbf{w}, \mathbf{w}^{\mathrm{H}})$ with respect to $\mathbf{w}^{\mathrm{H}}$ and formally treating $\mathbf{w}$ as a constant vector, we readily obtain

$$\frac{\partial}{\partial \mathbf{w}^{\mathrm{H}}} J(\mathbf{w}, \mathbf{w}^{\mathrm{H}}) = -\mathbf{p} + \mathbf{R}\mathbf{w}, \tag{B.31}$$

where we have made use of Eqs. (B.26) to (B.28). Setting Eq. (B.31) equal to zero and solving for $\mathbf{w}_o$, we obtain the same result reported in Eq. (B.20).

B.4 EXPRESSIONS FOR THE PARTIAL DERIVATIVES $\frac{\partial f}{\partial z}$ AND $\frac{\partial f}{\partial z^*}$

By definition, the complex variable $z = x + \mathrm{j}y$. We therefore have

$$x = \frac{1}{2}(z + z^*) \quad \text{and} \quad y = \frac{1}{2\mathrm{j}}(z - z^*)$$

Applying the Wirtinger Calculus, we may correspondingly write

$$\frac{\partial x}{\partial z} = \frac{1}{2} \quad \text{and} \quad \frac{\partial y}{\partial z} = \frac{-\mathrm{j}}{2}$$

for z^* treated formally as a constant. Accordingly, using the chain rule of calculus, we may go on to write

$$\begin{aligned} \frac{\partial f}{\partial z} &= \frac{\partial f}{\partial x}\frac{\partial x}{\partial z} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial z} \\ &= \frac{1}{2}\left(\frac{\partial f}{\partial x} - \mathrm{j}\frac{\partial f}{\partial y}\right) \end{aligned} \tag{B.32}$$

Similarly, we may show that

$$\frac{\partial f}{\partial z^*} = \frac{1}{2}\left(\frac{\partial f}{\partial x} + \mathrm{j}\frac{\partial f}{\partial y}\right) \tag{B.33}$$

It follows therefore for the function $f(z)$ to have a stationary point, a sufficient and necessary condition is for the two partial derivatives, $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$, to be both zero, which is intuitively satisfying.

APPENDIX C

Method of Lagrange Multipliers

Optimization consists of determining the values of some specified variables that minimize or maximize an *index of performance* or *objective function* that combines important properties of a system into a single real-valued number. The optimization may be *constrained* or *unconstrained*, depending on whether the variables are also required to satisfy side equations or not. Needless to say, the additional requirement to satisfy one or more side equations complicates the issue of constrained optimization. In this appendix, we derive the classical *method of Lagrange multipliers* for solving the *complex* version of a constrained optimization problem. The notation used in the derivation is influenced by the nature of applications that are of interest to us. We consider first the case when the problem involves a single-side equation and then the more general case of multiple-side equations.

C.1 OPTIMIZATION INVOLVING A SINGLE EQUALITY CONSTRAINT

Consider the minimization of a real-nonnegative objective function $f(\mathbf{w})$ that is a quadratic function of a parameter vector $\mathbf{w}$, subject to the *constraint*

$$\mathbf{w}^{\mathrm{H}}\mathbf{s} = g, \tag{C.1}$$

where $\mathbf{s}$ is a prescribed vector, g is a complex constant, and the superscript H denotes *Hermitian transposition* (i.e., transposition combined with complex conjugation). We may redefine the constraint by introducing a new function $c(\mathbf{w})$ that is linear in $\mathbf{w}$, as shown by

$$\begin{aligned} c(\mathbf{w}) &= \mathbf{w}^{\mathrm{H}}\mathbf{s} - g \\ &= 0 + \mathrm{j}0. \end{aligned} \tag{C.2}$$

In general, the vectors $\mathbf{w}$ and $\mathbf{s}$ and the function $c(\mathbf{w})$ are all *complex*. For example, in a beamforming application, the vector $\mathbf{w}$ represents a set of complex weights applied to the individual sensor outputs, the vector $\mathbf{s}$ represents a steering vector whose elements are defined by a prescribed "look" direction, and the function $f(\mathbf{w})$ to be minimized represents the mean-square value of the overall beamformer output. In a harmonic retrieval application, $\mathbf{w}$ represents the tap-weight vector of a finite-duration impulse response (FIR) filter, $\mathbf{s}$ represents a sinusoidal vector whose elements are determined by the angular frequency of a complex sinusoid contained in the filter input, and the function $f(\mathbf{w})$ represents the mean-square value of the filter output. In any event, assuming

that the issue is one of minimization, we may state the constrained optimization problem as follows:

$$\begin{gathered}\text{Minimize an objective function } f(\mathbf{w}),\\ \text{subject to the constraint } c(\mathbf{w}) = 0 + \mathrm{j}0.\end{gathered} \tag{C.3}$$

The *method of Lagrange multipliers* converts the problem of constrained minimization just described into one of unconstrained minimization by the introduction of *Lagrange multipliers*. First, we use the objective function $f(\mathbf{w})$ and the complex constraint function $c(\mathbf{w})$ to define a new real function

$$h(\mathbf{w}) = f(\mathbf{w}) + \lambda_1 \mathrm{Re}[c(\mathbf{w})] + \lambda_2 \mathrm{Im}[c(\mathbf{w})], \tag{C.4}$$

where λ_1 and λ_2 are *real Lagrange multipliers* and

$$c(\mathbf{w}) = \mathrm{Re}[c(\mathbf{w})] + \mathrm{j}\,\mathrm{Im}[c(\mathbf{w})]. \tag{C.5}$$

Next, we define the *complex Lagrange multiplier*:

$$\lambda = \lambda_1 + \mathrm{j}\lambda_2. \tag{C.6}$$

We may then rewrite Eq. (C.4) in the form

$$h(\mathbf{w}) = f(\mathbf{w}) + \lambda^* c(\mathbf{w}), \tag{C.7}$$

where the asterisk denotes complex conjugation and the product term $\lambda^* c(\mathbf{w})$ is real. The latter requirement is satisfied by proper choice of the constraint function $c(\mathbf{w})$ of Eq. (C.2), which is where the complex constant g comes into play.

To minimize the overall objective function $h(\mathbf{w})$ in Eq. (C.7), we apply the Wirtinger calculus described in Appendix B and then set the resulting partial derivative to zero. Specifically, differentiating $h(\mathbf{w})$ with respect to $\mathbf{w}^{\mathrm{H}}$ and treating $\mathbf{w}$ formally as a constant, we write

$$\begin{aligned}\frac{\partial}{\partial \mathbf{w}^{\mathrm{H}}} h(\mathbf{w}, \mathbf{w}^{\mathrm{H}}) &= \frac{\partial}{\partial \mathbf{w}^{\mathrm{H}}} f(\mathbf{w}) + \lambda^* \frac{\partial}{\partial \mathbf{w}^{\mathrm{H}}} c(\mathbf{w})\\ &= 0 + \mathrm{j}0.\end{aligned} \tag{C.8}$$

The partial derivative of Eq. (C.8) and the constraint function of Eq. (C.2) define the optimum solutions for the unknown weight vector $\mathbf{w}$ and the Lagrange multiplier λ.

In optimization theory, Eq. (C.3), describing minimization of the objective function $f(\mathbf{w})$ subject to the constraint $c(\mathbf{w})$, is called the *primal equation*. Correspondingly, Eq. (C.8) is called the *adjoint equation,* which is where the actual optimization is carried out.

C.2 OPTIMIZATION INVOLVING MULTIPLE EQUALITY CONSTRAINTS

Consider next the minimization of a real function $f(\mathbf{w})$ that is a quadratic function of the vector $\mathbf{w}$, subject to a set of *multiple linear constraints*

$$\mathbf{w}^{\mathrm{H}}\mathbf{s}_k = g_k, \qquad k = 1, 2, \ldots, K, \tag{C.9}$$

where the number of constraints, K, is less than the dimension of the vector $\mathbf{w}$ and the g_k are complex constants. We may state the multiple-constrained optimization problem as follows:

$$\begin{gathered}\text{Minimize a real function } f(\mathbf{w}), \text{ subject to the}\\ \text{constraints } c_k(\mathbf{w}) = 0 + \mathrm{j}0 \text{ for } k = 1, 2, \ldots, K.\end{gathered} \tag{C.10}$$

The solution to this optimization problem is readily obtained by generalizing the results of Section C.1. Specifically, we formulate a system of simultaneous equations consisting of the adjoint equation

$$\frac{\partial f}{\partial \mathbf{w}^{\mathrm{H}}} + \sum_{k=1}^{K} \frac{\partial}{\partial \mathbf{w}^{\mathrm{H}}}(\lambda_k^* c_k(\mathbf{w})) = 0 + \mathrm{j}0 \tag{C.11}$$

and the primal equation

$$c_k(\mathbf{w}) = 0 + \mathrm{j}0, \qquad k = 1, 2, \ldots, K. \tag{C.12}$$

This system of equations defines the optimum solutions for the vector $\mathbf{w}$ and the set of complex Lagrange multipliers $\lambda_1, \lambda_2, \ldots, \lambda_k$.

C.3 OPTIMUM BEAMFORMER

By way of an example, consider the problem of finding the weight vector $\mathbf{w}$ of a beamformer that minimizes the function

$$f(\mathbf{w}) = \mathbf{w}^{\mathrm{H}}\mathbf{w}, \tag{C.13}$$

subject to the constraint

$$c(\mathbf{w}) = \mathbf{w}^{\mathrm{H}}\mathbf{s} - g = 0 + \mathrm{j}0. \tag{C.14}$$

The adjoint equation for this problem is

$$\frac{\partial}{\partial \mathbf{w}^{\mathrm{H}}}(\mathbf{w}^{\mathrm{H}}\mathbf{w}) + \lambda^* \frac{\partial}{\partial \mathbf{w}^{\mathrm{H}}}(\mathbf{w}^{\mathrm{H}}\mathbf{s} - g) = 0 + \mathrm{j}0. \tag{C.15}$$

Invoking the rules of partial differentiation under Wirtinger calculus, we have

$$\frac{\partial}{\partial \mathbf{w}^{\mathrm{H}}}(\mathbf{w}^{\mathrm{H}}\mathbf{w}) = \mathbf{w}$$

and

$$\frac{\partial}{\partial \mathbf{w}^{\mathrm{H}}}(\mathbf{w}^{\mathrm{H}}\mathbf{s} - \mathrm{g}) = \mathbf{s}.$$

Substituting these two results into Eq. (C.15), we get

$$\mathbf{w} + \lambda^*\mathbf{s} = 0 + \mathrm{j}0. \tag{C.16}$$

Next, solving Eq. (C.16) for the unknown λ, we obtain

$$\begin{aligned} \lambda &= -\frac{\mathbf{w}^{\mathrm{H}}\mathbf{s}}{\mathbf{s}^{\mathrm{H}}\mathbf{s}} \\ &= -\frac{g}{\mathbf{s}^{\mathrm{H}}\mathbf{s}}. \end{aligned} \tag{C.17}$$

Finally, substituting Eq. (C.17) into Eq. (C.16) and solving for the optimum value $\mathbf{w}_o$ of the weight vector $\mathbf{w}$, we get

$$\mathbf{w}_o = \left(\frac{g^*}{\mathbf{s}^{\mathrm{H}}\mathbf{s}}\right)\mathbf{s}. \tag{C.18}$$

This solution is optimum in the sense that $\mathbf{w}_o$ satisfies the constraint of Eq. (C.14) and has the minimum Euclidean length possible.

APPENDIX D

Estimation Theory

Estimation theory is a branch of probability and statistics that deals with the problem of deriving information about properties of random variables and stochastic processes, given a set of observed samples. This problem arises frequently in the study of communications and control systems. *Maximum likelihood* is by far the most general and powerful method of estimation. This method was first used by R. A. Fisher, who is considered to be the father of modern statistics, exemplified by his pioneering work on evolutionary biology (for details of the method, see the paper by Fisher, 1922). In principle, the method of maximum likelihood may be applied to any estimation problem, with the proviso that we formulate the joint probability density function of the available set of observed data. The method then yields almost all the well-known estimates as special cases.

D.1 LIKELIHOOD FUNCTION

The method of maximum likelihood is based on a relatively simple idea (Kmenta, 1971):

> Different populations naturally generate different data samples, and any given data sample is more *likely* to have been generated from some population than from others.

Let $f_{\mathbf{U}}(\mathbf{u}|\boldsymbol{\theta})$ denote the *conditional joint probability density function* of the *random vector* $\mathbf{U}$ represented by the observed *sample* vector $\mathbf{u}$ with elements $u_1, u_2, \ldots, u_M$, where $\boldsymbol{\theta}$ is a *parameter vector* with elements $\theta_1, \theta_2, \ldots, \theta_K$. The method of maximum likelihood is based on the principle that we should estimate the parameter vector $\boldsymbol{\theta}$ by its most *plausible values*, given the observed sample vector $\mathbf{u}$. In other words, the maximum-likelihood estimators of $\theta_1, \theta_2, \ldots, \theta_k$ are those values of the parameter vector for which the conditional joint probability density function $f_{\mathbf{U}}(\mathbf{u}|\boldsymbol{\theta})$ is a maximum.

The name *likelihood function*, denoted by $l(\boldsymbol{\theta})$, is given to the conditional joint probability density function $f_{\mathbf{U}}(\mathbf{u}|\boldsymbol{\theta})$, viewed as a function of the parameter vector $\boldsymbol{\theta}$. Specifically, we write

$$l(\boldsymbol{\theta}) = f_{\mathbf{U}}(\mathbf{u}|\boldsymbol{\theta}). \tag{D.1}$$

Although the conditional joint probability density function and the likelihood function have exactly the same formula, it is important that we appreciate the physical distinction

between them. In the case of the conditional joint probability density function, the parameter vector $\boldsymbol{\theta}$ is fixed and the observation vector $\mathbf{u}$ is variable. In the case of the likelihood function, the parameter vector $\boldsymbol{\theta}$ is variable and the observation vector $\mathbf{u}$ is fixed.

In many cases, it turns out to be more convenient to work with the natural logarithm of the likelihood function rather than with the likelihood itself. Thus, using $L(\boldsymbol{\theta})$ to denote the *log-likelihood function*, we write

$$\begin{aligned} L(\boldsymbol{\theta}) &= \ln[\,l(\boldsymbol{\theta})\,] \\ &= \ln[\,f_{\mathbf{U}}(\mathbf{u}\,|\,\boldsymbol{\theta})\,]. \end{aligned} \tag{D.2}$$

The logarithmic function $L(\boldsymbol{\theta})$ is a *monotonic transformation* of $l(\boldsymbol{\theta})$. This means that whenever $l(\boldsymbol{\theta})$ decreases, its logarithm $L(\boldsymbol{\theta})$ also decreases. Since $l(\boldsymbol{\theta})$, being a formula for a conditional joint probability density function, can never become negative, it follows that there is no problem in evaluating the logarithmic function $L(\boldsymbol{\theta})$. We conclude, therefore, that the parameter vector for which the likelihood function $l(\boldsymbol{\theta})$ is a maximum is exactly the same as the parameter vector for which the log-likelihood function $L(\boldsymbol{\theta})$ is a maximum.

To obtain the ith element of the maximum-likelihood estimate of the parameter vector $\boldsymbol{\theta}$, we differentiate the log-likelihood function with respect to θ_i and set the result equal to zero. We thus get a set of first-order conditions:

$$\frac{\partial L}{\partial \theta_i} = 0, \qquad i = 1, 2, \ldots, K. \tag{D.3}$$

The first derivative of the log-likelihood function with respect to the parameter θ_i is called the *score* for that parameter. The vector of such parameters is known as the *scores vector* (i.e., the gradient vector). The scores vector is identically zero at the maximum-likelihood estimates of the parameters [i.e., at the values of $\boldsymbol{\theta}$ that result from the solutions of Eq. (D.3)].

To find how effective the method of maximum likelihood is, we can compute the *bias* and *variance* for the estimate of each parameter. However, this is frequently difficult to do. Thus, rather than approach the computation directly, we may derive a *lower bound* on the variance of any *unbiased* estimate. We say an estimate is unbiased if the average value of the estimate equals the parameter we are trying to estimate. Later, we show how the variance of the maximum-likelihood estimate compares with this lower bound.

D.2 CRAMÉR–RAO INEQUALITY

Let $\mathbf{U}$ be a random vector with conditional joint probability density function $f_{\mathbf{U}}(\mathbf{u}\,|\,\boldsymbol{\theta})$, where $\mathbf{u}$ is the observed sample vector with elements $u_1, u_2, \ldots, u_M$ and $\boldsymbol{\theta}$ is the parameter vector with elements $\theta_1, \theta_2, \ldots, \theta_K$. Using the definition of Eq. (D.2) for the log-likelihood function $L(\boldsymbol{\theta})$ in terms of the conditional joint probability density function $f_{\mathbf{U}}(\mathbf{u}\,|\,\boldsymbol{\theta})$, we form the K-by-K matrix

$$\mathbf{J} = -\begin{bmatrix} \mathbb{E}\left[\frac{\partial^2 \mathrm{L}}{\partial\theta_1^2}\right] & \mathbb{E}\left[\frac{\partial^2 L}{\partial\theta_1\partial\theta_2}\right] & \cdots & \mathbb{E}\left[\frac{\partial^2 L}{\partial\theta_1\partial\theta_K}\right] \\ \mathbb{E}\left[\frac{\partial^2 L}{\partial\theta_2\partial\theta_1}\right] & \mathbb{E}\left[\frac{\partial^2 L}{\partial\theta_2^2}\right] & \cdots & \mathbb{E}\left[\frac{\partial^2 L}{\partial\theta_2\partial\theta_K}\right] \\ \vdots & \vdots & & \vdots \\ \mathbb{E}\left[\frac{\partial^2 L}{\partial\theta_K\partial\theta_1}\right] & \mathbb{E}\left[\frac{\partial^2 L}{\partial\theta_K\partial\theta_2}\right] & \cdots & \mathbb{E}\left[\frac{\partial^2 L}{\partial\theta_K^2}\right] \end{bmatrix}. \tag{D.4}$$

The matrix **J** is called *Fisher's information matrix*.

Let **I** denote the inverse of Fisher's information matrix **J**. Let I_{ii} denote the ith diagonal element (i.e., the element in the ith row and ith column) of the inverse matrix **I**. Let $\hat{\theta}_i$ be *any* unbiased estimate of the parameter θ_i, based on the observed sample vector **u**. We may then write (Van Trees, 1968)

$$\mathrm{var}[\hat{\theta}_i] \geq I_{ii}, \qquad i = 1, 2, \ldots, K. \tag{D.5}$$

Equation (D.5) is called the *Cramér–Rao inequality*. It enables us to construct a lower limit (greater than zero) for the variance of any unbiased estimator, provided, of course, that we know the functional form of the log-likelihood function. The lower limit is called the *Cramér–Rao lower bound* (CRLB).

If we can find an unbiased estimator whose variance equals the CRLB, then, according to Eq. (D.5), there is no other unbiased estimator with a smaller variance. Such an estimator is said to be *efficient*.

D.3 PROPERTIES OF MAXIMUM-LIKELIHOOD ESTIMATORS

Not only is the method of maximum likelihood based on an intuitively appealing idea, as indicated in Section D.1, the resulting estimates also have some desirable properties. Indeed, under quite general conditions, the following *asymptotic* properties may be proved (Kmenta, 1971):

1. Maximum-likelihood estimators are *consistent*. That is, the value of θ_i for which the score $\partial L/\partial\theta_i$ is identically zero *converges in probability* to the true value of the parameter $\theta_i, i = 1, 2, \ldots, K$, as the *sample size M* approaches infinity.
2. Maximum-likelihood estimators are *asymptotically efficient*; that is,

$$\lim_{M\to\infty}\left\{\frac{\mathrm{var}[\theta_{i,\,\mathrm{ml}} - \theta_i]}{I_{ii}}\right\} = 1, \qquad i = 1, 2, \ldots, K,$$

   where $\theta_{i,\,\mathrm{ml}}$ is the maximum-likelihood estimate of parameter θ_i and I_{ii} is the ith diagonal element of the inverse of Fisher's information matrix.
3. Maximum-likelihood estimators are *asymptotically Gaussian*.

In practice, we find that the large-sample (asymptotic) properties of maximum-likelihood estimators hold rather well for sample size $M \geq 50$.

D.4 CONDITIONAL MEAN ESTIMATOR

Another classic problem in estimation theory is that of the *Bayes estimation of a random parameter*. There are different answers to this problem, depending on how the *cost function* in the Bayes estimation is formulated (Van Trees, 1968). A particular type of the Bayes estimator of interest to us in this book is the so-called *conditional mean estimator*. We now wish to do two things: (1) derive the formula for the conditional mean estimator from first principles and (2) show that such an estimator is the same as a minimum mean-square-error estimator.

Toward those ends, consider a *random parameter* x. We are given an *observation* y that depends on x, and the requirement is to estimate x. Let $\hat{x}(y)$ denote an *estimate* of the parameter x; the symbol $\hat{x}(y)$ emphasizes the fact that the estimate is a function of the observation y. Let $C(x, \hat{x}(y))$ denote a *cost function* that depends on both x and $\hat{x}(y)$. Then, according to Bayes' estimation theory, we may write the following expression for the *risk* (Van Trees, 1968):

$$\begin{aligned}\mathcal{R} &= \mathbb{E}[C(x, \hat{x}(y))] \\ &= \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} C(x, \hat{x}(y)) f_{X,Y}(x, y)\, dy.\end{aligned} \tag{D.6}$$

Here, $f_{X,Y}(x, y)$ is the joint probability density function of x and y. For a specified cost function $C(x, \hat{x}(y))$, the *Bayes estimate* is defined as the estimate $\hat{x}(y)$ that *minimizes* the risk $\mathcal{R}$.

A cost function of particular interest (and one that is very much in the spirit of the material covered in this book) is the *mean-square error*, specified as the square of the *estimation error*, which is itself defined as the difference between the actual parameter value x and the estimate $\hat{x}(y)$; that is,

$$\varepsilon = x - \hat{x}(y). \tag{D.7}$$

Correspondingly, the cost function is defined by

$$C(x, \hat{x}(y)) = C(x - \hat{x}(y)), \tag{D.8}$$

or, more simply,

$$C(\varepsilon) = \varepsilon^2. \tag{D.9}$$

Thus, the cost function varies with the estimation error ε in the manner indicated in Fig. D.1. It is assumed here that x and y are both real. Accordingly, for the situation at hand, we may rewrite Eq. (D.6) as

$$\mathcal{R}_{\text{ms}} = \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} [x - \hat{x}(y)]^2 f_{X,Y}(x, y)\, dy, \tag{D.10}$$

where the subscripts in the risk $\mathcal{R}_{\text{ms}}$ indicate the use of the mean-square error as its basis. From probability theory, we have

$$f_{X,Y}(x, y) = f_X(x|y) f_Y(y), \tag{D.11}$$

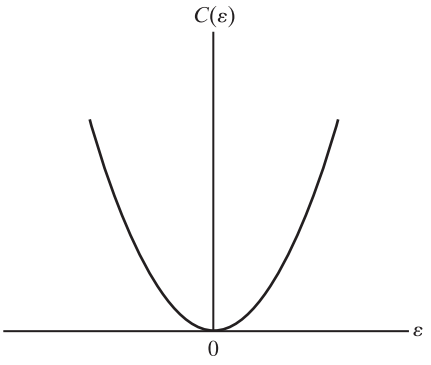


FIGURE D.1 Mean-square error as a quadratic cost function.

where $f_X(x \mid y)$ is the conditional probability density function of x, given y, and $f_Y(y)$ is the (marginal) probability density function of y. Hence, using Eq. (D.11) in Eq. (D.10), we have

$$\mathcal{R}_{\text{ms}} = \int_{-\infty}^{\infty} \mathrm{d}y f_Y(y) \int_{-\infty}^{\infty} [x - \hat{x}(y)]^2 f_X(x|y)\,\mathrm{d}x. \tag{D.12}$$

We now recognize that the inner integral and $f_Y(y)$ in Eq. (D.12) are both nonnegative. We may therefore minimize the risk $\mathcal{R}_{\text{ms}}$ simply by minimizing the inner integral. Let the estimate so obtained be denoted by $\hat{x}_{\text{ms}}(y)$. We find $\hat{x}_{\text{ms}}(y)$ by differentiating the inner integral with respect to $\hat{x}(y)$ and then setting the result equal to zero.

To simplify the presentation, let I denote the inner integral in Eq. (D.12). Then, differentiating I with respect to $\hat{x}(y)$ yields

$$\frac{\mathrm{d}I}{\mathrm{d}\hat{x}} = -2\int_{-\infty}^{\infty} x f_X(x|y)\,\mathrm{d}x + 2\hat{x}(y)\int_{-\infty}^{\infty} f_X(x|y)\,\mathrm{d}x. \tag{D.13}$$

The second integral on the right-hand side of Eq. (D.13) represents the total area under a probability density function and therefore equals unity. Hence, setting the derivative $\mathrm{d}I/\mathrm{d}\hat{x}$ equal to zero, we obtain

$$\hat{x}_{\text{ms}}(y) = \int_{-\infty}^{\infty} x f_X(x|y)\,\mathrm{d}x. \tag{D.14}$$

The solution defined by Eq. (D.14) is a unique minimum.

The estimator $\hat{x}_{\text{ms}}(y)$ defined in Eq. (D.14) is naturally a *minimum mean-square-error estimator*—hence the use of the subscript ms. For another interpretation of this estimator, we recognize that the integral on the right-hand side of the equation is just the *conditional mean* of the parameter x, given the observation y.

We therefore conclude that *the minimum mean-square-error estimator and the conditional mean estimator are indeed one and the same*. In other words, we have

$$\hat{x}_{\text{ms}}(y) = \mathbb{E}[x|y]. \tag{D.15}$$

Substituting Eq. (D.15) for the estimate $\hat{x}(y)$ in Eq. (D.12), we find that the inner integral is just the *conditional variance* of the parameter x, given y. Accordingly, the minimum value of the risk $\mathcal{R}_{\text{ms}}$ is just the average of this conditional variance over all observations y.

APPENDIX E

Eigenanalysis

In this appendix, we expand on the statistical characterization of a discrete-time stochastic process that is stationary in the wide sense. From Chapter 1, we recall that the ensemble-average correlation matrix of such a process is Hermitian. An important aspect of a Hermitian matrix is that it permits a useful decomposition of the matrix in terms of its eigenvalues and associated eigenvectors. This form of representation is commonly referred to as *eigenanalysis*, which is basic to the study of statistical signal processing.

We begin the discussion of eigenanalysis by outlining the eigenvalue problem in the context of the correlation matrix. We then study the properties of eigenvalues and eigenvectors of the correlation matrix and a related optimum filtering problem. We finish the discussion by briefly describing strategies for eigenvalue computations and related issues.

E.1 THE EIGENVALUE PROBLEM

Let the Hermitian matrix $\mathbf{R}$ denote the M-by-M correlation matrix of a wide-sense stationary discrete-time stochastic process pertaining to the M-by-1 observation vector $\mathbf{u}(n)$. In general, this matrix may contain complex elements. We wish to find an M-by-1 nonzero vector $\mathbf{q}$ that satisfies the condition

$$\mathbf{R}\mathbf{q} = \lambda\mathbf{q} \tag{E.1}$$

for some constant λ. This condition states that the vector $\mathbf{q}$ is linearly transformed to the vector $\lambda\mathbf{q}$ by the Hermitian matrix $\mathbf{R}$. Since λ is a constant, the vector $\mathbf{q}$ has special significance, in that it is left *invariant in direction* (in the M-dimensional space) by a linear transformation. For a typical M-by-M matrix $\mathbf{R}$, there will be M such vectors. To show this, we first rewrite Eq. (E.1) in the form

$$(\mathbf{R} - \lambda\mathbf{I})\mathbf{q} = \mathbf{0}, \tag{E.2}$$

where $\mathbf{I}$ is the M-by-M identity matrix and $\mathbf{0}$ is the M-by-1 null vector. The matrix $(\mathbf{R} - \lambda\mathbf{I})$ has to be singular. Hence, Eq. (E.2) has a nonzero solution in the vector $\mathbf{q}$ if and only if the determinant of the matrix $(\mathbf{R} - \lambda\mathbf{I})$ equals zero; that is,

$$\det(\mathbf{R} - \lambda\mathbf{I}) = 0. \tag{E.3}$$

This determinant, when expanded, is clearly a polynomial in λ of degree M. We thus find that, in general, Eq. (E.3) has M roots. Correspondingly, Eq. (E.2) has M solutions

in the vector $\mathbf{q}$. Equation (E.3) is called the *characteristic equation* of the matrix $\mathbf{R}$. Let $\lambda_1, \lambda_2, \ldots, \lambda_M$ denote the M roots of this equation. These roots are called the *eigenvalues* of the matrix $\mathbf{R}$. Note that, in general, the use of root finding in the characteristic equation (E.3) is a poor method for computing the eigenvalues of the matrix $\mathbf{R}$; the issue of eigenvalue computations is considered later, in Section E.5.

Let λ_i denote the ith eigenvalue of the matrix $\mathbf{R}$. Also, let $\mathbf{q}_i$ be a nonzero vector such that

$$\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i. \tag{E.4}$$

The vector $\mathbf{q}_i$ is called the *eigenvector* associated with λ_i. An eigenvector can correspond to only one eigenvalue. However, an eigenvalue may have many eigenvectors. For example, if $\mathbf{q}_i$ is an eigenvector associated with eigenvalue λ_i, then so is $a\mathbf{q}_i$ for any scalar $a \neq 0$.

Although the M-by-M matrix $\mathbf{R}$ has M eigenvalues, they need not be distinct. The matrix $\mathbf{R}$ is said to be *defective (degenerate)* if it has an eigenvalue λ_i whose *algebraic multiplicity* exceeds its *geometric multiplicity*. The algebraic multiplicity refers to the *order* of the eigenvalue λ_i, and the geometric multiplicity refers to the *dimensionality* of the associated eigenvector $\mathbf{q}_i$. A serious consequence of eigenvalue degeneracy is nondiagonalibility of the matrix $\mathbf{R}$. (The issue of diagonalization is discussed in Section E.5.)

EXAMPLE 1 Defective Matrix

The standard example of a defective matrix is

$$\mathbf{R} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

The eigenvalues of $\mathbf{R}$ are

$$\lambda_1 = \lambda_2 = 0.$$

An associated eigenvector $\mathbf{q}$ must satisfy the condition

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\mathbf{q} = 0,$$

which is satisfied by

$$\mathbf{q} = \begin{bmatrix} q_1 \\ 0 \end{bmatrix}.$$

The algebraic multiplicity of the eigenvalue $\lambda = 0$ is two, as it is a double eigenvalue. However, the geometric multiplicity of this eigenvalue is one since the associated eigenvector $\mathbf{q}$ is one-dimensional. The given matrix $\mathbf{R}$ is therefore defective.

EXAMPLE 2 White Noise

Consider the M-by-M correlation matrix of a white-noise process described by the diagonal matrix

$$\mathbf{R} = \text{diag}(\sigma^2, \sigma^2, \ldots, \sigma^2),$$

where σ^2 is the variance of a sample of the process. *This correlation matrix* $\mathbf{R}$ *has a single degenerate eigenvalue equal to the variance* σ^2 *with multiplicity M. However, any M-by-1 random vector qualifies as the associated eigenvector*, which shows that (for white noise) one eigenvalue σ^2 has M linearly independent eigenvectors. The matrix $\mathbf{R} = \sigma^2\mathbf{I}$ is therefore *not* defective.

EXAMPLE 3 Complex Sinusoid

Consider next the M-by-M correlation matrix of a time series whose elements are samples of a complex sinusoid with random phase and unit power. This correlation matrix may be written as

$$\mathbf{R} = \begin{bmatrix} 1 & e^{\mathrm{j}\omega} & \cdots & e^{\mathrm{j}(M-1)\omega} \\ e^{-\mathrm{j}\omega} & 1 & \cdots & e^{\mathrm{j}(M-2)\omega} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-\mathrm{j}(M-1)\omega} & e^{-\mathrm{j}(M-2)\omega} & \cdots & 1 \end{bmatrix},$$

where ω is the angular frequency of the complex sinusoid. The M-by-1 vector

$$\mathbf{q} = [1, e^{\mathrm{j}\omega}, \ldots, e^{\mathrm{j}(M-1)\omega}]^{\mathrm{T}},$$

where the superscript T denotes transposition, is an eigenvector of the correlation matrix $\mathbf{R}$, and the corresponding eigenvalue is M (i.e., the dimension of the matrix $\mathbf{R}$). In other words, a complex sinusoidal vector represents an eigenvector of its own correlation matrix, except for the trivial operation of complex conjugation.

Note that the correlation matrix $\mathbf{R}$ has *rank* one, which means that any column of $\mathbf{R}$ may be expressed as a linear combination of the remaining columns (i.e., the matrix $\mathbf{R}$ has only one independent column). It also means that the other eigenvalue of $\mathbf{R}$ is zero with multiplicity $M-1$, and this eigenvalue has $M-1$ linearly independent eigenvectors.

E.2 PROPERTIES OF EIGENVALUES AND EIGENVECTORS

In this section, we discuss the various properties of the eigenvalues and eigenvectors of the correlation matrix $\mathbf{R}$ of a stationary discrete-time stochastic process. Some of the properties derived here are direct consequences of the Hermitian property and the nonnegative definiteness of the correlation matrix $\mathbf{R}$, which were established in Section 1.3.

Property 1. *If* $\lambda_1, \lambda_2, \ldots, \lambda_M$ *denote the eigenvalues of the correlation matrix* $\mathbf{R}$, *then the eigenvalues of the matrix* $\mathbf{R}^k$ *equal* $\lambda_1^k, \lambda_2^k, \ldots, \lambda_M^k$ *for any integer* $k > 0$.

Repeated premultiplication of both sides of Eq. (E.1) by the matrix $\mathbf{R}$ yields

$$\mathbf{R}^k\mathbf{q} = \lambda^k\mathbf{q}. \tag{E.5}$$

This equation shows that (1) if λ is an eigenvalue of $\mathbf{R}$, then λ^k is an eigenvalue of $\mathbf{R}^k$, which is the desired result, and (2) every eigenvector of $\mathbf{R}$ is also an eigenvector of $\mathbf{R}^k$.

Property 2. *Let* $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ *be the eigenvectors corresponding, respectively, to the distinct eigenvalues* $\lambda_1, \lambda_2, \ldots, \lambda_M$ *of the M-by-M correlation matrix* $\mathbf{R}$. *Then, the eigenvectors* $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ *are linearly independent.*

We say that the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ are *linearly dependent* if there are scalars $v_1, v_2, \ldots, v_M$, not all zero, such that

$$\sum_{i=1}^{M} v_i \mathbf{q}_i = \mathbf{0}. \tag{E.6}$$

If no such scalars exist, we say that the eigenvectors are *linearly independent*.

We will establish Property 2 by contradiction. Suppose that Eq. (E.6) holds for certain scalars v_i that are not all zero. Repeated multiplication of Eq. (E.6) by matrix $\mathbf{R}$ and the use of Eq. (E.5) yield the following set of M equations:

$$\sum_{i=1}^{M} v_i \lambda_i^{k-1} \mathbf{q}_i = \mathbf{0}, \qquad k = 1, 2, \ldots, M. \tag{E.7}$$

This set of equations may be written in the form of the single matrix equation

$$[v_1 \mathbf{q}_1, v_2 \mathbf{q}_2, \ldots, v_M \mathbf{q}_M]\mathbf{S} = \mathbf{0}, \tag{E.8}$$

where

$$\mathbf{S} = \begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^{M-1} \\ 1 & \lambda_2 & \lambda_2^2 & \cdots & \lambda_2^{M-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_M & \lambda_M^2 & \cdots & \lambda_M^{M-1} \end{bmatrix}. \tag{E.9}$$

The matrix $\mathbf{S}$ is called a *Vandermonde matrix* (Strang, 1980). When the λ_i are distinct, the Vandermonde matrix $\mathbf{S}$ is nonsingular. Therefore, we may postmultiply Eq. (E.8) by the inverse matrix $\mathbf{S}^{-1}$, obtaining

$$[v_1 \mathbf{q}_1, v_2 \mathbf{q}_2, \ldots, v_M \mathbf{q}_M] = \mathbf{O}.$$

Hence, each column $v_i \mathbf{q}_i = \mathbf{0}$. Since the eigenvectors $\mathbf{q}_i$ are not zero, this condition can be satisfied if and only if the v_i are all zero, which contradicts the assumption that the scalars v_i are not all zero. In other words, the eigenvectors are linearly independent.

We may put Property 2 to an important use by having the linearly independent eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ serve as a *basis* for the representation of an arbitrary vector $\mathbf{w}$ with the same dimension as the eigenvectors themselves. In particular, we may express the arbitrary vector $\mathbf{w}$ as a linear combination of the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ as

$$\mathbf{w} = \sum_{i=1}^{M} v_i \mathbf{q}_i, \tag{E.10}$$

where $v_1, v_2, \ldots, v_M$ are constants. Suppose now we apply a linear transformation to the vector $\mathbf{w}$ by premultiplying it by the matrix $\mathbf{R}$, obtaining

$$\mathbf{R}\mathbf{w} = \sum_{i=1}^{M} v_i \mathbf{R}\mathbf{q}_i. \tag{E.11}$$

By definition, $\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i$. Therefore, we may express the result of this linear transformation in the equivalent form

$$\mathbf{R}\mathbf{w} = \sum_{i=1}^{M} v_i \lambda_i \mathbf{q}_i. \tag{E.12}$$

We thus see that when a linear transformation is applied to an arbitrary vector $\mathbf{w}$ defined in Eq. (E.10), the eigenvectors remain independent of each other, and the effect of the transformation is simply to multiply each eigenvector by its respective eigenvalue.

Property 3. *Let $\lambda_1, \lambda_2, \ldots, \lambda_M$ be the eigenvalues of the M-by-M correlation matrix $\mathbf{R}$. Then, all these eigenvalues are real and nonnegative.*

To prove this property, we first use Eq. (E.1) to express the condition on the ith eigenvalue λ_i as

$$\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i, \qquad i = 1, 2, \ldots, M. \tag{E.13}$$

Premultiplying both sides of this equation by $\mathbf{q}_i^{\mathrm{H}}$, the Hermitian transpose of eigenvector $\mathbf{q}_i$, we get

$$\mathbf{q}_i^{\mathrm{H}}\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i^{\mathrm{H}}\mathbf{q}_i, \qquad i = 1, 2, \ldots, M. \tag{E.14}$$

The inner product $\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_i$ is a positive scalar, representing the squared Euclidean length of the eigenvector $\mathbf{q}_i$; that is, $\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_i > 0$. We may therefore divide both sides of Eq. (E.14) by $\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_i$ and so express the ith eigenvalue as

$$\lambda_i = \frac{\mathbf{q}_i^{\mathrm{H}}\mathbf{R}\mathbf{q}_i}{\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_i}, \qquad i = 1, 2, \ldots, M. \tag{E.15}$$

Since the correlation matrix $\mathbf{R}$ is always nonnegative definite, the Hermitian form $\mathbf{q}_i^{\mathrm{H}}\mathbf{R}\mathbf{q}_i$ in the numerator of this ratio is always real and nonnegative; that is $\mathbf{q}_i^{\mathrm{H}}\mathbf{R}\mathbf{q}_i \geq 0$. Therefore, it follows from Eq. (E.15) that $\lambda_i \geq 0$ for all i. That is, all the eigenvalues of the correlation matrix $\mathbf{R}$ are always real and nonnegative.

The correlation matrix $\mathbf{R}$ is positive definite, except in noise-free sinusoidal and noise-free array signal-processing problems, which are rare events, so we usually have $\mathbf{q}_i^{\mathrm{H}}\mathbf{R}\mathbf{q}_i > 0$ and, correspondingly, $\lambda_i > 0$ for all i. That is, the eigenvalues of the correlation matrix $\mathbf{R}$ are almost always real and positive.

The ratio of the Hermitian form $\mathbf{q}_i^{\mathrm{H}}\mathbf{R}\mathbf{q}_i$ to the inner product $\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_i$ on the right-hand side of Eq. (E.15) is called the *Rayleigh quotient* of the vector $\mathbf{q}_i$. We may thus state that an eigenvalue of the correlation matrix equals the Rayleigh quotient of the corresponding eigenvector.

Property 4. *Let $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ be the eigenvectors corresponding, respectively, to the distinct eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$ of the M-by-M correlation matrix $\mathbf{R}$. Then, the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ are orthogonal to each other.*

Let $\mathbf{q}_i$ and $\mathbf{q}_j$ denote any two eigenvectors of the correlation matrix $\mathbf{R}$. We say that these two eigenvectors are *orthogonal* to each other if

$$\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_j = 0, \qquad i \neq j. \tag{E.16}$$

Using Eq. (E.1), we may express the conditions on the eigenvectors $\mathbf{q}_i$ and $\mathbf{q}_j$ as

$$\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i \tag{E.17}$$

and

$$\mathbf{R}\mathbf{q}_j = \lambda_j \mathbf{q}_j. \tag{E.18}$$

Premultiplying both sides of Eq. (E.17) by the Hermitian-transposed vector $\mathbf{q}_j^{\mathrm{H}}$, we get

$$\mathbf{q}_j^{\mathrm{H}}\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_j^{\mathrm{H}}\mathbf{q}_i. \tag{E.19}$$

Since the correlation matrix $\mathbf{R}$ is Hermitian, $\mathbf{R}^{\mathrm{H}} = \mathbf{R}$. Also, from Property 3, we know that the eigenvalue λ_j is real for all j. Hence, taking the Hermitian transpose of both sides of Eq. (E.18), we get

$$\mathbf{q}_j^{\mathrm{H}}\mathbf{R} = \lambda_j \mathbf{q}_j^{\mathrm{H}}. \tag{E.20}$$

Postmultiplying both sides of Eq. (E.20) by the vector $\mathbf{q}_i$ yields

$$\mathbf{q}_j^{\mathrm{H}}\mathbf{R}\mathbf{q}_i = \lambda_j \mathbf{q}_j^{\mathrm{H}}\mathbf{q}_i. \tag{E.21}$$

Subtracting Eq. (E.21) from Eq. (E.19), we obtain

$$(\lambda_i - \lambda_j)\mathbf{q}_j^{\mathrm{H}}\mathbf{q}_i = 0. \tag{E.22}$$

Since the eigenvalues of the correlation matrix $\mathbf{R}$ are assumed to be distinct, we have $\lambda_i \neq \lambda_j$. Accordingly, the condition of Eq. (E.22) holds if and only if

$$\mathbf{q}_j^{\mathrm{H}}\mathbf{q}_i = 0, \qquad i \neq j, \tag{E.23}$$

which is the desired result. That is, the eigenvectors $\mathbf{q}_i$ and $\mathbf{q}_j$ are *orthogonal* to each other for $i \neq j$.

Property 5: Unitary Similarity Transformation. *Let $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ be the eigenvectors corresponding, respectively, to the distinct eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$ of the M-by-M correlation matrix $\mathbf{R}$. Define the M-by-M matrix*

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M],$$

where

$$\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}.$$

Define the M-by-M diagonal matrix

$$\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_M).$$

Then, the correlation matrix $\mathbf{R}$ may be diagonalized as follows:

$$\mathbf{Q}^{\mathrm{H}}\mathbf{R}\mathbf{Q} = \mathbf{\Lambda}.$$

The condition that $\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_i = 1$ for $i = 1, 2, \ldots, M$ requires that each eigenvector be *normalized* to have a *length* of unity. The *squared length*, or *squared norm*, of a vector $\mathbf{q}_i$ is defined as the inner product $\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_i$. The orthogonality condition that $\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_j = 0$ for $i \neq j$ follows from Property 4. When both of these conditions are simultaneously satisfied, that is, when

$$\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}, \tag{E.24}$$

we say that the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ form an *orthonormal* set. By definition, the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ satisfy the equations [see Eq. (E.1)]

$$\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i, \qquad i = 1, 2, \ldots, M. \tag{E.25}$$

The M-by-M matrix $\mathbf{Q}$ has as its columns the orthonormal set of eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$; that is,

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M]. \tag{E.26}$$

The M-by-M diagonal matrix $\mathbf{\Lambda}$ has the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$ for the elements of its main diagonal:

$$\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_M). \tag{E.27}$$

Accordingly, we may rewrite the set of M equations (E.25) as the single matrix equation

$$\mathbf{RQ} = \mathbf{Q\Lambda}. \tag{E.28}$$

Owing to the orthonormal nature of the eigenvectors, as defined in Eq. (E.24), we have

$$\mathbf{Q}^{\mathrm{H}}\mathbf{Q} = \mathbf{I}.$$

Equivalently, we may write

$$\mathbf{Q}^{-1} = \mathbf{Q}^{\mathrm{H}}. \tag{E.29}$$

That is, the matrix $\mathbf{Q}$ is nonsingular, with an inverse $\mathbf{Q}^{-1}$ equal to the Hermitian transpose of $\mathbf{Q}$. A matrix that has this property is called a *unitary matrix*.

Thus, premultiplying both sides of Eq. (E.28) by the Hermitian-transposed matrix $\mathbf{Q}^{\mathrm{H}}$ and using the property of Eq. (E.29), we get the desired result:

$$\mathbf{Q}^{\mathrm{H}}\mathbf{RQ} = \mathbf{\Lambda}. \tag{E.30}$$

This transformation is called the *unitary similarity transformation*.

We have thus proved an important result: The correlation matrix $\mathbf{R}$ may be *diagonalized* by a unitary similarity transformation. Furthermore, the matrix $\mathbf{Q}$ that is used to diagonalize $\mathbf{R}$ has as its columns an orthonormal set of eigenvectors associated with $\mathbf{R}$. The resulting diagonal matrix $\mathbf{\Lambda}$ has as its diagonal elements the eigenvalues of $\mathbf{R}$.

By postmultiplying both sides of Eq. (E.28) by the inverse matrix $\mathbf{Q}^{-1}$ and then using the property of Eq. (E.29), we may also write

$$\begin{aligned}\mathbf{R} &= \mathbf{Q\Lambda Q}^{\mathrm{H}} \\ &= \sum_{i=1}^{M} \lambda_i \mathbf{q}_i \mathbf{q}_i^{\mathrm{H}},\end{aligned} \tag{E.31}$$

where M is the dimension of matrix $\mathbf{R}$. Let the projection $\mathbf{P}_i$ denote the outer product $\mathbf{q}_i\mathbf{q}_i^{\mathrm{H}}$. Then, it is a straightforward matter to show that

$$\mathbf{P}_i = \mathbf{P}_i^2 = \mathbf{P}_i^{\mathrm{H}},$$

which, in effect, means that $\mathbf{P}_i = \mathbf{q}_i\mathbf{q}_i^{\mathrm{H}}$ is a *rank-one projection*. Thus, Eq. (E.31) states that the correlation matrix of a wide-sense stationary process equals the linear combination of all such rank-one projections, each being weighted by its respective eigenvalue. This result is known as *Mercer's theorem*; it is also referred to as the *spectral theorem*.

Property 6. *Let $\lambda_1, \lambda_2, \ldots, \lambda_M$ be the eigenvalues of the M-by-M correlation matrix* $\mathbf{R}$. *Then, the sum of these eigenvalues equals the trace of matrix* $\mathbf{R}$.

The *trace* of a square matrix is defined as the sum of the diagonal elements of the matrix. Taking the trace of both sides of Eq. (E.30), we may write

$$\text{tr}[\mathbf{Q}^{\text{H}}\mathbf{R}\mathbf{Q}] = \text{tr}[\mathbf{\Lambda}], \tag{E.32}$$

where tr denotes the trace operator. The diagonal matrix $\mathbf{\Lambda}$ has as its diagonal elements the eigenvalues of $\mathbf{R}$. Hence, we have

$$\text{tr}[\mathbf{\Lambda}] = \sum_{i=1}^{M} \lambda_i. \tag{E.33}$$

Using a rule in matrix algebra,[1] we may write

$$\text{tr}[\mathbf{Q}^{\text{H}}\mathbf{R}\mathbf{Q}] = \text{tr}[\mathbf{R}\mathbf{Q}\mathbf{Q}^{\text{H}}].$$

However, $\mathbf{Q}\mathbf{Q}^{\text{H}}$ equals the identity matrix $\mathbf{I}$. Hence, we have

$$\text{tr}[\mathbf{Q}^{\text{H}}\mathbf{R}\mathbf{Q}] = \text{tr}[\mathbf{R}].$$

Accordingly, we may rewrite Eq. (E.32) as

$$\text{tr}[\mathbf{R}] = \sum_{i=1}^{M} \lambda_i. \tag{E.34}$$

We have thus shown that the trace of the correlation matrix $\mathbf{R}$ equals the sum of the eigenvalues of $\mathbf{R}$. Although, in proving this result, we used a property that requires the matrix $\mathbf{R}$ to be Hermitian with distinct eigenvalues, the result applies to any square matrix.

Property 7. *The correlation matrix* $\mathbf{R}$ *is ill conditioned if the ratio of the largest eigenvalue to the smallest eigenvalue of* $\mathbf{R}$ *is large.*

To appreciate the impact of Property 7, it is important that we recognize the fact that the development of an algorithm for the effective solution of a signal-processing problem and the understanding of associated *perturbation theory* go hand in hand (Van Loan, 1989). We may illustrate the synergism between these two fields by considering the linear system of equations

$$\mathbf{A}\mathbf{w} = \mathbf{d},$$

where the matrix $\mathbf{A}$ and the vector $\mathbf{d}$ are data-dependent quantities and $\mathbf{w}$ is a coefficient vector characterizing a finite-duration impulse response (FIR) filter of interest. An elementary formulation of perturbation theory tells us that if the matrix $\mathbf{A}$ and vector $\mathbf{d}$ are perturbed by small amounts, $\delta\mathbf{A}$ and $\delta\mathbf{d}$, respectively, and if $\|\delta\mathbf{A}\|/\|\mathbf{A}\|$ and $\|\delta\mathbf{d}\|/\|\mathbf{d}\|$ are both on the order of some $\varepsilon \ll 1$, then we have (Golub & Van Loan, 1996)

$$\frac{\delta\|\mathbf{w}\|}{\|\mathbf{w}\|} \leq \varepsilon\, \chi(\mathbf{A}),$$

[1] In matrix algebra, we have the following rule: Let $\mathbf{A}$ be an M-by-N matrix and $\mathbf{B}$ be an N-by-M matrix. Then, the trace of the matrix product $\mathbf{AB}$ equals the trace of $\mathbf{BA}$.

where $\delta\mathbf{w}$ is the change produced in $\mathbf{w}$ due to $\delta\mathbf{A}$ and $\delta\mathbf{d}$ and where $\chi(\mathbf{A})$ is the *condition number* of matrix $\mathbf{A}$ with respect to inversion. The condition number is so called because it describes the ill condition or bad behavior of matrix $\mathbf{A}$ quantitatively. In particular, it is defined as (Wilkinson, 1963; Strang, 1980; Golub & Van Loan, 1996)

$$\chi(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|, \tag{E.35}$$

where $\|\mathbf{A}\|$ is a *norm* of matrix $\mathbf{A}$ and $\|\mathbf{A}^{-1}\|$ is the corresponding norm of the inverse matrix $\mathbf{A}^{-1}$. The norm of a matrix is a number assigned to the matrix that is in some sense a measure of the magnitude of the matrix. We find it natural to require that the norm of a matrix satisfy the following conditions:

1. $\|\mathbf{A}\| \geq 0$, with equality if and only if $\mathbf{A} = \mathbf{O}$.
2. $\|c\mathbf{A}\| = |c|\|\mathbf{A}\|$, where c is any real number and $|c|$ is its magnitude.
3. $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$.
4. $\|\mathbf{AB}\| \leq \|\mathbf{A}\|\|\mathbf{B}\|$.

Condition 3 is the *triangle inequality*, and condition 4 indicates *mutual consistency*. There are several ways of defining the norm $\|\mathbf{A}\|$ that satisfy the preceding conditions (Ralston, 1965). For our present discussion, however, we find it convenient to use the *spectral norm*,[2] defined as the square root of the largest eigenvalue of the matrix product $\mathbf{A}^{\mathrm{H}}\mathbf{A}$, where $\mathbf{A}^{\mathrm{H}}$ is the Hermitian transpose of $\mathbf{A}$; that is,

$$\|\mathbf{A}\|_{\mathrm{s}} = (\text{largest eigenvalue of } \mathbf{A}^{\mathrm{H}}\mathbf{A})^{1/2}. \tag{E.36}$$

Since, for any matrix $\mathbf{A}$, the product $\mathbf{A}^{\mathrm{H}}\mathbf{A}$ is always Hermitian and nonnegative definite, it follows that the eigenvalues of $\mathbf{A}^{\mathrm{H}}\mathbf{A}$ are all real and nonnegative, as required. Moreover, from Eq. (E.15), we note that an eigenvalue of $\mathbf{A}^{\mathrm{H}}\mathbf{A}$ equals the Rayleigh coefficient of the corresponding eigenvector. Squaring both sides of Eq. (E.36) and using this property, we may therefore write[3]

$$\begin{aligned}\|\mathbf{A}\|_{\mathrm{s}}^2 &= \max \frac{\mathbf{x}^{\mathrm{H}}\mathbf{A}^{\mathrm{H}}\mathbf{A}\mathbf{x}}{\mathbf{x}^{\mathrm{H}}\mathbf{x}} \\ &= \max \frac{\|\mathbf{A}\mathbf{x}\|^2}{\|\mathbf{x}\|^2},\end{aligned}$$

where the denominator $\|\mathbf{x}\|^2$ is the squared Euclidean norm, or squared length, of vector $\mathbf{x}$, and likewise for the numerator $\|\mathbf{A}\mathbf{x}\|^2$. We may thus express the spectral norm of matrix $\mathbf{A}$ in the equivalent form

$$\|\mathbf{A}\|_{\mathrm{s}} = \max \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|}. \tag{E.37}$$

[2]Another matrix norm of interest is the *Frobenius norm*, defined by (Stewart, 1973)

$$\|\mathbf{A}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}|a_{ij}|^2},$$

where M and N are the dimensions of matrix $\mathbf{A}$ and a_{ij} is the ijth element of $\mathbf{A}$.

[3]Note that the vector $\mathbf{x}$ is one of the eigenvectors of $\mathbf{A}^{\mathrm{H}}\mathbf{A}$. Hence, at this stage, we can only say that $\|\mathbf{A}\|_s^2$ is the *maximum Rayleigh quotient* of the eigenvectors. However, Property 7 may be extended to any vector after the minimax theorem is proved. (See Property 9.)

According to this relation, the spectral norm of $\mathbf{A}$ measures the largest amount by which any vector (eigenvector or not) is amplified by matrix multiplication, and the vector that is amplified the most is the eigenvector that corresponds to the largest eigenvalue of $\mathbf{A}^{\mathrm{H}}\mathbf{A}$ (Strang, 1980).

Consider now the application of the definition in Eq. (E.36) to the correlation matrix $\mathbf{R}$. Since $\mathbf{R}$ is Hermitian, we have $\mathbf{R}^{\mathrm{H}} = \mathbf{R}$. Hence, from Property 1, we infer that if $\lambda_{\max}$ is the largest eigenvalue of $\mathbf{R}$, the largest eigenvalue of $\mathbf{R}^{\mathrm{H}}\mathbf{R}$ equals $\lambda_{\max}^2$. Accordingly, the spectral norm of the correlation matrix $\mathbf{R}$ is

$$\|\mathbf{R}\|_{\mathrm{s}} = \lambda_{\max}. \tag{E.38}$$

Similarly, we may show that the spectral norm of $\mathbf{R}^{-1}$, the inverse of the correlation matrix, is

$$\|\mathbf{R}^{-1}\|_{\mathrm{s}} = \frac{1}{\lambda_{\min}}, \tag{E.39}$$

where $\lambda_{\min}$ is the smallest eigenvalue of $\mathbf{R}$. Thus, by adopting the spectral norm as the basis of the condition number, we have shown that the condition number of the correlation matrix $\mathbf{R}$ is

$$\chi(\mathbf{R}) = \frac{\lambda_{\max}}{\lambda_{\min}}. \tag{E.40}$$

This ratio is commonly referred to as the *eigenvalue spread*, or the *eigenvalue ratio*, of the correlation matrix. Note that we always have $\chi(\mathbf{R}) \geq 1$.

Suppose that the correlation matrix $\mathbf{R}$ is *normalized* so that the magnitude of the largest element, $r(0)$, equals unity. Then, if the condition number or eigenvalue spread of the correlation matrix $\mathbf{R}$ is large, the inverse matrix $\mathbf{R}^{-1}$ contains some very large elements. This behavior may cause trouble in solving a system of equations involving $\mathbf{R}^{-1}$. In such a case, we say that the correlation matrix $\mathbf{R}$ is *ill conditioned*—hence the justification of Property 7.

Property 8. *The eigenvalues of the correlation matrix of a discrete-time stochastic process are bounded by the minimum and maximum values of the power spectral density of the process.*

Let λ_i, $\mathbf{q}_i$, and $i = 1, 2, \ldots, M$, denote the eigenvalues of the M-by-M correlation matrix $\mathbf{R}$ of a discrete-time stochastic process $u(n)$ and their associated eigenvectors, respectively. Reproducing Eq. (E.15) for convenience of presentation, we have

$$\lambda_i = \frac{\mathbf{q}_i^{\mathrm{H}}\mathbf{R}\mathbf{q}_i}{\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_i}, \qquad i = 1, 2, \ldots, M. \tag{E.41}$$

The Hermitian form in the numerator of Eq. (E.41) may be expressed in its expanded form as

$$\mathbf{q}_i^{\mathrm{H}}\mathbf{R}\mathbf{q}_i = \sum_{k=1}^{M}\sum_{l=1}^{M} q_{ik}^{*} r(l-k) q_{il}, \tag{E.42}$$

where the asterisk denotes *complex conjugation*, q_{ik}^* is the kth element of the row vector $\mathbf{q}_i^{\mathrm{H}}$, $r(l-k)$ is the klth element of the matrix $\mathbf{R}$, and q_{il} is the lth element of the column vector $\mathbf{q}_i$. Using the Einstein–Wiener–Khintchine relation of Eq. (1.116), we may write

$$r(l-k) = \frac{1}{2\pi}\int_{-\pi}^{\pi} S(\omega)e^{\mathrm{j}\omega(l-k)}\,\mathrm{d}\omega, \tag{E.43}$$

where $S(\omega)$ is the power spectral density of the process $u(n)$. Hence, we may rewrite Eq. (E.42) as

$$\begin{aligned}\mathbf{q}_i^{\mathrm{H}}\mathbf{R}\mathbf{q}_i &= \frac{1}{2\pi}\sum_{k=1}^{M}\sum_{l=1}^{M} q_{ik}^* q_{il}\int_{-\pi}^{\pi} S(\omega)e^{\mathrm{j}\omega(l-k)}\,\mathrm{d}\omega \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi}\mathrm{d}\omega S(\omega)\sum_{k=1}^{M} q_{ik}^* e^{-\mathrm{j}\omega k}\sum_{l=1}^{M} q_{il}e^{\mathrm{j}\omega l}.\end{aligned} \tag{E.44}$$

Let the discrete-time Fourier transform of the sequence $q_{i1}^*, q_{i2}^*, \ldots, q_{iM}^*$ be denoted by

$$Q_i'(e^{\mathrm{j}\omega}) = \sum_{k=1}^{M} q_{ik}^* e^{-\mathrm{j}\omega k}. \tag{E.45}$$

Then, using Eq. (E.45) in Eq. (E.44), we get

$$\mathbf{q}_i^{\mathrm{H}}\mathbf{R}\mathbf{q}_i = \frac{1}{2\pi}\int_{-\pi}^{\pi} |\,Q_i'(e^{\mathrm{j}\omega})\,|^2 S(\omega)\,\mathrm{d}\omega. \tag{E.46}$$

Similarly, we may show that

$$\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_i = \frac{1}{2\pi}\int_{-\pi}^{\pi} |\,Q_i'(e^{\mathrm{j}\omega})\,|^2\,\mathrm{d}\omega. \tag{E.47}$$

Accordingly, we may use Eq. (E.41) to redefine the eigenvalue λ_i of the correlation matrix $\mathbf{R}$ in terms of the associated power spectral density as

$$\lambda_i = \frac{\displaystyle\int_{-\pi}^{\pi} |\,Q_i'(e^{\mathrm{j}\omega})\,|^2 S(\omega)\,\mathrm{d}\omega}{\displaystyle\int_{-\pi}^{\pi} |\,\mathrm{Q_i'(e^{j\omega})}\,|^2\,\mathrm{d}\omega}. \tag{E.48}$$

Let $S_{\min}$ and $S_{\max}$ denote the absolute minimum and maximum values, respectively, of the power spectral density $S(\omega)$. Then,

$$\int_{-\pi}^{\pi} |\,Q_i'(e^{\mathrm{j}\omega})\,|^2 S(\omega)\,\mathrm{d}\omega \geq S_{\min}\int_{-\pi}^{\pi} |\,Q_i'(e^{\mathrm{j}\omega})\,|^2\,\mathrm{d}\omega \tag{E.49}$$

and

$$\int_{-\pi}^{\pi} |\,Q_i'(e^{\mathrm{j}\omega})\,|^2 S(\omega)\,\mathrm{d}\omega \leq S_{\max}\int_{-\pi}^{\pi} |\,Q_i'(e^{\mathrm{j}\omega})\,|^2\,\mathrm{d}\omega. \tag{E.50}$$

Hence, using Eq. (E.48), we deduce that the eigenvalues λ_i are bounded by the maximum and minimum values of the associated power spectral density as follows:

$$S_{\min} \leq \lambda_i \leq S_{\max}, \qquad i = 1, 2, \ldots, M. \tag{E.51}$$

Correspondingly, the eigenvalue spread $\chi(\mathbf{R})$ is bounded as

$$\chi(\mathbf{R}) = \frac{\lambda_{\max}}{\lambda_{\min}} \leq \frac{S_{\max}}{S_{\min}}. \tag{E.52}$$

It is of interest to note that as the dimension M of the correlation matrix approaches infinity, the maximum eigenvalue $\lambda_{\max}$ approaches $S_{\max}$, and the minimum eigenvalue $\lambda_{\min}$ approaches $S_{\min}$. Accordingly, the eigenvalue spread $\lambda(\mathbf{R})$ of the correlation matrix $\mathbf{R}$ approaches the ratio $S_{\max}/S_{\min}$ as the dimension M of the matrix $\mathbf{R}$ approaches infinity.

Property 9: Minimax Theorem. *Let the M-by-M correlation matrix* $\mathbf{R}$ *have eigenvalues* $\lambda_1, \lambda_2, \ldots, \lambda_M$ *that are arranged in decreasing order as follows*:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M.$$

The minimax theorem states that

$$\lambda_k = \min_{\dim(\mathcal{S})=k} \max_{\substack{\mathbf{x}\in\mathcal{S} \\ \mathbf{x}\neq\mathbf{0}}} \frac{\mathbf{x}^{\mathrm{H}}\mathbf{R}\mathbf{x}}{\mathbf{x}^{\mathrm{H}}\mathbf{x}}, \qquad k = 1, 2, \ldots M, \tag{E.53}$$

where $\mathcal{S}$ *is a subspace of the vector space of all M-by-1 complex vectors, dim* $(\mathcal{S})$ *denotes the dimension of subspace* $\mathcal{S}$, *and* $\mathbf{x} \in \mathcal{S}$ *signifies that the vector* $\mathbf{x}$ *(assumed to be nonzero) varies over the subspace* $\mathcal{S}$.

Let $\mathbb{C}^M$ denote a complex vector space of dimension M. For the purpose of our present discussion, we define the *complex (linear) vector space* $\mathbb{C}^M$ as the set of all complex vectors that can be expressed as a linear combination of M *basis vectors*. Specifically, we may write

$$\mathbb{C}^M = \{\mathbf{y}\}, \tag{E.54}$$

where

$$\mathbf{y} = \sum_{i=1}^{M} a_i \mathbf{q}_i \tag{E.55}$$

is any complex vector, the $\mathbf{q}_i$ are the basis vectors, and the a_i are real-valued scalars. For the basis vectors, we may use any *orthonormal set* of vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ that satisfy the requirement

$$\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}. \tag{E.56}$$

In other words, each basis vector is *normalized* to have a Euclidean length or norm of unity and is *orthogonal* to every other basis vector in the set. The *dimension* M of the complex vector space $\mathbb{C}^M$ is the minimum number of basis vectors required to span the entire space.

The basis functions define the "coordinates" of a complex vector space. Any complex vector of compatible dimension may then be represented simply as a point in that space. Indeed, the idea of a complex vector space is a natural generalization of Euclidean geometry. Central to this idea is a *subspace*. We say that $\mathcal{S}$ is a subspace of the complex vector space $\mathbb{C}^M$ if $\mathcal{S}$ involves a *subset* of the M basis vectors that define $\mathbb{C}^M$. In other words, a subspace of dimension k is defined as the set of complex vectors that can be written as a linear combination of the basis vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k$; that is,

$$\mathbf{x} = \sum_{i=1}^{k} a_i \mathbf{q}_i. \tag{E.57}$$

Obviously, $k \leq M$. Note, however, that the dimension of the vector $\mathbf{x}$ is M.

These ideas are illustrated in the three-dimensional (real) vector space depicted in Fig. E.1. The $(\mathbf{q}_1, \mathbf{q}_2)$-plane represents a subspace $\mathcal{S}$ of dimensionality two. The representations of vector $\mathbf{y}$ and vector $\mathbf{x}$ (i.e., the part of $\mathbf{y}$ lying in subspace $\mathcal{S}$) are indicated in the figure.

Returning to the issue at hand, namely, a proof of the *minimax theorem* described in Eq. (E.53), we may proceed as follows: We first use the spectral theorem of Eq. (E.31) to decompose the M-by-M correlation matrix $\mathbf{R}$ as

$$\mathbf{R} = \sum_{i=1}^{M} \lambda_i \mathbf{q}_i \mathbf{q}_i^{\mathrm{H}},$$

where the λ_i are the eigenvalues of $\mathbf{R}$ and the $\mathbf{q}_i$ are the associated eigenvectors. In view of the fact that the orthonormality conditions of Eq. (E.24) are satisfied by the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$, we may adopt them as the M basis vectors of the complex vector

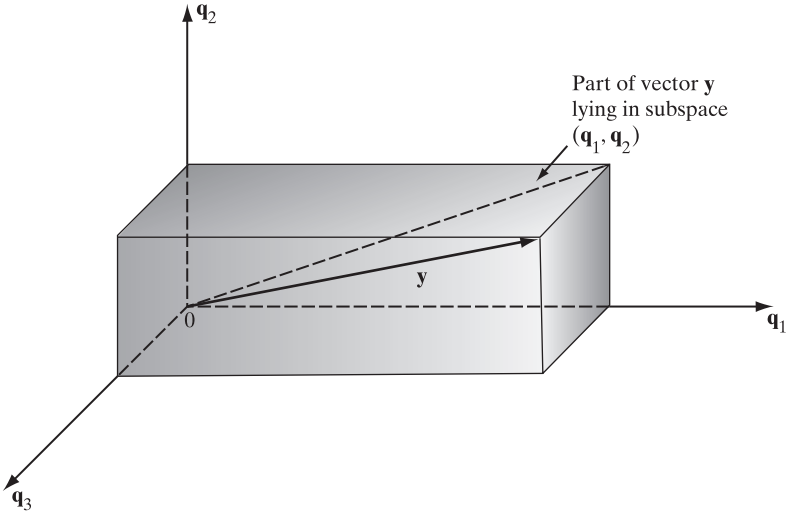


FIGURE E.1 Projection of a vector onto a subspace in a three-dimensional (real) vector space.

space $\mathbb{C}^M$. Let an M-by-1 vector $\mathbf{x}$ be constrained to lie in a subspace $\mathcal{S}$ of dimension k, as defined in Eq. (E.57). Then, using Eq. (E.31), we may express the Rayleigh quotient of the vector $\mathbf{x}$ as

$$\frac{\mathbf{x}^{\mathrm{H}}\mathbf{R}\mathbf{x}}{\mathbf{x}^{\mathrm{H}}\mathbf{x}} = \frac{\sum_{i=1}^{k} a_i^2 \lambda_i}{\sum_{i=1}^{k} a_i^2}. \tag{E.58}$$

Equation (E.58) states that the Rayleigh quotient of a vector $\mathbf{x}$ lying in the subspace $\mathcal{S}$ of dimension k (i.e., the subspace spanned by the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k$) is a *weighted mean* of the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_k$. Since, by assumption, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k$, it follows that, for any subspace $\mathcal{S}$ of dimension k,

$$\max_{\substack{\mathbf{x}\in\mathcal{S} \\ \mathbf{x}\neq\mathbf{0}}} \frac{\mathbf{x}^{\mathrm{H}}\mathbf{R}\mathbf{x}}{\mathbf{x}^{\mathrm{H}}\mathbf{x}} \leq \lambda_k.$$

This result implies that

$$\min_{\dim(\mathcal{S})=k} \max_{\substack{\mathbf{x}\in\mathcal{S} \\ \mathbf{x}\neq\mathbf{0}}} \frac{\mathbf{x}^{\mathrm{H}}\mathbf{R}\mathbf{x}}{\mathbf{x}^{\mathrm{H}}\mathbf{x}} \leq \lambda_k. \tag{E.59}$$

We next prove that, for any subspace $\mathcal{S}$ of dimension k spanned by the eigenvectors $\mathbf{q}_{i_1}, \mathbf{q}_{i_2}, \ldots, \mathbf{q}_{i_k}$ where $\{i_1, i_2, \ldots, i_k\}$ is a subset of $\{1, 2, \ldots, M\}$, there exists at least one nonzero vector $\mathbf{x}$ common to $\mathcal{S}$ and the subspace $\mathcal{S}'$ spanned by the eigenvectors $\mathbf{q}_k, \mathbf{q}_{k+1}, \ldots, \mathbf{q}_M$. To do so, we consider the system of M homogeneous equations,

$$\sum_{j=1}^{k} a_j \mathbf{q}_{i_j} = \sum_{i=k}^{M} b_i \mathbf{q}_i, \tag{E.60}$$

where the $(M+1)$ unknowns are made up as follows:

1. A total of k scalars, namely, $a_1, a_2, \ldots, a_k$, on the left-hand side.
2. A total of $M-k+1$ scalars, namely, $b_k, b_{k+1}, \ldots, b_M$, on the right-hand side.

Hence, the system of equations (E.60) will always have a nontrivial solution. Moreover, we know from Property 2 that the eigenvectors $\mathbf{q}_{i_1}, \mathbf{q}_{i_2}, \ldots, \mathbf{q}_{i_k}$ are *linearly independent*, as are the eigenvectors $\mathbf{q}_k, \mathbf{q}_{k+1}, \ldots, \mathbf{q}_M$. It follows therefore that there is at least one *nonzero vector* $\mathbf{x} = \sum_{j=1}^{k} a_j \mathbf{q}_{i_j}$ that is *common* to the space of $\mathbf{q}_{i_1}, \mathbf{q}_{i_2}, \ldots, \mathbf{q}_{i_k}$ and the space of $\mathbf{q}_k, \mathbf{q}_{k+1}, \ldots, \mathbf{q}_M$. Thus, using Eqs. (E.60), (E.57), and (E.41), we may also express the Rayleigh quotient of the vector $\mathbf{x}$ as a weighted mean of the eigenvalues $\lambda_k, \lambda_{k+1}, \ldots, \lambda_M$, as shown by

$$\frac{\mathbf{x}^{\mathrm{H}}\mathbf{R}\mathbf{x}}{\mathbf{x}^{\mathrm{H}}\mathbf{x}} = \frac{\sum_{i=k}^{M} b_i^2 \lambda_i}{\sum_{i=k}^{M} b_i^2}. \tag{E.61}$$

Since, by assumption, $\lambda_k \geq \lambda_{k+1} \geq \cdots \geq \lambda_M$, and since $\mathbf{x}$ is also a vector in the subspace $\mathcal{S}$, we may write

$$\max_{\substack{\mathbf{x}\in\mathcal{S}\\ \mathbf{x}\neq\mathbf{0}}} \frac{\mathbf{x}^{\mathrm{H}}\mathbf{R}\mathbf{x}}{\mathbf{x}^{\mathrm{H}}\mathbf{x}} \geq \lambda_k.$$

Therefore,

$$\min_{\dim(\mathcal{S})=k} \max_{\substack{\mathbf{x}\in\mathcal{S}\\ \mathbf{x}\neq\mathbf{0}}} \frac{\mathbf{x}^{\mathrm{H}}\mathbf{R}\mathbf{x}}{\mathbf{x}^{\mathrm{H}}\mathbf{x}} \geq \lambda_k, \tag{E.62}$$

because $\mathcal{S}$ is an arbitrary subspace of dimension k.

All that remains for us to do is to combine the results of Eqs. (E.59) and (E.62), and the minimax theorem of Eq. (E.53) describing Property 9 follows immediately.

From Property 9, we may make two important observations:

1. The minimax theorem, as stated in Eq. (E.53), does not require any special knowledge of the eigenstructure (i.e., eigenvalues and eigenvectors) of the correlation matrix $\mathbf{R}$. Indeed, the theorem may be adopted as the basis for defining the eigenvalues λ_k for $k = 1, 2, \ldots, M$.
2. The minimax theorem points to a unique twofold feature of the eigenstructure of the correlation matrix: (a) The eigenvectors represent the particular basis for an M-dimensional space that is most efficient in the energy sense, and (b) the eigenvalues are certain energies of the M-by-1 input (observation) vector $\mathbf{u}(n)$. This issue is pursued in greater depth under Property 10.

Another noteworthy point is that Eq. (E.53) may also be formulated in the following alternative, but equivalent, form:

$$\lambda_k = \max_{\dim(\mathcal{S}')=M-k+1} \min_{\substack{\mathbf{x}\in\mathcal{S}'\\ \mathbf{x}\neq\mathbf{0}}} \frac{\mathbf{x}^{\mathrm{H}}\mathbf{R}\mathbf{x}}{\mathbf{x}^{\mathrm{H}}\mathbf{x}}. \tag{E.63}$$

Equation (E.63) is referred to as the *maximin theorem*.

From Eqs. (E.53) and (E.63), we may readily deduce the following two special cases:

1. For $k = M$, the subspace $\mathcal{S}$ occupies the complex vector space $\mathbb{C}^M$ entirely. Under this condition, Eq. (E.53) reduces to

$$\lambda_1 = \max_{\substack{\mathbf{x}\in\mathbb{C}^M\\ \mathbf{x}\neq\mathbf{0}}} \frac{\mathbf{x}^{\mathrm{H}}\mathbf{R}\mathbf{x}}{\mathbf{x}^{\mathrm{H}}\mathbf{x}}, \tag{E.64}$$

   where λ_1 is the *largest eigenvalue* of the correlation matrix $\mathbf{R}$.
2. For $k = 1$, the subspace $\mathcal{S}'$ occupies the complex vector space $\mathbb{C}^M$ entirely. Under this condition, Eq. (E.63) reduces to

$$\lambda_M = \min_{\substack{\mathbf{x}\in\mathbb{C}^M\\ \mathbf{x}\neq\mathbf{0}}} \frac{\mathbf{x}^{\mathrm{H}}\mathbf{R}\mathbf{x}}{\mathbf{x}^{\mathrm{H}}\mathbf{x}}, \tag{E.65}$$

   where λ_M is the *smallest eigenvalue* of the correlation matrix $\mathbf{R}$.

Property 10: Karhunen–Loève expansion. *Let the M-by-1 vector* $\mathbf{u}(n)$ *denote a data sequence drawn from a wide-sense stationary process of zero mean and correlation*

matrix $\mathbf{R}$. *Let* $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ *be eigenvectors associated with the* M *eigenvalues of the matrix* $\mathbf{R}$. *Then, the vector* $\mathbf{u}(n)$ *may be expanded as a linear combination of these eigenvectors as follows*:

$$\mathbf{u}(n) = \sum_{i=1}^{M} c_i(n)\mathbf{q}_i. \tag{E.66}$$

The coefficients of the expansion are zero-mean, uncorrelated random variables defined by the inner product

$$c_i(n) = \mathbf{q}_i^{\mathrm{H}}\mathbf{u}(n), \qquad i = 1, 2, \ldots, M. \tag{E.67}$$

The representation of the random vector $\mathbf{u}(n)$ described by Eqs. (E.66) and (E.67) is the discrete-time version of the *Karhunen–Loève expansion*. In particular, Eq. (E.67) is the "analysis" part of the expansion, in that it defines $c_i(n)$ in terms of the input vector $\mathbf{u}(n)$. On the other hand, Eq. (E.66) is the "synthesis" part of the expansion, in that it reconstructs the original input vector $\mathbf{u}(n)$ from $c_i(n)$. Given the expansion of Eq. (E.66), the definition of $c_i(n)$ in Eq. (E.67) follows directly from the fact that the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ form an orthonormal set, assuming that they are all normalized to have unit length. Conversely, this same property may be used to derive Eq. (E.66), given Eq. (E.67).

The coefficients of the expansion are random variables characterized as

$$\mathbb{E}[c_i(n)] = 0, \qquad i = 1, 2, \ldots, M \tag{E.68}$$

and

$$\mathbb{E}[c_i(n)c_j^*(n)] = \begin{cases} \lambda_i & i = j \\ 0, & i \neq j \end{cases}. \tag{E.69}$$

Equation (E.68) states that all the coefficients of the expansion have zero mean; this follows directly from Eq. (E.67) and the fact the random vector $\mathbf{u}(n)$ is itself assumed to have zero mean. Equation (E.69) states that the coefficients of the expansion are uncorrelated and that each one of them has a mean-square value equal to the respective eigenvalue. This second equation is readily obtained by using the expansion of Eq. (E.66) in the definition of the correlation matrix $\mathbf{R}$ as the expectation of the outer product $\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)$ and then invoking the unitary similarity transformation (i.e., Property 5).

For a physical interpretation of the Karhunen–Loève expansion, we may view the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ as the coordinates of an M-dimensional space and thus represent the random vector $\mathbf{u}(n)$ by the set of its projections $c_1(n), c_2(n), \ldots, c_M(n)$ onto these axes, respectively. Moreover, from Eq. (E.66), we see that

$$\sum_{i=1}^{M} |c_i(n)|^2 = \|\mathbf{u}(n)\|^2, \tag{E.70}$$

where $\|\mathbf{u}(n)\|$ is the Euclidean norm of $\mathbf{u}(n)$. That is to say, the coefficient $c_i(n)$ has an energy equal to that of the observation vector $\mathbf{u}(n)$, measured along the ith coordinate. Naturally, this energy is a random variable whose mean value equals the ith eigenvalue, as shown by the formula

$$\mathbb{E}[|c_i(n)|^2] = \lambda_i, \qquad i = 1, 2, \ldots, M. \tag{E.71}$$

This result follows directly from Eqs. (E.67) and (E.69).

E.3 LOW-RANK MODELING

A key problem in statistical signal processing is that of *feature selection*, which refers to a process whereby a *data space* is transformed into a *feature space* that, in theory, has exactly the same dimension as the original data space. However, it would be desirable to design the transformation in such a way that the data vector can be represented by a reduced number of "effective" features and yet retain most of the intrinsic information content of the input data. In other words, the data vector undergoes a *reduction in dimensionality*.

To be specific, suppose we have an M-dimensional data vector $\mathbf{u}(n)$ representing a sample realization of a wide-sense stationary process. Suppose also that we would like to transmit this vector over a noisy channel using a new set of p distinct numbers, where $p < M$. Basically, this is a feature-selection problem, which may be solved with the help of the Karhunen–Loève expansion, described next.

According to Eq. (E.66), the data vector $\mathbf{u}(n)$ may be expanded as a linear combination of the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ associated with the respective eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$ of the correlation matrix $\mathbf{R}$ of $\mathbf{u}(n)$. It is assumed that the eigenvalues are all distinct and arranged in decreasing order, as shown by

$$\lambda_1 > \lambda_2 > \cdots > \lambda_i > \cdots > \lambda_M. \tag{E.72}$$

The data representation described in Eq. (E.66) using all the eigenvalues of matrix $\mathbf{R}$ is *exact*, in the sense that it involves *no* loss of information. Suppose, however, that we have *prior knowledge* that the $M - p$ eigenvalues $\lambda_{p+1}, \ldots, \lambda_M$ at the tail end of Eq. (E.72) are all very small. Then, we may take advantage of this prior knowledge by retaining the p largest eigenvalues of matrix $\mathbf{R}$ and thereby truncating the Karhunen–Loève expansion of Eq. (E.66) at the term $i = p$. Accordingly, we may define an *approximate reconstruction* of the data vector $\mathbf{u}(n)$ defined by:

$$\hat{\mathbf{u}}(n) = \sum_{i=1}^{p} c_i(n)\mathbf{q}_i, \qquad p < M. \tag{E.73}$$

The vector $\hat{\mathbf{u}}(n)$ has rank p, which is lower than the rank M of the original data vector $\mathbf{u}(n)$. For this reason, the data model defined by Eq. (E.73) is referred to as a *low-rank model*. The important point to note here is that we may reconstruct the approximation $\hat{\mathbf{u}}(n)$ by using the set of p numbers, $\{c_i(n); i = 1, 2, \ldots, p\}$. The $c_i(n)$ are themselves defined in terms of the data vector $\mathbf{u}(n)$ by Eq. (E.67). In other words, the new vector $\mathbf{c}(n)$, having $c_1(n), c_2(n), \ldots, c_p(n)$ as elements, may be viewed as the *reduced-rank representation* of the original data vector $\mathbf{u}(n)$.

Figure E.2 depicts the essence of the feature selection procedure. We start with an M-dimensional *data space* in which a particular point defines the location of the data vector $\mathbf{u}(n)$. This point is transformed, via Eq. (E.67), into a new point in a *feature*

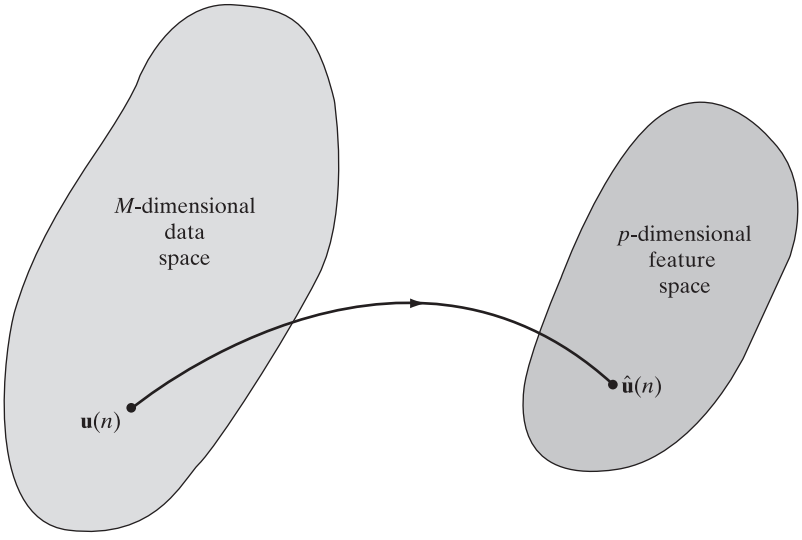


FIGURE E.2 Transformation involved in subspace decomposition.

space of dimension p that is lower than M. The transformation depicted in the figure is sometimes referred to as a *subspace*, or *reduced-rank*, *decomposition*.

Clearly, in using Eq. (E.73) to reconstruct the data vector $\mathbf{u}(n)$, an error is incurred due to the fact that $\hat{\mathbf{u}}(n)$ is of lower rank than $\mathbf{u}(n)$. The *reconstruction error vector* is defined by

$$\mathbf{e}(n) = \mathbf{u}(n) - \hat{\mathbf{u}}(n). \tag{E.74}$$

Using Eqs. (E.66) and (E.73) in Eq. (E.74) yields

$$\mathbf{e}(n) = \sum_{i=p+1}^{M} c_i(n)\mathbf{q}_i. \tag{E.75}$$

The mean-square error is therefore

$$\begin{aligned}
\varepsilon &= \mathbb{E}[\|\mathbf{e}(n)\|^2] \\
&= \mathbb{E}[\mathbf{e}^{\mathrm{H}}(n)\mathbf{e}(n)] \\
&= \mathbb{E}\left[\sum_{i=p+1}^{M}\sum_{j=p+1}^{M} c_i^*(n)c_j(n)\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_j\right] \\
&= \sum_{i=p+1}^{M}\sum_{j=p+1}^{M} \mathbb{E}[c_i^*(n)c_j(n)]\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_j \\
&= \sum_{i=p+1}^{M} \lambda_i,
\end{aligned} \tag{E.76}$$

which confirms that the data reconstruction defined by Eq. (E.73) is a good one, provided that the eigenvalues $\lambda_{p+1}, \ldots, \lambda_M$ are all very small compared with $\lambda_1, \ldots, \lambda_p$.

An Application of Low-Rank Modeling

To appreciate the practical value of the low-rank model based on Eq. (E.73), consider the transmission of a data vector $\mathbf{u}(n)$ over a *noisy communication channel*. Suppose the received signal is corrupted by *channel noise vector* $\boldsymbol{\nu}(n)$, which is modeled as additive white noise of zero mean. Then,

$$\mathbb{E}[\mathbf{u}(n)\boldsymbol{\nu}^{\mathrm{H}}(n)] = \mathbf{O} \tag{E.77}$$

and

$$\mathbb{E}[\boldsymbol{\nu}(n)\boldsymbol{\nu}^{\mathrm{H}}(n)] = \sigma^2\mathbf{I}. \tag{E.78}$$

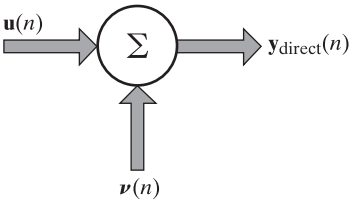
Equation (E.77) says that the noise vector $\boldsymbol{\nu}(n)$ is uncorrelated with the data vector $\mathbf{u}(n)$. Equation (E.78), with $\mathbf{I}$ denoting the identity matrix, says that the elements of the noise vector are uncorrelated with each other and that each element has variance σ^2.

In Fig. E.3, we describe two methods for accomplishing the data transmission over the channel. One method is *direct*, and the other is *indirect*. In the direct method shown in Fig. E.3(a), the received signal vector is given by

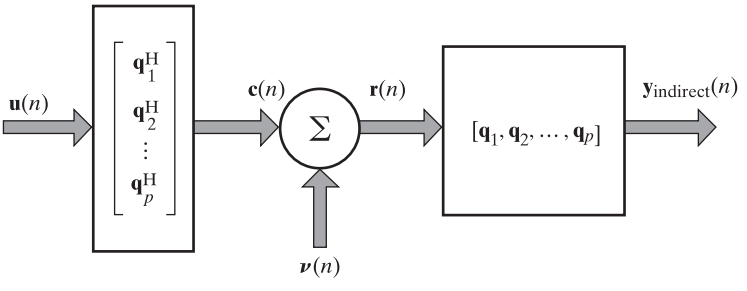
$$\mathbf{y}_{\text{direct}}(n) = \mathbf{u}(n) + \boldsymbol{\nu}(n). \tag{E.79}$$

The mean-square value of the *transmission error* is therefore

$$\begin{aligned}\varepsilon_{\text{direct}} &= \mathbb{E}[\|\mathbf{y}_{\text{direct}}(n) - \mathbf{u}(n)\|^2] \\ &= \mathbb{E}[\|\boldsymbol{\nu}(n)\|^2] \\ &= \mathbb{E}[\boldsymbol{\nu}^{\mathrm{H}}(n)\boldsymbol{\nu}(n)].\end{aligned}$$



(a)



(b)

FIGURE E.3 Data transmission using (a) a direct method and (b) an indirect method inspired by low-rank modeling.

From Eq. (E.78), we see that each element, say, $\nu_i(n)$, of the noise vector $\boldsymbol{\nu}(n)$ has variance σ^2. We may therefore express $\boldsymbol{\varepsilon}_{\text{direct}}$ simply as

$$\begin{aligned}\varepsilon_{\text{direct}} &= \sum_{i=1}^{M} \mathbb{E}[|\nu_i(n)|^2] \\ &= M\sigma^2,\end{aligned} \tag{E.80}$$

where M is the size of $\boldsymbol{\nu}(n)$.

Consider next the indirect method shown in Fig. E.3(b), in which the input vector $\mathbf{u}(n)$ is first applied to a *transmit filter bank* whose individual tap-weight vectors are set equal to the Hermitian transpose of the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_p$ associated with the p largest eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ of the correlation matrix $\mathbf{R}$ of $\mathbf{u}(n)$. The resulting p-by-1 vector $\mathbf{c}(n)$, whose elements are made up of the inner products of $\mathbf{u}(n)$ with $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_p$, in accordance with Eq. (E.67), constitutes the transmitted signal vector

$$\mathbf{c}(n) = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_p]^{\mathrm{H}}\mathbf{u}(n). \tag{E.81}$$

Correspondingly, the received signal vector is defined by

$$\mathbf{r}(n) = \mathbf{c}(n) + \boldsymbol{\nu}(n), \tag{E.82}$$

where the channel noise vector $\boldsymbol{\nu}(n)$ is now of size p, to be compatible with that of $\mathbf{c}(n)$. To reconstruct the original data vector $\mathbf{u}(n)$, the received signal vector $\mathbf{r}(n)$ is applied to a *receive filter bank*, whose individual tap-weight vectors are defined by the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_p$. The resulting output vector of the receiver is given by

$$\begin{aligned}\mathbf{y}_{\text{indirect}}(n) &= [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_p]\mathbf{r}(n) \\ &= [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_p]\mathbf{c}(n) + [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_p]\boldsymbol{\nu}(n).\end{aligned} \tag{E.83}$$

Hence, evaluating the mean-square value of the overall reconstruction error for the indirect method, we get

$$\begin{aligned}\boldsymbol{\varepsilon}_{\text{indirect}} &= \mathbb{E}[\|\mathbf{y}_{\text{indirect}}(n) - \mathbf{u}(n)\|^2] \\ &= \sum_{i=p+1}^{M} \lambda_i + p\sigma^2.\end{aligned} \tag{E.84}$$

The first term of Eq. (E.84) is due to the low-rank modeling of the data vector $\mathbf{u}(n)$ prior to transmission over the channel. The second term is due to the effect of channel noise.

Comparing Eq. (E.84) for the indirect method with Eq. (E.80) for the direct method, we readily see that the use of low-rank modeling offers an advantage, provided that we have

$$\sum_{i=p+1}^{M} \lambda_i < (M - p)\sigma^2. \tag{E.85}$$

This is an interesting result (Scharf & Tufts, 1987). It states that if the tail-end eigenvalues $\lambda_{p+1}, \ldots, \lambda_M$ of the correlation matrix of the data vector $\mathbf{u}(n)$ are all very small, the mean-square error produced by transmitting a low-rank approximation to the original data vector [as in Fig. E.3(b)] is less than the mean-square error produced by transmitting the original data vector without any approximation [as in Fig. E.3(a)].

The result described in Eq. (E.84) is particularly important in that it highlights the essence of what is commonly referred to as the "bias–variance trade-off." Specifically, a low-rank model is used for representing the data vector $\mathbf{u}(n)$, thereby incurring a *bias*. Interestingly enough, this is done knowingly, in return for a reduction in *variance*, namely, the part of the mean-square error due to the additive noise vector $\boldsymbol{\nu}(n)$. Indeed, the example described herein clearly illustrates the motivation for using a parsimonious (i.e., simpler) model that may not exactly match the underlying physics responsible for generating the data vector $\mathbf{u}(n)$, hence the bias, but the model is less susceptible to noise, hence a reduction in variance.

E.4 EIGENFILTERS

A fundamental issue in communication theory is that of determining an optimum FIR filter, with the optimization criterion being that of maximizing the output signal-to-noise ratio. In this section, we show that this filter optimization is linked to an eigenvalue problem.

Consider an FIR filter whose impulse response is denoted by the sequence w_n. The sequence $x(n)$ applied to the filter input consists of a useful *signal* component $u(n)$, plus an additive *noise* component $\nu(n)$. The signal $u(n)$ is drawn from a wide-sense stationary stochastic process of zero mean and correlation matrix $\mathbf{R}$. The zero-mean noise $\nu(n)$ is white with a constant power spectral density determined by the variance σ^2. It is assumed that the signal $u(n)$ and the noise $\nu(n)$ are uncorrelated; that is,

$$\mathbb{E}[u(n)\nu^*(m)] = 0 \qquad \text{for all } (n, m).$$

The filter output is denoted by $y(n)$. The situation described herein is depicted in Fig. E.4.

Since the filter is linear, the principle of superposition applies. We may therefore consider the effects of signal and noise separately. Let P_o denote the average power of the signal component of the filter output $y(n)$. We may thus write:

$$P_o = \mathbf{w}^{\mathrm{H}}\mathbf{R}\mathbf{w}, \tag{E.86}$$

where the elements of the vector $\mathbf{w}$ are the filter coefficients and $\mathbf{R}$ is the correlation matrix of the signal component $u(n)$ in the filter input $x(n)$.

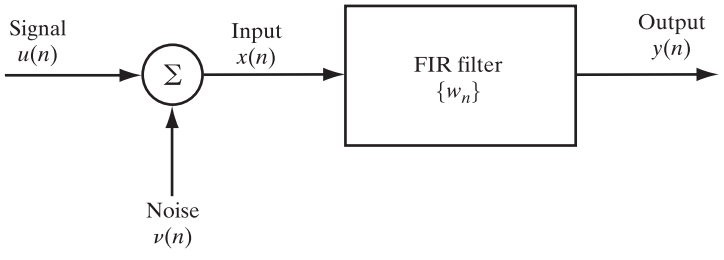


FIGURE E.4 FIR filtering.

Consider next the effect of noise acting alone. Let N_o denote the average power of the noise component in the filter output $y(n)$. This is a special case of Eq. (E.86), given by

$$N_o = \sigma^2 \mathbf{w}^H \mathbf{w}, \tag{E.87}$$

where σ^2 is the variance of the white noise in the filter input $x(n)$.

Let $(\text{SNR})_o$ denote the *output signal-to-noise ratio*. Dividing Eq. (E.86) by (E.87), we may write

$$\begin{aligned}(\text{SNR})_o &= \frac{P_o}{N_o} \\ &= \frac{\mathbf{w}^H \mathbf{R} \mathbf{w}}{\sigma^2 \mathbf{w}^H \mathbf{w}}.\end{aligned} \tag{E.88}$$

The optimization problem may now be stated as follows:

> Determine the coefficient vector $\mathbf{w}$ of an FIR filter so as to maximize the $(\text{SNR})_o$ subject to the constraint $\mathbf{w}^H \mathbf{w} = 1$.

Equation (E.88) shows that, except for the scaling factor $1/\sigma^2$, the $(\text{SNR})_o$ is equal to the Rayleigh quotient of the coefficient vector $\mathbf{w}$ of the FIR filter. We see therefore that the optimum-filtering problem, as stated herein, may be viewed as an eigenvalue problem. Indeed, the solution to the problem follows directly from the minimax theorem. Specifically, using the special form of the minimax theorem given in Eq. (E.64), we may state the following:

- The maximum value of the output signal-to-noise ratio is given by

$$(\text{SNR})_{o,\,\max} = \frac{\lambda_{\max}}{\sigma^2}, \tag{E.89}$$

  where $\lambda_{\max}$ is the largest eigenvalue of the correlation matrix $\mathbf{R}$. (Note that $\lambda_{\max}$ and σ^2 have the same units but different physical interpretations.)
- The coefficient vector of the optimum FIR filter that yields the maximum output signal-to-noise ratio of Eq. (E.89) is defined by

$$\mathbf{w}_o = \mathbf{q}_{\max}, \tag{E.90}$$

  where $\mathbf{q}_{\max}$ is the eigenvector associated with the largest eigenvalue $\lambda_{\max}$ of the correlation matrix $\mathbf{R}$. The correlation matrix $\mathbf{R}$ belongs to the signal component $u(n)$ in the filter input as indicated previously.

An FIR filter whose impulse response has coefficients equal to the elements of an eigenvector is called an *eigenfilter* (Makhoul, 1981). Accordingly, we may state that *the maximum eigenfilter (i.e., the eigenfilter associated with the largest eigenvalue of the correlation matrix of the signal component in the filter input) is the optimum filter*. It is important to note that the optimum filter described in this way is uniquely characterized by an eigendecomposition of the correlation matrix of the signal component in the filter input. The power spectrum of the white noise at the filter input merely affects the

maximum value of the $(\text{SNR})_o$. In particular, to compute the maximum eigenfilter, we may proceed as follows:

1. An eigendecomposition of the correlation matrix $\mathbf{R}$ is performed.
2. *Only the largest eigenvalue* $\lambda_{\max}$ and the associated eigenvector $\mathbf{q}_{\max}$ are retained.
3. The eigenvector $\mathbf{q}_{\max}$ defines the impulse response of the optimum filter. The eigenvalue $\lambda_{\max}$, divided by the noise variance σ^2, defines the maximum value of the $(\text{SNR})_o$.

The optimum filter so characterized may be viewed as the stochastic counterpart of a matched filter. The optimum filter just described maximizes the $(\text{SNR})_o$ for a *random signal* (i.e., a sample function of a discrete-time wide-sense stationary stochastic process) in additive white noise. A *matched filter*, on the other hand, maximizes $(\text{SNR})_o$ for a *known signal* in additive white noise (North, 1963; Haykin, 2001).

E.5 EIGENVALUE COMPUTATIONS

The origin of almost all canned eigenroutines may be traced back to routines published in Volume II, *Linear Algebra*, of the *Handbook for Automatic Computation*, coedited by Wilkinson and Reinsch (1971). This reference is the bible of eigenvalue computations.[4]

Some eigenroutines, for example, those written in C, can handle only real matrices. It is, however, a straightforward matter to extend the use of such eigenroutines to deal with Hermitian matrices. To this end, let $\mathbf{A}$ denote an M-by-M Hermitian matrix, written in terms of its real and imaginary parts as follows:

$$\mathbf{A} = \mathbf{A}_r + \mathrm{j}\mathbf{A}_i. \tag{E.91}$$

Correspondingly, let an associated M-by-1 eigenvector $\mathbf{q}$ be written as

$$\mathbf{q} = \mathbf{q}_r + \mathrm{j}\mathbf{q}_i. \tag{E.92}$$

The M-by-M complex eigenvalue problem

$$(\mathbf{A}_r + \mathrm{j}\mathbf{A}_i)(\mathbf{q}_r + \mathrm{j}\mathbf{q}_i) = \lambda(\mathbf{q}_r + \mathrm{j}\mathbf{q}_i) \tag{E.93}$$

may then be reformulated as the $2M$-by-$2M$ real eigenvalue problem:

$$\begin{bmatrix} \mathbf{A}_r & -\mathbf{A}_i \\ \mathbf{A}_i & \mathbf{A}_r \end{bmatrix} \begin{bmatrix} \mathbf{q}_r \\ \mathbf{q}_i \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{q}_r \\ \mathbf{q}_i \end{bmatrix}, \tag{E.94}$$

where the eigenvalue λ is a real number. The Hermitian property

$$\mathbf{A}^{\mathrm{H}} = \mathbf{A}$$

[4]For books on eigenvalue computations, the reader is referred to Cullum and Willoughby (1985), Saad (2011), Chaitlin-Chatellin and Ahues (1993), Golub and Van Loan (1996), Parlett (1998), and Press et al. (1988).

is equivalent to $\mathbf{A}_r^{\mathrm{T}} = \mathbf{A}_r$ and $\mathbf{A}_i^{\mathrm{T}} = -\mathbf{A}_i$. Accordingly, the $2M$-by-$2M$ matrix in Eq. (E.94) is not only real but also symmetric. Note, however, that for a given eigenvalue λ, the vector

$$\begin{bmatrix} -\mathbf{q}_i \\ \mathbf{q}_r \end{bmatrix}$$

is also an eigenvector. This means that if $\lambda_1, \lambda_2, \ldots, \lambda_M$ are the eigenvalues of the M-by-M Hermitian matrix $\mathbf{A}$, then the eigenvalues of the $2M$-by-$2M$ symmetric matrix of Eq. (E.94) are $\lambda_1, \lambda_1, \lambda_2, \lambda_2, \ldots, \lambda_M, \lambda_M$. We may therefore make two observations:

1. Each eigenvalue of the matrix in Eq. (E.94) has a multiplicity of two.
2. The associated eigenvectors consist of pairs, each of the form $\mathbf{q}_r + \mathrm{j}\mathbf{q}_i$ and $\mathrm{j}(\mathbf{q}_r + \mathrm{j}\mathbf{q}_i)$, differing merely by a rotation through 90°.

Thus, to solve the M-by-M complex eigenvalue problem of Eq. (E.93) with the aid of real eigenroutines, we choose one eigenvalue and one eigenvector from each pair associated with the augmented $2M$-by-$2M$ real eigenvalue problem of Eq. (E.94).

Strategies for Matrix Eigenvalue Computations

There are two different "strategies" behind practically all modern eigenroutines: *diagonalization* and *triangularization*. Since not all matrices can be diagonalized through a sequence of unitary similarity transformations, the diagonalization strategy applies only to Hermitian matrices, such as a correlation matrix. On the other hand, the triangularization strategy is general, in that it applies to any square matrix. These two strategies are described next.

Diagonalization. The idea behind this strategy is to nudge a Hermitian matrix $\mathbf{A}$ toward a diagonal form by the repeated application of unitary similarity transformations, such as

$$\begin{aligned} \mathbf{A} &\rightarrow \mathbf{Q}_1^{\mathrm{H}}\mathbf{A}\mathbf{Q}_1 \\ &\rightarrow \mathbf{Q}_2^{\mathrm{H}}\mathbf{Q}_1^{\mathrm{H}}\mathbf{A}\mathbf{Q}_1\mathbf{Q}_2 \\ &\rightarrow \mathbf{Q}_3^{\mathrm{H}}\mathbf{Q}_2^{\mathrm{H}}\mathbf{Q}_1^{\mathrm{H}}\mathbf{A}\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3, \end{aligned} \tag{E.95}$$

and so on. This sequence of unitary similarity transformations is, in theory, infinitely long. In practice, however, it is continued until we are close to a diagonal matrix. The elements of the diagonal matrix so obtained define the eigenvalues of the original Hermitian matrix $\mathbf{A}$. The associated eigenvectors are the column vectors of the accumulated sequence of transformations, as shown by

$$\mathbf{Q} = \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3 \ldots . \tag{E.96}$$

One method for implementing the diagonalization strategy of Eq. (E.95) is to use *Givens rotations*. This method is discussed in Appendix G.

Triangularization. The idea behind this second strategy is to reduce a Hermitian matrix $\mathbf{A}$ to a triangular form by a sequence of unitary similarity transformations. The

resulting iterative procedure is called the *QL algorithm*.[5] Suppose that we are given an M-by-M Hermitian matrix $\mathbf{A}_n$, where the subscript n refers to a particular adaptation cycle in the iterative procedure. Let the matrix $\mathbf{A}_n$ be *factored* into the form

$$\mathbf{A}_n = \mathbf{Q}_n\mathbf{L}_n, \tag{E.97}$$

where $\mathbf{Q}_n$ is a *unitary matrix* and $\mathbf{L}_n$ is a *lower triangular matrix* (i.e., the elements of the matrix $\mathbf{L}_n$ located above the main diagonal are all zero). At adaptation cycle $n+1$ in the iterative procedure, we use the known matrices $\mathbf{Q}_n$ and $\mathbf{L}_n$ to compute a new M-by-M matrix

$$\mathbf{A}_{n+1} = \mathbf{L}_n\mathbf{Q}_n. \tag{E.98}$$

Note that the factorization in Eq. (E.98) is written in the opposite order to that in Eq. (E.97). Since $\mathbf{Q}_n$ is a unitary matrix, we have $\mathbf{Q}_n^{-1} = \mathbf{Q}_n^{\mathrm{H}}$, so we may rewrite Eq. (E.97) as

$$\begin{aligned}\mathbf{L}_n &= \mathbf{Q}_n^{-1}\mathbf{A}_n \\ &= \mathbf{Q}_n^{\mathrm{H}}\mathbf{A}_n.\end{aligned} \tag{E.99}$$

Therefore, substituting Eq. (E.99) into Eq. (E.98), we get

$$\mathbf{A}_{n+1} = \mathbf{Q}_n^{\mathrm{H}}\mathbf{A}_n\mathbf{Q}_n. \tag{E.100}$$

Equation (E.100) shows that the Hermitian matrix $\mathbf{A}_{n+1}$ at adaptation cycle $n+1$ is indeed unitarily similar to the Hermitian matrix $\mathbf{A}_n$ at adaptation cycle n.

The QL algorithm thus consists of a sequence of unitary similarity transformations, summarized by writing

$$\mathbf{A}_n = \mathbf{Q}_n\mathbf{L}_n$$

and

$$\mathbf{A}_{n+1} = \mathbf{L}_n\mathbf{Q}_n,$$

where $n = 0, 1, 2, \ldots$. The algorithm is *initialized* by setting

$$\mathbf{A}_0 = \mathbf{A},$$

where $\mathbf{A}$ is the given M-by-M Hermitian matrix.

For a general matrix $\mathbf{A}$, the following theorem is the basis of the QL algorithm[6]:

> If the matrix $\mathbf{A}$ has eigenvalues of different absolute values, then the matrix $\mathbf{A}_n$ approaches a lower triangular form as the number of adaptation cycles, n, approaches infinity.

The eigenvalues of the original matrix $\mathbf{A}$ appear on the main diagonal of the lower triangular matrix resulting from the QL algorithm, in increasing order of absolute value.

To implement the factorization in Eq. (E.98), we may use Givens rotations. (See Appendix G, in which we discuss computations for the singular-value decomposition of a general matrix, which includes eigendecomposition as a special case.)

[5]The QL algorithm uses a lower triangular matrix. A companion algorithm, called the QR algorithm, uses an upper triangular matrix. The QR algorithm is not to be confused with the QR decomposition, which is discussed in Appendix G and used in Chapters 15 and 16.

[6]For a proof of this theorem, see Stoer and Bulirsch (1980). See also Stewart (1973) and Golub and Van Loan (1996) for an improved version of the QL algorithm.

APPENDIX F

Langevin Equation of Nonequilibrium Thermodynamics

F.1 BROWNIAN MOTION

Brownian motion, named in honor of the English botanist Robert Brown, refers to the erratic motion of little particles of plant pollens that jiggle around on a water surface due to continuous bombardment by water molecules. The physical description of Brownian motion presented here follows Reif (1965). Brownian motion is sometimes referred to as a *Wiener process,* so named in honor of Norbert Wiener for a mathematical description of it (Kloeden and Platen, 1995).

Consider a microscopic particle of mass m immersed in a viscous fluid at absolute temperature T. Suppose that the direction of the gravitational field points along the z-axis, and let $v_x(t)$ denote the x component of the center-of-mass velocity of that particle, measured at time t. By symmetry, the ensemble-average mean value of $v_x(t)$ must be zero, as shown by

$$\mathbb{E}[v_x(t)] = 0 \qquad \text{for all } t. \tag{F.1}$$

However, for an ensemble of such particles, velocity fluctuations occur in the course of time t, with the result that, in accordance with the *equipartition law of classical statistical mechanics*, the ensemble-average variance of the velocity $v_x(t)$ is inversely proportional to the mass m, as shown by

$$\mathbb{E}[v_x^2(t)] = \frac{k_B t}{m} \quad \text{for all } t, \tag{F.2}$$

where k_B is Baltzmann's constant. Thus, when the particle is little, fluctuations in the velocity $v_x(t)$ are large, which is intuitively satisfying. Equations (F.l) and (F.2), viewed together, describe a *random walk.*

For a formal definition of Brownian motion, we may therefore make the statement:

Brownian motion is a random walk that is Gaussian distributed.

F.2 LANGEVIN EQUATION

With this brief description of Brownian motion at hand, we are now ready to formulate the Langevin equation. To proceed with the formulation, we first recognize that the total force exerted on the particle by the molecules in the viscous fluid is made up of two components (Reif, 1965):

1. A *continuous damping force,* which is equal to $-\alpha v_x(t)$, in accordance with *Stoke's law* in fluid mechanics, where the constant α denotes the *coefficient of friction*.

2. A *fluctuating force,* denoted by $F_f(t)$, which is stochastic in nature; in statistical terms, the properties of $F_f(t)$ are specified on the average.

In the absence of any external force, the equation of motion of the particle is therefore defined by the sum of the damping and fluctuating forces:

$$m\frac{dv_x(t)}{dt} = -\alpha v_x(t) + F_f(t).$$

Dividing both sides of the equation by the mass m, we may equivalently write

$$\frac{dv_x(t)}{dt} = -\gamma v_x(t) + \Gamma(t), \tag{F.3}$$

where

$$\gamma = \frac{\alpha}{m}$$

denotes the *normalized coefficient of friction* and

$$\Gamma(t) = \frac{F_f(t)}{m}$$

denotes a stochastic (fluctuating) force exerted on the particle, normalized with respect to the particle's mass m.

The stochastic differential equation (F.3) is called the *Langevin equation;* correspondingly, $\Gamma(t)$ is called the *Langevin force,* which is assumed to have a Gaussian distribution. In physical terms, the Langevin equation describes the stochastic motion of a little particle in a viscous fluid if its initial conditions are specified. It is important because it was the first mathematical equation describing *nonequilibrium thermodynamics* (Langevin, 1908; Uhlenbeck & Ornstein, 1930; Reif, 1965).

It is the nonequilibrium characteristic that makes the Langevin equation befitting for the study of least-mean-square (LMS) learning theory, recognizing that the LMS algorithm is itself a *nonlinear example of the method of stochastic gradients.* The rationale for saying so is that when the LMS algorithm ultimately approaches the neighborhood of the Wiener solution, it performs a Brownian-like motion, as illustrated by the computer experiments described in Sections 6.7 and 6.8 of Chapter 6.

APPENDIX G

Rotations and Reflections

In Chapter 9, we emphasized the importance of singular-value decomposition (SVD) as a tool for solving the linear least-squares problem. In this appendix, we discuss the practical issue of how to compute the SVD of a data matrix. With numerical stability as a primary design objective, the recommended procedure for SVD computation is to work directly with the data matrix. In this context, we may mention two different algorithms for computing the SVD:

- The *QR algorithm,* which proceeds by using a sequence of planar reflections known as Householder transformations.
- The *cyclic Jacobi algorithm,* which employs a sequence of 2-by-2 plane rotations known as Jacobi rotations or Givens rotations.

The cyclic Jacobi algorithm and the QR algorithm are both *data adaptive* and *block-processing* oriented. They share a common goal, albeit in different ways:

> The diagonalization of the data matrix in a step-by-step fashion, to within some prescribed numerical precision.

It is important to note that plane rotations and reflections are wide ranging in their applications. In particular, they play a key role in the design of square-root Kalman filters and related linear adaptive filters. We therefore have reason in some chapters of the book to refer to some of the fundamental concepts presented in this appendix. However, the main focus of attention in this appendix is on numerically stable algorithms for SVD computation, using rotations and reflections. We begin the discussion by considering plane rotations; planar reflections are examined later.

G.1 PLANE ROTATIONS

An algebraic tool that is fundamental to the cyclic Jacobi algorithm is the 2-by-2 *orthogonal,* but *nonsymmetric, matrix*

$$\boldsymbol{\Theta} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}, \tag{G.1}$$

where the *cosine rotation parameter*

$$c = \cos\theta \tag{G.2}$$

and the *sine rotation parameter*

$$s = \sin\theta \tag{G.3}$$

are real parameters and where the matrix has the trigonometric *constraint*

$$c^2 + s^2 = 1. \tag{G.4}$$

[In SVD terminology (and eigenanalysis, for that matter), the term "orthogonal matrix" is used in the context of *real* data, whereas the term "unitary matrix" is used for complex data.] We refer to the transformation $\mathbf{\Theta}$ as a plane rotation, because multiplication of a 2-by-1 data vector by $\mathbf{\Theta}$ amounts to a plane rotation of that vector. This property holds whether the data vector is premultiplied or postmultiplied by $\mathbf{\Theta}$.

The transformation of Eq. (G.1) is referred to as the *Jacobi rotation* in honor of Jacobi (1846), who proposed a method for reducing a symmetric matrix to diagonal form. Equation (G.1) is also referred to as the *Givens rotation.* In this book we will use the latter terminology, or simply "plane rotation."

To illustrate the nature of this plane rotation, consider the case of a real 2-by-1 vector:

$$\mathbf{a} = \begin{bmatrix} a_i \\ a_k \end{bmatrix}.$$

Premultiplication of the vector $\mathbf{a}$ by $\mathbf{\Theta}$ yields

$$\begin{aligned} \mathbf{x} &= \mathbf{\Theta a} \\ &= \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} a_i \\ a_k \end{bmatrix} \\ &= \begin{bmatrix} ca_i & + sa_k \\ -sa_i & + ca_k \end{bmatrix}. \end{aligned}$$

We may readily show, in view of the definitions of the rotation parameters c and s, that the vector $\mathbf{x}$ has the same Euclidean length as the vector $\mathbf{a}$. Moreover, given that the angle θ is positive, the transformation $\mathbf{\Theta}$ *rotates* the vector $\mathbf{a}$ in a clockwise direction into the new position defined by $\mathbf{x}$, as illustrated in Fig. G.1. Note that the vectors $\mathbf{a}$ and $\mathbf{x}$ *remain in the same (i, k) plane*—hence the name "plane rotation."

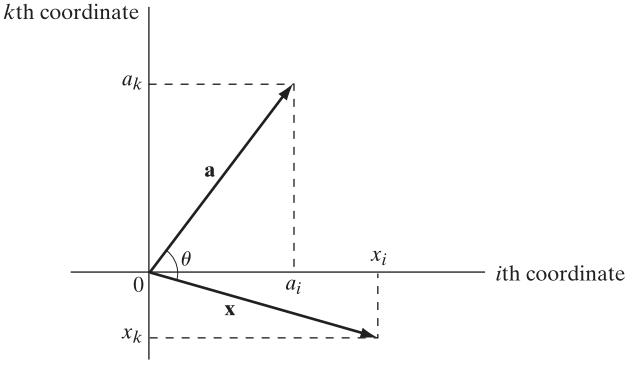


FIGURE G.1 Plane rotation of a real 2-by-1 vector.

G.2 TWO-SIDED JACOBI ALGORITHM

To pave the way for a development of the cyclic Jacobi algorithm, consider the simple case of a *real* 2-by-2 data matrix:

$$\mathbf{A} = \begin{bmatrix} a_{ii} & a_{ik} \\ a_{ki} & a_{kk} \end{bmatrix}. \tag{G.5}$$

We assume that $\mathbf{A}$ is nonsymmetric; that is, $a_{ki} \neq a_{ik}$. The requirement is to *diagonalize* this matrix. We do so by means of two plane rotations $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$, given by

$$\underbrace{\begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix}}_{\mathbf{\Theta}_1}^{\mathrm{T}} \underbrace{\begin{bmatrix} a_{ii} & a_{ik} \\ a_{ki} & a_{kk} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{bmatrix}}_{\mathbf{\Theta}_2} = \underbrace{\begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}}_{\text{diagonal matrix}}. \tag{G.6}$$

To design the two plane rotations indicated in Eq. (G.6), we proceed in two stages. Stage 1 transforms the data matrix $\mathbf{A}$ into a symmetric matrix; we refer to this stage as "symmetrization." Stage 2 diagonalizes the symmetric matrix resulting from stage 1; we refer to this second stage as "diagonalization." Of course, if the data matrix is symmetric to begin with, we proceed to stage 2 directly.

Stage 1: Symmetrization. To transform the 2-by-2 data matrix $\mathbf{A}$ into a symmetric matrix, we premultiply it by the transpose of a plane rotation $\mathbf{\Theta}$ and thus write

$$\underbrace{\begin{bmatrix} c & s \\ -s & c \end{bmatrix}}_{\mathbf{\Theta}^{\mathrm{T}}}^{\mathrm{T}} \underbrace{\begin{bmatrix} a_{ii} & a_{ik} \\ a_{ki} & a_{kk} \end{bmatrix}}_{\mathbf{A}} = \underbrace{\begin{bmatrix} y_{ii} & y_{ik} \\ y_{ki} & y_{kk} \end{bmatrix}}_{\mathbf{Y}}. \tag{G.7}$$

Expanding the left-hand side of Eq. (G.7) and equating terms, we get

$$y_{ii} = ca_{ii} - sa_{ki}, \tag{G.8}$$

$$y_{kk} = sa_{ik} + ca_{kk}, \tag{G.9}$$

$$y_{ik} = ca_{ik} - sa_{kk}, \tag{G.10}$$

and

$$y_{ki} = sa_{ii} + ca_{ki}. \tag{G.11}$$

The purpose of stage 1 is to compute the cosine–sine rotation pair (c, s) such that the 2-by-2 matrix $\mathbf{Y}$ produced by the plane rotation $\mathbf{\Theta}$ is symmetric. In other words, the elements y_{ik} and y_{ki} are required to equal each other.

We define a parameter ρ as the ratio of c to s; that is,

$$\rho = \frac{c}{s}. \tag{G.12}$$

We may relate ρ to the elements of the data matrix by setting $y_{ik} = y_{ki}$. Thus, imposing this definition on Eqs. (G.10) and (G.11), we obtain

$$\rho = \frac{a_{ii} + a_{kk}}{a_{ik} - a_{ki}}, \qquad a_{ki} \neq a_{ik}. \tag{G.13}$$

Next, we determine the value of s by eliminating c between Eqs. (G.4) and (G.12); hence,

$$s = \frac{\text{sgn}(\rho)}{\sqrt{1 + \rho^2}}. \tag{G.14}$$

The computation of c and s thus proceeds as follows:

- Use Eq. (G.13) to evaluate ρ.
- Use Eq. (G.14) to evaluate the sine parameter s.
- Use Eq. (G.12) to evaluate the cosine parameter c.

If $\mathbf{A}$ is symmetric to begin with, then $a_{ki} = a_{ik}$, in which case we have $s = 0$ and $c = 1$; that is, stage 1 is bypassed.

Stage 2: Diagonalization. The purpose of stage 2 is to diagonalize the symmetric matrix $\mathbf{Y}$ produced in stage 1. To do so, we premultiply and postmultiply it by $\mathbf{\Theta}_2^{\mathrm{T}}$ and $\mathbf{\Theta}_2$, respectively, where $\mathbf{\Theta}_2$ is a second plane rotation to be determined. This operation is simply an orthogonal similarity transformation applied to a symmetric matrix. We may thus write

$$\underbrace{\begin{bmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{bmatrix}^{\mathrm{T}}}_{\mathbf{\Theta}_2^{\mathrm{T}}} \underbrace{\begin{bmatrix} y_{ii} & y_{ik} \\ y_{ki} & y_{kk} \end{bmatrix}}_{\mathbf{Y}} \underbrace{\begin{bmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{bmatrix}}_{\mathbf{\Theta}_2} = \underbrace{\begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}}_{\mathbf{D}}, \tag{G.15}$$

where $y_{ik} = y_{ki}$. Expanding the left-hand side of Eq. (G.15) and then equating the respective diagonal terms, we get

$$d_1 = c_2^2 y_{ii} - 2c_2 s_2 y_{ki} + s_2^2 y_{kk} \tag{G.16}$$

and

$$d_2 = s_2^2 y_{ii} - 2c_2 s_2 y_{ki} + s_2^2 y_{kk}. \tag{G.17}$$

Now, let o_1 and o_2 denote the off-diagonal terms of the 2-by-2 matrix formed by carrying out the matrix multiplications indicated on the left-hand side of Eq. (G.15). From symmetry considerations, we have

$$o_1 = o_2. \tag{G.18}$$

Evaluating these off-diagonal terms and equating them to zero for diagonalization, we get

$$0 = (y_{ii} - y_{kk}) - \left(\frac{s_2}{c_2}\right) y_{ki} + \left(\frac{c_2}{s_2}\right) y_{ki}. \tag{G.19}$$

Equation (G.19) suggests that we introduce the following two definitions:

$$t = \frac{s_2}{c_2} \tag{G.20}$$

and

$$\zeta = \frac{y_{kk} - y_{ii}}{2y_{ki}}. \tag{G.21}$$

Accordingly, we may rewrite Eq. (G.19) as

$$t^2 + 2\zeta t - 1 = 0. \tag{G.22}$$

Equation (G.22) is quadratic in the parameter t; it therefore has two possible solutions, yielding the following different plane rotations:

1. *Inner rotation,* for which we have the solution

$$t = \frac{\text{sign}(\zeta)}{|\zeta| + \sqrt{1 + \zeta^2}}. \tag{G.23}$$

Having computed t, we may use Eqs. (G.4) and (G.20) to solve for c_2 and s_2, obtaining

$$c_2 = \frac{1}{\sqrt{1 + t^2}} \tag{G.24}$$

and

$$s_2 = tc_2. \tag{G.25}$$

We note from Eqs. (G.2), (G.3), and (G.20) that the rotation angle θ_2 is related to t by

$$\theta_2 = \arctan(t). \tag{G.26}$$

Hence, adoption of the solution given in Eq. (G.23) produces a plane rotation $\mathbf{\Theta}_2$ for which $|\theta_2|$ lies in the interval $[0, \pi/4]$; this rotation is therefore called an *inner rotation.* The computation proceeds as follows:

(a) Use Eq. (G.21) to compute ζ.
(b) Use Eq. (G.23) to compute t.
(c) Use Eqs. (G.24) and (G.25) to compute c_2 and s_2, respectively.

If the original matrix $\mathbf{A}$ is diagonal, then $a_{ik} = a_{ki} = 0$, in which case the angle $\theta_2 = 0$, so the matrix remains unchanged.

2. *Outer rotation,* for which we have the solution

$$t = -\text{sign}(\zeta)\left(|\zeta| + \sqrt{1 + \zeta^2}\right). \tag{G.27}$$

Having computed t, we may then evaluate c_2 and s_2 using Eqs. (G.24) and (G.25), respectively; we may do so because the derivations of these two equations are independent of the quadratic equation (G.22). In this second case, however, the use of Eq. (G.27) in Eq. (G.26) yields a plane rotation for which $|\theta_2|$ lies in the interval $[\pi/4, \pi/2]$. The rotation associated with the second solution is therefore referred to as an *outer rotation.* Note that if the original matrix $\mathbf{A}$ is diagonal, then $a_{ik} = a_{ki} = 0$, in which case $\theta_2 = \pi/2$. In this special case, the diagonal elements of the matrix are simply interchanged, as shown by

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_{ii} & 0 \\ 0 & a_{kk} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} a_{kk} & 0 \\ 0 & a_{ii} \end{bmatrix}. \tag{G.28}$$

Fusion of Rotations $\boldsymbol{\Theta}$ and $\boldsymbol{\Theta}_2$

Substituting the matrix $\mathbf{Y}$ of Eq. (G.7) into Eq. (G.15) and comparing the resulting equation with Eq. (G.6), we deduce the following definition for $\boldsymbol{\Theta}_1$ in terms of $\boldsymbol{\Theta}$ (determined in the symmetrization stage) and $\boldsymbol{\Theta}_2$ (determined in the diagonalization stage):

$$\boldsymbol{\Theta}_1^{\mathrm{T}} = \boldsymbol{\Theta}_2^{\mathrm{T}} \boldsymbol{\Theta}^{\mathrm{T}}.$$

Equivalently,

$$\boldsymbol{\Theta}_1 = \boldsymbol{\Theta} \boldsymbol{\Theta}_2. \tag{G.29}$$

In other words, in terms of the cosine–sine rotation parameters, we have

$$\underbrace{\begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix}}_{\boldsymbol{\Theta}_1} = \underbrace{\begin{bmatrix} c & s \\ -s & c \end{bmatrix}}_{\boldsymbol{\Theta}} \underbrace{\begin{bmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{bmatrix}}_{\boldsymbol{\Theta}_2}. \tag{G.30}$$

Expanding and equating terms, we thus obtain

$$c_1 = cc_2 - ss_2 \tag{G.31}$$

and

$$s_1 = sc_2 + cs_2. \tag{G.32}$$

For real data, from Eqs. (G.31) and (G.32), we find that the angles θ and θ_2, associated with the plane rotations $\boldsymbol{\Theta}$ and $\boldsymbol{\Theta}_2$, respectively, add to produce the angle θ_1 associated with $\boldsymbol{\Theta}_1$.

Two Special Cases

For reasons that will become apparent later in Section G.3, the Jacobi algorithm for computing the SVD has to be capable of handling two special cases:

Case 1: $a_{kk} = a_{ik} = 0$. In this case, we need to perform only the symmetrization of $\mathbf{A}$, as in

$$\begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} a_{ii} & 0 \\ a_{ki} & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} d_1 & 0 \\ 0 & 0 \end{bmatrix}. \tag{G.33}$$

Case 2: $a_{kk} = a_{ki} = 0$. In this case, we have

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a_{ii} & a_{ik} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{bmatrix} = \begin{bmatrix} d_1 & 0 \\ 0 & 0 \end{bmatrix}. \tag{G.34}$$

Additional Operations for Complex Data

The plane rotation described in Eq. (G.6) applies to real data only, because (to begin with) the cosine and sine parameters defining the rotation were all chosen to be real. To extend the application of this rotation to the more general case of complex data, we

have to perform additional operations on the data. At first sight, it may appear that we merely have to modify stage 1 (symmetrization) of the two-sided Jacobi algorithm so as to accommodate a complex 2-by-2 matrix. In reality, however, the issue of dealing with complex data (in the context of the Jacobi algorithm) is not so simple. The approach taken here is first to reduce the complex 2-by-2 data matrix of Eq. (G.5) to a real form and then proceed with the application of the two-sided Jacobi algorithm in the usual way. The *complex-to-real data reduction* is performed by following a two-stage procedure, described next.

Stage 1: Triangularization. Consider a complex 2-by-2 data matrix **A** having the form given in Eq. (G.5). Without loss of generality, we assume that the leading element a_{ii} is a positive real number. This assumption may be justified (if need be) by factoring out the exponential term $e^{j\theta_{ii}}$, where θ_{ii} is the phase angle of a_{ii}. The factorization has the effect of leaving inside the matrix a positive real term equal to the magnitude of a_{ii} and subtracting θ_{ii} from the phase angle of each of the remaining three complex terms in the matrix.

Let the matrix **A** so described be premultiplied by a 2-by-2 plane rotation for the purpose of triangularization, as shown by

$$\begin{bmatrix} c & s^* \\ -s & c \end{bmatrix} \begin{bmatrix} a_{ii} & a_{ik} \\ a_{ki} & a_{kk} \end{bmatrix} = \begin{bmatrix} \omega_{ii} & \omega_{ik} \\ 0 & \omega_{kk} \end{bmatrix}. \tag{G.35}$$

The cosine parameter c is real, as before, but the sine parameter s is now *complex*. To emphasize this point, we write

$$s = |s|e^{j\alpha}, \tag{G.36}$$

where $|s|$ is the magnitude of s and α is the phase angle of s. In addition, the (c, s) pair is required to satisfy the constraint

$$c^2 + |s|^2 = 1. \tag{G.37}$$

The objective is to choose the (c, s) pair so as to annihilate the kith (off-diagonal) term. To do this, we must satisfy the condition

$$-sa_{ii} + ca_{ki} = 0,$$

or, equivalently,

$$s = \frac{a_{ki}}{a_{ii}}\, c. \tag{G.38}$$

Substituting Eq. (G.38) into Eq. (G.37) and solving for the cosine parameter, we get

$$c = \frac{|a_{ii}|}{\sqrt{|a_{ii}|^2 + |a_{ki}|^2}}. \tag{G.39}$$

Note that, in Eq. (G.39), we have chosen to work with the *positive* real root for the cosine parameter c. Also, if a_{ki} is zero (i.e., the data matrix is upper triangular to begin with), then $c = 1$ and $s = 0$, in which case we may bypass stage 1. If, by the same token, a_{ik} is zero, we apply transposition and proceed to stage 2.

Having determined the values of c and s needed for the triangularization of the 2-by-2 matrix $\mathbf{A}$, we may now determine the elements of the resulting upper triangular matrix shown on the right-hand side of Eq. (G.35) as follows:

$$\omega_{ii} = ca_{ii} + s^*a_{ki}, \tag{G.40}$$

$$\omega_{ik} = ca_{ik} + s^*a_{kk}, \tag{G.41}$$

and

$$\omega_{kk} = -sa_{ik} + ca_{kk}, \tag{G.42}$$

where the asterisk denotes *complex conjugation.* Given that a_{ii} is positive real by design, the use of Eqs. (G.38) and (G.39) in Eq. (G.40) reveals that the diagonal element ω_{ii} is real and nonnegative; that is,

$$\omega_{ii} \geq 0. \tag{G.43}$$

In general, however, the remaining two elements ω_{ik} and ω_{kk} of the upper triangular matrix on the right-hand side of Eq. (G.35) are complex valued.

Stage 2: Phase Cancellation. To reduce the complex elements ω_{ik} and ω_{kk} to real form, we premultiply and postmultiply the upper triangular matrix on the right-hand side of Eq. (G.35) by a pair of *phase-cancelling diagonal matrices* as follows:

$$\begin{bmatrix} e^{-j\beta} & 0 \\ 0 & e^{-j\gamma} \end{bmatrix} \begin{bmatrix} \omega_{ii} & \omega_{ik} \\ 0 & \omega_{kk} \end{bmatrix} \begin{bmatrix} e^{j\beta} & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \omega_{ii} & |\omega_{ik}| \\ 0 & |\omega_{kk}| \end{bmatrix}. \tag{G.44}$$

The *rotation angles* β and γ of the premultiplying matrix are chosen so as to cancel the phase angles of ω_{ik} and ω_{kk}, respectively, as shown by

$$\beta = \arg(\omega_{ik}) \tag{G.45}$$

and

$$\gamma = \arg(\omega_{kk}). \tag{G.46}$$

The postmultiplying matrix is included so as to *correct* for the phase change in the element ω_{ii} produced by the premultiplying matrix. In other words, the combined process of premultiplication and postmultiplication in Eq. (G.44) leaves the diagonal element ω_{ii} unchanged.

Stage 2 thus yields an upper triangular matrix whose three nonzero elements are all real and nonnegative. Note that the procedure for reducing the complex 2-by-2 matrix $\mathbf{A}$ to a real upper triangular form requires four degrees of freedom, namely, the (c, s) pair and the angles β and γ. The way is now paved for us to proceed with the application of the Jacobi method for a real 2-by-2 matrix, as described earlier in the section.

Properties of the Givens Rotation

We conclude the section by summarizing the properties of the Givens (plane) rotation

$$\boldsymbol{\Theta} = \begin{bmatrix} c & s^* \\ -s & c \end{bmatrix}.$$

Property 1. *The cosine parameter c is always real, but the sine parameter s is complex valued when dealing with complex data.*

Property 2. *The parameters c and s are always constrained by the trigonometric relation*

$$c^2 + |s|^2 = 1.$$

Property 3. *The Givens rotation is non-Hermitian*; that is,

$$\boldsymbol{\Theta}^{\mathrm{H}} \neq \boldsymbol{\Theta}.$$

where the superscript H denotes *Hermitian transposition* (i.e., transposition combined with complex conjugation).

Property 4. *The Givens rotation is unitary;* that is,

$$\boldsymbol{\Theta}^{-1} = \boldsymbol{\Theta}^{\mathrm{H}}.$$

Property 5. *The Givens rotation is length preserving*; that is,

$$\|\boldsymbol{\Theta}\mathbf{x}\| = \|\mathbf{x}\|,$$

where $\mathbf{x}$ is an arbitrary 2-by-2 vector.

G.3 CYCLIC JACOBI ALGORITHM

We are now ready to describe the *cyclic Jacobi algorithm*, or *generalized Jacobi algorithm*, for a square data matrix by solving an appropriate sequence of 2-by-2 singular-value decomposition problems. The description will be presented for real data. To deal with complex data, we incorporate the complex-to-real data reduction developed in the latter part of the preceding section.

Let $\boldsymbol{\Theta}_1(i,k)$ denote a plane rotation in the (i,k) plane, where $k > i$. The matrix $\boldsymbol{\Theta}_1(i,k)$ is the same as the M-by-M identity matrix, except for the four strategic elements located on rows i, k and columns i, k:

$$\boldsymbol{\Theta}_1(i,k) = \begin{bmatrix} 1 & & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & \ddots & \vdots \\ 0 & & c_1 & \cdots & s_1 & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & & -s_1 & \cdots & c_1 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & \ddots & \vdots \\ 0 & & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{matrix} \\ \\ \leftarrow \text{ row } i \\ \\ \leftarrow \text{ row } k \\ \\ \\ \end{matrix} \qquad \text{(G.47)}$$

$$\begin{matrix} & & \uparrow & & \uparrow & & \\ & & \text{column } i & & \text{column } k & & \end{matrix}$$

Let $\boldsymbol{\Theta}_2(i,k)$ denote a second plane rotation in the (i,k) plane that is similarly defined; the dimension of this second transformation is also M. The Jacobi transformation of the data matrix $\mathbf{A}$ is thus described by

$$\mathbf{T}_{ik} : \mathbf{A} \leftarrow \boldsymbol{\Theta}_1^{\mathrm{T}}(i,k)\mathbf{A}\boldsymbol{\Theta}_2(i,k). \qquad \text{(G.48)}$$

The Jacobi rotations $\mathbf{\Theta}_1(i, k)$ and $\mathbf{\Theta}_2(i, k)$ are designed to annihilate the (i, k) and (k, i) elements of $\mathbf{A}$. Accordingly, the transformation $\mathbf{T}_{ik}$ produces a matrix $\mathbf{X}$ (equal to the updated value of $\mathbf{A}$) that is more diagonal than the original $\mathbf{A}$, in the sense that

$$\text{off}(\mathbf{X}) = \text{off}(\mathbf{A}) - a_{ik}^2 - a_{ki}^2, \tag{G.49}$$

where off($\mathbf{A}$) is the *norm of the off-diagonal elements*:

$$\text{off}(\mathbf{A}) = \sum_{i=1}^{M} \sum_{\substack{k=1 \\ k \neq 1}}^{M} a_{ik}^2 \qquad \text{for } \mathbf{A} = \{a_{ik}\}. \tag{G.50}$$

In the cyclic Jacobi algorithm, the transformation of Eq. (G.48) is applied for a total of $m = M(M-1)/2$ different index pairs ("pivots") that are selected in some fixed order. Such a sequence of m transformations is called a *sweep*. The construction of a sweep may be *cyclic by rows* or *cyclic by columns,* as illustrated shortly in Example 1. In either case, we obtain a new matrix $\mathbf{A}$ after each sweep, for which we compute off($\mathbf{A}$). If, on the one hand, off($\mathbf{A}$) $\leq \delta$, where δ is some small machine-dependent number, we stop the computation. If, on the other hand, off($\mathbf{A}$) $> \delta$, we repeat the computation. For typical values of δ [e.g., $\delta = 10^{-12}$ off($\mathbf{A}_0$), where $\mathbf{A}_0$ is the original matrix], the algorithm converges in about 4 to 10 sweeps for values of M in the range of 4 to 2000.

As far as we know, the row ordering or the column ordering is the only ordering that guarantees convergence of the Jacobi cyclic algorithm.[1] By "convergence," we mean that

$$\text{off}(\mathbf{A}^{(k)}) \to 0 \qquad \text{as } k \to \infty, \tag{G.51}$$

where $\mathbf{A}^{(k)}$ is the M-by-M matrix computed after sweep number k.

EXAMPLE 1

Consider a 4-by-4 *real* matrix $\mathbf{A}$. With the matrix dimension $M = 4$, we have a total of six orderings in each sweep. A sweep of orderings cyclic by rows is represented by

$$\mathbf{T}_\text{R} = \mathbf{T}_{34}\mathbf{T}_{24}\mathbf{T}_{23}\mathbf{T}_{14}\mathbf{T}_{13}\mathbf{T}_{12}.$$

A sweep of orderings cyclic by columns is represented by

$$\mathbf{T}_\text{C} = \mathbf{T}_{34}\mathbf{T}_{24}\mathbf{T}_{14}\mathbf{T}_{23}\mathbf{T}_{13}\mathbf{T}_{12}.$$

It is easily checked that the transformations $\mathbf{T}_{ik}$ and $\mathbf{T}_{pq}$ *commute* if two conditions hold:

1. The index i is neither p nor q.
2. The index k is neither p nor q.

Accordingly, we find that the transformations $\mathbf{T}_R$ and $\mathbf{T}_C$ are indeed equivalent, so they should be.

---

[1] A proof of the convergence of the Jacobi cyclic algorithm, based on row ordering or column ordering, is given in Forsythe and Henrici (1960). Subsequently, Luk and Park (1989) proved that many of the orderings used in a parallel implementation of the algorithm are equivalent to the row ordering and thus guarantee convergence as well.

Consider next the application of the transformation $\mathbf{T}_R$ (obtained from the sweep of orderings cyclic by rows) to the data matrix $\mathbf{A}$. In particular, using the rotation of Eq. (G.48), we may write the transformations

$$\mathbf{T}_{12} : \mathbf{A} \leftarrow \boldsymbol{\Theta}_1^{\mathrm{T}}(1,2)\mathbf{A}\boldsymbol{\Theta}_2(1,2),$$

$$\mathbf{T}_{13}\mathbf{T}_{12} : \mathbf{A} \leftarrow \boldsymbol{\Theta}_3^{\mathrm{T}}(1,3)\boldsymbol{\Theta}_1^{\mathrm{T}}(1,2)\mathbf{A}\boldsymbol{\Theta}_2(1,2)\boldsymbol{\Theta}_4(1,3),$$

$$\mathbf{T}_{14}\mathbf{T}_{13}\mathbf{T}_{12} : \mathbf{A} \leftarrow \boldsymbol{\Theta}_5^{\mathrm{T}}(1,4)\boldsymbol{\Theta}_3^{\mathrm{T}}(1,3)\boldsymbol{\Theta}_1^{\mathrm{T}}(1,2)\mathbf{A}\boldsymbol{\Theta}_2(1,2)\boldsymbol{\Theta}_4(1,3)\boldsymbol{\Theta}_6(1,4),$$

and so on. The final step in this sequence of transformations may be written as

$$\mathbf{T}_{\mathrm{R}} : \mathbf{A} \leftarrow \mathbf{U}^{\mathrm{T}}\mathbf{A}\mathbf{V},$$

which defines the singular-value decomposition of the real data matrix $\mathbf{A}$. The orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$ are respectively defined by

$$\mathbf{U} = \boldsymbol{\Theta}_1(1,2)\boldsymbol{\Theta}_3(1,3)\boldsymbol{\Theta}_5(1,4)\boldsymbol{\Theta}_7(2,3)\boldsymbol{\Theta}_9(2,4)\boldsymbol{\Theta}_{11}(3,4)$$

and

$$\mathbf{V} = \boldsymbol{\Theta}_2(1,2)\boldsymbol{\Theta}_4(1,3)\boldsymbol{\Theta}_6(1,4)\boldsymbol{\Theta}_8(2,3)\boldsymbol{\Theta}_{10}(2,4)\boldsymbol{\Theta}_{12}(3,4).$$

Rectangular Data Matrix

Thus far, we have focused attention on the cyclic Jacobi algorithm for computing the singular-value decomposition of a square matrix. To handle the more general case of a rectangular matrix, we may extend the use of this algorithm by proceeding as follows: Consider first the case of a K-by-M real data matrix $\mathbf{A}$ for which K is greater than M. We generate a square matrix by appending $K - M$ columns of zeros to $\mathbf{A}$. We may thus write

$$\widetilde{\mathbf{A}} = [\mathbf{A}, \mathbf{O}]. \tag{G.52}$$

We refer to $\widetilde{\mathbf{A}}$ as the *augmented data matrix*. We then proceed as before by applying the cyclic Jacobi algorithm to the K-by-K matrix $\widetilde{\mathbf{A}}$. In performing this computation, we require the use of special case 1 described in Eq. (G.33). In any event, we emerge with the factorization

$$\mathbf{U}^{\mathrm{T}}[\mathbf{A}, \mathbf{O}]\begin{bmatrix}\mathbf{V} & \mathbf{O}\\ \mathbf{O} & \mathbf{I}\end{bmatrix} = \mathrm{diag}(\sigma_1, \ldots, \sigma_M, 0, \ldots, 0). \tag{G.53}$$

The desired factorization of the original data matrix $\mathbf{A}$ is obtained by writing

$$\mathbf{U}^{\mathrm{T}}\mathbf{A}\mathbf{V} = \mathrm{diag}(\sigma_1, \ldots, \sigma_M). \tag{G.54}$$

If the dimension M of matrix $\mathbf{A}$ is greater than K, we augment the matrix by adding $M - K$ rows of zeros, we may thus write

$$\widetilde{\mathbf{A}} = \begin{bmatrix}\mathbf{A}\\ \mathbf{O}\end{bmatrix}. \tag{G.55}$$

We then treat the square matrix $\widetilde{\mathbf{A}}$ in a similar way as before. In this second situation, we require the use of special case 2 described in Eq. (G.34).

In the case of a complex rectangular data matrix $\mathbf{A}$, we may proceed in a fashion similar to that just described, except for a change in the characterization of matrices $\mathbf{U}$ and $\mathbf{V}$. For a real data matrix, the matrices $\mathbf{U}$ and $\mathbf{V}$ are both orthogonal, whereas for a complex data matrix, they are both unitary.

The strategy of matrix augmentation described herein represents a straightforward extension of the cyclic Jacobi algorithm for a square matrix. A drawback of this approach, however, is that the algorithm becomes too inefficient if the dimension K of matrix $\mathbf{A}$ is much greater than the dimension M or vice versa.[2]

G.4 HOUSEHOLDER TRANSFORMATION

We turn next to the *Householder transformation*, or the *Householder matrix*, which is so named in recognition of its originator (Householder, 1958a, b, 1964). To simplify the discussion, we consider a real M-by-1 vector whose Euclidean norm is

$$\|\mathbf{u}\| = (\mathbf{u}^{\mathrm{T}}\mathbf{u})^{1/2}.$$

Then, the Householder transformation is defined by the M-by-M matrix

$$\mathbf{Q} = \mathbf{I} - \frac{2\mathbf{u}\mathbf{u}^{\mathrm{T}}}{\|\mathbf{u}\|^2}, \tag{G.56}$$

where $\mathbf{I}$ is the M-by-M identity matrix.

For a geometric interpretation of the Householder transformation, consider an M-by-1 vector $\mathbf{x}$ premultiplied by the matrix $\mathbf{Q}$:

$$\begin{aligned}\mathbf{Q}\mathbf{x} &= \left(\mathbf{I} - \frac{2\mathbf{u}\mathbf{u}^{\mathrm{T}}}{\|\mathbf{u}\|^2}\right)\mathbf{x} \\ &= \mathbf{x} - \frac{2\mathbf{u}^{\mathrm{T}}\mathbf{x}}{\|\mathbf{u}\|^2}\,\mathbf{u}.\end{aligned} \tag{G.57}$$

By definition, the *projection* of $\mathbf{x}$ onto $\mathbf{u}$ is given by

$$\mathbf{P}_u(\mathbf{x}) = \frac{\mathbf{u}^{\mathrm{T}}\mathbf{x}}{\|\mathbf{u}\|^2}\mathbf{u}. \tag{G.58}$$

This projection is illustrated in Fig. G.2. In this figure, we have also included the vector representation of the product $\mathbf{Q}\mathbf{x}$. We see that $\mathbf{Q}\mathbf{x}$ is the mirror-image *reflection* of the vector $\mathbf{x}$ with respect to the hyperplane span $\{\mathbf{u}\}^{\perp}$, which is perpendicular to the

[2] An alternative approach that overcomes this difficulty is to proceed as follows (Luk, 1986):

1. *Triangularize* the K-by-M data matrix $\mathbf{A}$ by performing a QR decomposition, defined by

$$\mathbf{A} = \mathbf{Q}\begin{bmatrix}\mathbf{R} \\ \mathbf{O}\end{bmatrix},$$

where $\mathbf{Q}$ is a K-by-K orthogonal matrix and $\mathbf{R}$ is an M-by-M upper triangular matrix.

2. Diagonalize the matrix $\mathbf{R}$ using the cyclic Jacobi algorithm.
3. Combine the results of steps 1 and 2.

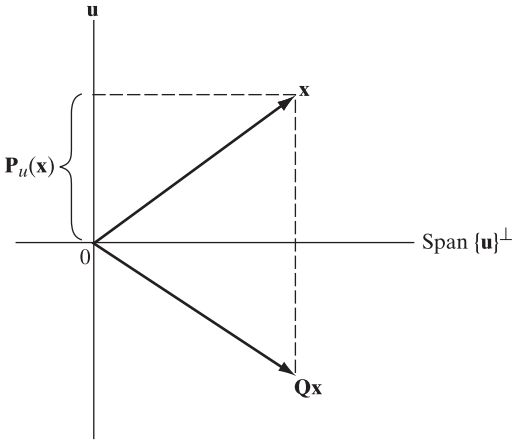


FIGURE G.2 Geometric interpretation of the Householder transformation.

vector **u**. It is for this reason that the Householder transformation is also known as the *Householder reflection*.[3]

The Householder transformation, defined in Eq. (G.56) for real-valued data, has the following properties:

Property 1. *The Householder transformation is symmetric*; that is,

$$\mathbf{Q}^{\mathrm{T}} = \mathbf{Q}. \tag{G.59}$$

Property 2. *The Householder transformation is orthogonal*; that is,

$$\mathbf{Q}^{-1} = \mathbf{Q}^{\mathrm{T}}. \tag{G.60}$$

Property 3. *The Householder transformation is length preserving*; that is,

$$\|\mathbf{Qx}\| = \|\mathbf{x}\|. \tag{G.61}$$

Property 3 is illustrated in Fig. G.2, where we see that the vector **x** and its reflection **Qx** have exactly the same length.

Property 4. *If two vectors undergo the same Householder transformation, their inner product remains unchanged.*

Consider any three vectors **x**, **y**, and **u**. Let the Householder matrix **Q** be defined in terms of the vector **u**, as in Eq. (G.56). Let the remaining two vectors **x** and **y** be transformed by **Q**, yielding **Qx** and **Qy**, respectively. Then, the inner product of these two transformed vectors is

$$\begin{aligned}(\mathbf{Qx})^{\mathrm{T}}(\mathbf{Qy}) &= \mathbf{x}^{\mathrm{T}}\mathbf{Q}^{\mathrm{T}}\mathbf{Qy} \\ &= \mathbf{x}^{\mathrm{T}}\mathbf{y},\end{aligned} \tag{G.62}$$

where we have made use of Property 2. Hence, the transformed vectors **Qx** and **Qy** have the same inner product as the original vectors **x** and **y**.

[3]For a tutorial review of the Householder transformation and its use in adaptive signal processing, see Steinhardt (1988).

Property 4 has important practical implications in the numerical solution of linear least-squares problems. Specifically, Householder transformations are used to reduce the given data matrix to a *sparse* matrix (i.e., one that consists mostly of zeros) that is "equivalent" to the original data matrix in some mathematical sense. Needless to say, the particular form of matrix sparseness used depends on the application of interest. Whatever the application, however, the form of data reduction described here is used to simplify the numerical computations involved in solving the problem. In this context, a popular form of data reduction is *triangularization*, referring to the reduction of a full data matrix to an upper triangular one. Given this form of data reduction, we may simply use *Gaussian elimination* to perform the matrix inversion and thereby compute the least-squares solution to the problem.

Property 5. *Given the Householder transformation* $\mathbf{Q}$, *the transformed vector* $\mathbf{Qx}$ *is a reflection of* $\mathbf{x}$ *above the hyperplane perpendicular to the vector* $\mathbf{u}$ *involved in the definition of* $\mathbf{Q}$.

This property is merely a restatement of Eq. (G.57). The following two limiting cases of Property 5 are especially noteworthy:

1. The vector $\mathbf{x}$ is a scalar multiple of $\mathbf{u}$. In this case, Eq. (G.57) simplifies to
$$\mathbf{Qx} = -\mathbf{x}.$$
2. The vector $\mathbf{x}$ is orthogonal to $\mathbf{u}$; that is, the inner product of $\mathbf{x}$ and $\mathbf{u}$ is zero. In this second case, Eq. (G.57) reduces to
$$\mathbf{Qx} = \mathbf{x}.$$

Property 6. *Let* $\mathbf{x}$ *be any nonzero M-by-1 vector with Euclidean norm* $\|\mathbf{x}\|$. *Let the M-by-1 vector* $\mathbf{1}$ *denote the first coordinate vector*

$$\mathbf{1} = [1, 0, \ldots, 0]^{\mathrm{T}}. \tag{G.63}$$

Then, there exists a Householder transformation $\mathbf{Q}$, *defined by the vector*

$$\mathbf{u} = \mathbf{x} - \|\mathbf{x}\|\mathbf{1} \tag{G.64}$$

such that the transformed vector $\mathbf{Qx}$ *corresponding to* $\mathbf{u}$ *is a linear multiple of the vector* $\mathbf{1}$.

With the vector $\mathbf{u}$ assigned the value in Eq. (G.64), we have

$$\begin{aligned} \|\mathbf{u}\|^2 &= \mathbf{u}^{\mathrm{T}}\mathbf{u} \\ &= (\mathbf{x} - \|\mathbf{x}\|\mathbf{1})^{\mathrm{T}}(\mathbf{x} - \|\mathbf{x}\|\mathbf{1}) \\ &= 2\|\mathbf{x}\|^2 - 2\|\mathbf{x}\|x_1 \\ &= 2\|\mathbf{x}\|(\|\mathbf{x}\| - x_1), \end{aligned} \tag{G.65}$$

where x_1 is the first element of the vector $\mathbf{x}$. Similarly, we may write

$$\begin{aligned} \mathbf{u}^{\mathrm{T}}\mathbf{x} &= (\mathbf{x} - \|\mathbf{x}\|\mathbf{1})^{\mathrm{T}}\mathbf{x} \\ &= \|\mathbf{x}\|^2 - \|\mathbf{x}\|x_1 \\ &= \|\mathbf{x}\|(\|\mathbf{x}\| - x_1). \end{aligned} \tag{G.66}$$

Accordingly, substituting Eqs. (G.65) and (G.66) into Eq. (G.57), we find that the transformed vector $\mathbf{Qx}$ corresponding to the defining vector $\mathbf{u}$ of Eq. (G.64) is given by

$$\begin{aligned}\mathbf{Qx} &= \mathbf{x} - \mathbf{u} \\ &= \mathbf{x} - (\mathbf{x} - \|\mathbf{x}\|\mathbf{1}) \\ &= \|\mathbf{x}\|\mathbf{1},\end{aligned} \tag{G.67}$$

which establishes Property 6 described in Eq. (G.64).

From Eq. (G.65), we observe that the Euclidean norm of $\mathbf{x}$ has to satisfy the condition

$$\|\mathbf{x}\| > |x_1|. \tag{G.68}$$

This condition merely says that not only the first element of $\mathbf{x}$ but also at least one other element must be nonzero. Then, the vector $\mathbf{u}$ defined by Eq. (G.64) is indeed effective.

Property 6 makes the Householder transformation a very powerful computational tool. Given a vector $\mathbf{x}$, we may use Eq. (G.64) to define the vector $\mathbf{u}$ such that the corresponding Householder transformation $\mathbf{Q}$ annihilates all the M elements of the vector $\mathbf{x}$, except for the first one. This result is equivalent to the application of $M-1$ plane rotations, with a minor difference: The determinant of the Householder matrix $\mathbf{Q}$ defined in Eq. (G.56) is

$$\begin{aligned}\det(\mathbf{Q}) &= \det\left(\mathbf{I} - \frac{2\mathbf{u}\mathbf{u}^{\mathrm{T}}}{\|\mathbf{u}\|^2}\right) \\ &= -1.\end{aligned} \tag{G.69}$$

Hence, the Householder transformation reverses the orientation of the configuration.

Note that it is through Properties 1, 5, and 6 that the Householder transformation distinguishes itself from the Givens rotation. The basic differences between these two unitary transformations are illustrated by comparing Fig. G.1 with Fig. G.2.

G.5 THE QR ALGORITHM

The starting point in the development of the *QR algorithm* for computating the SVD is that of finding a class of orthogonal matrices that preserve the singular values of a data matrix $\mathbf{A}$. In this context, assuming real-valued data, the matrix $\mathbf{A}$ is said to be *orthogonally equivalent* to another matrix $\mathbf{B}$ if

$$\mathbf{B} = \mathbf{PAQ}, \tag{G.70}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are orthogonal matrices; that is,

$$\mathbf{P}^{\mathrm{T}}\mathbf{P} = \mathbf{I}$$

and

$$\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{I}.$$

Consequently,

$$\begin{aligned}\mathbf{B}^{\mathrm{T}}\mathbf{B} &= \mathbf{Q}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{P}\mathbf{A}\mathbf{Q} \\ &= \mathbf{Q}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{Q}.\end{aligned} \tag{G.71}$$

Postmultiplying the correlation matrix $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ by an orthogonal matrix $\mathbf{Q}$ and premultiplying it by the transpose of the matrix $\mathbf{Q}$ leaves the eigenvalues of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ unchanged. Accordingly, the correlation matrices $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ and $\mathbf{B}^{\mathrm{T}}\mathbf{B}$, or, more simply, the matrices $\mathbf{A}$ and $\mathbf{B}$ themselves, are said to be *eigenequivalent*.

The purpose of using the transformation defined in Eq. (G.70) is to reduce the data matrix $\mathbf{A}$ to *upper bidiagonal* form, with eigenequivalence maintained, for which Householder transformations are well suited. The reduced data matrix $\mathbf{B}$ is said to be upper bidiagonal if all of its elements, except those on the main diagonal and the superdiagonal, are zero; that is, the ijth element of $\mathbf{B}$ is

$$b_{ij} = 0 \qquad \text{whenever} \quad i > j \text{ or } j > i + 1. \tag{G.72}$$

With the data matrix $\mathbf{A}$ reduced to upper bidiagonal form, the next step is to apply Householder bidiagonalization, as described next.

Householder Bidiagonalization

Consider a K-by-M data matrix $\mathbf{A}$, where $K \geq M$. Let $\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_M$ denote a set of K-by-K Householder matrices, and let $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_{M-2}$ denote another set of M-by-M Householder matrices. To reduce the data matrix $\mathbf{A}$ to upper bidiagonal form, we determine the products of Householder matrices

$$\mathbf{Q}_{\mathrm{B}} = \begin{cases} \mathbf{Q}_1 \mathbf{Q}_2 \ldots \mathbf{Q}_{M-1}, & K = M \\ \mathbf{Q}_1 \mathbf{Q}_2 \ldots \mathbf{Q}_M, & K > M \end{cases} \tag{G.73}$$

and

$$\mathbf{P}_{\mathrm{B}} = \mathbf{P}_1 \mathbf{P}_2 \ldots \mathbf{P}_{M-2} \tag{G.74}$$

such that

$$\mathbf{Q}_{\mathrm{B}}^{\mathrm{T}}\mathbf{A}\mathbf{P}_{\mathrm{B}} = \mathbf{B} = \begin{bmatrix} d_1 & f_2 & & \mathbf{O} \\ & d_2 & \ddots & \\ & & \ddots & f_M \\ \mathbf{O} & & & d_M \\ \hdashline & \mathbf{O} & & \end{bmatrix} \left.\vphantom{\begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix}}\right\} (K - M)\text{-by-}M \text{ null matrix.} \tag{G.75}$$

For $K > M$, premultiplication of the data matrix $\mathbf{A}$ by the Householder matrices $\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_M$ corresponds to reflecting the respective columns of $\mathbf{A}$, whereas postmultiplication by the Householder matrices $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_{M-2}$ corresponds to reflecting the respective rows of $\mathbf{A}$. The desired upper bidiagonal form is attained by "ping-ponging" column and row reflections. Note that for $K > M$, the number of Householder matrices constituting $\mathbf{Q}_{\mathrm{B}}$ is M, whereas those constituting $\mathbf{P}_{\mathrm{B}}$ number $M - 2$. Note also

that, by construction, the matrix product $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_{M-2}$ does *not* alter the first column of any matrix that it postmultiplies.

We illustrate this idea of data reduction by way of an example.

EXAMPLE 2

Consider a 5-by-4 data matrix $\mathbf{A}$ written in expanded form as

$$\mathbf{A} = \begin{bmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{bmatrix},$$

where the x's denote nonzero matrix entries. The upper bidiagonalization of $\mathbf{A}$ proceeds as follows: First, $\mathbf{Q}_1$ is chosen so that $\mathbf{Q}_1^{\mathrm{T}}\mathbf{A}$ has zeros in the positions shown:

$$\begin{bmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \otimes & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \otimes & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \otimes & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \otimes & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{bmatrix}.$$

Thus,

$$\mathbf{Q}_1^{\mathrm{T}}\mathbf{A} = \begin{bmatrix} \mathrm{x} & \mathrm{x} & \otimes & \otimes \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{bmatrix}. \tag{G.76}$$

Next, $\mathbf{P}_1$ is chosen so that $\mathbf{Q}_1^{\mathrm{T}}\mathbf{A}\mathbf{P}_1$ has zeros in the positions distinguished in the first row of $\mathbf{Q}_1^{\mathrm{T}}\mathbf{A}$, as in Eq. (G.76). Hence,

$$\mathbf{Q}_1^{\mathrm{T}}\mathbf{A}\mathbf{P}_1 = \left[\begin{array}{c|ccc} \mathrm{x} & \mathrm{x} & 0 & 0 \\ \hline 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{array}\right]. \tag{G.77}$$

Note that $\mathbf{P}_1$ does not affect the first column of the matrix $\mathbf{Q}_1^{\mathrm{T}}\mathbf{A}$.

The data reduction is continued by operating on the trailing 4-by-3 submatrix of $\mathbf{Q}^{\mathrm{T}}\mathbf{A}\mathbf{P}_1$ that has nonzero entries. Specifically, we choose $\mathbf{Q}_2$ and $\mathbf{P}_2$ such that

$$\mathbf{Q}_2^{\mathrm{T}}\mathbf{Q}_1^{\mathrm{T}}\mathbf{A}\mathbf{P}_1\mathbf{P}_2 = \left[\begin{array}{cc:cc} \mathrm{x} & \mathrm{x} & 0 & 0 \\ 0 & \mathrm{x} & \mathrm{x} & 0 \\ \hdashline 0 & 0 & \mathrm{x} & \mathrm{x} \\ 0 & 0 & \mathrm{x} & \mathrm{x} \\ 0 & 0 & \mathrm{x} & \mathrm{x} \end{array}\right]. \tag{G.78}$$

Next, we operate on the trailing 3-by-2 submatrix of $\mathbf{Q}_2^{\mathrm{T}}\mathbf{Q}_1^{\mathrm{T}}\mathbf{A}\mathbf{P}_1\mathbf{P}_2$ that has nonzero entries.

Specifically, we choose $\mathbf{Q}_3$ such that

$$\mathbf{Q}_3^{\mathrm{T}}\mathbf{Q}_2^{\mathrm{T}}\mathbf{Q}_1^{\mathrm{T}}\mathbf{A}\mathbf{P}_1\mathbf{P}_2 = \left[\begin{array}{ccc:c} \mathrm{x} & \mathrm{x} & 0 & 0 \\ 0 & \mathrm{x} & \mathrm{x} & 0 \\ 0 & 0 & \mathrm{x} & \mathrm{x} \\ \hdashline 0 & 0 & 0 & \mathrm{x} \\ 0 & 0 & 0 & \mathrm{x} \end{array}\right]. \tag{G.79}$$

Finally, we choose $\mathbf{Q}_4$ to operate on the trailing 2-by-1 submatrix of $\mathbf{Q}_3^{\mathrm{T}}\mathbf{Q}_2^{\mathrm{T}}\mathbf{Q}_1^{\mathrm{T}}\mathbf{A}\mathbf{P}_1\mathbf{P}_2$ such that we may write

$$\mathbf{B} = \mathbf{Q}_4^{\mathrm{T}}\mathbf{Q}_3^{\mathrm{T}}\mathbf{Q}_2^{\mathrm{T}}\mathbf{Q}_1^{\mathrm{T}}\mathbf{A}\mathbf{P}_1\mathbf{P}_2 = \begin{bmatrix} \mathrm{x} & \mathrm{x} & 0 & 0 \\ 0 & \mathrm{x} & \mathrm{x} & 0 \\ 0 & 0 & \mathrm{x} & \mathrm{x} \\ 0 & 0 & 0 & \mathrm{x} \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{G.80}$$

This completes the upper bidiagonalization of the data matrix $\mathbf{A}$.

The Golub–Kahan Step

The bidiagonalization of the data matrix $\mathbf{A}$ is followed by an *iterative process* that reduces the matrix further to *diagonal* form. Referring to Eq. (G.75), we see that the matrix $\mathbf{B}$ resulting from the bidiagonalization of $\mathbf{A}$ is zero below the Mth row. Evidently, the last $K - M$ rows of zeros in the matrix $\mathbf{B}$ do *not* contribute to the singular values of the original data matrix $\mathbf{A}$. Accordingly, it is convenient to delete the last $K - M$ rows of matrix $\mathbf{B}$ and thus treat it as a square matrix with dimension M. The basis of the diagonalization of matrix $\mathbf{B}$ is the *Golub–Kahan algorithm* (Golub & Kahan, 1965), which is an adaptation of the *QR algorithm* developed originally for solving the symmetric eigenvalue problem.[4]

Let $\mathbf{B}$ denote an M-by-M upper bidiagonal matrix having no zeros on its main diagonal or superdiagonal. The first adaptation cycle of the Golub–Kahan algorithm proceeds as follows (Golub & Kahan, 1965; Golub & Van Loan, 1996):

1. Identify the trailing 2-by-2 submatrix of the product $\mathbf{T} = \mathbf{B}^{\mathrm{H}}\mathbf{B}$, which has the form

$$\begin{bmatrix} d_{M-1}^2 + f_{M-1}^2 & d_{M-1}f_M \\ f_M d_{M-1} & d_M^2 + f_M^2 \end{bmatrix}, \tag{G.81}$$

where d_{M-1} and d_M are the trailing diagonal elements of matrix $\mathbf{B}$ and f_{M-1} and f_M are the trailing superdiagonal elements. [See the right-hand side of Eq. (G.75).] Let λ be the eigenvalue of this submatrix that is closer to $d_M^2 + f_M^2$; this particular eigenvalue λ is known as the *Wilkinson shift*.

2. Compute the Givens rotation parameters c_1 and s_1 such that

$$\begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} d_1^2 - \lambda \\ f_2 d_1 \end{bmatrix} = \begin{bmatrix} \star \\ 0 \end{bmatrix}, \tag{G.82}$$

[4]The explicit form of the QR algorithm is a variant of the QL algorithm discussed in Appendix E.

where d_1 and f_2 are, respectively, the leading main diagonal and superdiagonal elements of matrix $\mathbf{B}$. [See again the right-hand side of Eq. (G.75).] The element marked ☆ on the right-hand side of Eq. (G.82) indicates a nonzero element. Set

$$\boldsymbol{\Theta}_1^{\mathrm{T}} = \left[\begin{array}{cc:c} c_1 & s_1 & \\ -s_1 & c_1 & \mathbf{O} \\ \hdashline & \mathbf{O} & \mathbf{I} \end{array}\right]. \tag{G.83}$$

3. Apply the Givens rotation $\boldsymbol{\Theta}_1$ to matrix $\mathbf{B}$ directly. Since $\mathbf{B}$ is upper bidiagonal and $\boldsymbol{\Theta}_1$ is a rotation in the $(2,1)$-plane, it follows that the matrix product $\mathbf{B}\boldsymbol{\Theta}_1$ has the form (illustrated for the case of $M = 4$)

$$\mathbf{B}\boldsymbol{\Theta}_1 = \begin{bmatrix} \mathrm{x} & \mathrm{x} & 0 & 0 \\ z^{(1)} & \mathrm{x} & \mathrm{x} & 0 \\ 0 & 0 & \mathrm{x} & \mathrm{x} \\ 0 & 0 & 0 & \mathrm{x} \end{bmatrix},$$

where $z^{(1)}$ is a new element produced by the Givens rotation $\boldsymbol{\Theta}_1$.

4. Determine the sequence of Givens rotations $\mathbf{U}_1$, $\mathbf{V}_2$, $\mathbf{U}_2$, $\ldots$, $\mathbf{V}_{M-1}$, and $\mathbf{U}_{M-1}$ operating on $\mathbf{B}\boldsymbol{\Theta}_1$ in a "ping-pong" fashion so as to chase the unwanted nonzero element $z^{(1)}$ down the bidiagonal. This sequence of operations is illustrated, again for the case of $M = 4$, as follows:

$$\mathbf{U}_1^{\mathrm{T}}\mathbf{B}\boldsymbol{\Theta}_1 = \begin{bmatrix} \mathrm{x} & \mathrm{x} & z^{(2)} & 0 \\ 0 & \mathrm{x} & \mathrm{x} & 0 \\ 0 & 0 & \mathrm{x} & \mathrm{x} \\ 0 & 0 & 0 & \mathrm{x} \end{bmatrix};$$

$$\mathbf{U}_1^{\mathrm{T}}\mathbf{B}\boldsymbol{\Theta}_1\mathbf{V}_2 = \begin{bmatrix} \mathrm{x} & \mathrm{x} & 0 & 0 \\ 0 & \mathrm{x} & \mathrm{x} & 0 \\ 0 & z^{(3)} & \mathrm{x} & \mathrm{x} \\ 0 & 0 & 0 & \mathrm{x} \end{bmatrix};$$

$$\mathbf{U}_2^{\mathrm{T}}\mathbf{U}_1^{\mathrm{T}}\mathbf{B}\boldsymbol{\Theta}_1\mathbf{V}_2 = \begin{bmatrix} \mathrm{x} & \mathrm{x} & 0 & 0 \\ 0 & \mathrm{x} & \mathrm{x} & z^{(4)} \\ 0 & 0 & \mathrm{x} & \mathrm{x} \\ 0 & 0 & 0 & \mathrm{x} \end{bmatrix};$$

$$\mathbf{U}_2^{\mathrm{T}}\mathbf{U}_1^{\mathrm{T}}\mathbf{B}\boldsymbol{\Theta}_1\mathbf{V}_2\mathbf{V}_3 = \begin{bmatrix} \mathrm{x} & \mathrm{x} & 0 & 0 \\ 0 & \mathrm{x} & \mathrm{x} & 0 \\ 0 & 0 & \mathrm{x} & \mathrm{x} \\ 0 & 0 & z^{(5)} & \mathrm{x} \end{bmatrix};$$

$$\mathbf{U}_3^{\mathrm{T}}\mathbf{U}_2^{\mathrm{T}}\mathbf{U}_1^{\mathrm{T}}\mathbf{B}\boldsymbol{\Theta}_1\mathbf{V}_2\mathbf{V}_3 = \begin{bmatrix} \mathrm{x} & \mathrm{x} & 0 & 0 \\ 0 & \mathrm{x} & \mathrm{x} & 0 \\ 0 & 0 & \mathrm{x} & \mathrm{x} \\ 0 & 0 & 0 & \mathrm{x} \end{bmatrix}.$$

The adaptation cycle thus terminates with a new bidiagonal matrix **B** that is related to the original bidiagonal matrix **B** by

$$\mathbf{B} \leftarrow (\mathbf{U}_{M-1}^{\mathrm{T}} \ldots \mathbf{U}_2^{\mathrm{T}} \mathbf{U}_1^{\mathrm{T}}) \mathbf{B} (\boldsymbol{\Theta}_1 \mathbf{V}_2 \ldots \mathbf{V}_{M-1}) = \mathbf{U}^{\mathrm{T}} \mathbf{B} \mathbf{V}, \tag{G.84}$$

where

$$\mathbf{U} = \mathbf{U}_1 \mathbf{U}_2 \ldots \mathbf{U}_{M-1} \tag{G.85}$$

and

$$\mathbf{V} = \boldsymbol{\Theta}_1 \mathbf{V}_2 \ldots \mathbf{V}_{M-1}. \tag{G.86}$$

Steps 1 through 4 constitute one adaptation cycle of the Golub–Kahan algorithm. Typically, after a few adaptation cycles, the superdiagonal entry f_M becomes negligible. When f_M becomes sufficiently small, we can *deflate* the matrix and apply the algorithm to the smaller matrix. The criterion for the smallness of f_M is usually of the following form:

$$|f_M| \leq \varepsilon(|d_{M-1}| + |d_M|) \quad \text{where } \varepsilon \text{ is a small multiple of the machine precision.} \tag{G.87}$$

The description just presented leaves much unsaid about the Golub–Kahan algorithm for the diagonalization of a square data matrix. For a more detailed treatment of the algorithm, the reader is referred to the original paper of Golub and Kahan (1965) or the book by Golub and Van Loan (1996).

There is a significant improvement to be had in the Golub–Kahan algorithm. The algorithm has the property that it computes every singular value of a bidiagonal matrix **B** with an absolute error bound of about $\varepsilon \|\mathbf{B}\|$, where ε is the machine precision. Thus, large singular values (those near $\|\mathbf{B}\|$) are computed with high relative accuracy, but small ones (those near $\varepsilon \|\mathbf{B}\|$ or smaller) may be relatively inaccurate. The improved version of the algorithm computes every singular value to high relative accuracy, independently of its size. It also computes the singular vectors much more accurately. Approximately as fast as the old algorithm (and occasionally much faster), the new algorithm is a hybrid of the Golub–Kahan algorithm and a simplified version that corresponds to taking $\lambda = 0$ in Eq. (G.82). When $\lambda = 0$, the remainder of the algorithm can be stabilized so as to compute every matrix entry to high relative accuracy, whence the final accuracy of the singular values. [Analyses of this algorithm can be found in Demmel and Kahan (1990) and Deift et al. (1989).]

Summary of the QR Algorithm

The QR algorithm not only is mathematically elegant but also is a computationally powerful and highly versatile algorithm for SVD computation. Given a K-by-M data matrix **A**, the QR algorithm used to compute the SVD of **A** proceeds as follows:

1. Compute a sequence of Householder transformations that reduce the matrix **A** to upper bidiagonal form.
2. Apply the Golub–Kahan algorithm to the M-by-M nonzero submatrix resulting from step 1, and iterate this application until the superdiagonal elements become negligible in accordance with the criterion defined in Eq. (G.87).
3. Determine the SVD of the data matrix **A** as follows:
   - The diagonal elements of the matrix resulting from step 2 are the singular values of matrix **A**.

- The product of the Householder transformations of step 1 and the Givens rotations of step 2 involved in premultiplication defines the left singular vectors of **A**. The product of the Householder transformations and the Givens rotations involved in postmultiplication defines the right-singular vectors of **A**.

EXAMPLE 3

Consider the real-valued bidiagonal matrix

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix}.$$

Iteratively applying the Golub–Kahan algorithm to this matrix yields the sequence of results shown in Table G.1 for $\varepsilon = 10^{-4}$ in the stopping rule defined in Eq. (G.87). After two adaptation cycles of the algorithm, the (2, 3)-entry of the matrix **B** becomes small, at which point it is deflated. Accordingly, we now work on the 2-by-2 leading principal submatrix:

$$\begin{bmatrix} 0.8817 & 0.4323 \\ 0.0000 & 2.0791 \end{bmatrix}.$$

This submatrix is finally diagonalized in one step, yielding

$$\begin{bmatrix} 0.8596 & 0.0000 \\ 0.0000 & 2.1326 \end{bmatrix}.$$

The singular values of the bidiagonal matrix are thus computed to be

$$\sigma_1 = 0.8596,$$
$$\sigma_2 = 2.1326,$$

and

$$\sigma_3 = 3.2731.$$

The validity of this computation may be checked by comparing the trace of the matrix product $\mathbf{B}\mathbf{B}^{\mathrm{T}}$ with the sum of its eigenvalues, namely, $\sum_{i=1}^{3} \sigma_i^2$, which is left as an exercise for the reader.

TABLE G.1 First Two Adaptation Cycles of the Golub–Kahan Algorithm

Adaptation cycle number	Matrix B		
0	1.0000	1.0000	0.0000
	0.0000	2.0000	1.0000
	0.0000	0.0000	3.0000
1	0.9155	0.6627	0.0000
	0.0000	2.0024	0.0021
	0.0000	0.0000	3.2731
2	0.8817	0.4323	0.0000
	0.0000	2.0791	0.0000
	0.0000	0.0000	3.2731

APPENDIX H

Complex Wishart Distribution

The Wishart distribution plays an important role in statistical signal processing. In this appendix, we present a summary of some important properties of the distribution for complex-valued data. In particular, we derive a result that is pivotal to analyzing the convergence behavior of the traditional recursive least-squares (RLS) algorithm, presented in Problem 7 of Chapter 10. We begin the discussion with a definition of the complex Wishart distribution.

H.1 DEFINITION

Consider an M-by-M time-average (sample) correlation matrix

$$\mathbf{\Phi}(n) = \sum_{i=1}^{n} \mathbf{u}(i)\mathbf{u}^{\mathrm{H}}(i), \tag{H.1}$$

where the superscript H denotes *Hermitan transposition* (i.e., transposition combined with complex conjugation) and

$$\mathbf{u}(i) = [u_1(i), u_2(i), \ldots, u_M(i)]^{\mathrm{T}},$$

where the superscript T denotes *transposition*. In what follows, we assume that $\mathbf{u}(1)$, $\mathbf{u}(2), \ldots, \mathbf{u}(n)$ $(n > M)$ are *independently and identically distributed* (*i.i.d.*). We may then formally define the *complex Wishart distribution* as follows (Muirhead, 1982):

> If $\{u_1(i), u_2(i), \ldots, u_M(i) | i = 1, 2, \ldots, n\}, n \geq M$, is a sample from the M-dimensional Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{R})$, and if $\mathbf{\Phi}(n)$ is the time-average correlation matrix defined in Eq. (H.1), then the elements of $\mathbf{\Phi}(n)$ have the complex Wishart distribution $\mathcal{W}_M(n, \mathbf{R})$, which is characterized by the parameters M, n, and $\mathbf{R}$.

In specific terms, we may say that if matrix $\mathbf{\Phi}$ is $\mathcal{W}_M(n, \mathbf{R})$, then the probability density function of $\mathbf{\Phi}$ is

$$f(\mathbf{\Phi}) = \frac{1}{2^{Mn/2}\Gamma_M\left(\frac{1}{2}n\right)(\det(\mathbf{R}))^{n/2}} \mathrm{etr}\left(-\frac{1}{2}\mathbf{R}^{-1}\mathbf{\Phi}\right)(\det(\mathbf{\Phi}))^{(n-M-1)/2}, \tag{H.2}$$

where det denotes the determinant of the matrix in question, etr denotes the exponential raised to the trace of the pertinent matrix, and $\Gamma_M(a)$ is the *multivariate gamma function* defined by

$$\Gamma_M(a) = \int_{\mathbf{A}} \text{etr}(-\mathbf{A})(\det(\mathbf{A}))^{a-(M+1)/2} d\mathbf{A}, \tag{H.3}$$

where $\mathbf{A}$ is a positive definite matrix.

H.2 THE CHI-SQUARE DISTRIBUTION AS A SPECIAL CASE

For the special case of a univariate distribution (i.e., $M = 1$), Eq. (H.1) reduces to the scalar form

$$\varphi(n) = \sum_{i=1}^{n} |u(i)|^2. \tag{H.4}$$

Correspondingly, the correlation matrix $\mathbf{R}$ of $\mathbf{u}(n)$ reduces to the variance σ^2. Let

$$\chi^2(n) = \frac{\varphi(n)}{\sigma^2}. \tag{H.5}$$

Then, using Eq. (H.2), we may define the normalized probability density function of the normalized random variable $\chi^2(n)$ as

$$f(\chi^2) = \frac{\left(\frac{\chi^2}{2}\right)^{n/2-1} e^{-\chi^2/2}}{2^{n/2}\Gamma\left(\frac{1}{2}n\right)}, \tag{H.6}$$

where $\Gamma(1/2n)$ is the (scalar) *gamma function*.[1] The variable $\chi^2(n)$ is said to have a *chi-square distribution with n degrees of freedom*. We may thus view the complex Wishart distribution as a generalization of the univariate chi-square distribution.

[1]For the general case of a complex number g whose real part is positive, the *gamma function* $\Gamma(g)$ is defined by the definite integral (Wilks, 1962)

$$\Gamma(g) = \int_0^\infty x^{g-1} e^{-x} dx.$$

Integrating by parts, we readily find that

$$\Gamma(g) = (g-1)\Gamma(g-1).$$

For the case when g is a positive integer, we may express the gamma function as the factorial

$$\Gamma(g) = (g-1)!.$$

For the case of $g > 0$, but not an integer, we have

$$\Gamma(g) = (g-1)\Gamma(\delta),$$

where $0 < \delta < 1$.

For the particular case of $\delta = 1/2$, we have $\Gamma(\delta)\sqrt{\pi}$.

A useful property of a chi-square distribution with n degrees of freedom is that it is *reproductive with respect to* $1/2n$ (Wilks, 1962). That is, the rth moment of $\chi^2(n)$ is

$$\mathbb{E}[\chi^{2r}(n)] = \frac{2^r \Gamma\left(\frac{n}{2} + r\right)}{\Gamma\left(\frac{n}{2}\right)}. \tag{H.7}$$

Thus, the mean, the mean square, and the variance of $\chi^2(n)$ are, respectively, as follows:

$$\mathbb{E}[\chi^2(n)] = n; \tag{H.8}$$

$$\mathbb{E}[\chi^4(n)] = n(n+2); \tag{H.9}$$

$$\text{var}[\chi^2(n)] = n(n+2) - n^2 = 2n. \tag{H.10}$$

Moreover, putting $r = -1$ in Eq. (H.7), we find that the mean of the reciprocal of $\chi^2(n)$ is

$$\begin{aligned} \mathbb{E}\left[\frac{1}{\chi^2(n)}\right] &= \frac{1}{2}\frac{\Gamma\left(\frac{n}{2} - 1\right)}{\Gamma\left(\frac{n}{2}\right)} \\ &= \frac{1}{2}\frac{\Gamma\left(\frac{n}{2} - 1\right)}{\left(\frac{n}{2} - 1\right)\Gamma\left(\frac{n}{2} - 1\right)} = \frac{1}{n-2}. \end{aligned} \tag{H.11}$$

H.3 PROPERTIES OF THE COMPLEX WISHART DISTRIBUTION

The complex Wishart distribution has the following important properties (Muirhead, 1982; Anderson, 1984):

1. If $\mathbf{\Phi}$ is $\mathcal{W}_M(n, \mathbf{R})$ and $\mathbf{a}$ is any M-by-1 random vector distributed independently of $\mathbf{\Phi}$ with $\mathbb{P}(\mathbf{a} = \mathbf{0}) = 0$ (i.e., the probability that $\mathbf{a} = \mathbf{0}$ is zero), then $\mathbf{a}^{\mathrm{H}}\mathbf{\Phi a}/\mathbf{a}^{\mathrm{H}}\mathbf{R a}$ is chi-square distributed with n degrees of freedom and is independent of $\mathbf{a}$.
2. If $\mathbf{\Phi}$ is $\mathcal{W}_M(n, \mathbf{R})$ and $\mathbf{Q}$ is a matrix of dimensions M-by-k and rank k, then $\mathbf{Q}^{\mathrm{H}}\mathbf{\Phi Q}$ is $\mathcal{W}_k(n, \mathbf{Q}^{\mathrm{H}}\mathbf{R Q})$.
3. If $\mathbf{\Phi}$ is $\mathcal{W}_M(n, \mathbf{R})$ and $\mathbf{Q}$ is a matrix of dimensions M-by-k and rank k, then $(\mathbf{Q}^{\mathrm{H}}\mathbf{\Phi}^{-1}\mathbf{Q})^{-1}$ is $\mathcal{W}_k(n - M + k, (\mathbf{Q}^{\mathrm{H}}\mathbf{R}^{-1}\mathbf{Q})^{-1})$.
4. If $\mathbf{\Phi}$ is $\mathcal{W}_M(n, \mathbf{R})$ and $\mathbf{a}$ is any M-by-1 random vector distributed independently of $\mathbf{\Phi}$ with $\mathbb{P}(\mathbf{a} = \mathbf{0}) = 0$, then $\mathbf{a}^{\mathrm{H}}\mathbf{R}^{-1}\mathbf{a}/\mathbf{a}^{\mathrm{H}}\mathbf{\Phi}^{-1}\mathbf{a}$ is chi-square distributed with $n - M + 1$ degrees of freedom.
5. Let $\mathbf{\Phi}$ and $\mathbf{R}$ be partitioned into p and $M - p$ rows and columns, as illustrated by

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{\Phi}_{11} & \mathbf{\Phi}_{12} \\ \mathbf{\Phi}_{21} & \mathbf{\Phi}_{22} \end{bmatrix}$$

and

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{bmatrix}.$$

If $\mathbf{\Phi}$ is distributed according to $\mathcal{W}_M(n, \mathbf{R})$, then $\mathbf{\Phi}_{11}$ is distributed according to $\mathcal{W}_p(n, \mathbf{R}_{11})$.

H.4 EXPECTATION OF THE INVERSE CORRELATION MATRIX $\mathbf{\Phi}^{-1}(n)$

Property 4 of the complex Wishart distribution may be used to find the expectation of the inverse correlation matrix $\mathbf{\Phi}^{-1}(n)$, which, in certain situations, is associated with the convergence of the RLS algorithm in the mean square. Specifically, for any fixed and nonzero $\boldsymbol{\alpha}$ in $\mathbb{R}^M$, we know from Property 4 that $\boldsymbol{\alpha}^{\mathrm{H}}\mathbf{R}^{-1}\boldsymbol{\alpha}/\boldsymbol{\alpha}^{\mathrm{H}}\mathbf{\Phi}^{-1}\boldsymbol{\alpha}$ is chi-square distributed with $n - M + 1$ degrees of freedom. Let $\chi^2(n - M + 1)$ denote this ratio. Then, using the result described in Eq. (H.11), we may write

$$\begin{aligned} \mathbb{E}[\boldsymbol{\alpha}^{\mathrm{H}}\mathbf{\Phi}^{-1}(n)\boldsymbol{\alpha}] &= \boldsymbol{\alpha}^{\mathrm{H}}\mathbf{R}^{-1}\boldsymbol{\alpha}\mathbb{E}\left[\frac{1}{\chi^2(n - M + 1)}\right] \\ &= \frac{1}{n - M - 1}\boldsymbol{\alpha}^{\mathrm{H}}\mathbf{R}^{-1}\boldsymbol{\alpha}, \qquad n > M + 1, \end{aligned}$$

which, in turn, implies that

$$\mathbb{E}[\mathbf{\Phi}^{-1}(n)] = \frac{1}{n - M - 1}\mathbf{R}^{-1}, \qquad n > M + 1. \tag{H.12}$$

Glossary

TEXT CONVENTIONS

1. Boldfaced lowercase letters are used to denote column vectors. Boldfaced uppercase letters are used to denote matrices.
2. The estimate of a scalar, vector, or matrix is designated by the use of a hat ($\hat{}$) placed over the pertinent symbol.
3. The symbol $|\ |$ denotes the magnitude or absolute value of a complex scalar enclosed within. The symbol ang[ ] or arg[ ] denotes the phase angle of the scalar enclosed within.
4. The symbol $\|\ \|$ denotes the Euclidean norm of the vector or matrix enclosed within.
5. The symbol det( ) denotes the determinant of the square matrix enclosed within.
6. The open interval (a, b) of the variable x signifies that $a < x < b$. The closed interval $[a, b]$ signifies that $a \leq x \leq b$, and $(a, b]$ signifies that $a < x \leq b$.
7. The inverse of nonsingular (square) matrix $\mathbf{A}$ is denoted by $\mathbf{A}^{-1}$.
8. The pseudoinverse of matrix $\mathbf{A}$ (not necessarily square) is denoted by $\mathbf{A}^{+}$.
9. Complex conjugation of a scalar, vector, or matrix is denoted by the use of a superscript asterisk. Transposition of a vector or matrix is denoted by superscript T. Hermitian transposition (i.e., complex conjugation and transposition combined) of a vector or matrix is denoted by superscript H. Backward rearrangement of the elements of a vector is denoted by superscript B.
10. The symbol $\mathbf{A}^{-\mathrm{H}}$ denotes the Hermitian transpose of the inverse of a nonsingular (square) matrix $\mathbf{A}$.
11. The square root of a square matrix $\mathbf{A}$ is denoted by $\mathbf{A}^{1/2}$.
12. The symbol $\mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_M)$ denotes a diagonal matrix whose elements on the main diagonal are $\lambda_1, \lambda_2, \ldots, \lambda_M$.
13. The order of a linear predictor or the order of an autoregressive model is signified by a subscript added to the pertinent scalar or vector parameter.
14. The statistical expectation operator is denoted by $\mathbb{E}[\cdot]$, where the quantity enclosed is the random variable or random vector of interest. The variance of a random variable is denoted by $\mathrm{var}[\cdot]$, where the quantity enclosed is the random variable.

15. The conditional probability density function of random variable U, given that hypothesis H_i is true, is denoted by $f_U(u \mid \mathrm{H}_i)$, where u is a sample value of the random variable U.

16. The inner product of two vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as $\mathbf{x}^{\mathrm{H}}\mathbf{y} = \mathbf{y}^{\mathrm{T}}\mathbf{x}^*$. Another possible inner product is $\mathbf{y}^{\mathrm{H}}\mathbf{x} = \mathbf{x}^{\mathrm{T}}\mathbf{y}^*$. These two inner products are the complex conjugate of each other. The outer product of the vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as $\mathbf{x}\mathbf{y}^{\mathrm{H}}$. The inner product is a scalar, whereas the outer product is a matrix.

17. The trace of a square matrix $\mathbf{R}$ is denoted by $\mathrm{tr}[\mathbf{R}]$ and is defined as the sum of the diagonal elements of $\mathbf{R}$. The exponential raised to the trace of matrix $\mathbf{R}$ is denoted by $\mathrm{etr}[\mathbf{R}]$.

18. The autocorrelation function of stationary discrete-time stochastic process $u(n)$ is defined by

$$r(k) = \mathbb{E}[u(n)u^*(n-k)].$$

The cross-correlation function between two jointly stationary discrete-time stochastic processes $u(n)$ and $d(n)$ is defined by

$$p(-k) = \mathbb{E}[u(n-k)d^*(n)].$$

19. The ensemble-average correlation matrix of a random vector $\mathbf{u}(n)$ is defined by

$$\mathbf{R} = \mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)].$$

The use of subscripts assigned to the correlation matrix $\mathbf{R}$ is avoided in the book, except in Chapter 13 on adaptation in nonstationary environments, where we have

$$\mathbf{R}_u = \mathbb{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{H}}(n)]$$

and

$$\mathbf{R}_\omega = \mathbb{E}[\boldsymbol{\omega}(n)\boldsymbol{\omega}^{\mathrm{H}}(n)].$$

20. The ensemble-average cross-correlation vector between a random vector $\mathbf{u}(n)$ and a random variable $d(n)$ is defined by

$$\mathbf{p} = \mathbb{E}[\mathbf{u}(n)d^*(n)].$$

21. The time-average (sample) correlation matrix of a vector $\mathbf{u}(i)$ over the observation interval $1 \le i \le n$ is defined by

$$\boldsymbol{\Phi}(n) = \sum_{i=1}^{n} \mathbf{u}(i)\mathbf{u}^{\mathrm{H}}(i).$$

The exponentially weighted version of $\boldsymbol{\Phi}(n)$ is

$$\boldsymbol{\Phi}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)\mathbf{u}^{\mathrm{H}}(i),$$

where λ is the exponential weighting factor that lies in the interval $0 < \lambda \le 1$.

22. The time-average cross-correlation vector between a vector $\mathbf{u}(i)$ and scalar $d(i)$ over the observation interval $1 \le i \le n$ is defined by

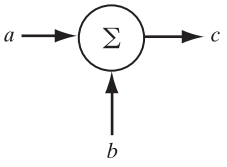
$$\mathbf{z}(n) = \sum_{i=1}^{n} \mathbf{u}(i)d^*(i).$$

Its exponentially weighted version is

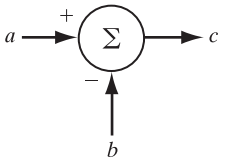
$$\mathbf{z}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)d^*(i).$$

23. The discrete-time Fourier transform of a time function $u(n)$ is denoted by $\mathrm{F}[u(n)]$. The inverse discrete-time Fourier transform of a frequency function $U(\omega)$ is denoted by $\mathrm{F}^{-1}[U(\omega)]$.

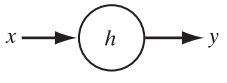
24. In constructing block diagrams (signal-flow graphs) involving matrix quantities, the following symbols are used: The symbol



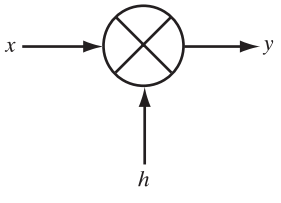
depicts an adder with $c = a + b$. The same symbol with algebraic signs added, viz.,



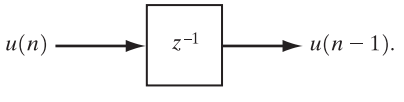
depicts a subtractor with $c = a - b$. The symbol



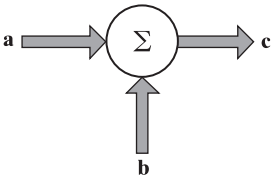
denotes a multiplier with $y = hx$. This multiplication is also represented as



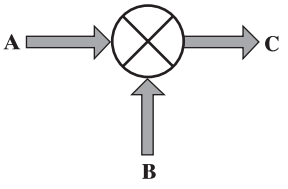
The unit-sample (delay) operator is depicted by



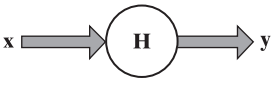
25. In constructing block diagrams (signal-flow graphs) involving matrix quantities, the following symbols are used: The symbol



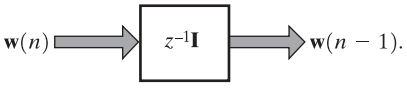
depicts the summation $\mathbf{c} = \mathbf{a} + \mathbf{b}$. The symbol



depicts multiplication, with product $\mathbf{C} = \mathbf{AB}$. The symbol



depicts a branch having transmittance $\mathbf{H}$, with $\mathbf{y} = \mathbf{Hx}$. The unit-sample operator is denoted by the symbol



ABBREVIATIONS

ADE	Algorithm-dependent equilibria
AGC	Automatic gain control
AIC	An information-theoretic criterion (due to Akaike)
ALE	Adaptive line enhancer
AR	Autoregressive (process)
ARMA	Autoregressive moving average (process)
BEFAP	Block exact fast affine projection
BIBO	Bounded input, bounded output
BLP	Backward linear prediction
BLUE	Best linear unbiased estimate
CDE	Channel-dependent equilibria

CMA	Constant-modulus algorithm
CRLB	Cramér–Rao lower bound
dB	Decibel
DCT	Discrete cosine transform
DFT	Discrete Fourier transform
DPCM	Differential pulse-code modulation
DSE-CMA	Dithered signed-error constant modulus algorithm
DTE	Data terminal equipment
EC	Echo canceller
FAP	Fast affine projection
FBLP	Forward and backward linear prediction
FDAF	Frequency-domain adaptive filter
FFT	Fast Fourier transform
FIR	Finite-duration impulse response
FLM	Fourth-least-mean
FLP	Forward linear prediction
FM	Frequency modulated (signal)
FSE	Fractionally spaced equalizer
FTF	Fast transversal (finite-duration impulse response) filtering algorithm
GAL	Gradient-adaptive lattice
GSC	Generalized sidelobe canceller
HOS	Higher-order statistics
Hz	Hertz
IDBD	Incremental delta-bar-delta (algorithm)
IF	Intermediate frequency
IFFT	Inverse fast Fourier transform
i.i.d.	Independent and identically distributed
IIR	Infinite-duration impulse response
INR	Interference-to-noise ratio
ISI	Intersymbol interference
KaGE	Kalman gain estimator
kb/s	Kilobits per second
KLMS	Kernel least mean square (algorithm)
KLT	Karhunen–Loève transform
LCMV	Linearly constrained minimum variance (algorithm)
LEM	Loudspeaker–enclosure–microphone
LMS	Least-mean-square (algorithm)
LPC	Linear predictive coding
LSB	Least significant bit

LSL	Least-squares lattice
LTI	Linear time invariant
M-ary PAM	Multilevel phase-amplitude modulation
MA	Moving average
MAIC	Minimum Akaike information-theoretic criterion
MAP	Maximum a posteriori probability
MaxEnt	Maximum entropy
MDL	Minimum description length (criterion)
MEM	Maximum-entropy method
MISO	Multiple input, single output
MLM	Maximum-likelihood method
MSD	Mean-square deviation
MSE	Mean-square error
MUSIC	Multiple signal classification (algorithm)
MVDR	Minimum-variance distortionless response
MVUE	Minimum-variance unbiased estimate
PARCOR	Partial correlation
PCM	Pulse-code modulation
pdf	probability density function
PEF	Prediction error filter
PN	Pseudonoise
PSK	Phase-shift keying
QAM	Quadrature amplitude modulation
QPSK	Quadrature phase-shift keying
QRD	QR-decomposition
QRD-LSL	QR-decomposition-based least-squares lattice (algorithm)
QRD-RLS	QR-decomposition-based recursive least squares (algorithm)
RBF	Radial basis function
RKHS	Reproduction Kernel Hilbert space
RLS	Recursive least squares (algorithm)
rms	Root mean square
RMSE	Root mean-square-error
s	Second
SIMO	Single input, multiple output
SISO	Single input, single output
SNR	Signal-to-noise ratio
SOS	Second-order statistics
SVD	Singular-value decomposition
VLSI	Very large-scale integration

PRINCIPAL SYMBOLS

$a_{M,k}(n)$	kth tap weight of forward prediction-error filter of order M (at adaptation cycle n), with $k = 0, 1, \ldots, M$; note that $a_{m,0}(n) = 1$ for all n
$\mathbf{a}_M(n)$	Tap-weight vector of forward prediction-error filter of order M (at adaptation cycle n)
$\mathbf{A}$	Data matrix in the covariance method
$\mathbf{A}(n)$	Data matrix in the prewindowing method
$b_m(n)$	Backward (a posteriori) prediction error produced at adaptation cycle n by prediction-error filter of order $m = 0, 1, \ldots$
$\mathbf{b}(n)$	Backward (a posteriori) prediction-error vector representing sequence of errors produced by backward prediction-error filters of orders $0, 1, \ldots, M$
$\mathcal{B}_M(n)$	Sum of weighted backward prediction-error squares produced by backward prediction-error filter of order M
c	Cosine parameter in Givens rotation
$c_{M,k}(n)$	kth tap weight of backward prediction-error filter of order M (at adaptation cycle n), with $k = 0, 1, \ldots, M$; note that $c_{M,M}(n) = 1$ for all n
$\mathbf{c}_M(n)$	Tap-weight vector of backward prediction-error filter of order M (at adaptation cycle n)
$\mathbf{c}(n)$	Weight-error vector in steepest-descent algorithm
$c_k(\tau_1, \tau_2, \ldots, \tau_k)$	kth-order cumulant
$C_k(\omega_1, \omega_2, \ldots, \omega_k)$	kth-order polyspectrum
$\mathcal{C}$	Contour in complex variable theory
$\mathbb{C}^M$	Complex vector space of dimension M
$\mathcal{C}(n)$	Convergence ratio
d	Differential operator
det( )	Determinant of the enclosed matrix
diag( )	Diagonal matrix
$d(n)$	Desired response
$\mathbf{d}$	Desired response vector in the covariance method
$\mathbf{d}(n)$	Desired response vector in the prewindowing method
D	Unit-delay operator (same as z^{-1})
$\mathbf{D}_{m+1}(n)$	Correlation matrix of backward prediction errors
$\mathcal{D}$	Mean-square deviation
dec( )	Function describing the decision performed by a threshold device
$e(n)$	A posteriori estimation error, or error signal

$e_m(n)$	A posteriori estimation error at the output of stage m in the joint-process estimator, using the recursive LSL algorithm or QRD-LSL algorithm
e	Base of natural logarithm
etr( )	Exponential raised to the trace of the enclosed matrix
exp	Exponential
$\mathbb{E}$	Expectation operator
$\mathbf{E}$	Perturbation matrix
$\mathscr{E}(\mathbf{w}, n)$	Cost function defined as the sum of weighted error squares, expressed as a function of adaptation cycle n and weight vector $\mathbf{w}$
$\mathscr{E}(\mathbf{w})$	Cost function defined as the sum of error squares, expressed as a function of the tap-weight vector $\mathbf{w}$
$\mathscr{E}_{\min}$	Minimum value of $\mathscr{E}(\mathbf{w})$
$\mathscr{E}(n)$	Cost function defined as the sum of weighted error squares, expressed as a function of adaptation cycle n
$f_M(n)$	Forward (a posteriori) prediction error produced at adaptation cycle n by forward prediction-error filter of order M
$\mathbf{f}(n)$	Forward (a posteriori) prediction error vector representing sequence of errors produced by forward prediction-error filters of orders $0, 1, \ldots, M$
$f_U(u)$	Probability density function of random variable U whose sample value equals u
$f_{\mathbf{U}}(\mathbf{u})$	Joint probability density function of the elements of random vector $\mathbf{U}$ whose sample value equals $\mathbf{u}$
$F_M(z)$	z-transform of sequence of forward prediction errors produced by forward prediction-error filter of order M
$\mathbf{F}(n+1, n)$	Transition matrix (of the process model)
$\mathscr{F}_M(n)$	Weighted sum of forward prediction-error squares produced by forward prediction-error filter of order M
$F[\,]$	Fourier transform operator
$F^{-1}[\,]$	Inverse Fourier transform operator
$g(\cdot)$	Nonlinear function used in blind equalization
$\mathbf{G}(n)$	Kalman gain
h_k	kth regression coefficient of joint-process estimation based on lattice predictor
H_i	ith hypothesis
$H(z)$	Transfer function of discrete-time linear filter
I	Subscript for signifying the in-phase (real) component of a complex baseband signal
$\mathbf{I}$	Identity matrix

$\mathbf{I}$	Inverse of Fisher's information matrix $\mathbf{J}$
j	Square root of -1
$J(\mathbf{w})$	Cost function used to formulate the Wiener filtering problem, expressed as a function of the tap-weight vector $\mathbf{w}$
$\mathbf{J}$	Fisher's information matrix
$\mathbf{k}(n)$	Gain vector in the RLS algorithm
$\mathbf{K}(n)$	Correlation matrix of weight-error vector $\boldsymbol{\varepsilon}(n)$
ln	Natural logarithm
$l(\boldsymbol{\theta})$	Log-likelihood function of parameter vector $\boldsymbol{\theta}$
$\mathbf{L}(n)$	Transformation matrix in the form of lower triangular matrix
$\mathscr{L}$	Linear time-invariant system
$\mathbb{L}^p$	Normed linear space
m	Variable order of linear predictor or autoregressive model
M	Final order of linear predictor or autoregressive model
M, K	Final order of autoregressive moving-average model
$\mathscr{M}$	Misadjustment (in the LMS algorithm)
n	Discrete-time, or number of adaptation cycles, applied to recursive algorithm
N	Data length
$\mathcal{N}$	Symbol signifying the Gaussian (normal) distribution
$\mathcal{N}$	Noise subspace
$O\ (M^k)$	Order of M^k
$p(-k)$	Element of cross-correlation vector $\mathbf{p}$ for lag k
$\mathbf{p}$	Cross-correlation vector between tap-input vector $\mathbf{u}(n)$ and desired response $d(n)$
P_M	Average value of (forward or backward) prediction-error power for prediction order M for stationary inputs
$\mathbf{P}(n)$	Matrix equal to the inverse of the time-average correlation matrix $\boldsymbol{\Phi}(n)$ used in formulating the RLS algorithm
q_{ki}	ith element of kth eigenvector
$\mathbf{q}_k$	kth eigenvector
Q	Subscript for signifying the quadrature (imaginary) component of a complex baseband signal
$\mathbf{Q}$	Unitary matrix that consists of normalized eigenvectors in the set $\{\mathbf{q}_k\}$ used as columns
$Q(\mathrm{y})$	Probability distribution function of standardized Gaussian random variable of zero mean and unit variance
$r(k)$	Element of (ensemble-average) correlation matrix $\mathbf{R}$ for lag k
$r^{-1}(n)$	Conversion factor (of zero mean and unit variance

$\mathbf{R}$	Ensemble-average correlation matrix of stationary discrete-time process $u(n)$
$\mathbb{R}^M$	Real vector space of dimension M
$\mathbf{s}$	Signal vector; steering vector
sgn( )	Signum function
$S(\omega)$	Power spectral density
$S_{\text{AR}}(\omega)$	Autoregressive (power) spectrum
$S_{\text{MEM}}(\omega)$	MEM (maximum-entropy-method) spectrum
$S_{\text{MVDR}}(\omega)$	Minimum-variance distortionless response spectrum
$\mathcal{S}$	System
$\mathcal{S}_d$	Decreasingly excited subspace
$\mathcal{S}_o$	Otherwise excited subspace
$\mathcal{S}_p$	Persistently excited subspace
$\mathcal{S}_u$	Unexcited subspace
t	Time
$\mathbf{t}$	Vector arising in joint-process estimation for nonstationary inputs
$\mathcal{T}$	Distance measure in the subspace decomposition procedure for blind deconvolution
$u(n)$	Sample value of tap input in FIR filter at time n
$\mathbf{u}(n)$	Tap-input vector consisting of $u(n), u(n-1), \ldots,$ as elements
$u_1(n)$	In-phase component of complex $u(n)$
$u_Q(n)$	Quadrature component of complex $u(n)$
$\mathbf{u}_k$	kth left-singular vector of data matrix $\mathbf{A}$
$\mathbf{U}$	Matrix of left-singular vectors of data matrix $\mathbf{A}$
$\mathcal{U}_n$	Space spanned by tap inputs $u(n), u(n-1), \ldots$
$\mathcal{U}(n)$	Sum of weighted squared values of tap inputs $u(i), i = 1, 2, \ldots, n$
$\mathbf{v}(n)$	Transformed weight-error vector in steepest-descent and LMS algorithms
$\mathbf{v}_k(n)$	kth right-singular vector of data matrix $\mathbf{A}$
$\mathbf{V}$	Matrix of right singular vectors of data matrix $\mathbf{A}$
$w_k(n)$	kth tap weight of FIR filter at time n
$w_{b,m,k}(n)$	kth tap weight of backward predictor of order m at adaptation cycle n
$w_{f,m,k}(n)$	kth tap weight of forward predictor of order m at adaptation cycle n
$\mathbf{w}(n)$	Tap-weight vector of FIR filter at time n
$\mathbf{w}_{b,m}(n)$	Tap-weight vector of backward predictor of order m at adaptation cycle n
$\mathbf{w}_{f,m}(n)$	Tap-weight vector of forward predictor of order m at adaptation cycle n

$\mathcal{W}$	Symbol signifying the Wishart distribution
$\mathbf{x}(n)$	State used in formulating Kalman filter theory
$\mathbf{y}(n)$	Observation used in formulating Kalman filter theory
$\mathcal{Y}_n$	Vector space spanned by observations $\mathbf{y}(n), \mathbf{y}(n-1), \ldots$
$y'(n)$	Modified output signal in the equation error method of designing IIR adaptive filters
z^{-1}	Unit-sample (delay) operator used in defining the z-transform of a sequence
$\mathbf{z}$	Time-average cross-correlation vector between tap-input vector $\mathbf{u}(i)$ and desired response $d(i)$
$Z(y)$	Standardized Gaussian probability density function
$\alpha(n)$	Innovations vector at time n
β	Constant used in the DCT-LMS algorithm
β	Constant used in the GAL algorithm
$\beta_m(n)$	Backward prediction error of order m at time n
$\gamma(n)$	Conversion factor used in the recursive LSL algorithm, and recursive QRD-LSL algorithm
$\Gamma(g)$	Gamma function of g
γ_1	Skewness of a random variable
γ_2	Kurtosis of a random variable
δ	Regularization parameter
$\boldsymbol{\delta}$	First coordinator vector (also denoted by $\mathbf{1}$)
δ_l	Kronecker delta, equal to unity for $l = 0$ and zero for $l \neq 0$
$\Delta_m(n)$	Cross-correlation between forward prediction error $f_m(n)$ and delayed backward prediction error $b_m(n-1)$
$\varepsilon_m(n)$	Angle-normalized joint-process estimation error for prediction order m
$\varepsilon_{b,m}(n)$	Angle-normalized backward prediction error for prediction order m
$\varepsilon_{f,m}(n)$	Angle-normalized forward prediction error for prediction order m
$\boldsymbol{\varepsilon}(n)$	Weight-error vector
$\eta(n)$	Forward (a priori) prediction error
$\boldsymbol{\theta}$	Parameter vector
$\boldsymbol{\Theta}$	Unitary rotation
κ_m	mth reflection of a lattice predictor for stationary environment
$\kappa_{b,m}(n)$	mth backward-reflection coefficient of a least-squares lattice for a nonstationary environment
$\kappa_{f,m}(n)$	mth forward-reflection coefficient of a least squares lattice for a nonstationary environment

$\kappa_4(\tau_1, \tau_2, \tau_3)$	Tricepstrum
λ	Exponential weighting vector in RLS, LSL, QR-RLS, and QRD-LSL algorithms
λ_k	kth eigenvalue of correlation matrix $\mathbf{R}$
$\lambda_{\max}$	Maximum eigenvalue of correlation matrix $\mathbf{R}$
$\lambda_{\min}$	Minimum eigenvalue of correlation matrix $\mathbf{R}$
$\mathbf{\Lambda}(n)$	Diagonal matrix of exponential weighting factors
μ	Mean value
μ	Step-size parameter in steepest-descent algorithm or LMS algorithm
$\nu(n)$	Sample value of white-noise process of zero mean
$\nu(n)$	Convolutional noise in Bussgang algorithm
$\boldsymbol{\nu}_1(n)$	Process noise vector in process equation
$\boldsymbol{\nu}_2(n)$	Measurement noise vector in measurement equation
$\boldsymbol{\nu}(n)$	Process noise vector in random-walk state model
$\boldsymbol{\pi}$	Vector in RLS algorithm
$\pi_m(n)$	mth parameter in QRD-LSL algorithm
$\xi(n)$	A priori estimation error
$\xi_u(n)$	Undisturbed estimation error
$\phi(t, k)$	t, kth element of time-average correlation matrix $\mathbf{\Phi}$
$\boldsymbol{\varphi}(n, n_0)$	Transition matrix arising in finite-precision analysis of RLS algorithms
$\mathbf{\Phi}$	Time-average correlation matrix
$\mathbf{\Phi}(n)$	Time-average correlation matrix expressed as a function of the observation interval n
$\chi^2(n)$	Chi-square distributed random variable with n degrees of freedom
$\chi(\mathbf{R})$	Eigenvalue spread (i.e., ratio of maximum eigenvalue to minimum eigenvalue) of correlation matrix $\mathbf{R}$
ω	Normalized angular frequency; $0 < \omega \leq 2\pi$, or $-\pi < \omega \leq \pi$
$\boldsymbol{\omega}(n)$	Process noise vector in Markov model
ρ_m	Correlation coefficient or normalized value of autocorrelation function for lag m
σ^2	Variance
τ_k	Time constant of kth natural mode of steepest-descent algorithm
$\tau_{\text{mse, av}}$	Time constant of a single decaying exponential that approximates the learning curve of LMS algorithm
$\nabla(n)$	Residual impulse response of a channel in blind equalization
$\boldsymbol{\nabla}$	Gradient vector

Bibliography

Abramovich, Y. I. (2000). "Convergence analysis of linearly constrained SMI and LSMI adaptive algorithms," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium,* Lake Louise, AB, Canada, pp. 255–259.

Adali, T., and H. Li (2010). "Complex-valued adaptive signal processing," in T. Adali and S. Haykin, eds., *Adaptive Signal Processing: Next-Generation Solutions,* Wiley Interscience, Hoboken, NJ, pp. 1–85.

Akaike, H. (1973). "Maximum likelihood identification of Gaussian autoregressive moving average models," *Biometrika,* vol. 60, pp. 255–265.

Akaike, H. (1974). "A new look at the statistical model identification," *IEEE Trans. Autom. Control,* vol. AC-19, pp. 716–723.

Akaike, H. (1977). "An entropy maximisation principle," in P. Krishnaiah, ed., *Proceedings of Symposium on Applied Statistics,* North-Holland, Amsterdam.

Albert, A. E., and L. S. Gardner, Jr. (1967). *Stochastic Approximation and Nonlinear Regression,* MIT Press, Cambridge, MA.

Alexander, S. T. (1986). *Adaptive Signal Processing: Theory and Applications,* Springer-Verlag, New York.

Alexander, S. T. (1987). "Transient weight misadjustment properties for the finite precision LMS algorithm," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-35, pp. 1250–1258.

Alexander, S. T., and A. L. Ghirnikar (1993). "A method for recursive least-squares filtering based upon an inverse QR decomposition," *IEEE Trans. Signal Process*., vol. 41, pp. 20–30.

Anderson, T. W. (1984). *An Introduction to Multivariate Statistical Analysis,* 2nd ed., Wiley, New York.

Applebaum, S. P. (1966). *Adaptive Arrays,* Rep. SPL TR 66-1, Syracuse University Research Corporation, Syracuse, NY.

Ardalan, S. H., and S. T. Alexander (1987). "Fixed-point roundoff error analysis of the exponentially windowed RLS algorithm for time varying systems," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-35, pp. 770–783.

Aronszajn, N. (1950). "Theory of reproducing kernels," *Trans. Am. Math. Soc.,* vol. 68, pp. 337–404.

Atal, B. S., and S. L. Hanauer (1971). "Speech analysis and synthesis by linear prediction of the speech wave," *J. Acoust. Soc. Am*., vol. 50, pp. 637–655.

Atal, B. S., and M. R. Schroeder (1970). "Adaptive predictive coding of speech signals," *Bell Syst. Tech. J*., vol. 49, pp. 1973–1986.

Austin, M. E. (1967). *Decision-Feedback Equalization for Digital Communication over Dispersive Channels,* Tech. Rep. 437, MIT Lincoln Laboratory, Lexington, MA.

Barrett, J. F., and D. G. Lampard (1955). "An expansion for some second-order probability distributions and its application to noise problems," *IRE Trans. Information Theory,* vol. IT-1, pp. 10–15.

Basar, T., and P. Bernhard (1991). *H^∞-Optimal Control and Related Minimax Design Problems: A Dynamic Game Approach,* Birkhauser, Boston.

Beaufays, F. (1995). "Transform-domain adaptive filters: An analytical approach," *IEEE Trans. Signal Process*., vol. 43, pp. 422–431.

Beaufays, F., and B. Widrow (1994). "Two-layer linear structures for fast adaptive filtering," in *World Congress on Neural Networks,* vol. III, San Diego, pp. 87–93.

Bellanger, M. G. (1988a). *Adaptive Filters and Signal Analysis,* Dekker, New York.

Bellanger, M. G. (1988b). "The FLS-QR algorithm for adaptive filtering," *Signal Process*., vol. 17, pp. 291–304.

Bellanger, M., G. Bonnerot, and M. Coudreuse (1976). "Digital filtering by polyphase network: Application to sample rate alteration and filter banks," *IEEE Trans. Acoust. Speech Signal Process.,* vol. ASSP-24, pp. 109–114.

Bellini, S. (1986). "Bussgang techniques for blind equalization," in GLOBECOM, Houston, TX, pp. 1634–1640.

Bellini, S. (1988). "Blind equalization," *Alta Freq*., vol. 57, pp. 445–450.

Bellini, S. (1994). "Bussgang techniques for blind deconvolution and equalization," in S. Haykin, ed., *Blind Deconvolution,* Prentice-Hall, Englewood Cliffs, NJ.

Bellman, R. (1961). *Adaptive Control Processes: A Guided Tour,* Princeton University Press, Princeton, NJ.

Bender, C. M., and S. A. Orszag (1999). *Advanced Mathematical Methods for Scientists and Engineers: Asymptotic Methods and Perturbation Theory,* Springer-Verlag, New York.

Benesty, J., T. Gänsler, D. R. Morgan, M. M. Sondhi, and S. L. Gay (2001). *Advances in Network and Acoustic Echo Cancellation,* Springer-Verlag, Berlin.

Benveniste, A., M. Goursat, and G. Ruget (1980). "Robust identification of a nonminimum phase system: Blind adjustment of a linear equalizer in data communications," *IEEE Trans. Autom. Control,* vol. AC-25, pp. 385–399.

Benveniste, A., M. Mètivier, and P. Priouret (1987). *Adaptive Algorithms and Stochastic Approximations,* Springer-Verlag, New York.

Bierman, G. J., and C. L. Thornton (1977). "Numerical comparison of Kalman filter algorithms: Orbit determination case study," *Automatica,* vol. 13, pp. 23–35.

Bitmead, R. R., and B. D. O. Anderson (1980a). "Lyapunov techniques for the exponential stability of linear difference equations with random coefficients," *IEEE Trans. Autom. Control,* vol. AC-25, pp. 782–787.

Bitmead, R. R., and B. D. O. Anderson (1980b). "Performance of adaptive estimation algorithms in dependent random environments," *IEEE Trans. Autom. Control,* vol. AC-25, pp. 788–794.

Boser, B. E., I. M. Guyon, and V. Vapnik (1992). "A training algorithm for optimal margin classifiers," in D. Haussler, ed., *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory,* July, Pittsburgh, PA, pp. 144–152.

Bottomley, G. E., and S. T. Alexander (1989). "A theoretical basis for the divergence of conventional recursive least squares filters," in *Proc. ICASSP,* Glasgow, pp. 908–911.

Bouboulis, P., and S. Theodoridis (2011). "Extension of Wirtinger's calculus to reproducing kernel Hilbert spaces and the complex kernel LMS," *IEEE Trans. Signal Process*., vol. 53, pp. 964–978.

Bouboulis, P., S. Theodoridis, and M. E. Mavroforakis (2012). "The Augmented Complex Kernel LMS," *IEEE Trans. Signal Process*., vol. 60, pp. 4962–4967.

Box, G. E. P., and G. M. Jenkins (1976). *Time Series Analysis: Forecasting and Control,* Holden-Day, San Francisco.

Brady, D. M. (1970). "An adaptive coherent diversity receiver for data transmission through dispersive media," in *Conf. Rec. ICC 70,* pp. 21-35–21-40.

Breining, C., et al. (1999). "Acoustic echo control," *IEEE Signal Process. Maga.,* vol. 16, no. 4, pp. 42–69.

Brillinger, D. R. (1975). *Time Series: Data Analysis and Theory,* Holt, Rinehart, and Winston, New York.

Brooks, L. W., and I. S. Reed (1972). "Equivalence of the likelihood ratio processor, the maximum signal-to-noise ratio filter, and the Wiener filter," *IEEE Trans. Aerospace Electron. Syst.,* vol. AES-8, pp. 690–692.

Burg, J. P. (1967). "Maximum entropy spectral analysis," in *37th Ann. Int. Meet., Soc. Explor. Geophys*., Oklahoma City, OK.

Burg, J. P. (1968). "A new analysis technique for time series data," *NATO Advanced Study Institute on Signal Processing,* Enschede, The Netherlands.

Burg, J. P. (1975). *Maximum entropy spectral analysis,* Ph.D. dissertation, Stanford University, Stanford, CA.

Bussgang, J. J. (1952). *Cross Correlation Functions of Amplitude-Distorted Gaussian Signals,* Tech. Rep. 216, MIT Research Laboratory of Electronics, Cambridge, MA.

Butterweck, H. J. (1995). "A steady-state analysis of the LMS adaptive algorithm without use of the independence assumption," in *Proc. ICASSP,* Detroit, pp. 1404–1407.

Butterweck, H. J. (2001). "A wave theory of long adaptive filters," *IEEE Trans. Circuits Syst. Fundam. Theory Appl.,* vol. 48, pp. 739–747.

Butterweck, H. J. (2003). "Traveling-wave model of long LMS filters," in S. Haykin and B. Widrow, eds., *Least-Mean-Square Adaptive Filters,* Wiley, New York, pp. 35–78.

Butterweck, H. J. (2011). "Steady-state analysis of long LMS filters," *Signal Process*., vol. 91, pp. 690–701.

Campisi, P., and K. Egiazarian, eds. (2007). *Blind Image Deconvolution,* CRC Press, Boca Raton, FL.

Capman, F., J. Boudy, and P. Lockwood (1995). "Acoustic echo cancellation using a fast QR-RLS algorithm and multirate schemes," in *Proc. ICASSP,* Detroit, pp. 969–971.

Capon, J. (1969). "High-resolution frequency-wavenumber spectrum analysis," *Proc. IEEE,* vol. 57, pp. 1408–1418.

Caraiscos, C., and B. Liu (1984). "A roundoff error analysis of the LMS adaptive algorithm," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-32, pp. 34–41.

Carayannis, G., D. G. Manolakis, and N. Kalouptsidis (1983). "A fast sequential algorithm for least-squares filtering and prediction," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-31, pp. 1394–1402.

Cavanaugh, J. E., and R. H. Shumway (1996). "On computing the expected Fisher information matrix for state-space model parameters," *Statistics and Probability Letters,* vol. 26, pp. 347–355.

Chaitlin-Chatellin, F., and M. Ahués (1993). *Eigenvalues of Matrices,* Wiley, New York.

Chang, R. W. (1971). "A new equalizer structure for fast start-up digital communications," *Bell Syst. Tech. J.,* vol. 50, pp. 1969–2014.

Chao, J., H. Perez, and S. Tsujii (1990). "A fast adaptive filter algorithm using eigenvalue reciprocals as step sizes," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-38, pp. 1343–1352.

Chen, B., S. Zhao, P. Zhu, and J. C. Principe (2012). "Quantized kernel least mean square algorithm," *IEEE Trans. Neural Networks Learning Syst.,* vol. 23, pp. 23–32.

Chen, J., H. Bes, J. Vandewalle, and P. Janssens (1988). "A new structure for sub-band acoustic echo canceller," in *Proc. IEEE ICASSP,* New York, pp. 2574–2577.

Childers, D. G., ed. (1978). *Modern Spectrum Analysis,* IEEE Press, New York.

Cioffi, J. M. (1987). "Limited-precision effects in adaptive filtering," *IEEE Trans. Circuits Syst*., vol. CAS-34, pp. 821–833.

Cioffi, J. M. (1988). "High speed systolic implementation of fast QR adaptive filters," in *Proc. IEEE ICASSP,* New York, pp. 1584–1588.

Cioffi, J. M. (1990). "The fast adaptive rotor's RLS algorithm," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-38, pp. 631–653.

Cioffi, J. M., and T. Kailath (1984). "Fast, recursive-least-squares transversal filters for adaptive filtering," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-32, pp. 304–337.

Clark, G. A., S. K. Mitra, and S. R. Parker (1981). "Block implementation of adaptive digital filters," *IEEE Trans. Circuits Syst.,* vol. CAS-28, pp. 584–592.

Clark, G. A., S. R. Parker, and S. K. Mitra (1983). "A unified approach to time- and frequency-domain realization of FIR adaptive digital filters," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-31, pp. 1073–1083.

Cohen, L. (1995). *Time-Frequency Analysis,* Prentice-Hall, Upper Saddle River, NJ.

Cohn, A. (1922). "Über die Anzahl der Wurzeln einer algebraischen Gleichung in einem Kreise," *Math. Z.,* vol. 14, pp. 110–148.

Cowan, C. F. N. (1987). "Performance comparisons of finite linear adaptive filters," *IEE Proc. (London),* part F, vol. 134, pp. 211–216.

Cowan, C. F. N., and P. M. Grant (1985). *Adaptive Filters,* Prentice-Hall, Englewood Cliffs, NJ.

Cullum, J. K., and R. A. Willoughby (1985). *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Vol. 1, Theory,* Society for Industrial and Applied Mathematics, Philadelphia.

Cutler, C. C. (1952). *Differential Quantization for Communication Signals,* U.S. Patent 2,605,361.

de Courville, M., and P. Duhamel (1998). "Adaptive filtering in subbands using a weighted criterion," *IEEE Trans. Signal Processing,* vol. 46, pp. 2359–2371.

Deift, P. J., Demmel, C. Tomal, and L.-C. Li (1989). *The Bidiagonal Singular Value Decomposition and Hamiltonian Mechanics,* Rep. 458, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, New York.

Demmel, J., and W. Kahan (1990). "Accurate singular values of bidiagonal matrices," *SIAM J. Sci. Stat. Comp*., vol. 11, pp. 873–912.

Dewilde, P. (1969). *Cascade scattering matrix synthesis,* Ph.D. dissertation, Stanford University, Stanford, CA.

Ding, Z. (1997). "On convergence analysis of fractionally spaced adaptive blind equalizers," *IEEE Trans. Signal Process*., vol. 35, pp. 650–657.

Ding, Z., and Y. Li (2001). *Blind Equalization and Identification,* Marcel Decker, New York.

DiToro, M. J. (1965). "A new method for high speed adaptive signal communication through any time variable and dispersive transmission medium," in *1st IEEE Annu. Commun. Conf.,* pp. 763–767.

Dolph, C. L. (1946). "A current distribution for broadside arrays which optimizes the relationship between beam width and side-lobe level," *Proc. IRE,* vol. 34, pp. 335–348. (See also the discussion on this paper in *Proc. IRE,* vol. 35, pp. 489–492, May 1947.)

Doob, L. J. (1953). *Stochastic Processes,* Wiley, New York.

Douglas, S. C. (1994). "A family of normalized LMS algorithms," *IEEE Signal Process. Letters,* vol. 1, pp. 49–51.

Douglas, S. C., and T. H.-Y Meng (1994). "Normalized data nonlinearities for LMS adaptation," *IEEE Trans. Acoust. Speech Signal Process.,* vol. 42, pp. 1352–1365.

Duda, R. O., P. E. Hart, and D. G. Stork (2001). *Pattern Classification,* 2nd ed., Wiley, New York.

Durbin, J. (1960). "The fitting of time series models," *Rev. Int. Stat. Inst.,* vol. 28, pp. 233–244.

Duren, P. (2000). *Theory of HP Spaces,* Dover Publications, Mineola, NY.

Eckart, G., and G. Young (1936). "The approximation of one matrix by another of lower rank," *Psychometrika,* vol. 1, pp. 211–218.

Eleftheriou, E., and D. D. Falconer (1986). "Tracking properties and steady state performance of RLS adaptive filter algorithms," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-34, pp. 1097–1110.

El-Jaroudi, A., and J. Makhoul (1991). "Discrete all-pole modeling," *IEEE Trans. Signal Process*., vol. 39, pp. 411–423.

Endres, T. J., S. N. Hulyalkar, C. H. Strolle, and T. A. Schaffer (2001). "Low-complexity and low-latency implementation of the Godard/CMA update," *IEEE Trans. Commun.,* vol. 49, pp. 219–225.

Falconer, D. D., and L. Ljung (1978). "Application of fast Kalman estimation to adaptive equalization," *IEEE Trans. Commun*., vol. COM-26, pp. 1439–1446.

Fang, Y., and T. W. S. Chow (1999). "Blind equalization of a noisy channel by linear neural network," *IEEE Trans. Neural Networks,* vol. 10, pp. 918–924.

Farhang-Boroujeny, B. (1998). *Adaptive Filters: Theory and Applications,* Wiley, New York.

Ferrara, E. R., Jr. (1980). "Fast implementation of LMS adaptive filters," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-28, pp. 474–475.

Ferrara, E. R., Jr. (1985). "Frequency-domain adaptive filtering," in C. F. N. Cowan and P. M. Grant, eds., *Adaptive Filters,* Prentice-Hall, Englewood Cliffs, NJ, pp. 145–179.

Fisher, R. A. (1922). "On the mathematical foundation of theoretical statistics," *Philos. Trans. Royal Soc. London,* vol. A-222, pp. 309–368.

Forsythe, G. E., and P. Henrici (1960). "The cyclic Jacobi method for computing the principal values of a complex matrix," *Trans. Am. Math. Soc*., vol. 94, pp. 1–23.

Francis, B. A. (1987). A *Course in H-Infinity Control Theory,* Springer-Verlag, New York.

Francis, B. A., and G. Zames (1984). "On H-infinity-optimal sensitivity theory for SISO feedback systems," *IEEE Trans. Autom. Control,* vol. AC-29, pp. 9–16.

Franks, L. E. (1969). *Signal Theory,* Prentice-Hall, Englewood Cliffs, NJ.

Fraser, D. C. (1967). *A new technique for the optimal smoothing of data,* Sc.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Frieden, B. R. (2004). *Science from Fisher Information: A Unification,* Cambridge University Press, New York.

Frost, O. L., III (1972). "An algorithm for linearly constrained adaptive array processing," *Proc. IEEE,* vol. 60, pp. 926–935.

Fuhl, J. (1994). Diploma thesis, Technical University of Vienna, Vienna, Austria.

Gabriel, W. F. (1976). "Adaptive arrays: An introduction," *Proc. IEEE,* vol. 64, pp. 239–272.

Gardner, W. A. (1984). "Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis and critique," *Signal Process*., vol. 6, pp. 113–133.

Gardner, W. A. (1987). "Nonstationary learning characteristics of the LMS algorithm," *IEEE Trans. Circuits Syst*., vol. CAS-34, pp. 1199–1207.

Gardner, W. A. (1990). *Introduction to Random Processes with Applications to Signals and Systems,* McGraw-Hill, New York.

Gardner, W. A. (1991). "A new method of channel identification," *IEEE Trans. Commun*., vol. COM-39, pp. 813–817.

Gardner, W. A., ed. (1994a). *Cyclostationarity in Communications and Signal Processing,* IEEE Press, New York.

Gardner, W. A. (1994b). "An introduction to cyclostationary signals," in W. A. Gardner, ed., *Cyclostationarity in Communications and Signal Processing,* IEEE Press, New York, pp. 1–90.

Gardner, W. A., and L. E. Franks (1975). "Characterization of cyclostationary random signal processes," *IEEE Trans. Information Theory,* vol. IT-21, pp. 4–14.

Gauss, C. F. (1809). *Theoria motus corporum coelestium in sectionibus conicus solem ambientum,* Hamburg (translation: Dover, New York, 1963).

Gay, S. L., and J. Benesty, eds. (2000). *Acoustic Signal Processing for Telecommunication,* Kluwer Academic Press, Boston.

Gay, S. L., and S. Tavathia (1995). "The fast affine projection algorithm," *Proc. IEEE ICASSP,* pp. 3023–3026.

Gelb, A., ed. (1974). *Applied Optimal Estimation,* MIT Press, Cambridge, MA.

Gentleman, W. M., and H. T. Kung (1981). "Matrix triangularization by systolic arrays," *Proc. SPIE,* vol. 298, *Real Time Signal Processing* IV, pp. 298–303.

Gersho, A. (1968). "Adaptation in a quantized parameter space," in *Proc. Allerton Conf. on Circuit and System Theory,* Urbana, IL, pp. 646–653.

Gesbert, D., P. Duhamel, and S. Mayrargue (1997). "On-line blind multichannel equalization based on mutually referenced filters." *IEEE Trans. Signal Process*., vol. 45, pp. 2307–2317.

Gilloire, A., and M. Vetterli (1992). "Adaptive filtering in subbands with critical sampling: Analysis, experiments, and applications to acoustic echo cancellation," *IEEE Trans. Circuits Syst*., vol. 40, pp. 1862–1875.

Gitlin, R. D., J. E. Mazo, and M. G. Taylor (1973). "On the design of gradient algorithms for digitally implemented adaptive filters," *IEEE Trans. Circuit Theory,* vol. CT-20, pp. 125–136.

Gitlin, R. D., and S. B. Weinstein (1981). "Fractionally spaced equalization: An improved digital transversal equalizer," *Bell Syst. Tech. J.,* vol. 60, pp. 275–296.

Glover, J. R., Jr. (1977). "Adaptive noise cancelling applied to sinusoidal interferences," *IEEE Trans. Acoust. Speech Signal Process.,* vol. ASSP-25, pp. 484–491.

Godard, D. N. (1974). "Channel equalization using a Kalman filter for fast data transmission," *IBM J. Res. Dev.,* vol. 18, pp. 267–273.

Godard, D. N. (1980). "Self-recovering equalization and carrier tracking in a two-dimensional data communication system," *IEEE Trans. Commun.,* vol. COM-28, pp. 1867–1875.

Godfrey R., and F. Rocca (1981). "Zero memory non-linear deconvolution," *Geophys. Prospect.,* vol. 29, pp. 189–228.

Golub, G. H., and W. Kahan (1965). "Calculating the singular values and pseudo-inverse of a matrix," *J. SIAM Numer. Anal. B.,* vol. 2, pp. 205–224.

Golub, G. H., and C. F. Van Loan (1996). *Matrix Computations,* 3rd ed., The Johns Hopkins University Press, Baltimore.

Goodwin, G. C., and R. L. Payne (1977). *Dynamic System Identification: Experiment Design and Data Analysis,* Academic Press, New York.

Gray, R. M. (1972). "On the asymptotic eigenvalue distribution of Toeplitz matrices," *IEEE Trans. Information Theory,* vol. IT-18, pp. 725–730.

Gray, R. M. (1990). "Quantization noise spectra," *IEEE Trans. Information Theory,* vol. 36, pp. 1220–1244.

Gray, R. M., and L. D. Davisson (1986). *Random Processes: A Mathematical Approach for Engineers,* Prentice-Hall, Englewood Cliffs, NJ.

Gray, W. (1979). *Variable norm deconvolution,* Ph.D. dissertation, Department of Geophysics, Stanford University, Stanford, CA.

Grenander, U., and G. Szegö (1958). *Toeplitz Forms and Their Applications,* University of California Press, Berkeley, CA.

Griffiths, L. J. (1975). "Rapid measurement of digital instantaneous frequency," *IEEE Trans. Acoust. Speech Signal Process.,* vol. ASSP-23, pp. 207–222.

Griffiths, L. J. (1977). "A continuously adaptive filter implemented as a lattice structure," *Proc. IEEE ICASSP,* Hartford, CT, pp. 683–686.

Griffiths, L. J. (1978). "An adaptive lattice structure for noise-cancelling applications," Proc. *IEEE ICASSP,* Tulsa, OK, pp. 87–90.

Griffiths, L. J., and C. W. Jim (1982). "An alternative approach to linearly constrained optimum beamforming," *IEEE Trans. Antennas Propag.*, vol. AP-30, pp. 27–34.

Griffiths, L. J., F. R. Smolka, and L. D. Trembly (1977). "Adaptive deconvolution: A new technique for processing time-varying seismic data," *Geophysics,* vol. 42, pp. 742–759.

Hänsler, E., and G. Schmidt (2004). *Acoustic Echo and Noise Control: A Practical Approach,* Wiley, New York.

Hänsler, E., and G. Schmidt, eds. (2008). *Speech and Audio Processing in Adverse Environments,* Springer, London.

Hanson, R. J., and C. L. Lawson (1969). "Extensions and applications of the Householder algorithm for solving linear least squares problems*," Math. Comput.,* vol. 23, pp. 787–812.

Hardy, G. H. (1915). "On the mean value of the modulus of an analytic function," *Proc. London Math. Soc*., vol. 214, pp. 269–277.

Hassibi, B. (2003). "On the robustness of LMS filters," in S. Haykin and B. Widrow, eds., L*east-Mean-Square Adaptive Filters,* Wiley, New York, pp. 105–144.

Hassibi, B., A. H. Sayed, and T. Kailath (1993). "LMS is H^{∞} optimal," in *Proceedings of the 32nd IEEE Conference on Decision and Control,* San Antonio, TX, pp. 74–80.

Hassibi, B., A. H. Sayed, and T. Kailath (1996). "H^{∞} optimality of the LMS algorithm," *IEEE Trans. Signal Process.,* vol. 44, pp. 267–280.

Hassibi, B., and T. Kailath (2001). "*H*-infinity bounds for least-squares estimators," *IEEE Trans. Autom. Control,* vol. 46, pp. 309–314.

Hassibi, B., A. H. Sayed, and T. Kailath (1996). "H^{∞}-optimality of the LMS algorithm," *IEEE Trans. Signal Process.,* vol. 44, pp. 267–280.

Hatzinakos, D. (1990). *Blind equalization based on polyspectra,* Ph.D. thesis, Northeastern University, Boston, MA.

Hatzinakos, D., and C. L. Nikias (1989). "Estimation of multipath channel response in frequency selective channels," *IEEE J. Sel. Areas Commun*., vol. 7, pp. 12–19.

Hatzinakos, D., and C. L. Nikias (1991). "Blind equalization using a tricepstrum based algorithm," *IEEE Trans. Commun*., vol. COM-39, pp. 669–682.

Haykin, S. (1989). *Modern Filters,* Macmillan, New York.

Haykin, S. (1991). *Adaptive Filter Theory,* 2nd ed., Prentice-Hall, Englewood Cliffs, NJ.

Haykin, S, ed. (1994). *Blind Deconvolution,* Prentice-Hall, Upper Saddle River, NJ.

Haykin, S. (1996). *Adaptive Filter Theory,* 3rd ed., Prentice-Hall, Upper Saddle River, NJ.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation,* Prentice-Hall: Upper Saddle River, NJ.

Haykin, S., ed. (2000). *Unsupervised Adaptive Filtering, Volume I: Blind Source Separation,* and *Volume II: Blind Deconvolution,* Wiley, New York.

Haykin, S. (2001). *Communication Systems,* 4th ed., Wiley, New York.

Haykin, S. (2009). *Neural Networks and Learning Machines,* 3rd ed., Prentice-Hall, Upper Saddle River, NJ.

Haykin, S. (2013). *Digital Communication Systems,* Wiley, 2013.

Ho, K. C. (2000). "A study of two adaptive filters in tandem," *IEEE Trans. Signal Process*., vol. 48, pp. 1626–1636.

Ho, Y. C. (1963). "On the stochastic approximation method and optimal filter theory," *J. Math. Anal. Appl*., vol. 6, pp. 152–154.

Ho, Y. C., and R. C. K. Lee (1964). "A Bayesian approach to problems in stochastic estimation and control," *IEEE Trans. Autom. Control,* vol. AC-9, pp. 333–339.

Householder, A. S. (1958a). "Unitary triangularization of a nonsymmetric matrix," *J. Assoc. Comput. Mach*., vol. 5, pp. 339–342.

Householder, A. S. (1958b). "The approximate solution of matrix problems," *J. Assoc. Comput. Mach*., vol. 5, pp. 204–243.

Householder, A. S. (1964). *The Theory of Matrices in Numerical Analysis,* Blaisdell, Waltham, MA.

Howells, P. W. (1965). *Intermediate Frequency Sidelobe Canceller,* U.S. Patent 3,202,990.

Howells, P. W. (1976). "Explorations in fixed and adaptive resolution at GE and SURC," *IEEE Trans. Antennas Propag*., vol. AP-24, Special Issue on Adaptive Antennas, pp. 575–584.

Hua, Y., K. Abed-Meraim, and M. Wax (1997). "Blind system identification using minimum noise subspace," *IEEE Trans. Signal Process*., vol. 45, pp. 770–773.

Hudson, J. E., and T. J. Shepherd (1989). "Parallel weight extraction by a systolic least squares algorithm," in *Proc. SPIE, vol. 1152, Advanced Algorithms and Architectures for Signal Processing* IV, pp. 68–77.

Hughes, D. T., and J. G. McWhirter (1995). "Penalty function method for sidelobe control in least squares adaptive beamforming," *Proc. SPIE, vol. 2563, Advanced Signal Processing Algorithms*.

IEEE Computer Society (2008). *IEEE Standard for Floating-Point Arithmetic,* IEEE, doi:10.1109/IEEESTD.2008.4610935, IEEE Std. 754-2008.

Ioannou, P. A. (1990). "Robust adaptive control," *Proc. Sixth Yale Workshop on Adaptive and Learning Systems,* Yale University, New Haven, CT, pp. 32–39.

Ioannou, P. A., and E. V. Kokotovic (1983). *Adaptive Systems with Reduced Models,* Springer-Verlag, New York.

Itakura, F., and S. Saito (1970). "A statistical method for estimation of speech spectral density and formant frequencies," *Electron. Commun. Japan,* vol. 53-A, pp. 36–43.

Itakura, F., and S. Saito (1971). "Digital filtering techniques for speech analysis and synthesis," in *Proc. 7th Int. Conf. Acoust*., Budapest, vol. 25-C-1, pp. 261–264.

Itakura, F., and S. Saito (1972). "On the optimum quantization of feature parameters in the PARCOR speech synthesizer," *IEEE 1972 Conf. Speech Commun. Process*., New York, pp. 434–437.

Jablon, N. K. (1992). "Joint blind equalization, carrier recovery, and timing recovery for high-order QAM constellations," *IEEE Trans. Signal Process*., vol. 40, pp. 1383–1398.

Jacobi, C. G. J. (1846). "Über ein leichtes Verfahren, die in der Theorie der Säkularstörungen vorkommenden Gleichungen numerisch aufzulösen," *J. Reine Angew. Math*. vol. 30, pp. 51–95.

Jacobs, R. A. (1988). "Increased rates of convergence through learning rate adaptation," *Neural Networks,* vol. 1, pp. 295–307.

Jacobson, D. (1973). "Optimal stochastic linear systems with exponential criteria and their relation to deterministic differential games," *IEEE Trans. Autom. Control,* vol. AC-18, pp. 124–131.

Jaynes, E. T. (1982). "On the rationale of maximum-entropy methods," *Proc. IEEE,* vol. 70, pp. 939–952.

Johnson, C. R., Jr. (1991). "Admissibility in blind adaptive channel equalization: A tutorial survey of an open problem," *IEEE Control Systems Mag*., vol. 11, pp. 3–15.

Johnson, C. R., Jr., et al. (2000). "The core of FSE-CMA behavior theory," in S. Haykin, ed., *Unsupervised Adaptive Filtering, Vol. II: Blind Deconvolution,* pp. 13–112, Wiley, New York.

Jumarie, G. (1990). "Nonlinear filtering, a weighted mean-square approach and a Bayesian one via the maximum entropy principle," *Signal Process*., vol. 21, pp. 323–338.

Kailath, T. (1960). *Estimating Filters for Linear Time-Invariant Channels,* Quarterly Progress Rep. 58, MIT Research Laboratory for Electronics, Cambridge, MA, pp. 185–197.

Kailath, T. (1968). "An innovations approach to least-squares estimation: Part 1. Linear filtering in additive white noise," *IEEE Trans. Autom. Control,* vol. AC-13, pp. 646–655.

Kailath, T. (1970). "The innovations approach to detection and estimation theory," *Proc. IEEE,* vol. 58, pp. 680–695.

Kailath, T. (1974). "A view of three decades of linear filtering theory," *IEEE Trans. Information Theory,* vol. IT-20, pp. 146–181.

Kailath, T., ed. (1977). *Linear Least-Squares Estimation,* Benchmark Papers in Electrical Engineering and Computer Science, Dowden, Hutchinson & Ross, Stroudsburg, PA.

Kailath, T. (1980). *Linear Systems,* Prentice-Hall, Englewood Cliffs, NJ.

Kailath, T., and P. A. Frost (1968). "An innovations approach to least-squares estimation: Part 2. Linear smoothing in additive white noise," *IEEE Trans. Autom. Control,* vol. AC-13, pp. 655–660.

Kailath, T., and R. A. Geesey (1973). "An innovations approach to least-squares estimation: Part 5. Innovation representations and recursive estimation in colored noise," *IEEE Trans. Autom. Control,* vol. AC-18, pp. 435–453.

Kalman, R. E. (1960). "A new approach to linear filtering and prediction problems," *Trans. ASME, J. Basic Eng*., vol. 82, pp. 35–45.

Kalman, R. E., and R. S. Bucy (1961). "New results in linear filtering and prediction theory," *Trans. ASME, J. Basic Eng*., vol. 83, pp. 95–108.

Kaminski, P. G., A. E. Bryson, and S. F. Schmidt (1971). "Discrete square root filtering: A survey of current techniques," *IEE Trans. Autom. Control,* vol. AC-16, pp. 727–735.

Kay, S. M. (1988). *Modern Spectral Estimation: Theory and Application,* Prentice-Hall, Englewood Cliffs, NJ.

Kellerman, W. (1985). "Kompensation akustischer Echos in Frequenzteilbändern," *Aachener Kolloquium,* Aachen, FRG, pp. 322–325.

Kelly, E. J., I. S. Reed, and W. L. Root (1960). "The detection of radar echoes in noise: I," *J. SIAM,* vol. 8, pp. 309–341.

Kelly, J. L, Jr., and R. F. Logan (1970). *Self-Adaptive Echo Canceller,* U.S. Patent 3,500,000.

Kimeldorf, G. S., and G. Wahba (1971). "Some results on Tchebycheffian spline functions," *J. Math. Anal. Appl.,* vol. 33, pp. 82–95.

Kirsch, A. (1996). *An Introduction to the Mathematical Theory of Inverse Problems,* New York, Springer-Verlag.

Klema, V. C., and A. J. Laub (1980). "The singular value decomposition: Its computation and some applications," *IEEE Trans Autom. Control,* vol. AC-25, pp. 164–176.

Kloeden, P. E., and E. Platen (1995). *Numerical Solution of Stochastic Differential Equations,* 2nd corrected printing, Springer-Verlag, New York.

Kmenta, J. (1971). *Elements of Econometrics,* Macmillan, New York.

Kolmogorov, A. N. (1939). "Sur l'interpolation et extrapolation des suites stationaries," *C.R. Acad. Sci*., Paris, vol. 208, pp. 2043–2045. [English translation reprinted in Kailath, 1977.]

Kolmogorov, A. N. (1968). "Three approaches to the quantitative definition of information," *Probl. Inf. Transm. USSR,* vol. 1, pp. 1–7.

Krein, M. G. (1945). "On a problem of extrapolation of A. N. Kolmogorov," *C. R. (Dokl.) Akad. Nauk SSSR,* vol. 46, pp. 306–309. [Reproduced in Kailath, 1977.]

Kullback, S., and R. A. Leibler (1951). "On information and sufficiency," *Ann. Math. Statist*., vol. 22, pp. 79–86.

Kung, H. T. (1982). "Why systolic architectures?" *Computer,* vol. 15, pp. 37–46.

Kung, H. T., and C. E. Leiserson (1978). "Systolic arrays (for VLSI)," *Sparse Matrix Proc. 1978, Soc. Ind. Appl. Math*., 1978, pp. 256–282. [A version of this paper is reproduced in Mead and Conway, 1980.]

Kushner, H. J. (1984). *Approximation and Weak Convergence Methods for Random Processes with Applications to Stochastic System Theory,* MIT Press, Cambridge, MA.

Kushner, H. J., and D. S. Clark (1978). *Stochastic Approximation Methods for Constrained and Unconstrained Systems,* Springer-Verlag, New York.

Lanczos, C. (1964). *Linear Differential Operators,* London: Van Nostrand.

Langevin, P. (1908). "Sur la théorie du mouvement brownien," *C. R. Acad. Sci. (Paris),* vol. 146, pp. 530–533.

Lawrence, R. E., and H. Kaufman (1971). "The Kalman filter for the equalization of a digital communication channel," *IEEE Trans. Commun. Technol*., vol. COM-19, pp. 1137–1141.

Lee, D. T. L., M. Morf, and B. Friedlander (1981). "Recursive least-squares ladder estimation algorithms," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 467–481.

Lee, J. C., and C. K. Un (1989). "Performance analysis of frequency-domain block LMS adaptive digital filters," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 173–189.

Legendre, A. M. (1810). "Méthode des moindres quarrés, pour trouver le milieu le plus probable entre les résultats de différentes observations," *Mem. Inst. France,* pp. 149–154.

Lehmer, D. H. (1961). "A machine method for solving polynomial equations," *J. Assoc. Comput. Mach.*, vol. 8, pp. 151–162.

Leung, H., and S. Haykin (1989). "Stability of recursive QRD-LS algorithms using finite-precision systolic array implementation," *IEEE Trans. Acoust. Speech Signal Process.*, vol. ASSP-37, pp. 760–763.

Levin, M. D., and C. F. N. Cowan (1994). "The performance of eight recursive least squares adaptive filtering algorithms in a limited precision environment," in *Proc. European Signal Process. Conf.*, Edinburgh, pp. 1261–1264.

Levinson, N. (1947). "The Wiener RMS (root-mean-square) error criterion in filter design and prediction," *J. Math Phys.*, vol. 25, pp. 261–278.

Li, Y., and Z. Ding (1995). "Convergence analysis of finite length blind adaptive equalizers," *IEEE Trans. Signal Process.*, vol. 43, pp. 2120–2129.

Li, Y., and Z. Ding (1996). "Global convergence of fractionally spaced Godard (CMA) adaptive equalizers," *IEEE Trans. Signal Process.*, vol. 44, pp. 818–826.

Liavas, A. P., and P. A. Regalia (1999). "On the numerical stability and accuracy of the conventional recursive least-squares algorithm," *IEEE Trans. Signal Process.*, vol. 47, pp. 88–96.

Liavas, A. P., P. A. Regalia, and J.-P Delmas (1999a). "Robustness of least-squares and subspace methods for blind channel identification/equalization with respect to effective channel undermodeling/overmodeling," *IEEE Trans. Signal Process.*, vol. 47, pp. 1636–1645.

Liavas, A. P., P. A. Regalia, and J.-P Delmas (1999b). "Blind channel approximation: Effective channel order determination," *IEEE Trans. Signal Process.*, vol. 47, pp. 336–344.

Ling, F. (1989). "Efficient least-squares lattice algorithms based on Givens rotation with systolic array implementations," in *Proc. IEEE ICASSP,* Glasgow, pp. 1290–1293.

Ling, F., and J. G. Proakis (1984). "Numerical accuracy and stability: Two problems of adaptive estimation algorithms caused by round-off error," in *Proc. ICASSP,* San Diego, pp. 30.3.1–30.3.4.

Ling, F., D. Manolakis, and J. G. Proakis (1985). "New forms of LS lattice algorithms and an analysis of their round-off error characteristics," in *Proc. IEEE ICASSP,* Tampa, pp. 1739–1742.

Liu, K. J. R., S.-F. Hsieh, and K. Yao (1992). "Systolic block Householder transformation for RLS algorithm with two-level pipelined implementation," IEEE *Trans. Signal Process.*, vol. 40, pp. 946–958.

Liu, W., J. C. Principe, and S. Haykin (2010). *Kernel Adaptive Filtering: A Comprehensive Introduction,* Wiley, New York.

Ljung, S., and L. Ljung (1985). "Error propagation properties of recursive least-squares adaptation algorithms," *Automatica,* vol. 21, pp. 157–167.

Lucky, R. W. (1965). "Automatic equalization for digital communication," *Bell Syst. Tech. J.*, vol. 44, pp. 547–588.

Lucky, R. W. (1966). "Techniques for adaptive equalization of digital communication systems," *Bell Syst. Tech. J.*, vol. 45, pp. 255–286.

Luk, F. T. (1986). "A triangular processor array for computing singular values," *Linear Algebra Applications,* vol. 77, pp. 259–273.

Luk, F. T., and H. Park (1989). "A proof of convergence for two parallel Jacobi SVD algorithms," *IEEE Trans. Comput.*, vol. 38, pp. 806–811.

Macchi, O. (1986a). "Advances in Adaptive Filtering," in E. Biglieri and G. Prati, eds., *Digital Communications,* North-Holland, Amsterdam, pp. 41–57.

Macchi, O. (1986b). "Optimization of adaptive identification for time-varying filters," *IEEE Trans. Autom. Control,* vol. AC-31, pp. 283–287.

Macchi, O. (1995). *Adaptive Processing: The LMS Approach with Applications in Transmission,* Wiley, New York.

Macchi, O., and N. J. Bershad (1991). "Adaptive recovery of a chirped sinusoid in noise, Part I: Performance of the RLS algorithm," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 39, pp. 583–594.

Macchi, O., and E. Eweda (1984). "Convergence analysis of self-adaptive equalizers," *IEEE Trans. Information Theory,* vol. IT-30, Special Issue on Linear Adaptive Filtering, pp. 161–176.

Mader, A., H. Puder, and G. V. Schmidt (2000). "Step-size control for acoustic echo cancellation filters—An overview," *Signal Process.*, vol. 80, pp. 1697–1719.

Mahmood, A. R. (2010). *Automatic step-size adaptation in incremental supervised learning,* M.Sc. Thesis, Department of Computing Science, University of Alberta, Edmonton, Canada.

Mahmood, A. R. (2013). Private communications.

Mahmood, A. R., R. S. Sutton, T. Degris, and P. M. Pilarski (2012). "Tuning-free step-size adaptation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP),* March 25–30, Kyoto, Japan, pp. 2121–2124.

Makhoul, J. (1975). "Linear prediction: A tutorial review," *Proc. IEEE,* vol. 63, pp. 561–580.

Makhoul, J. (1977). "Stable and efficient lattice methods for linear prediction," *IEEE Trans. Acoust. Speech Signal Process.*, vol. ASSP-25, pp. 423–428.

Makhoul, J. (1978). "A class of all-zero lattice digital filters: Properties and applications," *IEEE Trans. Acoust. Speech Signal Process.*, vol. ASSP-26, pp. 304–314.

Makhoul, J. (1981). "On the eigenvectors of symmetric Toeplitz matrices," *IEEE Trans. Acoust. Speech Signal Process.*, vol. ASSP-29, pp. 868–872.

Makhoul, J., and L. K. Cossell (1981). "Adaptive lattice analysis of speech," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 494–499.

Mannerkoski, J., and D. P. Taylor (1999). "Blind equalization using least-squares lattice prediction," *IEEE Trans. Signal Process.*, vol. 47, pp. 630–640.

Mansour, D., and A. H. Gray, Jr. (1982). "Unconstrained frequency-domain adaptive filters," *IEEE Trans. Acoust. Speech Signal Process.*, vol. ASSP-30, pp. 726–734.

Marden, M. (1949). "The geometry of the zeros of a polynomial in a complex variable," *Am. Math. Soc. Surveys,* no. 3, chap. 10, American Mathematical Society, New York.

Markel, J. D., and A. H. Gray, Jr. (1976). *Linear Prediction of Speech,* Springer-Verlag, New York.

Marple, S. L., Jr. (1987). *Digital Spectral Analysis with Applications,* Prentice-Hall, Englewood Cliffs, NJ.

Maybeck, P. S. (1979). *Stochastic Models, Estimation, and Control,* vol. 1, Academic Press, New York.

Mazo, J. E. (1979). "On the independence theory of equalizer convergence," *Bell Syst. Tech. J.*, vol. 58, pp. 963–993.

Mazo, J. E. (1980). "Analysis of decision-directed equalizer convergence," *Bell Syst. Tech. J.*, vol. 59, pp. 1857–1876.

McAulay, R. J. M. (1984). "Maximum likelihood spectral estimation and its application to narrow- band speech," *IEEE Trans. Signal Process.*, vol. ASSP-32, pp. 243–251.

McCool, J. M., et al. (1980). *Adaptive Line Enhancer,* U.S. Patent 4,238,746.

McDonald, R. A. (1966). "Signal-to-noise performance and idle channel performance of differential pulse code modulation systems with particular applications to voice signals," *Bell Syst. Tech. J.*, vol. 45, pp. 1123–1151.

McDonough, R. N., and A. D. Whalen (1995). *Detection of Signals in Noise,* 2nd ed., Academic Press, San Diego, CA.

McGee, W. F. (1971). "Complex Gaussian noise moments," *IEEE Trans. Information Theory,* vol. IT-17, pp. 149–157.

McWhirter, J. G. (1983). "Recursive least-squares minimization using a systolic array," *Proc. SPIE,* vol. 431, *Real-Time Signal Processing* VI, San Diego, pp. 105–112.

McWhirter, J. G., and I. K. Proudler (1993). "The QR family," in N. Kalouptsidis and S. Theodoridis, eds., *Adaptive System Identification and Signal Processing Algorithms,* Prentice-Hall, Englewood Cliffs, NJ, pp. 260–321.

McWhirter, J. G., and T. J. Shepherd (1989). "Systolic array processor for MVDR beamforming," *IEE Proc. (London), Part F,* vol. 136, pp. 75–80.

McWhirter, J. G., H. D. Rees, S. D. Hayward, and J. L. Mather (2000). "Adaptive radar processing," *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium,* Lake Louise, AB, Canada, pp. 25–30.

McWhorther, L. T., and L. L. Scharf (1995). "Nonlinear maximum likelihood estimation of autoregressive time series," *IEEE Trans. Signal Process.*, vol. 43, pp. 2909–2919.

Mead, C., and L. Conway (1980). *Introduction to VLSI Systems,* Addison-Wesley, Reading, MA.

Mercer, J. (1909). "Functions of positive and negative type and their connection with the Theory of Integral Equations," *Philos. Trans. Royal Soc. London,* vol. A-209, pp. 415–446.

Miller, K. S. (1974). *Complex Stochastic Processes: An Introduction to Theory and Application,* Addison-Wesley, Reading, MA.

Molisch, A. F. (2011). *Wireless Communications,* 2nd ed., Wiley, New York.

Monsen, P. (1971). "Feedback equalization for fading dispersive channels," *IEEE Trans. Information Theory,* vol. IT-17, pp. 56–64.

Monzingo, R. A., and T. W. Miller (1980). *Introduction to Adaptive Arrays,* Wiley-Interscience, New York.

Morf, M. (1974). *Fast algorithms for multivariable systems,* Ph.D. dissertation, Stanford University, Stanford, CA.

Morf, M., and T. Kailath (1975). "Square-root algorithms for least-squares estimation," *IEEE Trans. Autom. Control,* vol. AC-20, pp. 487–497.

Morf, M., and D. T. Lee (1978). "Recursive least squares ladder forms for fast parameter tracking," in *Proc. 1978 IEEE Conf. Decision Control,* San Diego, pp. 1362–1367.

Morse, P. M., and H. Feshbach (1953). *Methods of Theoretical Physics,* Pt. I, McGraw-Hill, New York.

Moulines, E., P. Duhamel, J.-F. Cardoso, and S. Mayrargue (1995). "Subspace methods for blind identification of multichannel FIR filters," *IEEE Trans. Signal Process*., vol. 43, pp. 516–525.

Moustakides, G. V. (1997). "Study of the transient phase of the forgetting factor RLS," *IEEE Trans. Signal Process*., vol. 45, pp. 2468–2476.

Muirhead, R. J. (1982). *Aspects of Multivariate Statistical Theory,* Wiley, New York.

Mullis, C. T., and L. L. Scharf (1991). "Quadratic estimation of power spectrum," in S. Haykin, ed., *Advances in Spectrum Analysis,* Prentice-Hall, Englewood Cliffs, NJ, pp. 1–57.

Nagumo, J. I., and A. Noda (1967). "A learning method for system identification," *IEEE Trans. Autom. Control,* vol. AC-12, pp. 282–287.

Narayan, S. S., A. M. Peterson, and M. J. Narashima (1983). "Transform domain LMS algorithm," *IEEE Trans. Acoust., Speech Signal Process*., vol. ASSP-31, pp. 609–615.

Narendra, K. S., and A. M. Annaswamy (1989). *Stable Adaptive Systems,* Prentice-Hall, Englewood Cliffs, NJ.

Nascimento, V. H., and A. H. Sayed (1999). "Unbiased and stable leakage-based adaptive filters," *IEEE Trans. Signal Process*., vol. 47, pp. 3261–3276.

Nascimento, V. H., and A. H. Sayed (2000). "On the learning mechanism of adaptive algorithms," *IEEE Trans. Signal Process*., vol. 48, pp. 1609–1625.

Nikias, C. L., and M. R. Raghuveer (1987). "Bispectrum estimation: A digital signal processing framework," *Proc. IEEE,* vol. 75, pp. 869–891.

North, D. O. (1963). "An analysis of the factors which determine signal/noise discrimination in pulsed carrier systems," *Proc. IEEE,* vol. 51, pp. 1016–1027.

North, R. C., J. R. Zeidler, W. H. Ku, and T. R. Albert (1993). "A floating-point arithmetic error analysis of direct and indirect coefficient updating techniques for adaptive lattice filters," *IEEE Trans. Signal Process*., vol. 41, pp. 1809–1823.

Oja, E. (1982). "A simplified neuron model as a principal component analyzer," *J. Math. Biol*., vol. 15, pp. 267–273.

Oppenheim, A. V., and R. W. Schafer (1989). *Discrete-Time Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ.

Owsley, N. L. (1985). "Sonar array processing," in S. Haykin, ed., *Array Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ, pp. 115–193.

Pan, C. T., and R. J. Plemmons (1989). "Least squares modifications with inverse factorizations: Parallel implications," *J. Comput. Appl. Math*., vol. 27, pp. 109–127.

Pan, R., and C. L. Nikias (1988). "The complex cepstrum of higher order cumulants and nonminimum phase identification," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-36, pp. 186–205.

Papadias, C. (1995). *Methods for blind equalization and identification of linear channels,* Ph.D. thesis, École Nationale Supérieure des Télécommunications, Paris, France.

Papadias, C. S., and D. T. M. Slock (1999). "Fractionally spaced equalization of linear polyphase channels and related blind techniques based on multichannel linear prediction," *IEEE Trans. Signal Process*., vol. 47, pp. 641–654.

Parlett, B. N. (1998). "The symmetric eigenvalue problem," in *Classics in Applied Mathematics,* vol. 20, Society for Industrial and Applied Mathematics, Philadelphia, PA. Corrected reprint of the 1980 original.

Petraglia, M. R., and S. K. Mitra (1993). "Performance analysis of adaptive filter structures based on subband decomposition," in *Proceedings of International Symposium on Circuits and Systems,* Chicago, pp. I.60–I.63.

Pinchas, M. (2012). *The Whole Story Behind Blind Adaptive Equalizers/Blind Deconvolution,* Bentham eBooks, Oak Park, IL.

Pinchas, M., and B. Z. Bobrovsky (2006). "A maximum entropy approach for blind deconvolution," *Signal Process*., vol. 86, pp. 2913–2931.

Plackett, R. L. (1950). "Some theorems in least squares," *Biometrika,* vol. 37, p. 149.

Pradhan, S. S., and V. U. Reddy (1999). "A new approach to subband adaptive filtering," *IEEE Trans. Signal Processing,* vol. 47, pp. 655–664.

Press, W. H., et al. (1988). *Numerical Recipes in C,* Cambridge University Press, Cambridge, U.K.

Priestley, M. B. (1981). *Spectral Analysis and Time Series,* vols. 1 and 2, Academic Press, New York.

Proakis, J. G., and J. H. Miller (1969). "An adaptive receiver for digital signaling through channels with intersymbol interference," *IEEE Trans. Information Theory,* vol. IT-15, pp. 484–497.

Proudler, I. K., J. G. McWhirter, and T. J. Shepherd (1988). "Fast QRD-based algorithms for least squares linear prediction," in *Proc. IMA Conf. Math. Signal Process*., Warwick, U.K.

Proudler, I. K., J. G. McWhirter, and T. J. Shepherd (1989). "QRD-based lattice filter algorithms," *Proceedings of the SPIE—The International Society for Optical Engineering,* vol. 1152, pp. 56–67.

Quatieri, T. F. (2001). *Discrete-Time Speech Signal Processing: Principles and Practice,* Prentice-Hall, Upper Saddle River, NJ.

Rader, C. M. (1990). "Linear systolic array for adaptive beamforming," in *1990 Digital Signal Processing Workshop,* New Paltz, NY, Sponsored by IEEE Signal Processing Society, pp. 5.2.1–5.2.2.

Ralston, A. (1965). *A First Course in Numerical Analysis,* McGraw-Hill, New York.

Reed, I. S. (1962). "On a moment theorem for complex Gaussian processes," *IRE Trans. Information Theory,* vol. IT-8, pp. 194–195.

Reed, I. S., J. D. Mallet, and L. E. Brennan (1974). "Rapid convergence rate in adaptive arrays," *IEEE Trans. Aerospace Electron. Syst*., vol. AES-10, pp. 853–863.

Reeves, A. H. (1975). "The past, present, and future of PCM," *IEEE Spectrum,* vol. 12, pp. 58–63.

Regalia, P. A., and G. Bellanger (1991). "On the duality between fast QR methods and lattice methods in least squares adaptive filtering," *IEEE Trans. Signal Process*., vol. 39, pp. 879–891.

Reif, F. (1965). *Fundamentals of Statistical and Thermal Physics,* McGraw-Hill, New York.

Rickard, J. T., and J. R. Zeidler (1979). "Second-order output statistics of the adaptive line enhancer," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-27, pp. 31–39.

Riesz, R. (1923). "Ueber die Randwerte einer analytischen Funktion," *Math. Z.,* vol. 18, pp. 87–95.

Rissanen, J. (1978). "Modelling by shortest data description," *Automatica,* vol. 14, pp. 465–471.

Rissanen, J. (1989). *Stochastic complexity in statistical enquiry,* Series in Computer Science, vol. 15, World Scientific, Singapore.

Robbins, H., and S. Monro (1951). "A stochastic approximation method," *Ann. Math. Stat*., vol. 22, pp. 400–407.

Robinson, E. A. (1957). "Predictive decomposition of seismic traces," *Geophysics,* vol. 22, pp. 767–778.

Robinson, E. A. (1982). "A historical perspective of spectrum estimation," *Proc. IEEE,* vol. 70, Special Issue on Spectral Estimation, pp. 885–907.

Robinson, E. A., and T. Durrani (1986). *Geophysical Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ.

Rombouts, G., and M. Moonen (2000). "A fast exact frequency domain implementation to the exponentially windowed affine projection algorithm," in *Proceedings of IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium,* Lake Louise, AB, Canada, pp. 342–346.

Rontogiannis, A. A., and S. Theodoridis (1998a). "Multichannel fast QRD-LS adaptive filtering: New techniques and algorithms," *IEEE Trans. Signal Process.,* vol. 46, pp. 2862–2876.

Rontogiannis, A. A., and S. Theodoridis (1998b). "New fast QR decomposition least squares adaptive algorithms," *IEEE Trans. Signal Process*., vol. 46, pp. 2113–2121.

Rosenbrock, H. H. (1970). *State-Space and Multivariable Theory,* Wiley, New York.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). "Representations by back-propagating errors," *Nature,* vol. 323, pp. 533–536.

Saad, Y. (2011). *Numerical Methods for Large Eigenvalue Problems,* 2nd ed., Society for Industrial and Applied Mathematics, Philadelphia, PA.

Saito, S., and F. Itakura (1966). *The Theoretical Consideration of Statistically Optimum Methods for Speech Spectral Density,* Rep. 3107, Electrical Communication Laboratory, N. T. T., Tokyo (in Japanese).

Sakrison, D. (1966). "Stochastic approximation: A recursive method for solving regression problems," in A. V. Balakrishnan, ed., *Advances in Communication Systems,* vol. 2, Academic Press, New York, pp. 51–106.

Sankaran, S. G., and A. A. Beex (2000). "Convergence behavior of affine projection algorithms," *IEEE Trans. Signal Process*., vol. 48, pp. 1086–1096.

Sato, Y. (1975a). "A method of self-recovering equalization for multilevel amplitude-modulation systems," *IEEE Trans. Commun.,* vol. 23, pp. 679–682.

Sato, Y. (1975b). "Two extensional applications of the zero-forcing equalization method," *IEEE Trans. Commun*., vol. COM-23, pp. 684–687.

Satorius, E. H., and S. T. Alexander (1979). "Channel equalization using adaptive lattice algorithms," *IEEE Trans. Commun*., vol. COM-27, pp. 899–905.

Satorius, E. H., and J. D. Pack (1981). "Application of least squares lattice algorithms to adaptive equalization," *IEEE Trans. Commun*., vol. COM-29, pp. 136–142.

Sayed, A. H. (2003). *Fundamentals of Adaptive Filtering,* IEEE-Wiley Interscience, Hoboken, NJ.

Sayed, A. H., and T. Kailath (1994). "A state-space approach to adaptive RLS filtering," *IEEE Signal Process. Mag*., vol. 11, pp. 18–60.

Scharf, L. L., and D. W. Tufts (1987). "Rank reduction for modeling stationary signals," *IEEE Trans. Acoust., Speech Signal Process*., vol. ASSP-35, pp. 350–355.

Schmidt, R. O. (1979). "Multiple emitter location and signal parameter estimation," in *Proc. RADC Spectral Estimation Workshop,* Griffith AFB, Rome, NY, pp. 243–258.

Schmidt, R. O. (1981). *A signal subspace approach to multiple emitter location and spectral estimation,* Ph.D. dissertation, Stanford University, Stanford, CA.

Schneider, W. A. (1978). "Integral formulation for migration in two and three dimensions," *Geophysics,* vol. 43, pp. 49–76.

Schnitter, P., and C. R. Johnson, Jr. (1999). "Dithered signed-error CMA: Robust, computationally efficient blind adaptive equalization," *IEEE Trans. Signal Process,* vol. 47, pp. 1592–1603.

Schölkopf, B., and A. J. Smola (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond,* MIT Press, Cambridge, MA.

Schroeder, M. R. (1966). "Vocoders: Analysis and synthesis of speech," *Proc. IEEE,* vol. 54, pp. 720–734.

Schur, I. (1917). "Über Potenzreihen, die im Innern des Einheitskreises beschränkt sind," *J. Reine Angew. Math*., vol. 147, pp. 205–232; vol. 148, pp. 122–145.

Schuster, A. (1898). "On the investigation of hidden periodicities with applications to a supposed 26-day period of meterological phenomena," *Terr. Magn. Atmos. Electr*., vol. 3, pp. 13–41.

Schwartz, G. (1978). "Estimating the dimension of a model," *Ann. Stat*., vol. 6, pp. 461–464.

Schwartz, G. E. (1989). "Estimating the dimension of a model," *Ann. Stat*., vol. 6, pp. 461–464.

Sethares, W. A., D. A. Lawrence, C. R. Johnson, Jr., and R. R. Bitmead (1986). "Parameter drift in LMS adaptive filters," *IEEE Trans. Acous. Speech Signal Process*., vol. ASSP-34, pp. 868–879.

Shalvi, O., and E. Weinstein (1990). "New criteria for blind equalization of non-minimum phase systems (channels)," *IEEE Trans. Information Theory,* vol. 36, pp. 312–321.

Shannon, C. E. (1948). "A mathematical theory of communication," *Bell Syst. Tech. J.,* vol. 27, pp. 379–423, 623–656.

Shepherd, T. J., and J. G. McWhirter (1993). "Systolic adaptive beamforming," in S. Haylan, J. Litua, and T. J. Shepherd, eds., *Radar Array Processing,* Springer-Verlag, New York, pp. 153–243.

Sherwood, D. T., and N. J. Bershad (1987). "Quantization effects in the complex LMS adaptive algorithm: Linearization using dither-theory," *IEEE Trans. Circuits Syst*., vol. CAS-34, pp. 848–854.

Shynk, J. J. (1992). "Frequency-domain and multirate adaptive filtering," *IEEE Signal Process. Mag*., vol. 9, no. 1, pp. 14–37.

Shynk, J. J., and B. Widrow (1986). "Bandpass adaptive pole-zero filtering," in *Proc. IEEE ICASSP,* Tokyo, Japan, pp. 2107–2110.

Shynk, J. J., R. P. Gooch, G. Krishnamurthy, and C. K. Chan (1991). "A comparative performance study of several blind equalization algorithms," in *Proc. SPIE, Adaptive Signal Processing,* vol. 1565, San Diego, pp. 102–117.

Skidmore, I. D., and I. K. Proudler (2001). "KAGE: A new fast RLS algorithm," in *Proc. IEEE ICASSP* 2001, Salt Lake City, UT.

Slepian, D. (1978). "Prolate spheroidal wave functions, Fourier analysis, and uncertainty—V: The discrete case," *Bell Syst. Tech*. J., vol. 57, pp. 1371–1430.

Slock, D. T. M. (1994). "Blind fractionally-spaced equalization, perfect-reconstruction filter banks and multichannel linear prediction," in *Proc. IEEE ICASSP,* Adelaide, Australia, vol. 4, pp. 585–588.

Slock, D. T. M., and T. Kailath (1991). "Numerically stable fast transversal filters for recursive least squares adaptive filtering," *IEEE Trans. Signal Process*. vol. 39, pp. 92–114.

Slock, D. T. M., and T. Kailath (1993). "Fast transversal RLS algorithms," in N. Kalouptsidis and S. Theodoridis, eds., *Adaptive System Identification and Signal Processing Algorithms,* Prentice-Hall, Englewood Cliffs, NJ, pp. 123–190.

Slock, D. T. M., and C. B. Papadias (1995). "Further results on blind identification and equalization of multiple FIR channels," in *Proc. ICASSP,* Detroit, vol. 3, pp. 1964–1967.

Solo, V. (1992). "The error variance of LMS with time-varying weights," *IEEE Trans. Signal Process*., vol. 40, pp. 803–813.

Sommen, P. C. W., and J. A. K. S. Jayasinghe (1988). "On frequency-domain adaptive filters using the overlap-add method," in *Proc. IEEE Int. Symp. Circuits Syst*., Espoo, Finland, pp. 27–30.

Sommen, P C. W., P. J. Van Gerwen, H. J. Kotmans, and A. E. J. M. Jansen (1987). "Convergence analysis of a frequency-domain adaptive filter with exponential power averaging and generalized window function," *IEEE Trans. Circuits Syst*., vol. CAS-34, pp. 788–798.

Sondhi, M. M. (1967). "An adaptive echo canceller," *Bell Syst. Tech. J*., vol. 46, pp. 497–511.

Sondhi, M. M. (1970). *Closed Loop Adaptive Echo Canceller Using Generalized Filter Networks,* U.S. Patent 3,499,999.

Sondhi, M., and D. A. Berkley (1980). "Silencing echoes in the telephone network," *Proc. IEEE,* vol. 68, pp. 948–963.

Sorenson, H. W. (1970). "Least-squares estimation: From Gauss to Kalman," *IEEE Spectrum,* vol. 7, pp. 63–68.

Spall, J. L. (2003). *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control,* Wiley, New York.

Steinhardt, A. O. (1988). "Householder transforms in signal processing," *IEEE ASSP Mag*., vol. 5, pp. 4–12.

Stengel, R. F. (1986). *Optimal Control and Estimation,* Dover Publications, Mineola, NY.

Stewart, G. W. (1973). *Introduction to Matrix Computations,* Academic Press, New York.

Stewart, G. W., and J. G. Sun (1990). *Matrix Perturbation Theory,* Academic Press, London.

Stewart, R. W., and R. Chapman (1990). "Fast stable Kalman filter algorithms utilizing the square root," in *Proc. ICASSP,* Albuquerque, NM, pp. 1815–1818.

Stoer, J., and R. Bullirsch (1980). *Introduction to Numerical Analysis,* Springer-Verlag, New York.

Strang, G. (1980). *Linear Algebra and Its Applications,* 2nd ed., Academic Press, New York.

Sutton, R. S. (1992). "Adapting bias by gradient descent: An incremental version of delta-bar-delta," *Proceedings of the Tenth National Conference on Artificial Intelligence,* MIT Press, Cambridge, MA, pp. 171–176.

Swami, A., and J. M. Mendel (1990). "Time and lag recursive computation of cumulants from a state-space model," *IEEE Trans. Autom. Control,* vol AC-35, pp. 4–17.

Swerling, P. (1958). *A Proposed Stagewise Differential Correction Procedure for Satellite Tracking and Prediction,* Rep. P-1292, Rand Corporation, Santa Monica, CA.

Swerling, P. (1963). "Comment on a statistical optimizing navigation procedure for space flight,"' *AIAA J*., vol. 1, p. 1968.

Szegö, G. (1939). *Orthogonal polynomials,* Colloquium Publications, no. 23, American Mathematical Society, Providence, RI.

Tanaka, M., S. Makino, and L. Kojima (1999). "A block exact fast affine projection algorithm," *IEEE Trans. Speech Audio Process*., vol. 7, pp. 79–86.

Theodoridis, S. (2012). Private communications.

Theodoridis, S., K. Slavakis, and I. Yamada (2011). "Adaptive learning in a world of projections: A unified framework for linear and nonlinear regression and classification tasks," *IEEE Signal Processing Magazine,* vol. 28, pp. 97–123.

Thomas, J. B. (1969). *An Introduction to Statistical Communication Theory,* Wiley, New York.

Thomson, D. J. (1982). "Spectral estimation and harmonic analysis," *Proc. IEEE,* vol. 70, pp. 1055–1096.

Tikhonov, A. N. (1963). "On solving incorrectly posed problems and method of regularization," *Doklady Akademii Nauk USSR,* vol. 151, pp. 501–504.

Tikhonov, A. N. (1973). "On regularization of ill-posed problems," *Doklady Akademii Nauk USSR,* vol. 153, pp. 49–52.

Tikhonov, A. N., and V. Y. Arsenin (1977). *Solutions of Ill-Posed Problems,* Washington, DC: W. H. Winston.

Tong, L., and S. Perreau (1998). "Multichannel blind identification: From subspace to maximum likelihood methods," *Proc. IEEE,* vol. 86, pp. 1951–1968.

Tong, L., G. Xu, B. Hassibi, and T. Kailath (1995). "Blind identification and equalization based on second-order statistics: A frequency-domain approach," *IEEE Trans. Information Theory,* vol. 41, pp. 329–334.

Tong, L., G. Xu, and T. Kailath (1993). "Fast blind equalization via antenna arrays," in *Proc. IEEE ICASSP,* Minneapolis, vol. 4, pp. 272–275.

Tong, L., G. Xu, and T. Kailath (1994a). "Blind identification and equalization based on secondorder statistics: A time-domain approach," *IEEE Trans. Information Theory,* vol. 40, pp. 340–349.

Tong, L., G. Xu, and T. Kailath (1994b). "Blind channel identification and equalization using spectral correlation measurements, Part II: A time-domain approach," in W. A. Gardner, ed., *Cyclostationarity in Communications and Signal Processing,* IEEE Press, New York, pp. 437–454.

Treichler, J. R. (1979). "Transient and convergent behavior of the adaptive line enhancer," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-27, pp. 53–62.

Treichler, J. R., and B. G. Agee (1983). "A new approach to multipath correction of constant modulus signals," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-31, pp. 459–471.

Tretter, S. A. (1976). *Introduction to Discrete-Time Signal Processing,* Wiley, New York.

Tugnait, J. K. (2003). "Blind equalization techniques," in J. G. Proakis, ed., *Encyclopedia of Telecommunications,* Wiley, New York.

Uhlenbeck, G. E., and L. S. Ornstein (1930). "The theory of the Brownian motion, I," *Phys. Rev*., vol. 36, p. 823, Section 2.

Ungerboeck, G. (1972). "Theory on the speed of convergence in adaptive equalizers for digital communication," *IBM J. Res. Dev,* vol. 16, pp. 546–555.

Ungerboeck, G. (1976). "Fractional tap-spacing equalizer and consequences for clock recovery in data modems," *IEEE Trans. Commun*., vol. COM-24, pp. 856–864.

Ulrych, T. J., and M. Ooe (1983). "Autoregressive and mixed autoregressive-moving average models and spectra," in S. Haykin, ed., *Nonlinear Methods of Spectral Analysis*, Springer-Verlag, New York, pp. 73–125.

Vaidyanathan, P. P. (1993). *Multirate Systems and Filter Banks,* Prentice-Hall, Englewood Cliffs, NJ.

Van den Bos, A. (1971). "Alternative interpretation of maximum entropy spectral analysis," *IEEE Trans. Information Theory,* vol. IT-17, pp. 493–194.

Van Loan, C. (1989). "Matrix computations in signal processing," in S. Haykin, ed., *Selected Topics in Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ.

Van Trees, H. L. (1968). *Detection, Estimation and Modulation Theory,* Part I, Wiley, New York.

Van Veen, B. (1992). "Minimum variance beamforming," in S. Haykin and A. Steinhardt, eds., *Adaptive Radar Detection and Estimation,* Wiley-Interscience, New York.

Van Veen, B. D., and K. M. Buckley (1988). "Beamforming: A versatile approach to spatial filtering," *IEEE ASSP Mag*., vol. 5, pp. 4–24.

Vapnik, V. (1998). *Statistical Learning Theory,* Wiley-Interscience, New York.

Verdú, S. (1984). "On the selection of memoryless adaptive laws for blind equalization in binary communications," in *Proc. 6th Intern. Conference on Analysis and Optimization of Systems,* Nice, France, pp. 239–249.

Verhaegen, M. H. (1989). "Round-off error propagation in four generally applicable, recursive, least-squares estimation schemes," *Automatica,* vol. 25, pp. 437–444.

Wakita, H. (1973). "Direct estimation of the vocal tract shape by inverse filtering of acoustic speech waveforms," *IEEE Trans. Audio Electroacoust*., vol. AU-21, pp. 417–427.

Walker, G. (1931). "On periodicity in series of related terms," *Proc. Royal Soc*., vol. A131, pp. 518–532.

Walzman, T., and M. Schwartz (1973). "Automatic equalization using the discrete frequency domain," *IEEE Trans. Information Theory,* vol. IT-19, pp. 59–68.

Ward, C. R., P. H. Hargrave, and J. G. McWhirter (1986). "A novel algorithm and architecture for adaptice digital beamforming," *IEEE Trans. Antennas Propag*., vol. AP-34, pp. 338–346.

Wax, M. (1995). "Model based processing in sensor arrays," in S. Haykin, ed., *Advances in Spectrum Analysis and Array Processing,* vol. 3, Prentice-Hall, Englewood Cliffs, NJ, pp. 1–47.

Wedin, P. A. (1972). "Perturbation bounds in connection with singular value decomposition," *Nordisk Tidskr. Informationsbehandling (BIT),* vol. 12, pp. 99–111.

Weisberg, S. (1980). *Applied Linear Regression,* Wiley, New York.

Weiss, A., and D. Mitra (1979). "Digital adaptive filters: Conditions for convergence, rates of convergence, effects of noise and errors arising from the implementation," *IEEE Trans. Information Theory,* vol. IT-25, pp. 637–652.

Werner, J. J. (1983). *Control of Drift for Fractionally Spaced Equalizers,* U.S. Patent 438 4355.

Whittaker, E. T., and G. N. Watson (1965). *A Course of Modern Analysis,* Cambridge University Press, Cambridge, U.K.

Whittle, P. (1963). "On the fitting of multivariate autoregressions and the approximate canonical factorization of a spectral density matrix," *Biometrika,* vol. 50, pp. 129–134.

Whittle, P. (1990). *Risk-Sensitive Optimal Control,* Wiley, New York.

Widrow, B. (1970). "Adaptive filters," in R. E. Kalman and N. Declaris, eds., *Aspects of Network and System Theory,* Holt, Rinehart and Winston, New York.

Widrow, B., and M. E. Hoff, Jr. (1960). "Adaptive switching circuits," *IRE WESCON Conv. Rec*., Pt. 4, pp. 96–104.

Widrow, B., and M. Kamenetsky (2003). "On the efficiency of adaptive algorithms," in S. Haykin and B. Widrow, eds., *Least-Mean-Square Adaptive Filters,* Wiley, New York, pp. 1–34.

Widrow, B., and S. D. Stearns (1985). *Adaptive Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ.

Widrow, B., and E. Walach (1984). "On the statistical efficiency of the LMS algorithm with nonstationary inputs," *IEEE Trans. Information Theory,* vol. IT-30, Special Issue on Linear Adaptive Filtering, pp. 211–221.

Widrow, B., J. McCool, and M. Ball (1975a). "The complex LMS algorithm," *Proc. IEEE,* vol. 63, pp. 719–720.

Widrow, B., et al. (1967). "Adaptive antenna systems," *Proc. IEEE,* vol. 55, pp. 2143–2159.

Widrow, B., et al. (1975b). "Adaptive noise cancelling: Principles and applications," *Proc. IEEE,* vol. 63, pp. 1692–1716.

Widrow, B., et al. (1976). "Stationary and nonstationary learning characteristics of the LMS adaptive filter," *Proc. IEEE,* vol. 64, pp. 1151–1162.

Wiener, N. (1949). *Extrapolation, Interpolation, and Smoothing of Stationary Time Series, with Engineering Applications,* MIT Press, Cambridge, MA (Originally issued as a classified National Defense Research Report in February 1942).

Wiener, N., and E. Hopf (1931). "On a class of singular integral equations," *Proc. Prussian Acad. Math-Phys. Ser,* p. 696.

Wilkinson, J. H. (1963). *Rounding Errors in Algebraic Processes,* Prentice-Hall, Englewood Cliffs, NJ.

Wilkinson, J. H., and C. Reinsch, eds. (1971). *Handbook for Automatic Computation, vol. 2, Linear Algebra,* Springer-Verlag, New York.

Wilks, S. S. (1962). *Mathematical Statistics,* Wiley, New York.

Wirtinger, W. (1927). "Zur Formalen Theorie der Funktionen von Mehr Komplexen Veräderlichen," *Math. Ann.,* vol. 97, pp. 357–375.

Wold, H. (1938). *A Study in the Analysis of Stationary Time Series,* Almqvist and Wiksell, Uppsala, Sweden.

Wolpert, D. H., and W. G. Macreedy (1997). "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput*., vol. 1, pp. 67–82.

Woodbury, M. (1950). *Inverting Modified Matrices,* Mem. Rep. 42, Statistical Research Group, Princeton University, Princeton, NJ.

Wozencraft, J. M., and I. M. Jacobs (1965). *Principles of Communications Engineering,* Wiley, New York.

Wylie, C. R., and L. C. Barrett (1982). *Advanced Engineering Mathematics,* 5th ed., McGraw-Hill, New York.

Yamamoto, S., and S. Kitayama (1982). "An adaptive echo canceller with variable step gain method," *Trans. IECE Japan,* vol. E65, pp. 1–8.

Yang, B. (1994). "A note on the error propagation analysis of recursive least squares algorithms," *IEEE Trans. Signal Process*., vol. 42, pp. 3523–3525.

Yang, V., and J. F. Böhme (1992). "Rotation-based RLS algorithms: Unified derivations, numerical properties and parallel implementations," *IEEE Trans. Signal Process*., vol. 40, pp. 1151–1167.

Yasukawa, H., and S. Shimada (1987). "Acoustic echo canceler with high speed quality," *Proc. IEEE ICASSP,* Dallas, TX, pp. 2125–2128.

Yule, G. U. (1927). "On a method of investigating periodicities in disturbed series, with special reference to Wöfer's sunspot numbers," *Philos. Trans. Royal Soc. London,* vol. A-226, pp. 267–298.

Zames, G. (1979). "On the metric complexity of causal linear systems: ε-entropy and ε-dimension for continuous time," *IEEE Trans. Autom. Control*, vol. AC-24, pp. 222–230.

Zames, G. (1981). "Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses," *IEEE Trans. Autom. Control,* vol. AC-26, pp. 301–320.

Zames, G. (1996). "Input-output feedback stability and robustness, 1959–85," *IEEE Control Systems,* vol. 16, pp. 61–66.

Zames, G., and B. A. Francis (1983). "Feedback, minimax sensitivity, and optimal robustness," *IEEE Trans. Autom. Control,* vol. AC-28, pp. 585–601.

Zeidler, J. R. (1990). "Performance analysis of LMS adaptive prediction filters," *Proc. IEEE,* vol. 78, pp. 1781–1806.

Zeidler, J. R., E. H. Satorius, D. M. Chabries, and H. T. Wexler (1978). "Adaptive enhancement of multiple sinusoids in uncorrelated noise," *IEEE Trans. Acous. Speech Signal Process*., vol. ASSP-26, pp. 240–254.

Suggested Readings

Abraham, J. A., et al. (1987). "Fault tolerance techniques for systolic arrays," *Computer,* vol. 20, pp. 65–75.

Alexander, S. T. (1986). "Fast adaptive filters: A geometrical approach," *IEEE ASSP Mag*., pp. 18–28.

Amari, S., A. Cichocki, and H. H. Yang (2000). "Blind signal separation and extraction: Neural and information approaches," in S. Haykin, ed., *Unsupervised Adaptive Filtering, Vol. I: Blind Source Separation,* Wiley, New York, pp. 63–138.

Anderson, B. D. O., and J. B. Moore (1979). *Linear Optimal Control,* Prentice-Hall, Englewood Cliffs, NJ.

Anderson, T. W. (1963). "Asymptotic theory for principal component analysis," *Ann. Math. Stat.,* vol. 34, pp. 122–148.

Andrews, H. C., and C. L. Patterson (1975). "Singular value decomposition and digital image processing," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-24, pp. 26–53.

Applebaum, S. P., and D. J. Chapman (1976). "Adaptive arrays with main beam constraints," *IEEE Trans. Antennas Propag*., vol. AP-24, pp. 650–662.

Ardalan, S. H. (1986). "Floating-point error analysis of recursive least-squares and least-mean-squares adaptive filters," *IEEE Trans. Circuits Syst*., vol. CAS-33, pp. 1192–1208.

Åström, K. J., and P. Eykhoff (1971). "System identification—A survey," *Automatica,* vol. 7, pp. 123–162.

Autonne, L. (1902). "Sur les groupes linéaires, réels et orthogonaux," *Bull. Soc. Math., France,* vol. 30, pp. 121–133.

Barron, A. R. (1993). "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Trans. Information Theory,* vol. 39, pp. 930–945.

Battiti, R. (1992). "First- and second-order methods for learning: Between steepest descent and Newton's method," *Neural Computation,* vol. 4, pp. 141–166.

Beaufays, F. (1995). *Two-layer linear structures for fast adaptive filtering,* Ph.D. dissertation, Stanford University, Stanford, CA.

Belfiore, C. A., and J. H. Park, Jr. (1979). "Decision feedback equalization," *Proc. IEEE,* vol. 67, pp. 1143–1156.

Bell, A. J. (2000). "Information theory, independent-component analysis and applications," in S. Haykin, ed., *Unsupervised Adaptive Filtering, Vol. I: Blind Source Separation,* Wiley, New York, pp. 237–264.

Bell, A. J., and T. J. Sejnowski (1995). "An information maximization approach to blind separation and blind deconvolution," *Neural Computation,* vol. 7, pp. 1129–1159.

Bellini, S., and F. Rocca (1986). "Blind deconvolution: Polyspectra or Bussgang techniques?" in E. Biglieri and G. Prati, eds., *Digital Communications,* North-Holland, Amsterdam, pp. 251–263.

Bellman, R. (1960). *Introduction to Matrix Analysis,* McGraw-Hill, New York.

Benesty, J., and T. Gänsler (2001). "A robust fast recursive least squares adaptive algorithm," in *Proc. IEEE ICASSP,* Salt Lake City, UT.

Benveniste, A. (1987). "Design of adaptive algorithms for the tracking of time-varying systems," *Int. J. Adaptive Control Signal Proc.,* vol. 1, pp. 3–29.

Benveniste, A., and M. Goursat (1984). "Blind equalizers," *IEEE Trans. Commun*., vol. COM-32, pp. 871–883.

Benveniste, A., and G. Ruget (1982). "A measure of the tracking capability of recursive stochastic algorithms with constant gains," *IEEE Trans. Autom. Control,* vol. AC-27, pp. 639–649.

Benvenuto, N., et al. (1986). "The 32 kb/s ADPCM coding standard," *AT&TJ*., vol. 65, pp. 12–22.

Benvenuto, N., and F. Piazza (1992). "On the complex backpropagation algorithm," *IEEE Trans. Signal Process*., vol. 40, pp. 967–969.

Bergmans, J. W. M. (1990). "Tracking capabilities of the LMS adaptive filter in the presence of gain variations," *IEEE Trans. Acoust. Speech Signal Process*., vol. 38, pp. 712–714.

Berkhout, A. J., and P. R. Zaanen (1976). "A comparison between Wiener filtering, Kalman filtering, and deterministic least squares estimation," *Geophysical Prospect*., vol. 24, pp. 141–197.

Bershad, N. J. (1986). "Analysis of the normalized LMS algorithm with Gaussian inputs," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-34, pp. 793–806.

Bershad, N. J., and P. L. Feintuch (1986). "A normalized frequency domain LMS adaptive algorithm," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-34, pp. 452–461.

Bershad, N. J., and O. Macchi (1991). "Adaptive recovery of a chirped sinusoid in noise, Part 2: Performance of the LMS algorithm," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-39, pp. 595–602.

Bershad, N. J., and L. Z. Qu (1989). "On the probability density function of the LMS adaptive filter weights," *IEEE Trans. Acoust. Speech Signal Process.,* vol. ASSP-37, pp. 43–56.

Birkett, A. N., and R. A. Goubran (1995). "Acoustic echo cancellation using NLMS-neural network structures," in *Proc. ICASSP,* vol. 5, Detroit, pp. 3035–3038.

Bitmead, P. R., and B. D. O. Anderson (1981). "Adaptive frequency sampling filters," *IEEE Trans. Circuits Syst*., vol. CAS-28, pp. 524–535.

Bitmead, P. R. (1983). "Convergence in distribution of LMS-type adaptive parameter studies," *IEEE Trans. Autom. Control,* vol. AC-28, pp. 54–60.

Bitmead, P. R. (1984). "Convergence properties of LMS adaptive estimators with unbounded dependent inputs," *IEEE Trans. Autom. Control,* vol. AC-29, pp. 477–479.

Björck, A. (1967). "Solving linear least squares problems by Gram–Schmidt orthogonalization," BIT, vol. 7, pp. 1–21.

Bode, H. W., and C. E. Shannon (1950). "A simplified derivation of linear least square smoothing and prediction theory," *Proc. IRE,* vol. 38, pp. 417–425.

Bojanczyk, A. W., and F. T. Luk (1990). "A unified systolic array for adaptive beamforming," *J. Parallel Distrib. Comput.,* vol. 8, pp. 388–392.

Boray, G. K., and M. D. Srinath (1992). "Conjugate gradient techniques for adaptive filtering," *IEEE Trans. Circuits Syst. Fundam. Theory Appl*., vol. 39, pp. 1–10.

Botto, J. L., and G. V. Moustakides (1989). "Stabilizing the fast Kalman algorithms," *IEEE Trans. Acoust., Speech Signal Process*., vol. ASSP-37, pp. 1342–1348.

Bracewell, R. N. (1986). *The Fourier Transform and Its Applications,* McGraw-Hill, New York.

Brent, R. P., F. T. Luk, and C. Van Loan (1983). "Decomposition of the singular value decomposition using mesh-connected processors," *J. VLSI Comput. Syst.,* vol. 1, pp. 242–270.

Brillinger, D. R., and M. Rosenblatt (1967). "Computation and interpretation of *k*-th order spectra," in B. Harris, ed., *Spectral Analysis of Time Series,* Wiley, New York.

Brogan, W. L. (1985). *Modern Control Theory,* 2nd ed., Prentice-Hall, Englewood Cliffs, NJ.

Broome, P. W. (1965). "Discrete orthonormal sequences," *J. Assoc. Comput. Machinary,* vol. 12, pp. 151–168.

Broomhead, D. S., and D. Lowe (1988). "Multi-variable functional interpolation and adaptive networks," *Complex Syst.,* vol. 2, pp. 269–303.

Brossier, J. M. (1992). *Egalisation Adaptive et Estimation de Phase: Application aux Communications Sous-marines,* Thèse de Docteur, del'Institut National Polytechnique de Grenoble, France.

Bruckstein, A., and T. Kailath (1987). "An inverse scattering framework for several problems in signal processing," *IEEE ASSP Mag*., vol. 4, pp. 6–20.

Bucklew, J. A., T. Kurtz, and W. A. Sethares (1993). "Weak convergence and local stability properties of fixed stepsize recursive algorithms," *IEEE Trans. Information Theory,* vol. 39, pp. 966–978.

Buckley, K. M., and L. J. Griffiths (1986). "An adaptive generalized sidelobe canceller with derivative constraints," *IEEE Trans. Antennas Propag*., vol. AP-34, pp. 311–319.

Bucy, R. S. (1994). *Lectures on Discrete Time Filtering,* Springer-Verlag, New York.

Burg, J. P. (1972). "The relationship between maximum entropy spectra and maximum likelihood spectra," *Geophysics,* vol. 37, pp. 375–376.

Cardoso, J.-F. (2000). "Entropic contrasts for source separation: Geometry and stability," in S. Haykin, ed., *Unsupervised Adaptive Filtering, Vol. I: Blind Source Separation,* Wiley, New York, pp. 139–189.

Carini, A., V. J. Mathews, and G. L. Sicuranza (1999). "Sufficient stability bounds for slowly varying direct-form recursive linear filters and their applications in adaptive IIR filters," *IEEE Trans. Signal Process.,* vol. 47, pp. 2561–2567.

Chazan, D., Y. Medan, and U. Shvadron (1988). "Noise cancellation for hearing aids," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-36, pp. 1697–1705.

Chen, S. (1995). "Nonlinear time series modelling and prediction using Gaussian RBF networks with enhanced clustering and RLS learning," *Electronics Letters,* vol. 31, no. 2, pp. 117–118.

Chen, S., S. McLaughlin, and B. Mulgrew (1994). "Complex-valued radial basis function network, Part I: Network architecture and learning algorithms," *Signal Proc*., vol. 35, pp. 19–31.

Chen, S., S. Mclaughlin, and B. Mulgrew (1994). "Complex-valued radial basis function network, Part II: Application to digital communications channel equalisation," *Signal Proc*., vol. 36, pp. 175–188.

Chester, D. L. (1990). "Why two hidden layers are better than one," *International Joint Conference on Neural Networks,* Washington, DC., vol. 1, pp. 265–268.

Chinrungrueng, C., and C. H. Sèquin (1995). "Optimal adaptive *k*-means algorithm with dynamic adjustment of learning rate," *IEEE Trans. Neural Networks,* vol. 6, pp. 157–169.

Chui, C. K., and G. Chen (1987). *Kalman Filtering with Real-time Application,* Springer-Verlag, New York.

Claesson, I., S. Nordholm, and P. Eriksson (1991). "Noise cancelling convergence rates for the LMS algorithm," *Mechanical Systems and Signal Processing,* vol. 5, pp. 375–388.

Clarke, T. L. (1990). "Generalization of neural networks to the complex plane," in *International Joint Conference on Neural Networks,* vol. II, San Diego, pp. 435–440.

Clarkson, P. M. (1993). *Optimal and Adaptive Signal Processing,* CRC Press, Boca Raton, FL.

Clarkson, P. M., and P. R. White (1987). "Simplified analysis of the LMS adaptive filter using a transfer function approximation," *IEEE Trans. Acoust. Speech Signal Process.,* vol. ASSP-35, pp. 987–993.

Classen, T. A. C. M., and W. F. G. Mecklanbräuker (1985). "Adaptive techniques for signal processing in communications," *IEEE Commun*., vol. 23, pp. 8–19.

Comon, P. (1994). "Independent component analysis: A new concept?" *Signal Processing,* vol. 36, pp. 287–314.

Comon, P., and P. Chevalier (2000). "Blind source separation: Models, concepts, algorithms and performance," in S. Haykin, ed., *Unsupervised Adaptive Filtering, Vol. I: Blind Source Separation,* Wiley, New York, pp. 191–236.

Compton, R. T. (1988). *Adaptive Antennas: Concepts and Performance,* Prentice-Hall, Englewood Cliffs, NJ.

Cowan, J. D. (1990). "Neural networks: The early days," in D. S. Touretzky, ed., *Advances in Neural Information Processing Systems 2,* Morgan Kaufman, San Mateo, CA, pp. 828–842.

Cox, H., R. M. Zeskind, and M. M. Owen (1987). "Robust adaptive beamforming," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-35, pp. 1365–1376.

Crochiere, R. E., and L. R. Rabiner (1983). *Multirate Digital Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ.

Cybenko, G. (1989). "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems,* vol. 2, pp. 303–314.

Davidson, G. W., and D. D. Falconer (1991). "Reduced complexity echo cancellation using orthonormal functions," *IEEE Trans. Circuits Syst.,* vol. 38, pp. 20–28.

De, P., and H. H. Fan (1999). "A delta least squares lattice algorithm for fast sampling," *IEEE Trans. Signal Process.,* vol. 47, pp. 2396–2406.

Delmas, J.-P, H. Gazzah, A. P. Liavas, and P. A. Regalia (2000). "Statistical analysis of some second-order methods for blind channel identification/equalization with respect to channel undermodeling," *IEEE Trans. Signal Process.,* vol. 48, pp. 1984–1998.

Demmel, J. (1994). *Designing High Performance Linear Algebra Software for Parallel Computers,* CS Division and Math Dept., UC Berkeley, CA.

Demmel, J., and K. Veselic´ (1989). *Jacobi's Method Is More Accurate Than QR,* Tech. Rep. 468, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, New York.

De Moor, B. L. R., and G. H. Golub (1989). *Generalized Singular Value Decompositions: A Proposal for a Standardized Nomenclature,* Manuscript NA-89-05, Numerical Analysis Project, Computer Science Department, Standard University, Stanford, CA.

Dentino, M., J. McCool, and B. Widrow (1978). "Adaptive filtering in the frequency domain," *Proc. IEEE,* vol. 66, no. 12, pp. 1658–1659.

Deprettere, E. F., ed. (1988). *SVD and Signal Processing: Algorithms, Applications, and Architectures,* North-Holland, Amsterdam.

Devijver, P. A., and J. Kittler (1982). *Pattern Recognition: A Statistical Approach,* Prentice-Hall International, London.

deVries, B., and J. C. Principe (1992). "The gamma model—A new neural model for temporal processing," *Neural Networks,* vol. 5, pp. 565–576.

deWaele, S., and P. M. T. Breersen (1998). "The Burg algorithm for segments," *IEEE Trans. Signal Process.,* vol. 48, pp. 2876–2880.

Dewilde, P., A. C. Vieira, and T. Kailath (1978). "On a generalized Szegö–Levinson realization algorithm for optimal linear predictors based on a network synthesis approach," *IEEE Trans. Circuits Syst.,* vol. CAS-25, pp. 663–675.

Dhrymus, P J. (1970). *Econometrics: Statistical Foundations and Applications,* Harper & Row, New York.

Ding, Z. (1994). "Blind channel identification and equalization using spectral correlation measurements, Part I: Frequency-domain approach," in W. A. Gardner, ed., *Cyclostationarity in Communications and Signal Processing,*" IEEE Press, New York, pp. 417–436.

Ding, Z., and Z. Mao (1995). "Knowledge based identification of fractionally sampled channels," in *Proc. ICASSP,* vol. 3, Detroit, pp. 1996–1999.

Ding, Z., C. R. Johnson, Jr., and R. A. Kennedy (1994). "Global convergence issues with linear blind adaptive equalizers," in S. Haykin, ed., *Blind Deconvolution,* Prentice-Hall, Englewood Cliffs, NJ.

Diniz, P. S. R. (1997). *Adaptive Filtering: Algorithms and Practical Implementation,* Kluwer Academic Publishers, Boston.

Diniz, P. S. R., and L. W. P. Biscainho (1992). "Optimal variable step size for the LMS/Newton algorithm with application to subband adaptive filtering," *IEEE Trans. Signal Process.,* vol. 40, pp. 2825–2829.

Dongarra, J. J., et al. (1979). *LINPACK User's Guide,* Society for Industrial and Applied Mathematics, Philadelphia.

Donoho, D. L. (1981). "On minimum entropy deconvolution," in D. F. Findlay, ed., *Applied Time Series Analysis II,* Academic Press, New York.

Dorny, C. N. (1975).

Douglas, S. C. (1995). "Generalized gradient adaptive step sizes for stochastic gradient adaptive filters," *Proc. of ICASP,* pp. 1396–1399.

Douglas, S. C., and S. Amari (2000). "Natural gradient adaptation," in S. Haykin, ed., *Unsupervised Adaptive Filtering, Vol. I: Blind Source Separation,* Wiley, New York, pp. 13–62.

Douglas, S. C., and S. Haykin (2000). "Relationships between blind deconvolution and blind source separation," in S. Haykin, ed., *Unsupervised Adaptive Filtering, Vol. II: Blind Deconvolution,* Wiley, New York, pp. 113–146.

Douglas, S. C., and W. Pan (1995). "Exact expectation analysis of the LMS adaptive filter," *Trans. IEEE Signal Process.,* vol. 43, pp. 2863–2871.

Douglas, S. C., Q. Zhu, and K. F. Smith (1998). "A pipelined LMS adaptive FIR filter architecture without adaptation delay," *IEEE Trans. Signal Process.,* vol. 46, pp. 775–779.

Doyle, J. C., K. Glover, P. Khargonekar, and B. Francis (1989). "State-space solutions to standard H_2 and H_∞ control problems," *IEEE Trans. Autom. Control,* vol. AC-34, pp. 831–847.

Duda, R. O., and P. E. Hart (1973). *Pattern Classification and Scene Analysis,* Wiley, New York.

Dugard, L., M. M'Saad, and I. D. Landau (1993). *Adaptive Systems in Control and Signal Processing,* Pergamon Press, Oxford, United Kingdom.

Duttweiler, D. L. (2000). "Speech enhancement—Proportionate normalized least-mean-squares adaptation in echo cancelers," *IEEE Trans. Speech Audio Process.,* vol. 8, pp. 508–518.

Duttweiler, D. L., and Y. S. Chen (1980). "A single-chip VLSI echo canceler," *Bell Syst. Tech. J.,* vol. 59, pp. 149–160.

Eckart, G., and G. Young (1939). "A principal axis transformation for non-Hermitian matrices," *Bull. Am. Math. Soc.,* vol. 45, pp. 118–121.

Edwards, A. W. F. (1972). *Likelihood,* Cambridge University Press, New York.

e Silva, T. O. (1995). "On the determination of the optimal pole position of Laguerre filters," *IEEE Trans. Signal Process.,* vol. 43, pp. 2079–2087.

Evans, J. B., P. Xue, and B. Liu (1993). "Analysis and implementation of variable step-size adaptive algorithms," *IEEE Trans. Signal Process.,* vol. 41, pp. 2517–2534.

Eweda, E. (1994). "Comparison of RLS, LMS, and sign algorithms for tracking randomly time-varying channels," *IEEE Trans. Signal Process*., vol. 42, pp. 2937–2944.

Eweda, E., and O. Macchi (1985). "Tracking error bounds of adaptive nonstationary filtering," *Automatica,* vol. 21, pp. 293–302.

Eweda, E., and O. Macchi (1987). "Convergence of the RLS and LMS adaptive filters," *IEEE Trans. Circuits Syst.,* vol. CAS-34, pp. 799–803.

Farden, D. C. (1981). "Stochastic approximation with correlated data," *IEEE Trans. Information Theory,* vol. IT-27, pp. 105–113.

Farden, D. C. (1981). "Tracking properties of adaptive signal processing algorithms," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-29, pp. 439–446.

Fejzo, Z., and H. Lev-Ari (1997). "Adaptive Laguerre-lattice filters," *IEEE Trans. Signal Process.,* vol. 45, pp. 3006–3016.

Feldkamp, L. A., G. V. Puskorius, L. I. Davis, Jr., and F. Yuan (1994). "Enabling concepts for application of neurocontrol," in *Proc. Eighth Yale Workshop on Adaptive and Learning Systems,* Yale University, New Haven, CT, pp. 168–173.

Feldkamp, L. A., T. M. Feldkamp, and D. V. Prokhorov (2001). "Neural network training with npr KF," in *International Joint Conference on Neural Networks,* Washington, DC.

Feldman, D. D., and L. J. Griffiths (1984). "A projection approach for robust adaptive beamforming," *IEEE Trans. Signal Process.,* vol. 42, pp. 867–876.

Feng, D.-Z., Z. Bao, and L.-C. Jiao (1998). "Total least mean square algorithm," *IEEE Trans. Signal Process.,* vol. 46, pp. 2122–2130.

Fernando, K. V., and B. N. Parlett (1994). "Accurate singular values and differential qd algorithms," *Numer. Math.,* vol. 67, pp. 191–229.

Fijalkow, I., J. R. Treichler, and C. R. Johnson, Jr. (1995). "Fractionally spaced blind equalization: Loss of channel disparity," in *Proc. ICASSP,* Detroit, pp. 1988–1991.

Fisher, B., and N. J. Bershad (1983). "The complex LMS adaptive algorithm—Transient weight mean and covariance with applications to the ALE," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-31, pp. 34–44.

Foley, J. B., and F. M. Boland (1987). "Comparison between steepest descent and LMS algorithms in adaptive filters," *IEE Proc. (London), Part F,* vol. 134, pp. 283–289.

Foley, J. B., and F. M. Boland (1988). "A note on the convergence analysis of LMS adaptive filters with Gaussian data," *IEEE Trans. Acoust. Speech Signal Process*., vol. 36, pp. 1087–1089.

Forney, G. D. (1972). "Maximum-likelihood sequence estimation of digital sequence in the presence of intersymbol interference," *IEEE Trans. Information Theory,* vol. IT-18, pp. 363–378.

Foschini, G. J. (1985). "Equalizing without altering or detecting data," *AT&T Tech. J*., Vol. 64, pp. 1885–1911.

Franks, L. E., ed. (1974). *Data Communication: Fundamentals of Baseband Transmission,* Benchmark Papers in Electrical Engineering and Computer Science, Dowden, Hutchinson & Ross, Stroudsburg, PA.

Friedlander, B. (1982). "Lattice filters for adaptive processing," *Proc. IEEE,* vol. 70, pp. 829–867.

Friedlander, B. (1988). "A signal subspace method for adaptive interference cancellation," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-36, pp. 1835–1845.

Friedlander, B., and B. Porat (1989). "Adaptive IIR algorithm based on high-order statistics," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-37, pp. 485–495.

Friedrichs, B. (1992). "Analysis of finite-precision adaptive filters. I. Computation of the residual signal variance," *Frequenz,* vol. 46, pp. 218–223.

Fukunaga, K. (1990). *Statistical Pattern Recognition,* 2nd ed., Academic Press, New York.

Funahashi, K. (1989). "On the approximate realization of continuous mappings by neural networks," *Neural Networks,* vol. 2, pp. 183–192.

Gabor, D., W. P. L. Wilby, and R. Woodcock (1961). "A universal non-linear filter, predictor and simulator which optimizes itself by a learning process," *IEE Proc. (London),* vol. 108, pt. B, pp. 422–438.

Gallivan, K. A., and C. E. Leiserson (1984). "High-performance architectures for adaptive filtering based on the Gram–Schmidt algorithm," in *Proc. SPIE,* vol. 495, *Real Time Signal Processing* VII, pp. 30–38.

Garbow, B. S., et al. (1977). *Matrix Eigensystem Routines—EISPACK Guide Extension,* Lecture Notes in Computer Science, vol. 51, Springer-Verlag, New York.

Gardner, W. A. (1993). "Cyclic Wiener filtering: Theory and method," *IEEE Trans. Signal Process*., vol. 41, pp. 151–163.

Gardner, W. A., and C. M. Spooner (1994). "The cumulant theory of cyclostationary time-series, Part I: Foundation," *IEEE Trans. Signal Process*., vol. 42, pp. 3387–3408.

Gay, S. L. (1998). "An efficient, fast converging adaptive filter for network echo cancellation," in *Proceedings of the Asilomar Conference on Signals, Systems, and Computers,* Pacific Grove, California.

Gentleman, W. M. (1973). "Least squares computations by Givens transformations without square-roots," *J. Inst. Math. Its Appl*., vol. 12, pp. 329–336.

Georgcou, G. N., and C. Koutsougeras (1992). "Complex domain backpropagation," *IEEE Trans. Circuits Syst. Analog Digital Signal Process.,* vol. 39, pp. 330–334.

Gersho, A. (1969). "Adaptive equalization of highly dispersive channels for data transmission," *Bell Syst. Tech. J*., vol. 48, pp. 55–70.

Gersho, A., B. Gopinath, and A. M. Oldyzko (1979). "Coefficient inaccuracy in transversal filtering," *Bell Syst. Tech. J*., vol. 58, pp. 2301–2316.

Ghogho, M., M. Ibnkahla, and N. J. Bershad (1998). "Analytic behavior of the LMS adaptive line enhancer for sinusoids computed by multiplicative and additive noise," *IEEE Trans. Signal Process.,* vol. 46, pp. 2386–2393.

Gholkar, V. A. (1990). "Mean square convergence analysis of LMS algorithm (adaptive filters)," *Electronics Letters,* vol. 26, pp. 1705–1706.

Giannakis, G. B., and S. D. Halford (1995). "Blind fractionally-spaced equalization of noisy FIR channels: Adaptive and optimal solutions," in *Proc. IEEE ICASSP,* Detroit, pp. 1972–1975.

Gibson, G. J., and C. F. N. Cowan (1990). "On the decision regions of multilayer perceptrons," *Proc. IEEE,* vol. 78, pp. 1590–1599.

Gibson, J. D. (1980). "Adaptive prediction in speech differential encoding systems," *Proc. IEEE,* vol. 68, pp. 488–525.

Gill, P. E., G. H. Golub, W. Murray, and M. A. Saunders (1974). "Methods of modifying matrix factorizations," *Math. Comput.*, vol. 28, pp. 505–535.

Gilloire, A., and M. Vetterli (1988). "Adaptive filtering in subbands," in *Proc. IEEE ICASSP '88,* New York, NY, pp. 1572–1575.

Gilloire, A., E. Moulines, D. Slock, and P. Duhamel (1996). "State of the art in acoustic echo cancellation," in A. R. Figueras-Vidal, ed., *Digital Signal Processing in Telecommunications,* Springer, Berlin, pp. 45–91.

Gitlin, R. D., and F. R. Magee, Jr. (1977). "Self-orthogonalizing adaptive equalization algorithms," *IEEE Trans. Commun.,* vol. COM-25, pp. 666–672.

Gitlin, R. D., and S. B. Weinstein (1979). "On the required tap-weight precision for digitally implemented mean-squared equalizers," *Bell Syst. Tech. J.,* vol. 58, pp. 301–321.

Givens, W. (1958). "Computation of plane unitary rotations transforming a general matrix to triangular form," *J. Soc. Ind. Appl. Math,* vol. 6, pp. 26–50.

Glaser, E. M. (1961). "Signal detection by adaptive filters," *IRE Trans. Information Theory,* vol. IT-7, pp. 87–98.

Godara, L. C., and A. Cantoni (1986). "Analysis of constrained LMS algorithm with application to adaptive beamforming using perturbation sequences," *IEEE Trans. Antennas Propag.*, vol. AP-34, pp. 368–379.

Golomb, S. W., ed. (1964). *Digital Communications with Space Applications,* Prentice-Hall, Englewood Cliffs, NJ.

Golub, G. H. (1965). "Numerical methods for solving linear least squares problems," *Numer. Math.*, vol. 7, pp. 206–216.

Golub, G. H., and C. Reinsch (1970). "Singular value decomposition and least squares problems," *Numer. Math.*, vol. 14, pp. 403–420.

Golub, G. H., F. T. Luk, and M. L. Overton (1981). "A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix," *ACM Trans. Math. Software,* vol. 7, pp. 149–169.

Goodwin, G. C., and K. S. Sin (1984). *Adaptive Filtering, Prediction and Control,* Prentice-Hall, Englewood Cliffs, NJ.

Gray, R. M. (1977). *Toeplitz and Circulant Matrices:* II, Tech. Rep. 6504-1, Information Systems Laboratory, Stanford University, Stanford, CA.

Green, M., and D. J. N. Limebber (1995). *Linear Robust Control,* Prentice-Hall, Englewood Cliffs, NJ.

Griffiths, L. J., and R. Prieto-diaz (1977). "Spectral analysis of natural seismic events using autoregressive techniques," *IEEE Trans. Geosci. Electron.*, vol. GE-15, pp. 13–25.

Gu, M., and S. C. Eisenstat (1994). *A Divide-and-Conquer Algorithm for the Bidiagonal SVD,* Research Report YALEU/DCS/RR-933, UC Berkeley, CA.

Gu, M., J. Demmel, and I. Dhillon (1994). *Efficient Computation of the Singular Value Decomposition with Applications to Least Squares Problems,* Department of Mathematics, UC Berkeley, CA.

Guillemin, E. A. (1949). *The Mathematics of Circuit Analysis,* Wiley, New York.

Gunnarsson, S., and L. Ljung (1989). "Frequency domain tracking characteristics of adaptive algorithms," *IEEE Trans. Signal Process.*, vol. 37, pp. 1072–1089.

Guo, L., L. Ljung, and G.-J. Wang (1997). "Necessary and sufficient conditions for stability of LMS," *IEEE Trans. Autom. Control,* vol. 42, pp. 761–770.

Gupta, I. J., and A. A. Ksienski (1986). "Adaptive antenna arrays for weak interfering signals," *IEEE Trans. Antennas Propag.*, vol. AP-34, pp. 420–426.

Gutowski, P. R., E. A. Robinson, and S. Treitel (1978). "Spectral estimation: Fact or fiction," *IEEE Trans. Geosci. Electron.*, vol. GE-16, pp. 80–84.

Hadhoud, M. M., and D. W. Thomas (1988). "The two-dimensional adaptive LMS (TDLMS) algorithm," *IEEE Trans. Circuits Syst.*, vol. CAS-35, pp. 485–494.

Hampel, F. R., et al. (1986). *Robust Statistics: The Approach Based on Influence Functions,* Wiley, New York.

Hänsler, E. (1992). "The hands-free telephone problem—An annotated bibliography," *Signal Process.,* vol. 27, pp. 259–271.

Hartenek, M., R. W. Stewart, J. G. McWhirter, and I. K. Proudler (1998). "Using algorithmic engineering to derive a fast parallel weight extraction algorithm based on QR recursive least squares," in J. G. McWhirter and I. K. Proudler, eds., *Mathematics in Signal Processing IV,* Oxford University Press, New York, pp. 127–137.

Hartman, E. J., J. D. Keeler, and J. M. Kowalski (1990). "Layered neural networks with Gaussian hidden units as universal approximations," *Neural Computation,* vol. 2, pp. 210–215.

Hassibi, B., A. H. Sayed, and T. Kailath (1994). "H^∞ optimality criteria for LMS and backpropagation," in J. D. Cowan, et al., eds., *Advances in Neural Information Processing Systems,* vol. 6, Morgan–Kaufmann, New York, pp. 351–359.

Hassibi, B., A. H. Sayed, and T. Kailath (1999). *Infinite-Quadratic Estimation and Control: A Unified Approach to H^2 and H^∞ Theories,* SIAM, Philadelphia.

Hastings-James, R., and M. W. Sage (1969). "Recursive generalized-least-squares procedure for online identification of process parameters," *IEE Proc. (London),* vol. 116, pp. 2057–2062.

Hatzinakos, D., and C. L. Nikias (1994). "Blind equalization based on higher-order statistics (HOS)," in S. Haykin, ed., *Blind Deconvolution,* Prentice-Hall, Englewood Cliffs, NJ.

Haykin, S., ed. (1983). *Nonlinear Methods of Spectral Analysis,* 2nd ed., Springer-Verlag, New York.

Haykin, S., ed. (1984). *Array Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ.

Haykin, S. (1989). "Adaptive filters: Past, present, and future," in *Proc. IMA Conf. Math. Signal Process*., Warwick, U.K.

Haykin, S., and L. Li (1995). "Nonlinear adaptive prediction of nonstationary signals," *IEEE Trans. Signal Process.,* vol. 43, pp. 526–535.

Haykin, S., and A. Ukrainec (1993). "Neural networks for adaptive signal processing," in N. Kalouptsidis and S. Theodoridis, eds., *Adaptive System Identification and Signal Processing Algorithms,* Prentice-Hall, Englewood Cliffs, NJ, pp. 512–553.

Haykin, S., A. H. Sayed, J. R. Zeidler, P. Yee, and P. C. Wei (1997). "Adaptive tracking of linear time-variant systems by extended RLS algorithms," *IEEE Trans. Signal Process.,* vol. 45, pp. 1118–1128.

Henseler, J., and E. J. Braspenning (1990). *Training Complex Multi-Layer Neural Networks,* Tech. Rep. CS90-02, Department of Computer Science, University of Limburg, Maastricht, The Netherlands.

Hérault, J., C. Jutten, and B. Ans (1985). "Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimetique un apprentissage non supervisé," in *Procedures of GRETSI,* Nice, France.

Herzberg, H., and R. Haimi-Cohen (1992). "A systolic array realization of an LMS adaptive filter and the effects of delayed adaptation," *IEEE Trans. Signal Process*, vol. 40, pp. 2799–2803.

Hodgkiss, W. S., Jr., and D. Alexandrou (1983). "Applications of adaptive least-squares lattice structures to problems in underwater acoustics," in *Proc. SPIE,* vol. 431, *Real Time Signal Processing* VI, pp. 48–54.

Homer, J., P. R. Bitmead, and I. Mareels (1998). "Quantifying the effects of dimension on the convergence rate of the LMS adaptive FIR estimator," *IEEE Trans. Signal Process.,* vol. 46, pp. 2611–2615.

Homer, J., I. Marells, P. R. Bitmead, B. Wahlberg, and F. Gustafsson (1998). "LMS estimation via structural detection," *IEEE Trans. Signal Process.,* vol. 46, pp. 2651–2663.

Honig, M. L., and D. G. Messerschmitt (1981). "Convergence properties of an adaptive digital lattice filter," *IEEE Trans. Acous. Speech Signal Process*., vol. ASSP-29, pp. 642–653.

Honig, M. L., and D. G. Messerschmitt (1984). *Adaptive Filters: Structures, Algorithms and Applications,* Kluwer, Boston.

Hornik, K., M. Stinchcomse, and H. White (1989). "Multilayer feedforward networks are universal approximators," *Neural Networks,* vol. 2, pp. 359–366.

Horowitz, L. L., and K. D. Senne (1981). "Performance advantage of complex LMS for controlling narrow-band adaptive arrays," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-29, pp. 722–736.

Hsia, T. C. (1983). "Convergence analysis of LMS and NLMS adaptive algorithms," in *Proc. ICASSP,* Boston, pp. 667–670.

Hsu, F. M. (1982). "Square root Kalman filtering for high-speed data received over fading dispersive HF channels," *IEEE Trans. Information Theory,* vol. IT-28, pp. 753–763.

Hu, Y. H. (1992). "CORDIC-based VLSI architectures for digital signal processing," *IEEE Signal Process. Maga.,* vol. 9, pp. 16–35.

Huber, P. J. (1981). *Robust Statistics,* Wiley, New York.

Hubing, N. E., and S. T. Alexander (1990). "Statistical analysis of the soft constrained initialization of recursive least squares algorithms," in *Proc. IEEE ICASSP,* Albuquerque, NM.

Hudson, J. E. (1981). *Adaptive Array Principles,* Peregrinus, London.

Huhta, J. C., and J. G. Webster (1973). "60-Hz interference in electrocardiography," *IEEE Trans. Biomed. Eng*., vol. BME-20, pp. 91–101.

Hyvärinen, A., J. Karhunen, and E. Oja (2001). *Independent Component Analysis,* Wiley, New York.

Jablon, N. K. (1986). "Steady state analysis of the generalized sidelobe canceller by adaptive noise canceling techniques," *IEEE Trans. Antennas Propag*., vol. AP-34, pp. 330–337.

Jablon, N. K. (1991). "On the complexity of frequency-domain adaptive filtering," *IEEE Trans. Signal Process*., vol. 39, pp. 2331–2334.

Jayant, N. S. (1986). "Coding speech," *IEEE Spectrum,* vol. 23, pp. 58–63.

Jayant, N. S., and P. Noll (1984). *Digital Coding of Waveforms,* Prentice-Hall, Englewood Cliffs, NJ.

Jazwinski, A. H. (1969). "Adaptive filtering," *Automatica,* vol. 5, pp. 475–485.

Jazwinski, A. H. (1970). *Stochastic Processes and Filtering Theory,* Academic Press, New York.

Johnson, C. R., Jr. (1984). "Adaptive IIR filtering: Current results and open issues," *IEEE Trans. Information Theory,* vol. IT-30, Special Issue on Linear Adaptive Filtering, pp. 237–250.

Johnson, C. R., Jr. (1988). *Lectures on Adaptive Parameter Estimation,* Prentice-Hall, Englewood Cliffs, NJ.

Johnson, C. R., Jr., S. Dasgupta, and W. A. Sethares (1988). "Averaging analysis of local stability of a real constant modulus algorithm adaptive filter," *IEEE Trans. Acoust. Speech Signal Process.,* vol. ASSP-36, pp. 900–910.

Johnson, D. H., and P. S. Rao (1990). "On the existence of Gaussian noise," in *1990 Digital Signal Processing Workshop,* New Paltz, NY, pp. 8.14.1–8.14.2.

Jones, S. K., R. K. Cavin, III, and W. M. Reed (1982). "Analysis of error-gradient adaptive linear equalizers for a class of stationary-dependent processes," *IEEE Trans. Information Theory,* vol. IT-28, pp. 318–329.

Jou, J.-Y., and A. Abraham (1986). "Fault-tolerant matrix arithmetic and signal processing on highly concurrent computing structures," *Proc. IEEE,* vol. 74, Special Issue on Fault Tolerance in VLSI, pp. 732–741.

Julier, S. J., and J. K. Uhlmann (1997). "A new extension of the Kalman filter to nonlinear systems," in *Proceedings of AeroSense: The 11th Symposium on Aerospace/Defence Sensing, Simulation and Control,* Orlando, FL.

Julier, S. J., J. K. Uhlmann, and H. Durrant-Whyte (1995). "A new approach for filtering nonlinear systems," *Proceedings of the American Control Conference,* Seattle, pp. 1628–1632.

Justice, J. H. (1985). "Array processing in exploration seismology," in S. Haykin, ed., *Array Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ, pp. 6–114.

Kadlec, I., and F. M. F. Gaston (1995). "Identification with directional parameter tracking for high-performance fixed-point implementations," in *Proc. 6th Irish DSP, and Control Colloquium,* Queen's University, Belfast, Northern Ireland, pp. 215–222.

Kailath, T. (1969). "A generalized likelihood ratio formula for random signals in Gaussian noise," *IEEE Trans. Information Theory,* vol. IT-15, pp. 350–361.

Kailath, T. (1981). *Lectures on Linear Least-Squares Estimation,* Springer-Verlag, New York.

Kailath, T. (1982). "Time-variant and time-invariant lattice filters for nonstationary processes," in I. Laudau, ed., *Outils et Modèles Mathématique pour l' Automatique, L' Analyse de Systemes et le Traitement du Signal,* vol. 2, CNRS, Paris, pp. 417–464.

Kailath, T. (1991). "Remarks on the origin of the displacement-rank concept," *Appl. Math. Comput.*, vol. 45, pp. 193–206.

Kailath, T. (1999). "Displacement structure and array algorithms," in T. Kailath and A. H. Sayed, eds., *Fast Reliable Algorithms for Matrices with Structure,* SIAM, Philadelphia, pp. 1–56.

Kailath, T., and A. H. Sayed (1999). *Fast Reliable Algorithms for Matrices with Structure,* SIAM, Philadelphia.

Kailath, T., A. Vieira, and M. Morf (1978). "Inverses of Toeplitz operators, innovations, and orthogonal polynomials," *SIAM Rev,* vol. 20, pp. 106–119.

Kallmann, H. J. (1940). "Transversal filters," *Proc. IRE,* vol. 28, pp. 302–310.

Kalouptsidis, N., and S. Theodoridis (1987). "Parallel implementation of efficient LS algorithms for filtering and prediction," *IEEE Trans. Acoust. Speech Signal Process.*, vol. ASSP-35, pp. 1565–1569.

Kalouptsidis, N., and S. Theodoridis, eds. (1993). *Adaptive System Identification and Signal Processing Algorithms,* Prentice-Hall, Englewood Cliffs, NJ.

Kang, G. S., and L. J. Fransen (1987). "Experimentation with an adaptive noise-cancellation filter," *IEEE Trans. Circuits Syst.*, vol. CAS-34, pp. 753–758.

Kautz, W. H. (1954). "Transient synthesis in the time domain," *IRE Trans. Circuit Theory,* vol. CT-1, pp. 29–39.

Kay, S. M., and L. S. Marple, Jr. (1981). "Spectrum analysis—A modern perspective," *Proc. IEEE,* vol. 69, pp. 1380–1419.

Khargonekar, P. P., and K. M. Nagpal (1991). "Filtering and smoothing in an H^{∞}-setting," *IEEE Trans. Autom. Control,* vol. AC-36, pp. 151–166.

Kim, M. S., and C. C. Guest (1990). "Modification of backpropagation networks for complex-valued signal processing in frequency domain," *International Joint Conference on Neural Networks,* vol. III, San Diego, pp. 27–31.

Kimura, H. (1984). "Robust realizability of a class of transfer functions," *IEEE Trans. Autom. Control,* vol. AC-29, pp. 788–793.

Knight, W. C., R. G. Pridham, and S. M. Kay (1981). "Digital signal processing for sonar," *Proc. IEEE,* vol. 69, pp. 1451–1506.

Koh, T., and E. J. Powers (1985). "Second-order Volterra filtering and its application to nonlinear system identification," *IEEE Trans. Acoust. Speech Signal Process.*, vol. ASSP-33, pp. 1445–1455.

Kolmogorov, A. N. (1941). "Stationary sequences in Hilbert Space," *Bull. Math. Univ. Moscow,* vol. 2, No. 6 (in Russian). (A translation of this paper by N.Artin is available in some libraries.)

Kumar, R. (1983). "Convergence of a decision-directed adaptive equalizer," in *Proc. Conf. Decision Control,* vol. 3, pp. 1319–1324.

Kung, S. Y. (1988). *VLSI Array Processors,* Prentice-Hall, Englewood Cliffs, NJ.

Kung, S. Y., H. J. Whitehouse, and T. Kailath, eds. (1985). *VLSI and Modern Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ.

Kung, S. Y., et al. (1987). "Wavefront array processors—Concept to implementation," *Computer,* vol. 20, pp. 18–33.

Kushner, H. J., and J. Yang (1995). "Analysis of adaptive step size SA algorithms for parameter tracking," *IEEE Trans. Autom. Control,* vol. 40, pp. 1403–1410.

Kuzminsky, A. M. (1982). "Self adjustment of noise canceler adaptation gain in a nonstationary environment," *Izvestiya VUZ, Radioelektronika,* vol. 25, pp. 78–80 (in Russian).

Kuzminsky, A. M. (1997). "A robust step size adaptation scheme for LMS adaptive filters," *IEEE Workshop on Digital Signal Processing,* July, pp. 33–36.

Lambert, R. H., and C. L. Nikias (2000). "Blind deconvolution of multipath mixtures," in S. Haykin, ed., *Unsupervised Adaptive Filtering, Vol. I: Blind Source Separation,* Wiley, New York, pp. 377–436.

Landau, I. D. (1984). "A feedback system approach to adaptive filtering," *IEEE Trans. Information Theory,* vol. IT-30, Special Issue on Linear Adaptive Filtering, pp. 251–262.

Lang, S. W., and J. H. McClellan (1979). "A simple proof of stability for all-pole linear prediction models," *Proc. IEEE,* vol. 67, pp. 860–861.

Lawson, C. L., and R. J. Hanson (1974). *Solving Least Squares Problems,* Prentice-Hall, Englewood Cliffs, NJ.

Lawson, J. E., and G. E. Uhlenbeck (1965). *Threshold Signals,* Dover Publications, New York.

Lee, D. T. L. (1980). *Canonical ladder form realizations and fast estimation algorithms,* Ph.D. dissertation, Stanford University, Stanford, CA.

Lee, J. C., and C. K. Un (1986). "Performance of transform-domain LMS adaptive algorithms," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-34, pp. 499–510.

Lee, Y. W. (1960). *Statistical Theory of Communication,* Wiley, New York.

Leung, H., and S. Haykin (1991). "The complex backpropagation algorithm," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-39, pp. 2101–2104.

Lev-Ari, H., T. Kailath, and J. Cioffi (1984). "Least-squares adaptive lattice and transversal filters: A unified geometric theory," *IEEE Trans. Information Theory,* vol. IT-30, pp. 222–236.

Levinson, N., and R. M. Redheffer (1970). *Complex Variables,* Holden-Day, San Francisco.

Lewis, A. (1992). "Adaptive filtering-applications in telephony," *BT Technol. J*., vol. 10, pp. 49–63.

Lewis, F. L. (1986). *Optimal Estimation: With an Introduction to Stochastic Control Theory,* Wiley, New York.

Li, X., and H. Fan (2000). "QR factorization based blind channel identification and equalization with second-order statistics," *IEEE Trans. Signal Process.,* vol. 48, pp. 60–69.

Li, Y., and K. J. R. Liu (1996). "Static and dynamic convergence behavior of adaptive blind equalizers," *IEEE Trans. Signal Process.,* vol. 44, pp. 2736–2745.

Li, Y., K. J. R. Liu, and Z. Ding (1996). "Length and cost dependence of local minima of unconstrained blind channel equalizers," *IEEE Trans. Signal Process.,* vol. 44, pp. 2726–2735.

Liapunov, A. M. (1966). *Stability of Motion,* trans. F. Abramovici and M. Shimshoni, Academic Press, New York.

Lii, K. S. and M. Rosenblatt (1982). "Deconvolution and estimation of transfer function phase and coefficients for non-Gaussian linear processes," *Ann. Stat*., vol. 10, pp. 1195–1208.

Liles, W. C., J. W. Demmel, and L. E. Brennan (1980). *Gram–Schmidt Adaptive Algorithms,* Tech. Rep. RADC-TR-79-319, RADC, Griffiss Air Force Base, NY.

Lin, D. W. (1984). "On digital implementation of the fast Kalman algorithm," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-32, pp. 998–1005.

Ling, F. (1991). "Givens rotation based least-squares lattice and related algorithms," *IEEE Trans. Signal Process.,* vol. 39, pp. 1541–1551.

Ling, F., and J. G. Proakis (1984). "Nonstationary learning characteristics of least squares adaptive estimation algorithms," in *Proc. IEEE ICASSP,* San Diego, pp. 3.7.1–3.74.

Ling, F., and J. G. Proakis (1986). "A recursive modified Gram–Schmidt algorithm with applications to least squares estimation and adaptive filtering," *IEEE Trans. Acoust. Speech Signal Process.,* vol. ASSP-34, pp. 829–836.

Ling, F., D. Manolakis, and J. G. Proakis (1986). "Numerically robust least-squares lattice-ladder algorithm with direct updating of the reflection coefficients," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-34, pp. 837–845.

Lippmann, R. E. (1987). "An introduction to computing with neural nets," *IEEE ASSP Mag.,* vol. 4, pp. 4–22.

Little, G. R., S. C. Gustafson, and R. A. Senn (1990). "Generalization of the backpropagation neural network learning algorithm to permit complex weights," *Appl. Opt.,* vol. 29, pp. 1591–1592.

Liu, Z.-S. (1995). "QR methods of *O(N)* complexity in adaptive parameter estimation," *IEEE Trans. Signal Process*., vol. 43, pp. 720–729.

Ljung, L. (1977). "Analysis of recursive stochastic algorithms," *IEEE Trans. Autom. Control*., vol. AC-22, pp. 551–575.

Ljung, L. (1984). "Analysis of stochastic gradient algorithms for linear regression problems," *IEEE Trans. Information Theory,* vol. IT-30, Special Issue on Linear Adaptive Filtering, pp. 151–160.

Ljung, L. (1987). *System Identification: Theory for the User,* Prentice-Hall, Englewood Cliffs, NJ.

Ljung, L., and S. Gunnarsson (1990). "Adaptation and tracking in system identification—A survey," *Automatica,* vol. 26, pp. 7–21.

Ljung, L., M. Morf, and D. Falconer (1978). "Fast calculation of gain matrices for recursive estimation schemes," *Int. J. Control,* vol. 27, pp. 1–19.

Ljung, L., and T. Söderström (1983). *Theory and Practice of Recursive Identification,* MIT Press, Cambridge, MA.

López-Valcarce, R., and S. Dasgupta (1999). "A new proof for the stability of equation-error models," *IEEE Signal Process. Letters,* vol. 6, pp. 148–150.

López-Valcarce, R., S. Dasgupta, R. Tempo, and M. Fu (2000). "Exponential asymptotic stability of time-varying inverse prediction error filters," *IEEE Trans. Signal Process.,* vol. 48, pp. 1928–1936.

Lorenz, H., G. M. Richter, M. Capaccioli, and G. Longo (1993). "Adaptive filtering in astronomical image processing. I. Basic considerations and examples," *Astron. Astrophys*., vol. 277, pp. 321–330.

Lucky, R. W. (1973). "A survey of the communication literature: 1968–1973," *IEEE Trans. Information Theory,* vol. IT-19, pp. 725–739.

Lucky, R. W., J. Salz, and E. J. Weldon, Jr. (1968). *Principles of Data Communication,* McGraw-Hill, New York.

Luenberger, D. G. (1969). *Optimization by Vector Space Methods,* Wiley, New York.

Luk, F. T., and S. Qiao (1989). "Analysis of a recursive least-squares signal-processing algorithm," *SIAM J. Sci. Stat. Comput*., vol. 10, pp. 407–418.

Lynch, M. R., and P. J. Rayner (1989). "The properties and implementation of the non-linear vector space connectionist model," in *Proc. First IEE Int. Conf. Artif. Neural Networks,* London, pp. 186–190.

Macchi, O., N. J. Bershad, and M. M-boup (1991). "Steady-state superiority of LMS over LS for time-varying line enhancer in noisy environment," *IEE Proc. (London), part F,* vol. 138, pp. 354–360.

Macchi, O., and M. Jaidane-Saidne (1989). "Adaptive IIR filtering and chaotic dynamics: Application to audio-frequency coding," *IEEE Trans. Circuits Syst*., vol. 36, pp. 591–599.

Macchi, O., and M. Turki (1992). "The nonstationarity degree: Can an adaptive filter be worse than no processing?" in *Proc. IFAC International Symposium on Adaptive Systems in Control and Signal Processing,* Grenoble, France, pp. 743–747.

Mäkliä, P. M. (1990). "Approximation of stable systems by Laguerre filters," *Automatica,* vol. 26, pp. 333–345.

Mandic, D. P., J. Baltersee, and J. A. Chambers (1998). "Nonlinear prediction of speech with a pipelined recurrent neural network and advanced learning algorithms," in A. Prochazka, J. Uhlir, P. J. W. Rayner, and N. G. Kingsbury, eds., *Signal Analysis and Prediction,* Birkhauser, Boston, pp. 291–309.

Mandic, D. P., and J. Chambers (1999). "Exploiting inherent relationships in RNN architectures," *Neural Networks,* vol. 12, pp. 1341–1345.

Manolakis, D., F. Ling, and J. G. Proakis (1987). "Efficient time-recursive least-squares algorithms for finite-memory adaptive filtering," *IEEE Trans. Circuits Syst*., vol. CAS-34, pp. 400–408.

Marcos, S., and O. Macchi (1987). "Tracking capability of the least mean square algorithm: Application to an asynchronous echo canceller," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP–35, pp. 1570–1578.

Marple, S. L., Jr. (1980). "A new autoregressive spectrum analysis algorithm," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-28, pp. 441–454.

Marple, S. L., Jr. (1981). "Efficient least squares FIR system identification," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-29, pp. 62–73.

Marshall, D. F., W. K. Jenkins, and J. J. Murphy (1989). "The use of orthogonal transforms for improving performance of adaptive filters," *IEEE Trans. Circuits Syst*., vol. 36, pp. 474–484.

Mason, S. J. (1956). "Feedback theory: Further properties of signal flow graphs," *Proc. IRE,* vol. 44, pp. 920–926.

Mathews, V. J., and Z. Xie (1993). "A stochastic gradient adaptive filter with gradient adaptive step size," *IEEE Trans. Signal Process*., vol. 41, pp. 2075–2087.

Mathias, R. (1995). "Accurate eigen system computation by Jacobi methods," *SIMAX,* vol. 16, pp. 977–1003.

Maybeck, P. S. (1979). *Stochastic Models, Estimation, and Control,* vol. 1, Academic Press, New York.

Maybeck, P. S. (1982). *Stochastic Models, Estimation, and Control,* vol. 2, Academic Press, New York.

McCanny, J. V., and J. G. McWhirter (1987). "Some systolic array developments in the United Kingdom," *Computer,* vol. 2, pp. 51–63.

McCool, J. M., et al. (1981). *An Adaptive Detector,* U.S. Patent 4,243.935.

McCulloch, W. S., and W. Pitts (1943). "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics,* vol. 5, pp. 115–133.

McLachlan, G. J., and K. E. Basford (1988). *Mixture Models: Inference and Applications to Clustering,* Dekker, New York.

McWhirter, J. G. (1989). "Algorithmic engineering—An emerging technology," *Proc. SPIE,* vol. 1152, *Real-Time Signal Processing* VI, San Diego.

Medaugh, R. S., and L. J. Griffiths (1981). "A comparison of two linear predictors," in *Proc. IEEE ICASSP,* Atlanta, pp. 293–296.

Mehra, R. K. (1972). "Approaches to adaptive filtering," *IEEE Trans. Autom. Control,* vol. AC-17, pp. 693–698.

Mendel, J. M. (1973). *Discrete Techniques of Parameter Estimation: The Equation Error Formulation,* Dekker, New York.

Mendel, J. M. (1974). "Gradient estimation algorithms for equation error formulations," *IEEE Trans. Autom. Control,* vol. AC-19, pp. 820–824.

Mendel, J. M. (1986). "Some modeling problems in reflection seismology," *IEEE ASSP Mag*., vol. 3, pp. 4–17.

Mendel, J. M. (1990). *Maximum-Likelihood Deconvolution: A Journey into Model-Based Signal Processing,* Springer-Verlag, New York.

Mendel, J. M. (1990). "Introduction," *IEEE Trans. Autom. Control,* vol. AC-35, Special Issue on Higher Order Statistics in System Theory and Signal Processing, p. 3.

Mendel, J. M. (1995). *Lessons in Digital Estimation Theory,* 2nd ed., Prentice-Hall, Upper Saddle River, NJ.

Merched, R., and A. H. Sayed (2000). "Order-recursive RLS Laguerre adaptive filtering," *IEEE Trans. Signal Process.,* vol. 48, pp. 3000–3010.

Merched, R., and A. H. Sayed (2001). "RLS-Laquette lattice adaptive filtering: Error-feedback, normalized, and array-based algorithms," *IEEE Trans. Signal Process.,* vol. 49, pp 2565–2576.

Merched, R., P. S. R. Diniz, and M. R. Petraglia (1999). "A new delayless subband adaptive filter structure," *IEEE Trans. Signal Process.,* vol. 47, pp. 1580–1591.

Meriam, K. A., et al. (1995). "Prediction error methods for time-domain blind identification of multichannel FIR filters," in *Proc. IEEE ICASSP,* Detroit, vol. 3, pp. 1968–1971.

Mermoz, H. F. (1981). "Spatial processing beyond adaptive beamforming," *J. Acoust. Soc. Am.,* vol. 70, pp. 74–79.

Messerchmitt, D. G. (1984). "Echo cancellation in speech and data transmission," *IEEE J. Sel. Areas Commun*., vol. SAC-2, pp. 283–297.

Metford, P. A. S., and S. Haykin (1985). "Experimental analysis of an innovations-based detection algorithm for surveillance radar," *IEE Proc. (London), Part F,* vol. 132, pp. 18–26.

Miao, K. X., H. Fan, and M. Doroslovački (1994). "Cascade normalized lattice adaptive IIR filters," *IEEE Trans. Signal Process.,* vol. 42, pp. 721–742.

Middleton, D. (1960). *An Introduction to Statistical Communication Theory,* McGraw-Hill, New York.

Miranda, M. D., and M. Gerken (1997). "A hybrid least squares QR-lattice algorithm using a priori errors," *IEEE Trans. Signal Process.,* vol. 45, pp. 2900–2911.

Moody, J. E., and C. J. Darken (1989). "Fast learning in networks of locally-tuned processing units," *Neural Computation,* vol. 1, pp. 281–294.

Moonen, M., and J. G. Mcwhirter (1993). "Systolic array for recursive least squares by inverse updating," *Electronics Letters,* vol. 29, No. 13, pp. 1217–1218.

Moonen, M., and J. Vandewalle (1990). "Recursive least squares with stabilized inverse factorization," *Signal Process*., vol. 21, pp. 1–15.

Morf, M., T. Kailath, and L. Ljung (1976). "Fast algorithms for recursive identification," in *Proc. 1976 IEEE Conf. Decision Control,* Clearwater Beach, FL, pp. 916–921.

Morf, M., A. Vieira, and D. T. Lee (1977). "Ladder forms for identification and speech processing," in *Proc. 1977 IEEE Conf. Decision Control,* New Orleans, pp. 1074–1078.

Moroney, P. (1983). *Issues in the Implementation of Digital Feedback Compensators,* MIT Press, Cambridge, MA.

Morozov, V. A. (1993). *Regularization Methods for Ill-Posed Problems,* CRC Press, Boca Raton, FL.

Moschner, J. L. (1970). *Adaptive Filter with Clipped Input Data,* Tech. Rep. 6796-1, Stanford University Center for Systems Research, Stanford, CA.

Mueller, M. S. (1981). "Least-squares algorithms for adaptive equalizers," *Bell Syst. Tech. J*., vol. 60, pp. 1905–1925.

Mueller, M. S. (1981). "On the rapid initial convergence of least-squares equalizer adjustment algorithms," *Bell Syst. Tech. J*., vol. 60, pp. 2345–2358.

Mulgrew, B. (1987). "Kalman filter techniques in adaptive filtering," *IEE Proc. (London), Part F,* vol. 134, pp. 239–243.

Mulgrew, B., and C. F. N. Cowan (1987). "An adaptive Kalman equalizer: Structure and performance," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-35, pp. 1727–1735.

Mulgrew, B., and C. F. N. Cowan (1988). *Adaptive Filters and Equalizers,* Kluwer, Boston.

Mullis, C. T., and R. A. Roberts (1976). "The use of second-order information in the approximation of discrete-time linear systems," *IEEE Trans. Acoust., Speech, Signal Process.,* vol. 24, pp. 226–238.

Murano, K., et al. (1990). "Echo cancellation and applications," *IEEE Commun*., vol. 28, pp. 49–55.

Musicus, B. R. (1985). "Fast MLM power spectrum estimation from uniformly spaced correlations," *IEEE Trans. Acoust., Speech, Signal Process*., vol. ASSP-33, pp. 1333–1335.

Narayan, S. S., and A. M. Peterson (1981). "Frequency domain LMS algorithm," *Proc. IEEE*., vol. 69, pp. 124–126.

Nau, R. F., and R. M. Oliver (1979). "Adaptive filtering revisited," *J. Oper. Res. Soc*., vol. 30, pp. 825–831.

Ngia, L. S. H., and J. Sjöberg (2000). "Efficient training of neural nets for nonlinear adaptive filtering using a recursive Levenberg–Marquardt algorithm," *IEEE Trans. Signal Process.,* vol. 48, pp. 1915–1927.

Nielsen, P. A., and J. B. Thomas (1988). "Effect of correlation on signal detection in arctic under-ice noise," in *Conf. Rec. Twenty-Second Asilomar Conference on Signals, Systems and Computers,* Pacific Grove, CA, pp. 445–450.

Nikias, C. L. (1991). "Higher-order spectral analysis," in S. Haykin, ed., *Advances in Spectrum Analysis and Array Processing,* vol. 1, Prentice-Hall, Englewood Cliffs, NJ.

Ninness, B., and J. C. Gómez (1998). "Frequency domain analysis of tracking and noise performance of adaptive algorithms," *IEEE Trans. Signal Process.,* vol. 46, pp. 1314–1332.

Norgaard, M., N. Poulsen, and O. Rav (2000). *Advances in Derivative-Free State Estimation for Nonlineer Systems,* TR, Technical University of Denmark.

Normile, J. O. (1983). "Adaptive filtering with finite wordlength constraints," *IEE Proc. (London),* part E, vol. 130, pp. 42–46.

Olmos, S., and P. Laguna (2000). "Steady-state MSE convergence of LMS adaptive filters with deterministic reference inputs," *IEEE Trans. Signal Process.,* vol. 48, pp. 2229–2241.

Oppenheim, A. V., and J. S. Lim (1981). "The importance of phase in signals," *Proc. IEEE,* vol. 69, pp. 529–541.

Owsley, N. L. (1973). "A recent trend in adaptive spatial processing for sensor arrays: Constrained adaptation," in J. W. R. Griffiths et al., eds., *Signal Processing,* Academic Press, New York, pp. 591–604.

Ozeki, K., and T. Umeda (1984). "An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties," *Electron. Commun. Japan,* vol. 67-A, pp. 126–132.

Panda, G., B. Mulgrew, C. F. N. Cowan, and P. M. Grant (1986). "A self-orthogonalizing efficient block adaptive filter," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-34, pp. 1573–1582.

Papadias, C. (2000). "Blind separation of independent sources based on multiuser kurtosis optimization criteria," in S. Haykin, ed., *Unsupervised Adaptive Filtering, Vol. II: Blind Deconvolution,* Wiley, New York. pp. 147–179.

Papadias, C. B., and D. T. M. Slock (1994). "New adaptive blind equalization algorithms for constant modulus constellations," in *Proc. IEEE ICASSP94,* Adelaide, Australia, pp. 321–324.

Papoulis, A. (1984). *Probability, Random Variables, and Stochastic Processes,* 2nd ed., McGraw-Hill, New York.

Parlett, B. N. (1980). *The Symmetric Eigenvalue Problem,* Prentice-Hall, Englewood Cliffs, NJ.

Parzen, E. (1962). "On the estimation of a probability density function and mode," *Ann. Math. Stat.,* vol. 33, pp. 1065–1076.

Patra, J. C., and G. Panda (1992). "Performance evaluation of finite precision LMS adaptive filters using probability density approach," *J. Inst. Electron. Telecommun. Eng.,* vol. 38, pp. 192–195.

Peacock, K. L., and S. Treitel (1969). "Predictive deconvolution: Theory and practice," *Geophysics,* vol. 34, pp. 155–169.

Petraglia, M. R., R. G. Alves, and P. S. R. Diniz (2000). "New structures for adaptive filtering in subbands with critical sampling," *IEEE Trans. Signal Process.,* vol. 48, pp. 3316–3327.

Picchi, G., and G. Prati (1984). "Self-orthogonalizing adaptive equalization in the discrete frequency domain," *IEEE Trans. Commun*., vol. COM-32, pp. 371–379.

Picchi, G., and G. Prati (1987). "Blind equalization and carrier recovery using a 'stop-and-go' decision-directed algorithm," *IEEE Trans. Commun*., vol. COM-35, pp. 877–887.

Porat, B., B. Friedlander, and M. Morf (1982). "Square root covariance ladder algorithms," *IEEE Trans. Autom. Control.*, vol. AC-27, pp. 813–829.

Porat, B., and T. Kailath (1983). "Normalized lattice algorithms for least-squares FIR system identification," *IEEE Trans. Acoust. Speech Signal Process.*, vol. ASSP-31, pp. 122–128.

Potter, J. E. (1963). *New Statistical Formulas,* Space Guidance Analysis Memo No. 40, Instrumentation Laboratory, MIT, Cambridge, MA.

Principe, J. C., D. Xu, and J. W. Fisher, III (2000). "Information-theoretic learning," in S. Haykin, ed., *Unsupervised Adaptive Filtering, Vol. I: Blind Source Separation,* Wiley, New York, pp. 265–320.

Proakis, J. G. (1975). "Advances in equalization for intersymbol interference," in A. V. Balakrishnan, ed., *Advances in Communication Systems,* vol. 4, Academic Press, New York, pp. 123–198.

Proakis, J. G. (1991). "Adaptive equalization for TDMA digital mobile radio," *IEEE Trans. Vehicular Technol.*, vol. 40, pp. 333–341.

Proudler, I. K., J. G. Mcwhirter, M. Moonen, and G. Hekstra (1996). "Formal derivation of a systolic array for recursive least squares estimation," *IEEE Trans. Circuits Syst. Analog Digital Signal Process.,* vol. 43, pp. 247–254.

Proudler, I. K., J. G. Mcwhirter, and T. J. Shepherd (1991). "Computationally efficient, QR decomposition approach to least squares adaptive filtering," *IEE Proc. (London), Part F,* vol. 138, pp. 341–353.

Puskorius, G. V., and L. A. Feldkamp (2001). "Parameter-based Kalman filter training: Theory and implementation," in S. Haykin, ed., *Kalman Filters and Neural Networks,* Wiley, New York.

Qin, L., and M. G. Belanger (1996). "Convergence analysis of a variable step-size normalized LMS adaptive filter algorithm," in *Proceedings of EUPSICO,* pp. 1231–1234.

Quirk, K. J., L. B. Milstein, and J. R. Zeidler (2000). "A performance bound for the LMS estimator," *IEEE Trans. Information Theory,* vol. 46, pp. 1150–1158.

Qureshi, S. (1982). "Adaptive equalization," *IEEE Commun. Soc. Mag.*, vol. 20, pp. 9–16.

Qureshi, S. U. H. (1985). "Adaptive equalization," *Proc. IEEE,* vol. 73, pp. 1349–1387.

Rabiner, L. R., and B. Gold (1975). *Theory and Application of Digital Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ.

Rabiner, L. R., and R. W. Schafer (1978). *Digital Processing of Speech Signals,* Prentice-Hall, Englewood Cliffs, NJ.

Rader, C. M., and A. O. Steinhardt (1986). "Hyperbolic Householder transformations," *IEEE Trans. Acoust. Speech, Signal Process.*, vol. ASSP-34, pp. 1589–1602.

Rao, C. R. (1973). *Linear Statistical Inference and Its Applications,* 2nd ed., Wiley, New York.

Rao, K. R., and P. Yip (1990). *Discrete Cosine Transform: Algorithms, Advantages, Applications,* Academic Press, San Diego.

Rao, S. K., and T. Kailath (1986). "What is a systolic algorithm?" in *Proc. SPIE, Highly Parallel Signal Processing Architectures,* San Diego, vol. 614, pp. 34–48.

Rayleigh, L. (1879). "Investigations in optics with special reference to the spectral scope," *Philos. Mag.*, vol. 8, pp. 261–274.

Rayner, P. J. W., and M. F. Lynch (1989), "A new connectionist model based on a non-linear adaptive filter," in *Proc. IEEE ICASSP,* Glasgow, pp. 1191–1194.

Reddi, S. S. (1979). "Multiple source location—A digital approach," *IEEE Trans. Aerospace Electron. Syst.*, vol. AES-15, pp. 95–105.

Reddi, S. S. (1984). "Eigenvector properties of Toeplitz matrices and their application to spectral analysis of time series," *Signal Process.*, pp. 45–56.

Reddy, V. U., B. Egardt, and T. Kailath (1981). "Optimized lattice-form adaptive line enhancer for a sinusoidal signal in broad-band noise," *IEEE Trans. Acoust. Speech Signal Process.*, vol. ASSP-29, pp. 702–710.

Regalia, P. A. (1992). "Numerical stability issues in fast least-squares adaptation algorithms," *Optical Engineering,* vol. 31, pp. 1144–1152.

Regalia, P. A. (1993). "Numerical stability properties of a QR-based fast least squares algorithm," *IEEE Trans. Signal Process.*, vol. 41, pp. 2096–2109.

Regalia, P. A. (1994). *Adaptive IIR Filtering in Signal Processing and Control,* Dekker, New York.

Regalia, P. A. (1994). "An unbiased equation error identifier and reduced order approximations," *IEEE Trans. Signal Process.,* vol. 42, pp. 1397–1412.

Reuter, M., and J. R. Zeidler (1999). "Nonlinear effects in LMS adaptive equalizers," *IEEE Trans. Signal Process.,* vol. 47, pp. 1570–1579.

Rickard, J. T., et al. (1981). "A performance analysis of adaptive line enhancer-augmented spectral detectors," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 534–541.

Rickard, J. T., J. R. Zeidler, M. J. Denting, and M. Shensa (1981). "A performance analysis of adaptive line enhancer-augmented spectral detectors," *IEEE Trans. Circuits Syst*., vol. CAS-28, no. 6, pp. 534–541.

Riddle, A. (1994). "Engineering software: Mathematical power tools," *IEEE Spectrum,* vol. 31, pp. 35–17, 95.

Rissanen, J. (1986). "Stochastic complexity and modeling," *Ann. Stat*., vol. 14, pp. 1080–1100.

Robinson, E. A. (1954). *Predictive decomposition of time series with applications for seismic exploration,* Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Robinson, E. A. (1984). "Statistical pulse compression," *Proc. IEEE,* vol. 72, pp. 1276–1289.

Robinson, E. A., and S. Treitel (1980). *Geophysical Signal Analysis,* Prentice-Hall, Englewood Cliffs, NJ.

Rosenblatt, M. (1985). *Stationary Sequences and Random Fields,* Birkhäuser, Stuttgart.

Rumelhart, D. E., and J. L. McClelland, eds. (1986). *Parallel Distributed Processing,* vol. 1, *Foundations,* MIT Press, Cambridge, MA.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). "Learning representations by backpropogating errors," *Nature,* vol. 323, pp. 533–536.

Rupp, M. (1998). "A family of adaptive filter algorithms with decorrelating properties," *IEEE Trans. Signal Process.,* vol. 46, pp. 771–775.

Sambur, M. R. (1978). "Adaptive noise cancelling for speech signals," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-26, pp. 419–423.

Samson, C. (1982). "A unified treatment of fast Kalman algorithms for identification," *Int. J. Control,* vol. 35, pp. 909–934.

Sanders, J. A., and F. Verhulst (1985). *Averaging Methods in Nonlinear Dynamical Systems,* Springer-Verlag, New York.

Sari, H. (1992). "Adaptive equalization of digital line-of-sight radio systems," in L. Dugard, M. M'Saad, and I. D. Landau, eds., *Adaptive Systems in Control and Signal Processing 1992,* Pergamon Press, Oxford, U.K., pp. 505–510.

Sato, Y. (1994). "Blind equalization and blind sequence estimation," *IEICE Trans. Commun*., vol. E77-B, pp. 545–556.

Satorius, E. H., et al. (1983), "Fixed-point implementation of adaptive digital filters," in *Proc. ICAASP,* Boston, pp. 33–36.

Sayed, A. H., and M. Rupp (1997). "Robustness issues in adaptive filtering," in V. K. Madisetti and D. B. Williams, eds., *The Digital Signal Processing Handbook,* CRC Press, Boca Raton, FL, pp. 21–1 to 21–35.

Sayyarrodsari, B., J. P. How, B. Hassibi, and A. Carrier (2001). "Estimation-based synthesis of H_∞-optimal adaptive FIR filters for filtered-LMS problems," *IEEE Trans. Signal Process.,* vol. 49, pp. 164–178.

Scharf, L. L. (1991). *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis,* Addison–Wesley, Reading, MA.

Scharf, L. L., and L. T. Mcwhorter (1994). "Quadratic estimators of the correlation matrix," in *IEEE ASSP Workshop on Statistical Signal and Array Processing,* Quebec City, Canada.

Scharf, L. L., and J. K. Thomas (1995). "Data adaptive low rank modelling," *National Radio Science Meeting,* Boulder, CO, p. 200.

Scharf, L. L., and J. K. Thomas (1998). "Wiener filters in canonical coordinates for transform coding, filtering, and quantizing," *IEEE Trans. Signal Process.,* vol. 46, pp. 647–654.

Schell, S. V, and W. A. Gardner (1993). "Spatio-temporal filtering and equalization for cyclostationary signals," in C. T. Leondes, ed., *Control and Dynamic Systems,* vol. 66, Academic Press, New York, pp. 1–85.

Schertler, T., and G. U. Schmidt (1997). "Implementation of a low-cost acoustic echo canceller," in *Proceedings of the IWAENC'07,* London, U.K., pp. 49–52.

Schetzen, M. (1981). "Nonlinear system modeling based on the Wiener theory," *Proc. IEEE,* vol. 69, pp. 1557–1572.

Schreiber, R. J. (1986). "Implementation of adaptive array algorithm," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-34, pp. 1038–1045.

Schroeder, M. R. (1985). "Linear predictive coding of speech: Review and current directions," *IEEE Commun. Mag*., vol. 23, pp. 54–61.

Schwartz, L. (1967). *Cours d'Analyse,* vol. II, Hermann, Paris, pp. 271–278.

Senne, K. D. (1968). *Adaptive Linear Discrete-Time Estimation,* Tech. Rep., 6778-5, Stanford University Center for Systems Research, Stanford, CA.

Sethares, W. A. (1993). "The least mean square family," in N. Kalouptsidis and S. Theodoridis, eds., *Adaptive System Identification and Signal Processing Algorithms,* Prentice-Hall, Englewood Cliffs, NJ, pp. 84–122.

Shan, T.-J., and T. Kailath (1985). "Adaptive beamforming for coherent signals and interference," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-33, pp. 527–536.

Shanbhag, N. R., and K. K. Parhi (1994). *Pipelined Adaptive Digital Filters,* Kluwer, Boston.

Sharpe, S. M., and L. W. Nolte (1981). "Adaptive MSE estimation," in *Proc. ICASSP,* Atlanta, pp. 518–521.

Shensa, M. J. (1980). "Non-Wiener solutions of the adaptive noise canceller with a noisy reference," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-28, pp. 468–473.

Shi, K. H., and F. Kozin (1986). "On almost sure convergence of adaptive algorithms," *IEEE Trans. Autom. Control,* vol. AC-31, pp. 471–474.

Shichor, E. (1982). "Fast recursive estimation using the lattice structure," *Bell Syst. Tech. J*., vol. 61, pp. 97–115.

Shynk, J. J. (1989). "Adaptive IIR filtering," *IEEE ASSP Mag*., vol. 6, pp. 4–21.

Sibul, L. H. (1984). "Application of singular value decomposition to adaptive beamforming," in *Proc. ICASSP,* San Diego, vol. 2, pp. 33.11/1–4.

Sicuranza, G. L. (1985). "Nonlinear digital filter realization by distributed arithmetic," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-33, pp 939–945.

Sicuranza, G. L., and G. Ramponi (1986). "Adaptive nonlinear digital filters using distributed arithmetic," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-34, pp. 518–526.

Skolnik, M. I. (1980). *Introduction to Radar Systems,* New York: McGraw-Hill.

Slock, D. T. M. (1989). *Fast algorithms for fixed-order recursive least-squares parameter estimation,* Ph.D. dissertation, Stanford University, Stanford, CA.

Slock, D. T. M. (1991). "Fractionally-spaced subband and multiresolution adaptive filters," *Proceedings of ICASSP91,* pp. 3693–3696.

Slock, D. T. M. (1992). "The backward consistency concept and roundoff error propagation dynamics in RLS algorithms," *Optical Engineering,* vol. 31, pp. 1153–1169.

Slock, D. T. M. (1993). "On the convergence behavior of the LMS and the normalized LMS filters," *IEEE Trans. Signal Process.,* vol. 41, pp. 2811–2825.

Slock, D. T. M. (1994). "Blind fractionally-spaced equalization, perfect-reconstruction filter banks and multichannel linear prediction," in *Proc. IEEE ICASSP,* Adelaide, Australia, vol. 4, pp. 585–588.

Slock, D. T. M., and C. B. Papadias (1995). "Further results on blind identification and equalization of multiple FIR channels," in *Proc. ICASSP,* Detroit, vol. 3, pp. 1964–1967.

Söderström, T., and P. Stoica (1981). "On the stability of dynamic models obtained by least-squares identification," *IEEE Trans. Autom. Control,* vol. 26, pp. 575–577.

Solo, V. (1989). "The limiting behavior of LMS," *IEEE Trans. Acoust. Speech Signal Process*., vol. 37, pp. 1909–1922.

Solo, V. (1997). "The stability of LMS," *IEEE Trans. Signal Process.,* vol. 45, pp. 3017–3026.

Solo, V., and X. Kong (1995). *Adaptive Signal Processing Algorithms,* Prentice-Hall, Englewood Cliffs, NJ.

Sondhi, M., and D. A. Berkley (1980). "Silencing echoes in the telephone network," *Proc. IEEE,* vol. 68, pp. 948–963.

Sondhi, M. M., and A. J. Presti (1966). "A self-adaptive echo canceller," *Bell Syst. Tech. J*., vol. 45, pp. 1851–1854.

Soni, T., J. R. Zeidler, and W. H. Ku (1995). "Behavior of the partial correlation coefficients of a least squares lattice filter in the presence of a nonstationary chirp input," *IEEE Trans. Signal Process*., vol. 43, pp. 852–863.

Sontag, E. D. (1990). *Mathematical Control Theory: Deterministic Finite Dimensional Systems,* Springer-Verlag.

Soo, J.-S., and K. K. Pagn (1991). "A multistep size (MSS) frequency domain adaptive filter," *IEEE Trans. Signal Process*., vol. 39, pp. 115–121.

Sorenson, H. W. (1967). "On the error behavior in linear minimum variance estimation problems," *IEEE Trans. Autom. Control,* vol. AC-12, pp. 557–562.

Sorenson, H. W., ed. (1985). *Kalman Filtering: Theory and Application,* IEEE Press, New York.

Special Issue on Adaptive Antennas (1976). *IEEE Trans. Antennas Propag*., vol. AP-24, September.

Special Issue on Adaptive Arrays (1983). *IEE Proc. Commun. Radar Signal Process.,* London, vol. 130, pp. 1–151.

Special Issue on Adaptive Filters (1987). *IEE Proc. Commun. Radar Signal Process*., London, vol. 134, Pt. F.

Special Issue on Adaptive Processing Antenna Systems (1986). *IEEE Trans. Antennas Propag*., vol. AP-34, pp. 273–462.

Special Issue on Adaptive Signal Processing (1981). *IEEE Trans. Circuits Syst*., vol. CAS-28, pp. 465–602.

Special Issue on Adaptive Systems (1976). *Proc. IEEE,* vol. 64, pp. 1123–1240.

Special Issue on Adaptive Systems and Applications (1987). *IEEE Trans. Circuits Syst*., vol. CAS-34, pp. 705–854.

Special Issue on Blind System Identification and Estimation (1998). *Proc. IEEE,* vol. 86, pp. 1903–2089.

Special Issue on Higher Order Statistics in System Theory and Signal Processing (1990). *IEEE Trans. Autom. Control,* vol. AC-35, pp. 1–56.

Special Issue on Linear Adaptive Filtering (1984). *IEEE Trans. Information Theory,* vol. IT-30, pp. 131–295.

Special Issue on Linear-Quadratic-Gaussian Problems (1971). *IEEE Trans. Autom. Control,* vol. AC-16, December.

Special Issue on Neural Networks (1990). *Proc. IEEE,* vol. 78: Neural Nets I, September; Neural Nets II, October.

Special Issue on Spectral Estimation (1982). *Proc. IEEE,* vol. 70, pp. 883–1125.

Special Issue on System Identification and Time-Series Analysis (1974). *IEEE Trans. Autom. Control,* vol. AC-19, pp. 638–951.

Special Issue on Systolic Arrays (1987). *Computer,* vol. 20, no. 7.

Strobach, P. (1990). *Linear Prediction Theory,* Springer-Verlag, New York.

Subba Rao, T., and M. M. Gabr (1980). "A test for linearity of stationary time series," *J. Time Series Analysis,* vol. 1, pp. 145–158.

Suzuki, H. (1994). "Adaptive signal processing for optimal transmission in mobile radio communications," *IEICE Trans. Commun*., vol. E77-B, pp. 535–544.

Tarrab, M., and A. Feuer (1988). "Convergence and performance analysis of the normalized LMS algorithm with uncorrelated Gaussian data," *IEEE Trans. Information Theory,* vol. IT-34, pp. 680–691.

Tetterington, D. M., A. F. M. Smith, and U. E. Makov (1985). *Statistical Analysis of Finite Mixture Distributions,* Wiley, New York.

Thakor, N. V., and Y.-S. Zhu (1991). "Applications of adaptive filtering to ECG analysis: Noise cancellation and arrhythmia detection," *IEEE Trans. Biomed. Eng*., vol. 38, pp. 785–794.

Theodoridis, S., C. M. S. See, and C. F. N. Cowan (1992). "Nonlinear channel equalization using clustering techniques," in *ICC,* Chicago, vol. 3, pp. 1277–1279.

Thomson, W. T. (1950). "Transmission of elastic waves through a stratified solid medium," *J. Appl. Phys*., vol. 21, pp. 89–93.

Tobias, O. J., J. C. M. Bermúdez, and N. J. Bershad (2000). "Mean weight behavior of the filtered-X LMS algorithm," *IEEE Trans. Signal Process.,* vol. 48, pp. 1061–1075.

Torkkola, K. (2000). "Blind separation of delayed and convolved sources," in S. Haykin, ed., *Unsupervised Adaptive Filtering, Vol. I: Blind Source Separation,* Wiley, New York, pp. 321–376.

Treichler, J. R., C. R. Johnson, Jr., and M. G. Larimore (1987). *Theory and Design of Adaptive Filters,* Wiley-Interscience, New York.

Treichler, J. R., and M. G. Larimore (1985). "New processing techniques based on the constant modulus adaptive algorithm," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-33, pp. 420–431.

Treichler, J. R., and M. G. Larimore (1985). "The tone capture properties of CMA-based interference suppressions," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-33, pp. 946–958.

Tugnait, J. K. (1994). "Testing for linearity of noisy stationary signals," *IEEE Trans. Signal Process*., vol. 42, pp. 2742–2748.

Tugnait, J. K. (1995). "On fractionally-spaced blind adaptive equalization under symbol timing offsets using Godard and related equalizers," in *Proc. ICASSP,* Detroit, vol. 3, pp. 1976–1979.

Ulrych, T. J., and R. W. Clayton (1976). "Time series modelling and maximum entropy," *Phys. Earth Planet. Inter*., vol. 12, pp. 188–200.

Vaidyanathan, P. P. (1987). "Quadrature mirror filter bands, M-band extensions and perfect reconstruction techniques," *IEEE ASSP Mag.* vol. 4, pp. 4–20.

Valenzuela, R. A. (1989). "Performance of adaptive equalization for indoor radio communications," *IEEE Trans. Commun*., vol. 37, pp. 291–293.

Van de Kerkhof, L. M., and W. J. W. Kitzen (1992). "Tracking of a time-varying acoustic impulse response by an adaptive filter," *IEEE Trans. Signal Process*., vol. 40, pp. 1285–1294.

Van Huffel, S., and J. Vandewalle (1988). "The partial total least squares algorithm," *J. Comput. Appl. Math*., vol. 21, pp. 333–341.

Van Huffel, S., J. Vandewalle, and A. Haegemans (1987). "An efficient and reliable algorithm for computing the singular subspace of a matrix, associated with its smallest singular values," *J. Comput. Appl. Math*., vol. 19, pp. 313–330.

Varvitsiotis, A. P., S. Theodoridis, and G. Moustakides (1989). "A new novel structure for adaptive LS FIR filtering based on QR decomposition," in *Proc. IEEE ICASSP,* Glasgow, pp. 904–907.

Vembu, S., S. Verdú, R. A. Kennedy, and W. Sethares (1994). "Convex cost functions in blind equalization," *IEEE Trans. Signal Process*., vol. 42, pp. 1952–1960.

Verhaegen, M. H., and P. Van Dooren (1986). "Numerical aspects of different Kalman filter implementations," *IEEE Trans. Autom. Control,* vol. AC-31, pp. 907–917.

Vetterli, M., and J. Kovačevič (1995). *Wavelets and Subband Coding,* Prentice-Hall, Englewood Cliffs, NJ.

Volder, J. E. (1959). "The CORDIC trigonometric computing technique," *IEEE Trans. Electron. Comput*., vol. EC-8, pp. 330–334.

Wahlberg, B. (1991). "System identification using Laguerre models," *IEEE Trans. Autom. Control,* vol. 36, pp. 551–562.

Walach, E., and B. Widrow (1984). "The least mean fourth (LMF) adaptive algorithm and its family," *IEEE Trans. Information Theory,* vol. IT-30, Special Issue on Linear Adaptive Filtering, pp. 275–283.

Wan, E. A., and R. van der Merwe (2001). "The unscented Kalman filter," in S. Haykin, ed., *Kalman Filtering and Neural Networks,* Wiley, New York, Chapter 7.

Ward, C. R., et al. (1984). "Application of a systolic array to adaptive beamforming," *IEE Proc. (London),* pt. F, vol. 131, pp. 638–645.

Wax, M., and T. Kailath (1985). "Detection of signals by information theoretic criteria," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-33, pp. 387–392.

Wax, M., and I. Ziskind (1989). "Detection of the number of coherent signals by the MDL principle," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-37, pp. 1190–1196.

Webb, A. R. (1994). "Functional approximation by feed-forward networks: A least-squares approach to generalisation," *IEEE Trans. Neural Networks,* vol. 6, pp. 363–371.

Wedin, P. A. (1973). "On the almost rank deficient case of the least squares problem," *Nordisk Tidskr. Informationsbehandling (BIT),* vol. 13, pp. 344–354.

Wedin, P. A. (1973). "Perturbation theory for pseudo-inverses," *Nordisk Tidskr. Informationsbehandling (BIT),* vol. 13, pp. 217–232.

Wei, P, J. R. Zeidler, and W. H. Ku (1994). "Adaptive recovery of a Doppler-shifted mobile communications signal using the RLS algorithm," in *Conf. Rec. Asilomar Conference on Signals, Systems, and Computers,* Pacific Grove, CA, vol. 2, pp. 1180–1184.

Weigend, A. S., D. E. Rumelhart, and B. A. Huberman (1991). "Generalization by weight elimination with application to forecasting," in *Advances in Neural Information Processing Systems 3,* Morgan Kaufman, San Mateo, CA, pp. 875–882.

Wellstead, P. E., G. R. Wagner, and J. R. Caldas-Pinto (1987). "Two-dimensional adaptive prediction, smoothing and filtering," *IEE Proc. (London),* part F, vol. 134, pp. 253–268.

Werbos, P. J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences,* Ph.D. dissertation, Harvard University, Cambridge, MA.

Werbos, P. J. (1993). *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting,* Wiley-Interscience, New York.

Wheelwright, S. C., and S. Makridakis (1973). "An examination of the use of adaptive filtering in forecasting," *Oper. Res Q*., vol. 24, pp. 55–64.

Widrow, B. (1966). *Adaptive Filters I: Fundamentals,* Rep. SEL-66-126 (TR 6764-6), Stanford Electronics Laboratories, Stanford, CA.

Widrow, B., K. M. Duvall, R. P. Gooch, and W. C. Newman (1982). "Signal cancellation phenomena in adaptive antennas: Causes and cures," *IEEE Trans. Antennas Propag*., vol. AP-30, pp. 469–478.

Widrow, B., and M. Lehr (1990). "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," *Proc IEEE,* vol. 78, Special Issue on Neural Networks I, September.

Widrow, B., et al. (1987). "Fundamental relations between the LMS algorithm and the DFT," *IEEE Trans. Circuits Syst*., vol. CAS-34, pp. 814–819.

Wiener, N. (1956). *The Theory of Prediction,* McGraw-Hill, New York.

Wiener, N. (1958). *Nonlinear Problems in Random Theory,* Wiley, New York.

Wilkinson, J. H. (1965). *The Algebraic Eigenvalue Problem,* Oxford University Press, Oxford, U.K.

Williams, J. R., and G. G. Ricker (1972). "Signal detectability performance of optimum Fourier receivers," *IEEE Trans. Audio Electroacoustics,* vol. AU-20, pp. 254–270.

Williams, R. J., and D. Zipser (1989). "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation,* vol. 1, pp. 270–280.

Williams, R. J., and D. Zipser (1995). "Gradient-based learning algorithms for recurrent networks and their computational complexity." In Chauvin and D. E. Rumelhart, eds., *Backpropagation: Theory, Architecture, and Applications,* Lawrence Erlbaum, Hillsdale, NJ, pp. 433–486.

Wilsky, A. S. (1979). *Digital Signal Processing and Control and Estimation Theory: Points of Tangency, Areas of Intersection, and Parallel Directions,* MIT Press, Cambridge, MA.

Yassa, F. F. (1987). "Optimality in the choice of the convergence factor for gradient-based adaptive algorithms," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-35, pp. 48–59.

Yoganandam, Y., V. U. Reddy, and T. Kailath (1988). "Performance analysis of the adaptive line enhancer for sinusoidal signals in broad-band noise," *IEEE Trans. Acoust. Speech Signal Process*., vol. ASSP-36, pp. 1749–1757.

YOUNG, P C. (1984). *Recursive Estimation and Time-Series Analysis,* Springer-Verlag, New York.

YUAN, J.-T. (2000). "QR-decomposition–based least-squares lattice interpolators," *IEEE Trans. Signal Process.,* vol. 48, pp. 70–79.

YUAN, J.-T., and J. A. STULLER (1995). "Least-squares order-recursive lattice smoothers," *IEEE Trans. Signal Process*., vol. 43, pp. 1058–1067.

ZHANG, Q-T., and S. HAYKIN (1983). "Tracking characteristics of the Kalman filter in a nonstationary environment for adaptive filter applications," in *Proc. IEEE ICASSP,* Boston, pp. 671–674.

ZHANG, Q-T., S. HAYKIN, and P. YIP (1989). "Performance limits of the innovations-based detection-algorithm," *IEEE Trans. Information Theory,* vol. IT-35, pp. 1213–1222.

ZIEGLER, R. A., and J. M. CIOFFI (1989). "A comparison of least squares and gradient adaptive equalization for multipath fading in wideband digital mobile radio," in *GLOBECOM,* vol. 1, New York, pp.102–106.

ZIEGLER, R. A., and J. M. CIOFFI (1992). "Adaptive equalization for digital wireless data transmission," in *Virginia Tech Second Symposium on Wireless Personal Communications Proceedings,* pp. 5/1–5/12.

Index