

Dimitris G. Manolakis
Vinay K. Ingle
Stephen M. Kogon



statistical and adaptive
**signal
processing**

SPECTRAL ESTIMATION, SIGNAL MODELING,
ADAPTIVE FILTERING AND ARRAY PROCESSING

Statistical and Adaptive Signal Processing

Recent Titles in the Artech House Signal Processing Library

Computer Speech Technology, Robert D. Rodman

Digital Signal Processing and Statistical Classification, George J. Miao and Mark A. Clements

Handbook of Neural Networks for Speech Processing,
Shigeru Katagiri, editor

Hilbert Transforms in Signal Processing, Stefan L. Hahn

Phase and Phase-Difference Modulation in Digital Communications,
Yuri Okunev

Signal Processing Fundamentals and Applications for Communications and Sensing Systems, John Minkoff

Signals, Oscillations, and Waves: A Modern Approach, David Vakman

Statistical Signal Characterization, Herbert L. Hirsch

Statistical Signal Characterization Algorithms and Analysis Programs, Herbert L. Hirsch

Voice Recognition, Richard L. Klevans and Robert D. Rodman

For further information on these and other Artech House titles, including previously considered out-of-print books now available through our In-Print-Forever® (IPF®) program, contact:

Artech House
685 Canton Street
Norwood, MA 02062
Phone: 781-769-9750
Fax: 781-769-6334
e-mail: artech@artechhouse.com

Artech House
46 Gillingham Street
London SW1V 1AH UK
Phone: +44 (0)20 7596-8750
Fax: +44 (0)20 7630-0166
e-mail: artech-uk@artechhouse.com

Find us on the World Wide Web at: www.artechhouse.com

Statistical and Adaptive Signal Processing

Spectral Estimation, Signal Modeling, Adaptive
Filtering, and Array Processing

Dimitris G. Manolakis

*Massachusetts Institute of Technology
Lincoln Laboratory*

Vinay K. Ingle

Northeastern University

Stephen M. Kogon

*Massachusetts Institute of Technology
Lincoln Laboratory*



**ARTECH
HOUSE**

BOSTON | LONDON
artechhouse.com

Library of Congress Cataloging-in-Publication Data

A catalog record for this book is available from the U.S. Library of Congress.

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library.

This is a reissue of a McGraw-Hill book.

Cover design by Igor Valdman

© 2005 ARTECH HOUSE, INC.

**685 Canton Street
Norwood, MA 02062**

All rights reserved. Printed and bound in the United States of America. No part of this book may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the publisher.

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Artech House cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

International Standard Book Number: 1-58053-610-7

10 9 8 7 6 5 4 3 2 1

To my beloved wife, Anna, and to the loving memory of my father, Gregory.
DGM

To my beloved wife, Usha, and adoring daughters, Natasha and Trupti.
VKI

To my wife and best friend, Lorna, and my children, Gabrielle and Matthias.
SMK

ABOUT THE AUTHORS

DIMITRIS G. MANOLAKIS, a native of Greece, received his education (B.S. in physics and Ph.D. in electrical engineering) from the University of Athens, Greece. He is currently a member of the technical staff at MIT Lincoln Laboratory, in Lexington, Massachusetts. Previously, he was a Principal Member, Research Staff, at Riverside Research Institute. Dr. Manolakis has taught at the University of Athens, Northeastern University, Boston College, and Worcester Polytechnic Institute; and he is coauthor of the textbook *Digital Signal Processing: Principles, Algorithms, and Applications* (Prentice-Hall, 1996, 3d ed.). His research experience and interests include the areas of digital signal processing, adaptive filtering, array processing, pattern recognition, and radar systems.

VINAY K. INGLE is Associate Professor of Electrical and Computer Engineering at Northeastern University. He received his Ph.D. in eletrical and computer engineering from Rensselaer Polytechnic Institute in 1981. He has broad research experience and has taught courses on topics including signal and image processing, stochastic processes, and estimation theory. Professor Ingle is coauthor of the textbooks *DSP Laboratory Using the ADSP-2101 Microprocessor* (Prentice-Hall, 1991) and *DSP Using Matlab* (PWS Publishing Co., Boston, 1996).

STEPHEN M. KOGON received the Ph.D. degree in electrical engineering from Georgia Institute of Technology. He is currently a member of the technical staff at MIT Lincoln Laboratory in Lexington, Massachusetts. Previously, he has been associated with Raytheon Co., Boston College, and Georgia Tech Research Institute. His research interests are in the areas of adaptive processing, array signal processing, radar, and statistical signal modeling.

CONTENTS

Preface	xvii		
1 Introduction	1		
1.1 Random Signals	1		
1.2 Spectral Estimation	8		
1.3 Signal Modeling	11		
<i>1.3.1 Rational or Pole-Zero Models / 1.3.2 Fractional Pole-Zero Models and Fractal Models</i>			
1.4 Adaptive Filtering	16		
<i>1.4.1 Applications of Adaptive Filters / 1.4.2 Features of Adaptive Filters</i>			
1.5 Array Processing	25		
<i>1.5.1 Spatial Filtering or Beamforming / 1.5.2 Adaptive Interference Mitigation in Radar Systems / 1.5.3 Adaptive Sidelobe Canceler</i>			
1.6 Organization of the Book	29		
2 Fundamentals of Discrete-Time Signal Processing	33		
2.1 Discrete-Time Signals	33		
<i>2.1.1 Continuous-Time, Discrete-Time, and Digital Signals / 2.1.2 Mathematical Description of Signals / 2.1.3 Real-World Signals</i>			
2.2 Transform-Domain Representation of Deterministic Signals	37		
<i>2.2.1 Fourier Transforms and Fourier Series / 2.2.2 Sampling of Continuous-Time Signals / 2.2.3 The Discrete Fourier Transform / 2.2.4 The z-Transform / 2.2.5 Representation of Narrowband Signals</i>			
2.3 Discrete-Time Systems	47		
<i>2.3.1 Analysis of Linear, Time-Invariant Systems / 2.3.2 Response to Periodic Inputs / 2.3.3 Correlation Analysis and Spectral Density</i>			
2.4 Minimum-Phase and System Invertibility	54		
<i>2.4.1 System Invertibility and Minimum-Phase Systems / 2.4.2 All-Pass Systems / 2.4.3 Minimum-Phase and All-Pass Decomposition / 2.4.4 Spectral Factorization</i>			
2.5 Lattice Filter Realizations	64		
<i>2.5.1 All-Zero Lattice Structures / 2.5.2 All-Pole Lattice Structures</i>			
2.6 Summary	70		
Problems	70		
3 Random Variables, Vectors, and Sequences	75		
3.1 Random Variables	75		
<i>3.1.1 Distribution and Density Functions / 3.1.2 Statistical Averages / 3.1.3 Some Useful Random Variables</i>			
3.2 Random Vectors	83		
<i>3.2.1 Definitions and Second-Order Moments / 3.2.2 Linear Transformations of Random Vectors / 3.2.3 Normal Random Vectors / 3.2.4 Sums of Independent Random Variables</i>			
3.3 Discrete-Time Stochastic Processes	97		
<i>3.3.1 Description Using Probability Functions / 3.3.2 Second-Order Statistical Description / 3.3.3 Stationarity /</i>			

<i>3.3.4 Ergodicity / 3.3.5 Random Signal Variability / 3.3.6 Frequency-Domain Description of Stationary Processes</i>	4.4 Pole-Zero Models 177
3.4 Linear Systems with Stationary Random Inputs 115	
<i>3.4.1 Time-Domain Analysis / 3.4.2 Frequency-Domain Analysis / 3.4.3 Random Signal Memory / 3.4.4 General Correlation Matrices / 3.4.5 Correlation Matrices from Random Processes</i>	
3.5 Whitening and Innovations Representation 125	
<i>3.5.1 Transformations Using Eigen-decomposition / 3.5.2 Transformations Using Triangular Decomposition / 3.5.3 The Discrete Karhunen-Loëve Transform</i>	
3.6 Principles of Estimation 133	
<i>3.6.1 Properties of Estimators / 3.6.2 Estimation of Mean / 3.6.3 Estimation of Variance</i>	
3.7 Summary 142	
Problems 143	
4 Linear Signal Models 149	
4.1 Introduction 149	
<i>4.1.1 Linear Nonparametric Signal Models / 4.1.2 Parametric Pole-Zero Signal Models / 4.1.3 Mixed Processes and the Wold Decomposition</i>	
4.2 All-Pole Models 156	
<i>4.2.1 Model Properties / 4.2.2 All-Pole Modeling and Linear Prediction / 4.2.3 Autoregressive Models / 4.2.4 Lower-Order Models</i>	
4.3 All-Zero Models 172	
<i>4.3.1 Model Properties / 4.3.2 Moving-Average Models / 4.3.3 Lower-Order Models</i>	
4.4 Pole-Zero Models 177	
<i>4.4.1 Model Properties / 4.4.2 Autoregressive Moving-Average Models / 4.4.3 The First-Order Pole-Zero Model 1: PZ(1,1) / 4.4.4 Summary and Dualities</i>	
4.5 Models with Poles on the Unit Circle 182	
4.6 Cepstrum of Pole-Zero Models 184	
<i>4.6.1 Pole-Zero Models / 4.6.2 All-Pole Models / 4.6.3 All-Zero Models</i>	
4.7 Summary 189	
Problems 189	
5 Nonparametric Power Spectrum Estimation 195	
5.1 Spectral Analysis of Deterministic Signals 196	
<i>5.1.1 Effect of Signal Sampling / 5.1.2 Windowing, Periodic Extension, and Extrapolation / 5.1.3 Effect of Spectrum Sampling / 5.1.4 Effects of Windowing: Leakage and Loss of Resolution / 5.1.5 Summary</i>	
5.2 Estimation of the Autocorrelation of Stationary Random Signals 209	
5.3 Estimation of the Power Spectrum of Stationary Random Signals 212	
<i>5.3.1 Power Spectrum Estimation Using the Periodogram / 5.3.2 Power Spectrum Estimation by Smoothing a Single Periodogram—The Blackman-Tukey Method / 5.3.3 Power Spectrum Estimation by Averaging Multiple Periodograms—The Welch-Bartlett Method / 5.3.4 Some Practical Considerations and Examples</i>	

5.4	Joint Signal Analysis	237	<i>Linear Prediction Using the Infinite Past—Whitening</i>
	<i>5.4.1 Estimation of Cross-Power Spectrum / 5.4.2 Estimation of Frequency Response Functions</i>		
5.5	Multitaper Power Spectrum Estimation	246	6.7 Inverse Filtering and Deconvolution 306
	<i>5.5.1 Estimation of Auto Power Spectrum / 5.5.2 Estimation of Cross Power Spectrum</i>		6.8 Channel Equalization in Data Transmission Systems 310
5.6	Summary	254	<i>6.8.1 Nyquist's Criterion for Zero ISI / 6.8.2 Equivalent Discrete-Time Channel Model / 6.8.3 Linear Equalizers / 6.8.4 Zero-Forcing Equalizers / 6.8.5 Minimum MSE Equalizers</i>
	Problems	255	
6	Optimum Linear Filters	261	6.9 Matched Filters and Eigenfilters 319
6.1	Optimum Signal Estimation	261	<i>6.9.1 Deterministic Signal in Noise / 6.9.2 Random Signal in Noise</i>
6.2	Linear Mean Square Error Estimation	264	6.10 Summary 325
	<i>6.2.1 Error Performance Surface / 6.2.2 Derivation of the Linear MMSE Estimator / 6.2.3 Principal-Component Analysis of the Optimum Linear Estimator / 6.2.4 Geometric Interpretations and the Principle of Orthogonality / 6.2.5 Summary and Further Properties</i>		Problems 325
6.3	Solution of the Normal Equations	274	
6.4	Optimum Finite Impulse Response Filters	278	
	<i>6.4.1 Design and Properties / 6.4.2 Optimum FIR Filters for Stationary Processes / 6.4.3 Frequency-Domain Interpretations</i>		
6.5	Linear Prediction	286	7 Algorithms and Structures for Optimum Linear Filters 333
	<i>6.5.1 Linear Signal Estimation / 6.5.2 Forward Linear Prediction / 6.5.3 Backward Linear Prediction / 6.5.4 Stationary Processes / 6.5.5 Properties</i>		
6.6	Optimum Infinite Impulse Response Filters	295	7.1 Fundamentals of Order-Recursive Algorithms 334
	<i>6.6.1 Noncausal IIR Filters / 6.6.2 Causal IIR Filters / 6.6.3 Filtering of Additive Noise / 6.6.4</i>		<i>7.1.1 Matrix Partitioning and Optimum Nesting / 7.1.2 Inversion of Partitioned Hermitian Matrices / 7.1.3 Levinson Recursion for the Optimum Estimator / 7.1.4 Order-Recursive Computation of the LDL^H Decomposition / 7.1.5 Order-Recursive Computation of the Optimum Estimate</i>
			7.2 Interpretations of Algorithmic Quantities 343
			<i>7.2.1 Innovations and Backward Prediction / 7.2.2 Partial Correlation / 7.2.3 Order Decomposition of the Optimum Estimate / 7.2.4 Gram-Schmidt Orthogonalization</i>
7.3	Order-Recursive Algorithms for Optimum FIR Filters	347	7.3.1 Order-Recursive Computation of the Optimum Filter / 7.3.2

<i>Lattice-Ladder Structure / 7.3.3</i>	8.4 Linear Least-Squares	
<i>Simplifications for Stationary</i>	<i>Signal Estimation</i>	411
<i>Stochastic Processes / 7.3.4</i>		
<i>Algorithms Based on the UDU^H</i>	<i>8.4.1 Signal Estimation and Linear</i>	
<i>Decomposition</i>	<i>Prediction / 8.4.2 Combined</i>	
7.4 Algorithms of Levinson	<i>Forward and Backward Linear</i>	
and Levinson-Durbin	<i>Prediction (FBLP) / 8.4.3</i>	
	<i>Narrowband Interference</i>	
7.5 Lattice Structures for	<i>Cancellation</i>	
Optimum FIR Filters		
and Predictors	8.5 LS Computations Using the	
	<i>Normal Equations</i>	416
<i>7.5.1 Lattice-Ladder Structures /</i>	<i>8.5.1 Linear LSE Estimation /</i>	
<i>7.5.2 Some Properties and</i>	<i>8.5.2 LSE FIR Filtering and</i>	
<i>Interpretations / 7.5.3 Parameter</i>	<i>Prediction</i>	
<i>Conversions</i>		
7.6 Algorithm of Schür	8.6 LS Computations Using	
	<i>Orthogonalization</i>	
<i>7.6.1 Direct Schür Algorithm /</i>	<i>Techniques</i>	422
<i>7.6.2 Implementation</i>	<i>8.6.1 Householder Reflections /</i>	
<i>Considerations / 7.6.3 Inverse</i>	<i>8.6.2 The Givens Rotations / 8.6.3</i>	
<i>Schür Algorithm</i>	<i>Gram-Schmidt Orthogonalization</i>	
7.7 Triangularization and Inversion	8.7 LS Computations Using	
of Toeplitz Matrices	<i>the Singular Value</i>	
	<i>Decomposition</i>	431
<i>7.7.1 LDL^H Decomposition of</i>	<i>8.7.1 Singular Value</i>	
<i>Inverse of a Toeplitz Matrix /</i>	<i>Decomposition / 8.7.2 Solution</i>	
<i>7.7.2 LDL^H Decomposition of a</i>	<i>of the LS Problem / 8.7.3</i>	
<i>Toeplitz Matrix / 7.7.3 Inversion</i>	<i>Rank-Deficient LS Problems</i>	
<i>of Real Toeplitz Matrices</i>		
7.8 Kalman Filter Algorithm	8.8 Summary	438
	<i>Problems</i>	439
<i>7.8.1 Preliminary Development /</i>		
<i>7.8.2 Development of Kalman Filter</i>		
7.9 Summary		
<i>Problems</i>		
8 Least-Squares Filtering	9 Signal Modeling	
and Prediction	and Parametric	
	Spectral Estimation	445
8.1 The Principle of Least	9.1 The Modeling Process:	
Squares	<i>Theory and Practice</i>	445
8.2 Linear Least-Squares	9.2 Estimation of All-Pole	
Error Estimation	<i>Models</i>	449
	<i>9.2.1 Direct Structures /</i>	
<i>8.2.1 Derivation of the Normal</i>	<i>9.2.2 Lattice Structures / 9.2.3</i>	
<i>Equations / 8.2.2 Statistical</i>	<i>Maximum Entropy Method / 9.2.4</i>	
<i>Properties of Least-Squares</i>	<i>Excitations with Line Spectra</i>	
<i>Estimators</i>		
8.3 Least-Squares FIR Filters	9.3 Estimation of Pole-Zero	
	<i>Models</i>	462
	<i>9.3.1 Known Excitation / 9.3.2</i>	
	<i>Unknown Excitation / 9.3.3</i>	

	<i>Nonlinear Least-Squares Optimization</i>	
9.4	Applicatons	467
	<i>9.4.1 Spectral Estimation / 9.4.2 Speech Modeling</i>	
9.5	Minimum-Variance Spectrum Estimation	471
9.6	Harmonic Models and Frequency Estimation Techniques	478
	<i>9.6.1 Harmonic Model / 9.6.2 Pisarenko Harmonic Decomposition / 9.6.3 MUSIC Algorithm / 9.6.4 Minimum-Norm Method / 9.6.5 ESPRIT Algorithm</i>	
9.7	Summary	493
	Problems	494
10 Adaptive Filters		499
10.1	Typical Applications of Adaptive Filters	500
	<i>10.1.1 Echo Cancelation in Communications / 10.1.2 Equalization of Data Communications Channels / 10.1.3 Linear Predictive Coding / 10.1.4 Noise Cancelation</i>	
10.2	Principles of Adaptive Filters	506
	<i>10.2.1 Features of Adaptive Filters / 10.2.2 Optimum versus Adaptive Filters / 10.2.3 Stability and Steady-State Performance of Adaptive Filters / 10.2.4 Some Practical Considerations</i>	
10.3	Method of Steepest Descent	516
10.4	Least-Mean-Square Adaptive Filters	524
	<i>10.4.1 Derivation / 10.4.2 Adaptation in a Stationary SOE / 10.4.3 Summary and Design Guidelines / 10.4.4 Applications of the LMS Algorithm / 10.4.5 Some Practical Considerations</i>	
10.5	Recursive Least-Squares Adaptive Filters	548
	<i>10.5.1 LS Adaptive Filters / 10.5.2 Conventional Recursive Least-Squares Algorithm / 10.5.3 Some Practical Considerations / 10.5.4 Convergence and Performance Analysis</i>	
10.6	RLS Algorithms for Array Processing	560
	<i>10.6.1 LS Computations Using the Cholesky and QR Decompositions / 10.6.2 Two Useful Lemmas / 10.6.3 The QR-RLS Algorithm / 10.6.4 Extended QR-RLS Algorithm / 10.6.5 The Inverse QR-RLS Algorithm / 10.6.6 Implementation of QR-RLS Algorithm Using the Givens Rotations / 10.6.7 Implementation of Inverse QR-RLS Algorithm Using the Givens Rotations / 10.6.8 Classification of RLS Algorithms for Array Processing</i>	
10.7	Fast RLS Algorithms for FIR Filtering	573
	<i>10.7.1 Fast Fixed-Order RLS FIR Filters / 10.7.2 RLS Lattice-Ladder Filters / 10.7.3 RLS Lattice-Ladder Filters Using Error Feedback Updatings / 10.7.4 Givens Rotation-Based LS Lattice-Ladder Algorithms / 10.7.5 Classification of RLS Algorithms for FIR Filtering</i>	
10.8	Tracking Performance of Adaptive Algorithms	590
	<i>10.8.1 Approaches for Nonstationary SOE / 10.8.2 Preliminaries in Performance Analysis / 10.8.3 The LMS Algorithm / 10.8.4 The RLS Algorithm with Exponential Forgetting / 10.8.5 Comparison of Tracking Performance</i>	
10.9	Summary	607
	Problems	608

11 Array Processing	621	11.8	Space-Time Adaptive Processing	683	
11.1	Array Fundamentals	622	11.9	Summary	685
	<i>11.1.1 Spatial Signals / 11.1.2 Modulation-Demodulation / 11.1.3 Array Signal Model / 11.1.4 The Sensor Array: Spatial Sampling</i>		Problems	686	
11.2	Conventional Spatial Filtering: Beamforming	631			
	<i>11.2.1 Spatial Matched Filter / 11.2.2 Tapered Beamforming</i>				
11.3	Optimum Array Processing	641	12.1	Higher-Order Statistics in Signal Processing	691
	<i>11.3.1 Optimum Beamforming / 11.3.2 Eigenanalysis of the Optimum Beamformer / 11.3.3 Interference Cancelation Performance / 11.3.4 Tapered Optimum Beamforming / 11.3.5 The Generalized Sidelobe Canceler</i>			<i>12.1.1 Moments, Cumulants, and Polyspectra / 12.1.2 Higher-Order Moments and LTI Systems / 12.1.3 Higher-Order Moments of Linear Signal Models</i>	
11.4	Performance Considerations for Optimum Beamformers	652	12.2	Blind Deconvolution	697
	<i>11.4.1 Effect of Signal Mismatch / 11.4.2 Effect of Bandwidth</i>		12.3	Unsupervised Adaptive Filters—Blind Equalizers	702
11.5	Adaptive Beamforming	659		<i>12.3.1 Blind Equalization / 12.3.2 Symbol Rate Blind Equalizers / 12.3.3 Constant-Modulus Algorithm</i>	
	<i>11.5.1 Sample Matrix Inversion / 11.5.2 Diagonal Loading with the SMI Beamformer / 11.5.3 Implementation of the SMI Beamformer / 11.5.4 Sample-by-Sample Adaptive Methods</i>		12.4	Fractionally Spaced Equalizers	709
11.6	Other Adaptive Array Processing Methods	671		<i>12.4.1 Zero-Forcing Fractionally Spaced Equalizers / 12.4.2 MMSE Fractionally Spaced Equalizers / 12.4.3 Blind Fractionally Spaced Equalizers</i>	
	<i>11.6.1 Linearly Constrained Minimum-Variance Beamformers / 11.6.2 Partially Adaptive Arrays / 11.6.3 Sidelobe Cancelers</i>		12.5	Fractional Pole-Zero Signal Models	716
11.7	Angle Estimation	678		<i>12.5.1 Fractional Unit-Pole Model / 12.5.2 Fractional Pole-Zero Models: FPZ (p, d, q) / 12.5.3 Symmetric α-Stable Fractional Pole-Zero Processes</i>	
	<i>11.7.1 Maximum-Likelihood Angle Estimation / 11.7.2 Cramér-Rao Lower Bound on Angle Accuracy / 11.7.3 Beamsplitting Algorithms / 11.7.4 Model-Based Methods</i>		12.6	Self-Similar Random Signal Models	725
				<i>12.6.1 Self-Similar Stochastic Processes / 12.6.2 Fractional Brownian Motion / 12.6.3 Fractional Gaussian Noise / 12.6.4 Simulation of Fractional Brownian Motions and Fractional Gaussian Noises / 12.6.5 Estimation of Long Memory /</i>	

	<i>12.6.6 Fractional Lévy Stable Motion</i>	
12.7	Summary	741
	Problems	742
Appendix A Matrix Inversion Lemma		745
Appendix B Gradients and Optimization in Complex Space		747
B.1	Gradient	747
B.2	Lagrange Multipliers	749
Appendix C MATLAB Functions		753
Appendix D Useful Results from Matrix Algebra		755
D.1	Complex-Valued Vector Space <i>Some Definitions</i>	755
	<i>12.6.6 Fractional Lévy Stable Motion</i>	
D.2	Matrices	756
	<i>D.2.1 Some Definitions / D.2.2 Properties of Square Matrices</i>	
D.3	Determinant of a Square Matrix	760
	<i>D.3.1 Properties of the Determinant / D.3.2 Condition Number</i>	
D.4	Unitary Matrices	762
	<i>D.4.1 Hermitian Forms after Unitary Transformations / D.4.2 Significant Integral of Quadratic and Hermitian Forms</i>	
D.5	Positive Definite Matrices	764
Appendix E Minimum Phase Test for Polynomials		767
Bibliography		769
Index		787

*One must learn by doing the thing;
for though you think you know it
You have no certainty, until you try.*

—Sophocles, *Trachiniae*

PREFACE

The principal goal of this book is to provide a unified introduction to the theory, implementation, and applications of statistical and adaptive signal processing methods. We have focused on the key topics of spectral estimation, signal modeling, adaptive filtering, and array processing, whose selection was based on the grounds of theoretical value and practical importance. The book has been primarily written with students and instructors in mind. The principal objectives are to provide an introduction to basic concepts and methodologies that can provide the foundation for further study, research, and application to new problems. To achieve these goals, we have focused on topics that we consider fundamental and have either multiple or important applications.

APPROACH AND PREREQUISITES

The adopted approach is intended to help both students and practicing engineers understand the fundamental mathematical principles underlying the operation of a method, appreciate its inherent limitations, and provide sufficient details for its practical implementation. The academic flavor of this book has been influenced by our teaching whereas its practical character has been shaped by our research and development activities in both academia and industry. The mathematical treatment throughout this book has been kept at a level that is within the grasp of upper-level undergraduate students, graduate students, and practicing electrical engineers with a background in digital signal processing, probability theory, and linear algebra.

ORGANIZATION OF THE BOOK

Chapter 1 introduces the basic concepts and applications of statistical and adaptive signal processing and provides an overview of the book. Chapters 2 and 3 review the fundamentals of discrete-time signal processing, study random vectors and sequences in the time and frequency domains, and introduce some basic concepts of estimation theory. Chapter 4 provides a treatment of parametric linear signal models (both deterministic and stochastic) in the time and frequency domains. Chapter 5 presents the most practical methods for the estimation of correlation and spectral densities. Chapter 6 provides a detailed study of the theoretical properties of optimum filters, assuming that the relevant signals can be modeled as stochastic processes with known statistical properties; and Chapter 7 contains algorithms and structures for optimum filtering, signal modeling, and prediction. Chapter

8 introduces the principle of least-squares estimation and its application to the design of practical filters and predictors. Chapters 9, 10, and 11 use the theoretical work in Chapters 4, 6, and 7 and the practical methods in Chapter 8, to develop, evaluate, and apply practical techniques for signal modeling, adaptive filtering, and array processing. Finally, Chapter 12 introduces some advanced topics: definition and properties of higher-order moments, blind deconvolution and equalization, and stochastic fractional and fractal signal models with long memory. Appendix A contains a review of the matrix inversion lemma, Appendix B reviews optimization in complex space, Appendix C contains a list of the MATLAB functions used throughout the book, Appendix D provides a review of useful results from matrix algebra, and Appendix E includes a proof for the minimum-phase condition for polynomials.

THEORY AND PRACTICE

It is our belief that sound theoretical understanding goes hand-in-hand with practical implementation and application to real-world problems. Therefore, the book includes a large number of computer experiments that illustrate important concepts and help the reader to easily implement the various methods. Every chapter includes examples, problems, and computer experiments that facilitate the comprehension of the material. To help the reader understand the theoretical basis and limitations of the various methods and apply them to real-world problems, we provide MATLAB functions for all major algorithms and examples illustrating their use. The MATLAB files and additional material about the book can be found at <http://www.artechhouse.com/default.asp?frame=Static/manolakismatlab.html>. A Solutions Manual with detailed solutions to all the problems is available to the instructors adopting the book for classroom use.

Dimitris G. Manolakis
Vinay K. Ingle
Stephen M. Kogon

Introduction

This book is an introduction to the theory and algorithms used for the analysis and processing of random signals and their applications to real-world problems. The fundamental characteristic of random signals is captured in the following statement: Although random signals are evolving in time in an unpredictable manner, their average statistical properties exhibit considerable regularity. This provides the ground for the description of random signals using statistical averages instead of explicit equations. When we deal with random signals, the main objectives are the statistical description, modeling, and exploitation of the dependence between the values of one or more discrete-time signals and their application to theoretical and practical problems.

Random signals are described mathematically by using the theory of probability, random variables, and stochastic processes. However, in practice we deal with random signals by using statistical techniques. Within this framework we can develop, at least in principle, theoretically *optimum signal processing* methods that can inspire the development and can serve to evaluate the performance of practical *statistical signal processing* techniques. The area of *adaptive signal processing* involves the use of optimum and statistical signal processing techniques to design signal processing systems that can modify their characteristics, during normal operation (usually in real time), to achieve a clearly predefined application-dependent objective.

The purpose of this chapter is twofold: to illustrate the nature of random signals with some typical examples and to introduce the four major application areas treated in this book: *spectral estimation*, *signal modeling*, *adaptive filtering*, and *array processing*. Throughout the book, the emphasis is on the application of techniques to actual problems in which the theoretical framework provides a foundation to motivate the selection of a specific method.

1.1 RANDOM SIGNALS

A *discrete-time signal* or *time series* is a set of observations taken sequentially in time, space, or some other independent variable. Examples occur in various areas, including engineering, natural sciences, economics, social sciences, and medicine.

A discrete-time signal $x(n)$ is basically a sequence of real or complex numbers called samples. Although the integer index n may represent any physical variable (e.g., time, distance), we shall generally refer to it as *time*. Furthermore, in this book we consider only time series with observations occurring at equally spaced intervals of time.

Discrete-time signals can arise in several ways. Very often, a discrete-time signal is obtained by periodically sampling a continuous-time signal, that is, $x(n) = x_c(nT)$, where $T = 1/F_s$ (seconds) is the sampling period and F_s (samples per second or hertz) is the sampling frequency. At other times, the samples of a discrete-time signal are obtained

by accumulating some quantity (which does not have an instantaneous value) over equal intervals of time, for example, the number of cars per day traveling on a certain road. Finally, some signals are inherently discrete-time, for example, daily stock market prices. *Throughout the book, except if otherwise stated, the terms signal, time series, or sequence will be used to refer to a discrete-time signal.*

The key characteristics of a time series are that the observations are *ordered* in time and that adjacent observations are *dependent* (related). To see graphically the relation between the samples of a signal that are l sampling intervals away, we plot the points $\{x(n), x(n+l)\}$ for $0 \leq n \leq N - 1 - l$, where N is the length of the data record. The resulting graph is known as the *l lag scatter plot*. This is illustrated in Figure 1.1, which shows a speech signal and two scatter plots that demonstrate the correlation between successive samples. We note that for adjacent samples the data points fall close to a straight line with a positive slope. This implies high correlation because every sample is followed by a sample with about the same amplitude. In contrast, samples that are 20 sampling intervals apart are much less correlated because the points in the scatter plot are randomly spread.

When successive observations of the series are dependent, we may use past observations to predict future values. If the prediction is exact, the series is said to be *deterministic*. However, in most practical situations we cannot predict a time series exactly. Such time

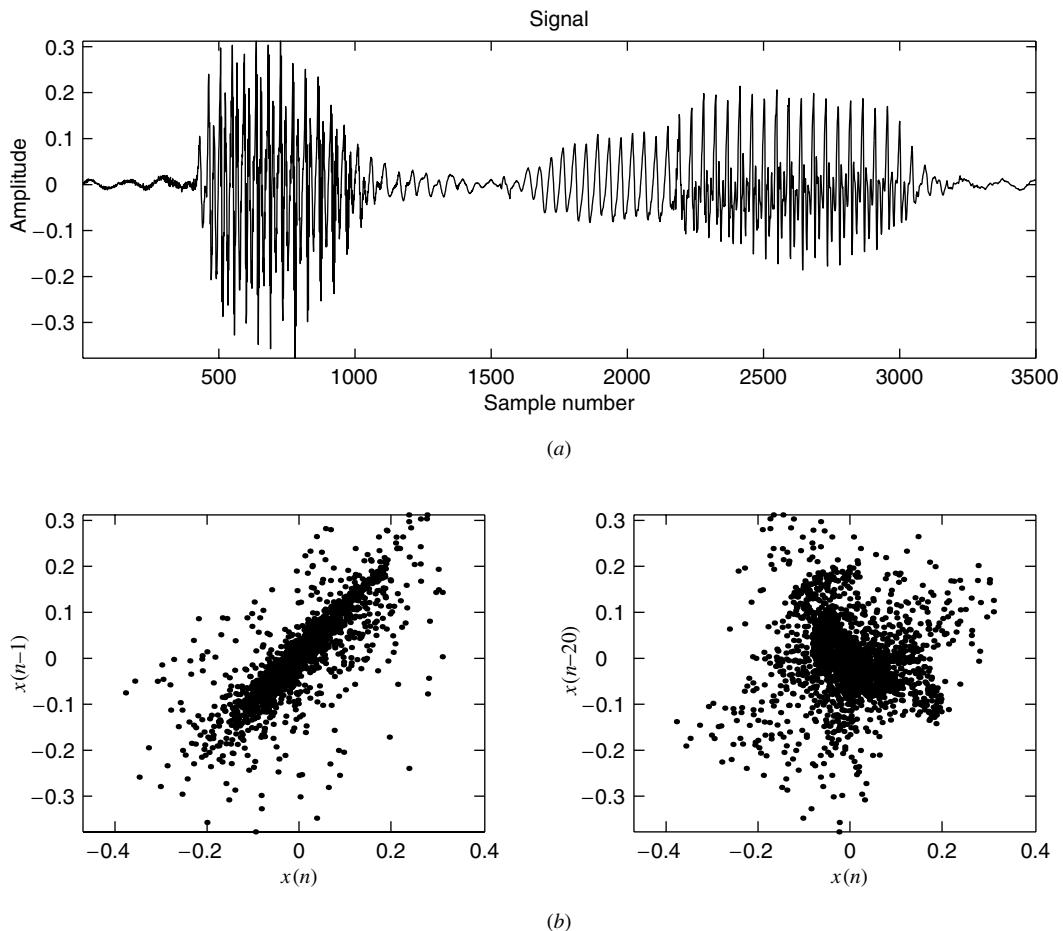


FIGURE 1.1

(a) The waveform for the speech signal “signal”; (b) two scatter plots for successive samples and samples separated by 20 sampling intervals.

series are called *random* or *stochastic*, and the degree of their predictability is determined by the dependence between consecutive observations. The ultimate case of randomness occurs when every sample of a random signal is independent of all other samples. Such a signal, which is completely unpredictable, is known as *white noise* and is used as a building block to simulate random signals with different types of dependence. To summarize, the fundamental characteristic of a random signal is the inability to precisely specify its values. In other words, a random signal is not predictable, it never repeats itself, and we cannot find a mathematical formula that provides its values as a function of time. As a result, random signals can only be mathematically described by using the theory of stochastic processes (see Chapter 3).

This book provides an introduction to the fundamental theory and a broad selection of algorithms widely used for the processing of discrete-time random signals. Signal processing techniques, dependent on their main objective, can be classified as follows (see Figure 1.2):

- **Signal analysis.** The primary goal is to extract useful information that can be used to understand the signal generation process or extract features that can be used for signal classification purposes. Most of the methods in this area are treated under the disciplines of *spectral estimation* and *signal modeling*. Typical applications include detection and classification of radar and sonar targets, speech and speaker recognition, detection and classification of natural and artificial seismic events, event detection and classification in biological and financial signals, efficient signal representation for data compression, etc.
- **Signal filtering.** The main objective of signal filtering is to improve the quality of a signal according to an acceptable criterion of performance. Signal filtering can be subdivided into the areas of frequency selective filtering, adaptive filtering, and array processing. Typical applications include noise and interference cancellation, echo cancellation, channel equalization, seismic deconvolution, active noise control, etc.

We conclude this section with some examples of signals occurring in practical applications. Although the description of these signals is far from complete, we provide sufficient information to illustrate their random nature and significance in signal processing applications.

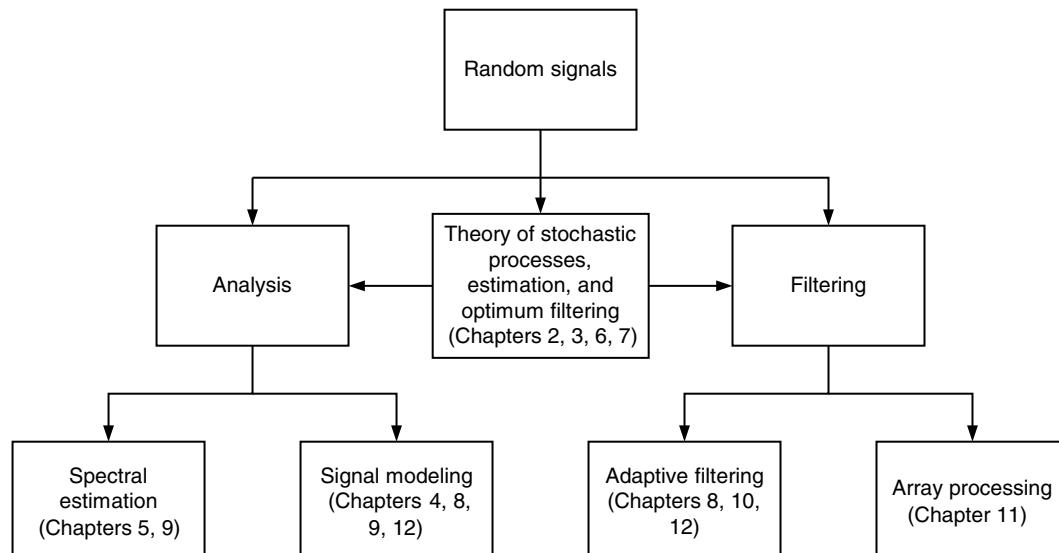


FIGURE 1.2

Classification of methods for the analysis and processing of random signals.

Speech signals. Figure 1.3 shows the spectrogram and speech waveform corresponding to the utterance “signal.” The spectrogram is a visual representation of the distribution of the signal energy as a function of time and frequency. We note that the speech signal has significant changes in both amplitude level and spectral content across time. The waveform contains segments of voiced (quasi-periodic) sounds, such as “e,” and unvoiced or fricative (noiselike) sounds, such as “g.”

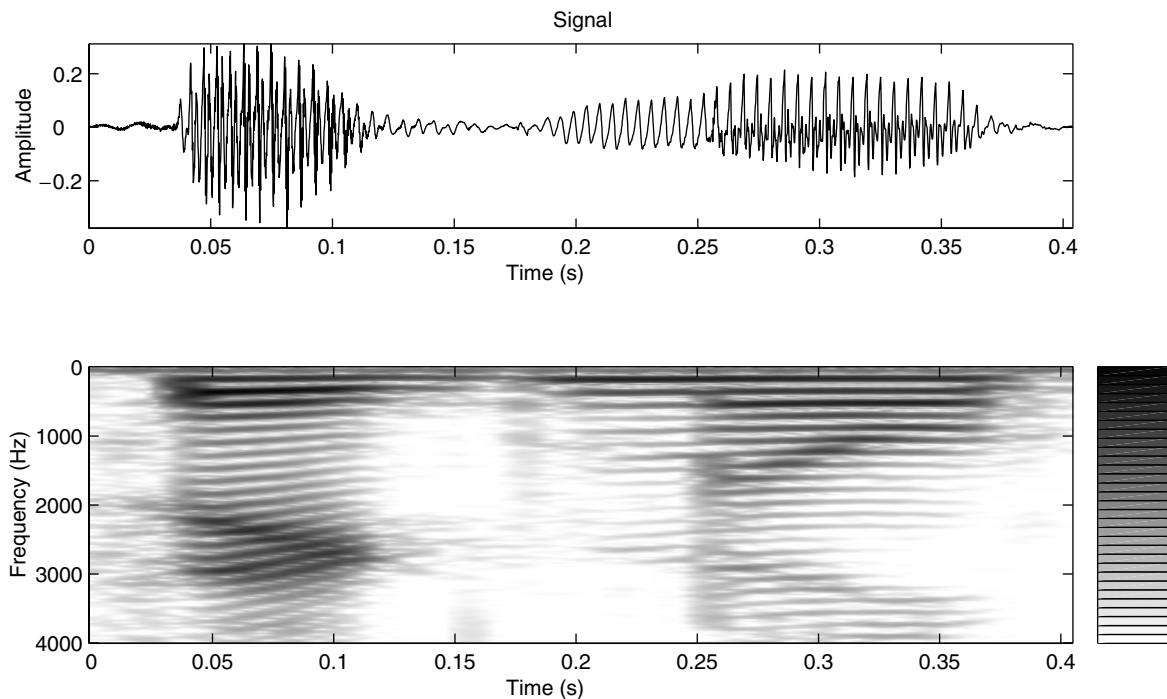


FIGURE 1.3

Spectrogram and acoustic waveform for the utterance “signal.” The horizontal dark bands show the resonances of the vocal tract, which change as a function of time depending on the sound or phoneme being produced.

Speech production involves three processes: generation of the sound excitation, articulation by the vocal tract, and radiation from the lips and/or nostrils. If the excitation is a quasi-periodic train of air pressure pulses, produced by the vibration of the vocal cords, the result is a voiced sound. Unvoiced sounds are produced by first creating a constriction in the vocal tract, usually toward the mouth end. Then we generate turbulence by forcing air through the constriction at a sufficiently high velocity. The resulting excitation is a broadband noiselike waveform.

The spectrum of the excitation is shaped by the vocal tract tube, which has a frequency response that resembles the resonances of organ pipes or wind instruments. The resonant frequencies of the vocal tract tube are known as *formant frequencies*, or simply *formants*. Changing the shape of the vocal tract changes its frequency response and results in the generation of different sounds. Since the shape of the vocal tract changes slowly during continuous speech, we usually assume that it remains almost constant over intervals on the order of 10 ms. More details about speech signal generation and processing can be found in Rabiner and Schafer 1978; O’Shaughnessy 1987; and Rabiner and Juang 1993.

Electrophysiological signals. Electrophysiology was established in the late eighteenth century when Galvani demonstrated the presence of electricity in animal tissues. Today, electrophysiological signals play a prominent role in every branch of physiology, medicine, and

biology. Figure 1.4 shows a set of typical signals recorded in a sleep laboratory (Rechtschaffen and Kales 1968). The most prominent among them is the electroencephalogram (EEG), whose spectral content changes to reflect the state of alertness and the mental activity of the subject. The EEG signal exhibits some distinctive waves, known as rhythms, whose dominant spectral content occupies certain bands as follows: delta (δ), 0.5 to 4 Hz; theta (θ), 4 to 8 Hz; alpha (α), 8 to 13 Hz; beta (β), 13 to 22 Hz; and gamma (γ), 22 to 30 Hz. During sleep, if the subject is dreaming, the EEG signal shows rapid low-amplitude fluctuations similar to those obtained in alert subjects, and this is known as rapid eye movement (REM) sleep. Some other interesting features occurring during nondreaming sleep periods resemble alphas-like activity and are known as sleep spindles. More details can be found in Duffy et al. 1989 and Niedermeyer and Lopes Da Silva 1998.

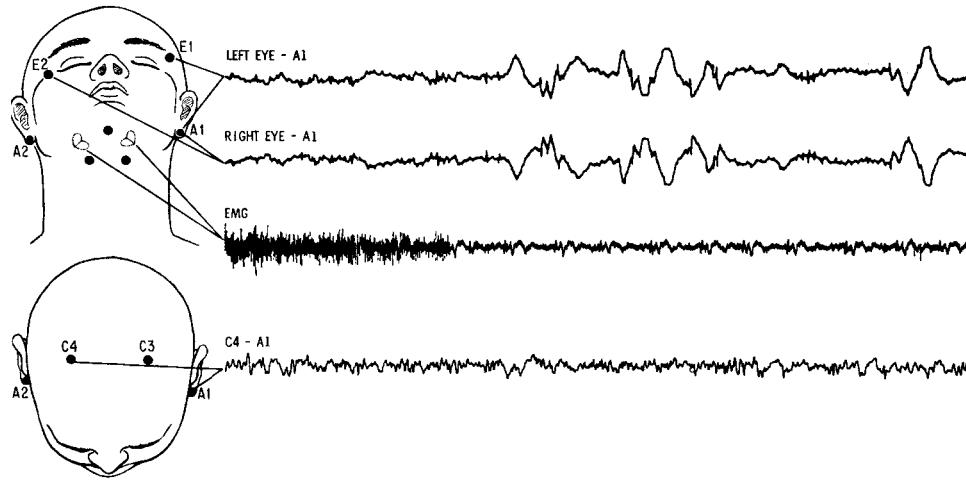
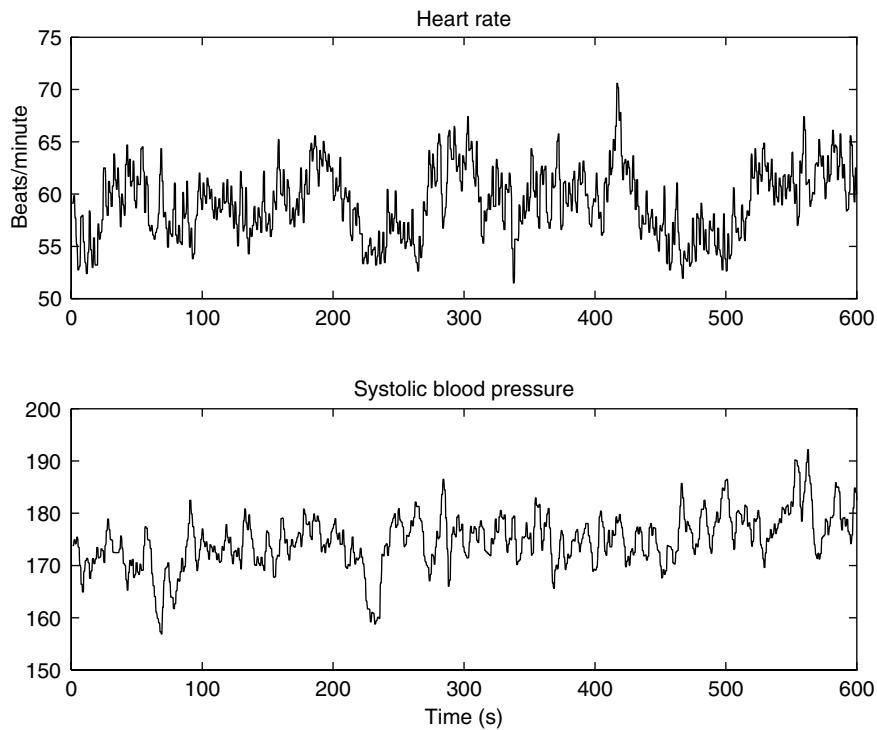


FIGURE 1.4

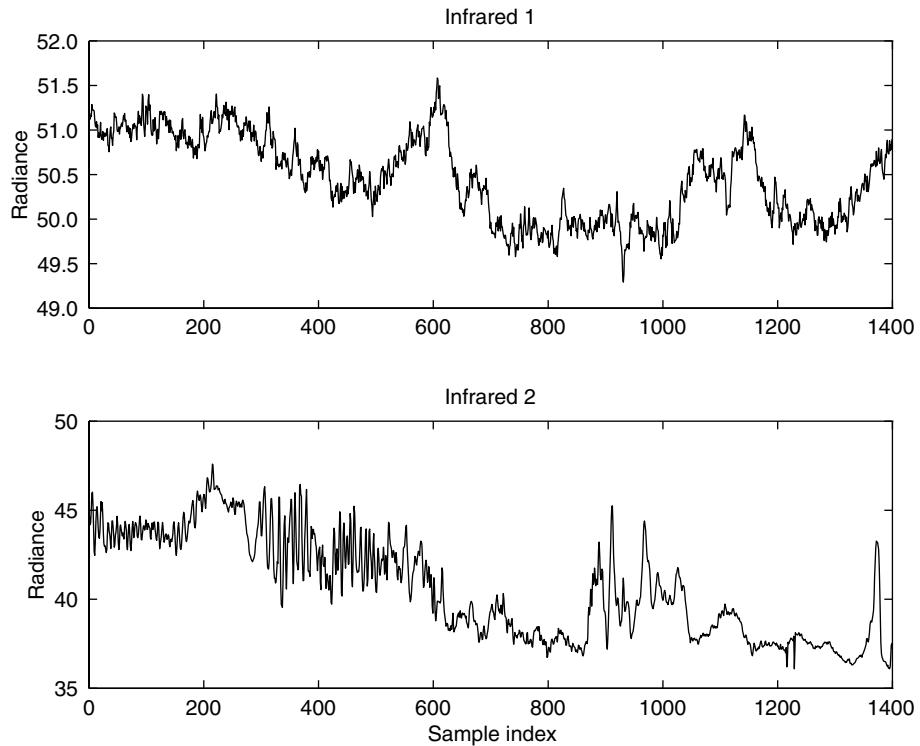
Typical sleep laboratory recordings. The two top signals show eye movements, the next one illustrates EMG (electromyogram) or muscle tonus, and the last one illustrates brain waves (EEG) during the onset of a REM sleep period (*from Rechtschaffen and Kales 1968*).

The beat-to-beat fluctuations in heart rate and other cardiovascular variables, such as arterial blood pressure and stroke volume, are mediated by the joint activity of the sympathetic and parasympathetic systems. Figure 1.5 shows time series for the heart rate and systolic arterial blood pressure. We note that both heart rate and blood pressure fluctuate in a complex manner that depends on the mental or physiological state of the subject. The individual or joint analysis of such time series can help to understand the operation of the cardiovascular system, predict cardiovascular diseases, and help in the development of drugs and devices for cardiac-related problems (Grossman et al. 1996; Malik and Camm 1995; Saul 1990).

Geophysical signals. Remote sensing systems use a variety of electro-optical sensors that span the infrared, visible, and ultraviolet regions of the spectrum and find many civilian and defense applications. Figure 1.6 shows two segments of infrared scans obtained by a space-based radiometer looking down at earth (Manolakis et al. 1994). The shape of the profiles depends on the transmission properties of the atmosphere and the objects in the radiometer's field-of-view (terrain or sky background). The statistical characterization and modeling of infrared backgrounds are critical for the design of systems to detect missiles against such backgrounds as earth's limb, auroras, and deep-space star fields (Sabins 1987; Colwell 1983). Other geophysical signals of interest are recordings of natural and man-made seismic events and seismic signals used in geophysical prospecting (Bolt 1993; Dobrin 1988; Sheriff 1994).

**FIGURE 1.5**

Simultaneous recordings of the heart rate and systolic blood pressure signals for a subject at rest.

**FIGURE 1.6**

Time series of infrared radiation measurements obtained by a scanning radiometer.

Radar signals. We conveniently define a radar system to consist of both a transmitter and a receiver. When the transmitter and receiver are colocated, the radar system is said to be monostatic, whereas if they are spatially separated, the system is bistatic. The radar first transmits a waveform, which propagates through space as electromagnetic energy, and then measures the energy returned to the radar via reflections. When the returns are due to an object of interest, the signal is known as a *target*, while undesired reflections from the earth's surface are referred to as *clutter*. In addition, the radar may encounter energy transmitted by a hostile opponent attempting to *jam* the radar and prevent detection of certain targets. Collectively, clutter and jamming signals are referred to as *interference*. The challenge facing the radar system is how to extract the targets of interest in the presence of sometimes severe interference environments. Target detection is accomplished by using adaptive processing methods that exploit characteristics of the interference in order to suppress these undesired signals.

A transmitted radar signal propagates through space as electromagnetic energy at approximately the speed of light $c = 3 \times 10^8$ m/s. The signal travels until it encounters an object that reflects the signal's energy. A portion of the reflected energy returns to the radar receiver along the same path. The round-trip delay of the reflected signal determines the distance or *range* of the object from the radar. The radar has a certain receive aperture, either a continuous aperture or one made up of a series of sensors. The relative delay of a signal as it propagates across the radar aperture determines its angle of arrival, or bearing. The extent of the aperture determines the accuracy to which the radar can determine the direction of a target. Typically, the radar transmits a series of pulses at a rate known as the *pulse repetition frequency*. Any target motion produces a phase shift in the returns from successive pulses caused by the *Doppler effect*. This phase shift across the series of pulses is known as the Doppler frequency of the target, which in turn determines the target radial velocity. The collection of these various parameters (range, angle, and velocity) allows the radar to locate and track a target.

An example of a radar signal as a function of range in kilometers (km) is shown in Figure 1.7. The signal is made up of a target, clutter, and thermal noise. All the signals have been normalized with respect to the thermal noise floor. Therefore, the normalized noise has unit variance (0 dB). The target signal is at a range of 100 km with a signal-to-noise ratio (SNR) of 15 dB. The clutter, on the other hand, is present at all ranges and is highly nonstationary. Its power levels vary from approximately 40 dB at near ranges down to the thermal noise floor (0 dB) at far ranges. Part of the nonstationarity in the clutter is due to the range falloff of the clutter as its power is attenuated as a function of range. However, the rises and dips present between 100 and 200 km are due to terrain-specific artifacts. Clearly, the target is not visible, and the clutter interference must be removed or canceled in order

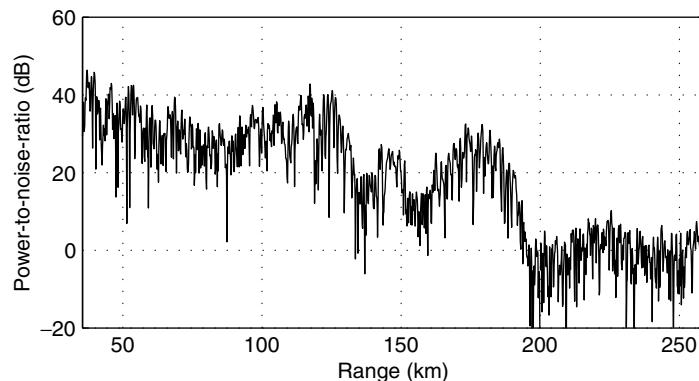


FIGURE 1.7
Example of a radar return signal, plotted as relative power with respect to noise versus range.

to detect the target. The challenge here is how to cancel such a nonstationary signal in order to extract the target signal and motivate the use of adaptive techniques that can adapt to the rapidly changing interference environment. More details about radar and radar signal processing can be found in Skolnik 1980; Skolnik 1990; and Nathanson 1991.

1.2 SPECTRAL ESTIMATION

The central objective of signal analysis is the development of quantitative techniques to study the properties of a signal and the differences and similarities between two or more signals from the same or different sources. The major areas of random signal analysis are (1) statistical analysis of signal amplitude (i.e., the sample values); (2) analysis and modeling of the correlation among the samples of an individual signal; and (3) joint signal analysis (i.e., simultaneous analysis of two signals in order to investigate their interaction or interrelationships). These techniques are summarized in Figure 1.8. The prominent tool in signal analysis is spectral estimation, which is a generic term for a multitude of techniques used to estimate the distribution of energy or power of a signal from a set of observations. Spectral estimation is a very complicated process that requires a deep understanding of the underlying theory and a great deal of practical experience. Spectral analysis finds many applications in areas such as medical diagnosis, speech analysis, seismology and geophysics, radar and sonar, nondestructive fault detection, testing of physical theories, and evaluating the predictability of time series.

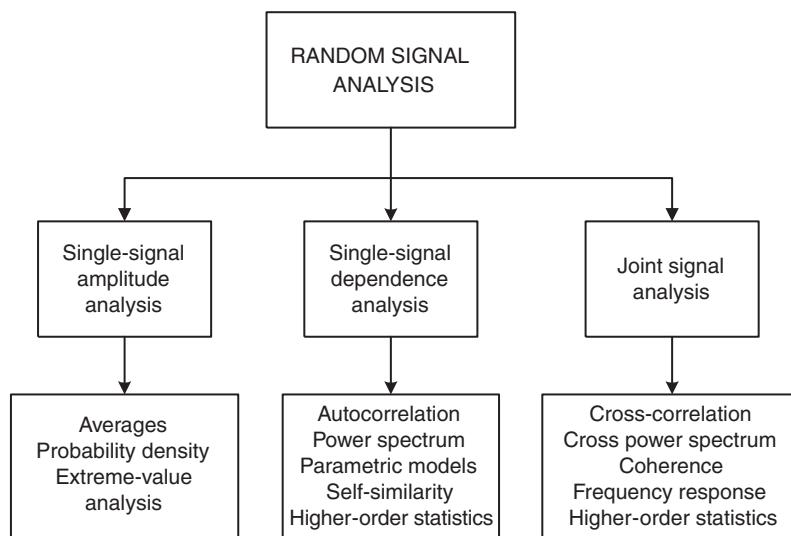


FIGURE 1.8

Summary of random signal analysis techniques.

Amplitude distribution. The range of values taken by the samples of a signal and how often the signal assumes these values together determine the signal variability. The signal variability can be seen by plotting the time series and is quantified by the histogram of the signal samples, which shows the percentage of the signal amplitude values within a certain range. The numerical description of signal variability, which depends only on the value of the signal samples and not on their ordering, involves quantities such as mean value, median, variance, and dynamic range.

Figure 1.9 shows the one-step increments, that is, the first difference $x_d(n) = x(n) - x(n-1)$, or approximate derivative of the infrared signals shown in Figure 1.6, whereas Figure 1.10 shows their histograms. Careful examination of the shape of the histogram curves indicates that the second signal jumps quite frequently between consecutive samples with large steps. In other words, the probability of large increments is significant, as exemplified

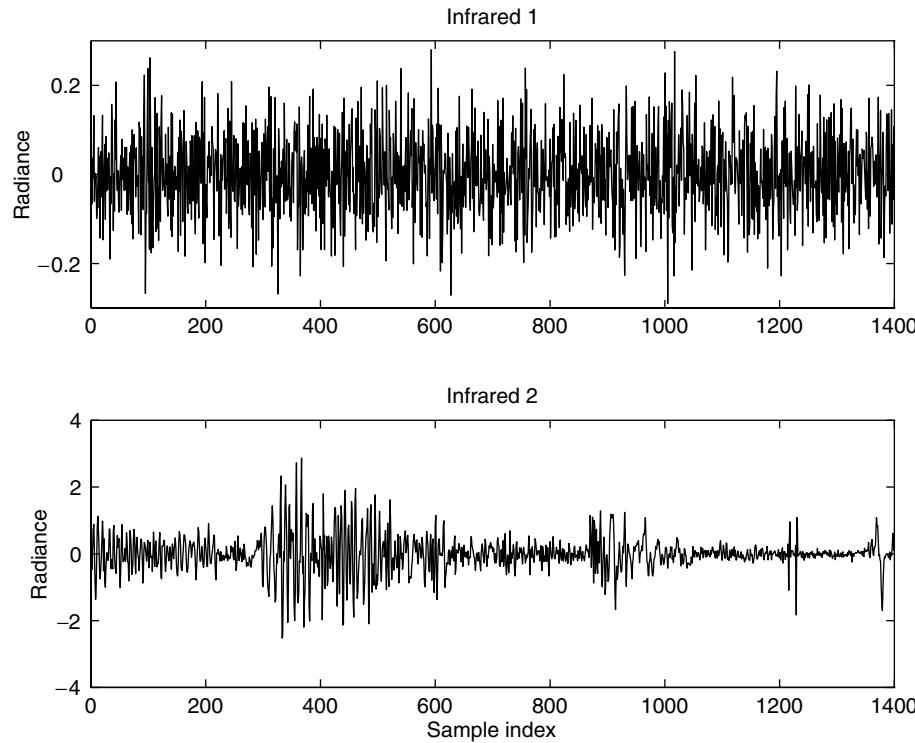


FIGURE 1.9

One-step-increment time series for the infrared data shown in Figure 1.6.

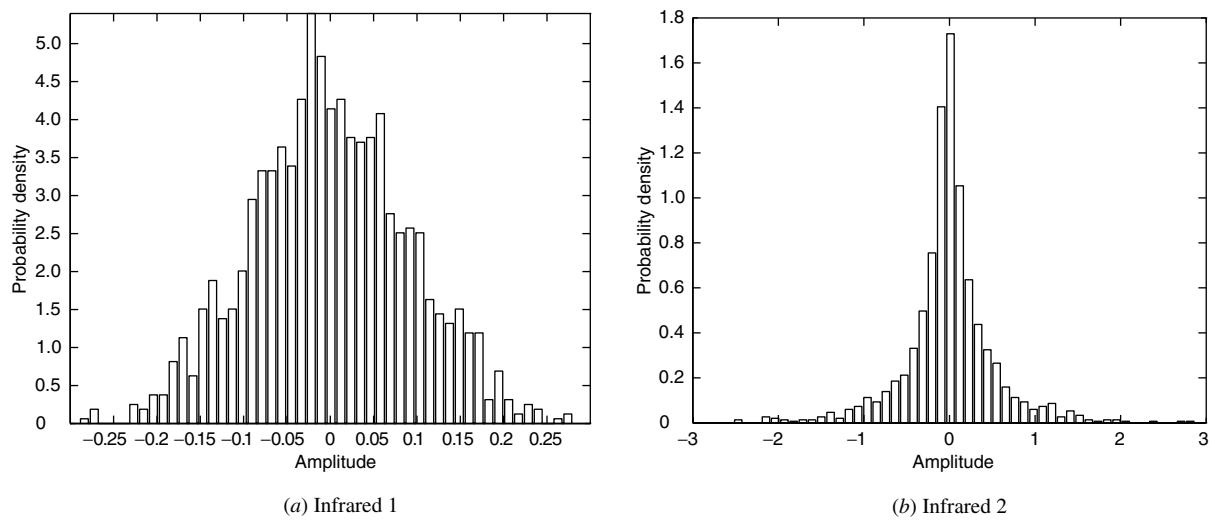


FIGURE 1.10

Histograms for the infrared increment signals.

by the fat tails of the histogram in Figure 1.10(b). The knowledge of the probability of extreme values is essential in the design of detection systems for digital communications, military surveillance using infrared and radar sensors, and intensive care monitoring. In general, the shape of the histogram, or more precisely the probability density, is very important in applications such as signal coding and event detection. Although many practical signals follow a Gaussian distribution, many other signals of practical interest have distributions that are non-Gaussian. For example, speech signals have a probability density that can be reasonably approximated by a gamma distribution (Rabiner and Schafer 1978).

The significance of the Gaussian distribution in signal processing stems from the following facts. First, many physical signals can be described by Gaussian processes. Second, the central limit theorem (see Chapter 3) states that any process that is the result of the combination of many elementary processes will tend, under quite general conditions, to be Gaussian. Finally, linear systems preserve the Gaussianity of their input signals. To understand the last two statements, consider N independent random quantities x_1, x_2, \dots, x_N with the same probability density $p(x)$ and pose the following question: When does the probability distribution $p_N(x)$ of their sum $x = x_1 + x_2 + \dots + x_N$ have the same shape (within a scale factor) as the distribution $p(x)$ of the individual quantities? The standard answer is that $p(x)$ should be Gaussian, because the sum of N Gaussian random variables is again a Gaussian, but with variance equal to N times that of the individual signals. However, if we allow for distributions with infinite variance, additional solutions are possible. The resulting probability distributions, known as *stable* or *Levy distributions*, have infinite variance and are characterized by a thin main lobe and fat tails, resembling the shape of the histogram in Figure 1.10(b). Interestingly enough, the Gaussian distribution is a stable distribution with finite variance (actually the only one). Because Gaussian and stable non-Gaussian distributions are invariant under linear signal processing operations, they are very important in signal processing.

Correlation and spectral analysis. Although scatter plots (see Figure 1.1) illustrate nicely the existence of correlation, to obtain quantitative information about the correlation structure of a time series $x(n)$ with zero mean value, we use the *empirical normalized autocorrelation sequence*

$$\hat{\rho}(l) = \frac{\sum_{n=l}^{N-1} x(n)x^*(n-l)}{\sum_{n=0}^{N-1} |x(n)|^2} \quad (1.2.1)$$

which is an estimate of the theoretical normalized autocorrelation sequence. For lag $l = 0$, the sequence is perfectly correlated with itself and we get the maximum value of 1. If the sequence does not change significantly from sample to sample, the correlation of the sequence with its shifted copies, though diminished, is still close to 1. Usually, the correlation decreases as the lag increases because distant samples become less and less dependent. Note that reordering the samples of a time series changes its autocorrelation but not its histogram.

We say that signals whose empirical autocorrelation decays fast, such as an exponential, have short-memory or short-range dependence. If the empirical autocorrelation decays very slowly, as a hyperbolic function does, we say that the signal has long-memory or long-range dependence. These concepts will be formulated in a theoretical framework in Chapter 3. Furthermore, we shall see in the next section that effective modeling of time series with short or long memory requires different types of models.

The spectral density function shows the distribution of signal power or energy as a function of frequency (see Figure 1.11). The autocorrelation and the spectral density of a signal form a Fourier transform pair and hence contain the same information. However, they present this information in different forms, and one can reveal information that cannot

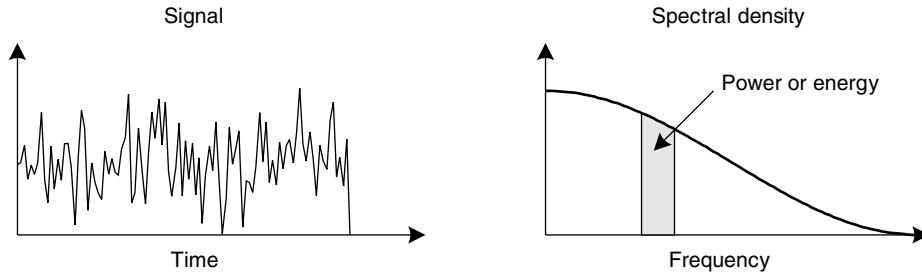
**FIGURE 1.11**

Illustration of the concept of power or energy spectral density function of a random signal.

be easily extracted from the other. It is fair to say that the spectral density is more widely used than the autocorrelation.

Although the correlation and spectral density functions are the most widely used tools for signal analysis, there are applications that require the use of correlations among three or more samples and the corresponding spectral densities. These quantities, which are useful when we deal with non-Gaussian processes and nonlinear systems, belong to the area of higher-order statistics and are described in Chapter 12.

Joint signal analysis. In many applications, we are interested in the relationship between two different random signals. There are two cases of interest. In the first case, the two signals are of the same or similar nature, and we want to ascertain and describe the similarity or interaction between them. For example, we may want to investigate if there is any similarity in the fluctuation of infrared radiation in the two profiles of Figure 1.6.

In the second case, we may have reason to believe that there is a *causal relationship* between the two signals. For example, one signal may be the input to a system and the other signal the output. The task in this case is to find an accurate description of the system, that is, a description that allows accurate estimation of future values of the output from the input. This process is known as *system modeling* or *identification* and has many practical applications, including understanding the operation of a system in order to improve the design of new systems or to achieve better control of existing systems.

In this book, we will study joint signal analysis techniques that can be used to understand the dynamic behavior between two or more signals. An interesting example involves using signals, like the ones in Figure 1.5, to see if there is any coupling between blood pressure and heart rate. Some interesting results regarding the effect of respiration and blood pressure on heart rate are discussed in Chapter 5.

1.3 SIGNAL MODELING

In many theoretical and practical applications, we are interested in generating random signals with certain properties or obtaining an efficient representation of real-world random signals that captures a desired set of their characteristics (e.g., correlation or spectral features) in the best possible way. We use the term *model* to refer to a mathematical description that provides an efficient representation of the “essential” properties of a signal.

For example, a finite segment $\{x(n)\}_{n=0}^{N-1}$ of any signal can be approximated by a linear combination of constant ($\lambda_k = 1$) or exponentially fading ($0 < \lambda_k < 1$) sinusoids

$$x(n) \simeq \sum_{k=1}^M a_k \lambda_k^n \cos(\omega_k n + \phi_k) \quad (1.3.1)$$

where $\{a_k, \lambda_k, \omega_k, \phi_k\}_{k=1}^M$ are the model parameters. A good model should provide an

accurate description of the signal with $4M \ll N$ parameters. From a practical viewpoint, we are most interested in *parametric models*, which assume a given functional form completely specified by a finite number of parameters. In contrast, *nonparametric models* do not put any restriction on the functional form or the number of model parameters.

If any of the model parameters in (1.3.1) is random, the result is a random signal. The most widely used model is given by

$$x(n) = \sum_{k=1}^M a_k \cos(\omega_k n + \phi_k)$$

where the amplitudes $\{a_k\}_1^N$ and the frequencies $\{\omega_k\}_1^N$ are constants and the phases $\{\phi_k\}_1^N$ are random. This model is known as the *harmonic process model* and has many theoretical and practical applications (see Chapters 3 and 9).

Suppose next that we are given a sequence $w(n)$ of independent and identically distributed observations. We can create a time series $x(n)$ with dependent observations, by linearly combining the values of $w(n)$ as

$$x(n) = \sum_{k=-\infty}^{\infty} h(k)w(n-k) \quad (1.3.2)$$

which results in the widely used *linear random signal model*. The model specified by the convolution summation (1.3.2) is clearly nonparametric because, in general, it depends on an infinite number of parameters. Furthermore, the model is a linear, time-invariant system with impulse response $h(k)$ that determines the *memory* of the model and, therefore, the dependence properties of the output $x(n)$. By properly choosing the weights $h(k)$, we can generate a time series with almost any type of dependence among its samples.

In practical applications, we are interested in linear parametric models. As we will see, parametric models exhibit a dependence imposed by their structure. However, if the number of parameters approaches the range of the dependence (in number of samples), the model can mimic any form of dependence. The list of desired features for a good model includes these: (1) the number of model parameters should be as small as possible (*parsimony*), (2) estimation of the model parameters from the data should be easy, and (3) the model parameters should have a physically meaningful interpretation.

If we can develop a successful parametric model for the behavior of a signal, then we can use the model for various applications:

1. To achieve a better understanding of the physical mechanism generating the signal (e.g., earth structure in the case of seismograms).
2. To track changes in the source of the signal and help identify their cause (e.g., EEG).
3. To synthesize artificial signals similar to the natural ones (e.g., speech, infrared backgrounds, natural scenes, data network traffic).
4. To extract parameters for pattern recognition applications (e.g., speech and character recognition).
5. To get an efficient representation of signals for data compression (e.g., speech, audio, and video coding).
6. To forecast future signal behavior (e.g., stock market indexes) (Pindyck and Rubinfeld 1998).

In practice, signal modeling involves the following steps: (1) selection of an appropriate model, (2) selection of the “right” number of parameters, (3) fitting of the model to the actual data, and (4) model testing to see if the model satisfies the user requirements for the particular application. As we shall see in Chapter 9, this process is very complicated and depends heavily on the understanding of the theoretical model properties (see Chapter 4), the amount of familiarity with the particular application, and the experience of the user.

1.3.1 Rational or Pole-Zero Models

13

SECTION 1.3
Signal Modeling

Suppose that a given sample $x(n)$, at time n , can be approximated by the previous sample weighted by a coefficient a , that is, $x(n) \approx ax(n - 1)$, where a is assumed constant over the signal segment to be modeled. To make the above relationship exact, we add an excitation term $w(n)$, resulting in

$$x(n) = ax(n - 1) + w(n) \quad (1.3.3)$$

where $w(n)$ is an excitation sequence. Taking the z -transform of both sides (discussed in Chapter 2), we have

$$X(z) = az^{-1}X(z) + W(z) \quad (1.3.4)$$

which results in the following system function:

$$H(z) = \frac{X(z)}{W(z)} = \frac{1}{1 - az^{-1}} \quad (1.3.5)$$

By using the identity

$$H(z) = \frac{1}{1 - az^{-1}} = 1 + az^{-1} + a^2z^{-2} + \dots \quad -1 < a < 1 \quad (1.3.6)$$

the single-parameter model in (1.3.3) can be expressed in the following nonparametric form

$$x(n) = w(n) + aw(n - 1) + a^2w(n - 2) + \dots \quad (1.3.7)$$

which clearly indicates that the model generates a time series with *exponentially decaying dependence*.

A more general model can be obtained by including a linear combination of the P previous values of the signal and of the Q previous values of the excitation in (1.3.3), that is,

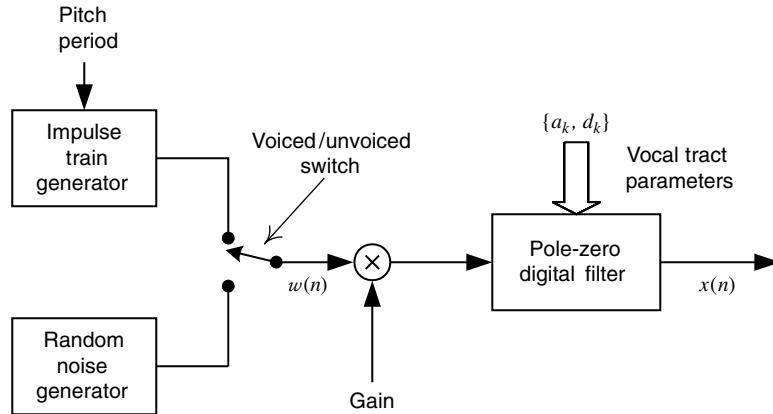
$$x(n) = \sum_{k=1}^P (-a_k)x(n - k) + \sum_{k=0}^Q d_k w(n - k) \quad (1.3.8)$$

The resulting system function

$$H(z) = \frac{X(z)}{W(z)} = \frac{\sum_{k=0}^Q d_k z^{-k}}{1 + \sum_{k=1}^P a_k z^{-k}} \quad (1.3.9)$$

is *rational*, that is, a ratio of two polynomials in the variable z^{-1} , hence the term *rational models*. We will show in Chapter 4 that *any* rational model has a dependence structure or memory that decays exponentially with time. Because the roots of the numerator polynomial are known as *zeros* and the roots of the denominator polynomial as *poles*, these models are also known as *pole-zero models*. In the time-series analysis literature, these models are known as *autoregressive moving-average (ARMA)* models.

Modeling the vocal tract. An example of the application of the pole-zero model is for the characterization of the speech production system. Most generally, speech sounds are classified as either voiced or unvoiced. For both of these types of speech, the production is modeled by exciting a linear system, the vocal tract, with an excitation having a flat, that is, constant, spectrum. The vocal tract, in turn, is modeled by using a pole-zero system, with the poles modeling the vocal tract resonances and the zeros serving the purpose of dampening the spectral response between pole frequencies. In the case of voiced speech, the input to the vocal tract model is a quasi-periodic pulse waveform, whereas for unvoiced speech the source is modeled as random noise. The system model of the speech production process is shown in Figure 1.12. The parameters of this model are the voiced/unvoiced

**FIGURE 1.12**

Speech synthesis system based on pole-zero modeling.

classification, the pitch period for voiced sounds, the gain parameter, and the coefficients $\{d_k\}$ and $\{a_k\}$ of the vocal tract filter (1.3.9). This model is widely used for low-bit-rate (less than 2.4 kbytes/s) speech coding, synthetic speech generation, and extraction of features for speech and speaker recognition (Rabiner and Schafer 1978; Rabiner and Juang 1993; Furui 1989).

1.3.2 Fractional Pole-Zero Models and Fractal Models

Although the dependence in (1.3.7) becomes stronger as the pole $a \rightarrow 1$, it cannot effectively model time series whose autocorrelation decays asymptotically as a power law. For $a = 1$, that is, for a pole on the unit circle (unit pole), we obtain an everlasting constant dependence, but the output of the model increases without limit and the model is said to be unstable. However, we can obtain a stable model with long memory by creating a *fractional unit pole*, that is, by raising (1.3.6) by a fractional power. Indeed, using the identity

$$H(z) = \frac{1}{(1 - z^{-1})^d} = 1 + dz^{-1} + \frac{d(d+1)}{2!}z^{-2} + \dots \quad -\frac{1}{2} < d < \frac{1}{2} \quad (1.3.10)$$

we have $x(n) = w(n) + dw(n-1) + \frac{d(d+1)}{2!}w(n-2) + \dots \quad (1.3.11)$

The weights $h_d(n)$ in (1.3.11) decay according to n^{d-1} as $n \rightarrow \infty$; that is, the dependence decays asymptotically as a power law or hyperbolically. Even if the model (1.3.11) is specified by one parameter, its implementation involves an infinite-order convolution summation. Therefore, its practical realization requires an approximation by a rational model that can be easily implemented by using a difference equation. If $w(n)$ is a sequence of independent Gaussian random variables, the process generated by (1.3.11) is known as *fractionally differenced Gaussian noise*. Rational models including one or more fractional poles are known in time-series analysis as *fractional autoregressive integrated moving-average* models and are studied in Chapter 12. The short-term dependence of these models is exponential, whereas their long-term dependence is hyperbolic.

In continuous time, we can create long dependence by using a fractional pole. This is illustrated by the following Laplace transform pair

$$\mathcal{L}\{t^{\beta-1}\} \propto \frac{1}{s^\beta} \quad \beta > 0 \quad (1.3.12)$$

which corresponds to an integrator for $\beta = 1$ and a *fractional integrator* for $0 < \beta < 1$. Clearly, the memory of a continuous-time system with impulse response $h_\beta(t) = t^{\beta-1}$ for

$t \geq 0$ and $h_\beta(t) = 0$ for $t < 0$ decays hyperbolically. The response of such a system to white Gaussian noise results in a nonstationary process called *fractional Brownian motion*. Sampling the fractional Brownian motion process at equal intervals and computing the one-step increments result in a stationary discrete-time process known as *fractional Gaussian noise*. Both processes exhibit long memory and are of great theoretical and practical interest and their properties and applications are discussed in Chapter 12.

Exciting a rational model with fractional Gaussian noise leads to a very flexible class of models that exhibit exponential short-range dependence and hyperbolic long-range dependence. The excitation of fractional models (either discrete-time or continuous-time) with statistically independent inputs whose amplitude changes are distributed according to a stable probability law leads to random signal models with long dependence and high amplitude variability. Such models have many practical applications and are also discussed in Chapter 12.

If we can reproduce an object by magnifying some portion of it, we say that the object is *scale-invariant* or *self-similar*. Thus, self-similarity is invariance with respect to scaling. Self-similar geometric objects are known as *fractals*. More specifically, a signal $x(t)$ is self-similar if $x(ct) = c^H x(t)$ for some $c > 0$. The constant H is known as the *self-similarity index*. It can easily be seen that a signal described by a power law, say, $x(t) = \alpha t^\beta$, is self-similar. However, such signals are of limited interest. A more interesting and useful type of signal is one that exhibits a weaker statistical version of self-similarity. A random signal is called (*statistically*) *self-similar* if its statistical properties are scale-invariant, that is, its statistics do not change under magnification or minification. Self-similar random signals are also known as *random fractals*. Figure 1.13 provides a visual illustration of the self-similar

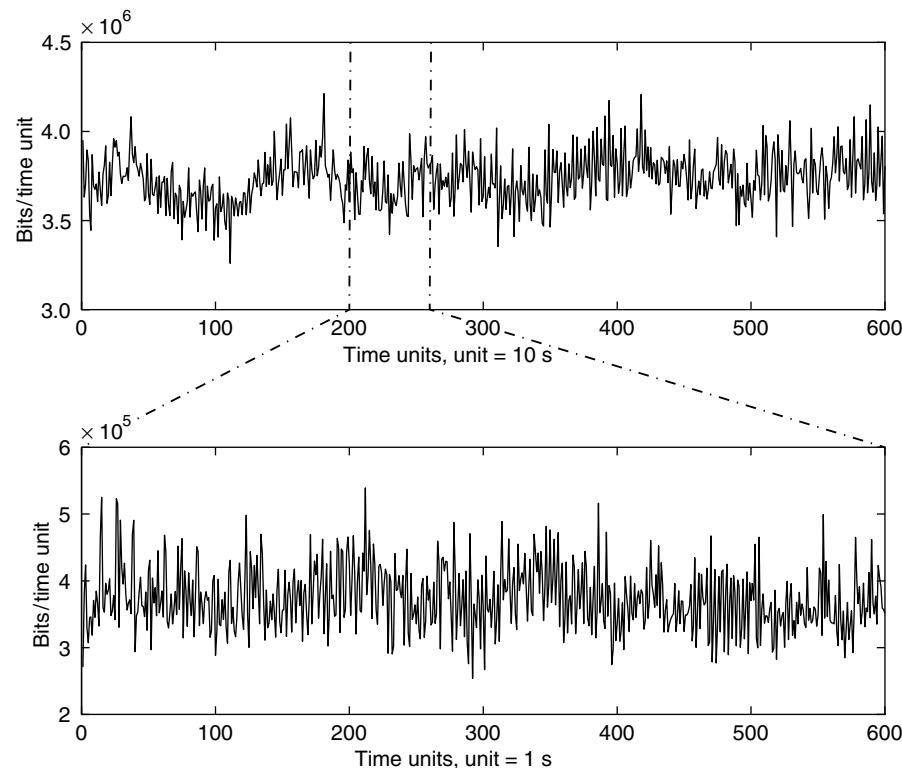


FIGURE 1.13

Pictorial illustration of self-similarity for the variable bit rate video traffic time series. The bottom series is obtained from the top series by expanding the segment between the two vertical lines. Although the two series have lengths of 600 and 60 s, they are remarkably similar visually and statistically (Courtesy of M. Garrett and M. Vetterli).

behavior of the variable bit rate video traffic time series. The analysis and modeling of such time series find extensive applications in Internet traffic applications (Michiel and Laevens 1997; Garrett and Willinger 1994).

A classification of the various signal models described previously is given in Figure 1.14, which also provides information about the chapters of the book where these signals are discussed.

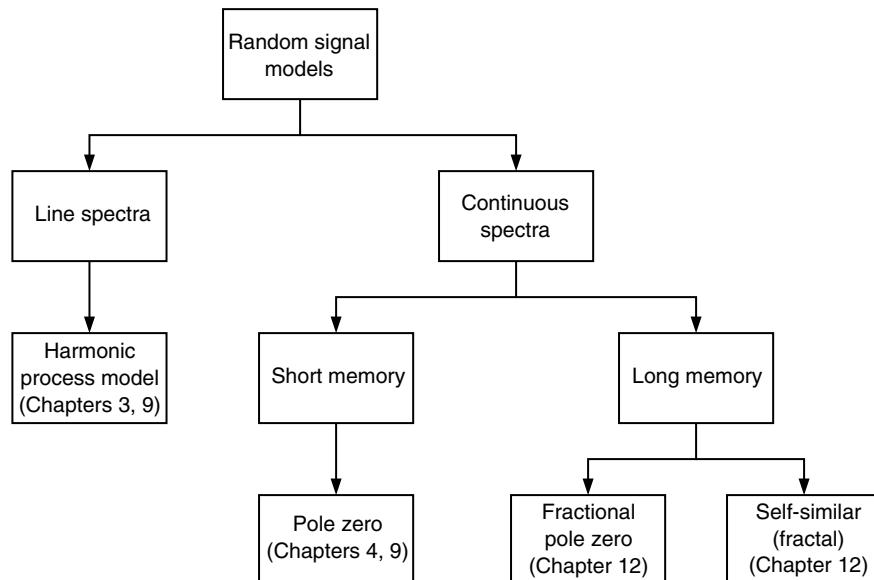


FIGURE 1.14

Classification of random signal models.

1.4 ADAPTIVE FILTERING

Conventional frequency-selective digital filters with *fixed* coefficients are designed to have a given frequency response chosen to alter the spectrum of the input signal in a desired manner. Their key features are as follows:

1. The filters are linear and time-invariant.
2. The design procedure uses the desired passband, transition bands, passband ripple, and stopband attenuation. We do *not* need to know the sample values of the signals to be processed.
3. Since the filters are frequency-selective, they work best when the various components of the input signal occupy nonoverlapping frequency bands. For example, it is easy to separate a signal and additive noise when their spectra do not overlap.
4. The filter coefficients are chosen during the design phase and are held constant during the normal operation of the filter.

However, there are many practical application problems that cannot be successfully solved by using fixed digital filters because either we do not have sufficient information to design a digital filter with fixed coefficients or the design criteria change during the normal operation of the filter. Most of these applications can be successfully solved by using special “smart” filters known collectively as *adaptive filters*. The distinguishing feature of adaptive filters is that they can modify their response to improve performance during operation without any intervention from the user.

1.4.1 Applications of Adaptive Filters

17

SECTION 1.4
Adaptive Filtering

The best way to introduce the concept of adaptive filtering is by describing some typical application problems that can be effectively solved by using an adaptive filter. The applications of adaptive filters can be sorted for convenience into four classes: (1) system identification, (2) system inversion, (3) signal prediction, and (4) multisensor interference cancelation (see Figure 1.15 and Table 1.1). We next describe each class of applications and provide a typical example for each case.

TABLE 1.1
Classification of adaptive filtering applications.

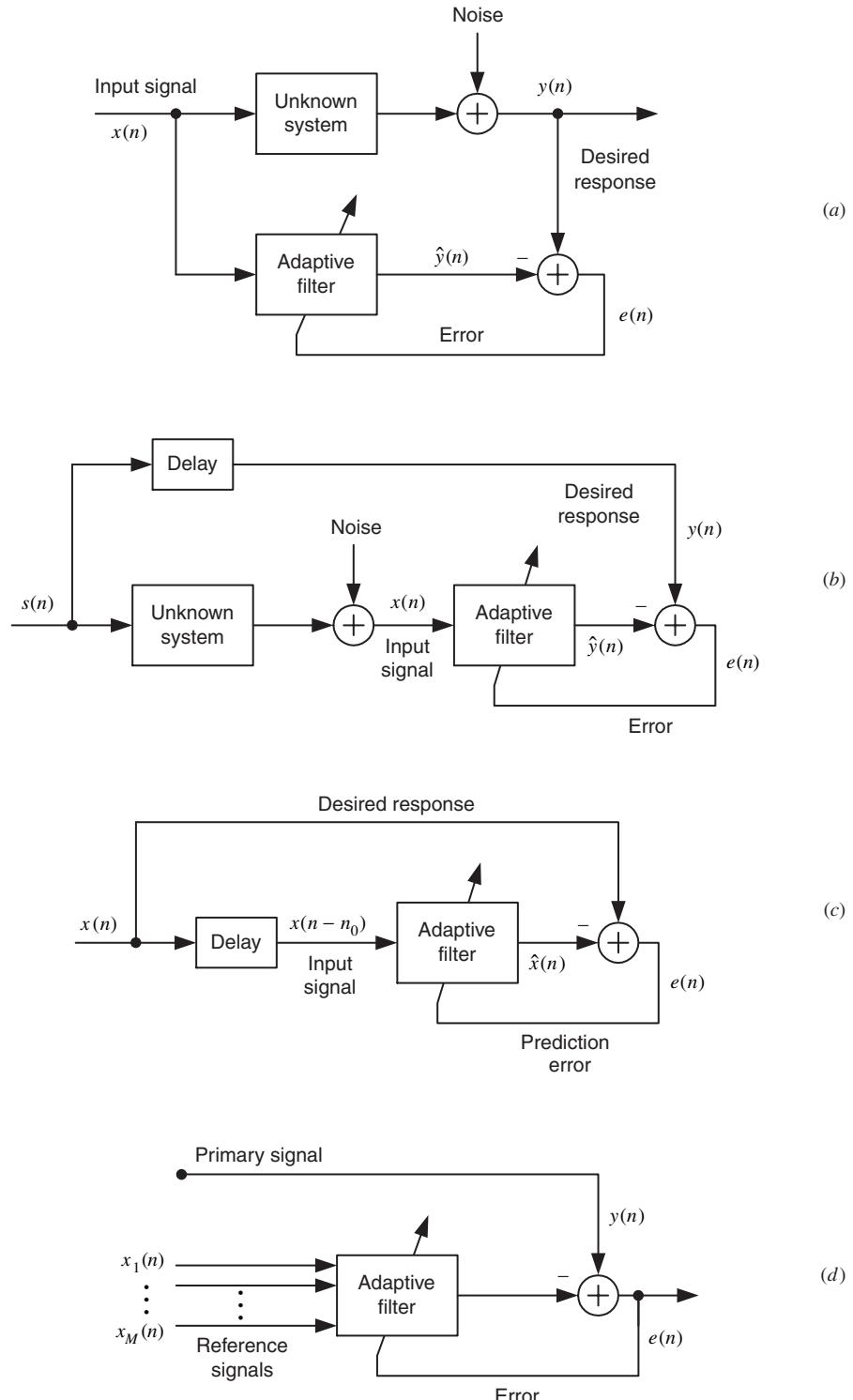
Application class	Examples
System identification	Echo cancellation Adaptive control Channel modeling
System inversion	Adaptive equalization Blind deconvolution
Signal prediction	Adaptive predictive coding Change detection Radio frequency interference cancelation
Multisensor interference cancelation	Acoustic noise control Adaptive beamforming

System Identification

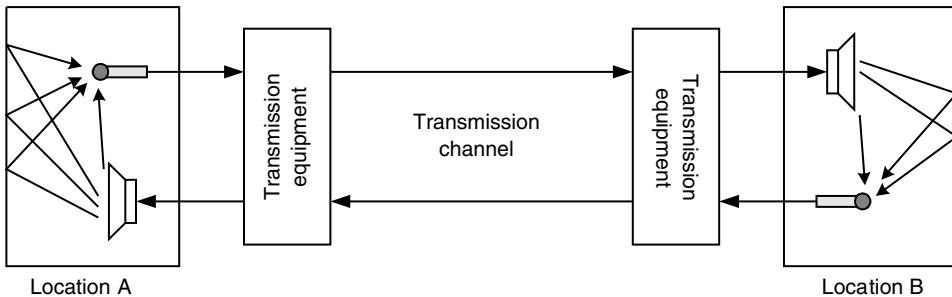
This class of applications, known also as system modeling, is illustrated in Figure 1.15(a). The system to be modeled can be either real, as in control system applications, or some hypothetical signal transmission path (e.g., the echo path). The distinguishing characteristic of the system identification application is that the input of the adaptive filter is noise-free and the desired response is corrupted by additive noise that is uncorrelated with the input signal. Applications in this class include echo cancellation, channel modeling, and identification of systems for control applications (Gitlin et al. 1992; Ljung 1987; Åström and Wittenmark 1990). In control applications, the purpose of the adaptive filter is to estimate the parameters or the state of the system and then to use this information to design a controller. In signal processing applications, the goal is to obtain a good estimate of the desired response according to the adopted criterion of performance.

Acoustic echo cancellation. Figure 1.16 shows a typical audio teleconferencing system that helps two groups of people, located at two different places, to communicate effectively. However, the performance of this system is degraded by the following effects: (1) The *reverberations* of the room result from the fact that the microphone picks up not only the speech coming from the talker but also reflections from the walls and furniture in the room. (2) *Echoes* are created by the acoustic coupling between the microphone and the loudspeaker located in the same room. Speech from room B not only is heard by the listener in room A but also is picked up by the microphone in room A, and unless it is prevented, will return as an echo to the speaker in room B.

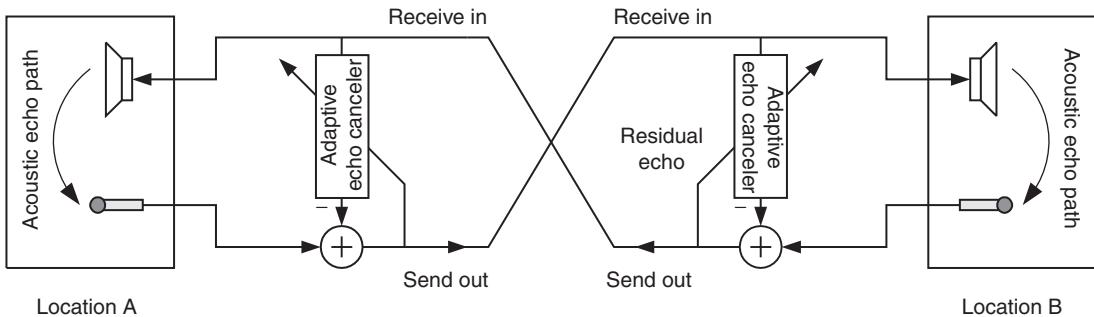
Several methods to deal with acoustic echoes have been developed. However, the most effective technique to prevent or control echoes is adaptive echo cancellation. The basic idea is very simple: To cancel the echo, we generate a replica or pseudo-echo and then subtract it from the real echo. To synthesize the echo replica, we pass the signal at the loudspeaker through a device designed to duplicate the reverberation and echo properties of the room (*echo path*), as is illustrated in Figure 1.17.

**FIGURE 1.15**

The four basic classes of adaptive filtering applications: (a) system identification, (b) system inversion, (c) signal prediction, and (d) multisensor interference cancelation.

**FIGURE 1.16**

Typical teleconferencing system without echo control.

**FIGURE 1.17**

Principle of acoustic echo cancellation using an adaptive echo canceler.

In practice, there are two obstacles to this approach. (1) The echo path is usually unknown before actual transmission begins and is quite complex to model. (2) The echo path is changing with time, since even the move of a talker alters the acoustic properties of the room. Therefore, we *cannot* design and use a *fixed* echo canceler with satisfactory performance for all possible connections. There are two possible ways around this problem:

1. Design a compromise *fixed* echo canceler based on some “average” echo path, assuming that we have sufficient information about the connections to be seen by the canceler.
2. Design an *adaptive echo canceler* that can “learn” the echo path when it is first turned on and afterward “tracks” its variations without any intervention from the designer. Since an adaptive canceler matches the echo patch for any given connection, it performs better than a fixed compromise canceler.

We stress that the main task of the canceler is to estimate the echo signal with sufficient accuracy; the estimation of the echo path is simply the means for achieving this goal. The performance of the canceler is measured by the attenuation of the echo. The adaptive echo canceler achieves this goal, by modifying its response, using the residual echo signal in an as-yet-unspecified way. More details about acoustic echo cancellation can be found in Gilloire et al. (1996).

System inversion

This class of applications, which is illustrated in Figure 1.15(b), is also known as inverse system modeling. The goal of the adaptive filter is to estimate and apply the inverse of the system. Dependent on the application, the input of the adaptive filter may be corrupted by additive noise, and the desired response may not be available. The existence of the inverse

system and its properties (e.g., causality and stability) creates additional complications. Typical applications include adaptive equalization (Gitlin et al. 1992), seismic deconvolution (Robinson 1984), and adaptive inverse control (Widrow and Walach 1994).

Channel equalization. To understand the basic principles of the channel equalization techniques, we consider a binary data communication system that transmits a band-limited analog pulse with amplitudes A (symbol 1) or $-A$ (symbol 0) every T_b s (see Figure 1.18). Here T_b is known as the *symbol interval* and $\mathcal{R}_b = 1/T_b$ as the *baud rate*. As the signal propagates through the channel, it is delayed and attenuated in a frequency-dependent manner. Furthermore, it is corrupted by additive noise and other natural or man-made interferences. The goal of the receiver is to measure the amplitude of each arriving pulse and to determine which one of the two possible pulses has been sent. The received signal is sampled once per symbol interval after filtering, automatic gain control, and carrier removal. The sampling time is adjusted to coincide with the “center” of the received pulse. The shape of the pulse is chosen to attain the maximum rate at which the receiver can still distinguish the different pulses. To achieve this goal, we usually choose a band-limited pulse that has periodic zero crossings every T_b s.

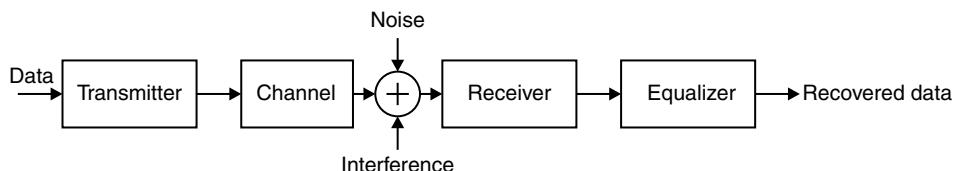


FIGURE 1.18

Simple model of a digital communications system.

If the periodic zero crossings of the pulse are preserved after transmission and reception, we can measure its amplitude without interference from overlapping adjacent pulses. However, channels that deviate from the ideal response (constant magnitude and linear phase) destroy the periodic zero-crossing property and the shape of the peak of the pulse. As a result, the tails of adjacent pulses interfere with the measurement of the current pulse and can lead to an incorrect decision. This type of degradation, which is known as *intersymbol interference (ISI)*, is illustrated in Figure 1.19.

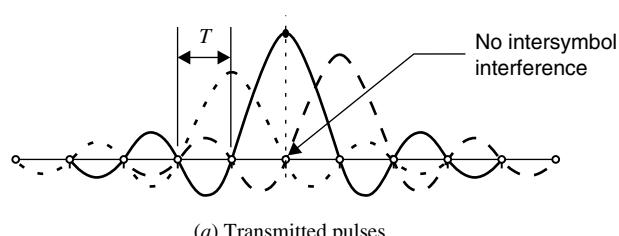
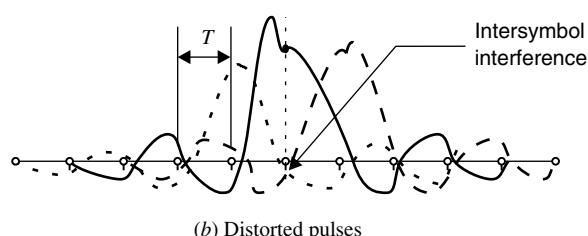


FIGURE 1.19
Pulse trains (a) without
intersymbol interference and (b)
with intersymbol interference.



We can compensate for the ISI distortion by using a linear filter called an *equalizer*. The goal of the equalizer is to restore the received pulse, as closely as possible, to its original shape. The equalizer transforms the channel to a near-ideal one if its response resembles the *inverse* of the channel. Since the channel is unknown and possibly time-varying, there are two ways to approach the problem: (1) Design a fixed compromise equalizer to obtain satisfactory performance over a broad range of channels, or (2) design an equalizer that can “learn” the inverse of the particular channel and then “track” its variation in real time.

The characteristics of the equalizer are adjusted by some algorithm that attempts to attain the best possible performance. The most appropriate criterion of performance for data transmission systems is the probability of symbol error. However it cannot be used for two reasons: (1) The “correct” symbol is unknown to the receiver (otherwise there would be no reason to communicate), and (2) the number of decisions (observations) needed to estimate the low probabilities of error is extremely large. Thus, practical equalizers assess their performance by using some function of the difference between the “correct” symbol and the output. The operation of practical equalizers involves two modes[†] of operation, dependent on how we substitute for the unavailable correct symbol sequence. (1) A known *training sequence* is transmitted, and the equalizer attempts to improve its performance by comparing its output to a synchronized replica of the training sequence stored at the receiver. Usually this mode is used when the equalizer starts a transmission session. (2) At the end of the training session, when the equalizer starts making reliable decisions, we can replace the training sequence with the equalizer’s own decisions.

Adaptive equalization is a mature technology that has had the greatest impact on digital communications systems, including voiceband, microwave and troposcatter radio, and cable TV modems (Qureshi 1985; Lee and Messerschmitt 1994; Gitlin et al. 1992; Bingham 1988; Treichler et al. 1996).

Signal prediction

In the next class of applications, the goal is to estimate the value $x(n_0)$ of a random signal by using a set of consecutive signal samples $\{x(n), n_1 \leq n \leq n_2\}$. There are three cases of interest: (1) forward prediction, when $n_0 > n_2$; (2) backward “prediction,” when $n_0 < n_1$; and (3) smoothing or interpolation, when $n_1 < n_0 < n_2$. Clearly, in the last case the value at $n = n_0$ is not used in the computation of the estimate. The most widely used type is forward linear prediction or simply *linear prediction*[‡] [see Figure 1.15(c)], where the estimate is formed by using a linear combination of past samples (Makhoul 1975).

Linear predictive coding (LPC). The efficient storage and transmission of analog signals using digital systems requires the minimization of the number of bits necessary to represent the signal while maintaining the quality to an acceptable level according to a certain criterion of performance. The conversion of an analog (continuous-time, continuous-amplitude) signal to a digital (discrete-time, discrete-amplitude) signal involves two processes: sampling and quantization. Sampling converts a continuous-time signal to a discrete-time signal by measuring its amplitude at equidistant intervals of time. Quantization involves the representation of the measured continuous amplitude using a finite number of symbols and always creates some amount of distortion (quantization noise).

For a fixed number of bits, decreasing the dynamic range of the signal (and therefore the range of the quantizer) decreases the required quantization step and therefore the average quantization error power. Therefore, we can decrease the quantization noise by reducing the dynamic range or equivalently the variance of the signal. If the signal samples are

[†]Another mode of operation, where the equalizer can operate without the benefit of a training sequence (blind or self-recovering mode), is discussed in Chapter 12.

[‡]As we shall see in Chapters 4 and 6, linear prediction is closely related, but not identical, to all-pole signal modeling.

significantly correlated, the variance of the difference between adjacent samples is smaller than the variance of the original signal. Thus, we can improve quality by quantizing this difference instead of the original signal. This idea is exploited by the linear prediction system shown in Figure 1.20. This system uses a linear predictor to form an estimate (prediction) $\hat{x}(n)$ of the present sample $x(n)$ as a linear combination of the M past samples, that is,

$$\hat{x}(n) = \sum_{k=1}^M a_k x(n-k) \quad (1.4.1)$$

The coefficients $\{a_k\}_1^M$ of the linear predictor are determined by exploiting the correlation between adjacent samples of the input signal with the objective of making the prediction error

$$e(n) = x(n) - \hat{x}(n) \quad (1.4.2)$$

as small as possible. If the prediction is good, the dynamic range of $e(n)$ should be smaller than the dynamic range of $x(n)$, resulting in a smaller quantization noise for the same number of bits or the same quantization noise with a smaller number of bits. The performance of the LPC system depends on the accuracy of the predictor. Since the statistical properties of the signal $x(n)$ are unknown and change with time, we *cannot* design an optimum fixed predictor. The established practical solution is to use an *adaptive* linear predictor that automatically adjusts its coefficients to compute a “good” prediction at each time instant. A detailed discussion of adaptive linear prediction and its application to audio, speech, and video signal coding is provided in Jayant and Noll (1984).

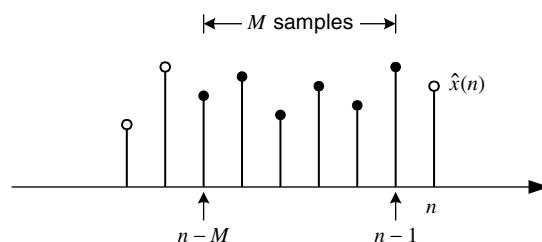


FIGURE 1.20

Illustration of the linear prediction of a signal $x(n)$ using a finite number of past samples.

Multisensor interference cancellation

The key feature of this class of applications is the use of multiple sensors to remove undesired interference and noise. Typically, a *primary* signal contains both the signal of interest and the interference. Other signals, known as *reference* signals, are available for the purposes of canceling the undesired interference [see Figure 1.15(d)]. These reference signals are collected using other sensors in which the signal of interest is not present or is so weak that it can be ignored. The amount of correlation between the primary and reference signals is measured and used to form an estimate of the interference in the primary signal, which is subsequently removed. Had the signal of interest been present in the reference signal(s), then this process would have resulted in the removal of the desired signal as well. Typical applications in which interference cancellation is employed include array processing for radar and communications, biomedical sensing systems, and active noise control (Widrow et al. 1975; Kuo and Morgan 1996).

Active noise control (ANC). The basic idea behind an ANC system is the cancellation of acoustic noise using destructive wave interference. To create destructive interference that cancels an acoustic noise wave (primary) at a point P , we can use a loudspeaker that creates, at the same point P , another wave (secondary) with the same frequency, the same amplitude, and 180° phase difference. Therefore, with appropriate control of the peaks and troughs

of the secondary wave, we can produce zones of destructive interference (quietness). ANC systems using digital signal processing technology find applications in air-conditioning ducts, aircraft, cars, and magnetic resonance imaging (MRI) systems (Elliott and Nelson 1993; Kuo and Morgan 1996).

Figure 1.21 shows the key components of an adaptive ANC system described in Crawford et al. 1997. The task of the loudspeaker is to generate an acoustic wave that is an 180° phase-inverted version of the signal $y(t)$ when it arrives at the error microphone. In this case the error signal $e(t) = y(t) + \hat{y}(t) = 0$, and we create a “quiet zone” around the microphone. If the acoustic paths (1) from the noise source to the reference microphone (G_x), (2) from the noise source to the error microphone (G_y), (3) from the secondary loudspeaker to the reference microphone (H_x), and (4) from the secondary loudspeaker to the error microphone ($H_{\hat{y}}$) are linear, time-invariant, and known, we can design a linear filter H such that $e(n) = 0$. For example, if the effects of H_x and $H_{\hat{y}}$ are negligible, the filter H should invert G_x to obtain $v(t)$ and then replicate G_y to synthesize $\hat{y}(t) \approx y(t)$. The quality of cancellation depends on the accuracy of these two modeling processes.

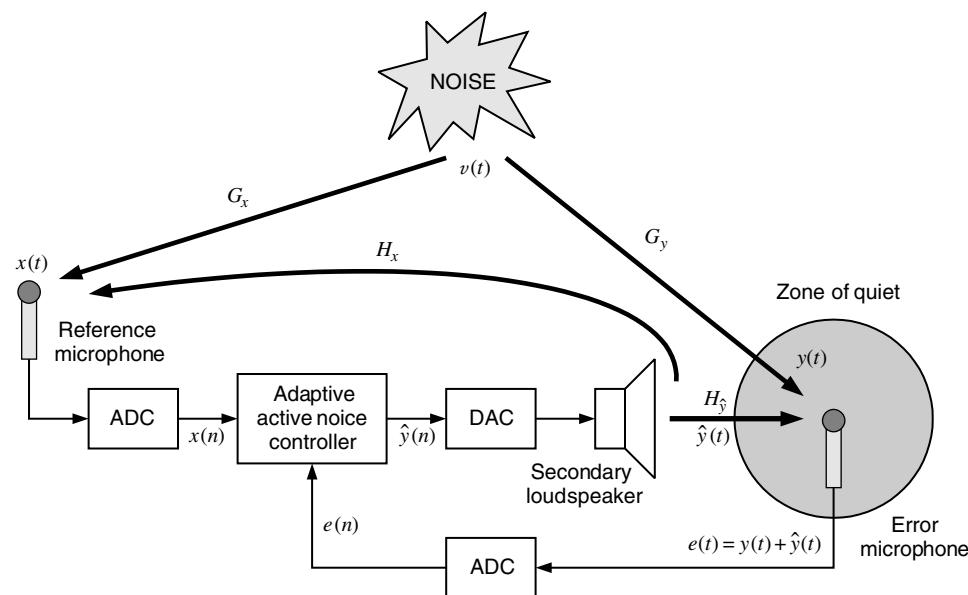


FIGURE 1.21

Block diagram of the basic components of an active noise control system.

In practice, the acoustic environment is unknown and time-varying. Therefore, we cannot design a fixed ANC filter with satisfactory performance. The only feasible solution is to use an adaptive filter with the capacity to identify and track the variation of the various acoustic paths and the spectral characteristics of the noise source in real time. The adaptive ANC filter adjusts its characteristics by trying to minimize the energy of the error signal $e(n)$. Adaptive ANC using digital signal processing technology is an active area of research, and despite several successes many problems remain to be solved before such systems find their way to more practical applications (Crawford et al. 1997).

1.4.2 Features of Adaptive Filters

Careful inspection of the applications discussed in the previous section indicates that every adaptive filter consists of the following three modules (see Figure 1.22).

1. **Filtering structure.** This module forms the output of the filter using measurements of the input signal or signals. The filtering structure is *linear* if the output is obtained as a linear combination of the input measurements; otherwise, it is said to be *nonlinear*. The structure is fixed by the designer, and its parameters are adjusted by the adaptive algorithm.
2. **Criterion of performance (COP).** The output of the adaptive filter and the desired response (when available) are processed by the COP module to assess its quality with respect to the requirements of the particular application.
3. **Adaptive algorithm.** The adaptive algorithm uses the value of the criterion of performance, or some function of it, and the measurements of the input and desired response (when available) to decide how to modify the parameters of the filter to improve its performance.

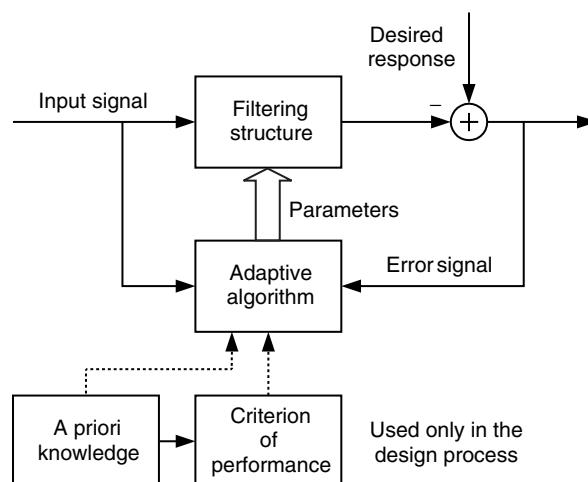


FIGURE 1.22
Basic elements of a general adaptive filter.

Every adaptive filtering application involves one or more input signals and a desired response signal that may or may not be accessible to the adaptive filter. We collectively refer to these relevant signals as the *signal operating environment (SOE)* of the adaptive filter. The design of any adaptive filter requires a great deal of *a priori information* about the SOE and a deep understanding of the particular application (Claasen and Mecklenbrauker 1985). This information is needed by the designer to choose the filtering structure and the criterion of performance and to design the adaptive algorithm. To be more specific, adaptive filters are designed for a specific type of input signal (speech, binary data, etc.), for specific types of interferences (additive white noise, sinusoidal signals, echoes of the input signals, etc.), and for specific types of signal transmission paths (e.g., linear time-invariant or time-varying). After the proper design decisions have been made, the only unknowns, when the adaptive filter starts its operation, are a set of parameters that are to be determined by the adaptive algorithm using signal measurements. Clearly, unreliable *a priori* information and/or incorrect assumptions about the SOE can lead to serious performance degradations or even unsuccessful adaptive filter applications.

If the characteristics of the relevant signals are constant, the goal of the adaptive filter is to find the parameters that give the best performance and then to stop the adjustment. However, when the characteristics of the relevant signals change with time, the adaptive filter should first find and then continuously readjust its parameters to track these changes.

A very influential factor in the design of adaptive algorithms is the availability of a desired response signal. We have seen that for certain applications, the desired response may not be available for use by the adaptive filter. In this book we focus on supervised

adaptive filters that require the use of a desired response signal and we simply call them adaptive filters (Chapter 10). Unsupervised adaptive filters are discussed in Chapter 12.

Suppose now that the relevant signals can be modeled by stochastic processes with known statistical properties. If we adopt the minimum mean square error as a criterion of performance, we can design, at least in principle, an optimum filter that provides the ultimate solution. From a theoretical point of view, the goal of the adaptive filter is to replicate the performance of the optimum filter without the benefit of knowing and using the *exact* statistical properties of the relevant signals. In this sense, the theory of optimum filters (see Chapters 6 and 7) is a prerequisite for the understanding, design, performance evaluation, and successful application of adaptive filters.

1.5 ARRAY PROCESSING

Array processing deals with techniques for the analysis and processing of signals collected by a group of sensors. The collection of sensors makes up the array, and the manner in which the signals from the sensors are combined and handled constitutes the processing. The type of processing is dictated by the needs of the particular application. Array processing has found widespread application in a large number of areas, including radar, sonar, communications, seismology, geophysical prospecting for oil and natural gas, diagnostic ultrasound, and multichannel audio systems.

1.5.1 Spatial Filtering or Beamforming

Generally, an array receives spatially propagating signals and processes them to emphasize signals arriving from a certain direction; that is, it acts as a spatially discriminating filter. This spatial filtering operation is known as *beamforming*, because essentially it emulates the function of a mechanically steered antenna. An array processor steers a beam to a particular direction by computing a properly weighted sum of the individual sensor signals. An example of the spatial response of the beamformer, known as the *beampattern*, is shown in Figure 1.23. The beamformer emphasizes signals in the direction to which it is steered while attenuating signals from other directions.

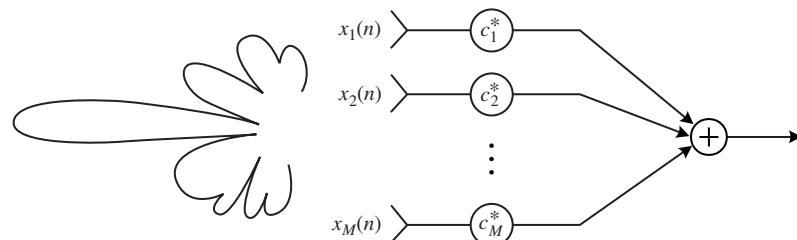
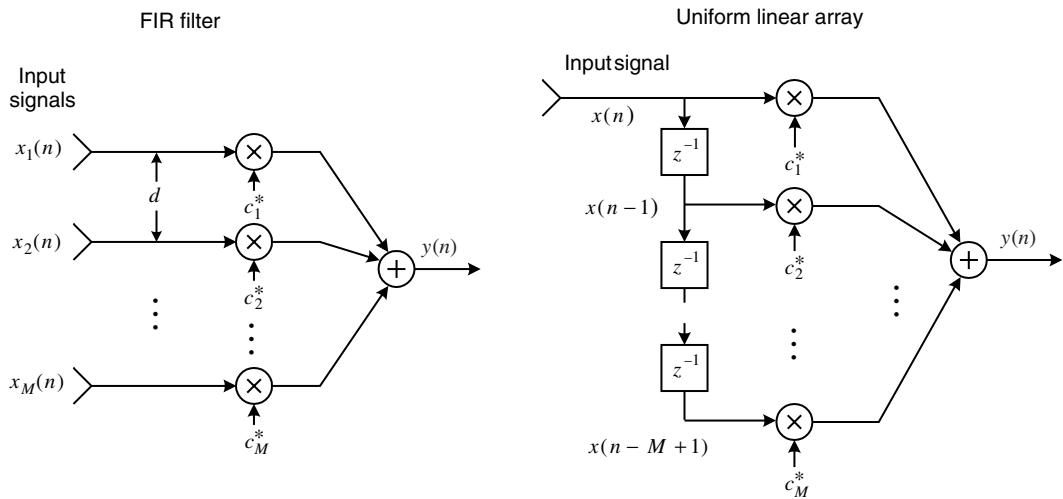


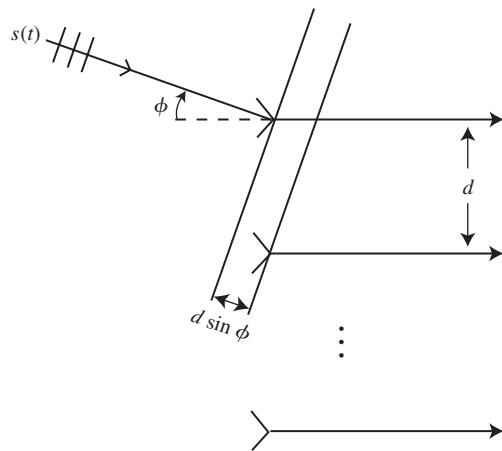
FIGURE 1.23

Example of the spatial response of an array, known as a beampattern, that emphasizes signals from a direction of interest, known as the look direction.

In the case of an array with sensors equally spaced on a line, known as a *uniform linear array (ULA)*, there is a direct analogy between beamforming and the frequency-selective filtering of a discrete-time signal using a *finite impulse response (FIR)* filter. This analogy between a beamformer and an FIR filter is illustrated in Figure 1.24. The array of sensors spatially samples the impinging waves so that in the case of a ULA, the sampling

**FIGURE 1.24**

Analogy between beamforming and frequency-selective FIR filtering.

**FIGURE 1.25**

Wave impinging on a uniform linear array with element spacing d .

is performed at equal spatial increments. By contrast, an FIR filter uses a uniformly time-sampled signal as its input. Consider a plane wave impinging on an array as in Figure 1.25. The spatial signal arrives at each sensor with a delay determined by the angle of arrival ϕ . In the case of a narrowband signal, this delay corresponds to an equal phase shift from sensor to sensor that results in a spatial frequency across the ULA of

$$u = \frac{d}{\lambda} \sin \phi \quad (1.5.1)$$

where λ is the wavelength of the signal and d is the uniform spacing of the sensors. This spatial frequency is analogous to the temporal frequency encountered in discrete-time signals. In the beamforming operation, the sensor signals are combined with weights on each of the sensor signals just as an FIR filter produces an output that is the weighted sum of time samples. As a frequency-selective FIR filter extracts signals at a frequency of interest, a beamformer seeks to emphasize signals with a certain spatial frequency (i.e., signals arriving from a particular angle). Thus, it is often beneficial to view a beamformer as a spatial frequency-selective filter.

Many times an array must contend with undesired signals arriving from other directions, which may prevent it from successfully extracting the signal of interest for which it was designed. In this case, the array must adjust its response to the data it receives to reject signals

from these other directions. The resulting array is an *adaptive array* as the beamforming weights are automatically determined by the array during its normal operation without the intervention of the designer. Drawing on the frequency-selective FIR filter comparison again, we see that an adaptive array is analogous to an adaptive FIR filter that adjusts its weights to pass signals at the desired frequency or signals with certain statistical properties while rejecting any signals that do not satisfy these requirements. Again, if we can model the SOE, using stationary processes with known statistical properties, we can design an *optimum* beamformer that minimizes or maximizes a certain criterion of performance. The optimum beamformer can be used to provide guidelines for the design of adaptive beamformers and used as a yardstick for their performance evaluation. The analysis, design, and performance evaluation of fixed, optimum, and adaptive beamformers are discussed in Chapter 11.

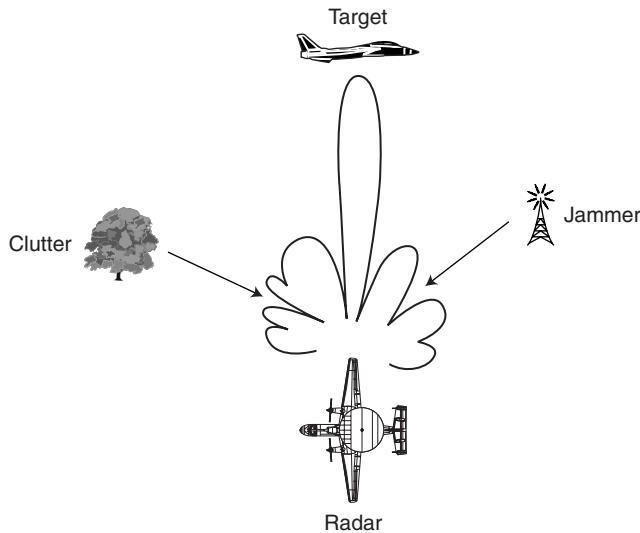
1.5.2 Adaptive Interference Mitigation in Radar Systems

The goal of an airborne surveillance radar system is to determine the presence of target signals. These targets can be either airborne or found on the ground below. Typical targets of interest are other aircraft, ground moving vehicles, or hostile missiles. The desired information from these targets is their relative distance from our airborne platform, known as the *range*, their angle with respect to the platform, and their relative speed. The processing of the radar consists of the following sequence:

- Filter out undesired signals through adaptive processing.
- Determine the presence of targets, a process known as *detection*.
- Estimate the parameters of all detected targets.

To sense these targets, the radar system transmits energy in the direction it is searching for targets. The transmitted energy propagates from the airborne radar to the target that reflects the radar signal. The reflection then propagates from the target back to the radar. Since the radar signal travels at the speed of light (3×10^8 m/s), the round-trip delay between transmission and reception of this signal determines the range of the target. The received signal is known as the *return*. The angle of the target is determined through the use of beamforming or spatial filtering using an array of sensor elements. To this end, the radar forms a bank of spatial filters evenly spaced in angle and determines which filter contains the target. For example, we might be interested in the angular sector between $-1^\circ \leq \phi \leq 1^\circ$. Then we might set up a bank of beamformers in this angular region with a spacing of 0.5° . If these spatial filters perform this operation nonadaptively, it is often referred to as *conventional beamforming*.

The detection of target signals is inhibited by the presence of other undesired signals known as *interference*. Two common types of interference are the reflections of the radar signal from the ground, known as *clutter*, and other transmitted energy at the same operating frequency as the radar, referred to as *jamming*. Jamming can be the hostile transmission of energy to prevent us from detecting certain signals, or it may be incidental, for example, from another radar. Such an interference scenario for an airborne surveillance radar is depicted in Figure 1.26. The interference signals are typically much larger than the target return. Thus, when a nonadaptive beamformer is used, interference leaks in through the sidelobes of the beamformer and prevents us from detecting the target. However, we can adjust the beamformer weights such that signals from the directions of the interference are rejected while other directions are searched for targets. If the weights are adapted to the received data in this way, then the array is known as an *adaptive array* and the operation is called *adaptive beamforming*. The use of an adaptive beamformer is also illustrated in Figure 1.26. We show the spatial response or *beampattern* of the adaptive array. Note that the peak gain of the beamformer is in the direction of the target. On the other hand, the clutter and jamming are rejected by placing *nulls* in the beampattern.

**FIGURE 1.26**

Example of adaptive beamformer used with an airborne surveillance radar for interference mitigation.

In practice, we do not know the directions of the interferers. Therefore, we need an adaptive beamformer that can determine its weights by estimating the statistics of the interference environment. If we can model the SOE using stochastic processes with known statistical properties, we can design an optimum beamformer that provides the ultimate performance. The discussion about adaptive filters in Section 1.4.2 applies to adaptive beamformers as well.

Once we have determined the presence of the target signal, we want to get a better idea of the exact angle it was received from. Recall that the beamformers have angles associated with them, so the angle of the beamformer in which the target was detected can serve as a rough estimate of the angle of the target. The coarseness of our initial estimate is governed by the spacing in angle of the filter bank of beamformers, for example, 1° . This resolution in angle of the beamformer is often called a *beamwidth*. To get a better estimate, we can use a variety of angle estimation methods. If the angle estimate can refine the accuracy down to one-tenth of a beamwidth, for example, 0.1° , then the angle estimator is said to achieve 10 : 1 beamsplitting. Achieving an angle accuracy better than the array beamwidth is often called *superresolution*.

1.5.3 Adaptive Sidelobe Canceler

Consider the scenario in Figure 1.26 from the adaptive beamforming example for interference mitigation in a radar system. However, instead of an array of sensors, consider a fixed (i.e., nonadaptive) channel that has high gain in the direction of the target. This response may have been the result of a highly directive dish antenna or a nonadaptive beamformer. Sometimes it is necessary to perform beamforming nonadaptively to limit the number of channels. One such case arises for very large arrays for which it is impractical to form channels by digitally sampling every element. The array is partitioned into subarrays that all form nonadaptive beams in the same direction. Then the subarray outputs form the spatial channels that are sampled. Each channel is highly directive, though with a lower resolution than the entire array. In the case of interference, it is then present in all these subarray chan-

nels and must be removed in some way. To restore its performance to the interference-free case, the radar system must employ a spatially adaptive method that removes the interference in the main channel. The *sidelobe canceler* is one such method and is illustrated in Figure 1.27.

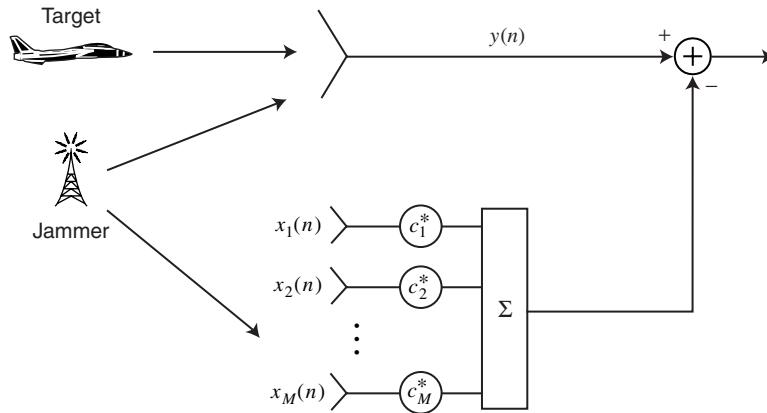


FIGURE 1.27

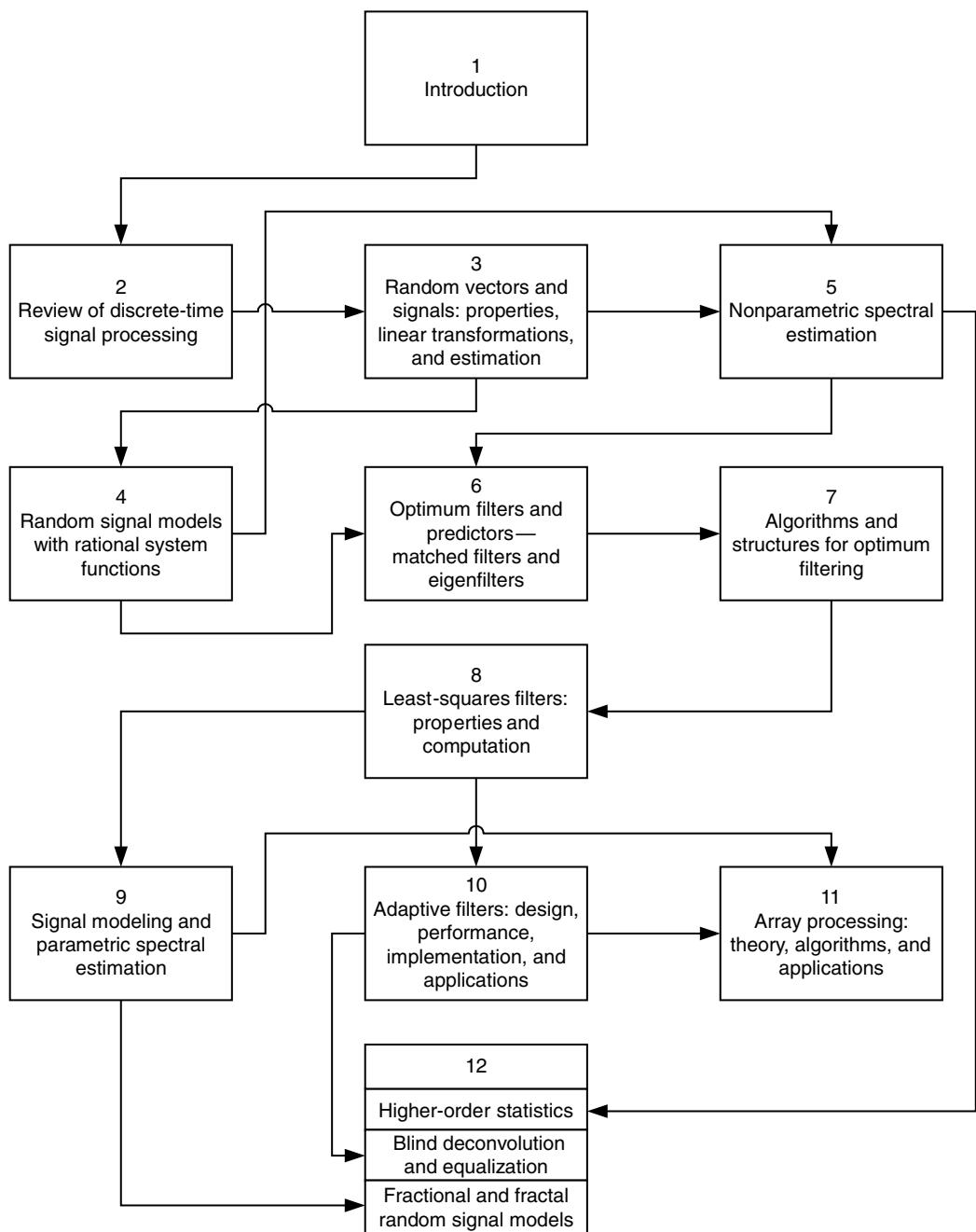
Sidelobe canceler with a highly directive main channel and auxiliary channels.

Note that the signal of interest is received from a particular direction in which we assume the main channel has a large gain. On the other hand, the jamming signal is received from another direction, and since it has much higher power than the attenuation of the antenna sidelobes, the jamming interference obscures the signals of interest. This high-gain channel is known as the main channel that contains both the signal of interest and the jamming interference. The sidelobe canceler uses one or more auxiliary channels in order to cancel the main-channel interference. These auxiliary channels typically have much lower gain in the direction in which the main channel is directed so that they contain only the interference. The signal of interest is weak enough that it is below the thermal noise floor in these auxiliary channels. Examples of these auxiliary channels would be omnidirectional sensors or even directive sensors pointed in the direction of the interference. Note that for very strong signals, the signal of interest may be present in the auxiliary channel, in which case signal cancelation can occur. Clearly, this application belongs to the class of multisensor interference cancellation shown in Figure 1.15.

The sidelobe canceler uses the auxiliary channels to form an estimate of the interference in the main channel. The estimate is computed by weighting the auxiliary channel in an adaptive manner dependent on the cross-correlation between the auxiliary channels and the main channel. The estimate of the main-channel interference is subtracted from the main channel. The result is an overall antenna response with a spatial null directed at the interference source while maintaining high gain in the direction of interest. Clearly, if we had sufficient a priori information, the problem could be solved by designing a fixed canceler. However, the lack of a priori information and the changing properties of the environment make an adaptive canceler the only viable solution.

1.6 ORGANIZATION OF THE BOOK

In this section we provide an overview of the main topics covered in the book so as to help the reader navigate through the material and understand the interdependence among the various chapters (see Figure 1.28).

**FIGURE 1.28**

Flowchart organization of the book's chapters.

In Chapter 2, we review the fundamental topics in discrete-time signal processing that can be used for both deterministic and random signals. Chapter 3 provides a concise review of the theory of random variables and random sequences and elaborates on certain topics that are crucial to developments in subsequent chapters. Reading these chapters is essential to familiarize the reader with notation and properties that are repeatedly used throughout the rest of the book. Chapter 5 presents the most practical methods for nonparametric estimation

of correlation and spectral densities. The use of these techniques for exploratory investigation of the relevant signal characteristics before performing any modeling or adaptive filtering is invaluable.

Chapters 4 and 6 provide a detailed study of the theoretical properties of signal models and optimum filters, assuming that the relevant signals can be modeled by stochastic processes with known statistical properties. In Chapter 7, we develop algorithms and structures for optimum filtering and signal modeling and prediction.

Chapter 8 introduces the general method of least squares and shows how to use it for the design of filters and predictors from actual signal observations. The statistical properties and the numerical computation of least-squares estimates are also discussed in detail.

Chapters 9, 10, and 11 use the theoretical work in Chapters 4, 6, and 7 and the practical methods in Chapter 8 to develop, evaluate, and apply practical techniques for signal modeling, adaptive filtering, and array processing. Finally, Chapter 12 illustrates the use of higher-order statistics, presents the basic ideas of blind deconvolution and equalization, and concludes with a concise introduction to fractional and random fractal signal models.

Fundamentals of Discrete-Time Signal Processing

In many disciplines, signal processing applications nowadays are almost always implemented using digital hardware operating on digital signals. The basic foundation of this modern approach is based on discrete-time system theory. This book also deals with statistical analysis and processing of discrete-time signals, and modeling of discrete-time systems. Therefore, the purpose of this chapter is to focus attention on some important issues of discrete-time signal processing that are of fundamental importance to signal processing, in general, and to this book, in particular. The intent of this chapter is not to teach topics in elementary digital signal processing but to review material that will be used throughout this book and to establish a consistent notation for it. There are several textbooks on these topics, and it is assumed that the reader is familiar with the theory of digital signal processing as found in Oppenheim and Schafer (1989); Proakis and Manolakis (1996).

We begin this chapter with a description and classification of signals in Section 2.1. Representation of deterministic signals from the frequency-domain viewpoint is presented in Section 2.2. In Section 2.3, discrete-time systems are defined, but the treatment is focused on *linear, time-invariant (LTI)* systems, which are easier to deal with mathematically and hence are widely used in practice. Section 2.4 on minimum-phase systems and system invertibility is an important section in this chapter that should be reviewed prior to studying the rest of the book. The last section, Section 2.5, is devoted to lattice and lattice/ladder structures for discrete-time systems (or filters). A brief summary of the topics discussed in this chapter is provided in Section 2.6.

2.1 DISCRETE-TIME SIGNALS

The physical world is replete with *signals*, that is, physical quantities that change as a function of time, space, or some other independent variable. Although the physical nature of signals arising in various applications may be quite different, there are signals that have some basic features in common. These attributes make it possible to classify signals into families to facilitate their analysis. On the other hand, the mathematical description and analysis of signals require *mathematical signal models* that allow us to choose the appropriate mathematical approach for analysis. Signal characteristics and the classification of signals based upon either such characteristics or the associated mathematical models are the subject of this section.

2.1.1 Continuous-Time, Discrete-Time, and Digital Signals

If we assume that to every set of assigned values of independent variables there corresponds a unique value of the physical quantity (dependent variable), then every signal can be viewed as a function. The dependent variable may be real, in which case we have a *real-valued signal*; or it may be complex, and then we talk about a *complex-valued signal*. The independent variables are always real.

Any signal whose samples are a single-valued function of one independent variable is referred to as a *scalar one-dimensional signal*. We will refer to it simply as a signal. These signals involve one dependent variable and one independent variable and are the signals that we mainly deal with in this book. The speech signal shown in Figure 1.1 provides a typical example of a scalar signal.

Let us now look at both the dependent and independent variables of a signal from a different perspective. Every signal variable may take on values from either a continuous set of values (*continuous variable*) or a discrete set of values (*discrete variable*). Signals whose dependent and independent variables are continuous are usually referred to as *continuous-time signals*, and we will denote these signals by the subscript c, such as $x_c(t)$. In contrast, signals where both the dependent and the independent variables are discrete are called *digital signals*. If only the independent variables are specified to be discrete, then we have a *discrete signal*. We note that a discrete signal is defined only at discrete values of the independent variables, but it may take on any value. Clearly, digital signals are a subset of the set of discrete signals.

In this book, we mainly deal with scalar discrete signals in which the independent variable is time. We refer to them as *discrete-time signals*. Such signals usually arise in practice when we *sample* continuous-time signals, that is, when we select values at discrete-time instances. In all practical applications, the values of a discrete-time signal can only be described by binary numbers with a finite number of bits. Hence, only a discrete set of values is possible; strictly speaking, this means that, in practice, we deal with only digital signals. Clearly, digital signals are the only signals amenable to direct digital computation. Any other signal has to be first converted to digital form before numerical processing is possible.

Because the discrete nature of the dependent variable complicates the analysis, the usual practice is to deal with discrete-time signals and then to consider the effects of the discrete amplitude as a separate issue. Obviously, these effects can be reduced to any desirable level by accordingly increasing the number of bits (or word length) in the involved numerical processing operations. Hence, in the remainder of the book, we limit our attention to discrete-time signals.

2.1.2 Mathematical Description of Signals

The mathematical analysis of a signal requires the availability of a mathematical description for the signal itself. The type of description, usually referred to as a *signal model*, determines the most appropriate mathematical approach for the analysis of the signal. We use the term *signal* to refer to either the signal itself or its mathematical description, that is, the signal model. The exact meaning will be apparent from the context. Clearly, this distinction is necessary if a signal can be described by more than one model. We start with the most important classification of signal models as either deterministic or random.

Deterministic signals

Any signal that can be described by an explicit mathematical relationship is called *deterministic*. In the case of continuous-time signals, this relationship is a given function of time, for example, $x_c(t) = A \cos(2\pi F_0 t + \theta)$, $-\infty < t < \infty$. For discrete-time signals

that, mathematically speaking, are sequences of numbers, this relationship may be either a functional expression, for example, $x(n) = a^n$, $-\infty < n < \infty$, or a table of values.

In general, we use the notation $x(n)$ to denote the sequence of numbers that represent a discrete-time signal. Furthermore, we use the term *nth sample* to refer to the value of this sequence for a specific value of n . Strictly speaking, the terminology is correct only if the discrete-time signal has been obtained by sampling a continuous-time signal $x_c(t)$. In the case of periodic sampling with sampling period T , we have $x(n) = x_c(nT)$, $-\infty < n < \infty$; that is, $x(n)$ is the n th sample of $x_c(t)$. Sometimes, just for convenience, we may plot $x_c(t)$ even if we deal with the signal $x(n)$. Finally, we note that sometimes it is convenient to form and manipulate complex-valued signals using a pair of real-valued signals as the real and imaginary components.

Basic signals. There are some basic discrete-time signals that we will repeatedly use throughout this book:

- The *unit sample* or *unit impulse* sequence $\delta(n)$, defined as

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases} \quad (2.1.1)$$

- The *unit step* sequence $u(n)$, defined as

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases} \quad (2.1.2)$$

- The exponential sequence of the form

$$x(n) = a^n \quad -\infty < n < \infty \quad (2.1.3)$$

If a is a complex number, that is, $a = re^{j\omega_0}$, $r > 0$, $\omega_0 \neq 0, \pi$, then $x(n)$ is complex-valued, that is,

$$x(n) = r^n e^{j\omega_0 n} = x_R(n) + jx_I(n) \quad (2.1.4)$$

$$\text{where } x_R(n) = r^n \cos \omega_0 n \quad \text{and} \quad x_I(n) = r^n \sin \omega_0 n \quad (2.1.5)$$

are the real and imaginary parts of $x(n)$, respectively. The complex exponential signal $x(n)$ and the real sinusoidal signals $x_R(n)$ and $x_I(n)$, which have a decaying (growing) envelope if $r < 1$ ($r > 1$), are very useful in the analysis of discrete-time signals and systems.

Signal classification. Deterministic signals can be classified as energy or power, periodic or aperiodic, of finite or infinite duration, causal or noncausal, and even or odd signals. Although we next discuss these concepts for discrete-time signals, a similar discussion applies to continuous-time signals as well.

- The total energy or simply the *energy* of a signal $x(n)$ is given by

$$E_x = \sum_{n=-\infty}^{\infty} |x(n)|^2 \geq 0 \quad (2.1.6)$$

The energy is zero if and only if $x(n) = 0$ for all n . The average power or simply the *power* of a signal $x(n)$ is defined as

$$P_x = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N |x(n)|^2 \geq 0 \quad (2.1.7)$$

A signal with finite energy, that is, $0 < E_x < \infty$, is called an *energy signal*. Signals with finite power, that is, $0 < P_x < \infty$, are referred to as *power signals*. Clearly, energy signals have zero power, and power signals have infinite energy.

- A discrete-time signal $x(n)$ is called *periodic* with fundamental period N if $x(n + N) = x(n)$ for all n . Otherwise it is called *aperiodic*. It can be seen that the complex exponential in (2.1.4) is periodic if and only if $\omega_0/(2\pi) = k/N$, that is, if $\omega_0/(2\pi)$ is a rational number. Clearly, a periodic signal is a power signal with power P given by

$$P_x = \frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2 \quad (2.1.8)$$

- We say that a signal $x(n)$ has *finite duration* if $x(n) = 0$ for $n < N_1$ and $n > N_2$, where N_1 and N_2 are finite integer numbers with $N_1 \leq N_2$. If $N_1 = -\infty$ and/or $N_2 = \infty$, the signal $x(n)$ has *infinite duration*.
- A signal $x(n)$ is said to be *causal* if $x(n) = 0$ for $n < 0$. Otherwise, it is called *noncausal*.
- Finally, a real-valued signal $x(n)$ is called *even* if $x(-n) = x(n)$ and *odd* if $x(-n) = -x(n)$.

Other classifications for deterministic signals will be introduced in subsequent sections.

Random signals

In contrast to the deterministic signals discussed so far, there are many other signals in practice that cannot be described to any reasonable accuracy by explicit mathematical relationships. The lack of such an explicit relationship implies that the signal evolves in time in an unpredictable manner from the point of view of the observer. Such signals are called *random*. The output of a noise generator, the height of waves in a stormy sea, and the acoustic pressures generated by air rushing through the human vocal tract are examples of random signals. At this point one could say that complete knowledge of the physics of the signal could provide an explicit mathematical relationship, at least within the limits of the uncertainty principle. However, such relationships are typically too complex to be of any practical use.

In general, although random signals are evolving in time in an unpredictable manner, their average properties can often be assumed to be deterministic; that is, they can be specified by explicit mathematical formulas. This concept is key to the modeling of a random signal as a stochastic process.

Thus, random signals are mathematically described by *stochastic processes* and can be analyzed by using *statistical* methods instead of explicit equations. The theory of probability, random variables, and stochastic processes provides the mathematical framework for the theoretical study of random signals.

2.1.3 Real-World Signals

The classification of various physical data as being either deterministic or random might be debated in many cases. For example, it might be argued that no physical data in practice can be truly deterministic since there is always a possibility that some unforeseen event in the future might influence the phenomenon producing the data in a manner that was not originally considered. On the other hand, it might be argued that no physical data are truly random since exact mathematical descriptions might be possible if sufficient knowledge of the basic mechanisms of the phenomenon producing the data were known. In practical terms, the decision as to whether physical data are deterministic or random is usually based upon the ability to reproduce the data by controlled experiments. If an experiment producing specific data of interest can be repeated many times with identical results (within the limits of experimental error), then the data can generally be considered deterministic. If an experiment cannot be designed that will produce identical results when the experiment is repeated, then the data must usually be considered random in nature.

2.2 TRANSFORM-DOMAIN REPRESENTATION OF DETERMINISTIC SIGNALS

37

SECTION 2.2
Transform-Domain
Representation
of Deterministic Signals

In the deterministic signal model, signals are assumed to be explicitly known for all time from $-\infty$ to $+\infty$. In this sense, no uncertainty exists regarding their past, present, or future amplitude values. The simplest description of any signal is an amplitude-versus-time plot. This “time history” of the signal is very useful for visual analysis because it helps in the identification of specific patterns, which can subsequently be used to extract useful information from the signal. However, quite often, information present in a signal becomes more evident by transformation of the signal into another domain. In this section, we review some transforms for the representation and analysis of discrete-time signals.

2.2.1 Fourier Transforms and Fourier Series

Frequency analysis is, roughly speaking, the process of decomposing a signal into frequency components, that is, complex exponential signals or sinusoidal signals. Although the physical meaning of frequency analysis is almost the same for any signal, the appropriate mathematical tools depend upon the type of signal under consideration. The two characteristics that specify the frequency analysis tools for deterministic signals are

- The nature of time: continuous-time or discrete-time signals.
- The existence of harmony: periodic or aperiodic signals.

Thus, we have the following four types of frequency analysis tools.

Fourier series for continuous-time periodic signals

If a continuous-time signal $x_c(t)$ is periodic with fundamental period T_p , it can be expressed as a linear combination of harmonically related complex exponentials

$$x_c(t) = \sum_{k=-\infty}^{\infty} \check{X}_c(k) e^{j2\pi k F_0 t} \quad (2.2.1)$$

where $F_0 = 1/T_p$ is the *fundamental frequency*, and

$$\check{X}_c(k) = \frac{1}{T_p} \int_0^{T_p} x_c(t) e^{-j2\pi k F_0 t} dt \quad (2.2.2)$$

which are termed the Fourier coefficients,[†] or the *spectrum* of $x_c(t)$.

It can be shown that the power of the signal $x_c(t)$ is given by Parseval's relation

$$P_x = \frac{1}{T_p} \int_0^{T_p} |x_c(t)|^2 dt = \sum_{k=-\infty}^{\infty} |\check{X}_c(k)|^2 \quad (2.2.3)$$

Since $|\check{X}_c(k)|^2$ represents the power in the k th frequency component, the sequence $|\check{X}_c(k)|^2$, $-\infty < k < \infty$, is called the *power spectrum* of $x_c(t)$ and shows the distribution of power within various frequency components. Since the power of $x_c(t)$ is confined to the discrete frequencies $0, \pm F_0, \pm 2F_0, \dots$, we say that $x_c(t)$ has a *line* or *discrete spectrum*.

Fourier transform for continuous-time aperiodic signals

The frequency analysis of a continuous-time, aperiodic signal can be done by using the Fourier transform

$$X_c(F) = \int_{-\infty}^{\infty} x_c(t) e^{-j2\pi F t} dt \quad (2.2.4)$$

[†]We use the notation $\check{X}_c(k)$ instead of $X_c(k)$ to distinguish it from the Fourier transform $X_c(F)$ introduced in (2.2.4).

which exists if $x_c(t)$ satisfies the *Dirichlet conditions*, which require that $x_c(t)$: (1) have a finite number of maxima or minima within any finite interval, (2) have a finite number of discontinuities within any finite interval, and (3) be absolutely integrable, that is,

$$\int_{-\infty}^{\infty} |x_c(t)| dt < \infty \quad (2.2.5)$$

The signal $x_c(t)$ can be synthesized from its *spectrum* $X_c(F)$ by using the following inverse Fourier transform formula

$$x_c(t) = \int_{-\infty}^{\infty} X_c(F) e^{j2\pi F t} dF \quad (2.2.6)$$

The energy of $x_c(t)$ can be computed in either the time or frequency domain using Parseval's relation

$$E_x = \int_{-\infty}^{\infty} |x_c(t)|^2 dt = \int_{-\infty}^{\infty} |X_c(F)|^2 dF \quad (2.2.7)$$

The function $|X_c(F)|^2 \geq 0$ shows the distribution of energy of $x_c(t)$ as a function of frequency. Hence, it is called the *energy spectrum* of $x_c(t)$. We note that continuous-time, aperiodic signals have *continuous* spectra.

Fourier series for discrete-time periodic signals

Any discrete-time periodic signal $x(n)$ with fundamental period N can be expressed by the following Fourier series

$$x(n) = \sum_{k=0}^{N-1} X_k e^{j(2\pi/N)kn} \quad (2.2.8)$$

$$\text{where } X_k = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/N)kn} \quad (2.2.9)$$

are the corresponding Fourier coefficients. The basis sequences $s_k(n) \triangleq e^{j(2\pi/N)kn}$ are periodic with fundamental period N in both time and frequency, that is, $s_k(n+N) = s_k(n)$ and $s_{k+N}(n) = s_k(n)$.

The sequence X_k , $k = 0, \pm 1, \pm 2, \dots$, is called the *spectrum* of the periodic signal $x(n)$. We note that $X_{k+N} = X_k$; that is, the spectrum of a discrete-time periodic signal is discrete and periodic with the same period.

The power of the periodic signal $x(n)$ can be determined by Parseval's relation

$$P_x = \frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2 = \sum_{k=0}^{N-1} |X_k|^2 \quad (2.2.10)$$

The sequence $|X_k|^2$ is known as the *power spectrum* of the periodic sequence $x(n)$.

Fourier transform for discrete-time aperiodic signals

Any discrete-time signal that is absolutely summable, that is,

$$\sum_{n=-\infty}^{\infty} |x(n)| < \infty \quad (2.2.11)$$

can be described by the discrete-time Fourier transform (DTFT)

$$X(e^{j\omega}) \triangleq \mathcal{F}[x(n)] = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n} \quad (2.2.12)$$

where $\omega = 2\pi f$ is the frequency variable in *radians per sampling interval* or simply in *radians per sample* and f is the frequency variable in *cycles per sampling interval* or simply

in *cycles per sample*. The signal $x(n)$ can be synthesized from its *spectrum* $X(e^{j\omega})$ by the inverse Fourier transform

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega \quad (2.2.13)$$

We will say that $x(n)$ and $X(e^{j\omega})$ form a *Fourier transform pair* denoted by

$$x(n) \xleftrightarrow{\mathcal{F}} X(e^{j\omega}) \quad (2.2.14)$$

The function $X(e^{j\omega})$ is periodic with fundamental period 2π . If $x(n)$ is real-valued, then $|X(e^{j\omega})| = |X(e^{-j\omega})|$ (even function) and $\angle X(e^{-j\omega}) = -\angle X(e^{j\omega})$ (odd function).

The energy of the signal can be computed in either the time or frequency domain using Parseval's relation

$$E_x = \sum_{n=-\infty}^{\infty} |x(n)|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X(e^{j\omega})|^2 d\omega \quad (2.2.15)$$

$$= \int_{-\pi}^{\pi} \frac{|X(e^{j\omega})|^2}{2\pi} d\omega \quad (2.2.16)$$

The function $|X(e^{j\omega})|^2/(2\pi) \geq 0$ and describes the distribution of the energy of the signal at various frequencies. Therefore, it is called the *energy spectrum* of $x(n)$.

Spectral classification of deterministic signals

So far we have discussed frequency analysis methods for periodic power signals and aperiodic energy signals. However, there are deterministic aperiodic signals with finite power. One such class of signals is the complex exponential $Ae^{j(\omega_0 n + \theta_0)}$ sequence [or equivalently, the sinusoidal sequence $A \cos(\omega_0 n + \theta_0)$], in which $\omega_0/(2\pi)$ is not a rational number. This sequence is not periodic, as discussed in Section 2.1.2; however it has a line spectrum at $\omega = \omega_0 + 2\pi k$, for any integer k , since

$$x(n) = Ae^{j(\omega_0 n + \theta_0)} = Ae^{j[(\omega_0 + 2\pi k)n + \theta_0]} \quad k = 0, \pm 1, \pm 2, \dots$$

(or at $\omega = \pm\omega_0 + 2\pi k$ for the sinusoidal sequence). Hence such sequences are termed as *almost periodic* and can be treated in the frequency domain in almost the same fashion.

Another interesting class of aperiodic power signals is those consisting of a linear combination of complex exponentials with nonharmonically related frequencies $\{\omega_l\}_{l=1}^L$, for example,

$$x(n) = \sum_{l=1}^L X_l e^{j\omega_l n} \quad (2.2.17)$$

Clearly, these signals have discrete (or line) spectra, but the lines are not uniformly distributed on the frequency axis. Furthermore, the distances between the various lines are not harmonically related. We will say that these signals have *discrete nonharmonic spectra*. Note that periodic signals have discrete *harmonic spectra*.

There is yet another class of power signals, for example, the unit-step signal $u(n)$ defined in (2.1.2). The Fourier transform of such signals exists only in the context of the theory of generalized functions, which allows the use of impulse functions in the frequency domain (Papoulis 1977); for example, the Fourier transform of the unit step $u(n)$ is given by

$$\mathcal{F}[u(n)] = \frac{1}{1 - e^{-j\omega}} + \sum_{k=-\infty}^{\infty} \pi \delta(\omega - 2\pi k) \quad (2.2.18)$$

Such signals have *mixed spectra*. The use of impulses also implies that the line spectrum can be represented in the frequency domain as a continuous spectrum by an impulse train. Figure 2.1 provides a classification of deterministic signals (with finite power or energy) in the frequency domain.

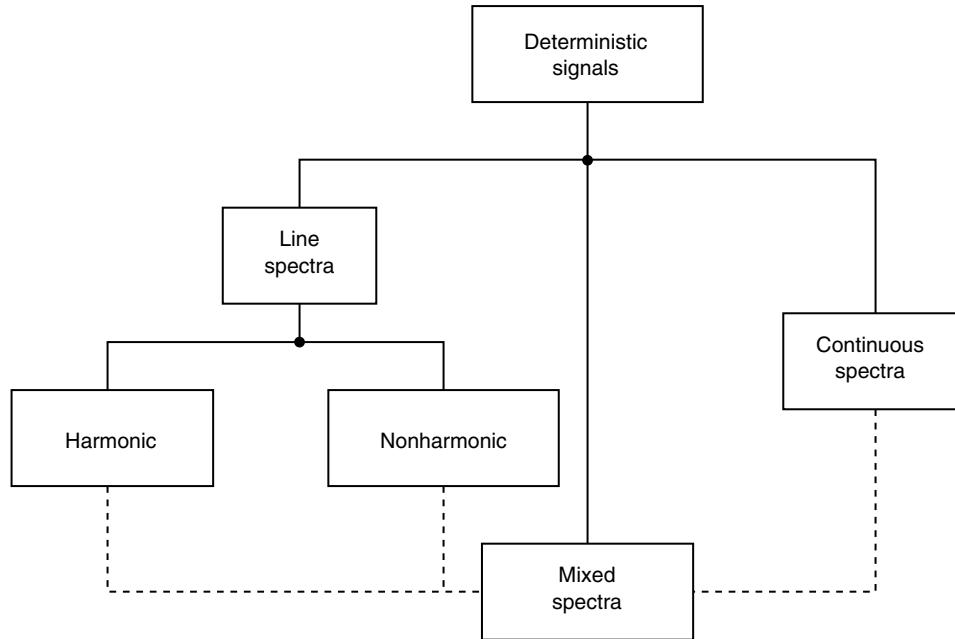


FIGURE 2.1
Spectral classification of deterministic (finite power or energy) signals.

2.2.2 Sampling of Continuous-Time Signals

In most practical applications, discrete-time signals are obtained by sampling continuous-time signals periodically in time. If $x_c(t)$ is a continuous-time signal, the discrete-time signal $x(n)$ obtained by periodic sampling is given by

$$x(n) = x_c(nT) \quad -\infty < n < \infty \quad (2.2.19)$$

where T is the *sampling period*. The quantity $F_s = 1/T$, the number of samples taken per unit of time, is called the *sampling rate* or *sampling frequency*.

Since (2.2.19) established a relationship between the signals $x_c(t)$ and $x(n)$, there should be a corresponding relation between the spectra

$$X_c(F) = \int_{-\infty}^{\infty} x_c(t) e^{-j2\pi F t} dt \quad (2.2.20)$$

and

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n} \quad (2.2.21)$$

of these signals.

To establish a relationship between $X_c(F)$ and $X(e^{j\omega})$, first we need to find a relation between the frequency variables F and ω . To this end, we note that periodic sampling imposes a relationship between t and n , namely, $t = nT = n/F_s$. Substituting $t = n/F_s$ into (2.2.20) and comparing with the exponentials in (2.2.20) and (2.2.21), we see that

$$2\pi \frac{F}{F_s} = \omega = 2\pi f \quad \text{or} \quad f = \frac{F}{F_s} \quad (2.2.22)$$

Since f appears to be a ratio frequency, it is also called a *relative frequency*. The term *normalized frequency* is also sometimes used for the discrete-time frequency variable f .

It can be shown (Proakis and Manolakis 1996; Oppenheim and Schafer 1989) that the spectra $X_c(F)$ of the continuous-time signal and $X(e^{j\omega})$ of the discrete-time signal are related by

$$X(e^{j2\pi F/F_s}) = F_s \sum_{k=-\infty}^{\infty} X_c(F - kF_s) \quad (2.2.23)$$

The right-hand side of (2.2.23) consists of a periodic repetition of the scaled continuous-time spectrum $F_s X_c(F)$ with period F_s . This periodicity is necessary because the spectrum of any discrete-time signal has to be periodic. To see the implications of (2.2.23), let us assume that $X_c(F)$ is band-limited, that is, $X_c(F) = 0$ for $|F| > B$, as shown in Figure 2.2. According to (2.2.23), the spectrum $X(F)$ is the superposition of an infinite number of replications of $X_c(F)$ at integer multiples of the sampling frequency F_s . Figure 2.2(b) illustrates the situation when $F_s \geq 2B$, whereas Figure 2.2(c) shows what happens if $F_s < 2B$. In the latter case, high-frequency components take on the identity of lower frequencies, a phenomenon known as *aliasing*. Obviously, aliasing can be avoided only if the sampled continuous-

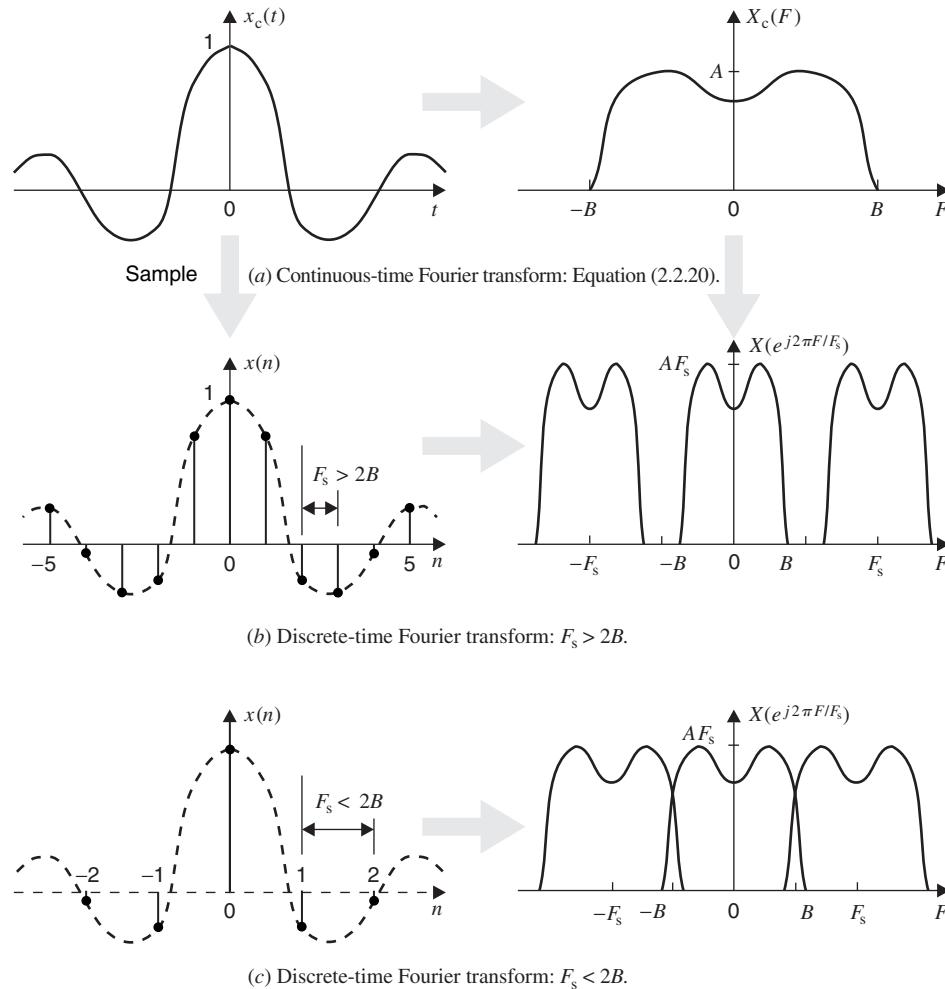


FIGURE 2.2
 Sampling operation.

time signal is band-limited and the sampling frequency F_s is equal to at least twice the bandwidth ($F_s \geq 2B$). This leads to the well-known sampling theorem, which can be stated as follows:

SAMPLING THEOREM. A band-limited, real-valued, continuous-time signal with bandwidth B can be uniquely recovered from its samples, provided that the sampling rate F_s is at least equal to twice the bandwidth, that is, provided that $F_s \geq 2B$.

If the conditions of the sampling theorem are fulfilled, that is, if $X_c(F) = 0$ for $|F| > B$ and $F_s \geq 2B$, then the signal $x_c(t)$ can be recovered from its samples $x(n) = x_c(nT)$ by using the following interpolation formula

$$x_c(t) = \sum_{n=-\infty}^{\infty} x_c(nT) \frac{\sin [(\pi/T)(t-nT)]}{(\pi/T)(t-nT)} \quad (2.2.24)$$

The minimum sampling rate of $F_s = 2B$ is called the *Nyquist rate*. In practice, the infinite summation in (2.2.24) has to be substituted by a finite one. Hence, only approximate reconstruction is possible.

2.2.3 The Discrete Fourier Transform

The N -point *discrete Fourier transform (DFT)* of an N -point sequence $\{x(n), n = 0, 1, \dots, N-1\}$ is defined by[†]

$$\tilde{X}(k) = \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/N)kn} \quad k = 0, 1, \dots, N-1 \quad (2.2.25)$$

The N -point sequence $\{x(n), n = 0, 1, \dots, N-1\}$ can be recovered from its DFT coefficients $\{\tilde{X}(k), k = 0, 1, \dots, N-1\}$ by the following *inverse DFT* formula:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) e^{j(2\pi/N)kn} \quad n = 0, 1, \dots, N-1 \quad (2.2.26)$$

We note that by its definition, the N -point DFT requires or provides information only for N samples of a discrete-time signal. Hence, it does not provide a frequency decomposition of the signal because any discrete-time signal must be specified for all discrete-time instances, $-\infty < n < \infty$. The use of DFT for frequency analysis depends on the signal values outside the interval $0 \leq n \leq N-1$. Depending on these values, we can obtain various interpretations of the DFT. The value of the DFT lies exactly in these interpretations.

DFT of finite-duration signals. Let $x(n)$ be a finite-duration signal with nonzero values over the range $0 \leq n \leq N-1$ and zero values elsewhere. If we evaluate $X(e^{j\omega})$ at N equidistant frequencies, say, $\omega_k = (2\pi/N)k$, $0 \leq k \leq N-1$, we obtain

$$X(e^{j\omega_k}) = X(e^{j2\pi k/N}) = \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/N)kn} = \tilde{X}(k) \quad (2.2.27)$$

which follows by comparing the last equation with (2.2.25). This implies that *the N -point DFT of a finite-duration signal with length N is equal to the Fourier transform of the signal at frequencies $\omega_k = (2\pi/N)k$, $0 \leq k \leq N-1$* . Hence, in this case, the N -point DFT corresponds to the uniform sampling of the Fourier transform of a discrete-time signal at N equidistant points, that is, sampling in the frequency domain.

[†]In many traditional textbooks, the DFT is denoted by $X(k)$. We will use the notation $\tilde{X}(k)$ to distinguish the DFT from the DTFT $X(e^{j\omega})$ function or its samples.

DFT of periodic signals. Suppose now that $x(n)$ is a periodic sequence with fundamental period N . This sequence can be decomposed into frequency components by using the Fourier series in (2.2.8) and (2.2.9). Comparison of (2.2.26) with (2.2.8) shows that

$$\tilde{X}(k) = NX_k \quad k = 0, 1, \dots, N - 1 \quad (2.2.28)$$

that is, *the DFT of one period of a periodic signal is given by the Fourier series coefficients of the signal scaled by the fundamental period*. Obviously, computing the DFT of a fraction of a period will lead to DFT coefficients that are not related to the Fourier series coefficients of the periodic signal.

The DFT can be efficiently computed by using a family of fast algorithms, referred to as *fast Fourier transform (FFT)* algorithms, with complexity proportional to $N \log_2 N$. Due to the efficiency offered by these algorithms, the DFT is widely used for the computation of spectra, correlations, and convolutions and for the implementation of digital filters.

2.2.4 The z -Transform

The z -transform of a sequence is a very powerful tool for the analysis of linear and time-invariant systems. It is defined by the following pair of equations:

$$X(z) \triangleq \mathcal{Z}[x(n)] = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \quad (2.2.29)$$

$$x(n) = \frac{1}{2\pi j} \oint_C X(z)z^{n-1} dz \quad (2.2.30)$$

Equation (2.2.29) is known as the *direct transform*, whereas equation (2.2.30) is referred to as the *inverse transform*. The set of values of z for which the power series in (2.2.29) converges is called the *region of convergence (ROC)* of $X(z)$. A sufficient condition for convergence is

$$\sum_{n=-\infty}^{\infty} |x(n)|z^{-n} < \infty \quad (2.2.31)$$

In general, the ROC is a ring in the complex plane; that is, $R_1 < |z| < R_2$. The values of R_1 and R_2 depend on the nature of the signal $x(n)$. For finite-duration signals, $X(z)$ is a polynomial in z^{-1} , and the ROC is the entire z -plane with a possible exclusion of the points $z = 0$ and/or $z = \pm\infty$. For causal signals with infinite duration, the ROC is, in general, $R_1 < |z| < \infty$, that is, the exterior of a circle. For anticausal signals [$x(n) = 0, n > 0$], the ROC is the interior of a circle, that is, $0 < |z| < R_2$. For two-sided infinite-duration signals, the ROC is, in general, a ring $R_1 < |z| < R_2$. The contour of integration in the inverse transform in (2.2.30) can be any counterclockwise closed path that encloses the origin and is inside the ROC.

If we compute the z -transform on the unit circle of the z -plane, that is, if we set $z = e^{j\omega}$ in (2.2.29) and (2.2.30), we obtain

$$X(z)|_{z=e^{j\omega}} = X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \quad (2.2.32)$$

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n} d\omega \quad (2.2.33)$$

which are the Fourier transform and inverse Fourier transform relating the signals $x(n)$ and $X(e^{j\omega})$. This relation holds only if the unit circle is inside the ROC.

The z -transform has many properties that are useful for the study of discrete-time signals and systems. Some of these properties are given in Table 2.1. Assuming that the involved Fourier transform exists, setting $z = e^{j\omega}$ in each of the properties of Table 2.1 gives a corresponding table of properties for the Fourier transform.

An important family of z -transforms is those for which $X(z)$ is a rational function, that is, a ratio of two polynomials in z or z^{-1} . The roots of the numerator polynomial, that is, the values of z for which $X(z) = 0$, are referred to as the *zeros* of $X(z)$. The roots of the denominator polynomial, that is, the values of z for which $|X(z)| = \infty$, are referred to as the *poles* of $X(z)$. Although zeros and poles may occur at $z = 0$ or $z = \pm\infty$, we usually do not count them. As will be seen throughout this book, the locations of poles and zeros play an important role in the analysis of signals and systems. To display poles and zeros in the z -plane, we use the symbols \times and \circ , respectively.

The inverse z -transform—that is, determining the signal $x(n)$ given its z -transform $X(z)$ —involves the computation of the contour integral in (2.2.30). However, most practical applications involve rational z -transforms that can be easily inverted using partial fraction expansion techniques. Finally, we note that a working familiarity with the z -transform technique is necessary for the complete understanding of the material in subsequent chapters.

2.2.5 Representations of Narrowband Signals

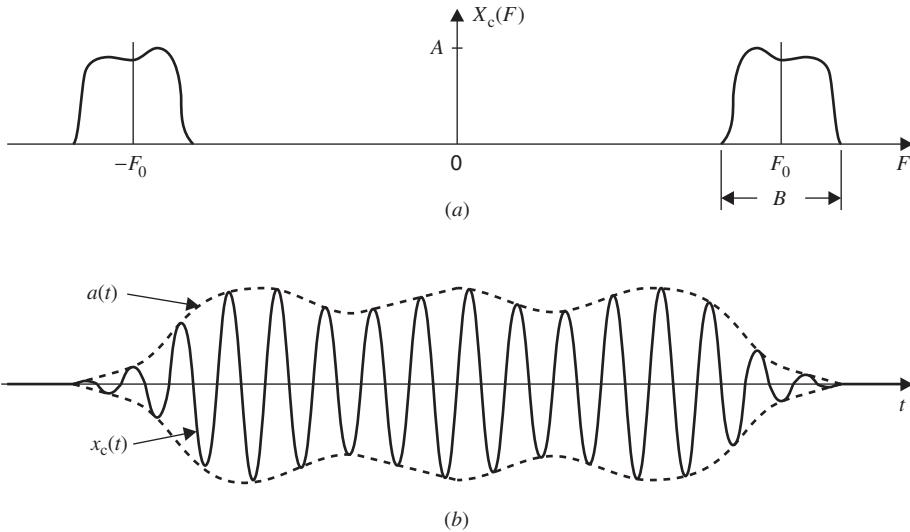
A signal is known as a *narrowband* signal if it is band-limited to a band whose width is small compared to the band center frequency. Such a narrowband signal transform $X_c(F)$ is shown in Figure 2.3(a), and the corresponding signal waveform $x_c(t)$ that it may represent is shown in Figure 2.3(b). The center frequency of $x_c(t)$ is F_0 , and its bandwidth is B , which is much less than F_0 . It is informative to note that the signal $x_c(t)$ appears to be a sinusoidal waveform whose amplitude and phase are both *varying slowly* with respect to the variations of the cosine wave. Therefore, such a signal can be represented by

$$x_c(t) = a(t) \cos [2\pi F_0 t + \theta(t)] \quad (2.2.34)$$

where $a(t)$ describes the amplitude variation (or envelope modulation) and $\theta(t)$ describes the phase modulation of a carrier wave of frequency F_0 Hz. Although (2.2.34) can be used to describe any arbitrary signal, the concepts of envelope and phase modulation are

TABLE 2.1
Properties of z -Transform.

Property	Time domain	z -Domain	ROC
Notation	$x(n)$ $x_1(n)$ $x_2(n)$	$X(z)$ $X_1(z)$ $X_2(z)$	$\text{ROC} : R_l < z < R_u$ $\text{ROC}_1 : R_{1l} < z < R_{1u}$ $\text{ROC}_2 : R_{2l} < z < R_{2u}$
Linearity	$a_1 x_1(n) + a_2 x_2(n)$	$a_1 X_1(z) + a_2 X_2(z)$	$\text{ROC}_1 \cap \text{ROC}_2$
Time shifting	$x(n - k)$	$z^{-k} X(z)$	$R_l < z < R_u$, except $z = 0$ if $k > 0$
Scaling in the z -domain	$a^n x(n)$	$X(a^{-1}z)$	$ a R_l < z < a R_u$
Time reversal	$x(-n)$	$X(z^{-1})$	$\frac{1}{R_l} < z < \frac{1}{R_u}$
Conjugation	$x^*(n)$	$X^*(z^*)$	ROC
Differentiation	$nx(n)$	$-z \frac{dX(z)}{dz}$	ROC
Convolution	$x_1(n) * x_2(n)$	$X_1(z)X_2(z)$	$\text{ROC}_1 \cap \text{ROC}_2$
Multiplication	$x_1(n)x_2(n)$	$\frac{1}{2\pi j} \oint_C X_1(v)X_2\left(\frac{z}{v}\right)v^{-1}dv$	$R_{1l}R_{2l} < z < R_{1u}R_{2u}$
Parseval's relation	$\sum_{n=-\infty}^{\infty} x_1(n)x_2^*(n)$	$\frac{1}{2\pi j} \oint_C X_1(v)X_2^*\left(\frac{1}{v^*}\right)v^{-1}dv$	

**FIGURE 2.3**

Narrowband signal: (a) Fourier transform and (b) waveform.

meaningless unless $a(t)$ and $\theta(t)$ vary slowly in comparison to $\cos 2\pi F_0 t$, or equivalently, unless $B \ll F_0$.

In literature, two approaches are commonly used to describe a narrowband signal. In the first approach, the signal is represented by using a *complex envelope*, while in the second approach the *quadrature component* representation is used. By using Euler's identity, it is easy to verify that (2.2.34) can be put in the form

$$x_c(t) = \operatorname{Re}[a(t)e^{j[2\pi F_0 t + \theta(t)]}] = \operatorname{Re}[a(t)e^{j\theta(t)}e^{j2\pi F_0 t}] \quad (2.2.35)$$

Let

$$\tilde{x}_c(t) \triangleq a(t)e^{j\theta(t)} \quad (2.2.36)$$

Then from (2.2.35) we obtain

$$x_c(t) = \operatorname{Re}[\tilde{x}_c(t)e^{j2\pi F_0 t}] \quad (2.2.37)$$

The complex-valued signal $\tilde{x}_c(t)$ contains both the amplitude and phase variations of $x_c(t)$, and hence it is referred to as the *complex envelope* of the narrowband signal $x_c(t)$. Similarly, again starting with (2.2.34) and this time using the trigonometric identity, we can write

$$x_c(t) = a(t) \cos 2\pi F_0 t \cos \theta(t) - a(t) \sin 2\pi F_0 t \sin \theta(t) \quad (2.2.38)$$

Let

$$x_{cl}(t) \triangleq a(t) \cos \theta(t) \quad (2.2.39)$$

$$x_{cq}(t) \triangleq a(t) \sin \theta(t) \quad (2.2.40)$$

which are termed the *in-phase* and the *quadrature* components of narrowband signal $x_c(t)$, respectively. Then (2.2.38) can be written as

$$x_c(t) = x_{cl}(t) \cos 2\pi F_0 t - x_{cq}(t) \sin 2\pi F_0 t \quad (2.2.41)$$

Clearly, the above two representations are related. If we expand (2.2.36), then we obtain

$$\tilde{x}_c(t) = x_{cl}(t) + j x_{cq}(t) \quad (2.2.42)$$

which implies that the in-phase and quadrature components are, respectively, the real and imaginary parts of the complex envelope $\tilde{x}_c(t)$. These representations will be used extensively in Chapter 11.

Bandpass sampling theorem. One application of the complex-envelope representation lies in the optimum sampling of narrowband signals. In a general sense, the narrowband signal $x_c(t)$ is also a bandpass signal that is approximately band-limited to $(F_0 + B/2)$ Hz.

According the sampling theorem in Section 2.2.2, the Nyquist sampling rate for $x_c(t)$ is then

$$F_s = 2 \left(F_0 + \frac{B}{2} \right) \approx 2F_0 \quad \text{for } B \ll F_0$$

However, since the effective bandwidth of $x_c(t)$ is $B/2$ Hz, the optimum rate should be B , which is much smaller than $2F_0$. To obtain this optimum rate, consider (2.2.34), which we can write as

$$\begin{aligned} x_c(t) &= a(t) \cos [2\pi F_0 t + \theta(t)] = a(t) \frac{e^{j[2\pi F_0 t + \theta(t)]} + e^{-j[2\pi F_0 t + \theta(t)]}}{2} \\ &= \frac{a(t)e^{j\theta(t)}}{2} e^{j2\pi F_0 t} + \frac{a(t)e^{-j\theta(t)}}{2} e^{-j2\pi F_0 t} \\ &= \frac{1}{2} \tilde{x}_c(t) e^{j2\pi F_0 t} + \frac{1}{2} \tilde{x}_c^*(t) e^{-j2\pi F_0 t} \end{aligned} \quad (2.2.43)$$

Using the transform properties from Table 2.1, we see that the Fourier transform of $x_c(t)$ is given by

$$X_c(F) = \frac{1}{2} [\tilde{X}_c(F - F_0) + \tilde{X}_c^*(-F - F_0)] \quad (2.2.44)$$

The first term in (2.2.44) is the Fourier transform of $\tilde{x}_c(t)$ shifted by F_0 , and hence it must be the positive band-limited portion of $X_c(F)$. Similarly, the second term in (2.2.44) is the Fourier transform of $\tilde{x}_c^*(t)$ shifted by $-F_0$ (or shifted left by F_0). Now the Fourier transform of $\tilde{x}_c^*(t)$ is $X_c^*(-F)$, and hence the second term must be the negative band-limited portion of $X_c(F)$.

We thus conclude that $\tilde{x}_c(t)$ is a *baseband* complex-valued signal limited to the band of width B , as shown in Figure 2.4. Furthermore, note that the sampling theorem of Section 2.2.2 is applicable to real- as well as complex-valued signals. Therefore, we can sample the complex envelope $\tilde{x}_c(t)$ at the Nyquist rate of B sampling intervals per second; and, by extension, we can sample the narrowband signal $x_c(t)$ at the same rate without aliasing. From (2.2.24), the sampling representation of $\tilde{x}_c(t)$ is given by

$$\tilde{x}_c(t) = \sum_{n=-\infty}^{\infty} \tilde{x}_c \left(\frac{n}{B} \right) \frac{\sin [\pi B(t - n/B)]}{\pi B(t - n/B)} \quad (2.2.45)$$

Substituting (2.2.45) and (2.2.36) in (2.2.37), we obtain

$$\begin{aligned} x_c(t) &= \operatorname{Re} \left\{ \sum_{n=-\infty}^{\infty} \tilde{x}_c \left(\frac{n}{B} \right) \frac{\sin [\pi B(t - n/B)]}{\pi B(t - n/B)} e^{j2\pi F_0 t} \right\} \\ &= \operatorname{Re} \left\{ \sum_{n=-\infty}^{\infty} a \left(\frac{n}{B} \right) e^{j\theta(n/B)} e^{j2\pi F_0 t} \frac{\sin [\pi B(t - n/B)]}{\pi B(t - n/B)} \right\} \\ &= \sum_{n=-\infty}^{\infty} a \left(\frac{n}{B} \right) \cos \left[2\pi F_0 t + \theta \left(\frac{n}{B} \right) \right] \frac{\sin [\pi B(t - n/B)]}{\pi B(t - n/B)} \end{aligned} \quad (2.2.46)$$

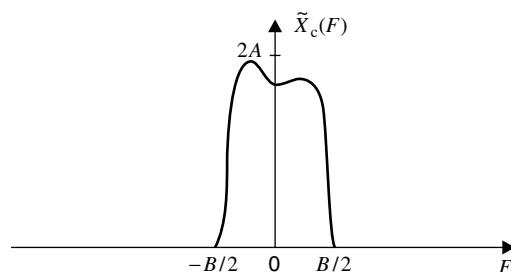


FIGURE 2.4
Fourier transform of a complex envelope $\tilde{x}_c(t)$.

which is the amplitude-phase form of the bandpass sampling theorem. Using trigonometric identity, the quadrature-component form of the theorem is given by

$$x_c(t) = \sum_{n=-\infty}^{\infty} \left[x_{cI} \left(\frac{n}{B} \right) \cos 2\pi F_0 t - x_{cQ} \left(\frac{n}{B} \right) \sin 2\pi F_0 t \right] \frac{\sin [\pi B(t - n/B)]}{\pi B(t - n/B)} \quad (2.2.47)$$

Applications of this theorem are considered in Chapter 11.

2.3 DISCRETE-TIME SYSTEMS

In this section, we review the basics of linear, time-invariant systems by emphasizing those aspects of particular importance to this book. For our purposes, a *system* is defined to be any physical device or algorithm that transforms a signal, called the *input* or *excitation*, into another signal, called the *output* or *response*. When the system is simply an algorithm, it may be realized in either hardware or software. Although a system can be specified from its parts and their functions, it will often turn out to be more convenient to characterize a system in terms of its response to specific signals. The mathematical relationships between the input and output signals of a system will be referred to as a (*system*) *model*. In the case of a *discrete-time system*, the model is simply a transformation that uniquely maps the input signal $x(n)$ to an output signal $y(n)$. This is denoted by

$$y(n) = H[x(n)] \quad -\infty < n < \infty \quad (2.3.1)$$

and is graphically depicted as in Figure 2.5.

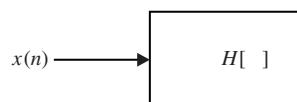


FIGURE 2.5
Block diagram representation of a discrete-time system.

2.3.1 Analysis of Linear, Time-Invariant Systems

The systems we shall deal with in this book are linear and time-invariant and are always assumed to be initially at rest. No initial conditions or other information will affect the output signal.

Time-domain analysis. The output of a linear, time-invariant system can always be expressed as the *convolution* summation between the input sequence $x(n)$ and the *impulse response* or *unit sample response* sequence $h(n) \triangleq H[\delta(n)]$ of the system, that is,

$$y(n) = x(n) * h(n) \triangleq \sum_{k=-\infty}^{\infty} x(k)h(n-k) \quad (2.3.2)$$

where $*$ denotes the convolution operation. It can easily be shown that an equivalent expression is

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) = h(n) * x(n) \quad (2.3.3)$$

Thus, given the input $x(n)$ to a linear, time-invariant system, the output $y(n)$ can be computed by using the impulse response $h(n)$ of the system and either formula (2.3.2) or (2.3.3).

If $x(n)$ and $h(n)$ are arbitrary sequences of finite duration, then the above convolution can also be computed by using a *matrix-vector multiplication* operation. Let $x(n)$, $0 \leq n \leq N-1$, and $h(n)$, $0 \leq n \leq M-1$, be two finite-duration sequences of lengths N

and $M (< N)$ respectively.[†] Then from (2.3.3), the sequence $y(n)$ is also a finite-duration sequence over $0 \leq n \leq L - 1$ with $L \triangleq N + M - 1$ samples. If the samples of $y(n)$ and $h(n)$ are arranged in the column vectors \mathbf{y} and \mathbf{h} , respectively, then from (2.3.3) we obtain

$$\begin{bmatrix} y(0) \\ \vdots \\ \vdots \\ y(M-1) \\ \vdots \\ y(N-1) \\ \vdots \\ y(L-1) \end{bmatrix} = \begin{bmatrix} x(0) & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ x(M-1) & \cdots & \cdots & x(0) \\ \vdots & \ddots & \ddots & \vdots \\ x(N-1) & \cdots & \cdots & x(N-M) \\ 0 & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & & \cdots & 0 & x(N-1) \end{bmatrix} \begin{bmatrix} h(0) \\ h(1) \\ \vdots \\ h(M-1) \end{bmatrix} \quad (2.3.4)$$

or

$$\mathbf{y} = \mathbf{X}\mathbf{h} \quad (2.3.5)$$

where the $L \times M$ matrix \mathbf{X} contains linear shifts in $x(n-k)$ for $n = 0, \dots, N-1$, which are arranged as rows. The matrix \mathbf{X} is termed an *input data* matrix. It has an interesting property that all the elements along any diagonal are equal. Such a matrix is called a *Toeplitz* matrix, and thus \mathbf{X} has a *Toeplitz* structure. Note that the first and the last $M-1$ rows of \mathbf{X} contain zero (or boundary) values. Therefore, the first and the last $M-1$ samples of $y(n)$ contain transient boundary effects. In passing, we note that the vector \mathbf{y} can also be obtained as

$$\mathbf{y} = \mathbf{H}\mathbf{x} \quad (2.3.6)$$

in which \mathbf{H} is a Toeplitz matrix obtained from (2.3.2). However, we will emphasize the approach given in (2.3.5) in subsequent chapters.

MATLAB provides a built-in function called `conv` that computes the convolution of two finite-duration sequences and is invoked by $\mathbf{y} = \text{conv}(\mathbf{h}, \mathbf{x})$. Alternatively, the convolution can also be implemented using (2.3.4) in which the Toeplitz data matrix \mathbf{X} is obtained using the function `toeplitz` (see Problem 2.4).

A system is called *causal* if the present value of the output signal depends only on the present and/or past values of the input signal. Although causality is necessary for the real-time implementation of discrete-time systems, it is not really a problem in off-line applications where the input signal has already been recorded. A necessary and sufficient condition for a linear, time-invariant system to be causal is that the impulse response $h(n) = 0$ for $n < 0$.

Stability is another important system property. There are various types of stability criteria. A system is called *bounded-input bounded-output (BIBO) stable* or simply *stable* if and only if every bounded input, namely, $|x(n)| \leq M_x < \infty$ for all n , produces a bounded output, that is, $|y(n)| \leq M_y < \infty$ for all n . Clearly, unstable systems generate unbounded output signals and, hence, are not useful in practical applications because they will result in an overflow in the output. It can be shown that an LTI system is BIBO stable if and only if

$$\sum_{n=-\infty}^{\infty} |h(n)| < \infty \quad (2.3.7)$$

Transform-domain analysis. In addition to the time-domain convolution approach, the output of a linear, time-invariant system can be determined by using transform techniques. Indeed, by using the convolution property of the z -transform (see Table 2.1), (2.3.2) yields

$$Y(z) = H(z)X(z) \quad (2.3.8)$$

[†]For the purpose of this illustration, we assume that the sequences begin at $n = 0$, but they may have any arbitrary finite duration.

where $X(z)$, $Y(z)$, and $H(z)$ are the z -transforms of the input, output, and impulse response sequences, respectively. The z -transform $H(z) = \mathcal{Z}[h(n)]$ of the impulse response is called the *system function* and plays a very important role in the analysis and characterization of linear, time-invariant systems. If the unit circle is inside the ROC of $H(z)$, the system is stable and $H(e^{j\omega})$ provides its frequency response.

Evaluating (2.3.8) on the unit circle gives

$$Y(e^{j\omega}) = H(e^{j\omega})X(e^{j\omega}) \quad (2.3.9)$$

where $H(e^{j\omega})$ is the *frequency response* function of the system. Since, in general, $H(e^{j\omega})$ is complex-valued, we have

$$H(e^{j\omega}) = |H(e^{j\omega})|e^{j\angle H(e^{j\omega})} \quad (2.3.10)$$

and $|H(e^{j\omega})|$ is the *magnitude response*, and $\angle H(e^{j\omega})$ is the *phase response* of the system. For a system with a real impulse response, $|H(e^{j\omega})|$ has even symmetry and $\angle H(e^{j\omega})$ has odd symmetry. The *group delay* response of a system with frequency response $H(e^{j\omega})$ is defined as

$$\tau(e^{j\omega}) = -\frac{d}{d\omega}\angle H(e^{j\omega}) \quad (2.3.11)$$

and provides a measure of the average delay of the system as a function of frequency.

Systems described by linear, constant-coefficient difference equations. A discrete-time system is called *practically realizable* if it satisfies the following conditions: (1) It requires a finite amount of memory, and (2) the amount of arithmetic operations required for the computation of each output sample is finite. Clearly, any system that does not satisfy either of these conditions cannot be implemented in practice.

If, in addition to being linear and time-invariant, we require a system to be causal and practically realizable, then the most general input/output description of such a system takes the form of a constant-coefficient, linear difference equation

$$y(n) = -\sum_{k=1}^P a_k y(n-k) + \sum_{k=0}^Q d_k x(n-k) \quad (2.3.12)$$

In case the system parameters $\{a_k, d_k\}$ depend on time, the system is linear and *time-varying*. If, however, the system parameters depend on either the input or output signals, then the system becomes *nonlinear*.

By limiting our attention to constant parameters and evaluating the z -transform of both sides of (2.3.12), we obtain

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^Q d_k z^{-k}}{1 + \sum_{k=1}^P a_k z^{-k}} \triangleq \frac{D(z)}{A(z)} \quad (2.3.13)$$

Clearly, a system with a rational system function can be described, within a gain factor, by the locations of its poles and zeros in the complex z -plane

$$H(z) = \frac{D(z)}{A(z)} = G \frac{\prod_{k=1}^Q (1 - z_k z^{-1})}{\prod_{k=1}^P (1 - p_k z^{-1})} \quad (2.3.14)$$

The system described by (2.3.12) or equivalently by (2.3.13) or (2.3.14) is stable if its poles, that is, the roots of the denominator polynomial $A(z)$, are all inside the unit circle.

The difference equation in (2.3.12) is implemented in MATLAB using the `filter` function. In its simplest form, this function is invoked by `y = filter(d,a,x)` where $d = [d_0, d_1, \dots, d_Q]$ and $a = [1, a_1, \dots, a_P]$ are the numerator and denominator coefficient arrays in (2.3.13), respectively.

If the coefficients a_k in (2.3.12) are zero, we have

$$y(n) = \sum_{k=0}^Q d_k x(n-k) \quad (2.3.15)$$

which compared to (2.3.3) yields

$$h(n) = \begin{cases} d_n & 0 \leq n \leq Q \\ 0 & \text{elsewhere} \end{cases} \quad (2.3.16)$$

that is, the system in (2.3.15) has an impulse response with finite duration and is called a *finite impulse response (FIR)* system. From (2.3.13), it follows that the system function of an FIR system is a polynomial in z^{-1} , and thus $H(z)$ has Q trivial poles at $z = 0$ and Q zeros. For this reason, FIR systems are also referred to as *all-zero (AZ)* systems. Figure 2.6 shows a straightforward block diagram realization of the FIR system (2.3.15) in terms of unit delays, adders, and multipliers.

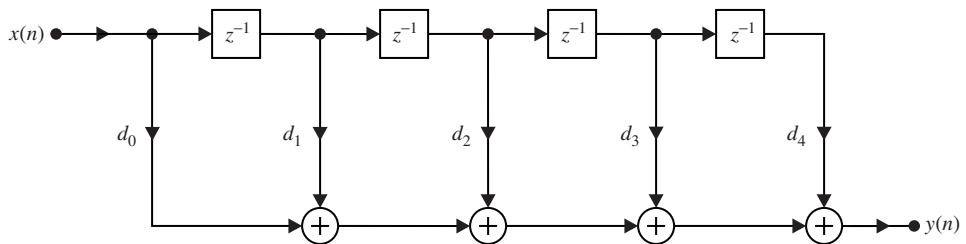


FIGURE 2.6
FIR filter realization (direct form).

In MATLAB, FIR filters are represented either by the values of the impulse response $h(n)$ or by the difference equation coefficients d_n . Therefore, for computational purposes, we can use either the `y = conv(h,x)` function or the `y = filter(d,[1],x)` function. There is a difference in the outputs of these two implementations that should be noted. The `conv` function produces all values of $y(n)$ in (2.3.4), while the output sequence from the filter function provides $y(0), \dots, y(N-1)$. This can be seen by referring to matrix \mathbf{X} in (2.3.4). The input data matrix \mathbf{X} contains only the first N rows; that is, the output of the filter function contains transient effects from the boundary at $n = 0$. For signal processing applications, the use of the `filter` function is strongly encouraged.

When a system has both poles and zeros, $H(z)$ can be expressed using partial fraction expansion form as follows

$$H(z) = \sum_{k=1}^P \frac{A_k}{1 - p_k z^{-k}} \quad (2.3.17)$$

if the poles are distinct and $Q < P$. The corresponding impulse response is then given by

$$h(n) = \sum_{k=1}^P A_k (p_k)^n u(n) \quad (2.3.18)$$

that is, each pole contributes an exponential mode of infinite duration to the impulse response. We conclude that the presence of any nontrivial pole in a system implies an infinite-

duration impulse response. We refer to such systems as *infinite impulse response (IIR)* systems. If $Q = 0$, the system has only poles, with zeros at $z = 0$, and is called an *all-pole (AP)* system. It should be stressed that although all-pole and pole-zero systems are IIR, not all IIR systems are *pole-zero (PZ)* systems. Indeed, there are many useful systems, for example, an ideal low-pass filter, that cannot be described by rational system functions of finite order. Figures 2.7 and 2.8 show direct-form realizations of an all-pole and a pole-zero system.

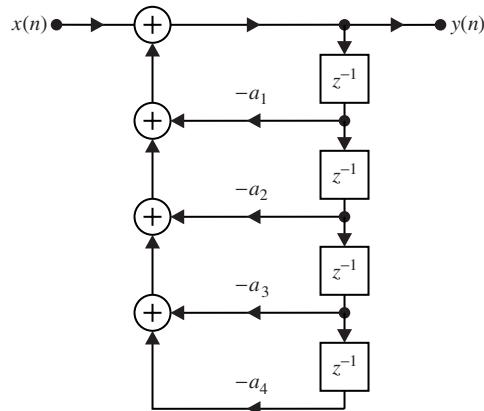


FIGURE 2.7
All-pole system realization (direct form).

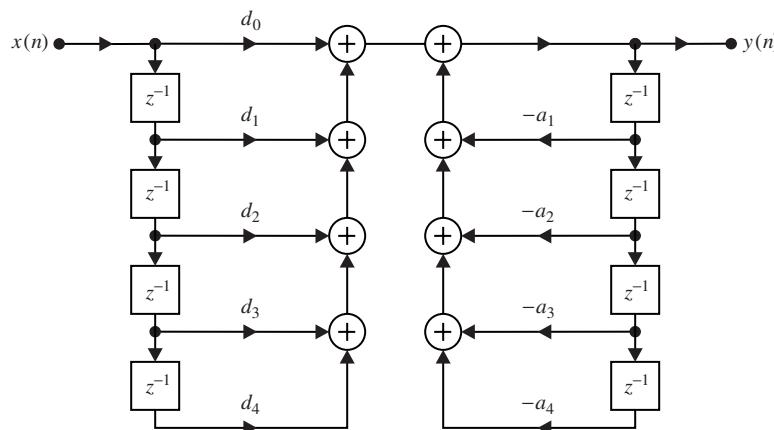


FIGURE 2.8
Pole-zero system realization (direct form).

2.3.2 Response to Periodic Inputs

Although the convolution summation formula can be used to compute the response of a stable system to any input signal, (2.3.8) cannot be used with periodic inputs because periodic signals do not possess a z -transform. However, a frequency domain formula similar to (2.3.9) can be developed for periodic inputs.

Let $x(n)$ be a periodic signal with fundamental period N . This signal can be expanded in a Fourier series as

$$x(n) = \sum_{k=0}^{N-1} X_k e^{j2\pi kn/N} \quad n = 0, 1, \dots, N-1 \quad (2.3.19)$$

where X_k are the Fourier series coefficients. Substituting (2.3.19) into (2.3.3) gives

$$y(n) = \sum_{k=0}^{N-1} X_k H(e^{j2\pi k/N}) e^{j2\pi kn/N} \quad (2.3.20)$$

where $H(e^{j2\pi k/N})$ are samples of $H(e^{j\omega})$. But (2.3.20) is just the Fourier series expansion of $y(n)$, hence

$$Y_k = H(e^{j2\pi k/N}) X_k \quad k = 0, 1, \dots, N-1 \quad (2.3.21)$$

Thus, the response of a linear, time-invariant system to a periodic input is also periodic with the same period. Figure 2.9 illustrates, in the frequency domain, the effect of an LTI system on the spectrum of aperiodic and periodic input signals.

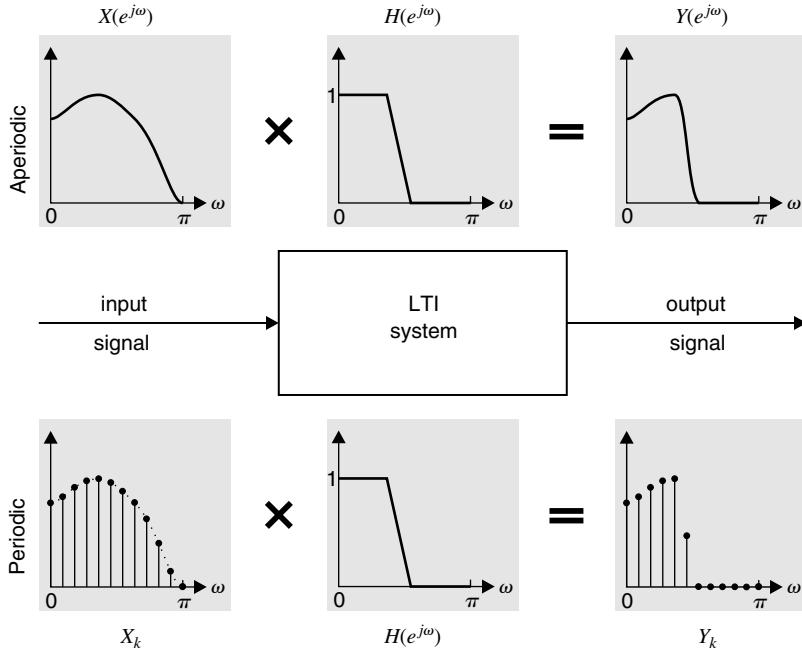


FIGURE 2.9

LTI system operation in the frequency domain.

EXAMPLE 2.3.1. Consider the system

$$y(n) = ay(n-1) + x(n) \quad 0 < a < 1$$

If we restrict the inputs of the system to be only periodic signals with fundamental period N , determine the impulse response of an equivalent FIR system that will provide an identical output to the system described above.

Solution. The system output can be described by (2.3.21), where

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - az^{-1}} = \mathcal{Z}\{a^n u(n)\}$$

From Figure 2.9, it is clearly seen that every system whose frequency response is identical to $H(e^{j\omega})$ at the sampling points $\omega_k = (2\pi/N)k$, $0 \leq k \leq N-1$, provides the same output when excited by a periodic signal having fundamental period N . An FIR system having this property can be obtained by taking the inverse N -point DFT of $\tilde{H}(k)$, $0 \leq k \leq N-1$. The resulting impulse response $\tilde{h}(n)$ is simply the N -point periodic extension of $h(n) = a^n u(n)$, that is,

$$\tilde{h}(n) = \sum_{l=-\infty}^{\infty} h(n+lN) = \sum_{l=0}^{\infty} a^{n+lN} = \frac{a^n}{1 - a^N} \quad 0 \leq n \leq N-1 \quad (2.3.22)$$

since $h(n+lN)$ for $l < 0$ does not contribute to the sum for $0 \leq n \leq N-1$.

The example above looked simple enough. Unfortunately, for somewhat more complicated all-pole filters, it becomes very difficult to evaluate the infinite summation in (2.3.22) in closed form, even if $h(n)$ is available, which is often not the case.

2.3.3 Correlation Analysis and Spectral Density

The investigation of system responses to specific input signals requires either the explicit computation of the output signal or measurements to relate characteristic properties of the output signal to corresponding characteristics of the system and the input signal. A fundamental tool needed for such analysis is the *correlation* between two signals that provides a quantitative measure of similarity between two signals. The *correlation sequence* between two discrete-time signals $x(n)$ and $y(n)$ is defined by

$$r_{xy}(l) = \begin{cases} \sum_{n=-\infty}^{\infty} x(n)y^*(n-l) & : \text{energy signals} \\ \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N x(n)y^*(n-l) & : \text{power signals} \end{cases} \quad (2.3.23)$$

where l is termed the *lag* (or shift) variable. The *autocorrelation* sequence of a signal is obtained by assuming that $y(n) = x(n)$, that is, if we correlate a signal with itself. Thus

$$r_{xx}(l) = \begin{cases} \sum_{n=-\infty}^{\infty} x(n)x^*(n-l) & : \text{energy signal} \\ \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N x(n)x^*(n-l) & : \text{power signal} \end{cases} \quad (2.3.24)$$

In this case, we use the simplified notation $r_x(l)$ or even $r(l)$ if there is no possibility of confusion.

The autocorrelation sequence $r_x(l)$ and the energy spectrum of a signal $x(n)$ form a Fourier transform pair

$$r_x(l) \xleftrightarrow{\mathcal{F}} R_x(e^{j\omega}) \quad (2.3.25)$$

Since, $R_x(e^{j\omega}) = |X(e^{j\omega})|^2$, the Wiener-Khintchine theorem (2.3.25) is usually used to define the energy spectral density function, $R_x(e^{j\omega})$. Clearly, $r_x(l)$ and $R_x(e^{j\omega})$ do not contain any phase information.

In many instances, we need to evaluate the cross-correlation between the input and output signals and the autocorrelation of the output signals. It can be easily shown that

$$r_{yx}(l) = h(l) * r_x(l) \quad (2.3.26)$$

$$r_y(l) = h^*(-l) * r_{yx}(l) = r_h(l) * r_x(l) \quad (2.3.27)$$

$$\text{where } r_h(l) = \sum_{n=-\infty}^{\infty} h(n)h^*(n-l) = h(l) * h^*(-l) \quad (2.3.28)$$

is the autocorrelation of the impulse response. Taking the z -transform of both sides in the above equations, we obtain

$$R_{yx}(z) = H(z)R_x(z) \quad (2.3.29)$$

$$R_y(z) = H^*\left(\frac{1}{z^*}\right)R_{yx}(z) = R_h(z)R_x(z) \quad (2.3.30)$$

$$\text{and } R_h(z) \triangleq H(z)H^*\left(\frac{1}{z^*}\right) \quad (2.3.31)$$

where $R_x(z)$, $R_y(z)$, and $R_h(z)$ are known as *complex spectral density* functions. Evaluating (2.3.30) on the unit circle $z = e^{j\omega}$ gives

$$R_y(e^{j\omega}) = R_h(e^{j\omega})R_x(e^{j\omega}) = |H(e^{j\omega})|^2 R_x(e^{j\omega}) \quad (2.3.32)$$

The output correlations $r_{xy}(l)$ and $r_y(l)$ for a periodic input with fundamental period N are computed via their spectral densities using the Fourier series. For example, it can be easily shown that

$$R_k^{(y)} = |H(e^{j2\pi k/N})|^2 R_k^{(x)} \quad 0 \leq k \leq N - 1 \quad (2.3.33)$$

where $R_k^{(x)}$, $R_k^{(y)}$ are the power spectral densities of $x(n)$ and $y(n)$, respectively.

In exploring the properties of the various system models, we shall need to excite them by some input. Of particular interest are deterministic inputs that have constant power spectrum values (such as the unit sample sequence) or inputs that have constant power spectrum envelopes (such as all-pass signals). Since we have already discussed the unit sample response, we next focus on all-pass signals.

All-pass signals have a flat-spectrum, that is,

$$R_x(e^{j\omega}) = |X(e^{j\omega})|^2 = G^2 \quad -\pi < \omega \leq \pi \quad (2.3.34)$$

and, therefore, $r_x(l) = G^2 \delta(l)$. The simplest example is $x(n) = \delta(n - k)$. A more interesting case is that of all-pass signals with nonlinear phase characteristic (see Section 2.4.2). The autocorrelation and the spectral density of the output $y(n)$ of LTI systems to all-pass excitations can be computed by the formulas used for unit impulse excitations, that is,

$$r_y(l) = G^2 r_h(l) = G^2 \sum_{n=-\infty}^{\infty} h(n)h^*(n-l) \quad (2.3.35)$$

and

$$R_y(z) = G^2 H(z)H^* \left(\frac{1}{z^*} \right) \quad (2.3.36)$$

By properly choosing G , we can always assume that $h(0) = 1$.

2.4 MINIMUM PHASE AND SYSTEM INVERTIBILITY

In this section, we introduce the concept of minimum phase and show how it is related to the invertibility of linear, time-invariant systems. Several properties of all-pass and minimum-phase systems are also discussed.

2.4.1 System Invertibility and Minimum-Phase Systems

A system $H[\cdot]$ with input $x(n)$, $-\infty < n < \infty$, and output $y(n)$, $-\infty < n < \infty$, is called *invertible* if we can uniquely determine its input signal from the output signal. This is possible if the correspondence between the input and output signals is one-to-one. The system that produces $x(n)$, when excited by $y(n)$, is denoted by H_{inv} and is called the *inverse* of system H . Obviously, the cascade of H and H_{inv} is the identity system. Obtaining the inverse of an arbitrary system is a very difficult problem. However, if a system is linear and time-invariant, then if its inverse exists, the inverse is also linear and time-invariant. Hence, if $h(n)$ is the impulse response of a linear, time-invariant system and $h_{\text{inv}}(n)$ that of its inverse, we have

$$[x(n) * h(n)] * h_{\text{inv}}(n) = x(n)$$

or

$$h(n) * h_{\text{inv}}(n) = \delta(n) \quad (2.4.1)$$

Thus, given $h(n)$, $-\infty < n < \infty$, we can obtain $h_{\text{inv}}(n)$, $-\infty < n < \infty$, by solving the convolution equation (2.4.1), which is not an easy task in general. However, (2.4.1) can be

converted to a simpler algebraic equation using the z -transform. Indeed, using the convolution theorem, we obtain

$$H_{\text{inv}}(z) = \frac{1}{H(z)} \quad (2.4.2)$$

where $H_{\text{inv}}(z)$ is the system function of the inverse system. If $H(z)$ is a pole-zero system, that is,

$$H(z) = \frac{D(z)}{A(z)} \quad (2.4.3)$$

then

$$H_{\text{inv}}(z) = \frac{A(z)}{D(z)} \quad (2.4.4)$$

Thus, the zeros of the system become the poles of its inverse, and vice versa. Furthermore, the inverse of an all-pole system is all-zero, and vice versa.

EXAMPLE 2.4.1. Consider a system with impulse response

$$h(n) = \delta(n) - \frac{1}{4}\delta(n-1)$$

Determine impulse response of the inverse system.

Solution. The system function of its inverse is

$$H_{\text{inv}}(z) = \frac{1}{1 - \frac{1}{4}z^{-1}}$$

which has a pole at $z = \frac{1}{4}$. If we choose the ROC as $|z| > \frac{1}{4}$, the inverse system is causal and stable, and

$$h_{\text{inv}}(n) = (\frac{1}{4})^n u(n)$$

However, if we choose the ROC as $|z| < \frac{1}{4}$, the inverse system is noncausal and unstable

$$h_{\text{inv}}(n) = -(\frac{1}{4})^n u(-n-1)$$

This simple example illustrates that the knowledge of the impulse response of a linear, time-invariant system does not uniquely specify its inverse. Additional information such as causality and stability would be helpful in many cases. This leads us to the concept of minimum-phase systems.

A discrete-time, linear, time-invariant system with impulse response $h(n)$ is called *minimum-phase* if both the system and its inverse system $h_{\text{inv}}(n)$ are causal and stable, that is,

$$h(n) * h_{\text{inv}}(n) = \delta(n) \quad (2.4.5)$$

$$h(n) = 0 \quad n < 0 \quad \text{and} \quad h_{\text{inv}}(n) = 0 \quad n < 0 \quad (2.4.6)$$

$$\sum_{n=0}^{\infty} |h(n)| < \infty \quad \text{and} \quad \sum_{n=0}^{\infty} |h_{\text{inv}}(n)| < \infty \quad (2.4.7)$$

We note that if a system is minimum-phase, its inverse is also minimum-phase. This is very important in deconvolution problems, where the inverse system has to be causal and stable for implementation purposes.

Sometimes, especially in geophysical applications, the stability requirements (2.4.7) are replaced by the less restrictive[†] finite energy conditions

$$\sum_{n=0}^{\infty} |h(n)|^2 < \infty \quad \text{and} \quad \sum_{n=0}^{\infty} |h_{\text{inv}}(n)|^2 < \infty \quad (2.4.8)$$

which are implied by (2.4.7). However, note that (2.4.8) does not necessarily imply (2.4.7).

[†]This definition of minimum phase allows singularities (poles or zeros) on the unit circle.

Clearly, a PZ system is minimum-phase if all its poles and zeros are inside the unit circle. Indeed, if all roots of $A(z)$ and $D(z)$ are inside the unit circle, the system $H(z)$ in (2.4.3) and its inverse $H_{\text{inv}}(z)$ in (2.4.4) are both causal and stable.

In an analogous manner, we can define a *maximum-phase system* as one in which both the system and its inverse are anticausal and stable. A PZ system then is maximum-phase if all its poles and zeros are outside the unit circle. Clearly, if $H(z)$ is minimum-phase, then $H(z^{-1})$ is maximum-phase. A system that is neither minimum-phase nor maximum-phase is called a *mixed-phase* system.

2.4.2 All-Pass Systems

We shall say that a linear, time-invariant system is *all-pass*, denoted by $H_{\text{ap}}(e^{j\omega})$, if

$$|H_{\text{ap}}(e^{j\omega})| = 1 \quad -\pi < \omega \leq \pi \quad (2.4.9)$$

The simplest all-pass system is characterized by

$$H_{\text{ap}}(z) = z^k$$

which simply time-shifts (delay $k < 0$, advance $k > 0$) the input signal.

A more interesting, nontrivial family of all-pass systems is characterized by the system function (dispersive all-pass systems)

$$H_{\text{ap}}(z) = \frac{a_P^* + a_{P-1}^* z^{-1} + \cdots + z^{-P}}{1 + a_1 z^{-1} + \cdots + a_P z^{-P}} = \frac{z^{-P} A^*(1/z^*)}{A(z)} \quad (2.4.10)$$

Indeed, it can be easily seen that

$$|H_{\text{ap}}(e^{j\omega})|^2 = H_{\text{ap}}(z) H_{\text{ap}}^*(z) \Big|_{z=e^{j\omega}} = 1 \quad (2.4.11)$$

In the case of real-valued coefficients, (2.4.10) takes the form

$$H_{\text{ap}}(z) = \frac{a_P + a_{P-1} z^{-1} + \cdots + z^{-P}}{1 + a_1 z^{-1} + \cdots + a_P z^{-P}} = \frac{z^{-P} A(z^{-1})}{A(z)} \quad (2.4.12)$$

The poles and zeros of an all-pass system are conjugate reciprocals of one another; that is, they are conjugate symmetric with respect to the unit circle. Indeed, if p_0 is a root of $A(z)$, then $1/p_0^*$ is a root of $A^*(1/z^*)$. Thus, if $p_0 \triangleq r e^{j\theta}$ is a pole of $H_{\text{ap}}(z)$, then $1/p_0^* = (1/r)e^{-j\theta}$ is a zero of the system. This typical pattern is illustrated in Figure 2.10

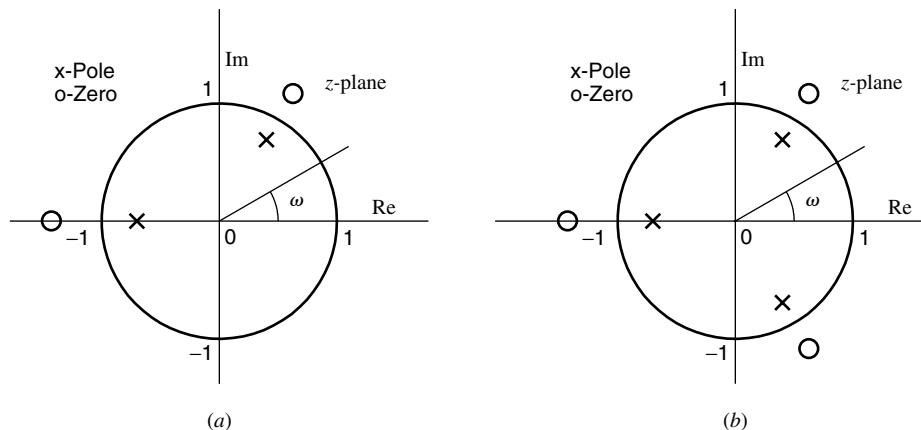


FIGURE 2.10

Typical pole-zero patterns of a PZ, all-pass system: (a) complex-valued coefficients and (b) real-valued coefficients.

for system functions with both complex and real coefficients. Therefore, the system function of any pole-zero all-pass system can be expressed as

$$H_{\text{ap}}(z) = \prod_{k=1}^P \frac{p_k^* - z^{-1}}{1 - p_k z^{-1}} \quad (2.4.13)$$

The similar expressions $(z^{-1} - p_k^*)/(1 - p_k z^{-1})$ and $(1 - p_k z^{-1})/(z^{-1} - p_k^*)$ [the negative and inverse of (2.4.13), respectively] are often used in the literature. For systems with real parameters, singularities should appear in complex conjugate pairs.

Properties of all-pass systems. All-pass systems have some interesting properties. We list these properties without proofs. Some of these proofs are trivial, and others are explored in problems.

1. The output energy of a stable all-pass system is equal to the input energy; that is,

$$E_y = \sum_{n=-\infty}^{\infty} |y(n)|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H_{\text{ap}}(e^{j\omega}) X(e^{j\omega}) \right|^2 d\omega = E_x \quad (2.4.14)$$

due to (2.4.9). This leads to a very interesting property for the cumulative energy of a causal all-pass system (see Problem 2.6).

2. A causal, stable, PZ, all-pass system with P poles has a phase response $\angle H_{\text{ap}}(e^{j\omega})$ that decreases monotonically from $\angle H_{\text{ap}}(e^{j0})$ to $\angle H_{\text{ap}}(e^{j0}) - 2\pi P$ as ω increases from 0 to 2π (see Problem 2.7).
3. All-pass systems have nonnegative group delay, which is defined as the negative of the first derivative of the phase response, that is,

$$\tau_{\text{ap}}(\omega) \triangleq -\frac{d}{d\omega} \angle H_{\text{ap}}(e^{j\omega}) \geq 0 \quad (2.4.15)$$

This property is a direct result of the second property.

4. The all-pass system function $H_{\text{ap}}(z)$

$$H_{\text{ap}}(z) = \frac{1 - \alpha z^{-1}}{z^{-1} - \alpha^*} \quad |\alpha| < 1 \quad (2.4.16)$$

satisfies

$$|H_{\text{ap}}(z)| = \begin{cases} < 1 & \text{if } |z| < 1 \\ = 1 & \text{if } |z| = 1 \\ > 1 & \text{if } |z| > 1 \end{cases} \quad (2.4.17)$$

For proof see Problem 2.10.

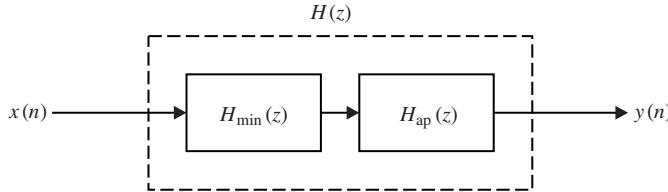
2.4.3 Minimum-Phase and All-Pass Decomposition

We next show that any causal, PZ system that has no poles or zeros on the unit circle can be expressed as

$$H(z) = H_{\text{min}}(z) H_{\text{ap}}(z) \quad (2.4.18)$$

where $H_{\text{min}}(z)$ is minimum-phase and $H_{\text{ap}}(z)$ is all-pass, as shown in Figure 2.11. Indeed, let $H(z)$ be a non-minimum-phase system with one zero $z = 1/a$, $|a| < 1$, outside the unit circle and all other poles and zeros inside the unit circle. Then $H(z)$ can be factored as

$$H(z) = H_1(z)(a - z^{-1}) \quad (2.4.19)$$

**FIGURE 2.11**

Minimum phase and all-pass decomposition.

where $H_1(z)$ is minimum-phase. Equivalently, (2.4.19) can be expressed as

$$\begin{aligned}
 H(z) &= H_1(z)(a - z^{-1}) \frac{1 - a^*z^{-1}}{1 - a^*z^{-1}} \\
 &= [H_1(z)(1 - a^*z^{-1})] \frac{a - z^{-1}}{1 - a^*z^{-1}} \\
 &= H_{\min}(z) \frac{a - z^{-1}}{1 - a^*z^{-1}}
 \end{aligned} \tag{2.4.20}$$

where $H_{\min}(z)$ is minimum-phase and the factor $(a - z^{-1})/(1 - a^*z^{-1})$ is all-pass, because $|a| < 1$. Note that the minimum-phase system was obtained from $H(z)$ by reflecting the zero $z = 1/a$, which was outside the unit circle, to the zero $z = a^*$ inside the unit circle. This approach can clearly be generalized for any PZ system. Thus, given a non-minimum-phase PZ system, we can create a minimum-phase one with the same magnitude response (or equivalently the same impulse response autocorrelation) by reflecting all poles and zeros that are outside the unit circle inside the unit circle. From the previous discussion it follows that there are 2^Q Q th-order AZ systems with the same magnitude response. This is illustrated in the following example.

EXAMPLE 2.4.2. For $Q = 2$, determine all four second-order AZ systems with the same magnitude response.

Solution. For a second-order all-zero system ($0 < a < 1, 0 < b < 1$) we obtain the following systems

$$\begin{aligned}
 H_{\min}(z) &= (1 - az^{-1})(1 - bz^{-1}) & H_{\max}(z) &= (1 - az)(1 - bz) \\
 H_{\text{mix}1}(z) &= (1 - az)(1 - bz^{-1}) & H_{\text{mix}2}(z) &= (1 - az^{-1})(1 - bz)
 \end{aligned} \tag{2.4.21}$$

that have the same spectrum

$$R(z) = H(z)H(z^{-1}) = (1 - az^{-1})(1 - bz^{-1})(1 - az)(1 - bz) \tag{2.4.22}$$

and the same autocorrelation

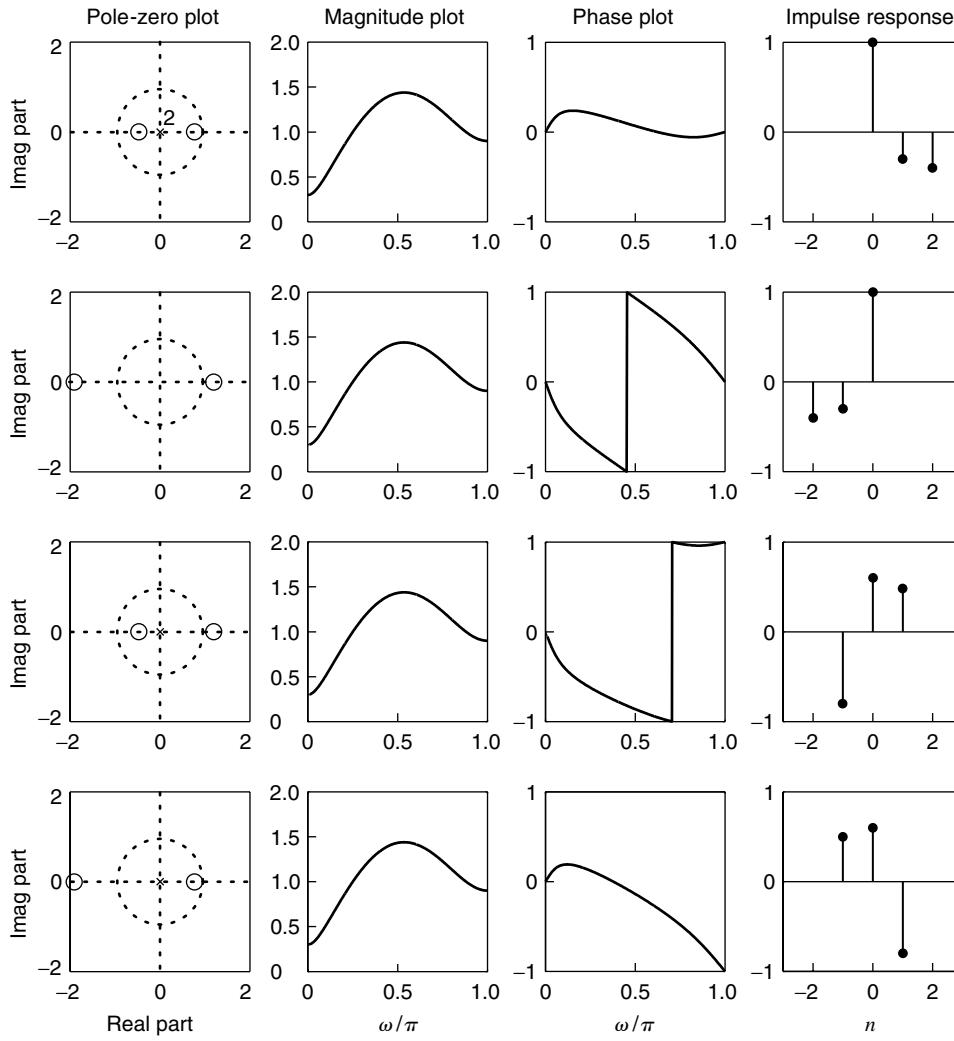
$$r(l) = \begin{cases} 1 + a^2b^2 + (a + b)^2 & l = 0 \\ -(a + b)(1 + ab) & l = 1, -1 \\ ab & l = 2, -2 \\ 0 & \text{otherwise} \end{cases} \tag{2.4.23}$$

but different impulse and phase responses, as shown in Figure 2.12.

EXAMPLE 2.4.3. Consider the following all-zero minimum-phase system:

$$\begin{aligned}
 H_{\min}(z) &= (1 - 0.8e^{j0.6\pi}z^{-1})(1 - 0.8e^{-j0.6\pi}z^{-1}) \\
 &\quad \times (1 - 0.8e^{j0.9\pi}z^{-1})(1 - 0.8e^{-j0.9\pi}z^{-1})
 \end{aligned} \tag{2.4.24}$$

Determine the maximum- and mixed-phase systems with the same magnitude response.

**FIGURE 2.12**

Pole-zero, frequency response, and impulse response plots for minimum-phase (row 1), maximum-phase (row 2), mixed-phase 1 (row 3), and mixed-phase 2 (row 4) systems in Example 2.4.2. Note that the abscissa in Phase plots are labeled in units of π radians.

Solution. To obtain a maximum-phase system with the same magnitude response, we reflect the zeros of $H_{\min}(z)$ from inside the unit circle to their conjugate reciprocal locations that are outside the unit circle by using the transformation $z_0 \rightarrow 1/z_0^*$. This leads to the following transformation for each first-order factor:

$$(1 - re^{j\theta} z^{-1}) \rightarrow r(1 - \frac{1}{r}e^{j\theta} z^{-1}) \quad (2.4.25)$$

The scaling factor r in the right-hand side is included to guarantee that the transformation does not scale the magnitude response. The resulting maximum-phase system is

$$\begin{aligned} H_{\max}(z) = & (0.8)^4 (1 - 1.25e^{j0.6\pi} z^{-1}) (1 - 1.25e^{-j0.6\pi} z^{-1}) \\ & \times (1 - 1.25e^{j0.9\pi} z^{-1}) (1 - 1.25e^{-j0.9\pi} z^{-1}) \end{aligned} \quad (2.4.26)$$

If we reflect only the zero at $0.8e^{\pm j0.6\pi}$, we obtain the mixed-phase system

$$\begin{aligned} H_1(z) = & (0.8)^2 (1 - 1.25e^{j0.6\pi} z^{-1}) (1 - 1.25e^{-j0.6\pi} z^{-1}) \\ & \times (1 - 0.8e^{j0.9\pi} z^{-1}) (1 - 0.8e^{-j0.9\pi} z^{-1}) \end{aligned} \quad (2.4.27)$$

Similarly, if we reflect only the zero at $0.8e^{\pm j0.9\pi}$, we obtain the second mixed-phase system

$$H_2(z) = (0.8)^2 (1 - 0.8e^{j0.6\pi} z^{-1}) (1 - 0.8e^{-j0.6\pi} z^{-1}) \\ \times (1 - 1.25e^{j0.9\pi} z^{-1}) (1 - 1.25e^{-j0.9\pi} z^{-1}) \quad (2.4.28)$$

Figure 2.13 shows the pole-zero, magnitude response, phase response, and group delay plots for all four systems. Clearly, the minimum-phase system has the smallest group delay, the maximum-phase system has the largest group delay, while the mixed-phase systems have in-between amounts of group delay across all frequencies. Finally, it can be easily shown that the system $H_{\max}(z)/H_{\min}(z)$ is an all-pass system.

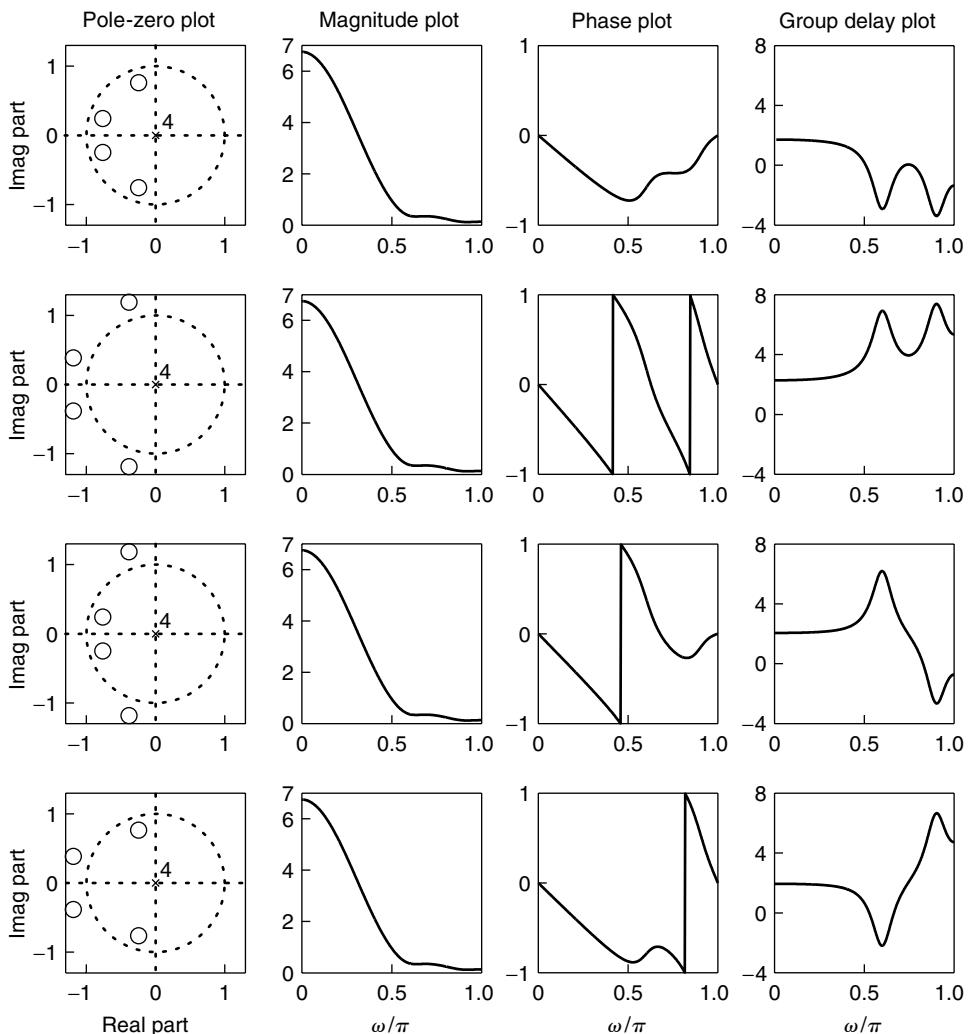


FIGURE 2.13

Pole-zero and frequency response plots for minimum-phase (row 1), maximum-phase (row 2), mixed-phase 1 (row 3), and mixed-phase 2 (row 4) systems in Example 2.4.3. Note that the abscissa in Phase plots are labeled in units of π radians while those in Group delay plots are labeled in sampling intervals.

The minimum- (maximum-) phase AZ system has all its zeros inside (outside) the unit circle. From (2.4.12), it follows that an all-pass system can be expressed as

$$H_{\text{ap}}(z) = \frac{H_{\max}(z)}{H_{\min}(z)} \quad (2.4.29)$$

where $H_{\min}(z)$ and $H_{\max}(z)$ are the P th-order minimum-phase and maximum-phase systems, respectively, with the same magnitude response. Indeed, it can be easily seen that

$$H_{\max}(z) = z^{-P} H_{\min}^* \left(\frac{1}{z^*} \right) \quad (2.4.30)$$

or $h_{\max}(n) = h_{\min}^*(P - n)$.

In practice, it is very important to find out if a given system is minimum-phase. Clearly, the definition cannot be used in practice because either the system $h(n)$ or its inverse is going to be IIR. Furthermore, most of the above properties using either $h(n)$ or $H(e^{j\omega})$ are not practical for use in real-world systems. However, if we deal with PZ systems, we can check if they are minimum-phase by computing the poles and zeros and check if they are inside the unit circle. This is, however, a computationally expensive procedure, especially for high-order systems. Fortunately, there are several tests that allow us to find out if the zeros of a polynomial are inside the unit circle without computing them. See Theorem 2.3.

Properties of minimum-phase systems. Minimum-phase systems have some very interesting properties. Next we list some of these properties without proofs. More details can be found in Oppenheim and Schafer (1989) and Proakis and Manolakis (1996).

1. For causal, stable systems with the same magnitude response, the minimum-phase system has *algebraically* the smallest group delay response at every frequency, that is, $\tau_{\min}(e^{j\omega}) \leq \tau(e^{j\omega})$, for all ω . Thus, strictly speaking, minimum-phase systems are *minimum group delay systems*. However, the term *minimum-phase* has been established in the engineering literature.
2. Of all causal and stable systems with the same magnitude response, the minimum-phase system minimizes the “energy delay”

$$\sum_{n=k}^{\infty} |h(n)|^2 \quad \text{for all } k = 0, 1, \dots, \infty \quad (2.4.31)$$

where $h(n)$ is the system impulse response.

3. The system $H(z)$ is minimum-phase if $\log |H(e^{j\omega})|$ and $\angle H(e^{j\omega})$ form a Hilbert transform pair.

EXAMPLE 2.4.4. In this example we illustrate the energy delay property of minimum-phase systems. Consider the all-zero minimum-phase system (2.4.24) given in Example 2.4.3 and repeated here:

$$\begin{aligned} H_{\min}(z) &= (1 - 0.8e^{j0.6\pi}z^{-1})(1 - 0.8e^{-j0.6\pi}z^{-1}) \\ &\quad \times (1 - 0.8e^{j0.9\pi}z^{-1})(1 - 0.8e^{-j0.9\pi}z^{-1}) \end{aligned}$$

In the top row of four plots in Figure 2.14, we depict the impulse responses of the minimum-, maximum-, and mixed-phase systems. The bottom plot contains the graph of the energy delay $\sum_{n=k}^{\infty} |h(n)|^2$ for $k = 0, 1, \dots, 4$, for each of the systems. As expected, the minimum-phase system has the least amount of energy delay while the maximum-phase system has the greatest amount of energy delay at each n . The graphs of the energy delays for mixed-phase systems are somewhere in between the above two graphs.

Additional properties of minimum-phase systems are explored in the problems.

2.4.4 Spectral Factorization

One interesting and practically useful question is the following: Can we completely determine the system $H(z)$ when $|R_x(e^{j\omega})|^2 = \sigma^2$ given $r_y(l)$ or, equivalently, the spectral density $R_y(e^{j\omega})$? The answer is not a unique one since all we know either from $r_y(l)$ or from $R_y(e^{j\omega})$ is the magnitude response $|H(e^{j\omega})|$, but not the phase response $\angle H(e^{j\omega})$. To obtain a unique system from (2.3.35) or (2.3.36), we have to impose additional

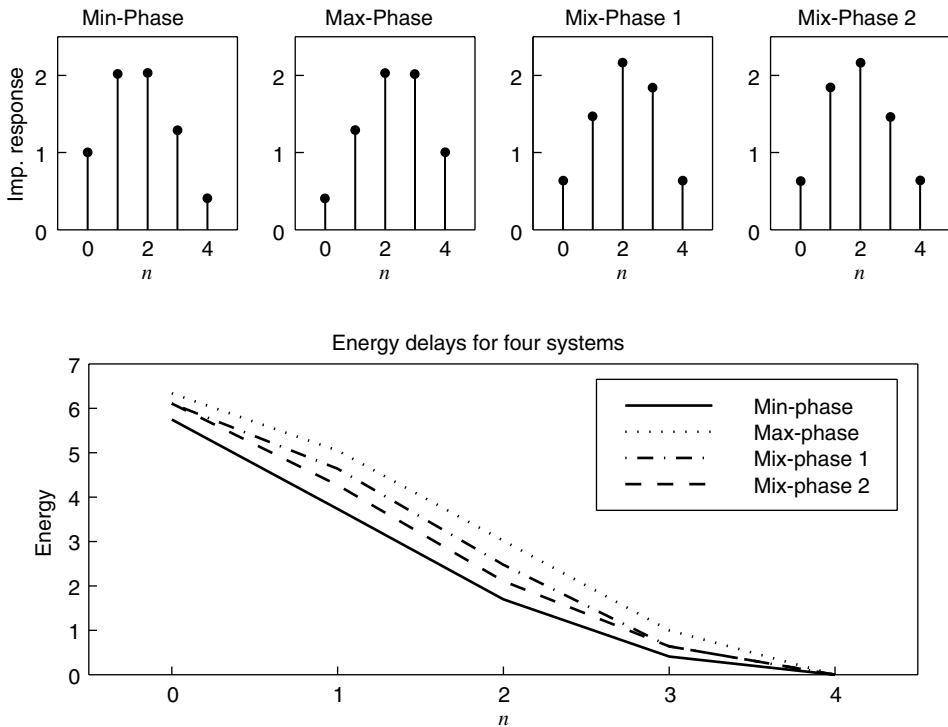


FIGURE 2.14

Impulse response plots of the four systems in the top row and the energy delay plots in the bottom row in Example 2.4.4.

conditions on $H(z)$. One such condition is that of a minimum-phase system. The process of obtaining the minimum-phase system that produces the signal $y(n)$ with autocorrelation $r_y(l)$ or spectral density $R_y(z)$ is called *spectral factorization*. Equivalently, the spectral factorization problem can be stated as the determination of a minimum-phase system from its magnitude response or from the autocorrelation of its impulse response.

Solving the spectral factorization problem by finding roots of $R_y(z)$ is known as the *root method*, and besides its practical utility, it illustrates some basic principles.

1. Every rational power spectral density has, within a scale factor, a unique minimum-phase factorization.
2. There are 2^{P+Q} rational systems with the same power spectral density, where Q and P are numerator and denominator polynomial degrees, respectively.
3. Not all possible rational functions are valid power spectral densities since for a valid $R_y(z)$ the roots should appear in pairs, z_k and $1/z_k^*$.

These principles can be generalized to any power spectral density by extending $P + Q \rightarrow \infty$. The spectral factorization procedure is guaranteed by the following theorem.

THEOREM 2.1. If $\ln R_y(z)$ is analytic in an open ring $\alpha < |z| < 1/\alpha$ in the z -plane and the ring includes the unit circle, then $R_y(z)$ can be factored as

$$R_y(z) = G^2 H_{\min}(z) H_{\min}^*(\frac{1}{z^*}) \quad (2.4.32)$$

where $H_{\min}(z)$ is a minimum-phase system.

Proof. Using the analyticity of $\ln R_y(z)$, we can expand $\ln R_y(z)$ in a Laurent series (Churchill and Brown 1984) as

$$\ln R_y(z) = \sum_{-\infty}^{\infty} g(l) z^{-l} \quad (2.4.33)$$

where the sequence $g(l)$ is known as the *cepstrum* of the sequence $r_y(l)$ (Oppenheim and Schafer 1989). Evaluating (2.4.33) on the unit circle, we obtain

$$\ln R_y(e^{j\omega}) = \sum_{-\infty}^{\infty} g(l)e^{-j\omega l} \quad (2.4.34)$$

or
$$g(l) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln R_y(e^{j\omega}) e^{j\omega l} d\omega \quad (2.4.35)$$

Since $R_y(e^{j\omega}) = |Y(e^{j\omega})|^2$ is a real, nonnegative function, the sequence $g(l)$ is a conjugate symmetric sequence, that is,

$$g(l) = g^*(-l) \quad (2.4.36)$$

and
$$G^2 \triangleq \exp g(0) = \exp \left[\frac{1}{2\pi} \int_{-\pi}^{\pi} \ln R_y(e^{j\omega}) d\omega \right] \geq 0 \quad (2.4.37)$$

From (2.4.33), we can express $R_y(z)$ in a factored form as

$$\begin{aligned} R_y(z) &= \exp \left[\sum_{-\infty}^{\infty} g(l)z^{-l} \right] = \exp \left[\sum_{-\infty}^{-1} g(l)z^{-l} + g(0) + \sum_{1}^{\infty} g(l)z^{-l} \right] \\ &= \exp g(0) \exp \left[\sum_{1}^{\infty} g(l)z^{-l} \right] \exp \left[\sum_{-\infty}^{-1} g(l)z^{-l} \right] \quad (2.4.38) \\ &= G^2 \exp \left[\sum_{1}^{\infty} g(l)z^{-l} \right] \exp \left[\sum_{1}^{\infty} g^*(l)z^l \right] \end{aligned}$$

where we used (2.4.36). After defining

$$H(z) \triangleq \exp \left[\sum_{1}^{\infty} g(l)z^{-l} \right] \quad |z| > \alpha \quad (2.4.39)$$

so that
$$H^* \left(\frac{1}{z^*} \right) = \exp \left[\sum_{1}^{\infty} g^*(l)z^l \right] \quad |z| < \frac{1}{\alpha} \quad (2.4.40)$$

we obtain the spectral factorization (2.3.36). Furthermore, from (2.4.37) we note that the constant G^2 is equal to the geometric mean of $R_y(e^{j\omega})$. From (2.4.39), note that $H(z)$ is the z -transform of a causal and stable sequence, hence it can be expanded as

$$H(z) = 1 + h(1)z^{-1} + h(2)z^{-2} + \dots \quad (2.4.41)$$

where $h(0) = \lim_{z \rightarrow \infty} H(z) = 1$. Also from (2.4.39) $H(z)$ corresponds to a minimum-phase system so that from (2.4.40) $H^*(1/z^*)$ is a stable, anticausal, and maximum-phase system.

The analyticity of $\ln R_y(z)$ is guaranteed by the Paley-Wiener theorem given below without proof (see Papoulis 1991).

THEOREM 2.2 (PALEY-WIENER THEOREM). The spectral factorization in (2.4.32) is possible if $R_y(z)$ satisfies the Paley-Wiener condition

$$\int_{-\pi}^{\pi} |\ln R_y(e^{j\omega})| d\omega < \infty$$

If $H(z)$ is known to be minimum-phase, the spectral factorization is unique.

In general, the solution of the spectral factorization problem is difficult. However, it is quite simple in the case of signals with rational spectral densities. Suppose that $R_y(z)$ is a rational complex spectral density function. Since $r_y(l) = r_y^*(-l)$ implies that $R_y(z) = R_y^*(1/z^*)$, if z_i is a root, then $1/z_i^*$ is also a root. If z_i is inside the unit circle, then $1/z_i^*$ is outside. To obtain the minimum-phase system $H(z)$ corresponding to $R_y(z)$, we determine

the poles and zeros of $R_y(z)$ and form $H(z)$ by choosing all poles and zeros that are inside the unit circle, that is,

$$H(z) = G \frac{\prod_{k=1}^Q (1 - z_k z^{-1})}{\prod_{k=1}^P (1 - p_k z^{-1})} \quad (2.4.42)$$

where $|z_k| < 1$, $k = 1, 2, \dots, Q$ and $|p_k| < 1$, $k = 1, 2, \dots, P$.

Before we illustrate this by an example, it should be emphasized that for real-valued coefficients $R_y(e^{j\omega})$ is a rational function of $\cos \omega$. Indeed, we have from (2.3.36) and (2.3.13)

$$R_y(z) = G^2 H(z) H^* \left(\frac{1}{z^*} \right) = G^2 \frac{D(z) D^*(1/z^*)}{A(z) A^*(1/z^*)} \quad (2.4.43)$$

$$\text{where } D(z) = \sum_{k=0}^Q d_k z^{-k} \quad \text{and} \quad A(z) = 1 + \sum_{k=1}^P a_k z^{-k} \quad (2.4.44)$$

Clearly, (2.4.43) can be written as

$$R_y(e^{j\omega}) = G^2 \frac{R_d(e^{j\omega})}{R_a(e^{j\omega})} = G^2 \frac{r_d(0) + 2 \sum_{l=1}^Q r_d(l) \cos l\omega}{r_a(0) + 2 \sum_{l=1}^P r_a(l) \cos l\omega} \quad (2.4.45)$$

where $r_d(l) = r_d^*(-l)$ and $r_a(l) = r_a^*(-l)$ are the autocorrelations of the coefficient sequences $\{d_0, d_1, \dots, d_Q\}$ and $\{1, a_1, \dots, a_P\}$, respectively. Since $\cos l\omega$ can be expressed as a polynomial

$$\cos l\omega = \sum_{i=0}^l \alpha_i (\cos \omega)^i$$

it follows that $R_y(e^{j\omega})$ is a rational function of $\cos \omega$.

EXAMPLE 2.4.5. Let

$$R_y(e^{j\omega}) = \frac{1.04 + 0.4 \cos \omega}{1.25 + \cos \omega}$$

Determine the minimum-phase system corresponding to $R_y(e^{j\omega})$.

Solution. Replacing $\cos \omega$ by $(e^{j\omega} + e^{-j\omega})/2$ or directly by $(z + z^{-1})/2$ gives

$$R_y(z) = \frac{1.04 + 0.2z + 0.2z^{-1}}{1.25 + 0.5z + 0.5z^{-1}} = 0.4 \frac{(z+5)(z+0.2)}{(z+2)(z+0.5)}$$

The required minimum-phase system $H(z)$ is

$$H(z) = \frac{z+0.2}{z+0.5} = \frac{1+0.2z^{-1}}{1+0.5z^{-1}} \quad (2.4.46)$$

2.5 LATTICE FILTER REALIZATIONS

In Section 2.3, we described simple FIR and IIR filter realizations using block diagram elements. These realizations are called filter structures for which there are many different types available for implementation (Proakis and Manolakis 1996). In this section, we discuss the lattice and lattice-ladder filters. The lattice filter is an implementation of a digital filter with rational system functions. This structure is used extensively in digital speech processing and in the implementation of adaptive filters, which are discussed in Chapter 10.

In Section 2.3, we discussed a direct-form realization of an AZ filter (see Figure 2.6). In this section, we present *lattice* structures for the realization of AZ filters. These structures will be used extensively throughout this book.

The basic AZ lattice is shown in Figure 2.15. Because the AZ lattice is often used to implement the inverse of an AP filter, we begin our introduction to the lattice by a realization of the AZ filter

$$A(z) = 1 + \sum_{l=1}^P a_l z^{-l} \quad (2.5.1)$$

The lattice in Figure 2.15(a) is the two-multiplier, or Itakura-Saito, lattice. The lattice has P parameters $\{k_m, 1 \leq m \leq P\}$ that map to the a_l direct-form parameters via a recursive relation that is derived below.

At the m th stage of the lattice, shown in Figure 2.15(a), we have the relations

$$f_m(n) = f_{m-1}(n) + k_m g_{m-1}(n-1) \quad 1 \leq m \leq P \quad (2.5.2)$$

$$g_m(n) = k_m^* f_{m-1}(n) + g_{m-1}(n-1) \quad 1 \leq m \leq P \quad (2.5.3)$$

and from Figure 2.15(b), we have

$$f_0(n) = g_0(n) = x(n) \quad (2.5.4)$$

$$y(n) = f_p(n) \quad (2.5.5)$$

Taking the z -transform of $f_m(n)$ and $g_m(n)$, we have

$$F_m(z) = F_{m-1}(z) + k_m z^{-1} G_{m-1}(z) \quad (2.5.6)$$

$$G_m(z) = k_m^* F_{m-1}(z) + z^{-1} G_{m-1}(z) \quad (2.5.7)$$

Dividing both equations by $X(z)$ and denoting the transfer functions from the input $x(n)$ to the outputs of the m th stage by $A_m(z)$ and $B_m(z)$, where

$$A_m(z) \triangleq \frac{F_m(z)}{F_0(z)} \quad B_m(z) \triangleq \frac{G_m(z)}{G_0(z)} \quad (2.5.8)$$

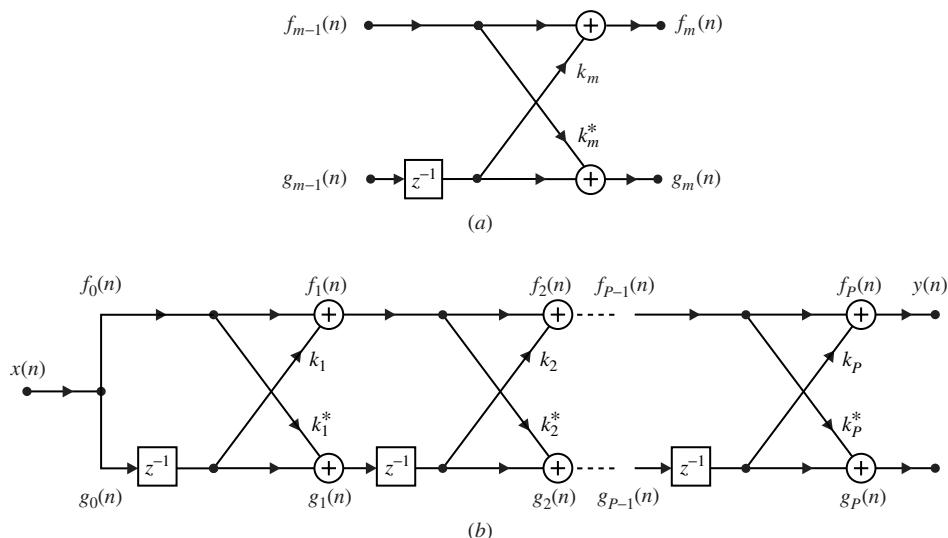


FIGURE 2.15

All-zero lattice structure.

we have

$$A_m(z) = A_{m-1}(z) + k_m z^{-1} B_{m-1}(z) \quad (2.5.9)$$

with

$$B_m(z) = k_m^* A_{m-1}(z) + z^{-1} B_{m-1}(z) \quad (2.5.10)$$

and

$$A_0(z) = B_0(z) = 1 \quad (2.5.11)$$

Thus, the desired $A(z)$ is obtained as the transfer function $A_P(z)$ at the P th stage of the lattice. Now (2.5.9) and (2.5.10) can be written in matrix form as

$$\begin{bmatrix} A_m(z) \\ B_m(z) \end{bmatrix} = \begin{bmatrix} 1 & k_m z^{-1} \\ k_m^* & z^{-1} \end{bmatrix} \begin{bmatrix} A_{m-1}(z) \\ B_{m-1}(z) \end{bmatrix} \quad (2.5.13)$$

$$= \mathbf{Q}_m(z) \begin{bmatrix} A_{m-1}(z) \\ B_{m-1}(z) \end{bmatrix} \quad (2.5.14)$$

where

$$\mathbf{Q}_m(z) \triangleq \begin{bmatrix} 1 & k_m z^{-1} \\ k_m^* & z^{-1} \end{bmatrix} \quad (2.5.15)$$

Then, using the recursive relation (2.5.13), we obtain

$$\begin{bmatrix} A_P(z) \\ B_P(z) \end{bmatrix} = \prod_{m=1}^P \mathbf{Q}_m(z) \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (2.5.16)$$

If we write $A_m(z)$ as

$$A_m(z) = \sum_{l=0}^m a_l^{(m)} z^{-l} \quad (2.5.17)$$

then we can show that

$$a_0^{(m)} = 1 \quad \text{for all } m \quad (2.5.18)$$

$$\text{and that} \quad B_m(z) = \sum_{l=0}^m b_l^{(m)} z^{-l} = z^{-m} A_m^* \left(\frac{1}{z^*} \right) \quad (2.5.19)$$

that is, for $m = 1, 2, \dots, P$

$$b_l^{(m)} = \begin{cases} a_{m-l}^{(m)*} & l = 1, 2, \dots, m-1 \\ 1 & l = m \end{cases} \quad (2.5.20)$$

The polynomial $B_m(z)$ is known as the conjugate *reverse* polynomial of $A_m(z)$ because its coefficients are the conjugates of those of $A_m(z)$ except that they are in reverse order. So since

$$A_m(z) = 1 + a_1^{(m)} z^{-1} + a_2^{(m)} z^{-2} + \dots + a_m^{(m)} z^{-m} \quad (2.5.21)$$

$$\text{then} \quad B_m(z) = a_m^{(m)*} + a_{m-1}^{(m)*} z^{-1} + \dots + a_1^{(m)*} z^{-(m-1)} + z^{-m} \quad (2.5.22)$$

If z_0 is a zero of $A_m(z)$, then z_0^{-1} is a zero of $B_m(z)$. Therefore, if $A_m(z)$ is minimum-phase, then $B_m(z)$ is maximum-phase.

Equations (2.5.19), (2.5.9), and (2.5.10) can be combined into a single equation

$$A_m(z) = A_{m-1}(z) + k_m z^{-m} A_{m-1}^* \left(\frac{1}{z^*} \right) \quad (2.5.23)$$

This equation can be used to derive the following relation between the coefficients at stage m in terms of the coefficients at stage $m-1$:

$$a_l^{(m)} = \begin{cases} 1 & l = 0 \\ a_l^{(m-1)} + k_m a_{m-l}^{(m-1)*} & l = 1, 2, \dots, m-1 \\ k_m & l = m \end{cases} \quad (2.5.24)$$

To solve for the coefficients of the transfer function of the complete P -stage lattice, compute (2.5.24) recursively, starting with $m = 1$ until $m = P$. The final coefficients a_l of the desired filter $A(z)$ are then given by

$$a_l = a_l^{(P)} \quad 0 \leq l \leq P \quad (2.5.25)$$

By substituting $m - l$ for l in (2.5.24), we have

$$a_{m-l}^{(m)} = a_{m-l}^{(m-1)} + k_m a_l^{(m-1)*} \quad (2.5.26)$$

Therefore, $a_l^{(m)}$ and $a_{m-l}^{(m)}$ can be computed simultaneously using $a_l^{(m-1)}$, $a_{m-l}^{(m-1)}$, and k_m .

The lattice parameters k_m can be recovered from the coefficients a_l by a backward recursion. Eliminating $z^{-1} B_{m-1}(z)$ from (2.5.9) and (2.5.10) and using (2.5.19), we obtain

$$A_{m-1}(z) = \frac{A_m(z) - k_m z^{-m} A_m^*(1/z^*)}{1 - |k_m|^2} \quad (2.5.27)$$

The recursion can be started by setting $a_l^{(P)} = a_l$, $0 \leq l \leq P$. Then, with $m = P, P - 1, \dots, 1$, we compute from (2.5.27)

$$\begin{aligned} k_m &= a_m^{(m)} \\ a_l^{(m-1)} &= \begin{cases} 1 & l = 0 \\ \frac{a_l^{(m)} - k_m a_{m-l}^{(m)*}}{1 - |k_m|^2} & 1 \leq l \leq m-1 \end{cases} \end{aligned} \quad (2.5.28)$$

This is the backward recursion to compute k_m from a_l . The computation in (2.5.28) is always possible except when some $|k_m| = 1$. Except for this indeterminate case, the mapping between the lattice parameters k_m and the coefficients a_l of the corresponding all-zero filter is unique.

The MATLAB function `[k] = df2latcf(a)` computes lattice coefficients k_m from polynomial coefficients a_k using (2.5.28). Similarly, the function `[a] = latcf2df(k)` computes the direct-form coefficients from the lattice form.

Although the AZ lattice filters are highly modular, their software implementation is more complex than the direct-form structures. To understand this implementation, we will consider the steps involved in determining one output sample in a P -stage AZ lattice. Assume that $x(n)$ is available over $1 \leq n \leq N$.

Input stage: The describing equation is

$$f_0(n) = g_0(n) = x(n) \quad 1 \leq n \leq N$$

Thus in the implementation, $f_0(n)$ and $g_0(n)$ can be replaced by the input sample $x(n)$, which is assumed to be available in array `x`.

Stage 1: The describing equations are

$$\begin{aligned} f_1(n) &= f_0(n) + k_1 g_0(n-1) = x(n) + k_1 x(n-1) \\ g_1(n) &= k_1^* f_0(n) + g_0(n-1) = k_1^* x(n) + x(n-1) \end{aligned}$$

Assuming that we have two arrays `f` and `g` of length P available to store $f_m(n)$ and $g_m(n)$ at each n , respectively, and two arrays `k` and `ck` of length P to store k_m and k_m^* , respectively, then the MATLAB fragment is

```
f(1) = x(n) + k(1)*x(n-1);
g(1) = ck(1)*x(n) + x(n-1);
```

At $n = 1$, we need $x(0)$ in the above equations. This is an initial condition and is assumed to be zero. Hence in the implementation, we need to augment the `x` array by prepending it with a zero. This should be done in the initialization part. Similarly, arrays `f` and `g` should be initialized to zero.

Stages 2 through P : The describing equations are

$$\begin{aligned}f_m(n) &= f_{m-1}(n) + k_m g_{m-1}(n-1) \\g_m(n) &= k_m^* f_{m-1}(n) + g_{m-1}(n-1)\end{aligned}$$

Note that we need old (i.e., at $n-1$) values of array g in $g_{m-1}(n-1)$. Although it is possible to avoid an additional array, for programming simplicity, we will assume that $g_m(n-1)$ is available in an array g_old of length P . This array should also be initialized to zero. The MATLAB fragment is

```
f(m) = f(m-1) + k(m)*g_old(m-1);  
g(m) = ck*f(m-1) + g_old(m-1);
```

Output stage: The describing equation is

$$y(n) = f_P(n)$$

Also we need to store the current $g_m(n)$ values in the g_old array for use in the calculations of the next output value. Thus the MATLAB fragment is

```
g_old = g;  
y = f(P);
```

Now we can go back to stage 1 with new input value and recursively compute the remaining output values.

The complete procedure is implemented in the function $y = \text{latcfilt}(k, x)$.

2.5.2 All-Pole Lattice Structures

The AZ lattice in Figure 2.15 can be restructured quite simply to yield a corresponding all-pole (AP) lattice structure. Let an AP system function be given by

$$H(z) = \frac{1}{1 + \sum_{l=1}^P a_l z^{-l}} = \frac{1}{A(z)} \quad (2.5.29)$$

which clearly is the *inverse* system of the AZ lattice of Figure 2.15. The difference equation corresponding to (2.5.29) is

$$y(n) + \sum_{l=1}^P a_l y(n-l) = x(n) \quad (2.5.30)$$

If we interchange $x(n)$ with $y(n)$ in (2.5.30), we will obtain the AZ system of (2.5.1). Therefore, the lattice structure of the AP system can be obtained from Figure 2.15(b) by interchanging $x(n)$ with $y(n)$. This lattice structure with P stages is shown in Figure 2.16(b). To determine the m th stage of the AP lattice, we consider (2.5.4) and (2.5.5) and interchange $x(n)$ with $y(n)$. Thus the lattice structure shown in Figure 2.16(b) has

$$f_P(n) = x(n) \quad (2.5.31)$$

as the input and $f_0(n) = g_0(n) = y(n)$ (2.5.32)

as the output. The signal quantities $\{f_m(n)\}_{m=0}^P$ then must be computed in descending order, which can be obtained by rearranging (2.5.2) but not (2.5.3). Thus we obtain

$$f_{m-1}(n) = f_m(n) - k_m g_{m-1}(n-1) \quad (2.5.33)$$

and $g_m(n) = k_m^* f_{m-1}(n) + g_{m-1}(n-1)$ (2.5.34)

These two equations represent the m th stage of the all-pole lattice, shown in Figure 2.16(a), where $f_m(n)$ and $g_{m-1}(n)$ are now the inputs to the m th stage and $f_{m-1}(n)$ and $g_m(n)$ are

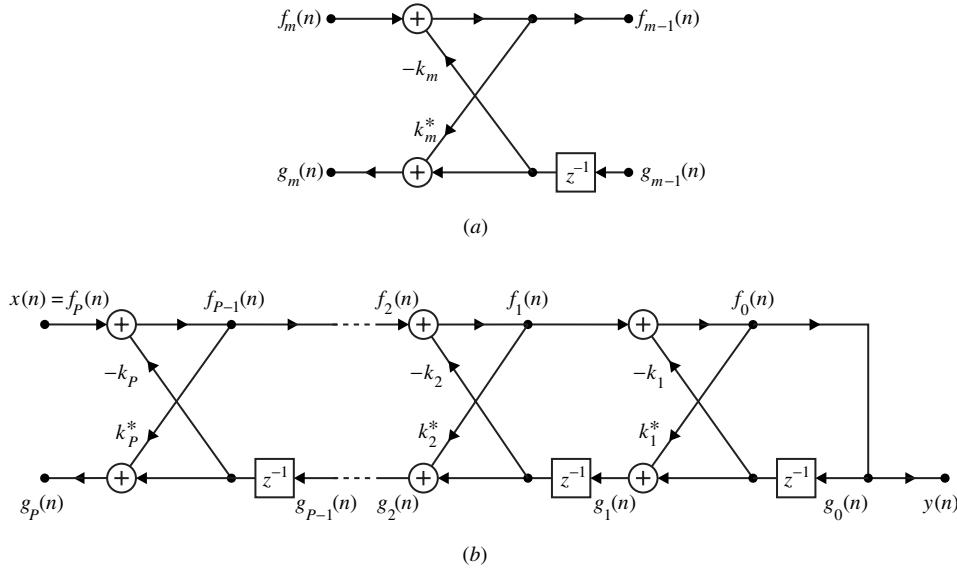


FIGURE 2.16
All-pole lattice structure.

the outputs. The transfer function from the input to the output is the same as that from $f_P(n)$ to $f_0(n)$. This transfer function is the inverse of the transfer function from $f_0(n)$ to $f_P(n)$. From (2.5.8), we conclude that the transfer function from $x(n)$ to $y(n)$ in Figure 2.16 is equal to

$$H(z) = \frac{Y(z)}{X(z)} = \frac{F_0(z)}{F_P(z)} = \frac{1}{A_P(z)} \quad (2.5.35)$$

where $A_P(z) = A(z)$ in (2.5.29). To multiply (2.5.35) by the gain G , we simply multiply either $x(n)$ or $y(n)$ by G in Figure 2.16(b).

Stability of all-pole systems. A causal LTI system is stable if all its poles are inside the unit circle. For all-pole systems described by the denominator polynomial $A_P(z)$, this implies that all its P roots are inside the unit circle, or alternatively, stability implies that $A_P(z)$ is a minimum-phase polynomial. Numerical implementation of polynomial root-finding operation is time-consuming. However, the following theorem shows how the lattice coefficients $\{k_m\}_{m=1}^P$ can be used for stability purposes.

THEOREM 2.3. The polynomial

$$A_P(z) = 1 + a_1^{(P)}z^{-1} + \cdots + a_P^{(P)}z^{-P} \quad (2.5.36)$$

is minimum-phase, that is, has all its zeros inside the unit circle if and only if

$$|k_m| < 1 \quad 1 \leq m \leq P \quad (2.5.37)$$

Proof. See Appendix E.

Therefore, if the lattice parameters k_m in Figure 2.16 are less than unity in magnitude, then the all-pole filter $H(z)$ in (2.5.35) is minimum-phase and stable since $A(z)$ is guaranteed to have all its zeros inside the unit circle.

Since the AP lattice coefficients are derived from the same procedure used for the AZ lattice filter, we can use the $k = \text{df2latcf}(a)$ function in MATLAB. Care must be taken to ignore the k_0 coefficient in the k array. Similarly, the $a = \text{latcf2df}(k)$ function can be used to convert the lattice k_m coefficients to the direct-form coefficients a_k provided that $k_0 = 1$ is used as the first element of the k array.

All-pass lattice

The transfer function from $f_P(n)$ to $g_P(n)$ in Figure 2.16(b) can be written as

$$\frac{G_P(z)}{F_P(z)} = \frac{G_P(z)}{G_0(z)} \frac{F_0(z)}{F_P(z)} \quad (2.5.38)$$

where we used the fact that $F_0(z) = G_0(z)$. From (2.5.8) and (2.5.19), we conclude that

$$\frac{G_P(z)}{F_P(z)} = \frac{B_P(z)}{A_P(z)} = \frac{z^{-P} A^*(1/z^*)}{A(z)} = \frac{a_P^* + a_{P-1}^* z^{-1} + \cdots + z^{-P}}{1 + a_1 z^{-1} + \cdots + a_P z^{-P}} \quad (2.5.39)$$

which is the transfer function of an all-pass filter, since its magnitude on the unit circle is unity at all frequencies.

2.6 SUMMARY

In this chapter we have reviewed the fundamental concepts of discrete-time signal processing in both the time and frequency domains. We introduced usual definitions and descriptions of signals, and we provided the analytical tools for linear system operations. Significant attention was also given to those topics that will be used extensively in the rest of the book. These topics include minimum-phase systems, inverse systems, and spectral factorization. Finally, filters, which will be used in the chapter on adaptive filters, were discussed in greater detail. It is important to grasp the material discussed in this chapter since it is fundamental to understanding concepts presented in the remaining chapters. Therefore, the reader should also consult any one of the widely used references on this subject (Proakis and Manolakis 1996; Oppenheim and Schafer 1989).

PROBLEMS

- 2.1** A continuous-time signal $x_c(t)$ is sampled by an A/D converter to obtain the sequence $x(n)$. It is processed by a digital filter $h(n) = 0.8^n u(n)$ to obtain the sequence $y(n)$, which is further reconstructed using an ideal D/A converter to obtain the continuous-time output $y_c(t)$. The sampling frequency of A/D and D/A converters is 100 sampling intervals per second.
- (a) If $x_c(t) = 2 \cos(40\pi t + \pi/3)$, what is the digital frequency ω_0 in $x(n)$?
 - (b) If $x_c(t)$ is as given above, determine the steady-state response $y_{c,ss}(t)$.
 - (c) Determine two different $x_c(t)$ signals that would give the same steady-state response $y_{c,ss}(t)$ above.

- 2.2** Let $x(n)$ be a sinusoidal sequence of frequency ω_0 and of finite length N , that is,

$$x(n) = \begin{cases} A \cos \omega_0 n & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases}$$

Thus $x(n)$ can be thought of as an infinite-length sinusoidal sequence multiplied by a rectangular window of length N .

- (a) If the DTFT of $x(n)$ is expressed in terms of the real and imaginary parts as

$$X(e^{j\omega}) \triangleq X_R(\omega) + j X_I(\omega)$$

determine analytical expressions for $X_R(\omega)$ and $X_I(\omega)$. Express $\cos \omega$ in terms of complex exponentials and use the modulation property of the DTFT to arrive at the result.

- (b) Choose $N = 32$ and $\omega_0 = \pi/4$, and plot $X_R(\omega)$ and $X_I(\omega)$ for $\omega \in [-\pi, \pi]$.
- (c) Compute the 32-point DFT of $x(n)$, and plot its real and imaginary samples. Superimpose the above DTFT plots on the DFT plots. Comment on the results.
- (d) Repeat the above two parts for $N = 32$ and $\omega_0 = 1.1\pi/4$. Why are the plots so markedly different?

- 2.3** Let $x(n) = \cos(\pi n/4)$, and assume that we have only 16 samples available for processing.
- Compute the 16-point DFT of these 16 samples, and plot their magnitudes. (Make sure that this is a stem plot.)
 - Now compute the 32-point DFT of the sequence formed by appending the above 16 samples with 16 zero-valued samples. This is called *zero padding*. Now plot the magnitudes of the DFT samples.
 - Repeat part (b) for the 64-point sequence by padding 48 zero-valued samples.
 - Explain the effect and hence the purpose of the zero padding operation on the DTFT spectrum.
- 2.4** Let $x(n) = \{1, 2, 3, 4, 3, 2, 1\}$ and $h(n) = \{-1, 0, 1\}$.
- Determine the convolution $y(n) = x(n) * h(n)$ using the matrix-vector multiplication approach given in (2.3.5).
 - Develop a MATLAB function to implement the convolution using the Toeplitz matrix in (2.3.4). The form of the function should be $y = \text{convtoep}(x, h)$.
 - Verify your function, using the sequences given in part (a) above.
- 2.5** Let $x(n) = (0.9)^n u(n)$.
- Determine $x(n) * x(n)$ analytically, and plot its first 101 samples.
 - Truncate $x(n)$ to the first 51 samples. Compute and plot the convolution $x(n) * x(n)$, using the `conv` function.
 - Assume that $x(n)$ is the impulse response of an LTI system. Determine the `filter` function coefficient vectors a and b . Using the `filter` function, compute and plot the first 101 samples of the convolution $x(n) * x(n)$.
 - Comment on your plots. Which MATLAB approach is best suited for infinite-length sequences and why?

- 2.6** Let $H_{ap}(z)$ be a causal and stable all-pass system excited by a causal input $x(n)$ producing the response $y(n)$. Show that for any time n_0 ,

$$\sum_{n=0}^{n_0} |y(n)|^2 \leq \sum_{n=0}^{n_0} |x(n)|^2 \quad (\text{P.1})$$

- 2.7** This problem examines monotone phase-response property of a causal and stable PZ all-pass system.

- (a) Consider the pole-zero diagram of a real first-order all-pass system

$$H(z) = \frac{p - z^{-1}}{1 - pz^{-1}}$$

Show that its phase response decreases monotonically from π (at $\omega = 0$) to $-\pi$ (at $\omega = 2\pi$).

- (b) Consider the pole-zero diagram of a real second-order all-pass system

$$H(z) = \left[\frac{(r \angle \theta) - z^{-1}}{1 - (r \angle \theta)^* z^{-1}} \right] \left[\frac{(r \angle \theta)^* - z^{-1}}{1 - (r \angle \theta) z^{-1}} \right]$$

Show that its phase response decreases monotonically as ω increases from 0 to π .

- (c) Generalize the results of parts (a) and (b) to show that the phase response of a causal and stable PZ all-pass system decreases monotonically from $\angle[H(e^{j0})]$ to $\angle[H(e^{j0})] - 2\pi P$ as ω increases from 0 to π .

- 2.8** This problem explores the minimum group delay property of the minimum-phase systems.

- (a) Consider the following stable minimum-, maximum-, and mixed-phase systems

$$H_{\min}(z) = (1 - 0.25z^{-1})(1 + 0.5z^{-1})$$

$$H_{\max}(z) = (0.25 - z^{-1})(0.5 + z^{-1})$$

$$H_{\text{mix}}(z) = (1 - 0.25z^{-1})(0.5 + z^{-1})$$

which have the same magnitude response. Compute and plot group delay responses. Observe that the minimum-phase system has the minimum group delay.

- (b) Using (2.4.18) and Problem 2.7, prove the minimum group delay property of the minimum-phase systems.

- 2.9** Given the following spectral density functions, express them in minimum- and maximum-phase components.

$$(a) R_y(z) = \frac{1 - 2.5z^{-1} + z^{-2}}{1 - 2.05z^{-1} + z^{-2}}$$

$$(b) R_y(z) = \frac{3z^2 - 10 + 3z^{-2}}{3z^2 + 10 + 3z^{-2}}$$

- 2.10** Consider the all-pass system function $H_{\text{ap}}(z)$ given by

$$H_{\text{ap}}(z) = \frac{1 - \alpha z^{-1}}{z^{-1} - \alpha^*} \quad |\alpha| < 1 \quad (\text{P.2})$$

- (a) Determine $|H_{\text{ap}}(z)|^2$ as a ratio of polynomials in z .

- (b) Show that

$$D_{|H|}^2(z) - A_{|H|}^2(z) = (|z|^2 - 1)(1 - |\alpha|^2)$$

where

$$|H_{\text{ap}}(z)|^2 = \frac{D_{|H|}^2(z)}{A_{|H|}^2(z)}$$

- (c) Using $|\alpha| < 1$ and the above result, show that

$$|H_{\text{ap}}(z)| \begin{cases} < 1 & \text{if } |z| < 1 \\ = 1 & \text{if } |z| = 1 \\ > 1 & \text{if } |z| > 1 \end{cases}$$

- 2.11** Consider the system function of a stable system of the form

$$H(z) = \frac{a + bz^{-1} + cz^{-2}}{c + bz^{-1} + az^{-2}}$$

- (a) Show that the magnitude of the frequency response function $|H(e^{j\omega})|$ is equal to 1 for all frequencies, that is, it is an all-pass system.

- (b) Let

$$H(z) = \frac{3 - 2z^{-1} + z^{-2}}{1 - 2z^{-1} + 3z^{-2}}$$

Determine both the magnitude and the phase of the frequency response $H(e^{j\omega})$, and plot these functions over $[0, \pi]$.

- 2.12** Consider the system function of a third-order FIR system

$$H(z) = 12 + 28z^{-1} - 29z^{-2} - 60z^{-3}$$

- (a) Determine the system functions of all other FIR systems whose magnitude responses are identical to that of $H(z)$.
(b) Which of these systems is a minimum-phase system and which one is a maximum-phase system?
(c) Let $h_k(n)$ denote the impulse response of the k th FIR system determined in part (a) and define the energy delay of the k th system by

$$\mathcal{E}_k(n) \triangleq \sum_{m=n}^{\infty} |h_k(m)|^2 \quad 0 \leq n \leq 3$$

for all values of k . Show that

$$\mathcal{E}_{\min}(n) \leq \mathcal{E}_k(n) \leq \mathcal{E}_{\max}(n) \quad 0 \leq n \leq 3$$

and

$$\mathcal{E}_{\min}(\infty) = \mathcal{E}_k(\infty) = \mathcal{E}_{\max}(\infty) = 0$$

where $\mathcal{E}_{\min}(n)$ and $\mathcal{E}_{\max}(n)$ are energy delays of the minimum-phase and maximum-phase systems, respectively.

2.13 Consider the system function

$$H(z) = \frac{1 + z^{-1} - 6z^{-2}}{1 + \frac{1}{4}z^{-1} - \frac{1}{8}z^{-2}}$$

- (a) Show that the system $H(z)$ is not minimum-phase.
- (b) Construct a minimum-phase system $H_{\min}(z)$ such that $|H_{\min}(e^{j\omega})| = |H(e^{j\omega})|$.
- (c) Is $H(z)$ a maximum-phase system? If yes, explain why. If not, then construct a maximum-phase system $H_{\max}(z)$ such that $|H_{\max}(e^{j\omega})| = |H(e^{j\omega})|$.

2.14 Implement the following system as a parallel connection of two all-pass systems:

$$H(z) = \frac{3 + 9z^{-1} + 9z^{-2} + 3z^{-3}}{12 + 10z^{-1} + 2z^{-2}}$$

2.15 Determine the impulse response of an all-pole system with lattice parameters

$$k_1 = 0.2 \quad k_2 = 0.3 \quad k_3 = 0.5 \quad k_4 = 0.7$$

Draw the direct- and lattice form structures of the above system.

Random Variables, Vectors, and Sequences

So far we have dealt with deterministic signals, that is, signals whose amplitude is uniquely specified by a mathematical formula or rule. However, there are many important examples of signals whose precise description (i.e., as deterministic signals) is extremely difficult, if not impossible. As mentioned in Section 2.1, such signals are called *random signals*. Although random signals are evolving in time in an unpredictable manner, their average properties can be often assumed to be deterministic; that is, they can be specified by explicit mathematical formulas. This is the key for the modeling of a random signal as a stochastic process.

Our aim in the subsequent discussions is to present some basic results from the theory of random variables, random vectors, and discrete-time stochastic processes that will be useful in the chapters that follow. We assume that most readers have some basic knowledge of these topics, and so parts of this chapter may be treated as a review exercise. However, some specific topics are developed in greater depth with a viewpoint that will serve as a foundation for the rest of the book. A more complete treatment can be found in Papoulis (1991), Helstrom (1992), and Stark and Woods (1994).

3.1 RANDOM VARIABLES

The concept of random variables begins with the definition of probability. Consider an experiment with a finite or infinite number of unpredictable outcomes from a *universal set*, denoted by $\mathcal{S} = \{\zeta_1, \zeta_2, \dots\}$. A collection of subsets of \mathcal{S} containing \mathcal{S} itself and that is closed under countable set operations is called a *σ field* and denoted by \mathcal{F} . Elements of \mathcal{F} are called *events*. The unpredictability of these events is measured by a nonnegative set function $\Pr\{\zeta_k\}$, $k = 1, 2, \dots$, called the *probability* of event ζ_k . This set function satisfies three well-known and intuitive axioms (Papoulis 1991) such that the probability of any event produced by set-theoretic operations on the events of \mathcal{S} can be uniquely determined. Thus, any situation of random nature, abstract or otherwise, can be studied using the axiomatic definition of probability by defining an appropriate probability space $(\mathcal{S}, \mathcal{F}, \Pr)$.

In practice it is often difficult, if not impossible, to work with this probability space for two reasons. First, the basic space contains abstract events and outcomes that are difficult to manipulate. In engineering applications, we want random outcomes that can be measured and manipulated in a meaningful way by using numerical operations. Second, the probability function $\Pr\{\cdot\}$ is a set function that again is difficult, if not impossible, to manipulate by using calculus. These two problems are addressed through the concept of the random variable.

DEFINITION 3.1 (RANDOM VARIABLE). A random variable $x(\zeta)$ is a mapping that assigns a real number x to every outcome ζ from an abstract probability space. This mapping should satisfy the following two conditions: (1) the interval $\{x(\zeta) \leq x\}$ is an event in the abstract probability space for every x ; (2) $\Pr\{x(\zeta) = \infty\} = 0$ and $\Pr\{x(\zeta) = -\infty\} = 0$.

A complex-valued random variable is defined by $x(\zeta) = x_R(\zeta) + jx_I(\zeta)$ where $x_R(\zeta)$ and $x_I(\zeta)$ are real-valued random variables. We will discuss complex-valued random variables in Section 3.2. Strictly speaking, a random variable is neither random nor a variable but is a function or a mapping. As shown in Figure 3.1, the domain of a random variable is the universal set \mathcal{S} , and its range is the real line \mathbb{R} . Since random variables are numbers, they can be added, subtracted, or manipulated otherwise.

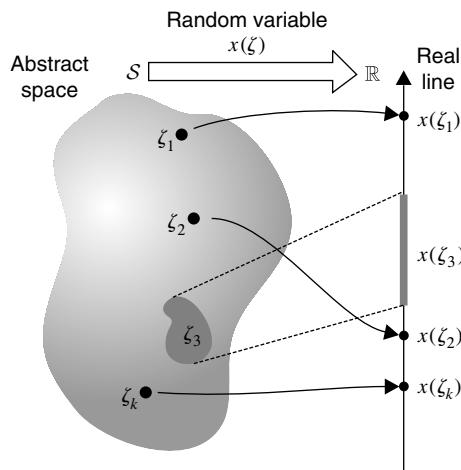


FIGURE 3.1
 Graphical illustration of random variable mapping.

An important comment on notation. We will use $x(\zeta)$, $y(\zeta)$, \dots , to denote random variables and the corresponding lowercase alphabet without parentheses to denote their values; for example, $x(\zeta) = x$ means that the random variable $x(\zeta)$ takes value equal to x . We believe that this notation will not cause any confusion because the meaning of the lowercase variable will be clear from the context.[†] A specific value of the random variable realization will be denoted by $x(\zeta_0) = x_0$ (corresponding to a particular event ζ_0 in the original space).

A random variable is called *discrete-valued* if x takes a discrete set of values $\{x_k\}$; otherwise, it is termed a *continuous-valued* random variable. A *mixed* random variable takes both discrete and continuous values.

3.1.1 Distribution and Density Functions

The probability set function $\Pr\{x(\zeta) \leq x\}$ is a function of the set $\{x(\zeta) \leq x\}$, but it is also a number that varies with x . Hence it is also a function of a point x on the real line \mathbb{R} . This point function is the well-known *cumulative distribution function* (cdf) $F_x(x)$ of a random variable $x(\zeta)$ and is defined by

$$F_x(x) \triangleq \Pr\{x(\zeta) \leq x\} \quad (3.1.1)$$

The second important probability function is the *probability density function* (pdf) $f_x(x)$,

[†]Traditionally, the uppercase alphabet is used to denote random variables. We have reserved the use of uppercase alphabet for transform-domain quantities.

which is defined as a formal derivative

$$f_x(x) \triangleq \frac{dF_x(x)}{dx} \quad (3.1.2)$$

Note that the pdf $f_x(x)$ is not the probability, but must be multiplied by a certain interval Δx to obtain a probability, that is,

$$f_x(x)\Delta x \approx \Delta F_x(x) \triangleq F_x(x + \Delta x) - F_x(x) = \Pr\{x < x(\zeta) \leq x + \Delta x\} \quad (3.1.3)$$

Integrating both sides of (3.1.2), we obtain

$$F_x(x) = \int_{-\infty}^x f_x(v) dv \quad (3.1.4)$$

For discrete-valued random variables, we use the *probability mass function* (pmf) p_k , defined as the probability that random variable $x(\zeta)$ takes a value equal to x_k , or

$$p_k \triangleq \Pr\{x(\zeta) = x_k\} \quad (3.1.5)$$

These probability functions satisfy several important properties (Papoulis 1991), such as

$$0 \leq F_x(x) \leq 1 \quad F_x(-\infty) = 0 \quad F_x(\infty) = 1 \quad (3.1.6)$$

$$f_x(x) \geq 0 \quad \int_{-\infty}^{\infty} f_x(x) dx = 1 \quad (3.1.7)$$

Using these functions and their properties, we can compute the probabilities of any event (or interval) on \mathbb{R} . For example,

$$\Pr\{x_1 < x(\zeta) \leq x_2\} = F_x(x_2) - F_x(x_1) = \int_{x_1}^{x_2} f_x(x) dx \quad (3.1.8)$$

3.1.2 Statistical Averages

To completely characterize a random variable, we have to know its probability density function. In practice, it is desirable to summarize some of the key aspects of a density function by using a few numbers rather than to specify the entire density function. These numbers, which are called *statistical averages* or *moments*, are evaluated by using the mathematical expectation operation. Although density functions are needed to theoretically compute moments, in practice, moments are easily estimated without the explicit knowledge of density functions.

Mathematical expectation

This is one of the most important operations in the theory of random variables. It is generally used to describe various statistical averages, and it is also needed in estimation theory. The *expected* or *mean value* of a random variable $x(\zeta)$ is given by

$$E\{x(\zeta)\} \triangleq \mu_x = \begin{cases} \sum_k x_k p_k & x(\zeta) \text{ discrete} \\ \int_{-\infty}^{\infty} x f_x(x) dx & x(\zeta) \text{ continuous} \end{cases} \quad (3.1.9)$$

Although, strictly speaking, to compute $E\{x(\zeta)\}$ we need the definitions for both the discrete and continuous random variables, we will follow the engineering practice of using the expression for the continuous random variable (which can also describe a discrete random variable if we allow impulse functions in its pdf). The expectation operation computes a statistical average by using the density $f_x(x)$ as a weighting function. Hence, the mean μ_x can be regarded as the “location” (or the “center of gravity”) of the density $f_x(x)$, as shown in Figure 3.2(a). If $f_x(x)$ is symmetric about $x = a$, then $\mu_x = a$ and, in particular, if $f_x(x)$

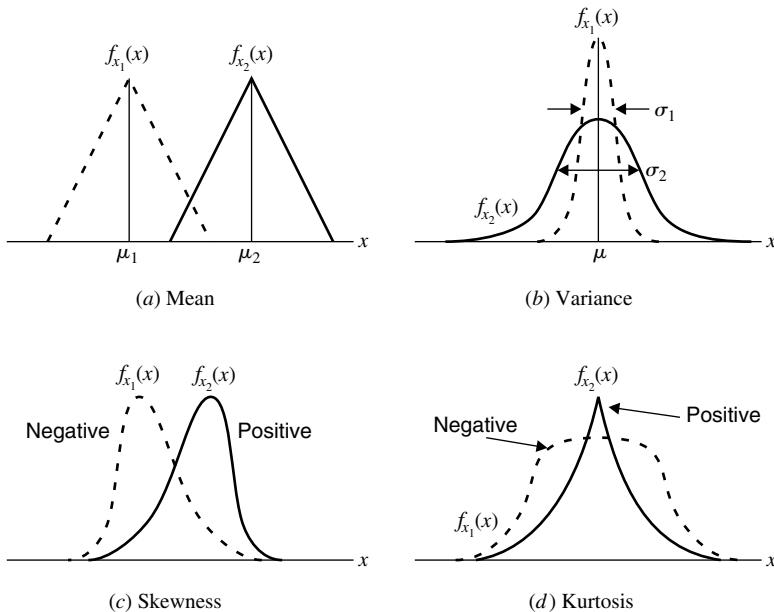


FIGURE 3.2

Illustration of mean, standard deviation, skewness, and kurtosis.

is an even function, then $\mu_x = 0$. One important property of expectation is that it is a linear operation, that is,

$$E\{\alpha x(\zeta) + \beta\} = \alpha \mu_x + \beta \quad (3.1.10)$$

Let $y(\zeta) = g[x(\zeta)]$ be a random variable obtained by transforming $x(\zeta)$ through a suitable function.[†] Then the expectation of $y(\zeta)$ is given by

$$E\{y(\zeta)\} \triangleq E\{g[x(\zeta)]\} = \int_{-\infty}^{\infty} g(x) f_x(x) dx \quad (3.1.11)$$

Moments

Using the expectation operations (3.1.9) and (3.1.11), we can define various moments of the random variable $x(\zeta)$ that describe certain useful aspects of the density function. Let $g[x(\zeta)] = x^m(\zeta)$. Then

$$r_x^{(m)} \triangleq E\{x^m(\zeta)\} = \int_{-\infty}^{\infty} x^m f_x(x) dx \quad (3.1.12)$$

is called the *mth-order moment* of $x(\zeta)$. In particular, $r_x^{(0)} = 1$, and the first-order moment $r_x^{(1)} = \mu_x$. The second-order moment $r_x^{(2)} = E\{x^2(\zeta)\}$ is called the *mean-squared value*, and it plays an important role in estimation theory. Note that

$$E\{x^2(\zeta)\} \neq E^2\{x(\zeta)\} \quad (3.1.13)$$

Corresponding to these moments we also have central moments. Let $g[x(\zeta)] = [x(\zeta) - \mu_x]^m$, then

$$\gamma_x^{(m)} \triangleq E\{[x(\zeta) - \mu_x]^m\} = \int_{-\infty}^{\infty} (x - \mu_x)^m f_x(x) dx \quad (3.1.14)$$

is called the *mth-order central moment* of $x(\zeta)$. In particular, $\gamma_x^{(0)} = 1$ and $\gamma_x^{(1)} = 0$, which is obvious. Clearly, a random variable's moments and central moments are identical if its

[†]Such a function $g(\cdot)$ is called a *Baire function* (Papoulis 1991).

mean value is zero. The second central moment is of considerable importance and is called the *variance* of $x(\zeta)$, denoted by σ_x^2 . Thus

$$\text{var}[x(\zeta)] \triangleq \sigma_x^2 \triangleq \gamma_x^{(2)} = E\{[x(\zeta) - \mu_x]^2\} \quad (3.1.15)$$

The quantity $\sigma_x = \sqrt{\gamma_x^{(2)}}$ is called the *standard deviation* of $x(\zeta)$ and is a measure of the spread (or dispersion) of the observed values of $x(\zeta)$ around its mean μ_x [see Figure 3.2(b)]. The relation between a random variable's moments and central moments is given by (see Problem 3.3)

$$\gamma_x^{(m)} = \sum_{k=0}^m \binom{m}{k} (-1)^k \mu_x^k r_x^{(m-k)} \quad (3.1.16)$$

In particular, and also from (3.1.15), we have

$$\sigma_x^2 = r_x^{(2)} - \mu_x^2 = E\{x^2(\zeta)\} - E^2\{x(\zeta)\} \quad (3.1.17)$$

The quantity *skewness* is related to the third-order central moment and characterizes the degree of asymmetry of a distribution around its mean, as shown in Figure 3.2(c). It is defined as a *normalized* third-order central moment, that is,

$$\text{Skew} \triangleq \tilde{\kappa}_x^{(3)} \triangleq E \left\{ \left[\frac{x(\zeta) - \mu_x}{\sigma_x} \right]^3 \right\} = \frac{1}{\sigma_x^3} \gamma_x^{(3)} \quad (3.1.18)$$

and is a *dimensionless* quantity. It is a pure number that attempts to describe leaning of the shape of the distribution. The skewness is zero if the density function is symmetric about its mean value, is positive if the shape leans towards the right, or is negative if it leans towards the left.

The quantity related to the fourth-order central moment is called *kurtosis*, which is also a dimensionless quantity. It measures the relative flatness or peakedness of a distribution about its mean as shown in Figure 3.2(d). This relative measure is with respect to a normal distribution, which will be introduced in the next section. The kurtosis is defined as

$$\text{Kurtosis} \triangleq \tilde{\kappa}_x^{(4)} \triangleq E \left\{ \left[\frac{x(\zeta) - \mu_x}{\sigma_x} \right]^4 \right\} - 3 = \frac{1}{\sigma_x^4} \gamma_x^{(4)} - 3 \quad (3.1.19)$$

where the term -3 makes the kurtosis $\tilde{\kappa}_x^{(4)} = 0$ for the normal distribution [see (3.1.40) for explanation].

Chebyshev's inequality. A useful result in the interpretation and use of the mean μ and the variance σ^2 of a random variable is given by Chebyshev's inequality. Given a random variable $x(\zeta)$ with its mean μ_x and variance σ_x^2 , we have the inequality

$$\Pr\{|x(\zeta) - \mu_x| \geq k\sigma_x\} \leq \frac{1}{k^2} \quad k > 0 \quad (3.1.20)$$

The interpretation of the above inequality is that regardless of the shape of $f_x(x)$, the random variable $x(\zeta)$ deviates from its mean by k times its standard deviation with probability less than or equal to $1/k^2$.

Characteristic functions

The Fourier and Laplace transforms find many uses in probability theory through the concepts of characteristic and moment generating functions. The *characteristic function* of a random variable $x(\zeta)$ is defined by the integral

$$\Phi_x(\xi) \triangleq E\{e^{j\xi x(\zeta)}\} = \int_{-\infty}^{\infty} f_x(x) e^{j\xi x} dx \quad (3.1.21)$$

which can be interpreted as the Fourier transform of $f_x(x)$ with sign reversal in the complex exponential. To avoid confusion with the cdf, we do not use $F_x(\xi)$ to denote this Fourier transform. Furthermore, the variable ξ in $\Phi_x(\xi)$ is not and should not be interpreted as frequency. When $j\xi$ in (3.1.21) is replaced by a complex variable s , we obtain the *moment generating function* defined by

$$\bar{\Phi}_x(s) \triangleq E\{e^{sx(\xi)}\} = \int_{-\infty}^{\infty} f_x(x)e^{sx} dx \quad (3.1.22)$$

which again can be interpreted as the Laplace transform of $f_x(x)$ with sign reversal. Expanding e^{sx} in (3.1.22) in a Taylor series at $s = 0$, we obtain

$$\begin{aligned} \bar{\Phi}_x(s) &= E\{e^{sx(\xi)}\} = E\left\{1 + sx(\xi) + \frac{[sx(\xi)]^2}{2!} + \cdots + \frac{[sx(\xi)]^m}{m!} + \cdots\right\} \\ &= 1 + s\mu_x + \frac{s^2}{2!}r_x^{(2)} + \cdots + \frac{s^m}{m!}r_x^{(m)} + \cdots \end{aligned} \quad (3.1.23)$$

provided every moment $r_x^{(m)}$ exists. Thus from (3.1.23) we infer that if all moments of $x(\xi)$ are known (and exist), then we can assemble $\bar{\Phi}_x(s)$ and upon inverse Laplace transformation, we can determine the density function $f_x(x)$. If we differentiate $\bar{\Phi}_x(s)$ with respect to s , we obtain

$$r_x^{(m)} = \left. \frac{d^m[\bar{\Phi}_x(s)]}{ds^m} \right|_{s=0} = (-j)^m \left. \frac{d^m[\Phi_x(\xi)]}{d\xi^m} \right|_{\xi=0} \quad m = 1, 2, \dots \quad (3.1.24)$$

which provides the m th-order moment of the random variable $x(\xi)$.

The functions $\Phi_x(\xi)$ and $\bar{\Phi}_x(s)$ possess all the properties associated with the Fourier and Laplace transforms, respectively. Thus, since $f_x(x)$ is always a real-valued function, $\Phi_x(\xi)$ is conjugate symmetric; and if $f_x(x)$ is also an even function, then $\Phi_x(\xi)$ is a real-valued even function. In addition, they possess several properties due to the basic nature of the pdf. Therefore, the characteristic function $\Phi_x(\xi)$ always exists[†] since

$$\int |f_x(x)| dx = \int f_x(x) dx = 1$$

and $\Phi_x(\xi)$ is maximum at the origin, that is,

$$|\Phi_x(\xi)| \leq \Phi_x(0) = 1 \quad (3.1.25)$$

since $f_x(x) \geq 0$.

Cumulants

These statistical descriptors are similar to the moments, but provide better information for higher-order moment analysis, which we will consider in detail in Chapter 12. The cumulants are derived by considering the moment generating function's natural logarithm. This logarithm is commonly referred to as the *cumulant generating function* and is given by

$$\bar{\Psi}_x(s) \triangleq \ln \bar{\Phi}_x(s) = \ln E\{e^{sx(\xi)}\} \quad (3.1.26)$$

When s is replaced by $j\xi$ in (3.1.26), the resulting function is known as the *second characteristic function* and is denoted by $\Psi_x(\xi)$.

The *cumulants* $\kappa_x^{(m)}$ of a random variable $x(\xi)$ are defined as the derivatives of the cumulant generating function, that is,

$$\kappa_x^{(m)} \triangleq \left. \frac{d^m[\bar{\Psi}_x(s)]}{ds^m} \right|_{s=0} = (-j)^m \left. \frac{d^m[\Psi_x(\xi)]}{d\xi^m} \right|_{\xi=0} \quad m = 1, 2, \dots \quad (3.1.27)$$

[†]We will generally choose the characteristic function over the moment generating function.

Clearly, $\kappa_x^{(0)} = 0$. It can be shown that (see Problem 3.4) for a zero-mean random variable, the first five cumulants as functions of the central moments are given by

$$\kappa_x^{(1)} = r_{(x)}^{(1)} = \mu_x = 0 \quad (3.1.28)$$

$$\kappa_x^{(2)} = \gamma_x^{(2)} = \sigma_x^2 \quad (3.1.29)$$

$$\kappa_x^{(3)} = \gamma_x^{(3)} \quad (3.1.30)$$

$$\kappa_x^{(4)} = \gamma_x^{(4)} - 3\sigma_x^4 \quad (3.1.31)$$

$$\kappa_x^{(5)} = \gamma_x^{(5)} - 10\gamma_x^{(3)}\sigma_x^2 \quad (3.1.32)$$

which show that the first two cumulants are identical to the first two central moments. Clearly due to the logarithmic function in (3.1.26), cumulants are useful for dealing with products of characteristic functions (see Section 3.2.4).

3.1.3 Some Useful Random Variables

Random variable models are needed to describe (or approximate) complex physical phenomena using simple parameters. For example, the random phase of a sinusoidal carrier can be described by a uniformly distributed random variable so that we can study its statistical properties. This approximation allows us to investigate random signals in a sound mathematical way. We will describe three continuous random variable models although there are several other known continuous as well as discrete models available in the literature.

Uniformly distributed random variable. This is an appropriate model in situations in which random outcomes are “equally likely.” Here $x(\zeta)$ assumes values on \mathbb{R} according to the pdf

$$f_x(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b \\ 0 & \text{elsewhere} \end{cases} \quad (3.1.33)$$

where $a < b$ are specified parameters. This pdf is shown in Figure 3.3. The corresponding

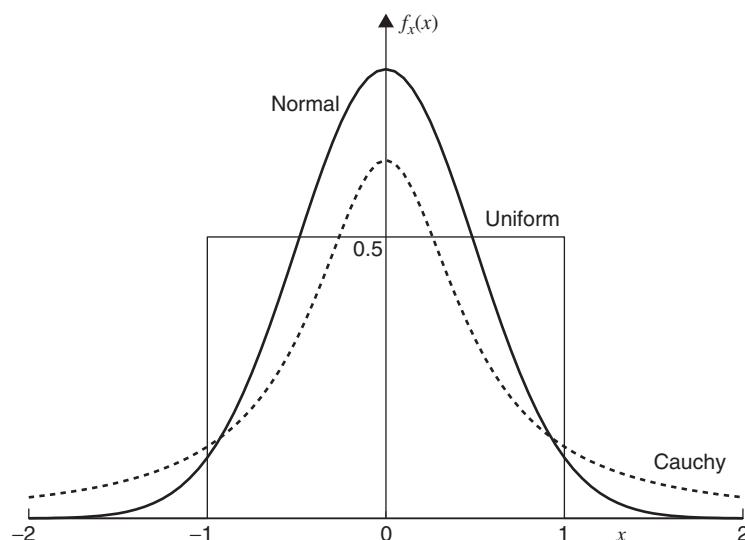


FIGURE 3.3

Probability density functions of useful random variables.

cdf is given by

$$F_x(x) = \int_{-\infty}^x f_x(v) dv = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ 1 & x > b \end{cases} \quad (3.1.34)$$

and the characteristic function is given by

$$\Phi_x(\xi) = \frac{e^{j\xi b} - e^{j\xi a}}{j\xi(b-a)} \quad (3.1.35)$$

The mean and the variance of this random variable are given by, respectively,

$$\mu_x = \frac{a+b}{2} \quad \text{and} \quad \sigma_x^2 = \frac{(b-a)^2}{12} \quad (3.1.36)$$

Normal random variable. This is the most useful and convenient model in many applications, as we shall see later. It is also known as a *Gaussian* random variable, and we will use both terms interchangeably. The pdf of a normally distributed random variable $x(\zeta)$ with mean μ_x and standard deviation σ_x is given by

$$f_x(x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu_x}{\sigma_x}\right)^2\right] \quad (3.1.37)$$

where $-\infty < \mu < \infty$ and $\sigma \geq 0$ (see Figure 3.3). The characteristic function of the normal random variable is given by

$$\Phi_x(\xi) = \exp(j\mu_x\xi - \frac{1}{2}\sigma_x^2\xi^2) \quad (3.1.38)$$

Clearly, the pdf of a normal random variable is completely described by its mean μ_x and standard deviation σ_x and is denoted by $\mathcal{N}(\mu_x, \sigma_x^2)$. We note that *all higher-order moments of a normal random variable can be determined in terms of the first two moments*, that is,

$$\gamma_x^{(m)} = E\{[x(\zeta) - \mu_x]^m\} = \begin{cases} 1 \cdot 3 \cdot 5 \cdots (m-1)\sigma_x^m & \text{if } m \text{ even} \\ 0 & \text{if } m \text{ odd} \end{cases} \quad (3.1.39)$$

In particular, we obtain the fourth moment as

$$\gamma_x^{(4)} = 3\sigma_x^4 \quad (3.1.40)$$

or from (3.1.19), kurtosis = 0, which explains the term -3 in (3.1.19).

From (3.1.37), we observe that the Gaussian random variable is completely determined by its first two moments (mean μ_x and variance σ_x^2), which means that the higher moments do not provide any additional information about the Gaussian density function. In fact, all higher-order moments can be obtained in terms of the first two moments [see Equation (3.1.39)]. Thus for a non-Gaussian random variable, we would like to know how different that random variable is from a Gaussian random variable (this is also known as a departure from the Gaussian-ness). This measurement of the deviation from being Gaussian is given by the cumulants that were defined in (3.1.27). Roughly speaking, the cumulants are like central moments (which measure deviations from the mean) of non-Gaussian random variables for Gaussian departure. Also from (3.1.30) and (3.1.31), we see that all higher-order (that is, $m > 2$) cumulants of a Gaussian random variable are zero. This fact is used in the analysis and estimation of non-Gaussian random variables (and later for non-Gaussian random processes).

Cauchy random variable. This is an appropriate model in which a random variable takes large values with significant probability (heavy-tailed distribution). The Cauchy pdf with parameters μ and β is given by

$$f_x(x) = \frac{\beta}{\pi} \frac{1}{(x-\mu)^2 + \beta^2} \quad (3.1.41)$$

and is shown in Figure 3.3. The corresponding cdf is given by

$$F_x(x) = 0.5 + \frac{1}{\pi} \arctan \frac{x - \mu}{\beta} \quad (3.1.42)$$

and the characteristic function is given by

$$\Phi_x(\xi) = \exp(j\mu\xi - \beta|\xi|) \quad (3.1.43)$$

The Cauchy random variable has mean $\mu_x = \mu$. However, its variance does not exist because $E\{x^2\}$ fails to exist in any sense, and hence the moment generating function does not exist, in general. It has the property that the sum of M independent Cauchy random variables is also Cauchy (see Example 3.2.3). Thus a Cauchy random variable is an example of an infinite-variance random variable.

Random number generators. Random numbers, by definition, are truly unpredictable, and hence it is not possible to generate them by using a well-defined algorithm on a computer. However, in many simulation studies, we need to use sequences of numbers that appear to be random and that possess required properties, for example, Gaussian random numbers in a Monte Carlo analysis. These numbers are called *pseudo random numbers*, and many excellent algorithms are available to generate them on a computer (Park and Miller 1988). In MATLAB, the function `rand` generates numbers that are uniformly distributed over $(0, 1)$ while the function `randn` generates $\mathcal{N}(0, 1)$ pseudo random numbers.

3.2 RANDOM VECTORS

In many applications, a group of signal observations can be modeled as a collection of random variables that can be grouped to form a *random vector*. This is an extension of the concept of random variable and generalizes many scalar quantities to vectors and matrices. One example of a random vector is the case of a complex-valued random variable $x(\zeta) = x_R(\zeta) + jx_I(\zeta)$, which can be considered as a group of $x_R(\zeta)$ and $x_I(\zeta)$. In this section, we provide a review of the basic properties of random vectors and related results from linear algebra. We first begin with real-valued random vectors and then extend their concepts to complex-valued random vectors.

3.2.1 Definitions and Second-Order Moments

A real-valued vector containing M random variables

$$\mathbf{x}(\zeta) = [x_1(\zeta), x_2(\zeta), \dots, x_M(\zeta)]^T \quad (3.2.1)$$

is called a *random M vector* or a random vector when dimensionality is unimportant. As usual, superscript T denotes the transpose of the vector. We can think of a real-valued random vector as a mapping from an abstract probability space to a vector-valued, real space \mathbb{R}^M . Thus the range of this mapping is an M -dimensional space.

Distribution and density functions

A random vector is completely characterized by its *joint* cumulative distribution function, which is defined by

$$F_{\mathbf{x}}(x_1, \dots, x_M) \triangleq \Pr\{x_1(\zeta) \leq x_1, \dots, x_M(\zeta) \leq x_M\} \quad (3.2.2)$$

and is often written as

$$F_{\mathbf{x}}(\mathbf{x}) = \Pr\{\mathbf{x}(\zeta) \leq \mathbf{x}\} \quad (3.2.3)$$

for convenience. A random vector can be also characterized by its *joint* probability density function, which is defined by

$$\begin{aligned} f_{\mathbf{x}}(\mathbf{x}) &= \lim_{\Delta x_1 \rightarrow 0} \frac{\Pr\{x_1 < x_1(\zeta) \leq x_1 + \Delta x_1, \dots, x_M < x_M(\zeta) \leq x_M + \Delta x_M\}}{\Delta x_1 \cdots \Delta x_M} \\ &\quad \vdots \\ &\quad \Delta x_M \rightarrow 0 \\ &\triangleq \frac{\partial}{\partial x_1} \cdots \frac{\partial}{\partial x_M} F_{\mathbf{x}}(\mathbf{x}) \end{aligned} \quad (3.2.4)$$

The function

$$f_{x_j}(x_j) = \int_{(M-1)}^{\mathbf{x}} \cdots \int f_{\mathbf{x}}(\mathbf{x}) dx_1 \cdots dx_{j-1} dx_{j+1} \cdots dx_M \quad (3.2.5)$$

is known as a *marginal* density function and describes individual random variables. Thus the probability functions defined for a random variable in the previous section are more appropriately called marginal functions. The joint pdf $f_{\mathbf{x}}(\mathbf{x})$ must be multiplied by a certain M -dimensional region $\Delta \mathbf{x}$ to obtain a probability. From (3.2.4) we obtain

$$F_{\mathbf{x}}(\mathbf{x}) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_M} f_{\mathbf{x}}(\mathbf{v}) dv_1 \cdots dv_M = \int_{-\infty}^{\mathbf{x}} f_{\mathbf{x}}(\mathbf{v}) d\mathbf{v} \quad (3.2.6)$$

These joint probability functions also satisfy several important properties that are similar to (3.1.6) through (3.1.8) for random variables. In particular, note that both $f_{\mathbf{x}}(\mathbf{x})$ and $F_{\mathbf{x}}(\mathbf{x})$ are positive multidimensional functions.

The joint [and conditional probability (see Papoulis 1991)] functions can also be used to define the concept of independent random variables. Two random variables $x_1(\zeta)$ and $x_2(\zeta)$ are *independent* if the events $\{x_1(\zeta) \leq x_1\}$ and $\{x_2(\zeta) \leq x_2\}$ are jointly independent, that is, if

$$\Pr\{x_1(\zeta) \leq x_1, x_2(\zeta) \leq x_2\} = \Pr\{x_1(\zeta) \leq x_1\} \Pr\{x_2(\zeta) \leq x_2\}$$

which implies that

$$F_{x_1, x_2}(x_1, x_2) = F_{x_1}(x_1) F_{x_2}(x_2) \quad \text{and} \quad f_{x_1, x_2}(x_1, x_2) = f_{x_1}(x_1) f_{x_2}(x_2) \quad (3.2.7)$$

Complex-valued random variables and vectors

As we shall see in later chapters, in applications such as channel equalization, array processing, etc., we encounter complex signal and noise models. To formulate these models, we need to describe complex random variables and vectors, and then extend our standard definitions and results to the complex case. A complex random variable is defined as $x(\zeta) = x_R(\zeta) + j x_I(\zeta)$, where $x_R(\zeta)$ and $x_I(\zeta)$ are real-valued random variables. Thus we can think of $x(\zeta)$ as a mapping from an abstract probability space \mathcal{S} to a complex space \mathbb{C} . Alternatively, $x(\zeta)$ can be thought of as a real-valued random vector $[x_R(\zeta), x_I(\zeta)]^T$ with a joint cdf $F_{x_R, x_I}(x_R, x_I)$ or a joint pdf $f_{x_1, x_2}(x_1, x_2)$ that will allow us to define its statistical averages. The mean of $x(\zeta)$ is defined as

$$E\{x(\zeta)\} = \mu_x = E\{x_R(\zeta) + j x_I(\zeta)\} = \mu_{x_R} + j \mu_{x_I} \quad (3.2.8)$$

and the variance is defined as

$$\sigma_x^2 = E\{|x(\zeta)|^2\} - |\mu_x|^2 \quad (3.2.9)$$

which can be shown to be equal to

$$\sigma_x^2 = E\{|x(\zeta)|^2\} - |E\{x(\zeta)\}|^2 \quad (3.2.10)$$

[†]We will not make any distinction in notation between a real-valued and a complex-valued random variable. The actual type should be evident from the context.

A complex-valued random vector is given by

$$\mathbf{x}(\zeta) = \mathbf{x}_R(\zeta) + j\mathbf{x}_I(\zeta) = \begin{bmatrix} x_{R1}(\zeta) \\ \vdots \\ x_{RM}(\zeta) \end{bmatrix} + j \begin{bmatrix} x_{I1}(\zeta) \\ \vdots \\ x_{IM}(\zeta) \end{bmatrix} \quad (3.2.11)$$

and we can think of a complex-valued random vector as a mapping from an abstract probability space to a vector-valued, complex space \mathbb{C}^M . The cdf for the complex-valued random vector $\mathbf{x}(\zeta)$ is then defined as

$$F_{\mathbf{x}}(\mathbf{x}) \triangleq \Pr\{\mathbf{x}(\zeta) \leq \mathbf{x}\} \triangleq \Pr\{\mathbf{x}_R(\zeta) \leq \mathbf{x}_R, \mathbf{x}_I(\zeta) \leq \mathbf{x}_I\} \quad (3.2.12)$$

while its joint pdf is defined as

$$\begin{aligned} f_{\mathbf{x}}(\mathbf{x}) &= \lim_{\Delta x_{R1} \rightarrow 0} \frac{\Pr\{\mathbf{x}_R < \mathbf{x}_R(\zeta) \leq \mathbf{x}_R + \Delta \mathbf{x}_R, \mathbf{x}_I < \mathbf{x}_I(\zeta) \leq \mathbf{x}_I + \Delta \mathbf{x}_I\}}{\Delta x_{R1} \Delta x_{I1} \cdots \Delta x_{RM} \Delta x_{IM}} \\ &\quad \vdots \\ &\quad \Delta x_{IM} \rightarrow 0 \\ &\triangleq \frac{\partial}{\partial x_{R1}} \frac{\partial}{\partial x_{I1}} \cdots \frac{\partial}{\partial x_{RM}} \frac{\partial}{\partial x_{IM}} F_{\mathbf{x}}(\mathbf{x}) \end{aligned} \quad (3.2.13)$$

From (3.2.13), the cdf is obtained by integrating the pdf over all real and imaginary parts, that is

$$F_{\mathbf{x}}(\mathbf{x}) = \int_{-\infty}^{x_{R1}} \cdots \int_{-\infty}^{x_{IM}} f_{\mathbf{x}}(\mathbf{v}) d\nu_{R1} \cdots d\nu_{IM} = \int_{-\infty}^{\mathbf{x}} f_{\mathbf{x}}(\mathbf{v}) d\mathbf{v} \quad (3.2.14)$$

where the single integral in the last expression is used as a compact notation for multidimensional integrals and should not be confused with a complex-contour integral. These probability functions for a complex-valued random vector possess properties similar to those of the real-valued random vectors. In particular,

$$\int_{-\infty}^{\infty} f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} = 1 \quad (3.2.15)$$

Statistical description

Clearly the above probability functions require an enormous amount of information that is not easy to obtain or is too complex mathematically for practical use. In practical applications, random vectors are described by less complete but more manageable statistical averages.

Mean vector. As we have seen before, the most important statistical operation is the expectation operation. The marginal expectation of a random vector $\mathbf{x}(\zeta)$ is called the *mean vector* and is defined by

$$\boldsymbol{\mu}_{\mathbf{x}} = E\{\mathbf{x}(\zeta)\} = \begin{bmatrix} E\{x_1(\zeta)\} \\ \vdots \\ E\{x_M(\zeta)\} \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_M \end{bmatrix} \quad (3.2.16)$$

where the integral is taken over the entire \mathbb{C}^M space. The components of $\boldsymbol{\mu}$ are the means of the individual random variables.

Correlation and covariance matrices. The second-order moments of a random vector $\mathbf{x}(\zeta)$ are given as matrices and describe the spread of its distribution. The *autocorrelation matrix* is defined by

$$\mathbf{R}_{\mathbf{x}} \triangleq E\{\mathbf{x}(\zeta)\mathbf{x}^H(\zeta)\} = \begin{bmatrix} r_{11} & \cdots & r_{1M} \\ \vdots & \ddots & \vdots \\ r_{M1} & \cdots & r_{MM} \end{bmatrix} \quad (3.2.17)$$

where superscript H denotes the conjugate transpose operation, the diagonal terms

$$r_{ii} \triangleq E\{|x_i(\zeta)|^2\} \quad i = 1, \dots, M \quad (3.2.18)$$

are the second-order moments, denoted earlier as $r_{x_i}^{(2)}$, of random variables $x_i(\zeta)$, and the off-diagonal terms

$$r_{ij} \triangleq E\{x_i(\zeta)x_j^*(\zeta)\} = r_{ji}^* \quad i \neq j \quad (3.2.19)$$

measure the *correlation*, that is, the statistical similarity between the random variables $x_i(\zeta)$ and $x_j(\zeta)$. From (3.2.19) we note that the correlation matrix \mathbf{R}_x is conjugate symmetric or *Hermitian*, that is, $\mathbf{R}_x = \mathbf{R}_x^H$.

The *autocovariance matrix* is defined by

$$\boldsymbol{\Gamma}_x \triangleq E\{[\mathbf{x}(\zeta) - \boldsymbol{\mu}_x][\mathbf{x}(\zeta) - \boldsymbol{\mu}_x]^H\} \triangleq \begin{bmatrix} \gamma_{11} & \cdots & \gamma_{1M} \\ \vdots & \ddots & \vdots \\ \gamma_{M1} & \cdots & \gamma_{MM} \end{bmatrix} \quad (3.2.20)$$

where the diagonal terms

$$\gamma_{ii} = E\{|x_i(\zeta) - \mu_i|^2\} \quad i = 1, \dots, M \quad (3.2.21)$$

are the (self-)variances of $x_i(\zeta)$ denoted earlier as $\sigma_{x_i}^2$ while the off-diagonal terms

$$\gamma_{ij} = E\{[x_i(\zeta) - \mu_i][x_j(\zeta) - \mu_j]^*\} = E\{x_i(\zeta)x_j^*(\zeta)\} - \mu_i\mu_j^* = \gamma_{ji}^* \quad i \neq j \quad (3.2.22)$$

are the values of the *covariance* between $x_i(\zeta)$ and $x_j(\zeta)$. The covariance matrix $\boldsymbol{\Gamma}_x$ is also a Hermitian matrix. The covariance γ_{ij} can also be expressed in terms of standard deviations of $x_i(\zeta)$ and $x_j(\zeta)$ as $\gamma_{ij} = \rho_{ij}\sigma_i\sigma_j$, where

$$\rho_{ij} \triangleq \frac{\gamma_{ij}}{\sigma_i\sigma_j} = \rho_{ji} \quad (3.2.23)$$

is called the *correlation coefficient* between $x_i(\zeta)$ and $x_j(\zeta)$. Note that

$$|\rho_{ij}| \leq 1 \quad i \neq j \quad \rho_{ii} = 1 \quad (3.2.24)$$

The correlation coefficient measures the degree of statistical similarity between two random variables. If $|\rho_{ij}| = 1$, then random variables are said to be *perfectly correlated*; but if $\rho_{ij} = 0$ (that is, when the covariance $\gamma_{ij} = 0$), then $x_i(\zeta)$ and $x_j(\zeta)$ are said to be *uncorrelated*.

The autocorrelation and autocovariance matrices are related. Indeed, we can easily see that

$$\boldsymbol{\Gamma}_x \triangleq E\{[\mathbf{x}(\zeta) - \boldsymbol{\mu}_x][\mathbf{x}(\zeta) - \boldsymbol{\mu}_x]^H\} = \mathbf{R}_x - \boldsymbol{\mu}_x\boldsymbol{\mu}_x^H \quad (3.2.25)$$

which shows that these two moments have essentially the same amount of information. In fact, if $\boldsymbol{\mu}_x = \mathbf{0}$, then $\boldsymbol{\Gamma}_x = \mathbf{R}_x$. The autocovariance measures a *weaker* form of interaction between random variables called *correlatedness* that should be contrasted with the *stronger* form of independence that we described in (3.2.7). If random variables $x_i(\zeta)$ and $x_j(\zeta)$ are independent, then they are also uncorrelated since (3.2.7) implies that

$$E\{x_i(\zeta)x_j^*(\zeta)\} = E\{x_i(\zeta)\}E\{x_j^*(\zeta)\} \quad \text{or} \quad \gamma_{ij} = 0 \quad (3.2.26)$$

but uncorrelatedness does not imply independence unless random variables are jointly Gaussian (see Problem 3.15). The autocorrelation also measures another weaker form of interaction called *orthogonality*. Random variables $x_i(\zeta)$ and $x_j(\zeta)$ are *orthogonal* if their correlation

$$r_{ij} = E\{x_i(\zeta)x_j^*(\zeta)\} = 0 \quad i \neq j \quad (3.2.27)$$

Clearly, from (3.2.26) if one or both random variables have zero means, then uncorrelatedness also implies orthogonality.

We can also define correlation and covariance functions between two random vectors. Let $\mathbf{x}(\zeta)$ and $\mathbf{y}(\zeta)$ be random M - and L -vectors, respectively. Then the $M \times L$ matrix

$$\mathbf{R}_{\mathbf{xy}} \triangleq E\{\mathbf{xy}^H\} = \begin{bmatrix} E\{x_1(\zeta)y_1^*(\zeta)\} & \cdots & E\{x_1(\zeta)y_L^*(\zeta)\} \\ \vdots & \ddots & \vdots \\ E\{x_M(\zeta)y_1^*(\zeta)\} & \cdots & E\{x_M(\zeta)y_L^*(\zeta)\} \end{bmatrix} \quad (3.2.28)$$

is called a *cross-correlation* matrix whose elements r_{ij} are the correlations between random variables $x_i(\zeta)$ and $y_j(\zeta)$. Similarly the $M \times L$ matrix

$$\boldsymbol{\Gamma}_{\mathbf{xy}} \triangleq E\{[\mathbf{x}(\zeta) - \boldsymbol{\mu}_x][\mathbf{y}(\zeta) - \boldsymbol{\mu}_y]^H\} = \mathbf{R}_{\mathbf{xy}} - \boldsymbol{\mu}_x \boldsymbol{\mu}_y^H \quad (3.2.29)$$

is called a *cross-covariance* matrix whose elements c_{ij} are the covariances between $x_i(\zeta)$ and $y_j(\zeta)$. In general the cross-matrices are not square matrices, and even if $M = L$, they are not necessarily symmetric. Two random vectors $\mathbf{x}(\zeta)$ and $\mathbf{y}(\zeta)$ are said to be

- Uncorrelated if

$$\boldsymbol{\Gamma}_{\mathbf{xy}} = \mathbf{0} \Rightarrow \mathbf{R}_{\mathbf{xy}} = \boldsymbol{\mu}_x \boldsymbol{\mu}_y^H \quad (3.2.30)$$

- Orthogonal if

$$\mathbf{R}_{\mathbf{xy}} = \mathbf{0} \quad (3.2.31)$$

Again, if $\boldsymbol{\mu}_x$ or $\boldsymbol{\mu}_y$ or both are zero vectors, then (3.2.30) implies (3.2.31).

3.2.2 Linear Transformations of Random Vectors

Many signal processing applications involve linear operations on random vectors. Linear transformations are relatively simple mappings and are given by the matrix operation

$$\mathbf{y}(\zeta) = g[\mathbf{x}(\zeta)] = \mathbf{Ax}(\zeta) \quad (3.2.32)$$

where \mathbf{A} is an $L \times M$ (not necessarily square) matrix. The random vector $\mathbf{y}(\zeta)$ is completely described by the density function $f_y(\mathbf{y})$. If $L > M$, then only M $y_i(\zeta)$ random variables can be independently determined from $\mathbf{x}(\zeta)$. The remaining $(L - M)$ $y_i(\zeta)$ random variables can be obtained from the first $y_i(\zeta)$ random variables. Thus we need to determine $f_y(\mathbf{y})$ for M random variables from which we can determine $f_y(\mathbf{y})$ for all L random variables. If $M > L$, then we can augment \mathbf{y} into an M -vector by introducing auxiliary random variables

$$y_{L+1}(\zeta) = x_{L+1}(\zeta), \dots, y_M(\zeta) = x_M(\zeta) \quad (3.2.33)$$

to determine $f_y(\mathbf{y})$ for M random variables from which we can determine $f_y(\mathbf{y})$ for the original L random variables. Therefore, for the determination of the pdf $f_y(\mathbf{y})$, we will assume that $L = M$ and that \mathbf{A} is nonsingular.

Furthermore, we will first consider the case in which both $\mathbf{x}(\zeta)$ and $\mathbf{y}(\zeta)$ are real-valued random vectors, which also implies that \mathbf{A} is a real-valued matrix. This approach is necessary because the complex case leads to a slightly different result. Then the pdf $f_y(\mathbf{y})$ is given by

$$f_y(\mathbf{y}) = \frac{f_x(g^{-1}(\mathbf{y}))}{|\mathbf{J}|} \quad (3.2.34)$$

where \mathbf{J} is called the *Jacobian* of the transformation (3.2.32), given by

$$\mathbf{J} = \det \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \dots & \frac{\partial y_M}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_M} & \dots & \frac{\partial y_M}{\partial x_M} \end{bmatrix} = \det \mathbf{A} \quad (3.2.35)$$

From (3.2.34) and (3.2.35), the pdf of $\mathbf{y}(\zeta)$ is given by

$$f_{\mathbf{y}}(\mathbf{y}) = \frac{f_{\mathbf{x}}(\mathbf{A}^{-1}\mathbf{y})}{|\det \mathbf{A}|} \quad \text{real-valued random vector} \quad (3.2.36)$$

from which moment computations of any order of $\mathbf{y}(\zeta)$ can be performed. Now we consider the case of the complex-valued random vectors. Then by applying the above approach to both real and imaginary parts, the result (3.2.36) becomes

$$f_{\mathbf{y}}(\mathbf{y}) = \frac{f_{\mathbf{x}}(\mathbf{A}^{-1}\mathbf{y})}{|\det \mathbf{A}|^2} \quad \text{complex-valued random vector} \quad (3.2.37)$$

This shows that sometimes we can get different results depending upon whether we assume real- or complex-valued random vectors in our analysis.

Determining $f_{\mathbf{y}}(\mathbf{y})$ is, in general, tedious except in the case of Gaussian random vectors, as we shall see later. In practice, the knowledge of $\boldsymbol{\mu}_{\mathbf{y}}$, $\boldsymbol{\Gamma}_{\mathbf{y}}$, $\boldsymbol{\Gamma}_{\mathbf{xy}}$, or $\boldsymbol{\Gamma}_{\mathbf{yx}}$ is sufficient in many applications. If we take the expectation of both sides of (3.2.32), we find that the mean vector is given by

$$\boldsymbol{\mu}_{\mathbf{y}} = E\{\mathbf{y}(\zeta)\} = E\{\mathbf{Ax}(\zeta)\} = \mathbf{A}E\{\mathbf{x}(\zeta)\} = \mathbf{A}\boldsymbol{\mu}_{\mathbf{x}} \quad (3.2.38)$$

The autocorrelation matrix of $\mathbf{y}(\zeta)$ is given by

$$\mathbf{R}_{\mathbf{y}} = E\{\mathbf{yy}^H\} = E\{\mathbf{Axx}^H\mathbf{A}^H\} = \mathbf{A}E\{\mathbf{xx}^H\}\mathbf{A}^H = \mathbf{A}\mathbf{R}_{\mathbf{x}}\mathbf{A}^H \quad (3.2.39)$$

Similarly, the autocovariance matrix of $\mathbf{y}(\zeta)$ is given by

$$\boldsymbol{\Gamma}_{\mathbf{y}} = \mathbf{A}\boldsymbol{\Gamma}_{\mathbf{x}}\mathbf{A}^H \quad (3.2.40)$$

Consider the cross-correlation matrix

$$\mathbf{R}_{\mathbf{xy}} = E\{\mathbf{x}(\zeta)\mathbf{y}^H(\zeta)\} = E\{\mathbf{x}(\zeta)\mathbf{x}^H(\zeta)\mathbf{A}^H\} \quad (3.2.41)$$

$$= E\{\mathbf{x}(\zeta)\mathbf{x}^H(\zeta)\}\mathbf{A}^H = \mathbf{R}_{\mathbf{x}}\mathbf{A}^H \quad (3.2.42)$$

and hence $\mathbf{R}_{\mathbf{yx}} = \mathbf{A}\mathbf{R}_{\mathbf{x}}$. Similarly, the cross-covariance matrices are

$$\boldsymbol{\Gamma}_{\mathbf{xy}} = \boldsymbol{\Gamma}_{\mathbf{x}}\mathbf{A}^H \quad \text{and} \quad \boldsymbol{\Gamma}_{\mathbf{yx}} = \mathbf{A}\boldsymbol{\Gamma}_{\mathbf{x}} \quad (3.2.43)$$

3.2.3 Normal Random Vectors

If the components of the random vector $\mathbf{x}(\zeta)$ are jointly normal, then $\mathbf{x}(\zeta)$ is a normal random M -vector. Again, the pdf expressions for the real- and complex-valued cases are slightly different, and hence we consider these cases separately. The real-valued normal random vector has the pdf

$$f_{\mathbf{x}}(\mathbf{x}) = \frac{1}{(2\pi)^{M/2}|\boldsymbol{\Gamma}_{\mathbf{x}}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})^T \boldsymbol{\Gamma}_{\mathbf{x}}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})\right] \quad \text{real} \quad (3.2.44)$$

with mean $\boldsymbol{\mu}_{\mathbf{x}}$ and covariance $\boldsymbol{\Gamma}_{\mathbf{x}}$. It will be denoted by $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Gamma}_{\mathbf{x}})$. The term in the exponent $(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})^T \boldsymbol{\Gamma}_{\mathbf{x}}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})$ is a positive definite quadratic function of x_i and is also given by

$$(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})^T \boldsymbol{\Gamma}_{\mathbf{x}}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}}) = \sum_{i=1}^M \sum_{j=1}^M \langle \boldsymbol{\Gamma}_{\mathbf{x}}^{-1} \rangle_{ij} (x_i - \mu_i)(x_j - \mu_j) \quad (3.2.45)$$

where $\langle \boldsymbol{\Gamma}_{\mathbf{x}}^{-1} \rangle_{ij}$ denotes the (i, j) th element of $\boldsymbol{\Gamma}_{\mathbf{x}}^{-1}$. The characteristic function of the normal random vector is given by

$$\Phi_{\mathbf{x}}(\boldsymbol{\xi}) = \exp(j\boldsymbol{\xi}^T \boldsymbol{\mu}_{\mathbf{x}} - \frac{1}{2}\boldsymbol{\xi}^T \boldsymbol{\Gamma}_{\mathbf{x}} \boldsymbol{\xi}) \quad (3.2.46)$$

where $\boldsymbol{\xi}^T = [\xi_1, \dots, \xi_M]$.

The complex-valued normal random vector has the pdf

$$f_{\mathbf{x}}(\mathbf{x}) = \frac{1}{\pi^M |\boldsymbol{\Gamma}_{\mathbf{x}}|} \exp[-(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})^H \boldsymbol{\Gamma}_{\mathbf{x}}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})] \quad \text{complex} \quad (3.2.47)$$

with mean $\boldsymbol{\mu}_{\mathbf{x}}$ and covariance $\boldsymbol{\Gamma}_{\mathbf{x}}$. This pdf will be denoted by $\mathcal{CN}(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Gamma}_{\mathbf{x}})$. If $\mathbf{x}(\zeta)$ is a scalar complex-valued random variable $x(\zeta)$ with mean μ_x and variance σ_x^2 , then (3.2.47) reduces to

$$f_x(x) = \frac{1}{\pi \sigma_x^2} \exp\left(-\frac{|x - \mu|^2}{\sigma_x^2}\right) \quad (3.2.48)$$

which should be compared with the pdf given in (3.1.37). Note that the pdf in (3.1.37) is not obtained by setting the imaginary part of $x(\zeta)$ in (3.2.48) equal to zero. For a more detailed discussion on this aspect, see Therrien (1992) or Kay (1993). The term $(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})^H \boldsymbol{\Gamma}_{\mathbf{x}}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})$ in the exponent of (3.2.47) is also a positive definite quadratic function and is given by

$$(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})^H \boldsymbol{\Gamma}_{\mathbf{x}}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}}) = \sum_{i=1}^M \sum_{j=1}^M \langle \boldsymbol{\Gamma}_{\mathbf{x}}^{-1} \rangle_{ij} (x_i - \mu_i)^* (x_j - \mu_j) \quad (3.2.49)$$

The characteristic function for the complex-valued normal random vector is given by

$$\Phi_{\mathbf{x}}(\xi) = \exp[j \operatorname{Re}(\xi^H \boldsymbol{\mu}_{\mathbf{x}}) - \frac{1}{4} \xi^H \boldsymbol{\Gamma}_{\mathbf{x}} \xi] \quad (3.2.50)$$

The normal distribution is a useful model of a random vector because of its many important properties:

1. The pdf is completely specified by the mean vector and the covariance matrix, which are relatively easy to estimate in practice. All other higher-order moments can be obtained from these parameters.
2. If the components of $\mathbf{x}(\zeta)$ are mutually uncorrelated, then they are also independent. (See Problem 3.15.) This is useful in many derivations.
3. A linear transformation of a normal random vector is also normal. This can be easily seen by using (3.2.38), (3.2.40), and (3.2.44) in (3.2.36); that is, for the real-valued case we obtain

$$f_{\mathbf{y}}(\mathbf{y}) = \frac{1}{(2\pi)^{M/2} |\boldsymbol{\Gamma}_{\mathbf{y}}|^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}})^T \boldsymbol{\Gamma}_{\mathbf{y}}^{-1} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}})\right] \quad \text{real} \quad (3.2.51)$$

This result can also be proved by using the moment generating function in (3.2.46) (see Problem 3.6). Similarly for the complex-valued case, from (3.2.37) and (3.2.47) we obtain

$$f_{\mathbf{y}}(\mathbf{y}) = \frac{1}{\pi^M |\boldsymbol{\Gamma}_{\mathbf{y}}|} \exp[-(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}})^H (\mathbf{A}^{-1})^H \boldsymbol{\Gamma}_{\mathbf{x}}^{-1} \mathbf{A}^{-1} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}})] \quad \text{complex} \quad (3.2.52)$$

4. The fourth-order moment of a normal random vector

$$\mathbf{x}(\zeta) = [x_1(\zeta) \ x_2(\zeta) \ x_3(\zeta) \ x_4(\zeta)]^T$$

can be expressed in terms of its second-order moments. For the real case, that is, when $\mathbf{x}(\zeta) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Gamma}_{\mathbf{x}})$, we have

$$\begin{aligned} E\{x_1(\zeta)x_2(\zeta)x_3(\zeta)x_4(\zeta)\} &= E\{x_1(\zeta)x_2(\zeta)\}E\{x_3(\zeta)x_4(\zeta)\} \\ &\quad + E\{x_1(\zeta)x_3(\zeta)\}E\{x_2(\zeta)x_4(\zeta)\} \\ &\quad + E\{x_1(\zeta)x_4(\zeta)\}E\{x_2(\zeta)x_3(\zeta)\} \end{aligned} \quad (3.2.53)$$

For the complex case, that is, when $\mathbf{x}(\zeta) \sim \mathcal{CN}(\mathbf{0}, \mathbf{\Gamma}_x)$, we have

$$\begin{aligned} E\{x_1^*(\zeta)x_2(\zeta)x_3^*(\zeta)x_4(\zeta)\} &= E\{x_1^*(\zeta)x_2(\zeta)\}E\{x_3^*(\zeta)x_4(\zeta)\} \\ &\quad + E\{x_1^*(\zeta)x_4(\zeta)\}E\{x_2(\zeta)x_3^*(\zeta)\} \end{aligned} \quad (3.2.54)$$

The proof of (3.2.53) is tedious but straightforward. However, the proof of (3.2.54) is complicated and is discussed in Kay (1993).

3.2.4 Sums of Independent Random Variables

In many applications, a random variable $y(\zeta)$ can be expressed as a linear combination of M statistically independent random variables $\{x_k(\zeta)\}_1^M$, that is,

$$y(\zeta) = c_1x_1(\zeta) + c_2x_2(\zeta) + \cdots + c_Mx_M(\zeta) = \sum_{k=1}^M c_kx_k(\zeta) \quad (3.2.55)$$

where $\{c_k\}_1^M$ is a set of fixed coefficients. In these situations, we would like to compute the first two moments and the pdf of $y(\zeta)$. The moment computation is straightforward, but the pdf computation requires the use of characteristic functions. When these results are extended to the sum of an infinite number of statistically independent random variables, we obtain a powerful theorem called the *central limit theorem (CLT)*. Another interesting concept develops when the sum of IID random variables preserves their distribution, which results in stable distributions.

Mean. Using the linearity of the expectation operator and taking the expectation of both sides of (3.2.55), we obtain

$$\mu_y = \sum_{k=1}^M c_k\mu_{x_k} \quad (3.2.56)$$

Variance. Again by using independence, the variance of $y(\zeta)$ is given by

$$\sigma_y^2 = E \left\{ \left| \sum_{k=1}^M c_k[x_k(\zeta) - \mu_{x_k}] \right|^2 \right\} = \sum_{k=1}^M |c_k|^2 \sigma_{x_k}^2 \quad (3.2.57)$$

where we have used the statistical independence between random variables.

Probability density function. Before we derive the pdf of $y(\zeta)$ in (3.2.55), we consider two special cases. First, let

$$y(\zeta) = x_1(\zeta) + x_2(\zeta) \quad (3.2.58)$$

where $x_1(\zeta)$ and $x_2(\zeta)$ are statistically independent. Then its characteristic function is given by

$$\Phi_y(\xi) = E\{e^{j\xi y(\zeta)}\} = E\{e^{j\xi[x_1(\zeta) + x_2(\zeta)]}\} = E\{e^{j\xi x_1(\zeta)}\}E\{e^{j\xi x_2(\zeta)}\} \quad (3.2.59)$$

where the last equality follows from the independence. Hence

$$\Phi_y(\xi) = \Phi_{x_1}(\xi)\Phi_{x_2}(\xi) \quad (3.2.60)$$

or from the convolution property of the Fourier transform

$$f_y(y) = f_{x_1}(y) * f_{x_2}(y) \quad (3.2.61)$$

From (3.2.60) the second characteristic function of $y(\zeta)$ is given by

$$\Psi_y(\xi) = \Psi_{x_1}(\xi) + \Psi_{x_2}(\xi) \quad (3.2.62)$$

or the m th-order cumulant of $y(\xi)$ is given by

$$\kappa_y^{(m)} = \kappa_{x_1}^{(m)} + \kappa_{x_2}^{(m)} \quad (3.2.63)$$

These results can be easily generalized to the sum of M independent random variables.

EXAMPLE 3.2.1. Let $\{x_k(\xi)\}_{k=1}^4$ be four IID random variables uniformly distributed over $[-0.5, 0.5]$. Compute and plot the pdfs of $y_M(\xi) \triangleq \sum_{k=1}^M x_k$ for $M = 2, 3$, and 4 . Compare these pdfs with that of a zero-mean Gaussian random variable.

Solution. Let $f(x)$ be the pdf of a uniform random variable over $[-0.5, 0.5]$, that is,

$$f(x) = \begin{cases} 1 & -0.5 \leq x \leq 0.5 \\ 0 & \text{otherwise} \end{cases} \quad (3.2.64)$$

Then from (3.2.61)

$$f_{y_2}(y) = f(y) * f(y) = \begin{cases} 1+y & -1 \leq y \leq 0 \\ 1-y & 0 \leq y \leq 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.2.65)$$

Similarly, we have

$$f_{y_3}(y) = f_{y_2}(y) * f(y) = \begin{cases} \frac{1}{2}(y + \frac{3}{2})^2 & -\frac{3}{2} \leq y \leq -\frac{1}{2} \\ \frac{3}{4} - y^2 & -\frac{1}{2} \leq y \leq \frac{1}{2} \\ \frac{1}{2}(y - \frac{3}{2})^2 & \frac{1}{2} \leq y \leq \frac{3}{2} \\ 0 & \text{otherwise} \end{cases} \quad (3.2.66)$$

$$\text{and } f_{y_4}(y) = f_{y_3}(y) * f(y) = \begin{cases} \frac{1}{6}(y + 2)^3 & -2 \leq y \leq -1 \\ -\frac{1}{2}y^3 - y^2 + \frac{2}{3} & -1 \leq y \leq 0 \\ \frac{2}{3} + \frac{1}{2}y^3 - y^2 & 0 \leq y \leq 1 \\ -\frac{1}{6}(-2 + y)^3 & 1 \leq y \leq 2 \\ 0 & \text{otherwise} \end{cases} \quad (3.2.67)$$

The plots of $f_{y_2}(y)$, $f_{y_3}(y)$, and $f_{y_4}(y)$ are shown in Figure 3.4 along with the zero-mean Gaussian pdf. The variance of the Gaussian random variable is chosen so that 99.92 percent of the pdf area is over $[-2, 2]$. We observe that as M increases, the pdf plots appear to get closer to the shape of the Gaussian pdf. This observation will be explored in detail in the CLT.

Next, let $y(\xi) = ax(\xi) + b$; then the characteristic function of $y(\xi)$ is

$$\Phi_y(\xi) = E\{e^{j[a\xi x(\xi) + b]\xi}\} = E\{e^{ja\xi x(\xi)} e^{jb\xi}\} = \Phi_x(a\xi) e^{jb\xi} \quad (3.2.68)$$

and by using the properties of the Fourier transform, the pdf of $y(\xi)$ is given by

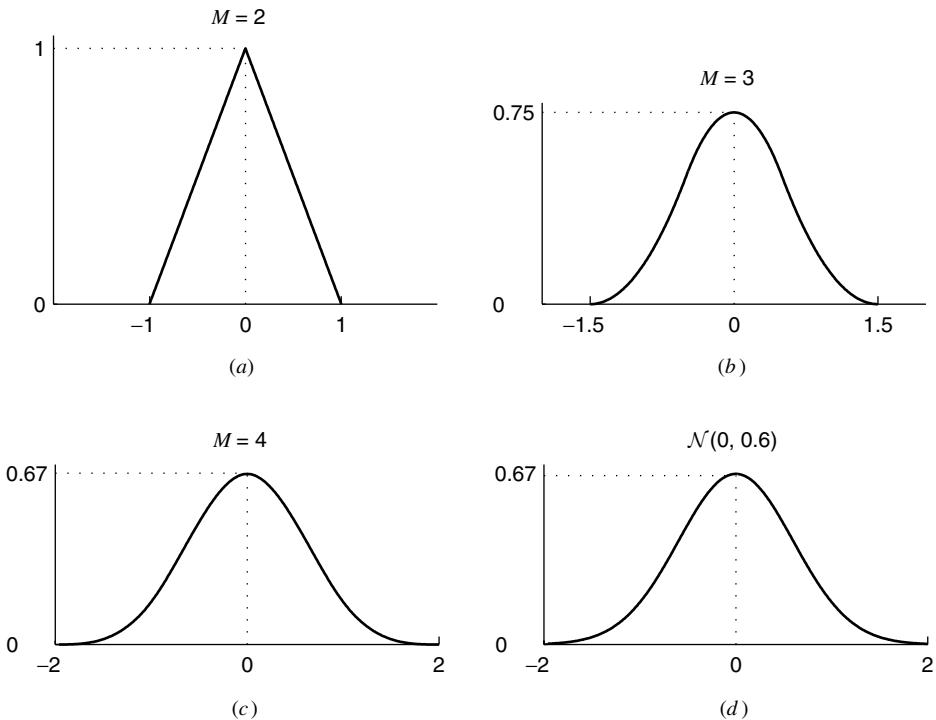
$$f_y(y) = \frac{1}{|a|} f_x\left(\frac{y - b}{a}\right) \quad (3.2.69)$$

From (3.2.68), the second characteristic function is given by

$$\Psi_y(\xi) = \Psi_x(a\xi) + jb\xi \quad (3.2.70)$$

and the cumulants are given by

$$\begin{aligned} \kappa_y^{(m)} &= (-j)^m \left. \frac{d^m \Psi_y(\xi)}{d\xi^m} \right|_{\xi=0} = a^m (-j)^m \left. \frac{d^m \Psi_x(a\xi)}{d\xi^m} \right|_{\xi=0} \\ &= a^m \kappa_x^{(m)} \quad m > 1 \end{aligned} \quad (3.2.71)$$

**FIGURE 3.4**

The pdf plots of (a) sum of two, (b) sum of three, (c) sum of four, and (d) Gaussian random variables in Example 3.2.1.

Finally, consider $y(\xi)$ in (3.2.55). Using the results in (3.2.60) and (3.2.68), we have

$$\Phi_y(\xi) = \prod_{k=1}^M \Phi_{x_k}(c_k \xi) \quad (3.2.72)$$

from which the pdf of $y(\xi)$ is given by

$$f_y(y) = \frac{1}{|c_1|} f_{x_1}\left(\frac{y}{c_1}\right) * \frac{1}{|c_2|} f_{x_2}\left(\frac{y}{c_2}\right) * \cdots * \frac{1}{|c_M|} f_{x_M}\left(\frac{y}{c_M}\right) \quad (3.2.73)$$

From (3.2.62) and (3.2.70), the second characteristic function is given by

$$\Psi_y(\xi) = \sum_{k=1}^M \Psi_{x_k}(c_k \xi) \quad (3.2.74)$$

and hence from (3.2.63) and (3.2.71), the cumulants of $y(\xi)$ are

$$\kappa_y^{(m)} = \sum_{k=1}^M c_k^m \kappa_{x_k}^{(m)} \quad (3.2.75)$$

where c_k^m is the m th power of c_k .

In the following two examples, we consider two interesting cases in which the sum of IID random variables retains their original distribution. The first case concerns Gaussian random variables that have finite variances while the second case involves Cauchy random variables that possess infinite variance.

EXAMPLE 3.2.2. Let $x_k(\xi) \sim \mathcal{N}(\mu_k, \sigma_k^2)$, $k = 1, \dots, M$ and let $y(k) = \sum_1^M x_k(\xi)$. The characteristic function of $x_k(\xi)$ is

$$\Phi_{x_k}(\xi) = \exp\left(j\mu_k\xi - \frac{\xi^2\sigma_k^2}{2}\right)$$

and hence from (3.2.72), we have

$$\Phi_y(\xi) = \exp\left(j\xi \sum_{k=1}^M \mu_k - \frac{\xi^2 \sum_{k=1}^M \sigma_k^2}{2}\right)$$

which means that $y(\xi)$ is also a Gaussian random variable with mean $\sum_{k=1}^M \mu_k$ and variance $\sum_{k=1}^M \sigma_k^2$, that is, $y(\xi) \sim \mathcal{N}(\sum_{k=1}^M \mu_k, \sum_{k=1}^M \sigma_k^2)$. In particular, if the $x_k(\xi)$ are IID with a pdf $\mathcal{N}(\mu, \sigma^2)$, then

$$\Phi_y(\xi) = \exp\left(jM\mu\xi - \frac{\xi^2 M\sigma^2}{2}\right) = \exp\left[M\left(j\xi\mu - \frac{\xi^2\sigma^2}{2}\right)\right] \quad (3.2.76)$$

This behavior of $y(\xi)$ is in contrast with that of the sum of the IID random variables in Example 3.2.1 in which the uniform pdf changed its form after M -fold convolutions.

EXAMPLE 3.2.3. As a second case, consider M IID random variables $\{x_k(\xi)\}_{k=1}^M$ with Cauchy distribution

$$f_{x_k}(x) = \frac{\beta}{\pi} \frac{1}{(x - \alpha)^2 + \beta^2}$$

and let $y(k) = \sum_1^M x_k(\xi)$. Then from (3.1.43), we have

$$\Phi_x(\xi) = \exp(j\alpha\xi - \beta|\xi|)$$

and hence

$$\Phi_y(\xi) = \exp(jM\alpha\xi - M\beta|\xi|) = \exp[M(j\alpha\xi - \beta|\xi|)] \quad (3.2.77)$$

This once again shows that the sum random variable has the same distribution (up to a scale factor) as that of the individual random variables, which in this case is the Cauchy distribution.

From these examples, we note that the Gaussian and the Cauchy random variables are *invariant*, or that they have a “self-reproducing” property under linear transformations. These two examples also raise some interesting questions. Are there any other random variables that possess this *invariance* property? If such random variables exist, what is the form of their pdfs or, alternatively, of their characteristic functions, and what can we say about their means and variances? From (3.2.76) and (3.2.77), observe that if the characteristic function has a general form

$$\Phi_{x_k}(\xi) = a^{\theta(\xi)} \quad (3.2.78)$$

where a is some constant and $\theta(\xi)$ is some function of ξ , then we have

$$\Phi_y(s) = a^{M\theta(s)} \quad (3.2.79)$$

that is, the characteristic function of the sum has the same *functional form* except for a change in scale. Are Gaussian and Cauchy both special cases of some general situation? These questions are answered by the concept of *stable* (more appropriately, linearly invariant or self-reproducing) *distributions*.

Stable distributions. These distributions satisfy the “stability” property, which in simple terms means that the distributions are preserved (or that they self-reproduce) under convolution. The only stable distribution that has finite variance is the Gaussian distribution, which has been well understood and is used extensively in the literature and in

practice. The remaining stable distributions have infinite variances (and in some cases, infinite means) which means that the corresponding random variables exhibit large fluctuations. These distributions can then be used to model signals with large variability and hence are finding increasing use in many diverse applications such as the gravitational fields of stars, temperature distributions in a nuclear reaction, or stock market fluctuations (Lamperti 1996; Samorodnitsky and Taqqu 1994; Feller 1966).

Before we formally define stable distributions, we introduce the following notation for convenience

$$y(\zeta) \stackrel{d}{=} x(\zeta) \quad (3.2.80)$$

to indicate that the random variables $x(\zeta)$ and $y(\zeta)$ have the same distribution. For example, if $y(\zeta) = ax(\zeta) + b$, we have

$$F_y(y) = F_x\left(\frac{y-b}{a}\right) \quad (3.2.81)$$

and therefore $x(\zeta) \stackrel{d}{=} ax(\zeta) + b$.

DEFINITION 3.2. Let $x_1(\zeta), x_2(\zeta), \dots, x_M(\zeta)$ be IID random variables with a common distribution $F_x(x)$ and let $s_M(\zeta) = x_1(\zeta) + \dots + x_M(\zeta)$ be their sum. The distribution $F_x(x)$ is said to be *stable* if for each M there exist constants $a_M > 0$ and b_M such that

$$s_M(\zeta) \stackrel{d}{=} a_M x(\zeta) + b_M \quad (3.2.82)$$

and that $F_x(x)$ is not concentrated at one point.

If (3.2.82) holds for $b_M = 0$, we say that $F_x(x)$ is stable in the *strict sense*. The condition that $F_x(x)$ is not concentrated at one point is necessary because such a distribution is always stable. Thus it is a degenerate case that is of no practical interest. A stable distribution is called *symmetric stable* if the distribution is symmetric, which also implies that it is strictly stable.

It can be shown that for any stable random variable $x(\zeta)$ there is a number α , $0 < \alpha \leq 2$, such that the constant a_M in (3.2.82) is $a_M = M^{1/\alpha}$. The number α is known as the *index of stability* or *characteristic exponent*. A stable random variable $x(\zeta)$ with index α is called α *stable*.

Since there is no closed-form expression for the probability density function of stable random variables, except in special cases, they are specified by their characteristic function $\Phi(\xi)$. This characteristic function is given by

$$\Phi(\xi) = \begin{cases} \exp\{j\mu\xi - |\sigma\xi|^\alpha \cdot [1 - j\beta \operatorname{sign}(\xi) \tan\left(\frac{\pi\alpha}{2}\right)]\} & \alpha \neq 1 \\ \exp\{j\mu\xi - |\sigma\xi|^\alpha \cdot [1 - j\beta \left(\frac{2}{\pi}\right) \operatorname{sign}(\xi) \ln|\xi|]\} & \alpha = 1 \end{cases} \quad (3.2.83)$$

where $\operatorname{sign}(\xi) = \xi/|\xi|$ if $\xi \neq 0$ and zero otherwise. We shall use the notation $S_\alpha(\sigma, \beta, \mu)$ to denote the stable random variable defined by (3.2.83). The parameters in (3.2.83) have the following meaning:

1. The characteristic exponent α , $0 < \alpha \leq 2$, determines the shape of the distribution and hence the flatness of the tails.
2. The skewness (or alternatively, symmetry) parameter β , $-1 < \beta < 1$, determines the symmetry of the distribution: $\beta = 0$ specifies a symmetric distribution, $\beta < 0$ a left-skewed distribution, and $\beta > 0$ a right-skewed distribution.
3. The scale parameter σ , $0 \leq \sigma < \infty$, determines the range or dispersion of the stable distribution.
4. The location parameter μ , $-\infty < \mu < \infty$, determines the center of the distribution.

We next list some useful properties of stable random variables.

95

- For $0 < \alpha < 2$, the tails of a stable distribution decay as a power law, that is,

$$\Pr[|x(\zeta) - \mu| \geq x] \simeq \frac{C}{x^\alpha} \quad \text{as } x \rightarrow \infty \quad (3.2.84)$$

where C is a constant that depends on the scale parameter σ . As a result of this behavior, α -stable random variables have infinite second-order moments. In particular,

$$\begin{aligned} E\{|x(\zeta)|^p\} &< \infty & \text{for any } 0 < p \leq \alpha \\ E\{|x(\zeta)|^p\} &= \infty & \text{for any } p > \alpha \end{aligned} \quad (3.2.85)$$

Also $\text{var}[x(\zeta)] = \infty$ for $0 < \alpha < 2$, and $E\{|x(\zeta)|\} = \infty$ if $0 < \alpha < 1$.

- A stable distribution is symmetric about μ iff $\beta = 0$. A symmetric α -stable distribution is denoted as $S\alpha S$, and its characteristic function is given by

$$\Phi(\xi) = \exp(j\mu\xi - |\sigma\xi|^\alpha) \quad (3.2.86)$$

- If $x(\zeta)$ is $S\alpha S$ with $\alpha = 2$ in (3.2.83), we have a Gaussian distribution with variance equal to $2\sigma^2$, that is, $\mathcal{N}(\mu, 2\sigma^2)$, whose tails decay exponentially and not as a power law. Thus, the Gaussian is the only stable distribution with finite variance.
- If $x(\zeta)$ is $S\alpha S$ with $\alpha = 1$, we have a Cauchy distribution with density

$$f_x(x) = \frac{\sigma/\pi}{(x - \mu)^2 + \sigma^2} \quad (3.2.87)$$

A standard ($\mu = 0, \sigma = 1$) Cauchy random variable $x(\zeta)$ can be generated from a $[0, 1]$ uniform random variable $u(\zeta)$, by using the transformation $x = \tan[\pi(u - \frac{1}{2})]$.

- If $x(\zeta)$ is $S\alpha S$ with $\alpha = \frac{1}{2}$, we have a *Levy distribution*, which has both infinite variance and infinite mean. The pdf of this distribution does not have a functional form and hence must be computed numerically.

In Figure 3.5, we display characteristic and density functions of Gaussian, Cauchy, and Levy random variables. The density plots were computed numerically using the MATLAB function `stablepdf`.

Infinitely divisible distributions. A distribution $F_x(x)$ is infinitely divisible if and only if for each M there exists a distribution $F_M(x)$ such that

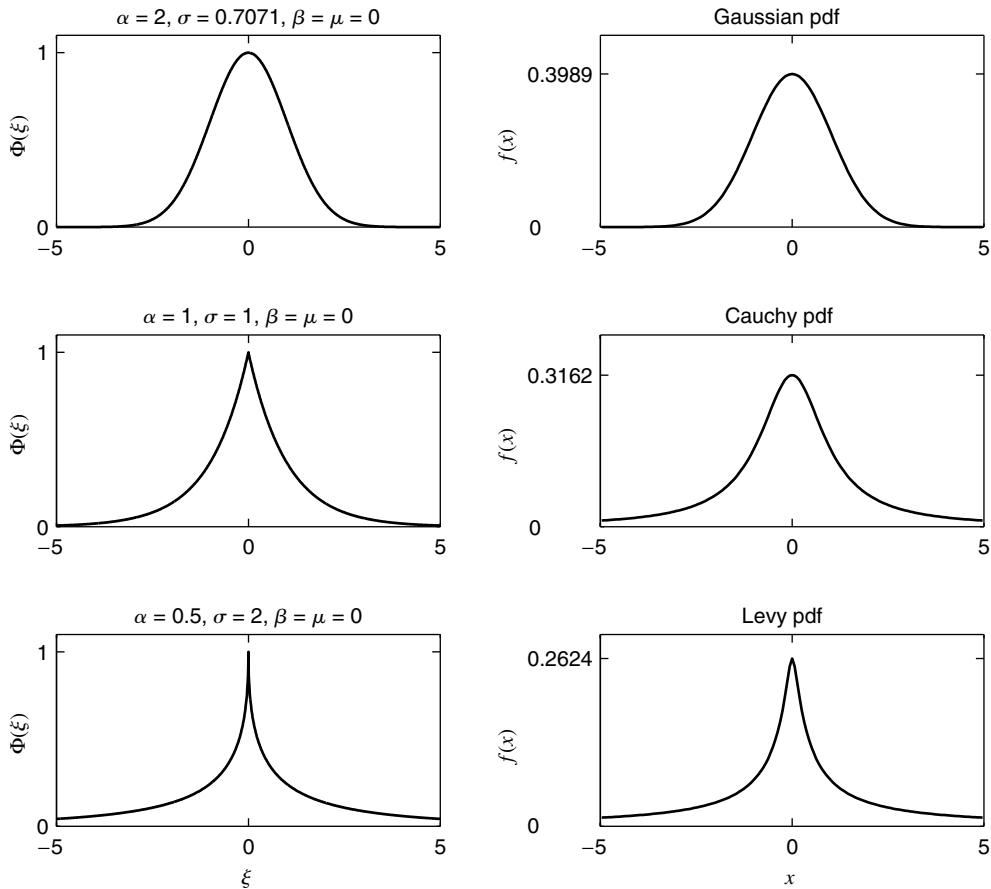
$$f_x(x) = f_M(x) * f_M(x) * \cdots * f_M(x) \quad (3.2.88)$$

or by using the convolution theorem,

$$\Phi_x(\xi) = \Phi_M(\xi) \Phi_M(\xi) \cdots \Phi_M(\xi) = \Phi_M^M(\xi) \quad (3.2.89)$$

that is, for each M the random variable $x(\zeta)$ can be represented as the sum $x(\zeta) = x_1(\zeta) + \cdots + x_M(\zeta)$ of M IID random variables with a *common* distribution $F_M(x)$. Clearly, all stable distributions are infinitely divisible. An example of an infinitely divisible pdf is shown in Figure 3.6 for $M = 4, \alpha = 1.5, \mu = 0$, and $\beta = 0$.

Central limit theorem. Consider the random variable $y(\zeta)$ defined in (3.2.55). We would like to know about the convergence of its distribution as $M \rightarrow \infty$. If $y(\zeta)$ is a sum of IID random variables with a stable distribution, the distribution of $y(\zeta)$ also converges to a stable distribution. What result should we expect if the individual distributions are not stable and, in particular, are of finite variance? As we observed in Example 3.2.1, the sum of uniformly distributed independent random variables appears to converge to a Gaussian distribution. Is this result valid for any other distribution? The following version of the CLT answers these questions.

**FIGURE 3.5**

The characteristic and density function plots of Gaussian, Cauchy, and Levy random variables.

THEOREM 3.1 (CENTRAL LIMIT THEOREM). Let $\{x_k(\zeta)\}_{k=1}^M$ be a collection of random variables such that $x_1(\zeta), x_2(\zeta), \dots, x_M(\zeta)$ (a) are mutually independent and (b) have the same distribution, and (c) the mean and variance of each random variable exist and are finite, that is, $\mu_{x_k} < \infty$ and $\sigma_{x_k}^2 < \infty$ for all $k = 1, 2, \dots, M$. Then, the distribution of the normalized sum

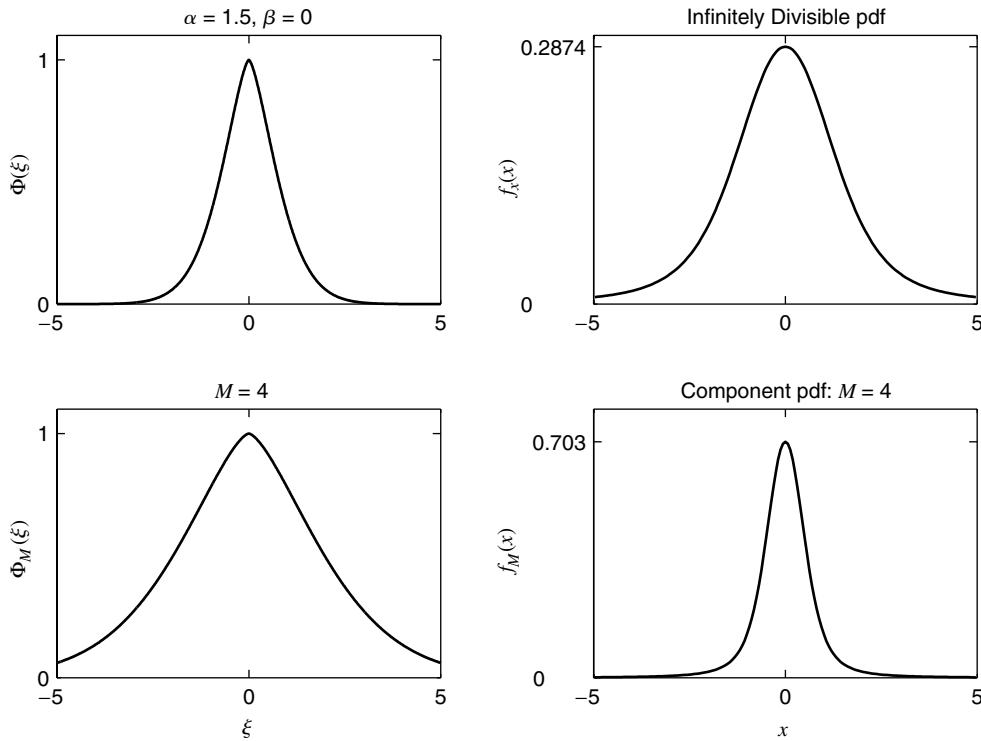
$$y_M(\zeta) = \frac{\sum_{k=1}^M x_k(\zeta) - \mu_{y_M}}{\sigma_{y_M}}$$

approaches that of a normal random variable with zero mean and unit standard deviation as $M \rightarrow \infty$.

Proof. See Borkar (1995).

Comments. The following important comments are in order regarding the CLT.

1. Since we are assuming IID components in the normalized sum, the above theorem is known as the *equal-component case of the CLT*.
2. It should be emphasized that the convergence in the above theorem is in *distribution* (cdf) and not necessarily in density (pdf). Suppose we have M discrete and IID random variables. Then their normalized sum will always remain discrete no matter how large M is, but the distribution of the sum will converge to the integral of the Gaussian pdf.

**FIGURE 3.6**

The characteristic and density function plots of an infinitely divisible distribution.

3. The word *central* in the CLT is a reminder that the distribution converges to the Gaussian distribution *around the center*, that is, around the mean. Note that while the limit distribution is found to be Gaussian, frequently the Gaussian limit gives a poor approximation for the tails of the actual distribution function of the sum when M is finite, even though the actual value under consideration might seem to be quite large.
4. As a final point, we note that in the above theorem the assumption of finite variance is critical to obtain a Gaussian limit. This implies that *all* distributions with finite variance will converge to the Gaussian when independent copies of their random variables are added. What happens if the variance is infinite? Then in this case the sum converges to one of the stable distributions. For example, as shown in Example 3.2.3, the sum of Cauchy random variables converges to a Cauchy distribution.

3.3 DISCRETE-TIME STOCHASTIC PROCESSES

After studying random variables and vectors, we can now extend these concepts to discrete-time signals (or sequences). Many natural sequences can be characterized as random signals because we cannot determine their values precisely, that is, they are unpredictable. A natural mathematical framework for the description of these discrete-time random signals is provided by discrete-time stochastic processes.

To obtain a formal definition, consider an experiment with a finite or infinite number of unpredictable outcomes from a sample space $\mathcal{S} = \{\zeta_1, \zeta_2, \dots\}$, each occurring with a probability $\Pr\{\zeta_k\}, k = 1, 2, \dots$. By some rule we assign to each element ζ_k of \mathcal{S} a deterministic sequence $x(n, \zeta_k), -\infty < n < \infty$. The sample space \mathcal{S} , the probabilities $\Pr\{\zeta_k\}$, and the sequences $x(n, \zeta_k), -\infty < n < \infty$, constitute a *discrete-time stochastic*

process or random sequence. Formally,

$x(n, \zeta), -\infty < n < \infty$, is a random sequence if for a fixed value n_0 of n , $x(n_0, \zeta)$ is a random variable.

The set of all possible sequences $\{x(n, \zeta)\}$ is called an *ensemble*, and each individual sequence $x(n, \zeta_k)$, corresponding to a specific value of $\zeta = \zeta_k$, is called a *realization* or a *sample sequence* of the ensemble.

There are four possible interpretations of $x(n, \zeta)$, depending on the character of n and ζ , as illustrated in Figure 3.7:

- $x(n, \zeta)$ is a random variable if n is *fixed* and ζ is a variable.
- $x(n, \zeta)$ is a sample sequence if ζ is *fixed* and n is a variable.
- $x(n, \zeta)$ is a number if both n and ζ are *fixed*.
- $x(n, \zeta)$ is a stochastic process if both n and ζ are variables.

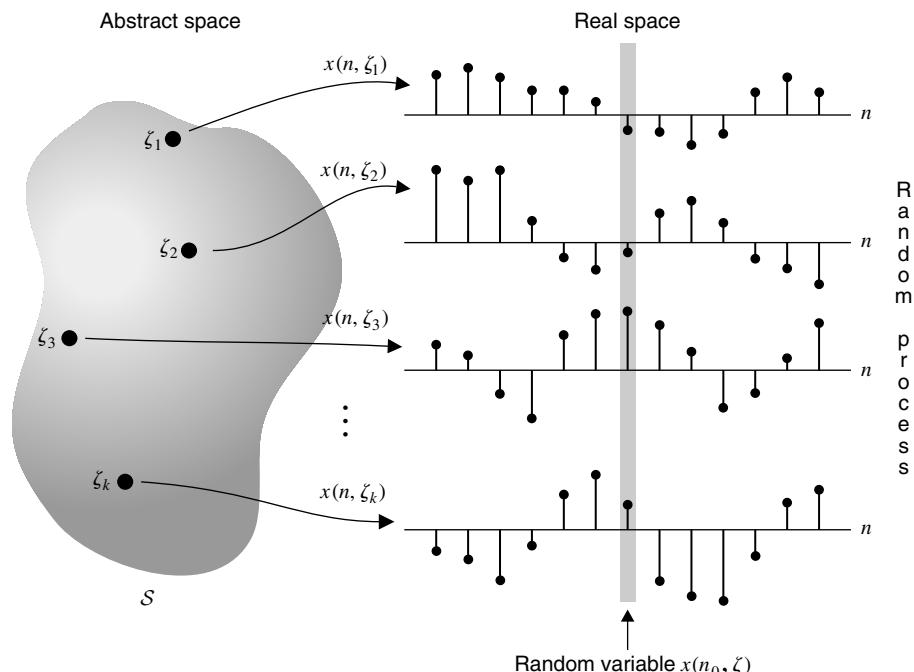


FIGURE 3.7

Graphical description of random sequences.

A random sequence is also called a *time series* in the statistics literature. It is a sequence of random variables, or it can be thought of as an *infinite-dimensional* random vector. As with any collection of infinite objects, one has to be careful with the asymptotic (or convergence) properties of a random sequence. If n is a continuous variable taking values in \mathbb{R} , then $x(n, \zeta)$ is an *uncountable* collection of random variables or an ensemble of waveforms. This ensemble is called a continuous-time stochastic process or a *random process*. Although these processes can be handled similarly to sequences, they are more difficult to deal with in a rigorous mathematical manner than sequences are. Furthermore, practical signal processing requires discrete-time signals. Hence in this book we consider random sequences rather than random waveforms.

Finally, in passing we note that the word *stochastic* is derived from the Greek word *stochastikos*, which means skillful in aiming or guessing. Hence, the terms *random process* and *stochastic process* will be used interchangeably throughout this book.

As mentioned before, a deterministic signal is by definition exactly predictable. This assumes that there exists a certain functional relationship that completely describes the signal, even if this relationship is not available. The unpredictability of a random process is, in general, the combined result of two things. First, the selection of a single realization is based on the outcome of a random experiment. Second, no functional description is available for all realizations of the ensemble. However, in some special cases, such a functional relationship is available. This means that after the occurrence of a specific realization, its future values can be predicted exactly from its past ones. If the future samples of any realization of a stochastic process can be predicted from the past ones, the process is called *predictable* or *deterministic*; otherwise, it is said to be a *regular process*. For example, the process $x(n, \zeta) = c$, where c is a random variable, is a predictable stochastic process because every realization is a discrete-time signal with constant amplitude. In practice, we most often deal with regular stochastic processes.

The simplest description of any random signal is provided by an amplitude-versus-time plot. Inspection of this plot provides qualitative information about some significant features of the signal that are useful in many applications. These features include, among others, the following:

1. The frequency of occurrence of various signal amplitudes, described by the probability distribution of samples.
2. The degree of dependence between two signal samples, described by the correlation between them.
3. The existence of “cycles” or quasi-periodic patterns, obtained from the signal power spectrum (which will be described in Section 3.3.6).
4. Indications of variability in the mean, variance, probability density, or spectral content.

The first feature above, the amplitude distribution, is obtained by plotting the histogram, which is an estimate of the first-order probability density of the underlying stochastic process. The probability density indicates waveform features such as “spikiness” and boundedness. Its form is crucial in the design of reliable estimators, quantizers, and event detectors.

The dependence between two signal samples (which are random variables) is given theoretically by the autocorrelation sequence and is quantified in practice by the empirical correlation (see Chapter 1), which is an estimate of the autocorrelation sequence of the underlying process. It affects the rate of amplitude change from sample to sample.

Cycles in the data are related to sharp peaks in the power spectrum or periodicity in the autocorrelation. Although the power spectrum and the autocorrelation contain the same information, they present it in different fashions.

Variability in a given quantity (e.g., variance) can be studied by evaluating this quantity for segments that can be assumed locally stationary and then analyzing the segment-to-segment variation. Such short-term descriptions should be distinguished from long-term ones, where the whole signal is analyzed as a single segment.

All the above features, to a lesser or greater extent, are interrelated. Therefore, it is impossible to point out exactly the effect of each one upon the visual appearance of the signal. However, a lot of insight can be gained by introducing the concepts of signal variability and signal memory, which are discussed in Sections 3.3.5 and 3.4.3 respectively.

3.3.1 Description Using Probability Functions

From Figure 3.7, it is clear that at $n = n_0$, $x(n_0, \zeta)$ is a random variable that requires a first-order probability function, say cdf $F_x(x; n_0)$, for its description. Similarly, $x(n_1, \zeta)$ and $x(n_2, \zeta)$ are joint random variables at instances n_1 and n_2 , respectively, requiring a joint cdf $F_x(x_1, x_2; n_1, n_2)$. Stochastic processes contain infinitely many such random variables. Hence they are completely described, in a statistical sense, if their k th-order distribution

function

$$F_x(x_1, \dots, x_k; n_1, \dots, n_k) = \Pr\{x(n_1) \leq x_1, \dots, x(n_k) \leq x_k\} \quad (3.3.1)$$

is known for every value of $k \geq 1$ and for all instances n_1, n_2, \dots, n_k . The k th-order pdf is given by

$$f_x(x_1, \dots, x_k; n_1, \dots, n_k) \triangleq \frac{\partial^{2k} F_x(x_1, \dots, x_k; n_1, \dots, n_k)}{\partial x_{R1} \cdots \partial x_{Ik}} \quad k \geq 1 \quad (3.3.2)$$

Clearly, the probabilistic description requires a lot of information that is difficult to obtain in practice except for simple stochastic processes. However, many (but not all) properties of a stochastic process can be described in terms of *averages* associated with its first- and second-order densities.

For simplicity, in the rest of the book, we will use a compact notation $x(n)$ to represent either a random process $x(n, \zeta)$ or a single realization $x(n)$, which is a member of the ensemble. Thus we will drop the variable ζ from all notations involving random variables, vectors, or processes. We believe that this will not cause any confusion and that the exact meaning will be clear from the context. Also the random process $x(n)$ is assumed to be complex-valued unless explicitly specified as real-valued.

3.3.2 Second-Order Statistical Description

The second-order statistic of $x(n)$ at time n is specified by its *mean value* $\mu_x(n)$ and its *variance* $\sigma_x^2(n)$, defined by

$$\mu_x(n) = E\{x(n)\} = E\{x_R(n) + jx_I(n)\} \quad (3.3.3)$$

$$\text{and} \quad \sigma_x^2(n) = E\{|x(n) - \mu_x(n)|^2\} = E\{|x(n)|^2\} - |\mu_x(n)|^2 \quad (3.3.4)$$

respectively. Note that both $\mu_x(n)$ and $\sigma_x(n)$ are, in general, deterministic sequences.

The second-order statistics of $x(n)$ at two different times n_1 and n_2 are given by the two-dimensional autocorrelation (or autocovariance) sequences. The *autocorrelation sequence* of a discrete-time random process is defined as the joint moment of the random variables $x(n_1)$ and $x(n_2)$, that is,

$$r_{xx}(n_1, n_2) = E\{x(n_1)x^*(n_2)\} \quad (3.3.5)$$

It provides a measure of the dependence between values of the process at two different times. In this sense, it also provides information about the time variation of the process. The *autocovariance* sequence of $x(n)$ is defined by

$$\begin{aligned} \gamma_{xx}(n_1, n_2) &= E\{[x(n_1) - \mu_x(n_1)][x(n_2) - \mu_x(n_2)]^*\} \\ &= r_{xx}(n_1, n_2) - \mu_x(n_1)\mu_x^*(n_2) \end{aligned} \quad (3.3.6)$$

We will use notations such as $\gamma_x(n_1, n_2)$, $r_x(n_1, n_2)$, $\gamma(n_1, n_2)$, or $r(n_1, n_2)$ when there is no confusion as to which signal we are referring. Note that, in general, the second-order statistics are defined on a two-dimensional grid of integers.

The statistical relation between two stochastic processes $x(n)$ and $y(n)$ that are jointly distributed (i.e., they are defined on the same sample space \mathcal{S}) can be described by their *cross-correlation* and *cross-covariance* functions, defined by

$$r_{xy}(n_1, n_2) = E\{x(n_1)y^*(n_2)\} \quad (3.3.7)$$

$$\begin{aligned} \text{and} \quad \gamma_{xy}(n_1, n_2) &= E\{[x(n_1) - \mu_x(n_1)][y(n_2) - \mu_y(n_2)]^*\} \\ &= r_{xy}(n_1, n_2) - \mu_x(n_1)\mu_y^*(n_2) \end{aligned} \quad (3.3.8)$$

The *normalized cross-correlation* of two random processes $x(n)$ and $y(n)$ is defined by

$$\rho_{xy}(n_1, n_2) = \frac{\gamma_{xy}(n_1, n_2)}{\sigma_x(n_1)\sigma_y(n_2)} \quad (3.3.9)$$

Some definitions

We now describe some useful types of stochastic processes based on their statistical properties. A random process is said to be

- An *independent* process if

$$f_x(x_1, \dots, x_k; n_1, \dots, n_k) = f_1(x_1; n_1) \cdots f_k(x_k; n_k) \quad \forall k, n_i, i = 1, \dots, k \quad (3.3.10)$$

that is, $x(n)$ is a sequence of independent random variables. If all random variables have the same pdf $f(x)$ for all k , then $x(n)$ is called an IID (independent and identically distributed) random sequence.

- An *uncorrelated* process if $x(n)$ is a sequence of uncorrelated random variables, that is,

$$\gamma_x(n_1, n_2) = \begin{cases} \sigma_x^2(n_1) & n_1 = n_2 \\ 0 & n_1 \neq n_2 \end{cases} = \sigma_x^2(n_1) \delta(n_1 - n_2) \quad (3.3.11)$$

Alternatively, we have

$$r_x(n_1, n_2) = \begin{cases} \sigma_x^2(n_1) + |\mu_x(n_1)|^2 & n_1 = n_2 \\ \mu_x(n_1)\mu_x^*(n_2) & n_1 \neq n_2 \end{cases} \quad (3.3.12)$$

- An *orthogonal* process if it is a sequence of orthogonal random variables, that is,

$$r_x(n_1, n_2) = \begin{cases} \sigma_x^2(n_1) + |\mu_x(n_1)|^2 & n_1 = n_2 \\ 0 & n_1 \neq n_2 \end{cases} = E\{|x(n_1)|^2\}\delta(n_1 - n_2) \quad (3.3.13)$$

- An *independent increment* process if $\forall k > 1$ and $\forall n_1 < n_2 < \dots < n_k$, the *increments*

$$\{x(n_1)\}, \{x(n_2) - x(n_1)\}, \dots, \{x(n_k) - x(n_{k-1})\}$$

are jointly independent. For such sequences, the k th-order probability function can be constructed as products of the probability functions of its increments.

- A *wide-sense periodic (WSP)* process with period N if

$$\mu_x(n) = \mu_x(n + N) \quad \forall n \quad (3.3.14)$$

and $r_x(n_1, n_2) = r_x(n_1 + N, n_2) = r_x(n_1, n_2 + N) = r_x(n_1 + N, n_2 + N)$ $\quad (3.3.15)$

Note that in the above definition, $\mu_x(n)$ is periodic in one dimension while $r_x(n_1, n_2)$ is periodic in two dimensions.

- A *wise-sense cyclostationary* process if there exists an integer N such that

$$\mu_x(n) = \mu_x(n + N) \quad \forall n \quad (3.3.16)$$

and $r_x(n_1, n_2) = r_x(n_1 + N, n_2 + N)$ $\quad (3.3.17)$

Note that in the above definition, $r_x(n_1, n_2)$ is *not* periodic in a two-dimensional sense. The correlation sequence is invariant to shift by N in *both* of its arguments.

- If all k th-order distributions of a stochastic process are jointly Gaussian, then it is called a Gaussian random sequence.

We can also extend some of these definitions to the case of two joint stochastic processes. The random processes $x(n)$ and $y(n)$ are said to be

- *Statistically independent* if for all values of n_1 and n_2

$$f_{xy}(x, y; n_1, n_2) = f_x(x; n_1)f_y(y; n_2) \quad (3.3.18)$$

- *Uncorrelated* if for every n_1 and n_2

$$\gamma_{xy}(n_1, n_2) = 0 \quad \text{or} \quad r_{xy}(n_1, n_2) = \mu_x(n_1)\mu_y^*(n_2) \quad (3.3.19)$$

- *Orthogonal* if for every n_1 and n_2

$$r_{xy}(n_1, n_2) = 0 \quad (3.3.20)$$

3.3.3 Stationarity

A random process $x(n)$ is called *stationary* if statistics determined for $x(n)$ are equal to those for $x(n + k)$, for every k . More specifically, we have the following definition.

DEFINITION 3.3 (STATIONARY OF ORDER N). A stochastic process $x(n)$ is called *stationary of order N* if

$$f_x(x_1, \dots, x_N; n_1, \dots, n_N) = f_x(x_1, \dots, x_N; n_1 + k, \dots, n_N + k) \quad (3.3.21)$$

for any value of k . If $x(n)$ is stationary for all orders $N = 1, 2, \dots$, it is said to be *strict-sense stationary (SSS)*.

An IID sequence is SSS. However, SSS is more restrictive than necessary for most practical applications. A more relaxed form of stationarity, which is sufficient for practical problems, occurs when a random process is *stationary up to order 2*, and it is also known as *wide-sense stationarity*.

DEFINITION 3.4 (WIDE-SENSE STATIONARITY). A random signal $x(n)$ is called *wide-sense stationary (WSS)* if

1. Its mean is a constant independent of n , that is,

$$E\{x(n)\} = \mu_x \quad (3.3.22)$$

2. Its variance is also a constant independent of n , that is,

$$\text{var}[x(n)] = \sigma_x^2 \quad (3.3.23)$$

and

3. Its autocorrelation depends only on the distance $l = n_1 - n_2$, called *lag*, that is,

$$r_x(n_1, n_2) = r_x(n_1 - n_2) = r_x(l) = E\{x(n + l)x^*(n)\} = E\{x(n)x^*(n - l)\} \quad (3.3.24)$$

From (3.3.22), (3.3.24), and (3.3.6) it follows that the autocovariance of a WSS signal also depends only on $l = n_1 - n_2$, that is,

$$\gamma_x(l) = r_x(l) - |\mu_x|^2 \quad (3.3.25)$$

EXAMPLE 3.3.1. Let $w(n)$ be a zero-mean, uncorrelated Gaussian random sequence with variance $\sigma^2(w) = 1$.

- a. Characterize the random sequence $w(n)$.
- b. Define $x(n) = w(n) + w(n - 1)$, $-\infty < n < \infty$. Determine the mean and autocorrelation of $x(n)$. Also characterize $x(n)$.

Solution. Note that the variance of $w(n)$ is a constant.

- a. Since uncorrelatedness implies independence for Gaussian random variables, $w(n)$ is an independent random sequence. Since its mean and variance are constants, it is at least stationary in the first order. Furthermore, from (3.3.12) or (3.3.13) we have

$$r_w(n_1, n_2) = \sigma^2 \delta(n_1 - n_2) = \delta(n_1 - n_2)$$

Hence $w(n)$ is also a WSS random process.

- b. The mean of $x(n)$ is zero for all n since $w(n)$ is a zero-mean process. Consider

$$\begin{aligned} r_x(n_1, n_2) &= E\{x(n_1)x(n_2)\} \\ &= E\{[w(n_1) + w(n_1 - 1)][w(n_2) + w(n_2 - 1)]\} \\ &= r_w(n_1, n_2) + r_w(n_1, n_2 - 1) + r_w(n_1 - 1, n_2) \\ &\quad + r_w(n_1 - 1, n_2 - 1) \\ &= \sigma^2 \delta(n_1 - n_2) + \sigma^2 \delta(n_1 - n_2 + 1) \\ &\quad + \sigma^2 \delta(n_1 - 1 - n_2) + \sigma^2 \delta(n_1 - 1 - n_2 + 1) \\ &= 2\delta(n_1 - n_2) + \delta(n_1 - n_2 + 1) + \delta(n_1 - n_2 - 1) \end{aligned}$$

Clearly, $r_x(n_1, n_2)$ is a function of $n_1 - n_2$. Hence

$$r_x(l) = 2\delta(l) + \delta(l+1) + \delta(l-1)$$

Therefore, $x(n)$ is a WSS sequence. However, it is not an independent random sequence since both $x(n)$ and $x(n+1)$ depend on $w(n)$.

EXAMPLE 3.3.2 (WIENER PROCESS). Toss a fair coin at each n , $-\infty < n < \infty$. Let

$$w(n) = \begin{cases} +S & \text{if heads is outcome} \\ -S & \text{if tails is outcome} \end{cases} \quad \begin{array}{ll} \Pr(H) = 0.5 & \\ \Pr(T) = 0.5 & \end{array}$$

where S is a step size. Clearly, $w(n)$ is an independent random process with

$$E\{w(n)\} = 0$$

and

$$E\{w^2(n)\} = \sigma_w^2 = S^2 \left(\frac{1}{2} \right) + S^2 \left(\frac{1}{2} \right) = S^2$$

Define a new random process $x(n)$, $n \geq 1$, as

$$\begin{aligned} x(1) &= w(1) \\ x(2) &= x(1) + w(2) = w(1) + w(2) \\ &\vdots \\ x(n) &= x(n-1) + w(n) = \sum_{i=1}^n w(i) \end{aligned}$$

Note that $x(n)$ is a running sum of independent steps or increments; thus it is an independent increment process. Such a sequence is called a *discrete Wiener process* or *random walk*. We can easily see that

$$\begin{aligned} E\{x(n)\} &= E \left\{ \sum_{i=1}^n w(i) \right\} = 0 \\ \text{and } E\{x^2(n)\} &= E \left\{ \sum_{i=1}^n w(i) \sum_{k=1}^n w(k) \right\} = E \left\{ \sum_{i=1}^n \sum_{k=1}^n w(i)w(k) \right\} \\ &= \sum_{i=1}^n \sum_{k=1}^n E\{w(i)w(k)\} = \sum_{i=1}^n E\{w^2(i)\} = nS^2 \end{aligned}$$

Therefore, random walk is a nonstationary (or *evolutionary*) process with zero mean and variance that grows with n , the number of steps taken.

It should be stressed at this point that although any strict-sense stationary signal is wide-sense stationary, the inverse is not always true, except if the signal is Gaussian. However in practice, it is very rare to encounter a signal that is stationary in the wide sense but not stationary in the strict sense (Papoulis 1991).

Two random signals $x(n)$ and $y(n)$ are called *jointly wide-sense stationary* if each is wide-sense stationary and their cross-correlation depends only on $l = n_1 - n_2$

$$r_{xy}(l) = E\{x(n)y^*(n-l)\}; \gamma_{x,y}(l) = r_{xy}(l) - \mu_x \mu_y^* \quad (3.3.26)$$

Note that as a consequence of wide-sense stationarity the two-dimensional correlation and covariance sequences become one-dimensional sequences. This is a very important result that ultimately allows for a nice spectral description of stationary random processes.

Properties of autocorrelation sequences

The autocorrelation sequence of a stationary process has many important properties (which also apply to autocovariance sequences, but we will discuss mostly correlation sequences). Vector versions of these properties are discussed extensively in Section 3.4.4, and their proofs are explored in the problems.

PROPERTY 3.3.1. The average power of a WSS process $x(n)$ satisfies

$$r_x(0) = \sigma_x^2 + |\mu_x|^2 \geq 0 \quad (3.3.27)$$

$$\text{and} \quad r_x(0) \geq |r_x(l)| \quad \text{for all } l \quad (3.3.28)$$

Proof. See Problem 3.21 and Property 3.3.6.

This property implies that the correlation attains its maximum value at zero lag and this value is nonnegative. The quantity $|\mu_x|^2$ is referred to as the *average dc power*, and the quantity $\sigma_x^2 = \gamma_x(0)$ is referred to as the *average ac power* of the random sequence. The quantity $r_x(0)$ then is the *total* average power of $x(n)$.

PROPERTY 3.3.2. The autocorrelation sequence $r_x(l)$ is a conjugate symmetric function of lag l , that is,

$$r_x^*(-l) = r_x(l) \quad (3.3.29)$$

Proof. It follows from Definition 3.4 and from (3.3.24).

PROPERTY 3.3.3. The autocorrelation sequence $r_x(l)$ is nonnegative definite; that is, for any $M > 0$ and any vector $\alpha \in \mathbb{R}^M$

$$\sum_{k=1}^M \sum_{m=1}^M \alpha_k r_x(k-m) \alpha_m^* \geq 0 \quad (3.3.30)$$

This is a necessary and sufficient condition for a sequence $r_x(l)$ to be the autocorrelation sequence of a random sequence.

Proof. See Problem 3.22.

Since in this book we exclusively deal with wide-sense stationary processes, we will use the term *stationary* to mean wide-sense stationary. The properties of autocorrelation and cross-correlation sequences of jointly stationary processes, $x(n)$ and $y(n)$, are summarized in Table 3.1.

Although SSS and WSS forms are widely used in practice, there are processes with different forms of stationarity. Consider the following example.

EXAMPLE 3.3.3. Let $x(n)$ be a real-valued random process generated by the system

$$x(n) = \alpha x(n-1) + w(n) \quad n \geq 0 \quad x(-1) = 0 \quad (3.3.31)$$

where $w(n)$ is a stationary random process with mean μ_w and $r_w(l) = \sigma_w^2 \delta(l)$. The process $x(n)$ generated using (3.3.31) is known as a *first-order autoregressive*, or AR(1), process,[†] and the process $w(n)$ is known as a *white noise* process (defined in Section 3.3.6). Determine the mean $\mu_x(n)$ of $x(n)$ and comment on its stationarity.

Solution. To compute the mean of $x(n)$, we express it as a function of $\{w(n), w(n-1), \dots, w(0)\}$ as follows

$$\begin{aligned} x(0) &= \alpha x(-1) + w(0) = w(0) \\ x(1) &= \alpha x(0) + w(1) = \alpha w(0) + w(1) \\ &\vdots \\ x(n) &= \alpha^n w(0) + \alpha^{n-1} w(1) + \dots + w(n) = \sum_{k=0}^n \alpha^k w(n-k) \end{aligned}$$

[†]Note that from (3.3.31), $x(n-1)$ completely determines the distribution for $x(n)$, and $x(n)$ completely determines the distribution for $x(n+1)$, and so on. If

$$f_{x(n)|x(n-1)\dots}(x_n|x_{n-1}\dots) = f_{x(n)|x(n-1)}(x_n|x_{n-1})$$

then the process is termed a *Markov process*.

Hence the mean of $x(n)$ is given by

$$\mu_x(n) = E \left\{ \sum_{k=0}^n \alpha^k w(n-k) \right\} = \mu_w \left(\sum_{k=0}^n \alpha^k \right) = \begin{cases} \frac{1-\alpha^{n+1}}{1-\alpha} \mu_w & \alpha \neq 1 \\ (n+1)\mu_w & \alpha = 1 \end{cases}$$

Clearly, the mean of $x(n)$ depends on n , and hence it is nonstationary. However, if we assume that $|\alpha| < 1$ (which implies that the system is BIBO stable), then as $n \rightarrow \infty$, we obtain

$$\mu_x(n) = \mu_w \frac{1-\alpha^{n+1}}{1-\alpha} \xrightarrow{n \rightarrow \infty} \frac{\mu_w}{1-\alpha}$$

Thus $x(n)$ approaches first-order stationarity for large n . Similar analysis for the autocorrelation of $x(n)$ shows that $x(n)$ approaches wide-sense stationarity for large n (see Problem 3.23).

The above example illustrates a form of stationarity called *asymptotic* stationarity. A stochastic process $x(n)$ is *asymptotically stationary* if the statistics of random variables $x(n)$ and $x(n+k)$ become stationary as $k \rightarrow \infty$. When LTI systems are driven by zero-mean uncorrelated-component random processes, the output process becomes asymptotically stationary in the *steady state*. Another useful form of stationarity is given by stationary increments. If the increments $\{x(n) - x(n-k)\}$ of a process $x(n)$ form a stationary process for every k , we say that $x(n)$ is a process with *stationary increments*. Such processes can be used to model data in various practical applications (see Chapter 12).

The simplest way, to examine in practice if a real-world signal is stationary, is to investigate the physical mechanism that produces the signal. If this mechanism is time-invariant, then the signal is stationary. In case it is impossible to draw a conclusion based on physical considerations, we should rely on statistical methods (Bendat and Piersol 1986; Priestley 1981). Note that stationarity in practice means that a random signal has statistical properties that do not change over the time interval we observe the signal. For evolutionary signals the statistical properties change continuously with time. An example of a highly nonstationary random signal is the signals associated with the vibrations induced in space vehicles during launch and reentry. However, there is a kind of random signal whose statistical properties change slowly with time. Such signals, which are stationary over short periods, are called *locally stationary* signals. Many signals of great practical interest, such as speech, EEG, and ECG, belong to this family of signals.

Finally, we note that general techniques for the analysis of nonstationary signals do not exist. Thus only special methods that apply to specific types of nonstationary signals can be developed. Many such methods remove the nonstationary component of the signal, leaving behind another component that can be analyzed as stationary (Bendat and Piersol 1986; Priestley 1981).

3.3.4 Ergodicity

A stochastic process consists of the ensemble and a probability law. If this information is available, the statistical properties of the process can be determined in a quite straightforward manner. However, in the real world, we have access to only a limited number (usually one) of realizations of the process. The question that arises then is, Can we infer the statistical characteristics of the process from a single realization?

This is possible for the class of random processes that are called *ergodic* processes. Roughly speaking, ergodicity implies that all the statistical information can be obtained from any single representative member of the ensemble.

Time averages

All the statistical averages that we have defined up to this point are known as *ensemble averages* because they are obtained by “freezing” the time variable and averaging over the ensemble (see Fig. 3.7). Averages of this type are formally defined by using the expectation

operator $E\{ \cdot \}$. Ensemble averaging is not used frequently in practice, because it is impractical to obtain the number of realizations needed for an accurate estimate. Thus the need for a different kind of average, based on only one realization, naturally arises. Obviously such an average can be obtained only by time averaging.

The *time average* of a quantity, related to a discrete-time random signal, is defined as

$$\langle(\cdot)\rangle \triangleq \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N (\cdot) \quad (3.3.32)$$

Note that, owing to its dependence on a single realization, any time average is itself a random variable. The time average is taken over all time because all realizations of a stationary random process exist for all time; that is, they are power signals.

For every ensemble average we can define a corresponding time average. The following time averages are of special interest:

$$\begin{aligned} \text{Mean value} &= \langle x(n) \rangle \\ \text{Mean square} &= \langle |x(n)|^2 \rangle \\ \text{Variance} &= \langle |x(n) - \langle x(n) \rangle|^2 \rangle \\ \text{Autocorrelation} &= \langle x(n)x^*(n-l) \rangle \\ \text{Autocovariance} &= \langle [x(n) - \langle x(n) \rangle][x(n-l) - \langle x(n) \rangle]^* \rangle \\ \text{Cross-correlation} &= \langle x(n)y^*(n-l) \rangle \\ \text{Cross-covariance} &= \langle [x(n) - \langle x(n) \rangle][y(n-l) - \langle y(n) \rangle]^* \rangle \end{aligned} \quad (3.3.33)$$

It is necessary to mention at this point the remarkable similarity between time averages and the correlation sequences for deterministic power signals. Although this is just a formal similarity, due to the fact that random signals are power signals, both quantities have the same properties. However, we should always keep in mind that although time averages are random variables (because they are functions of ζ), the corresponding quantities for deterministic power signals are fixed numbers or deterministic sequences.

Ergodic random processes

As we have already mentioned, in many practical applications only one realization of a random signal is available instead of the entire ensemble. In general, a single member of the ensemble does not provide information about the statistics of the process. However, if the process is stationary and ergodic, then all statistical information can be derived from only one typical realization of the process.

A random signal $x(n)$ is called *ergodic*[†] if its ensemble averages equal appropriate time averages. There are several degrees of ergodicity (Papoulis 1991). We will discuss two of them: ergodicity in the mean and ergodicity in correlation.

DEFINITION 3.5 (ERGODIC IN THE MEAN). A random process $x(n)$ is ergodic *in the mean* if

$$\langle x(n) \rangle = E\{x(n)\} \quad (3.3.34)$$

DEFINITION 3.6 (ERGODIC IN CORRELATION). A random process $x(n)$ is *ergodic in correlation* if

$$\langle x(n)x^*(n-l) \rangle = E\{x(n)x^*(n-l)\} \quad (3.3.35)$$

Note that since $\langle x(n) \rangle$ is constant and $\langle x(n)x^*(n-l) \rangle$ is a function of l , if $x(n)$ is ergodic in both the mean and correlation, then it is also WSS. Thus only stationary signals can be ergodic. On the other hand, WSS does not imply ergodicity of any kind. Fortunately,

[†] Strictly speaking, the form of ergodicity that we will use is called *mean-square ergodicity* since the underlying convergence of random variables is in the mean-square sense (Stark and Woods 1994). Therefore, equalities in the definitions are in the mean-square sense.

in practice almost all stationary processes are also ergodic, which is very useful for the estimation of their statistical properties. From now on we will use the term *ergodic* to mean both ergodicity in the mean and ergodicity in correlation.

DEFINITION 3.7 (JOINT ERGODICITY). Two random signals are called *jointly ergodic* if they are individually ergodic and in addition

$$\langle x(n)y^*(n-l) \rangle = E\{x(n)y^*(n-l)\} \quad (3.3.36)$$

A physical interpretation of ergodicity is that one realization of the random signal $x(n)$, as time n tends to infinity, takes on values with the same statistics as the value $x(n_1)$, corresponding to all samples of the ensemble members at a given time $n = n_1$.

In practice, it is of course impossible to use the time-average formulas introduced above, because only finite records of data are available. In this case, it is common practice to replace the operator (3.3.32) by the operator

$$\langle(\cdot)\rangle_N = \frac{1}{2N+1} \sum_{n=-N}^N (\cdot) \quad (3.3.37)$$

to obtain *estimates* of the true quantities. Our desire in such problems is to find estimates that become increasingly accurate (in a sense to be defined in Section 3.6) as the length $2N + 1$ of the record of used data becomes larger.

Finally, to summarize, we note that whereas stationarity ensures the time invariance of the statistics of a random signal, ergodicity implies that any statistics can be calculated either by averaging over all members of the ensemble at a fixed time or by time-averaging over any single representative member of the ensemble.

3.3.5 Random Signal Variability

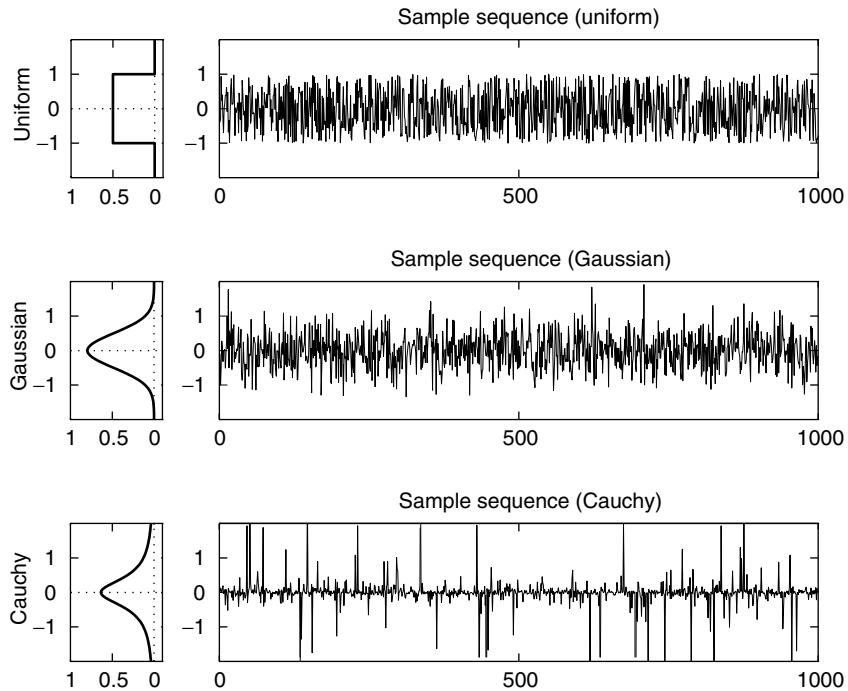
If we consider a stationary random sequence $w(n)$ that is IID with zero mean, its key characteristics depend on its first-order density. Figure 3.8 shows the probability density functions and sample realizations for IID processes with uniform, Gaussian, and Cauchy probability distributions. In the case of the uniform distribution, the amplitude of the random variable is limited to a range, with values occurring outside this interval with zero probability. On the other hand, the Gaussian distribution does not have a finite interval of support, allowing for the possibility of any value. The same is true of the Cauchy distribution, but its characteristics are dramatically different from those of the Gaussian distribution. The center lobe of the density is much narrower while the tails that extend out to infinity are significantly higher. As a result, the realization of the Cauchy random process contains numerous spikes or extreme values while the remainder of the process is more compact about the mean. Although the Gaussian random process allows for the possibility of large values, the probability of their occurrence is so small that they are not found in realizations of the process.

The major difference between the Gaussian and Cauchy distributions lies in the area found under the tails of the density as it extends out to infinity. This characteristic is related to the *variability* of the process. The heavy tails, as found in the Cauchy distribution, result in an abundance of spikes in the process, a characteristic referred to as *high variability*. On the other hand, a distribution such as the Gaussian does not allow for extreme values and indicates *low variability*. The extent of the variability of a given distribution is determined by the heaviness of the tails. Distributions with heavy tails are called *long-tailed* distributions and have been used extensively as models of impulsive random processes.

DEFINITION 3.8. A distribution is called *long-tailed* if its tails decay hyperbolically or algebraically as

$$Pr\{|x(n)| \geq x\} \sim Cx^{-\alpha} \quad \text{as } x \rightarrow \infty \quad (3.3.38)$$

where C is a constant and the variable α determines the rate of decay of the distribution.

**FIGURE 3.8**

Probability density functions and sample realizations of an IID process with uniform, Gaussian, and Cauchy distributions.

By means of comparison, the Gaussian distribution has an exponential rate of decay. The implication of the algebraically decaying tail is that the process has infinite variance, that is,

$$\sigma_x^2 = E\{|x(n)|^2\} = \infty$$

and therefore lacks second-order moments. The lack of second-order moments means that, in addition to the variance, the correlation functions of these processes do not exist. Since most signal processing algorithms are based on second-order moment theory, infinite variance has some extreme implications for the way in which such processes are treated.

In this book, we shall model high variability, and hence infinite variance, using the family of symmetric stable distributions. The reason is twofold: First, a linear combination of stable random variables is stable. Second, stable distributions appear as limits in central limit theorems (see stable distributions in Section 3.2.4). Stable distributions are characterized by a parameter α , $0 < \alpha \leq 2$. They are Cauchy when $\alpha = 1$ and Gaussian when $\alpha = 2$. However, they have finite variance only when $\alpha = 2$.

In practice, the type of data under consideration governs the variability of the modeling distribution. Random signals restricted to a certain interval, such as the phase of complex random signals, are well suited for uniform distributions. On the other hand, signals allowing for any possible value but generally confined to a region are better suited for Gaussian models. However, if a process contains spikes and therefore has high variability, it is best characterized by a long-tailed distribution such as the Cauchy distribution. Impulsive signals have been found in a variety of applications, such as communication channels, radar signals, and electronic circuit noise. In all cases, the variability of the process dictates the appropriate model.

Discrete-time stationary random processes have correlation sequences that are functions of a single index. This leads to nice and powerful representations in both the frequency and the z -transform domains.

Power spectral density

The *power spectral density* (PSD, or more appropriately autoPSD) of a stationary stochastic process $x(n)$ is a Fourier transformation of its autocorrelation sequence $r_x(l)$. If $r_x(l)$ is periodic (which corresponds to a wide-sense periodic stochastic process) in l , then the DTFS discussed in Section 2.2.1 can be used to obtain the PSD, which has the form of a *line spectrum*. If $r_x(l)$ is nonperiodic, the DTFT discussed in Section 2.2.1 can be used provided that $r_x(l)$ is absolutely summable. This means that the process $x(n)$ must be a zero-mean process. In general, a stochastic process can be a mixture of periodic and nonperiodic components.[†]

If we allow impulse functions in the DTFT to represent periodic (or almost periodic) sequences and non-zero-mean processes (see Section 2.2.1), then we can define the PSD as

$$R_x(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r_x(l)e^{-j\omega l} \quad (3.3.39)$$

where ω is the frequency in radians per sample. If the process $x(n)$ is a zero-mean nonperiodic process, then (3.3.39) is enough to determine the PSD. If $x(n)$ is periodic (including nonzero mean) or almost periodic, then the PSD is given by

$$R_x(e^{j\omega}) = \sum_i 2\pi A_i \delta(\omega - \omega_i) \quad (3.3.40)$$

where the A_i are amplitudes of $r_x(l)$ at frequencies ω_i . For discussion purposes we will assume that $x(n)$ is a zero-mean nonperiodic process. The autocorrelation $r_x(l)$ can be recovered from the PSD by using the inverse DTFT as

$$r_x(l) = \frac{1}{2\pi} \int_{-\pi}^{\pi} R_x(e^{j\omega}) e^{j\omega l} d\omega \quad (3.3.41)$$

EXAMPLE 3.3.4. Determine the PSD of a zero-mean WSS process $x(n)$ with $r_x(l) = a^{|l|}$, $-1 < a < 1$.

Solution. From (3.3.39) we have

$$\begin{aligned} R_x(e^{j\omega}) &= \sum_{l=-\infty}^{\infty} a^{|l|} e^{-j\omega l} \quad -1 < a < 1 \\ &= \frac{1}{1 - ae^{j\omega}} + \frac{1}{1 - ae^{-j\omega}} - 1 \\ &= \frac{1 - a^2}{1 + a^2 - 2a \cos \omega} \quad -1 < a < 1 \end{aligned} \quad (3.3.42)$$

which is a real-valued, even, and nonnegative function of ω .

Properties of the autoPSD. The power spectral density $R_x(e^{j\omega})$ has three key properties that follow from corresponding properties of the autocorrelation sequence and the DTFT.

[†]Periodic components are predictable processes as discussed before. However, some nonperiodic components can also be predictable. Hence nonperiodic components are not always regular processes.

PROPERTY 3.3.4. The autoPSD $R_x(e^{j\omega})$ is a real-valued periodic function of frequency with period 2π for any (real- or complex-valued) process $x(n)$. If $x(n)$ is real-valued, then $R_x(e^{j\omega})$ is also an even function of ω , that is,

$$R_x(e^{j\omega}) = R_x(e^{-j\omega}) \quad (3.3.43)$$

Proof. It follows from autocorrelation and DTFT properties.

PROPERTY 3.3.5. The autoPSD is nonnegative definite, that is,

$$R_x(e^{j\omega}) \geq 0 \quad (3.3.44)$$

Proof. This follows from the nonnegative definiteness of the autocorrelation sequence [see also discussions leading to (3.4.27)].

PROPERTY 3.3.6. The area under $R_x(e^{j\omega})$ is *nonnegative* and it equals the average power of $x(n)$. Indeed, from (3.3.41) it follows with $l = 0$ that

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} R_x(e^{j\omega}) d\omega = r_x(0) = E\{|x(n)|^2\} \geq 0 \quad (3.3.45)$$

Proof. It follows from Property 3.3.5.

White noise. A random sequence $w(n)$ is called a (second-order) *white noise process* with mean μ_w and variance σ_w^2 , denoted by

$$w(n) \sim \text{WN}(\mu_w, \sigma_w^2) \quad (3.3.46)$$

if and only if $E\{w(n)\} = \mu_w$ and

$$r_w(l) = E\{w(n)w^*(n-l)\} = \sigma_w^2 \delta(l) \quad (3.3.47)$$

which implies that $R_w(e^{j\omega}) = \sigma_w^2 \quad -\pi \leq \omega \leq \pi$ (3.3.48)

The term *white noise* is used to emphasize that all frequencies contribute the same amount of power, as in the case of white light, which is obtained by mixing all possible colors by the same amount. If, in addition, the pdf of $x(n)$ is Gaussian, then the process is called a (second-order) *white Gaussian noise process*, and it will be denoted by $\text{WGN}(\mu_w, \sigma_w^2)$.

If the random variables $w(n)$ are independently and identically distributed with mean μ_w and variance σ_w^2 , then we shall write

$$w(n) \sim \text{IID}(\mu_w, \sigma_w^2) \quad (3.3.49)$$

This is sometimes referred to as a *strict* white noise.

We emphasize that the conditions of uncorrelatedness or independence do not put any restriction on the form of the probability density function of $w(n)$. Thus we can have an IID process with any type of probability distribution. Clearly, white noise is the simplest random process because it does not have any structure. However, we will see that it can be used as the basic building block for the construction of processes with more complicated dependence or correlation structures.

Harmonic processes. A *harmonic process* is defined by

$$x(n) = \sum_{k=1}^M A_k \cos(\omega_k n + \phi_k) \quad \omega_k \neq 0 \quad (3.3.50)$$

where M , $\{A_k\}_1^M$, and $\{\omega_k\}_1^M$ are constants and $\{\phi_k\}_1^M$ are pairwise independent random variables uniformly distributed in the interval $[0, 2\pi]$. It can be shown (see Problem 3.9) that $x(n)$ is a stationary process with mean

$$E\{x(n)\} = 0 \quad \text{for all } n \quad (3.3.51)$$

$$r_x(l) = \frac{1}{2} \sum_{k=1}^M A_k^2 \cos \omega_k l \quad -\infty < l < \infty \quad (3.3.52)$$

We note that $r_x(l)$ consists of a sum of “in-phase” cosines with the same frequencies as in $x(n)$.

If $\omega_k/(2\pi)$ are rational numbers, $r_x(l)$ is periodic and can be expanded as a Fourier series. These series coefficients provide the power spectrum $R_x(k)$ of $x(n)$. However, because $r_x(l)$ is a linear superposition of cosines, it always has a line spectrum with $2M$ lines of strength $A_k^2/4$ at frequencies $\pm\omega_k$ in the interval $[-\pi, \pi]$. If $r_x(l)$ is periodic, then the lines are equidistant (i.e., harmonically related), hence the name *harmonic process*. If $\omega/(2\pi)$ is irrational, then $r_x(l)$ is almost periodic and can be treated in the frequency domain in almost the same fashion. Hence the power spectrum of a harmonic process is given by

$$R_x(e^{j\omega}) = \sum_{k=-M}^M 2\pi \left(\frac{A_k^2}{4} \right) \delta(\omega - \omega_k) = \sum_{k=-M}^M \frac{\pi}{2} A_k^2 \delta(\omega - \omega_k), \quad -\pi < \omega \leq \pi \quad (3.3.53)$$

EXAMPLE 3.3.5. Consider the following harmonic process

$$x(n) = \cos(0.1\pi n + \phi_1) + 2 \sin(1.5n + \phi_2)$$

where ϕ_1 and ϕ_2 are IID random variables uniformly distributed in the interval $[0, 2\pi]$. The first component of $x(n)$ is periodic with $\omega_1 = 0.1\pi$ and period equal to 20 while the second component is almost periodic with $\omega_2 = 1.5$. Thus the sequence $x(n)$ is almost periodic. A sample function realization of $x(n)$ is shown in Figure 3.9(a). The mean of $x(n)$ is

$$\mu_x(n) = E\{x(n)\} = E\{\cos(0.1\pi n + \phi_1) + 2 \sin(1.5n + \phi_2)\} = 0$$

and the autocorrelation sequence (using mutual independence between ϕ_1 and ϕ_2) is

$$\begin{aligned} r_x(n_1, n_2) &= E\{x(n_1)x_2^*(n_2)\} \\ &= E\{\cos(0.1\pi n_1 + \phi_1)\cos(0.1\pi n_2 + \phi_1)\} \\ &\quad + E\{2 \sin(1.5n_1 + \phi_2)2 \sin(1.5n_2 + \phi_2)\} \\ &= \frac{1}{2} \cos[0.1\pi(n_1 - n_2)] + 2 \cos[1.5(n_1 - n_2)] \end{aligned}$$

or

$$r_x(l) = \frac{1}{2} \cos 0.1\pi l + 2 \cos 1.5l \quad l = n_1 - n_2$$

Thus the line spectrum $R_{\omega_k}^{(x)}$ is given by

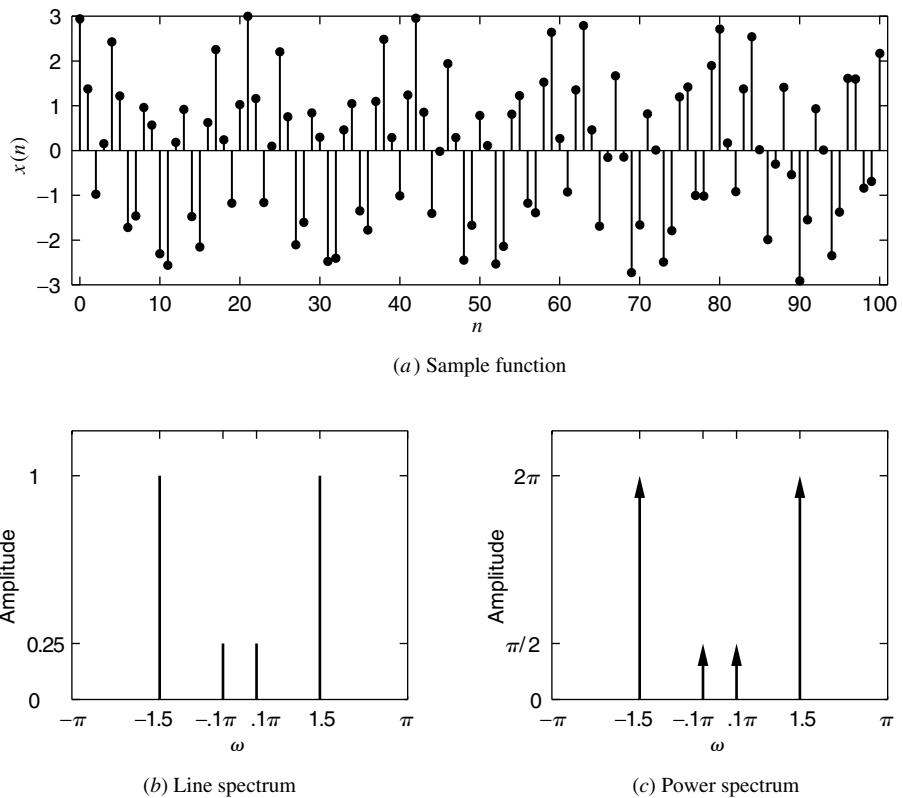
$$R_{\omega_k}^{(x)} = \begin{cases} 1 & \omega_1 = -1.5 \\ \frac{1}{4} & \omega_2 = -0.1\pi \\ \frac{1}{4} & \omega_3 = 0.1\pi \\ 1 & \omega_4 = 1.5 \end{cases}$$

and the power spectrum $R_x(e^{j\omega})$ is given by

$$R_x(e^{j\omega}) = 2\pi\delta(\omega + 1.5) + \frac{\pi}{2}\delta(\omega + 0.1\pi) + \frac{\pi}{2}\delta(\omega - 0.1\pi) + 2\pi\delta(\omega - 1.5)$$

The line spectrum of $x(n)$ is shown in Figure 3.9(b) and the corresponding power spectrum in Figure 3.9(c).

The harmonic process is predictable because any given realization is a sinusoidal sequence with fixed amplitude, frequency, and phase. We stress that the independence of the phases is required to guarantee the stationarity of $x(n)$ in (3.3.50). The uniform distribution of the phases is necessary to make $x(n)$ a stationary process (see Problem 3.9). The harmonic process (3.3.50), in general, is non-Gaussian; however, it becomes Gaussian if the amplitudes A_k are random variables with a Rayleigh distribution (Porat 1994).

**FIGURE 3.9**

The time and frequency-domain description of the harmonic process in Example 3.3.5.

EXAMPLE 3.3.6. Consider a complex-valued process given by

$$x(n) = Ae^{j\omega_0 n} = |A|e^{j(\omega_0 n + \phi)}$$

where A is a complex-valued random variable and ω_0 is constant. The mean of $x(n)$

$$E\{x(n)\} = E\{A\}e^{j\omega_0 n}$$

can be constant only if $E\{A\} = 0$. If $|A|$ is constant and ϕ is uniformly distributed on $[0, 2\pi]$, then we have $E\{A\} = |A|E\{e^{j\phi}\} = 0$. In this case the autocorrelation is

$$r_x(n_1, n_2) = E\{Ae^{j(\omega_0 n_1 + \phi)} A^* e^{-j(\omega_0 n_2 + \phi)}\} = |A|^2 e^{j(n_1 - n_2)\omega_0}$$

Since the mean is constant and the autocorrelation depends on the difference $l \triangleq n_1 - n_2$, the process is wide-sense stationary.

The above example can be generalized to harmonic processes of the form

$$x(n) = \sum_{k=1}^M A_k e^{j(\omega_k n + \phi_k)} \quad (3.3.54)$$

where M , $\{A_k\}_1^M$, and $\{\omega_k\}_1^M$ are constants and $\{\phi_k\}_1^M$ are pairwise independent random variables uniformly distributed in the interval $[0, 2\pi]$. The autocorrelation sequence is

$$r_x(l) = \sum_{k=1}^M |A_k|^2 e^{j\omega_k l} \quad (3.3.55)$$

and the power spectrum consists of M impulses with amplitudes $2\pi|A_k|^2$ at frequencies ω_k . If the amplitudes $\{A_k\}_{k=1}^M$ are random variables, mutually independent of the random phases, the quantity $|A_k|^2$ is replaced by $E\{|A_k|^2\}$.

Cross-power spectral density

The cross-power spectral density of two zero-mean and jointly stationary stochastic processes provides a description of their statistical relations in the frequency domain and is defined as the DTFT of their cross-correlation, that is,

$$R_{xy}(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r_{xy}(l)e^{-j\omega l} \quad (3.3.56)$$

The cross-correlation $r_{xy}(l)$ can be recovered by the inverse DTFT

$$r_{xy}(l) = \frac{1}{2\pi} \int_{-\pi}^{\pi} R_{xy}(e^{j\omega})e^{j\omega l} d\omega \quad (3.3.57)$$

The cross-spectrum $R_{xy}(e^{j\omega})$ is, in general, a complex function of ω . From $r_{xy}(l) = r_{yx}^*(-l)$ it follows that

$$R_{xy}(e^{j\omega}) = R_{yx}^*(e^{j\omega}) \quad (3.3.58)$$

This implies that $R_{xy}(e^{j\omega})$ and $R_{yx}(e^{j\omega})$ have the same magnitude but opposite phase.

The normalized cross-spectrum

$$\mathcal{G}_{xy}(e^{j\omega}) \triangleq \frac{R_{xy}(e^{j\omega})}{\sqrt{R_x(e^{j\omega})}\sqrt{R_y(e^{j\omega})}} \quad (3.3.59)$$

is called the *coherence function*. Its squared magnitude

$$|\mathcal{G}_{xy}(e^{j\omega})|^2 = \frac{|R_{xy}(e^{j\omega})|^2}{R_x(e^{j\omega})R_y(e^{j\omega})} \quad (3.3.60)$$

is known as the *magnitude square coherence (MSC)* and can be thought of as a sort of correlation coefficient in the frequency domain. If $x(n) = y(n)$, then $\mathcal{G}_{xy}(e^{j\omega}) = 1$ (maximum correlation) whereas if $x(n)$ and $y(n)$ are uncorrelated, then $R_{xy}(l) = 0$ and hence $\mathcal{G}_{xy}(e^{j\omega}) = 0$. In other words, $0 \leq |\mathcal{G}_{xy}(e^{j\omega})| \leq 1$.

Complex spectral density functions

If the sequences $r_x(l)$ and $r_{xy}(l)$ are absolutely summable within a certain ring of the complex z plane, we can obtain their z -transforms

$$R_x(z) = \sum_{l=-\infty}^{\infty} r_x(l)z^{-l} \quad (3.3.61)$$

$$R_{xy}(z) = \sum_{l=-\infty}^{\infty} r_{xy}(l)z^{-l} \quad (3.3.62)$$

which are known as the *complex spectral density* and *complex cross-spectral density* functions, respectively. If the unit circle, defined by $z = e^{j\omega}$, is within the region of convergence of the above summations, then

$$R_x(e^{j\omega}) = R_x(z)|_{z=e^{j\omega}} \quad (3.3.63)$$

$$R_{xy}(e^{j\omega}) = R_{xy}(z)|_{z=e^{j\omega}} \quad (3.3.64)$$

The correlation and power spectral density properties of random sequences are summarized in Table 3.1.

EXAMPLE 3.3.7. Consider the random sequence given in Example 3.3.4 with autoPSD in (3.3.42)

$$R_x(e^{j\omega}) = \frac{1 - a^2}{1 + a^2 - 2a \cos \omega} \quad |a| < 1$$

Determine the complex autoPSD $R_x(z)$.

Solution. The complex autoPSD is given by $R_x(z) = R_x(e^{j\omega})|_{e^{j\omega}=z}$. Since

$$\cos \omega = \frac{e^{j\omega} + e^{-j\omega}}{2} = \frac{z + z^{-1}}{2} \Big|_{z=e^{j\omega}}$$

we obtain

$$R_x(z) = \frac{1 - a^2}{1 + a^2 - 2a \left(\frac{z + z^{-1}}{2} \right)} = \frac{(a - a^{-1})z^{-1}}{1 - (a + a^{-1})z^{-1} + z^{-2}} \quad |a| < |z| < \frac{1}{|a|}$$

Now the inverse z -transform of $R_x(z)$ determines the autocorrelation sequence $r_x(l)$, that is,

$$\begin{aligned} R_x(z) &= \frac{(a - a^{-1})z^{-1}}{1 - (a + a^{-1})z^{-1} + z^{-2}} = \frac{(a - a^{-1})z^{-1}}{(1 - az^{-1})(1 - a^{-1}z^{-1})} \\ &= \frac{1}{(1 - az^{-1})} - \frac{1}{(1 - a^{-1}z^{-1})} \quad |a| < |z| < |a|^{-1} \\ \text{or} \quad r_x(l) &= a^l u(l) + (a^{-1})^l u(-l - 1) = a^{|l|} \end{aligned} \quad (3.3.65)$$

This approach can be used to determine autocorrelation sequences from autoPSD functions.

Table 3.1 provides a summary of correlation and spectral properties of stationary random sequences.

TABLE 3.1
Summary of correlation and spectral properties of stationary random sequences.

Definitions	
Mean value	$\mu_x = E\{x(n)\}$
Autocorrelation	$r_x(l) = E\{\{x(n)x^*(n-l)\}$
Autocovariance	$\gamma_x(l) = E\{\{[x(n) - \mu_x][x(n-l) - \mu_x]\}^*\}$
Cross-correlation	$r_{xy}(l) = E\{x(n)y^*(n-l)\}$
Cross-covariance	$\gamma_{xy}(l) = E\{\{[x(n) - \mu_x][y(n-l) - \mu_y]\}^*\}$
Power spectral density	$R_x(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r_x(l)e^{-j\omega l}$
Cross-power spectral density	$R_{xy}(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r_{xy}(l)e^{-j\omega l}$
Magnitude square coherence	$ \mathcal{G}_{xy}(e^{j\omega}) ^2 = R_{xy}(e^{j\omega}) ^2 / [R_x(e^{j\omega})R_y(e^{j\omega})]$

Interrelations	
$\gamma_x(l) = r_x(l) - \mu_x ^2$	
$\gamma_{xy}(l) = r_{xy}(l) - \mu_x \mu_y^*$	

Properties	
Autocorrelation	Auto-PSD
$r_x(l)$ is nonnegative definite	$R_x(e^{j\omega}) \geq 0$ and real
$r_x(l) = r_x^*(-l)$	$R_x(e^{j\omega}) = R_x(e^{-j\omega})$ [real $x(n)$]
$ r_x(l) \leq r_x(0)$	$R_x(z) = R_x^*(1/z^*)$
$ \rho_x(l) \leq 1$	$R_x(z) = R_x(z^{-1})$ [real $x(n)$]
Cross-correlation	Cross-PSD
$r_{xy}(l) = r_{yx}^*(-l)$	
$ r_{xy}(l) \leq [r_x(0)r_y(0)]^{1/2} \leq \frac{1}{2}[r_x(0) + r_y(0)]$	$R_{xy}(z) = R_{yx}^*(1/z^*)$ $0 \leq \mathcal{G}_{xy}(e^{j\omega}) \leq 1$
$ \rho_{xy}(l) \leq 1$	

This section deals with the processing of stationary random sequences using linear, time-invariant (LTI) systems. We focus on expressing the second-order statistical properties of the output in terms of the corresponding properties of the input and the characteristics of the system.

3.4.1 Time-Domain Analysis

The first question to ask when we apply a random signal to a system is, Just what is the meaning of such an operation? We ask this because a random process is not just a single sequence but an ensemble of sequences (see Section 3.3). However, since each realization of the stochastic process is a deterministic signal, it is an acceptable input producing an output that is clearly a single realization of the output stochastic process. For an LTI system, each pair of input-output realizations is described by the convolution summation

$$y(n, \zeta) = \sum_{k=-\infty}^{\infty} h(k)x(n-k, \zeta) \quad (3.4.1)$$

If the sum in the right side of (3.4.1) exists for all ζ such that $\Pr\{\zeta\} = 1$, then we say that we have almost-everywhere convergence or convergence with probability 1 (Papoulis 1991). The existence of such convergence is ruled by the following theorem (Brockwell and Davis 1991).

THEOREM 3.2. If the process $x(n, \zeta)$ is stationary with $E\{|x(n, \zeta)|\} < \infty$ and if the system is BIBO-stable, that is, $\sum_{-\infty}^{\infty} |h(k)| < \infty$, then the output $y(n, \zeta)$ of the system in (3.4.1) converges absolutely with probability 1, or

$$y(n, \zeta) = \sum_{k=-\infty}^{\infty} h(k)x(n-k, \zeta) \quad \text{for all } \zeta \in \mathcal{A}, \Pr\{\mathcal{A}\} = 1 \quad (3.4.2)$$

and is stationary. Furthermore, if $E\{|x(n, \zeta)|^2\} < \infty$, then $E\{|y(n, \zeta)|^2\} < \infty$ and $y(n, \zeta)$ converges in the mean square to the same limit and is stationary.

A less restrictive condition of finite energy on the system impulse response $h(n)$ also guarantees the mean square existence of the output process, as stated in the following theorem.

THEOREM 3.3. If the process $x(n, \zeta)$ is zero-mean and stationary with $\sum_{l=-\infty}^{\infty} |r_x(l)| < \infty$, and if the system (3.4.1) satisfies the condition

$$\sum_{n=-\infty}^{\infty} |h(n)|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 d\omega < \infty \quad (3.4.3)$$

then the output $y(n, \zeta)$ converges in the mean square sense and is stationary.

The above two theorems are applicable when input processes have finite variances. However, IID sequences with α -stable distributions have infinite variances. If the impulse response of the system in (3.4.1) decays fast enough, then the following theorem (Brockwell and Davis 1991) guarantees the absolute convergence of $y(n, \zeta)$ with probability 1. These issues are of particular importance for inputs with high variability and are discussed in Section 3.3.5.

THEOREM 3.4. Let $x(n, \zeta)$ be an IID sequence of random variables with α -stable distribution, $0 < \alpha < 2$. If the impulse response $h(n)$ satisfies

$$\sum_{n=-\infty}^{\infty} |h(n)|^{\delta} < \infty \quad \text{for some } \delta \in (0, \alpha)$$

then the output $y(n, \zeta)$ in (3.4.1) converges absolutely with probability 1.

Clearly, a complete description of the output stochastic process $y(n)$ requires the computation of an infinite number of convolutions. Thus, a better alternative would be to determine the statistical properties of $y(n)$ in terms of the statistical properties of the input and the characteristics of the system. For Gaussian signals, which are used very often in practice, first- and second-order statistics are sufficient.

Output mean value. If $x(n)$ is stationary, its first-order statistic is determined by its mean value μ_x . To determine the mean value of the output, we take the expected value of both sides of (3.4.1):

$$\mu_y = \sum_{k=-\infty}^{\infty} h(k)E\{x(n-k)\} = \mu_x \sum_{k=-\infty}^{\infty} h(k) = \mu_x H(e^{j0}) \quad (3.4.4)$$

Since μ_x and $H(e^{j0})$ are constant, μ_y is also constant. Note that $H(e^{j0})$ is the dc gain of the spectrum.

Input-output cross-correlation. If we take complex conjugate of (3.4.1), premultiply it by $x(n+l)$, and take the expectation of both sides, we have

$$E\{x(n+l)y^*(l)\} = \sum_{k=-\infty}^{\infty} h^*(k)E\{x(n+l)x^*(n-k)\}$$

or $r_{xy}(l) = \sum_{k=-\infty}^{\infty} h^*(k)r_{xx}(l+k) = \sum_{m=-\infty}^{\infty} h^*(-m)r_{xx}(l-m)$

Hence, $r_{xy}(l) = h^*(-l) * r_{xx}(l) \quad (3.4.5)$

Similarly, $r_{yx}(l) = h(l) * r_{xx}(l) \quad (3.4.6)$

Output autocorrelation. Postmultiplying both sides of (3.4.1) by $y^*(n-l)$ and taking the expectation, we obtain

$$E\{y(n)y^*(n-l)\} = \sum_{k=-\infty}^{\infty} h(k)E\{x(n-k)y^*(n-l)\} \quad (3.4.7)$$

or $r_{yy}(l) = \sum_{k=-\infty}^{\infty} h(k)r_{xy}(l-k) = h(l) * r_{xy}(l) \quad (3.4.8)$

From (3.4.5) and (3.4.8) we get

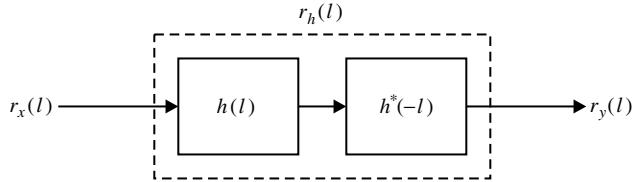
$$r_y(l) = h(l) * h^*(-l) * r_x(l) \quad (3.4.9)$$

or $r_y(l) = r_h(l) * r_x(l) \quad (3.4.10)$

where $r_h(l) \triangleq h(l) * h^*(-l) = \sum_{n=-\infty}^{\infty} h(n)h^*(n-l) \quad (3.4.11)$

is the autocorrelation of the impulse response and is called the system correlation sequence.

Since μ_y is constant and $r_y(l)$ depends only on the lag l , the response of a stable system to a stationary input is also a stationary process. A careful examination of (3.4.10) shows that *when a signal $x(n)$ is filtered by an LTI system with impulse response $h(n)$ its autocorrelation is “filtered” by a system with impulse response equal to the autocorrelation of its impulse response*, as shown in Figure 3.10.

**FIGURE 3.10**

An equivalent LTI system for autocorrelation filtration.

Output power. The power $E\{|y(n)|^2\}$ of the output process $y(n)$ is equal to $r_y(0)$, which from (3.4.9) and (3.4.10) and the symmetry property of $r_x(l)$ is

$$\begin{aligned} P_y &= r_y(0) = r_h(l) * r_x(l)|_{l=0} \\ &= \sum_{k=-\infty}^{\infty} r_h(k)r_x(-k) = \sum_{k=-\infty}^{\infty} [h(k) * h^*(-k)]r_x(k) \\ &= \sum_{k=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} h(m)h^*(m-k)r_x(k) \end{aligned} \quad (3.4.12)$$

$$= \sum_{k=-\infty}^{\infty} r_h(k)r_x(k) \quad (3.4.13)$$

or for FIR filters with $\mathbf{h} = [h(0) \ h(1) \ \cdots \ h(M-1)]^T$, (3.4.12) can be written as

$$P_y = \mathbf{h}^H \mathbf{R}_x \mathbf{h} \quad (3.4.14)$$

Finally, we note that when $\mu_x = 0$, we have $\mu_y = 0$ and $\sigma_y^2 = P_y$.

Output probability density function. Finding the probability density of the output of an LTI system is very difficult, except in some special cases. Thus, if $x(n)$ is a Gaussian process, then the output is also a Gaussian process with mean and autocorrelation given by (3.4.4) and (3.4.10). Also if $x(n)$ is IID, the probability density of the output is obtained by noting that $y(n)$ is a weighted sum of independent random variables. Indeed, the probability density of the sum of independent random variables is the convolution of their probability densities or the products of their characteristic functions. Thus if the input process is an IID stable process then the output process is also stable whose probability density can be computed by using characteristic functions.

3.4.2 Frequency-Domain Analysis

To obtain the output autoPSD and complex autoPSD, we recall that if $H(z) = \mathcal{Z}\{h(n)\}$, then, for real $h(n)$,

$$\mathcal{Z}\{h^*(-n)\} = H^*\left(\frac{1}{z^*}\right) \quad (3.4.15)$$

From (3.4.5), (3.4.6), and (3.4.9) we obtain

$$R_{xy}(z) = H^*\left(\frac{1}{z^*}\right) R_x(z) \quad (3.4.16)$$

$$R_{yx}(z) = H(z) R_x(z) \quad (3.4.17)$$

and

$$R_y(z) = H(z) H^*\left(\frac{1}{z^*}\right) R_x(z) \quad (3.4.18)$$

For a stable system, the unit circle $z = e^{j\omega}$ lies within the ROCs of $H(z)$ and $H(z^{-1})$. Thus,

$$R_{xy}(e^{j\omega}) = H^*(e^{j\omega})R_x(e^{j\omega}) \quad (3.4.19)$$

$$R_{yx}(e^{j\omega}) = H(e^{j\omega})R_x(e^{j\omega}) \quad (3.4.20)$$

and

$$R_y(e^{j\omega}) = H(e^{j\omega})H^*(e^{j\omega})R_x(e^{j\omega}) \quad (3.4.21)$$

or

$$R_y(e^{j\omega}) = |H(e^{j\omega})|^2 R_x(e^{j\omega}) \quad (3.4.22)$$

Thus, if we know the input and output autocorrelations or autospectral densities, we can determine the magnitude response of a system, but not its phase response. Only cross-correlation or cross-spectral densities can provide phase information [see (3.4.19) and (3.4.20)].

It can easily be shown that the power of the output is

$$E\{|y(n)|^2\} = r_{yy}(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 R_x(e^{j\omega}) d\omega \quad (3.4.23)$$

$$= \sum_{l=-\infty}^{\infty} r_x(l)r_h(l) \quad (3.4.24)$$

which is equivalent to (3.4.13).

Consider now a narrowband filter with frequency response

$$H(e^{j\omega}) = \begin{cases} 1 & \omega_c - \frac{\Delta\omega}{2} \leq \omega \leq \omega_c + \frac{\Delta\omega}{2} \\ 0 & \text{elsewhere} \end{cases} \quad (3.4.25)$$

The power of the filter output is

$$E\{|y(n)|^2\} = \frac{1}{2\pi} \int_{\omega_c - \Delta\omega/2}^{\omega_c + \Delta\omega/2} R_x(e^{j\omega}) d\omega \simeq \frac{\Delta\omega}{\pi} R_x(e^{j\omega_c}) \quad (3.4.26)$$

assuming that $\Delta\omega$ is sufficiently small and that $R_x(e^{j\omega})$ is continuous at $\omega = \omega_c$. Since $E\{|y(n)|^2\} \geq 0$, $R_x(e^{j\omega_c})$ is also nonnegative for all ω_c and $\Delta\omega$, hence

$$R_x(e^{j\omega}) \geq 0 \quad -\pi \leq \omega \leq \pi \quad (3.4.27)$$

Hence, the PSD $R_x(e^{j\omega})$ is nonnegative definite for any random sequence $x(n)$ real (or complex). Furthermore, $R_x(e^{j\omega}) d\omega/(2\pi)$, has the interpretation of power, or $R_x(e^{j\omega})$ is a power density as a function of frequency (in radians per sample). Table 3.2 shows various input-output relationships in both the time and frequency domains.

TABLE 3.2
Second-order moments of stationary random sequences processed by linear, time-invariant systems.

Time domain	Frequency domain	z Domain
$y(n) = h(n) * x(n)$	Not available	Not available
$r_{yx}(l) = h(l) * r_x(l)$	$R_{yx}(e^{j\omega}) = H(e^{j\omega})R_x(e^{j\omega})$	$R_{yx}(z) = H(z)R_x(z)$
$r_{xy}(l) = h^*(-l) * r_x(l)$	$R_{xy}(e^{j\omega}) = H^*(e^{j\omega})R_x(e^{j\omega})$	$R_{xy}(z) = H^*(1/z^*)R_x(z)$
$r_y(l) = h(l) * r_{xy}(l)$	$R_y(e^{j\omega}) = H(e^{j\omega})R_{xy}(e^{j\omega})$	$R_y(z) = H(z)R_{xy}(z)$
$r_y(l) = h(l) * h^*(-l) * r_x(l)$	$R_y(e^{j\omega}) = H(e^{j\omega}) ^2 R_x(e^{j\omega})$	$R_y(z) = H(z)H^*(1/z^*)R_x(z)$

3.4.3 Random Signal Memory

Given the “zero-memory” process $w(n) \sim \text{IID}(0, \sigma_w^2)$, we can introduce dependence by passing it through an LTI system. The extent and degree of the imposed dependence are dictated by the shape of the system’s impulse response. The probability density of $w(n)$ is

not explicitly involved. Suppose now that we are given the resulting linear process $x(n)$, and we want to quantify its memory. For processes with finite variance we can use the *correlation length*

$$L_c = \frac{1}{r_x(0)} \sum_{l=0}^{\infty} r_x(l) = \sum_{l=0}^{\infty} \rho_x(l)$$

which equals the area under the normalized autocorrelation sequence curve and shows the maximum distance at which two samples are significantly correlated.

An IID process has no memory and is completely described by its first-order density. A linear process has memory introduced by the impulse response of the generating system. If $w(n)$ has finite variance, the memory of the process is determined by the autocorrelation of the impulse response because $r_x(l) = \sigma_w^2 r_h(l)$. Also, the higher-order densities of the process are nonzero. Thus, the variability of the output—that is, what amplitudes the signal takes, how often, and how fast the amplitude changes from sample to sample—is the combined effect of the input probability density and the system memory.

DEFINITION 3.9. A stationary process $x(n)$ with finite variance is said to have *long memory* if there exist constants α , $0 < \alpha < 1$, and $C_r > 0$ such that

$$\lim_{l \rightarrow \infty} \frac{1}{C_r \sigma_x^2} r_x(l) l^\alpha = 1$$

This implies that the autocorrelation has fat or heavy tails, that is, asymptotically decays as a power law

$$\rho_x(l) \simeq C_r |l|^{-\alpha} \quad \text{as } l \rightarrow \infty$$

and slowly enough that

$$\sum_{l=-\infty}^{\infty} \rho_x(l) = \infty$$

that is, a long-memory process has infinite correlation length. If

$$\sum_{l=-\infty}^{\infty} \rho_x(l) < \infty$$

we say that the process has *short memory*. This is the case for autocorrelations that decay exponentially, for example, $\rho_x(l) = a^{|l|}$, $-1 < a < 1$.

An equivalent definition of long memory can be formulated in terms of the power spectrum (Beran 1994; Samorodnitsky and Taqqu 1994).

DEFINITION 3.10. A stationary process $x(n)$ with finite variance is said to have *long memory* if there exist constants β , $0 < \beta < 1$, and $C_R > 0$ such that

$$\lim_{\omega \rightarrow 0} \frac{1}{C_R \sigma_x^2} R_x(e^{j\omega}) |\omega|^\beta = 1$$

This asymptotic definition implies that

$$R_x(e^{j\omega}) \simeq \frac{C_R \sigma_x^2}{|\omega|^\beta} \quad \text{as } \omega \rightarrow 0$$

and

$$R_x(0) = \sum_{l=-\infty}^{\infty} r_x(l) = \infty$$

The first-order density determines the mean value and the variance of a process, whereas the second-order density determines the autocorrelation and power spectrum. There is a coupling between the probability density and the autocorrelation or power spectrum of a

process. However, this coupling is not extremely strong because there are processes that have different densities and the same autocorrelation. Thus, we can have random signal models with short or long memory and low or high variability. Random signal models are discussed in Chapters 4 and 12.

3.4.4 General Correlation Matrices

We first begin with the properties of general correlation matrices. Similar properties apply to covariance matrices.

PROPERTY 3.4.1. The correlation matrix of a random vector \mathbf{x} is conjugate symmetric or Hermitian, that is,

$$\mathbf{R}_x = \mathbf{R}_x^H \quad (3.4.28)$$

Proof. This follows easily from (3.2.19).

PROPERTY 3.4.2. The correlation matrix of a random vector \mathbf{x} is nonnegative definite (n.n.d.); or for every nonzero complex vector $\mathbf{w} = [w_1 \ w_2 \ \cdots \ w_M]^T$, the quadratic form $\mathbf{w}^H \mathbf{R}_x \mathbf{w}$ is nonnegative, that is,

$$\mathbf{w}^H \mathbf{R}_x \mathbf{w} \geq 0 \quad (3.4.29)$$

Proof. To prove (3.4.29), we define the dot product

$$\alpha = \mathbf{w}^H \mathbf{x} = \mathbf{x}^T \mathbf{w}^* = \sum_{k=1}^M w_k^* x_k \quad (3.4.30)$$

The mean square value of the random variable α is

$$E\{|\alpha|^2\} = E\{\mathbf{w}^H \mathbf{x} \mathbf{x}^H \mathbf{w}\} = \mathbf{w}^H E\{\mathbf{x} \mathbf{x}^H\} \mathbf{w} = \mathbf{w}^H \mathbf{R}_x \mathbf{w} \quad (3.4.31)$$

Since $E\{|\alpha|^2\} \geq 0$, it follows that $\mathbf{w}^H \mathbf{R}_x \mathbf{w} \geq 0$. We also note that a matrix is called *positive definite* (p.d.) if $\mathbf{w}^H \mathbf{R}_x \mathbf{w} > 0$.

Eigenvalues and eigenvectors of \mathbf{R}

For a Hermitian matrix \mathbf{R} we wish to find an $M \times 1$ vector \mathbf{q} that satisfies the condition

$$\mathbf{R}\mathbf{q} = \lambda \mathbf{q} \quad (3.4.32)$$

where λ is a constant. This condition implies that the linear transformation performed by matrix \mathbf{R} does not change the direction of vector \mathbf{q} . Thus $\mathbf{R}\mathbf{q}$ is a *direction-invariant* mapping. To determine the vector \mathbf{q} , we write (3.4.32) as

$$(\mathbf{R} - \lambda \mathbf{I})\mathbf{q} = \mathbf{0} \quad (3.4.33)$$

where \mathbf{I} is the $M \times M$ identity matrix and $\mathbf{0}$ is an $M \times 1$ vector of zeros. Since \mathbf{q} is arbitrary, the only way (3.4.33) is satisfied is if the determinant of $\mathbf{R} - \lambda \mathbf{I}$ equals zero, that is,

$$\det(\mathbf{R} - \lambda \mathbf{I}) = 0 \quad (3.4.34)$$

This equation is an M th-order polynomial in λ and is called the *characteristic equation* of \mathbf{R} . It has M roots $\{\lambda_i\}_{i=1}^M$, called *eigenvalues*, which, in general, are distinct. If (3.4.34) has repeated roots, then \mathbf{R} is said to have *degenerate* eigenvalues. For each eigenvalue λ_i we can satisfy (3.4.32)

$$\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i \quad i = 1, \dots, M \quad (3.4.35)$$

where the \mathbf{q}_i are called *eigenvectors* of \mathbf{R} . Therefore, the $M \times M$ matrix \mathbf{R} has M eigenvectors. To uniquely determine \mathbf{q}_i , we use (3.4.35) along with the normality condition that $\|\mathbf{q}_i\| = 1$. A MATLAB function `[Lambda, Q] = eig(R)` is available to compute eigenvalues and eigenvectors of \mathbf{R} .

There are further properties of the autocorrelation matrix \mathbf{R} based on its eigenanalysis, which we describe below. Consider a matrix \mathbf{R} that is Hermitian and nonnegative definite ($\mathbf{w}^H \mathbf{R} \mathbf{w} \geq 0$) with eigenvalues $\{\lambda_i\}_{i=1}^M$ and eigenvectors $\{\mathbf{q}_i\}_{i=1}^M$.

PROPERTY 3.4.3. The matrix \mathbf{R}^k ($k = 1, 2, \dots$) has eigenvalues $\lambda_1^k, \lambda_2^k, \dots, \lambda_M^k$.

Proof. See Problem 3.16.

PROPERTY 3.4.4. If the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_M$ are distinct, the corresponding eigenvectors $\{\mathbf{q}_i\}_{i=1}^M$ are *linearly independent*.

Proof. This property can be proved by using Property 3.4.3. If there exists M not-all-zero scalars $\{\alpha_i\}_{i=1}^M$, such that

$$\sum_{i=1}^M \alpha_i \mathbf{q}_i = \mathbf{0} \quad (3.4.36)$$

then the eigenvectors $\{\mathbf{q}_i\}_{i=1}^M$ are said to be *linearly dependent*. Assume that (3.4.36) is true for some not-all-zero scalars $\{\alpha_i\}_{i=1}^M$ and that the eigenvalues $\{\lambda_i\}_{i=1}^M$ are distinct. Now multiply (3.4.36) repeatedly by \mathbf{R}^k , $k = 0, \dots, M-1$ and use Property 3.4.3 to obtain

$$\sum_{i=1}^M \alpha_i \mathbf{R}^k \mathbf{q}_i = \sum_{i=1}^M \alpha_i \lambda_i^k \mathbf{q}_i = \mathbf{0} \quad k = 0, \dots, M-1 \quad (3.4.37)$$

which can be arranged in a matrix format for $i = 1, \dots, M$ as

$$[\alpha_1 \mathbf{q}_1 \quad \alpha_2 \mathbf{q}_2 \quad \alpha_3 \mathbf{q}_3 \quad \dots \quad \alpha_M \mathbf{q}_M] \begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 & \dots & \lambda_1^{M-1} \\ 1 & \lambda_2 & \lambda_2^2 & \dots & \lambda_2^{M-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_M & \lambda_M^2 & \dots & \lambda_M^{M-1} \end{bmatrix} = \mathbf{0} \quad (3.4.38)$$

Since all the λ_i are distinct, the matrix containing the λ_i in (3.4.38) above is nonsingular. This matrix is called a *Vandermonde* matrix. Therefore, premultiplying both sides of (3.4.38) by the inverse of the Vandermonde matrix, we obtain

$$[\alpha_1 \mathbf{q}_1 \quad \alpha_2 \mathbf{q}_2 \quad \alpha_3 \mathbf{q}_3 \quad \dots \quad \alpha_M \mathbf{q}_M] = \mathbf{0} \quad (3.4.39)$$

Since eigenvectors $\{\mathbf{q}_i\}_{i=1}^M$ are not zero vectors, the only way (3.4.39) can be satisfied is if all $\{\alpha_i\}_{i=1}^M$ are zero. This implies that (3.4.36) cannot be satisfied for any set of not-all-zero scalars $\{\alpha_i\}_{i=1}^M$, which further implies that $\{\mathbf{q}_i\}_{i=1}^M$ are linearly independent.

PROPERTY 3.4.5. The eigenvalues $\{\lambda_i\}_{i=1}^M$ are real and *nonnegative*.

Proof. From (3.4.35), we have

$$\mathbf{q}_i^H \mathbf{R} \mathbf{q}_i = \lambda_i \mathbf{q}_i^H \mathbf{q}_i \quad i = 1, 2, \dots, M \quad (3.4.40)$$

Since \mathbf{R} is positive semidefinite, the quadratic form $\mathbf{q}_i^H \mathbf{R} \mathbf{q}_i \geq 0$. Also since $\mathbf{q}_i^H \mathbf{q}_i$ is an inner product, $\mathbf{q}_i^H \mathbf{q}_i > 0$. Hence

$$\lambda_i = \frac{\mathbf{q}_i^H \mathbf{R} \mathbf{q}_i}{\mathbf{q}_i^H \mathbf{q}_i} \geq 0 \quad i = 1, 2, \dots, M \quad (3.4.41)$$

Furthermore, if \mathbf{R} is positive definite, then $\lambda_i > 0$ for all $1 \leq i \leq M$. The quotient in (3.4.41) is a useful quantity and is known as the *Raleigh quotient* of vector \mathbf{q}_i .

PROPERTY 3.4.6. If the eigenvalues $\{\lambda_i\}_{i=1}^M$ are distinct, then the corresponding eigenvectors are orthogonal to one another, that is,

$$\lambda_i \neq \lambda_j \Rightarrow \mathbf{q}_i^H \mathbf{q}_j = 0 \quad \text{for } i \neq j \quad (3.4.42)$$

Proof. Consider (3.4.35). We have

$$\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i \quad (3.4.43)$$

$$\text{and} \quad \mathbf{R}\mathbf{q}_j = \lambda_j \mathbf{q}_j \quad (3.4.44)$$

for some $i \neq j$. Premultiplying both sides of (3.4.43) by \mathbf{q}_j^H , we obtain

$$\mathbf{q}_j^H \mathbf{R}\mathbf{q}_i = \mathbf{q}_j^H \lambda_i \mathbf{q}_i = \lambda_i \mathbf{q}_j^H \mathbf{q}_i \quad (3.4.45)$$

Taking the conjugate transpose of (3.4.44), using the Hermitian property (3.4.28) of \mathbf{R} , and using the realness Property 3.4.5 of eigenvalues, we get

$$\mathbf{q}_j^H \mathbf{R} = \lambda_j \mathbf{q}_j^H \quad (3.4.46)$$

Now postmultiplying (3.4.46) by \mathbf{q}_i and comparing with (3.4.45), we conclude that

$$\lambda_i \mathbf{q}_j^H \mathbf{q}_i = \lambda_j \mathbf{q}_j^H \mathbf{q}_i \quad \text{or} \quad (\lambda_i - \lambda_j) \mathbf{q}_j^H \mathbf{q}_i = 0 \quad (3.4.47)$$

Since the eigenvalues are assumed to be distinct, the only way (3.4.47) can be satisfied is if $\mathbf{q}_j^H \mathbf{q}_i = 0$ for $i \neq j$, which further proves that the corresponding eigenvectors are orthogonal to one another.

PROPERTY 3.4.7. Let $\{\mathbf{q}_i\}_{i=1}^M$ be an orthonormal set of eigenvectors corresponding to the distinct eigenvalues $\{\lambda_i\}_{i=1}^M$ of an $M \times M$ correlation matrix \mathbf{R} . Then \mathbf{R} can be diagonalized as follows:

$$\mathbf{\Lambda} = \mathbf{Q}^H \mathbf{R} \mathbf{Q} \quad (3.4.48)$$

where the orthonormal matrix $\mathbf{Q} \triangleq [\mathbf{q}_1 \cdots \mathbf{q}_M]$ is known as an *eigenmatrix* and $\mathbf{\Lambda}$ is an $M \times M$ diagonal eigenvalue matrix, that is,

$$\mathbf{\Lambda} \triangleq \text{diag}(\lambda_1, \dots, \lambda_M) \quad (3.4.49)$$

Proof. Arranging the vectors in (3.4.35) in a matrix format, we obtain

$$[\mathbf{R}\mathbf{q}_1 \ \mathbf{R}\mathbf{q}_2 \ \cdots \ \mathbf{R}\mathbf{q}_M] = [\lambda_1 \mathbf{q}_1 \ \lambda_2 \mathbf{q}_2 \ \cdots \ \lambda_M \mathbf{q}_M]$$

which, by using the definitions of \mathbf{Q} and $\mathbf{\Lambda}$, can be further expressed as

$$\mathbf{R}\mathbf{Q} = \mathbf{Q}\mathbf{\Lambda} \quad (3.4.50)$$

Since \mathbf{q}_i , $i = 1, \dots, M$, is an orthonormal set of vectors, the eigenmatrix \mathbf{Q} is unitary, that is, $\mathbf{Q}^{-1} = \mathbf{Q}^H$. Now premultiplying both sides of (3.4.50) by \mathbf{Q}^H , we obtain the desired result.

This diagonalization of the autocorrelation matrix plays an important role in filtering and estimation theory, as we shall see later. From (3.4.48) the correlation matrix \mathbf{R} can also be written as

$$\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^H = \lambda_1 \mathbf{q}_1 \mathbf{q}_1^H + \cdots + \lambda_M \mathbf{q}_M \mathbf{q}_M^H = \sum_{m=1}^M \lambda_m \mathbf{q}_m \mathbf{q}_m^H \quad (3.4.51)$$

which is known as the *spectral theorem*, or *Mercer's theorem*. If \mathbf{R} is positive definite (and hence invertible), its inverse is given by

$$\mathbf{R}^{-1} = (\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^H)^{-1} = \mathbf{Q}\mathbf{\Lambda}^{-1}\mathbf{Q}^H = \sum_{m=1}^M \frac{1}{\lambda_m} \mathbf{q}_m \mathbf{q}_m^H \quad (3.4.52)$$

because $\mathbf{\Lambda}$ is a diagonal matrix.

PROPERTY 3.4.8. The trace of \mathbf{R} is the summation of all eigenvalues, that is,

$$\text{tr}(\mathbf{R}) = \sum_{i=1}^M \lambda_i \quad (3.4.53)$$

Proof. See Problem 3.17.

$$\det \mathbf{R} = |\mathbf{R}| = \prod_{i=1}^M \lambda_i = |\Lambda| \quad (3.4.54)$$

Proof. See Problem 3.18.

PROPERTY 3.4.10. Determinants of \mathbf{R} and Γ are related by

$$|\mathbf{R}| = |\Gamma|(1 + \boldsymbol{\mu}_x^H \Gamma \boldsymbol{\mu}_x) \quad (3.4.55)$$

Proof. See Problem 3.19.

3.4.5 Correlation Matrices from Random Processes

A stochastic process can also be represented as a random vector, and its second-order statistics given by the mean vector and the correlation matrix. Obviously, these quantities are functions of the index n . Let an $M \times 1$ random vector $\mathbf{x}(n)$ be derived from the random process $x(n)$ as follows:

$$\mathbf{x}(n) \triangleq [x(n) \ x(n-1) \ \cdots \ x(n-M+1)]^T \quad (3.4.56)$$

Then its mean is given by an $M \times 1$ vector

$$\boldsymbol{\mu}_x(n) = [\mu_x(n) \ \mu_x(n-1) \ \cdots \ \mu_x(n-M+1)]^T \quad (3.4.57)$$

and the correlation by an $M \times M$ matrix

$$\mathbf{R}_x(n) = \begin{bmatrix} r_x(n, n) & \cdots & r_x(n, n-M+1) \\ \vdots & \ddots & \vdots \\ r_x(n-M+1, n) & \cdots & r_x(n-M+1, n-M+1) \end{bmatrix} \quad (3.4.58)$$

Clearly, $\mathbf{R}_x(n)$ is Hermitian since $r_x(n-i, n-j) = r_x^*(n-j, n-i)$, $0 \leq i, j \leq M-1$. This vector representation will be useful when we discuss optimum filters.

Correlation matrices of stationary processes

The correlation matrix $\mathbf{R}_x(n)$ of a general stochastic process $x(n)$ is a Hermitian $M \times M$ matrix defined in (3.4.58) with elements $r_x(n-i, n-j) = E\{x(n-i)x^*(n-j)\}$. For stationary processes this matrix has an interesting additional structure. First, $\mathbf{R}_x(n)$ is a constant matrix \mathbf{R}_x ; then using (3.3.24), we have

$$r_x(n-i, n-j) = r_x(j-i) = r_x(l \triangleq j-i) \quad (3.4.59)$$

Finally, by using conjugate symmetry $r_x(l) = r_x^*(-l)$, the matrix \mathbf{R}_x is given by

$$\mathbf{R}_x = \begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \cdots & r_x(M-1) \\ r_x^*(1) & r_x(0) & r_x(1) & \cdots & r_x(M-2) \\ r_x^*(2) & r_x^*(1) & r_x(0) & \cdots & r_x(M-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_x^*(M-1) & r_x^*(M-2) & r_x^*(M-3) & \cdots & r_x(0) \end{bmatrix} \quad (3.4.60)$$

It can be easily seen that \mathbf{R}_x is Hermitian and Toeplitz.[†] Thus, the autocorrelation matrix of a stationary process is Hermitian, nonnegative definite, and Toeplitz. Note that \mathbf{R}_x is not persymmetric because elements along the main antidiagonal are not equal, in general.

[†]A matrix is called *Toeplitz* if the elements along each diagonal, parallel to the main diagonal, are equal.

Eigenvalue spread and spectral dynamic range

The ill conditioning of a matrix \mathbf{R}_x increases with its condition number $\mathcal{X}(\mathbf{R}_x) = \lambda_{\max}/\lambda_{\min}$. When \mathbf{R}_x is a correlation matrix of a stationary process, then $\mathcal{X}(R_x)$ is bounded from above by the dynamic range of the PSD $R_x(e^{j\omega})$ of the process $x(n)$. The larger the spread in eigenvalues, the wider (or less flat) the variation of the PSD function. This is also related to the dynamic range or to the data spread in $x(n)$ and is a useful measure in practice. This result is given by the following theorem, in which we have dropped the subscript of $R_x(e^{j\omega})$ for clarity.

THEOREM 3.5. Consider a zero-mean stationary random process with autoPSD

$$R(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r(l)e^{-j\omega l}$$

$$\text{then } \min_{\omega} R(e^{j\omega}) \leq \lambda_i \leq \max_{\omega} R(e^{j\omega}) \quad \text{for all } i = 1, 2, \dots, M \quad (3.4.61)$$

Proof. From (3.4.41) we have

$$\lambda_i = \frac{\mathbf{q}_i^H \mathbf{R} \mathbf{q}_i}{\mathbf{q}_i^H \mathbf{q}_i} \quad (3.4.62)$$

Consider the quadratic form

$$\mathbf{q}_i^H \mathbf{R} \mathbf{q}_i = \sum_{k=1}^M \sum_{l=1}^M q_i(k) r(l-k) q_i(l)$$

where $\mathbf{q}_i = [q_i(1) \ q_i(2) \ \cdots \ q_i(M)]^T$. Using (3.3.41) and the stationarity of the process, we obtain

$$\begin{aligned} \mathbf{q}_i^H \mathbf{R} \mathbf{q}_i &= \frac{1}{2\pi} \sum_k \sum_l q_i^*(k) q_i(l) \int_{-\pi}^{\pi} R(e^{j\omega}) e^{j\omega(l-k)} d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} R(e^{j\omega}) \left[\sum_{k=1}^M q_i^*(k) e^{-j\omega k} \right] \left[\sum_{l=1}^M q_i(l) e^{j\omega l} \right] d\omega \end{aligned} \quad (3.4.63)$$

$$\text{or } \mathbf{q}_i^H \mathbf{R} \mathbf{q}_i = \frac{1}{2\pi} \int_{-\pi}^{\pi} R(e^{j\omega}) |Q(e^{j\omega})|^2 d\omega \quad (3.4.64)$$

Similarly, we have

$$\mathbf{q}_i^H \mathbf{q}_i = \frac{1}{2\pi} \int_{-\pi}^{\pi} |Q(e^{j\omega})|^2 d\omega \quad (3.4.65)$$

Substituting (3.4.64) and (3.4.65) in (3.4.62), we obtain

$$\lambda_i = \frac{\int_{-\pi}^{\pi} |Q(e^{j\omega})|^2 R(e^{j\omega}) d\omega}{\int_{-\pi}^{\pi} |Q(e^{j\omega})|^2 d\omega} \quad (3.4.66)$$

However, since $R(e^{j\omega}) \geq 0$, we have the following inequality:

$$\begin{aligned} \min_{\omega} R(e^{j\omega}) \int_{-\pi}^{\pi} |Q(e^{j\omega})|^2 d\omega &\leq \int_{-\pi}^{\pi} |Q(e^{j\omega})|^2 R(e^{j\omega}) d\omega \\ &\leq \max_{\omega} R(e^{j\omega}) \int_{-\pi}^{\pi} |Q(e^{j\omega})|^2 d\omega \end{aligned}$$

from which we easily obtain the desired result. The above result also implies that

$$\mathcal{X}(\mathbf{R}) \triangleq \frac{\lambda_{\max}}{\lambda_{\min}} \leq \frac{\max_{\omega} R(e^{j\omega})}{\min_{\omega} R(e^{j\omega})} \quad (3.4.67)$$

which becomes equality as $M \rightarrow \infty$.

In many practical and theoretical applications, it is desirable to represent a random vector (or sequence) with a linearly equivalent vector (or sequence) consisting of uncorrelated components. If \mathbf{x} is a correlated random vector and if \mathbf{A} is a nonsingular matrix, then the linear transformation

$$\mathbf{w} = \mathbf{Ax} \quad (3.5.1)$$

results in a random vector \mathbf{w} that contains the same “information” as \mathbf{x} , and hence random vectors \mathbf{x} and \mathbf{w} are said to be linearly equivalent. Furthermore, if \mathbf{w} has uncorrelated components and \mathbf{A} is lower-triangular, then each component w_i of \mathbf{w} can be thought of as *adding “new” information* (or *innovation*) to \mathbf{w} that is not present in the remaining components. Such a representation is called an *innovations representation* and provides additional insight into the understanding of random vectors and sequences. Additionally, it can simplify many theoretical derivations and can result in computationally efficient implementations.

Since $\Gamma_{\mathbf{w}}$ must be a diagonal matrix, we need to diagonalize the Hermitian, positive definite matrix $\Gamma_{\mathbf{x}}$ through the transformation matrix \mathbf{A} . There are two approaches to this diagonalization. One approach is to use the eigenanalysis presented in Section 3.4.4, which results in the well-known Karhunen-Loëve (KL) transform. The other approach is to use triangularization methods from linear algebra, which leads to the LDU (UDL) and LU (UL) decompositions. These vector techniques can be further extended to random sequences that give us the KL expansion and the spectral factorizations, respectively.

3.5.1 Transformations Using Eigendecomposition

Let \mathbf{x} be a random vector with mean vector $\mu_{\mathbf{x}}$ and covariance matrix $\Gamma_{\mathbf{x}}$. The linear transformation

$$\mathbf{x}_0 = \mathbf{x} - \mu_{\mathbf{x}} \quad (3.5.2)$$

results in a zero-mean vector \mathbf{x}_0 with correlation (and covariance) matrix equal to $\Gamma_{\mathbf{x}}$. This transformation shifts the origin of the M -dimensional coordinate system to the mean vector. We will now consider the zero-mean random vector \mathbf{x}_0 for further transformations.

Orthonormal transformation

Let $\mathbf{Q}_{\mathbf{x}}$ be the eigenmatrix of $\Gamma_{\mathbf{x}}$, and let us choose $\mathbf{Q}_{\mathbf{x}}^H$ as our linear transformation matrix \mathbf{A} in (3.2.32). Consider

$$\mathbf{w} = \mathbf{Q}_{\mathbf{x}}^H \mathbf{x}_0 = \mathbf{Q}_{\mathbf{x}}^H (\mathbf{x} - \mu_{\mathbf{x}}) \quad (3.5.3)$$

Then

$$\mu_{\mathbf{w}} = \mathbf{Q}_{\mathbf{x}}^H (E\{\mathbf{x}_0\}) = \mathbf{0} \quad (3.5.4)$$

and from (3.2.39) and (3.4.48)

$$\Gamma_{\mathbf{w}} = \mathbf{R}_{\mathbf{w}} = E\{\mathbf{Q}_{\mathbf{x}}^H \mathbf{x}_0 \mathbf{x}_0^H \mathbf{Q}_{\mathbf{x}}\} = \mathbf{Q}_{\mathbf{x}}^H \Gamma_{\mathbf{x}} \mathbf{Q}_{\mathbf{x}} = \Lambda_{\mathbf{x}} \quad (3.5.5)$$

Since $\Lambda_{\mathbf{x}}$ is diagonal, $\Gamma_{\mathbf{w}}$ is also diagonal, and hence this transformation has some interesting properties:

1. The random vector \mathbf{w} has zero mean, and its components are mutually uncorrelated (and hence orthogonal). Furthermore, if \mathbf{x} is $\mathcal{N}(\mu_{\mathbf{x}}, \Gamma_{\mathbf{x}})$, then \mathbf{w} is $\mathcal{N}(\mathbf{0}, \Lambda_{\mathbf{x}})$ with independent components.
2. The variances of random variables $w_i, i = 1, \dots, M$, are equal to the eigenvalues of $\Gamma_{\mathbf{x}}$.
3. Since the transformation matrix $\mathbf{A} = \mathbf{Q}_{\mathbf{x}}^H$ is orthonormal, the transformation is called an *orthonormal transformation* and the distance measure

$$d^2(\mathbf{x}_0) \triangleq \mathbf{x}_0^H \Gamma_{\mathbf{x}}^{-1} \mathbf{x}_0 \quad (3.5.6)$$

is preserved under the transformation. This distance measure is also known as the *Mahalanobis distance*; and in the case of normal random vectors, it is related to the log-likelihood function.

4. Since $\mathbf{w} = \mathbf{Q}_x^H(\mathbf{x} - \boldsymbol{\mu}_x)$, we have

$$w_i = \mathbf{q}_i^H(\mathbf{x} - \boldsymbol{\mu}_x) = \|\mathbf{x} - \boldsymbol{\mu}_x\| \cos[\angle(\mathbf{x} - \boldsymbol{\mu}_x, \mathbf{q}_i)] \quad i = 1, \dots, M \quad (3.5.7)$$

which is the projection of $\mathbf{x} - \boldsymbol{\mu}_x$ onto the unit vector \mathbf{q}_i . Thus \mathbf{w} represents \mathbf{x} in a new coordinate system that is shifted to $\boldsymbol{\mu}_x$ and spanned by $\mathbf{q}_i, i = 1, \dots, M$. A geometric interpretation of this transformation for a two-dimensional case is shown in Figure 3.11, which shows a contour of $d^2(\mathbf{x}_0) = \mathbf{x}^H \boldsymbol{\Gamma}_x^{-1} \mathbf{x} = \mathbf{w}^H \boldsymbol{\Lambda}_x^{-1} \mathbf{w}$ in the \mathbf{x} and \mathbf{w} coordinate systems ($\mathbf{w} = \mathbf{Q}_x^H \mathbf{x}$).

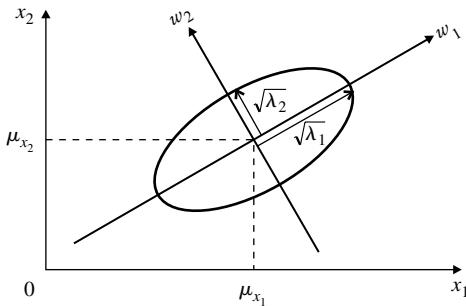


FIGURE 3.11
Orthogonal transformation in two dimensions.

Isotropic transformation

In the above orthonormal transformation, the autocorrelation matrix \mathbf{R}_w is diagonal but not an identity matrix \mathbf{I} . This can be achieved by an additional linear mapping of $\boldsymbol{\Lambda}_x^{-1/2}$. Let

$$\mathbf{y} = \boldsymbol{\Lambda}_x^{-1/2} \mathbf{w} = \boldsymbol{\Lambda}_x^{-1/2} \mathbf{Q}_x^H \mathbf{x}_0 = \boldsymbol{\Lambda}_x^{-1/2} \mathbf{Q}_x^H (\mathbf{x} - \boldsymbol{\mu}_x) \quad (3.5.8)$$

$$\text{Then } \mathbf{R}_y = \boldsymbol{\Lambda}_x^{-1/2} \mathbf{Q}_x^H \boldsymbol{\Gamma}_x \mathbf{Q}_x \boldsymbol{\Lambda}_x^{-1/2} = \boldsymbol{\Lambda}_x^{-1/2} \boldsymbol{\Lambda}_x \boldsymbol{\Lambda}_x^{-1/2} = \mathbf{I} \quad (3.5.9)$$

This is called an *isotropic transformation* because all components of \mathbf{y} are zero-mean, uncorrelated random variables with unit variance.[†] The geometric interpretation of this transformation for a two-dimensional case is shown in Figure 3.12. It clearly shows that there is not only a shift and rotation but also a scaling of the coordinate axis so that the distribution is equal in all directions, that is, it is direction-invariant. Because the transformation $\mathbf{A} = \boldsymbol{\Lambda}_x^{-1/2} \mathbf{Q}_x^H$ is orthogonal but not orthonormal, the distance measure $d^2(\mathbf{x}_0)$ is not preserved under this mapping. Since the correlation matrix after this transformation is an identity matrix \mathbf{I} , it is invariant under any orthonormal mapping, that is,

$$\mathbf{Q}^H \mathbf{I} \mathbf{Q} = \mathbf{Q}^H \mathbf{Q} = \mathbf{I} \quad (3.5.10)$$

This fact can be used for simultaneous diagonalization of two Hermitian matrices.

EXAMPLE 3.5.1. Consider a stationary sequence with correlation matrix

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

where $-1 < a < 1$. The eigenvalues

$$\lambda_1 = 1 + a \quad \lambda_2 = 1 - a$$

[†]In the literature, an isotropic transformation is also known as a *whitening* transformation. We believe that this terminology is not accurate because both vectors $\mathbf{Q}_x^H \mathbf{x}_0$ and $\boldsymbol{\Lambda}_x^{-1/2} \mathbf{Q}_x^H \mathbf{x}_0$ have uncorrelated coefficients.

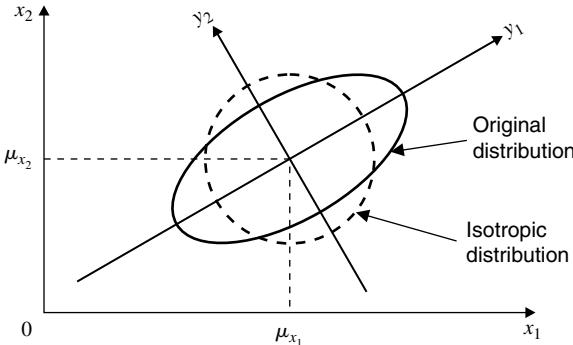


FIGURE 3.12
Isotropic transformation in two dimensions.

are obtained from the characteristic equation

$$\det(\mathbf{R}_x - \lambda \mathbf{I}) = \det \begin{bmatrix} 1 - \lambda & a \\ a & 1 - \lambda \end{bmatrix} = (1 - \lambda)^2 - a^2 = 0$$

To find the eigenvector \mathbf{q}_1 , we solve the linear system

$$\begin{bmatrix} 1 & a \\ a & 1 \end{bmatrix} \begin{bmatrix} q_1^{(1)} \\ q_2^{(1)} \end{bmatrix} = (1 + a) \begin{bmatrix} q_1^{(1)} \\ q_2^{(1)} \end{bmatrix}$$

which gives $q_1^{(1)} = q_2^{(1)}$. Similarly, we find that $q_1^{(2)} = -q_2^{(2)}$. If we normalize both vectors to unit length, we obtain the eigenvectors

$$\mathbf{q}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{q}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

From the above results we see that $\det \mathbf{R}_x = 1 - a^2 = \lambda_1 \lambda_2$ and $\mathbf{Q}^H \mathbf{Q} = \mathbf{I}$, where $\mathbf{Q} = [\mathbf{q}_1 \mathbf{q}_2]$.

3.5.2 Transformations Using Triangular Decomposition

The linear transformations discussed above were based on diagonalization of hermitian matrices through eigenvalue-eigenvector decomposition. These are useful in many detection and estimation problems. Triangular matrix decomposition leads to transformations that result in *causal* or *anticausal* linear filtering of associated sequences. Hence these mappings play an important role in linear filtering. There are two such decompositions: the *lower-diagonal-upper (LDU)* one leads to causal filtering while the *upper-diagonal-lower (UDL)* one results in anticausal filtering.

Lower-diagonal-upper decomposition

Any Hermitian, positive definite matrix \mathbf{R} can be factored as (Goulob and Van Loan 1989)

$$\mathbf{R} = \mathbf{L} \mathbf{D}_L \mathbf{L}^H \tag{3.5.11}$$

or equivalently

$$\mathbf{L}^{-1} \mathbf{R} \mathbf{L}^{-H} = \mathbf{D}_L \tag{3.5.12}$$

where \mathbf{L} is a *unit lower triangular* matrix, \mathbf{D}_L is a diagonal matrix with positive elements, and \mathbf{L}^H is a *unit upper triangular* matrix. The MATLAB function `[L, D]=ldlt(R)`, given in Section 5.2, computes the LDU decomposition.

Since \mathbf{L} is unit lower triangular, we have $\det \mathbf{R} = \prod_{i=1}^M \xi_i^l$, where ξ_1^l, \dots, ξ_M^l are the diagonal elements of \mathbf{D}_L . If we define the linear transformation

$$\mathbf{w} = \mathbf{L}^{-1} \mathbf{x} \triangleq \mathbf{B} \mathbf{x} \tag{3.5.13}$$

we find that

$$\mathbf{R}_w = E\{\mathbf{w}\mathbf{w}^H\} = \mathbf{L}^{-1}E\{\mathbf{x}\mathbf{x}^H\}\mathbf{L}^{-H} = \mathbf{L}^{-1}\mathbf{R}\mathbf{L}^{-H} = \mathbf{D}_L \quad (3.5.14)$$

Clearly, the components of \mathbf{w} are orthogonal, and the elements ξ_1^l, \dots, ξ_M^l are their second moments. Therefore, this transformation appears to be similar to the orthogonal one. However, the vector \mathbf{w} is not obtained as a simple rotation of \mathbf{x} . To understand this mapping, we first note that $\mathbf{B} = \mathbf{L}^{-1}$ is also a unit lower triangular matrix (Goulob and Van Loan 1989). Then we can write (3.5.13) as

$$\begin{bmatrix} w_1 \\ \vdots \\ w_i \\ \vdots \\ w_M \end{bmatrix} = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & & & \vdots \\ b_{i1} & \cdots & 1 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ b_{M1} & \cdots & b_{Mi} & \cdots & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_M \end{bmatrix} \quad (3.5.15)$$

where b_{ik} are elements of \mathbf{B} . From (3.5.15) we conclude that w_i is a linear combination of $x_k, k \leq i$, that is,

$$w_i = \sum_{k=1}^i b_{ik}x_k \quad 1 \leq i \leq M \quad (3.5.16)$$

If the signal vector \mathbf{x} consists of consecutive samples of a discrete-time stochastic process $x(n)$, that is,

$$\mathbf{x} = [x(n) \ x(n-1) \ \cdots \ x(n-M+1)]^T \quad (3.5.17)$$

then (3.5.16) can be interpreted as a causal linear filtering of the random sequence (see Chapter 2). This transformation will be used extensively in optimum linear filtering and prediction problems.

A similar LDU decomposition of autocovariance matrices can be performed by following the identical steps above. In this case, the components of the transformed vector \mathbf{w} are uncorrelated, and the elements $\xi_i^u, 1 \leq i \leq M$, of \mathbf{D}_U are variances.

Upper-diagonal-lower decomposition

This diagonalization is almost identical to the previous one and involves factorization of a Hermitian, positive definite matrix into an upper-diagonal-lower form. It is given by

$$\mathbf{R} = \mathbf{U}\mathbf{D}_U\mathbf{U}^H \quad (3.5.18)$$

or equivalently $\mathbf{U}^{-1}\mathbf{R}\mathbf{U}^{-H} = \mathbf{D}_U = \text{diag}(\xi_1^u, \dots, \xi_M^u)$ (3.5.19)

in which the matrix \mathbf{U} is unit upper triangular, the matrix \mathbf{U}^H is unit lower triangular, and the matrix \mathbf{D}_U is diagonal with positive elements. Note that $\mathbf{U}^H \neq \mathbf{L}$ and $\mathbf{D}_U \neq \mathbf{D}_L$. Following the same analysis as above, we have $\det \mathbf{R} = \det \mathbf{D}_U = \prod_{i=1}^M \xi_i^u$. Since $\mathbf{A} = \mathbf{U}^{-1}$ is unit upper triangular in the transformation $\mathbf{w} = \mathbf{U}^{-1}\mathbf{x}$, the components of \mathbf{w} are orthogonal and are obtained by linear combinations of $x_k, k \geq i$, that is,

$$w_i = \sum_{k=i}^M l_{ik}x_k \quad 1 \leq i \leq M \quad (3.5.20)$$

This represents an anticausal filtering of a random sequence if \mathbf{x} is a signal vector. Table 3.3 compares and contrasts orthogonal and triangular decompositions. We note that the LDU decomposition does not have the nice geometric interpretation (rotation of the coordinate system) of the eigendecomposition transformation.

Generation of real-valued random vectors with given second-order moments. Suppose that we want to generate M samples, say, x_1, x_2, \dots, x_M , of a real-valued random vector \mathbf{x} with mean $\mathbf{0}$ and a given symmetric and positive definite autocorrelation matrix \mathbf{R}_x .

TABLE 3.3
Comparison of orthogonal and triangular decompositions
for zero-mean random vectors.

Orthogonal decomposition	Triangular decomposition
$\mathbf{R} = E\{\mathbf{x}\mathbf{x}^H\}$	$\mathbf{R} = E\{\mathbf{x}\mathbf{x}^H\}$
$\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i$	
$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_M]$	\mathbf{L} = unit lower triangular
$\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_M)$	$\mathbf{D} = \text{diag}(\xi_1, \xi_2, \dots, \xi_M)$
$\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^H = \sum_{i=1}^M \lambda_i \mathbf{q}_i \mathbf{q}_i^H$	$\mathbf{R} = \mathbf{L}\mathbf{D}\mathbf{L}^H$
$\mathbf{\Lambda} = \mathbf{Q}^H \mathbf{R} \mathbf{Q}$	$\mathbf{D} = \mathbf{L}^{-1} \mathbf{R} \mathbf{L}^{-H}$
$\mathbf{R}^{-1} = \mathbf{Q}\mathbf{\Lambda}^{-1}\mathbf{Q}^H = \sum_{i=1}^M \frac{1}{\lambda_i} \mathbf{q}_i \mathbf{q}_i^H$	$\mathbf{R}^{-1} = \mathbf{L}^{-H} \mathbf{D}^{-1} \mathbf{L}^{-1}$
$\mathbf{\Lambda}^{-1} = \mathbf{Q}^H \mathbf{R}^{-1} \mathbf{Q}$	$\mathbf{D}^{-1} = \mathbf{L}^{-H} \mathbf{R}^{-1} \mathbf{L}^{-1}$
$\det \mathbf{R} = \det \mathbf{\Lambda} = \prod_{i=1}^M \lambda_i$	$\det \mathbf{R} = \det \mathbf{D} = \prod_{i=1}^M \xi_i$
$\text{tr } \mathbf{R} = \text{tr } \mathbf{\Lambda} = \sum_{i=1}^M \lambda_i$	
Whitening (noncausal)	Whitening (causal)
$\mathbf{w} = \mathbf{Q}^H \mathbf{x}$	$\mathbf{w} = \mathbf{L}^{-1} \mathbf{x}$
$E\{\mathbf{w}\mathbf{w}^H\} = \mathbf{\Lambda}$	$E\{\mathbf{w}\mathbf{w}^H\} = \mathbf{D}$

The innovations representation given in this section suggests three approaches to generate samples of such a random vector. The general approach is to factor \mathbf{R}_x , using either the orthonormal or the triangularization transformation, to obtain the diagonal matrix ($\mathbf{\Lambda}_x$ or $\mathbf{D}_L^{(x)}$ or $\mathbf{D}_U^{(x)}$), generate M samples of an IID sequence with the obtained diagonal variances, and then transform these samples by using the inverse transformation matrix (\mathbf{Q}_x or \mathbf{L}_x or \mathbf{U}_x). We hasten to add that, in general, the original distribution of the IID samples will not be preserved unless the samples are jointly normal. Therefore, in the following discussion, we assume that a normal pseudorandom number generator is used to generate M independent samples of \mathbf{w} . The three methods are as follows.

Eigendecomposition approach. First factor \mathbf{R}_x as $\mathbf{R}_x = \mathbf{Q}_x \mathbf{\Lambda}_x \mathbf{Q}_x^H$. Then generate \mathbf{w} , using the distribution $\mathcal{N}(\mathbf{0}, \mathbf{\Lambda}_x)$. Finally, compute the desired vector \mathbf{x} , using $\mathbf{x} = \mathbf{Q}_x \mathbf{w}$.

LDU triangularization approach. First factor \mathbf{R}_x as $\mathbf{R}_x = \mathbf{L}_x \mathbf{D}_L^{(x)} \mathbf{L}_x^H$. Then generate \mathbf{w} , using the distribution $\mathcal{N}(\mathbf{0}, \mathbf{D}_L^{(x)})$. Finally, compute the desired vector \mathbf{x} , using $\mathbf{x} = \mathbf{L}_x \mathbf{w}$.[†]

UDL triangularization approach. First factor \mathbf{R}_x as $\mathbf{R}_x = \mathbf{U}_x \mathbf{D}_U^{(x)} \mathbf{U}_x^H$. Then generate \mathbf{w} , using the distribution $\mathcal{N}(\mathbf{0}, \mathbf{D}_U^{(x)})$. Finally, compute the desired vector \mathbf{x} , using $\mathbf{x} = \mathbf{U}_x \mathbf{w}$.

Additional discussion and more complete treatment on the generation of random vectors are given in Johnson (1994).

3.5.3 The Discrete Karhunen-Loève Transform

In many signal processing applications, it is convenient to represent the samples of a random signal in another set of numbers (or coefficients) so that this new representation possesses some useful properties. For example, for coding purposes we want to transform a signal

[†]If we use the Cholesky decomposition $\mathbf{R}_x = \tilde{\mathbf{L}}_x \tilde{\mathbf{L}}_x^H$, where $\tilde{\mathbf{L}}_x = \{\mathbf{D}_L^{(x)}\}^{1/2} \mathbf{L}_x$, then $\mathbf{w} = \mathcal{N}(\mathbf{0}, \mathbf{I})$ will generate \mathbf{x} with the given correlation \mathbf{R}_x , using $\mathbf{x} = \tilde{\mathbf{L}}_x \mathbf{w}$.

so that its energy is concentrated in only a few coefficients (which are then transmitted); or for optimal filtering purposes we may want uncorrelated samples so that the filtering complexity is reduced or the signal-to-noise ratio is enhanced. A general approach is to expand a signal as a linear combination of orthogonal basis functions so that components of the signal with respect to basis functions do not interfere with one another. There are several such basis functions; the most widely known is the set of complex exponentials used in DTFT (or DFT) that are used in linear filtering, as we discussed in Section 3.4. Other examples are functions used in discrete cosine transform, discrete sine transform, Haar transform, etc., which are useful in coding applications (Jain 1989).

As discussed in this section, a set of orthogonal basis functions for which the signal components are statistically uncorrelated to one another is based on the second-order properties of the random process and, in particular, on the diagonalization of its covariance matrix. It is also an optimal representation of the signal in the sense that it provides a representation with the *smallest mean square error* among all other orthogonal transforms. This has applications in the analysis of random signals as well as in coding. This transform was first suggested by Karhunen and Loève for continuous random processes. It was extended to discrete random signals by Hotelling and is also known as the Hotelling transform. In keeping with the current nomenclature, we will call it the *discrete Karhunen-Loève transform (DKLT)* (Fukunaga 1990).

Development of the DKLT

Let $\mathbf{x} = [x_1 \ x_2 \ \dots \ x_M]^T$ be a zero-mean[†] random vector with autocorrelation matrix \mathbf{R}_x . We want to represent \mathbf{x} using the linear transformation

$$\mathbf{w} = \mathbf{A}^H \mathbf{x} \quad \mathbf{A}^{-1} = \mathbf{A}^H \quad (3.5.21)$$

where \mathbf{A} is a unitary matrix. Then

$$\mathbf{x} = \mathbf{Aw} = \sum_{i=1}^M w_i \mathbf{a}_i \quad \mathbf{a}_i^H \mathbf{a}_j = 0 \quad i \neq j \quad (3.5.22)$$

Let us represent \mathbf{x} using the first m , $1 \leq m \leq M$, components of \mathbf{w} , that is,

$$\hat{\mathbf{x}} \triangleq \sum_{i=1}^m w_i \mathbf{a}_i \quad 1 \leq m \leq M \quad (3.5.23)$$

Then from (3.5.22) and (3.5.23), the error between \mathbf{x} and $\hat{\mathbf{x}}$ is given by

$$\mathbf{e}_m \triangleq \mathbf{x} - \hat{\mathbf{x}} = \sum_{i=1}^M w_i \mathbf{a}_i - \sum_{i=1}^m w_i \mathbf{a}_i = \sum_{i=m+1}^M w_i \mathbf{a}_i \quad (3.5.24)$$

and hence the mean-squared error (MSE) is

$$E_m \triangleq E\{\mathbf{e}_m^H \mathbf{e}_m\} = \sum_{i=m+1}^M \mathbf{a}_i^H E\{|w_i|^2\} \mathbf{a}_i = \sum_{i=m+1}^M E\{|w_i|^2\} \mathbf{a}_i^H \mathbf{a}_i \quad (3.5.25)$$

Since from (3.5.21) $w_i = \mathbf{a}_i^H \mathbf{x}$, we have $E\{|w_i|^2\} = \mathbf{a}_i^H \mathbf{R}_x \mathbf{a}_i$. Now we want to determine the matrix \mathbf{A} that will minimize the MSE E_m subject to $\mathbf{a}_i^H \mathbf{a}_i = 1$, $i = m + 1, \dots, M$ so that from (3.5.25)

$$E_m = \sum_{i=m+1}^M E\{|w_i|^2\} = \sum_{i=m+1}^M \mathbf{a}_i^H \mathbf{R}_x \mathbf{a}_i \quad \mathbf{a}_i^H \mathbf{a}_i = 1 \quad i = m + 1, \dots, M \quad (3.5.26)$$

[†]If the mean is not zero, then we perform the transformation on the mean-subtracted vector, using the covariance matrix.

This optimization can be done by using the Lagrange multiplier approach (Appendix B); that is, we minimize

$$\sum_{i=m+1}^M \mathbf{a}_i^H \mathbf{R}_x \mathbf{a}_i + \sum_{i=m+1}^M \lambda_i (1 - \mathbf{a}_i^H \mathbf{a}_i) \quad i = m+1, \dots, M$$

Hence after setting the gradient equal to zero,

$$\nabla_{\mathbf{a}_i} \left[\sum_{i=m+1}^M \mathbf{a}_i^H \mathbf{R}_x \mathbf{a}_i + \sum_{i=m+1}^M \lambda_i (1 - \mathbf{a}_i^H \mathbf{a}_i) \right] = (\mathbf{R}_x \mathbf{a}_i)^* - (\lambda_i \mathbf{a}_i)^* = 0 \quad (3.5.27)$$

we obtain

$$\mathbf{R}_x \mathbf{a}_i = \lambda_i \mathbf{a}_i \quad i = m+1, \dots, M$$

which is equivalent to (3.4.35) in the eigenanalysis of Section 3.4.4. Hence λ_i is the eigenvalue, and the corresponding \mathbf{a}_i is the eigenvector of \mathbf{R}_x . Clearly, since $1 \leq m \leq M$, the transformation matrix \mathbf{A} should be chosen as the eigenmatrix \mathbf{Q} . Hence

$$\begin{bmatrix} \uparrow \\ \mathbf{w} \\ \downarrow \end{bmatrix} = \begin{bmatrix} \leftarrow & \mathbf{q}_1^H & \rightarrow \\ \leftarrow & \mathbf{q}_2^H & \rightarrow \\ \vdots & \vdots & \vdots \\ \leftarrow & \mathbf{q}_M^H & \rightarrow \end{bmatrix} \begin{bmatrix} \uparrow \\ \mathbf{x} \\ \downarrow \end{bmatrix}$$

or more concisely

$$\mathbf{w} = \mathbf{Q}^H \mathbf{x} \quad (3.5.28)$$

provides an orthonormal transformation so that the transformed vector \mathbf{w} is a zero-mean, uncorrelated random vector with autocorrelation $\mathbf{\Lambda}$. This transformation is called the DKLT, and its inverse relationship (or synthesis) is given by

$$\begin{bmatrix} \uparrow \\ \mathbf{x} \\ \downarrow \end{bmatrix} = \begin{bmatrix} \uparrow & \uparrow & \cdots & \uparrow \\ \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_M \\ \downarrow & \downarrow & \cdots & \downarrow \end{bmatrix} \begin{bmatrix} \uparrow \\ \mathbf{w} \\ \downarrow \end{bmatrix} \quad (3.5.29)$$

or

$$\mathbf{x} = \mathbf{Q}\mathbf{w} = \mathbf{q}_1 w_1 + \mathbf{q}_2 w_2 + \cdots + \mathbf{q}_M w_M \quad (3.5.30)$$

From Section 3.5.1, the geometric interpretation of this transformation is that $\{w_k\}_1^M$ are projections of the vector \mathbf{x} with respect to the rotated coordinate system of $\{\mathbf{q}_k\}_1^M$. The eigenvalues λ_i also have an interesting interpretation, as we shall see in the following representation.

Optimal reduced-basis representation

Generally we would expect any transformation to provide only few meaningful components so that we can use only those basis vectors resulting in a smaller representation error. To determine this *reduced-basis representation* property of the DKLT, let us use first $K < M$ eigenvectors (instead of all \mathbf{q}_i). Then from (3.5.26), we have

$$E_K = \sum_{i=K+1}^M \lambda_i \quad (3.5.31)$$

In other words, the MSE in the reduced-basis representation, when the first K basis vectors are used, is the sum of the remaining eigenvalues (which are never negative). Therefore, to obtain a minimum MSE (that is, an optimum) representation, the procedure is to choose K eigenvectors corresponding to the K largest eigenvalues.

Application in data compression. The DKLT is a transformation on a random vector that produces a zero-mean, uncorrelated vector and that can minimize the mean square representation error. One of its popular applications is data compression in communications

and, in particular, in speech and image coding. Suppose we want to send a sample function of a speech process $x_c(t)$. If we sample this waveform and obtain M samples $\{x(n)\}_0^{M-1}$, then we need to send M data values. Instead, if we analyze the correlation of $\{x(n)\}_0^{M-1}$ and determine that M values can be approximated by a smaller K numbers of w_i and the corresponding \mathbf{q}_i , then we can compute these K data values $\{\hat{w}_i\}_1^K$ at the transmitter and send them to the receiver through the communication channel. At the receiver, we can reconstruct $\{x(n)\}_0^{M-1}$ by using (3.5.23), as shown in Figure 3.13. Obviously, both the transmitter and receiver must have the information about the eigenvectors $\{\mathbf{q}_i\}_1^M$. A considerable amount of compression is achieved if K is much smaller than M .

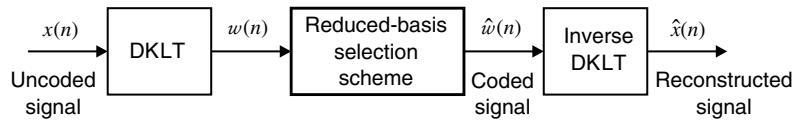


FIGURE 3.13

Signal coding scheme using the DKLT.

Periodic random sequences

As we noted in the previous section, the correlation matrix of a stationary process is Toeplitz. If the autocorrelation sequence of a random process is periodic with fundamental period M , its correlation matrix becomes *circulant*. All rows (columns) of a circulant matrix are obtained by circular rotation of its first row (column). Using (3.4.60) and the periodicity relation $r_x(l) = r_x(l - M)$, we obtain

$$\mathbf{R}_x = \begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \cdots & r_x(M-1) \\ r_x(M-1) & r_x(0) & r_x(1) & \cdots & r_x(M-2) \\ r_x(M-2) & r_x(M-1) & r_x(0) & \cdots & r_x(M-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_x(1) & r_x(2) & r_x(3) & \cdots & r_x(0) \end{bmatrix} \quad (3.5.32)$$

which is a circulant matrix. We note that a circulant matrix is Toeplitz but not vice versa.

If we define the M -point DFT of the periodic sequence $r_x(l)$

$$\tilde{R}_x(k) = \sum_{l=0}^{M-1} r_x(l) W_M^{kl} \quad (3.5.33)$$

where $W_M \triangleq e^{-j2\pi/M}$, and the vector

$$\mathbf{w}_k \triangleq \frac{1}{\sqrt{M}} [1 \ W_M^k \ W_M^{2k} \ \cdots \ W_M^{(M-1)k}]^T \quad 0 \leq k \leq M-1 \quad (3.5.34)$$

we can easily see that multiplying the first row of \mathbf{R}_x by the vector \mathbf{w}_k results in $\tilde{R}_x(k)/\sqrt{M}$. Using $W_M^{-k} = W_M^{(M-1)k}$, we find that the product of the second row by \mathbf{w}_k is equal to $\tilde{R}_x(k)W_M^k/\sqrt{M}$. In general, the i th row by \mathbf{w}_k gives $\tilde{R}_x(k)W_M^{(i-1)k}/\sqrt{M}$. Therefore, we have

$$\mathbf{R}_x \mathbf{w}_k = \tilde{R}_x(k) \mathbf{w}_k \quad 0 \leq k \leq M-1 \quad (3.5.35)$$

which shows that the normalized DFT vectors \mathbf{w}_k are the eigenvectors of the circulant matrix \mathbf{R}_x with corresponding eigenvalues the DFT coefficients $\tilde{R}_x(k)$. Therefore, *the DFT provides the DKLT of periodic random sequences*. We recall that $\tilde{R}_x(k)$ are samples of the DTFT $R_x(e^{j2\pi k/M})$ of the finite-length sequence $r_x(l)$, $0 \leq l \leq M-1$.

If we define the $M \times M$ matrix

$$\mathbf{W} \triangleq [\mathbf{w}_0 \ \mathbf{w}_1 \ \cdots \ \mathbf{w}_{M-1}] \quad (3.5.36)$$

we can show that

$$\mathbf{W}^H \mathbf{W} = \mathbf{W} \mathbf{W}^H = \mathbf{I} \quad (3.5.37)$$

that is, the matrix \mathbf{W} is unitary. The set of equations (3.5.35) can be written as

$$\mathbf{W}^H \mathbf{R}_x \mathbf{W} = \text{diag}\{\tilde{R}_x(0), \tilde{R}_x(1), \dots, \tilde{R}_x(M-1)\} \quad (3.5.38)$$

which shows that the DFT performs the diagonalization of circulant matrices. Although there is no fast algorithm for the diagonalization of general Toeplitz matrices, in many cases we can use the DFT to approximate the DKLT of stationary random sequences. The approximation is adequate if the correlation becomes negligible for $|l| > M$, which is the case for many stationary processes. This explains the fact that the eigenvectors of a Toeplitz matrix resemble complex exponentials for large values of M . The DKLT also can be extended to handle the representation of random sequences. These issues are further explored in Therrien (1992), Gray (1972), and Fukunaga (1990).

3.6 PRINCIPLES OF ESTIMATION THEORY

The key assumption underlying our discussion up to this point was that the probability distributions associated with the problem under consideration were known. As a result, all required probabilities, autocorrelation sequences, and PSD functions either could be derived from a set of assumptions about the involved random processes or were given a priori. However, in most practical applications, this is the exception rather than the rule. Therefore, the properties and parameters of random variables and random processes should be obtained by collecting and analyzing finite sets of measurements. In this section, we introduce some basic concepts of estimation theory that will be used repeatedly in the rest of the book. Complete treatments of estimation theory can be found in Kay (1993), Helstrom (1995), Van Trees (1968), and Papoulis (1991).

3.6.1 Properties of Estimators

Suppose that we collect N observations $\{x(n)\}_0^{N-1}$ from a stationary stochastic process and use them to estimate a parameter θ (which we assume to be real-valued) of the process using some function $\hat{\theta}[\{x(n)\}_0^{N-1}]$. The same results can be used for a set of measurements $\{x_k(n)\}_1^N$ obtained from N sensors sampling stochastic processes with the same distributions. The function $\hat{\theta}[\{x(n)\}_0^{N-1}]$ is known as an *estimator* whereas the value taken by the estimator, using a particular set of observations, is called a *point estimate* or simply an *estimate*. The intention of the estimator design is that the estimate should be as close to the true value of the parameter as possible. However, if we use another set of observations or a different number of observations from the same set, it is highly unlikely that we will obtain the same estimate. As an example of an estimator, consider estimating the mean μ_x of a stationary process $x(n)$ from its N observations $\{x(n)\}_0^{N-1}$. Then the natural estimator is a simple arithmetic average of these observations, given by

$$\hat{\mu}_x = \hat{\theta}[\{x(n)\}_0^{N-1}] = \frac{1}{N} \sum_{n=0}^{N-1} x(n) \quad (3.6.1)$$

Similarly, a natural estimator of the variance σ_x^2 of the process $x(n)$ would be

$$\hat{\sigma}_x^2 = \hat{\theta}[\{x(n)\}_0^{N-1}] = \frac{1}{N} \sum_{n=0}^{N-1} [x(n) - \hat{\mu}_x]^2 \quad (3.6.2)$$

If we repeat this procedure a large number of times, we will obtain a large number of estimates, which can be used to generate a histogram showing the distribution of the estimates. Before the collection of observations, we would like to describe all sets of data that can be obtained by using the random variables $\{x(n, \zeta)\}_0^{N-1}$. The obtained set of N observations $\{x(n)\}_0^{N-1}$ can thus be regarded as one realization of the random variables $\{x(n, \zeta)\}_0^{N-1}$ defined on an N -dimensional sample space. In this sense, the estimator $\hat{\theta}[\{x(n, \zeta)\}_0^{N-1}]$ becomes a random variable whose distribution can be obtained from the joint distribution of the random variables $\{x(n, \zeta)\}_0^{N-1}$. This distribution is called the *sampling distribution* of the estimator and is a fundamental concept in estimation theory because it provides all the information we need to evaluate the quality of an estimator.

The sampling distribution of a “good” estimator should be concentrated as closely as possible about the parameter that it estimates. To determine how “good” an estimator is and how different estimators of the same parameter compare with one another, we need to determine their sampling distributions. Since it is *not* always possible to derive the exact sampling distributions, we have to resort to properties that use the lower-order moments (mean, variance, mean square error) of the estimator.

Bias of estimator. The *bias* of an estimator $\hat{\theta}$ of a parameter θ is defined as

$$B(\hat{\theta}) \triangleq E[\hat{\theta}] - \theta \quad (3.6.3)$$

while the *normalized* bias is defined as

$$\varepsilon_b \triangleq \frac{B(\hat{\theta})}{\theta} \quad \theta \neq 0 \quad (3.6.4)$$

When $B(\hat{\theta}) = 0$, the estimator is said to be *unbiased* and the pdf of the estimator is centered exactly at the true value θ . Generally, one should select estimators that are unbiased such as the mean estimator in (3.6.1) or very nearly unbiased such as the variance estimator in (3.6.2). However, it is not always wise to select an unbiased estimator, as we will see below and in Section 5.2 on the estimation of autocorrelation sequences.

Variance of estimator. The *variance* of the estimator $\hat{\theta}$ is defined by

$$\text{var}(\hat{\theta}) = \sigma_{\hat{\theta}}^2 \triangleq E\{|\hat{\theta} - E\{\hat{\theta}\}|^2\} \quad (3.6.5)$$

which measures the spread of the pdf of $\hat{\theta}$ around its average value. Therefore, one would select an estimator with the smallest variance. However, this selection is not always compatible with the small bias requirement. As we will see below, reducing variance may result in an increase in bias. Therefore, a balance between these two conflicting requirements is required, which is provided by the mean square error property. The *normalized standard deviation* (also called the coefficient of variation) is defined by

$$\varepsilon_r \triangleq \frac{\sigma_{\hat{\theta}}}{\theta} \quad \theta \neq 0 \quad (3.6.6)$$

Mean square error. The *mean square error (MSE)* of the estimator is given by

$$\text{MSE}(\theta) = E\{|\hat{\theta} - \theta|^2\} = \sigma_{\hat{\theta}}^2 + |B_{\hat{\theta}}|^2 \quad (3.6.7)$$

Indeed, we have

$$\begin{aligned} \text{MSE}(\theta) &= E\{|\theta - E\{\hat{\theta}\} - (\hat{\theta} - E\{\hat{\theta}\})|^2\} \\ &= E\{|\theta - E\{\hat{\theta}\}|^2\} + E\{|\hat{\theta} - E\{\hat{\theta}\}|^2\} \end{aligned} \quad (3.6.8)$$

$$\begin{aligned} &\quad -(\theta - E\{\hat{\theta}\})E\{(\hat{\theta} - E\{\hat{\theta}\})^*\} - (\theta - E\{\hat{\theta}\})^*E\{\hat{\theta} - E\{\hat{\theta}\}\} \\ &= |\theta - E\{\hat{\theta}\}|^2 + E\{|\hat{\theta} - E\{\hat{\theta}\}|^2\} \end{aligned} \quad (3.6.9)$$

which leads to (3.6.7) by using (3.6.3) and (3.6.5). Ideally, we would like to minimize the MSE, but this minimum is not always zero. Hence minimizing variance can increase the bias. The *normalized MSE* is defined as

$$\varepsilon \triangleq \frac{\text{MSE}(\theta)}{\theta} \quad \theta \neq 0 \quad (3.6.10)$$

Cramér-Rao lower bound. If it is possible to minimize the MSE when the bias is zero, then clearly the variance is also minimized. Such estimators are called *minimum variance unbiased* estimators, and they attain an important minimum bound on the variance of the estimator, called the *Cramér-Rao lower bound* (CRLB), or *minimum variance bound*. If $\hat{\theta}$ is unbiased, then it follows that $E\{\hat{\theta} - \theta\} = 0$, which may be expressed as

$$\int_{-\infty}^{\infty} \cdots \int (\hat{\theta} - \theta) f_{\mathbf{x};\theta}(\mathbf{x}; \theta) d\mathbf{x} = 0 \quad (3.6.11)$$

where $\mathbf{x}(\zeta) = [x_1(\zeta), x_2(\zeta), \dots, x_N(\zeta)]^T$ and $f_{\mathbf{x};\theta}(\mathbf{x}; \theta)$ is the joint density of $\mathbf{x}(\zeta)$, which depends on a fixed but unknown parameter θ . If we differentiate (3.6.11) with respect to θ , assuming real-valued $\hat{\theta}$, we obtain

$$0 = \int_{-\infty}^{\infty} \cdots \int \frac{\partial}{\partial \theta} [(\hat{\theta} - \theta) f_{\mathbf{x};\theta}(\mathbf{x}; \theta)] d\mathbf{x} = \int_{-\infty}^{\infty} \cdots \int (\hat{\theta} - \theta) \frac{\partial f_{\mathbf{x};\theta}(\mathbf{x}; \theta)}{\partial \theta} d\mathbf{x} - 1 \quad (3.6.12)$$

Using the fact

$$\begin{aligned} \frac{\partial \ln[f_{\mathbf{x};\theta}(\mathbf{x}; \theta)]}{\partial \theta} &= \frac{1}{f_{\mathbf{x};\theta}(\mathbf{x}; \theta)} \frac{\partial f_{\mathbf{x};\theta}(\mathbf{x}; \theta)}{\partial \theta} \\ \text{or} \quad \frac{\partial f_{\mathbf{x};\theta}(\mathbf{x}; \theta)}{\partial \theta} &= \frac{\partial \ln[f_{\mathbf{x};\theta}(\mathbf{x}; \theta)]}{\partial \theta} f_{\mathbf{x};\theta}(\mathbf{x}; \theta) \end{aligned} \quad (3.6.13)$$

and substituting (3.6.13) in (3.6.12), we get

$$\int_{-\infty}^{\infty} \cdots \int \left\{ (\hat{\theta} - \theta) \frac{\partial \ln[f_{\mathbf{x};\theta}(\mathbf{x}; \theta)]}{\partial \theta} \right\} f_{\mathbf{x};\theta}(\mathbf{x}; \theta) d\mathbf{x} = 1 \quad (3.6.14)$$

Clearly, the left side of (3.6.14) is simply the expectation of the expression inside the brackets, that is,

$$E \left\{ (\hat{\theta} - \theta) \frac{\partial \ln[f_{\mathbf{x};\theta}(\mathbf{x}; \theta)]}{\partial \theta} \right\} = 1 \quad (3.6.15)$$

Using the *Cauchy-Schwarz inequality* (Papoulis 1991; Stark and Woods 1994) $|E\{x(\zeta)y(\zeta)\}|^2 \leq E\{|x(\zeta)|^2\}E\{|y(\zeta)|^2\}$, we obtain

$$E\{(\hat{\theta} - \theta)^2\} E \left\{ \left(\frac{\partial \ln[f_{\mathbf{x};\theta}(\mathbf{x}; \theta)]}{\partial \theta} \right)^2 \right\} \geq E^2 \left\{ (\hat{\theta} - \theta) \frac{\partial \ln[f_{\mathbf{x};\theta}(\mathbf{x}; \theta)]}{\partial \theta} \right\} = 1 \quad (3.6.16)$$

The first term on the left-hand side is the variance of the estimator $\hat{\theta}$ since it is unbiased. Hence

$$\text{var}(\hat{\theta}) \geq \frac{1}{E\{[\partial \ln f_{\mathbf{x};\theta}(\mathbf{x}; \theta)/\partial \theta]^2\}} \quad (3.6.17)$$

which is one form of the CRLB and can also be expressed as

$$\text{var}(\hat{\theta}) \geq -\frac{1}{E\{\partial^2 \ln f_{\mathbf{x};\theta}(\mathbf{x}; \theta)/\partial \theta^2\}} \quad (3.6.18)$$

The function $\ln f_{\mathbf{x};\theta}(\mathbf{x}; \theta)$ is called the *log likelihood function* of θ . The CRLB expresses the minimum error variance of any estimator $\hat{\theta}$ of θ in terms of the joint density $f_{\mathbf{x};\theta}(\mathbf{x}; \theta)$

of observations. Hence every unbiased estimator must have a variance greater than a certain number. An unbiased estimate that satisfies the CRLB (3.6.18) with equality is called an *efficient* estimate. If such an estimate exists, then it can be obtained as a unique solution to the likelihood equation

$$\frac{\partial \ln f_{\mathbf{x};\theta}(\mathbf{x}; \theta)}{\partial \theta} = 0 \quad (3.6.19)$$

The solution of (3.6.19) is called the *maximum likelihood (ML)* estimate. Note that if the efficient estimate does not exist, then the ML estimate will not achieve the lower bound and hence it is difficult to ascertain how closely the variance of any estimate will approach the bound. The CRLB can be generalized to handle the estimation of vector parameters (Therrien 1992).

Consistency of estimator. If the MSE of the estimator can be made to approach zero as the sample size N becomes large, then from (3.6.7) both the bias and the variance will tend to zero. Then the sampling distribution will tend to concentrate about θ , and eventually as $N \rightarrow \infty$, the sampling distribution will become an impulse at θ . This is an important and desirable property, and the estimator that possesses it is called a *consistent* estimator.

Confidence interval. If we know the sampling distribution of an estimator, we can use the observations to compute an interval that has a specified probability of covering the unknown true parameter value. This interval is called a *confidence interval*, and the coverage probability is called the *confidence level*. When we interpret the meaning of confidence intervals, it is important to remember that it is the interval that is the random variable, and not the parameter. This concept will be explained in the sequel by means of specific examples.

3.6.2 Estimation of Mean

The natural estimator of the mean μ_x of a stationary sequence $x(n)$ from the observations $\{x(n)\}_0^{N-1}$ is the *sample mean*, given by

$$\hat{\mu}_x = \frac{1}{N} \sum_{n=0}^{N-1} x(n) \quad (3.6.20)$$

The estimate $\hat{\mu}_x$ is a random variable that depends on the number and values of the observations. Changing N or the set of observations will lead to another value for $\hat{\mu}_x$. Since the mean of the estimator is given by

$$E\{\hat{\mu}_x\} = \mu_x \quad (3.6.21)$$

the estimator $\hat{\mu}_x$ is unbiased. If $x(n) \sim WN(\mu_x, \sigma_x^2)$, we have

$$\text{var}(\hat{\mu}_x) = \frac{\sigma_x^2}{N} \quad (3.6.22)$$

because the samples of the process are uncorrelated random variables. This variance, which is a measure of the estimator's quality, increases if $x(n)$ is nonwhite.

Indeed, for a correlated random sequence, the variance of $\hat{\mu}_x$ is given by (see Problem 3.30)

$$\text{var}(\hat{\mu}_x) = N^{-1} \sum_{l=-N}^N \left(1 - \frac{|l|}{N}\right) \gamma_x(l) \leq N^{-1} \sum_{l=-N}^N |\gamma_x(l)| \quad (3.6.23)$$

where $\gamma_x(l)$ is the covariance sequence of $x(n)$. If $\gamma_x(l) \rightarrow 0$ as $l \rightarrow \infty$, then $\text{var}(\hat{\mu}_x) \rightarrow 0$ as $N \rightarrow \infty$ and hence $\hat{\mu}_x$ is a consistent estimator of μ_x . If $\sum_{l=-\infty}^{\infty} |\gamma_x(l)| < \infty$, then

from (3.6.23)

$$\lim_{N \rightarrow \infty} N \operatorname{var}(\hat{\mu}_x) = \lim_{N \rightarrow \infty} \sum_{l=-N}^N \left(1 - \frac{|l|}{N}\right) \gamma_x(l) = \sum_{l=-\infty}^{\infty} \gamma_x(l) \quad (3.6.24)$$

The expression for $\operatorname{var}(\hat{\mu}_x)$ in (3.6.23) can also be put in the form (see Problem 3.30)

$$\operatorname{var}(\hat{\mu}_x) = \frac{\sigma_x^2}{N} [1 + \Delta_N(\rho_x)] \quad (3.6.25)$$

where $\Delta_N(\rho_x) = 2 \sum_{l=1}^N \left(1 - \frac{l}{N}\right) \rho_x(l) \quad \rho_x(l) = \frac{\gamma_x(l)}{\sigma_x^2}$ (3.6.26)

When $\Delta_N(\rho_x) \geq 0$, the variance of the estimator increases as the amount of correlation among the samples of $x(n)$ increases. This implies that as the correlation increases, we need more samples to retain the quality of the estimate because each additional sample carries “less information.” For this reason the estimation of long-memory processes and processes with infinite variance is extremely difficult.

Sampling distribution. If we know the joint pdf of the random variables $\{x(n)\}_0^{N-1}$, we can determine, at least in principle, the pdf of $\hat{\mu}_x$. For example, if it is assumed that the observations are IID as $\mathcal{N}(\mu_x, \sigma_x^2)$ then from (3.6.21) and (3.6.22), it can be seen that $\hat{\mu}_x$ is normal with mean μ_x and variance σ_x^2/N , that is,

$$f_{\hat{\mu}_x}(\hat{\mu}_x) = \frac{1}{\sqrt{2\pi}(\sigma_x/\sqrt{N})} \exp\left[-\frac{1}{2}\left(\frac{\hat{\mu}_x - \mu_x}{\sigma_x/\sqrt{N}}\right)^2\right] \quad (3.6.27)$$

which is the sampling distribution of the mean. If N is large, then from the central limit theorem, the sampling distribution of the sample mean (3.6.27) is usually very close to the normal distribution, even if the individual distributions are not normal.

If we know the standard deviation σ_x , we can compute the probability

$$\Pr\left\{\mu_x - k \frac{\sigma_x}{\sqrt{N}} < \hat{\mu}_x < \mu_x + k \frac{\sigma_x}{\sqrt{N}}\right\} \quad (3.6.28)$$

that the random variable $\hat{\mu}_x$ is within a certain interval specified by two fixed quantities. A simple rearrangement of the above inequality leads to

$$\Pr\left\{\hat{\mu}_x - k \frac{\sigma_x}{\sqrt{N}} < \mu_x < \hat{\mu}_x + k \frac{\sigma_x}{\sqrt{N}}\right\} \quad (3.6.29)$$

which gives the probability that the fixed quantity μ_x lies between the two random variables $\hat{\mu}_x - k\sigma_x/\sqrt{N}$ and $\hat{\mu}_x + k\sigma_x/\sqrt{N}$. Hence (3.6.29) provides the probability that an interval with fixed length $2k\sigma_x/\sqrt{N}$ and randomly centered at the estimated mean includes the true mean. If we choose k so that the probability defined by (3.6.29) is equal to 0.95, the interval is known as the 95 percent confidence interval. To understand the meaning of this reasoning, we stress that for each set of measurements we compute a confidence interval that either contains or does not contain the true mean. However, if we repeat this process for a large number of observation sets, about 95 percent of the obtained confidence intervals will include the true mean. We stress that by no means does this imply that a confidence interval includes the true mean with probability 0.95.

If the variance σ_x^2 is unknown, then it has to be determined from the observations. This results in two modifications of (3.6.29). First, σ_x is replaced by

$$\hat{\sigma}_x^2 = \frac{1}{N-1} \sum_{n=0}^{N-1} [x(n) - \hat{\mu}_x]^2 \quad (3.6.30)$$

which implies that the center and the length of the confidence interval are different for each set of observations. Second, the random variable $(\hat{\mu}_x - \mu_x)/(\hat{\sigma}_x/\sqrt{N})$ is distributed according to *Student's t distribution with $v = N - 1$ degrees of freedom* (Parzen 1960), which tends to a Gaussian for large values of N . In these cases, the factor k in (3.6.29) is replaced by the appropriate value t of Student's distribution, using $N - 1$ degrees of freedom, for the desired level of confidence.

If the observations are normal but not IID, then from (3.6.25), the mean estimator $\hat{\mu}_x$ is normal with mean μ and variance $(\sigma_x^2/N)[1 + \Delta_N(\rho_x)]$. It is now easy to construct exact confidence intervals for $\hat{\mu}_x$ if $\rho_x(l)$ is known, and approximate confidence intervals if $\rho_x(l)$ is to be estimated from the observations. For large N , the variance $\text{var}(\hat{\mu}_x)$ can be approximated by

$$\begin{aligned}\text{var}(\hat{\mu}_x) &= \frac{\sigma_x^2}{N}[1 + \Delta_N(\rho_x)] \\ &\simeq \frac{\sigma_x^2}{N} \left[1 + 2 \sum_1^N \rho_x(l) \right] \\ &\triangleq \frac{v}{N} \quad v = \sigma_x^2 \left\{ 1 + 2 \sum_1^N \rho_x(l) \right\}\end{aligned}\tag{3.6.31}$$

and hence an approximate 95 percent confidence interval for $\hat{\mu}_x$ is given by

$$\left(\hat{\mu}_x - 1.96 \sqrt{\frac{v}{N}}, \hat{\mu}_x + 1.96 \sqrt{v/N} \right)\tag{3.6.32}$$

This means that, on average, the above interval will enclose the true value μ_x on 95 percent of occasions. For many practical random processes (especially those modeled as ARMA processes), the result in (3.6.32) is a good approximation.

EXAMPLE 3.6.1. Consider the AR(1) process

$$x(n) = ax(n-1) + w(n) \quad -1 < a < 1$$

where $w(n) \sim \text{WN}(0, \sigma_w^2)$. We wish to compute the variance of the mean estimator $\hat{\mu}_x$ of the process $x(n)$. Using straightforward calculations, we obtain

$$\mu_x = 0 \quad \sigma_x^2 = \frac{\sigma_w^2}{1-a^2} \quad \text{and} \quad \rho_x(l) = a^{|l|}$$

From (3.6.26) we evaluate the term

$$\Delta_N(\rho) = \frac{2a}{1-a} \left[1 - \frac{1}{N(1-a)} + \frac{a^N}{N(1-a)} \right] \simeq \frac{2a}{1-a} \quad \text{for } N \gg 1$$

When $a \rightarrow 1$, that is, when the dependence between the signal samples increases, then the factor $\Delta_N(\rho)$ takes large values and the quality of estimator decreases drastically. Similar conclusions can be drawn using the approximation (3.6.31)

$$v = \left(1 + 2 \sum_1^\infty a^l \right) \frac{\sigma_w^2}{1-a^2} = \frac{\sigma_w^2}{(1-a)^2}$$

We will next verify these results using two Monte Carlo simulations: one for $a = 0.9$, which represents high correlations among samples, and the other for $a = 0.1$. Using a Gaussian pseudorandom number generator with mean 0 and variance $\sigma_w^2 = 1$, we generated $N = 100$ samples of the AR(1) process $x(n)$. Using v in (3.6.31) and (3.6.32), we next computed the confidence intervals. For $a = 0.9$, we obtain

$$v = 100 \quad \text{and} \quad \text{confidence interval: } (\hat{\mu}_x - 1.96, \hat{\mu}_x + 1.96)$$

and for $a = 0.1$, we obtain

$$v = 1.2345 \quad \text{and} \quad \text{confidence interval: } (\hat{\mu}_x - 0.2178, \hat{\mu}_x + 0.2178)$$

Clearly, when the dependence between signal samples increases, the quality of the estimator decreases drastically and hence the confidence interval is wider. To have the same confidence interval, we should increase the number of samples N .

We next estimate the mean, using (3.6.20), and we repeat the experiment 10,000 times. Figure 3.14 shows histograms of the computed means for $a = 0.9$ and $a = 0.1$. The confidence intervals are also shown as dotted lines around the true mean. The histograms are approximately Gaussian in shape. The histogram for the high-correlation case is wider than that for the low-correlation case, which is to be expected. The 95 percent confidence intervals also indicate that very few estimates are outside the interval.

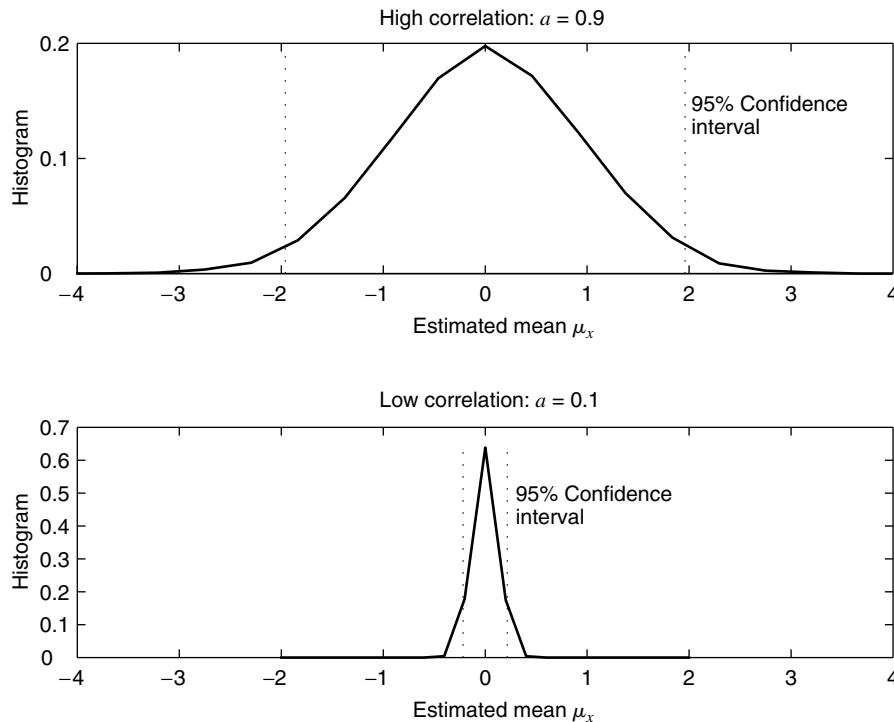


FIGURE 3.14
Histograms of mean estimates in Example 3.6.1.

3.6.3 Estimation of Variance

The natural estimator of the variance σ_x^2 of a stationary sequence $x(n)$ from the observations $\{x(n)\}_0^{N-1}$ is the *sample variance*, given by

$$\hat{\sigma}_x^2 \triangleq \frac{1}{N} \sum_{n=0}^{N-1} \{x(n) - \hat{\mu}_x\}^2 \quad (3.6.33)$$

By using the mean estimate $\hat{\mu}_x$ from (3.6.20), the mean of the variance estimator can be shown to equal (see Problem 3.31)

$$E\{\hat{\sigma}_x^2\} = \sigma_x^2 - \text{var}(\hat{\mu}_x) = \sigma_x^2 - \frac{1}{N} \sum_{l=-N}^N \left(1 - \frac{|l|}{N}\right) \gamma_x(l) \quad (3.6.34)$$

If the sequence $x(n)$ is uncorrelated, then

$$E\{\hat{\sigma}_x^2\} = \sigma_x^2 - \frac{\sigma_x^2}{N} = \left(\frac{N-1}{N}\right) \sigma_x^2 \quad (3.6.35)$$

From (3.6.34) or (3.6.35), it is obvious that the estimator in (3.6.33) is biased. If $\gamma_x(l) \rightarrow 0$ as $l \rightarrow \infty$, then $\text{var}(\hat{\mu}_x) \rightarrow 0$ as $N \rightarrow \infty$ and hence $\hat{\sigma}_x^2$ is an asymptotically unbiased estimator of σ_x^2 . In practical applications, the variance estimate is nearly unbiased for large N . Note that if we use the actual mean μ_x in (3.6.33), then the resulting estimator is unbiased.

The general expression for the variance of the variance estimator is fairly complicated and requires higher-order moments. It can be shown that for either estimators

$$\text{var}(\hat{\sigma}_x^2) \approx \frac{\gamma_x^{(4)}}{N} \quad \text{for large } N \quad (3.6.36)$$

where $\gamma_x^{(4)}$ is the fourth central moment of $x(n)$ (Brockwell and Davis 1991). Thus the estimator in (3.6.33) is also consistent.

Sampling distribution. In the case of the mean estimator, the sampling distribution involved the distribution of sums of random variables. The variance estimator involves the sum of the squares of random variables, for which the sampling distribution computation is complicated. For example, if there are N independent measurements from an $\mathcal{N}(0, 1)$ distribution, then the sampling distribution of the random variable

$$\chi_N^2 = x_1^2 + x_2^2 + \cdots + x_N^2 \quad (3.6.37)$$

is given by the *chi-squared distribution with N degrees of freedom*. The general form of χ_N^2 with v degrees of freedom is

$$f_{\chi_v^2}(x) = \frac{1}{2^{v/2}\Gamma(v/2)} x^{v/2-1} \exp\left(-\frac{x}{2}\right) \quad 0 \leq x \leq \infty \quad (3.6.38)$$

where $\Gamma(v/2) = \int_0^\infty e^{-t} t^{v/2-1} dt$ is the gamma function with argument $v/2$.

For the variance estimator in (3.6.33), it can be shown (Parzen 1960) that $N\hat{\sigma}_x^2$ is distributed as chi squared with $v = N - 1$ degrees of freedom. This means that, for any set of N observations, there will only be $N - 1$ independent deviations $\{x(n) - \hat{\mu}_x\}$, since their sum is zero from the definition of the mean. Assuming that the observations are $\mathcal{N}(\mu, \sigma^2)$, the random variables $x(n)/\sigma$ will be $\mathcal{N}(\mu/\sigma, 1)$ and hence the random variable

$$\frac{N\hat{\sigma}_x^2}{\sigma^2} = \frac{1}{\sigma^2} \sum_{n=0}^{N-1} [x(n) - \hat{\mu}_x]^2 \quad (3.6.39)$$

will be chi squared distributed with $v = N - 1$. Therefore, using values of the chi-squared distribution, confidence intervals for the variance estimator can be computed. In particular, since $N\hat{\sigma}_x^2/\sigma^2$ is distributed as χ_v^2 , the 95 percent limits of the form

$$\Pr\left\{\chi_v\left(\frac{0.05}{2}\right) < N\hat{\sigma}_x^2/\sigma^2 \leq \chi_v\left(1 - \frac{0.05}{2}\right)\right\} = 0.95 \quad (3.6.40)$$

can be obtained from chi-squared tables (Fisher and Yates 1938). By rearranging (3.6.40), the random variable $\sigma^2/\hat{\sigma}_x^2$ satisfies

$$\Pr\left\{\frac{N}{\chi_v(0.975)} < \frac{\sigma^2}{\hat{\sigma}_x^2} \leq \frac{N}{\chi_v(0.025)}\right\} = 0.95 \quad (3.6.41)$$

Using $l_1 = N/\chi_v(0.975)$ and $l_2 = N/\chi_v(0.025)$, we see that (3.6.41) implies that

$$\Pr\{l_2\hat{\sigma}_x^2 \geq \sigma^2 \text{ and } l_1\hat{\sigma}_x^2 < \sigma^2\} = 0.95 \quad (3.6.42)$$

Thus the 95 percent confidence interval based on the estimate $\hat{\sigma}_x^2$ is $(l_1\hat{\sigma}_x^2, l_2\hat{\sigma}_x^2)$. Note that this interval is sensitive to the validity of the normal assumption of random variables leading to (3.6.39). This is not the case for the confidence intervals for the mean estimates

because, thanks to the central limit theorem, the computation of the interval can be based on the normal assumption.

EXAMPLE 3.6.2. Consider again the AR(1) process given in Example 3.6.1:

$$x(n) = ax(n-1) + w(n) \quad -1 < a < 1 \quad w(n) \sim WN(0, 1)$$

with $\mu_x = 0 \quad \sigma_x^2 = \frac{\sigma_w^2}{1-a^2} \quad \text{and} \quad \rho_x(l) = a^{|l|}$ (3.6.43)

We wish to compute the mean of the variance estimator $\hat{\sigma}_x^2$ of the process $x(n)$. From (3.6.34), we obtain

$$E[\hat{\sigma}_x^2] = \sigma_x^2 \left[1 - \frac{1}{N} \sum_{l=-N}^N \left(1 - \frac{|l|}{N} \right) a^{|l|} \right] \quad (3.6.44)$$

When $a \rightarrow 1$, that is, when the dependence between the signal samples increases, the mean of the estimate deviates significantly from the true value σ_x^2 and the quality of the estimator decreases drastically. For small dependence, the mean is very close to σ_x^2 . These conclusions can be verified using two Monte Carlo simulations as before: one for $a = 0.9$, which represents high correlations among samples, and the other for $a = 0.1$. Using a Gaussian pseudorandom number generator with mean 0 and unit variance, we generated $N = 100$ samples of the AR(1) process $x(n)$. The computed parameters according to (3.6.43) and (3.6.44) are

$$\begin{aligned} a = 0.9: \quad \sigma_x^2 &= 5.2632 \quad E\{\hat{\sigma}_x^2\} = 4.3579 \\ a = 0.1: \quad \sigma_x^2 &= 1.0101 \quad E\{\hat{\sigma}_x^2\} = 0.9978 \end{aligned}$$

We next estimate the variance by using (3.6.33) and repeat the experiment 10,000 times. Figure 3.15 shows histograms of computed variances for $a = 0.9$ and for $a = 0.1$. The computed

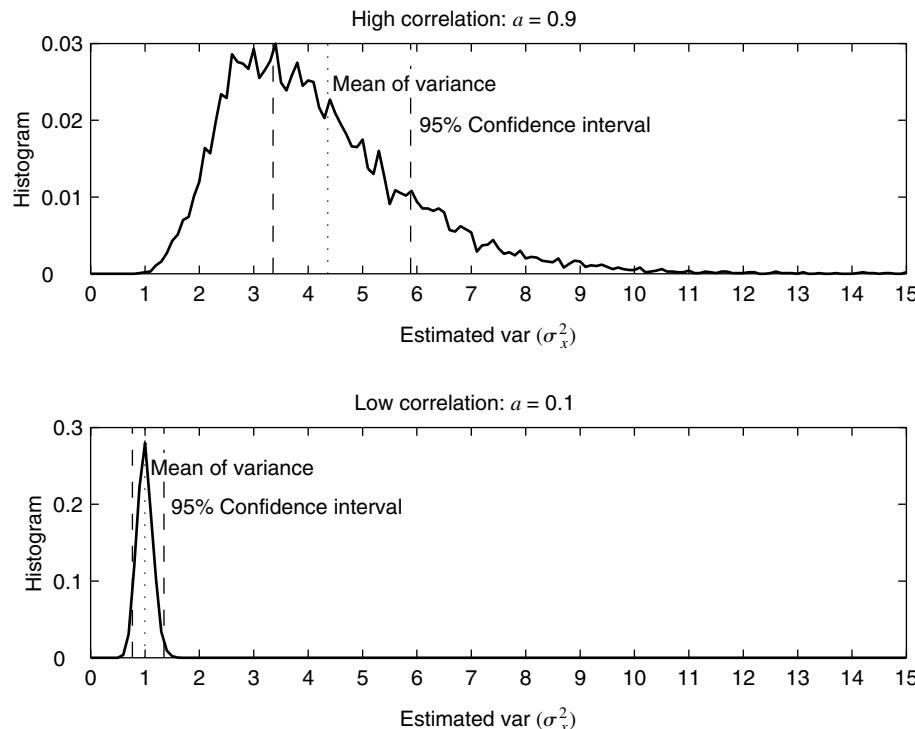


FIGURE 3.15
Histograms of variance estimates in Example 3.6.2.

means of the variance estimates are also shown as dotted lines. Clearly, the histogram is much wider for the high-correlation case and much narrower (almost symmetric and Gaussian) for the low-correlation case.

The 95 percent confidence intervals are given by $(l_1 \hat{\sigma}_x^2, l_2 \hat{\sigma}_x^2)$, where $l_1 = N/\chi_v(0.975)$ and $l_2 = N/\chi_v(0.025)$. The values of l_1 and l_2 are obtained from the chi-squared distribution curves (Jenkins and Watts 1968). For $N = 100$, $l_1 = 0.77$ and $l_2 = 1.35$; hence the 95 percent confidence intervals for σ_x^2 are

$$(0.77\hat{\sigma}_x^2, 1.35\hat{\sigma}_x^2)$$

also shown as dashed lines around the mean value $E\{\hat{\sigma}_x^2\}$. The confidence interval for the high-correlation case, $a = 0.9$, does not appear to be a good interval, which implies that the approximation leading to (3.6.42) is not a good one for this case. Such is not the case for $a = 0.1$.

3.7 SUMMARY

In this chapter we provided an overview of the basic theory of discrete-time stochastic processes. We began with the notion of a random variable as a mapping from the abstract probability space to the real space, extended it to random vectors as a collection of random variables, and introduced discrete-time stochastic processes as an indexed family (or time series) of random variables. A complete probabilistic description of these random objects requires the knowledge of joint distribution or density functions, which is difficult to acquire except in simple cases. Therefore, the emphasis was placed on description using joint moments of distributions, and, in particular, the emphasis was placed on the second-order moments, which are relatively easy to estimate or compute in practice.

We defined the mean and the variance to describe random variables, and we provided three useful models of random variables. For random vector description, we defined the mean vector and the autocorrelation matrix. Linear transformations of random vectors were discussed, using densities and correlation matrices. The normal random vector was then introduced as a useful model of a random vector. A particularly simple linear transformation, namely, the sum of independent random variables, was used to introduce random variables with stable and infinitely divisible distributions. To describe stochastic processes, we proceeded to define mean and autocorrelation sequences. In many applications, the concept of stationary of random processes is a useful one that reduces the computational complexity. Assuming time invariance on the first two moments, we defined a wide-sense stationary (WSS) process in which the mean is a constant and correlation between random variables at two distinct times is a function of time difference or lag. The rest of the chapter was devoted to the analysis of WSS processes.

A stochastic process is generally observed in practice as a single sample function (a speech signal or a radar signal) from which it is necessary to estimate the first- and the second-order moments. This requires the notion of ergodicity, which provides a framework for the computation of statistical averages using time averages over a single realization. Although this framework requires theoretical results using mean square convergence, we provided a simple approach of using appropriate time averages. An important random signal characteristic called variability was introduced. The WSS processes were then described in the frequency domain using the power spectral density function, which is a physical quantity that can be measured in practice. Some random processes exhibiting flat spectral envelopes were analyzed including one of white noise. Since random processes are generally processed using linear systems, we described linear system operations with random inputs in both the time and frequency domains.

The properties of correlation matrices and sequences play an important role in filtering and estimation theory and were discussed in detail, including eigenanalysis. Another important random signal characteristic called memory was also introduced. Stationary random

signals were modeled using autocorrelation matrices, and the relationship between spectral flatness and eigenvalue spread was explored. These properties were used in an alternate representation of random vectors as well as processes using uncorrelated components which were based on diagonalization and triangularization of correlation matrices. This resulted in the discrete KL transform and KL expansion. These concepts will also be useful in later chapters on optimal filtering and adaptive filtering.

Finally, we concluded this chapter with the introduction of elementary estimation theory. After discussion of properties of estimators, two important estimators of mean and variance were treated in detail along with their sampling distributions. These topics will be useful in many subsequent chapters.

PROBLEMS

3.1 The exponential density function is given by

$$f_x(x) = \frac{1}{a} e^{-x/a} u(x) \quad (\text{P.1})$$

where a is a parameter and $u(x)$ is a unit step function.

- (a) Plot the density function for $a = 1$.
- (b) Determine the mean, variance, skewness, and kurtosis of the Rayleigh random variable with $a = 1$. Comment on the significance of these moments in terms of the shape of the density function.
- (c) Determine the characteristic function of the exponential pdf.

3.2 The Rayleigh density function is given by

$$f_x(x) = \frac{x}{\sigma^2} e^{-x^2/(2\sigma^2)} u(x) \quad (\text{P.2})$$

where σ is a parameter and $u(x)$ is a unit step function. Repeat Problem 3.1 for $\sigma = 1$.

3.3 Using the binomial expansion of $\{x(\zeta) - \mu_x\}^m$, show that the m th central moment is given by

$$M_m^{(x)} = \sum_{k=0}^m \binom{m}{k} (-1)^k \mu_x^k \xi_{m-k}^{(x)}$$

Similarly, show that

$$\xi_m^{(x)} = \sum_{k=0}^m \binom{m}{k} \mu_x^k M_{m-k}^{(x)}$$

3.4 Consider a zero-mean random variable $x(\zeta)$. Using (3.1.26), show that the first four cumulants of $x(\zeta)$ are given by (3.1.28) through (3.1.31).

3.5 A random vector $\mathbf{x}(\zeta) = [x_1(\zeta) \ x_2(\zeta)]^T$ has mean vector $\boldsymbol{\mu}_{\mathbf{x}} = [1 \ 2]^T$ and covariance matrix

$$\boldsymbol{\Gamma}_{\mathbf{x}} = \begin{bmatrix} 4 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

This vector is transformed to another random vector $\mathbf{y}(\zeta)$ by the following linear transformation:

$$\begin{bmatrix} y_1(\zeta) \\ y_2(\zeta) \\ y_3(\zeta) \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ -1 & 2 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x_1(\zeta) \\ x_2(\zeta) \end{bmatrix}$$

Determine (a) the mean vector $\boldsymbol{\mu}_{\mathbf{y}}$, (b) the autocovariance matrix $\boldsymbol{\Gamma}_{\mathbf{y}}$, and (c) the cross-correlation matrix $\mathbf{R}_{\mathbf{xy}}$.

- 3.6** Using the moment generating function, show that the linear transformation of a Gaussian random vector is also Gaussian.

- 3.7** Let $\{x_k(\zeta)\}_{k=1}^4$ be four IID random variables with exponential distribution (P.1) with $a = 1$. Let

$$y_k(\zeta) = \sum_{l=1}^k x_l(\zeta) \quad 1 \leq k \leq 4$$

- (a) Determine and plot the pdf of $y_2(\zeta)$.
- (b) Determine and plot the pdf of $y_3(\zeta)$.
- (c) Determine and plot the pdf of $y_4(\zeta)$.
- (d) Compare the pdf of $y_4(\zeta)$ with that of the Gaussian density.

- 3.8** For each of the following, determine whether the random process is (1) WSS or (2) m.s. ergodic in the mean.

- (a) $X(t) = A$, where A is a random variable uniformly distributed between 0 and 1.
- (b) $X_n = A \cos \omega_0 n$, where A is a Gaussian random variable with mean 0 and variance 1.
- (c) A Bernoulli process with $\Pr[X_n = 1] = p$ and $\Pr[X_n = -1] = 1 - p$.

- 3.9** Consider the harmonic process $x(n)$ defined in (3.3.50).

- (a) Determine the mean of $x(n)$.
- (b) Show that the autocorrelation sequence is given by

$$r_x(l) = \frac{1}{2} \sum_{k=1}^M |c_k|^2 \cos \omega_k l \quad -\infty < l < \infty$$

- 3.10** Suppose that the random variables ϕ_k in the real-valued harmonic process model are distributed with a pdf $f_{\phi_k}(\phi_k) = (1 + \cos \phi_k)/(2\pi)$, $-\pi \leq \phi_k \leq \pi$. Is the resulting stochastic process stationary?

- 3.11** A stationary random sequence $x(n)$ with mean $\mu_x = 4$ and autocovariance

$$\gamma_x(n) = \begin{cases} 4 - |n| & |n| \leq 3 \\ 0 & \text{otherwise} \end{cases}$$

is applied as an input to a linear shift-invariant (LSI) system whose impulse response $h(n)$ is

$$h(n) = u(n) - u(n - 4)$$

where $u(n)$ is a unit step sequence. The output of this system is another random sequence $y(n)$. Determine (a) the mean sequence $\mu_y(n)$, (b) the cross-covariance $\gamma_{xy}(n_1, n_2)$ between $x(n_1)$ and $y(n_2)$, and (c) the autocovariance $\gamma_y(n_1, n_2)$ of the output process $y(n)$.

- 3.12** A causal LTI system, which is described by the difference equation

$$y(n) = \frac{1}{2}y(n - 1) + x(n) + \frac{1}{3}x(n - 1)$$

is driven by a zero-mean WSS process with autocorrelation $r_x(l) = 0.5^{|l|}$.

- (a) Determine the PSD and the autocorrelation of the output sequence $y(n)$.
- (b) Determine the cross-correlation $r_{xy}(l)$ and cross-PSD $R_{xy}(e^{j\omega})$ between the input and output signals.

- 3.13** A WSS process with PSD $R_x(e^{j\omega}) = 1/(1.64 + 1.6 \cos \omega)$ is applied to a causal system described by the following difference equation

$$y(n) = 0.6y(n - 1) + x(n) + 1.25x(n - 1)$$

Compute (a) the PSD of the output and (b) the cross-PSD $R_{xy}(e^{j\omega})$ between input and output.

$$(a) \quad \mathbf{R}_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad (b) \quad \mathbf{R}_2 = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & 1 \end{bmatrix}$$

$$(c) \quad \mathbf{R}_3 = \begin{bmatrix} 1 & 1-j \\ 1+j & 1 \end{bmatrix} \quad (d) \quad \mathbf{R}_4 = \begin{bmatrix} 1 & \frac{1}{2} & 1 \\ \frac{1}{2} & 2 & \frac{1}{2} \\ 1 & 1 & 1 \end{bmatrix}$$

3.15 Consider a normal random vector $\mathbf{x}(\zeta)$ with components that are mutually uncorrelated, that is, $\rho_{ij} = 0$. Show that (a) the covariance matrix $\Gamma_{\mathbf{x}}$ is diagonal and (b) the components of $\mathbf{x}(\zeta)$ are mutually independent.

3.16 Show that if a real, symmetric, and nonnegative definite matrix \mathbf{R} has eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_M$, then the matrix \mathbf{R}^k has eigenvalues $\lambda_1^k, \lambda_2^k, \dots, \lambda_M^k$.

3.17 Prove that the trace of \mathbf{R} is given by

$$\text{tr } \mathbf{R} = \sum \lambda_i$$

3.18 Prove that the determinant of \mathbf{R} is given by

$$\det \mathbf{R} = |\mathbf{R}| = \prod \lambda_i = |\Lambda|$$

3.19 Show that the determinants of \mathbf{R} and Γ are related by

$$\det \mathbf{R} = \det \Gamma (1 + \mu^H \Gamma \mu)$$

3.20 Let $\mathbf{R}_{\mathbf{x}}$ be the correlation matrix of the vector $\mathbf{x} = [x(0) \ x(2) \ x(3)]^T$, where $x(n)$ is a zero-mean WSS process.

- (a) Check whether the matrix $\mathbf{R}_{\mathbf{x}}$ is Hermitian, Toeplitz, and nonnegative definite.
- (b) If we know the matrix $\mathbf{R}_{\mathbf{x}}$, can we determine the correlation matrix of the vector $\bar{\mathbf{x}} = [x(0) \ x(1) \ x(2) \ x(3)]^T$?

3.21 Using the nonnegativeness of $E\{|x(n+l) \pm x(n)|^2\}$, show that $r_x(0) \geq |r_x(l)|$ for all l .

3.22 Show that $r_x(l)$ is nonnegative definite, that is,

$$\sum_{l=1}^M \sum_{k=1}^M a_l r_x(l-k) a_k^* \geq 0 \quad \forall M, \quad \forall a_1, \dots, a_M$$

3.23 Let $x(n)$ be a random process generated by the AP(1) system

$$x(n) = \alpha x(n-1) + w(n) \quad n \geq 0 \quad x(-1) = 0$$

where $w(n)$ is an IID($0, \sigma_w^2$) process.

- (a) Determine the autocorrelation $r_x(n_1, n_2)$ function.
- (b) Show that $r_x(n_1, n_2)$ asymptotically approaches $r_x(n_1 - n_2)$, that is, it becomes shift-invariant.

3.24 Let \mathbf{x} be a random vector with mean $\mu_{\mathbf{x}}$ and autocorrelation $\mathbf{R}_{\mathbf{x}}$.

- (a) Show that $\mathbf{y} = \mathbf{Q}^T \mathbf{x}$ transforms \mathbf{x} to an uncorrelated component vector \mathbf{y} if \mathbf{Q} is the eigenmatrix of $\mathbf{R}_{\mathbf{x}}$.
- (b) Comment on the geometric interpretation of this transformation.

3.25 The mean and the covariance of a Gaussian random vector \mathbf{x} are given by, respectively,

$$\boldsymbol{\mu}_{\mathbf{x}} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Gamma}_{\mathbf{x}} = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix}$$

Plot the 1σ , 2σ , and 3σ concentration ellipses representing the contours of the density function in the (x_1, x_2) plane. *Hints:* The radius of an ellipse with major axis a (along x_1) and minor axis $b < a$ (along x_2) is given by

$$r^2 = \frac{a^2 b^2}{a^2 \sin^2 \theta + b^2 \cos^2 \theta}$$

where $0 \leq \theta \leq 2\pi$. Compute the 1σ ellipse specified by $a = \sqrt{\lambda_1}$ and $b = \sqrt{\lambda_2}$ and then rotate and translate each point $\mathbf{x}^{(i)} = [x_1^{(i)} \ x_2^{(i)}]^T$ using the transformation $\mathbf{w}^{(i)} = \mathbf{Q}_x \mathbf{x}^{(i)} + \boldsymbol{\mu}_x$.

3.26 Consider the process $x(n) = ax(n-1) + w(n)$, where $w(n) \sim WN(0, \sigma_w^2)$.

(a) Show that the $M \times M$ correlation matrix of the process is symmetric Toeplitz and is given by

$$\mathbf{R}_x = \frac{\sigma_w^2}{1-a^2} \begin{bmatrix} 1 & a & \cdots & a^{m-1} \\ a & 1 & \cdots & a^{m-2} \\ \vdots & \vdots & \ddots & \vdots \\ a^{m-1} & a^{m-2} & \cdots & 1 \end{bmatrix}$$

(b) Verify that

$$\mathbf{R}_x^{-1} = \frac{1}{\sigma_w^2} \begin{bmatrix} 1 & -a & 0 & \cdots & 0 \\ -a & 1+a^2 & -a & \cdots & 0 \\ 0 & -a & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & 1+a^2 & -a \\ 0 & 0 & \cdots & -a & 1 \end{bmatrix}$$

(c) Show that if

$$\mathbf{L}_x = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -a & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & -a & 1 \end{bmatrix}$$

then $\mathbf{L}_x^T \mathbf{R}_x \mathbf{L}_x = (1-a^2)\mathbf{I}$.

(d) For $\sigma_w^2 = 1$, $a = 0.95$, and $M = 8$ compute the DKLT and the DFT.

(e) Plot the eigenvalues of each transform in the same graph of the PSD of the process. Explain your findings.

(f) Plot the eigenvectors of each transform and compare the results.

(g) Repeat parts (e) and (f) for $M = 16$ and $M = 32$. Explain the obtained results.

(h) Repeat parts (e) to (g) for $a = 0.5$ and compare with the results obtained for $a = 0.95$.

3.27 Determine three different innovations representations of a zero-mean random vector \mathbf{x} with correlation matrix

$$\mathbf{R}_x = \begin{bmatrix} 1 & \frac{1}{4} \\ \frac{1}{4} & 1 \end{bmatrix}$$

3.28 Verify that the eigenvalues and eigenvectors of the $M \times M$ correlation matrix of the process $x(n) = w(n) + bw(n-1)$, where $w(n) \sim WN(0, \sigma_w^2)$ are given by $\lambda_k = R_x(e^{j\omega_k})$, $q_n^{(k)} = \sin \omega_k n$, $\omega_k = \pi k / (M+1)$, where $k = 1, 2, \dots, M$, (a) analytically and (b) numerically for $\sigma_w^2 = 1$ and $M = 8$. *Hint:* Plot the eigenvalues on the same graph with the PSD.

3.29 Consider the process $x(n) = w(n) + bw(n - 1)$.

147

PROBLEMS

(a) Compute the DKLT for $M = 3$.

(b) Show that the variances of the DKLT coefficients are $\sigma_x^2(1 + \sqrt{2}b)$, σ_x^2 , and $\sigma_x^2(1 - \sqrt{2}b)$.

3.30 Let $x(n)$ be a stationary random process with mean μ_x and covariance $\gamma_x(l)$. Let $\hat{\mu}_x = 1/N \sum_{n=0}^{N-1} x(n)$ be the sample mean from the observations $\{x(n)\}_{n=0}^{N-1}$.

(a) Show that the variance of $\hat{\mu}_x$ is given by

$$\text{var}(\hat{\mu}_x) = N^{-1} \sum_{l=-N}^N \left(1 - \frac{|l|}{N}\right) \gamma_x(l) \leq N^{-1} \sum_{l=-N}^N |\gamma_x(l)| \quad (\text{P.3})$$

(b) Show that the above result (P.3) can be expressed as

$$\text{var}(\hat{\mu}_x) = \frac{\sigma_x^2}{N} [1 + \Delta_N(\rho_x)] \quad (\text{P.4})$$

$$\text{where } \Delta_N(\rho_x) = 2 \sum_{l=1}^N \left(1 - \frac{l}{N}\right) \rho_x(l) \quad \rho_x(l) = \frac{\gamma_x(l)}{\sigma_x^2}$$

(c) Show that (P.3) reduces to $\text{var}(\hat{\mu}_x) = \sigma_x^2/N$ for a WN(μ_x, σ_x^2) process.

3.31 Let $x(n)$ be a stationary random process with mean μ_x , variance σ_x^2 , and covariance $\gamma_x(l)$. Let

$$\hat{\sigma}_x^2 \triangleq \frac{1}{N} \sum_{n=0}^{N-1} [x(n) - \hat{\mu}_x]^2$$

be the sample variance from the observations $\{x(n)\}_{n=0}^{N-1}$.

(a) Show that the mean of $\hat{\sigma}_x^2$ is given by

$$E\{\hat{\sigma}_x^2\} = \sigma_x^2 - \text{var}(\hat{\mu}_x) = \sigma_x^2 - \frac{1}{N} \sum_{l=-N}^N \left(1 - \frac{|l|}{N}\right) \gamma_x(l)$$

(b) Show that the above result reduces to $E\{\hat{\sigma}^2\} = (N - 1)\sigma_x^2/N$ for a WN(μ_x, σ_x^2) process.

3.32 The Cauchy distribution with mean μ is given by

$$f_x(x) = \frac{1}{\pi} \frac{1}{1 + (x - \mu)^2} \quad -\infty < x < \infty$$

Let $\{x_k(\zeta)\}_{k=k}^N$ be N IID random variables with the above distribution. Consider the mean estimator based on $\{x_k(\zeta)\}_{k=k}^N$

$$\hat{\mu}(\zeta) = \frac{1}{N} \sum_{k=1}^N x_k(\zeta)$$

Determine whether $\hat{\mu}(\zeta)$ is a consistent estimator of μ .

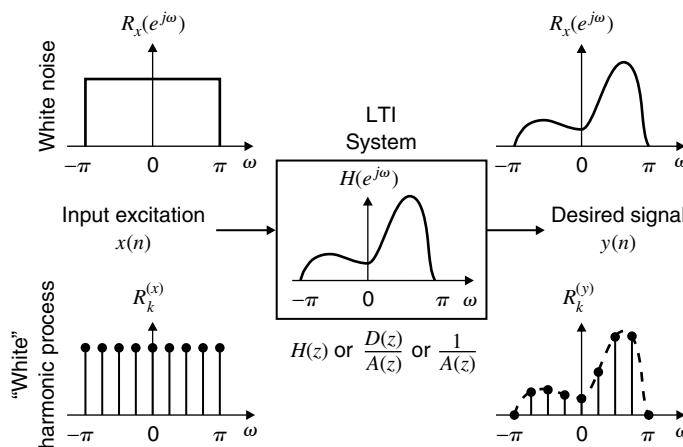
Linear Signal Models

In this chapter we introduce and analyze the properties of a special class of stationary random sequences that are obtained by driving a linear, time-invariant system with white noise. We focus on filters having a system function that is rational, that is, the ratio of two polynomials. The power spectral density of the resulting process is also rational, and its shape is completely determined by the filter coefficients. We will use the term *pole-zero* models when we want to emphasize the system viewpoint and the term *autoregressive moving-average* models to refer to the resulting random sequences. The latter term is not appropriate when the input is a harmonic process or a deterministic signal with a flat spectral envelope. We discuss the impulse response, autocorrelation, power spectrum, partial autocorrelation, and cepstrum of all-pole, all-zero, and pole-zero models. We express all these quantities in terms of the model coefficients and develop procedures to convert from one parameter set to another. Low-order models are studied in detail, because they are easy to analyze analytically and provide insight into the behavior and properties of higher-order models. An understanding of the correlation and spectral properties of a signal model is very important for the selection of the appropriate model in practical applications. Finally, we investigate a special case of pole-zero models with one or more unit poles. Pole-zero models are widely used for the modeling of stationary signals with short memory whereas models with unit poles are useful for the modeling of certain nonstationarity processes with trends.

4.1 INTRODUCTION

In Chapter 3 we defined and studied random processes as a mathematical tool to analyze random signals. In practice, we also need to generate random signals that possess certain known, second-order characteristics, or we need to describe observed signals in terms of the parameters of known random processes.

The simplest random signal model is the wide sense stationary white noise sequence $w(n) \sim WN(0, \sigma_w^2)$ that has uncorrelated samples and a flat PSD. It is also easy to generate in practice by using simple algorithms. If we filter white noise with a stable LTI filter, we can obtain random signals with almost any arbitrary aperiodic correlation structure or continuous PSD. If we wish to generate a random signal with a line PSD using the previous approach, we need an LTI filter with “line” frequency response; that is, we need an oscillator. Unfortunately, such a system is not *stable*, and its output cannot be stationary. Fortunately, random signals with line PSDs can be easily generated by using the *harmonic process* model (linear combination of sinusoidal sequences with statistically independent random phases) discussed in Section 3.3.6. Figure 4.1 illustrates the filtering of white noise and “white” (flat spectral envelope) harmonic process by an LTI filter. Signal models with mixed PSDs can be obtained by combining the above two models, a process justified by a powerful result known as the *Wold decomposition*.

**FIGURE 4.1**

Signal models with continuous and discrete (line) power spectrum densities.

When the LTI filter is specified by its impulse response, we have a *nonparametric* signal model because there is no restriction regarding the form of the model and the number of parameters is infinite. However, if we specify the filter by a finite-order rational system function, we have a *parametric* signal model described by a finite number of parameters. We focus on parametric models because they are simpler to deal with in practical applications. The two major topics we address in this chapter are (1) the derivation of the second-order moments of AP, AZ, and PZ models, given the coefficients of their system function, and (2) the design of an AP, AZ, or PZ system that produces a random signal with a given autocorrelation sequence or PSD function. The second problem is known as *signal modeling* and theoretically is equivalent to the spectral factorization procedure developed in Section 2.4.4. The modeling of harmonic processes is theoretically straightforward and does not require the use of a linear filter to change the amplitude of the spectral lines. The challenging problem in this case is the identification of the filter by observing its response to a harmonic process with a flat PSD. The modeling problem for continuous PSDs has a solution, at least in principle, for every regular random sequence.

In practical applications, the second-order moments of the signal to be modeled are not known a priori and have to be estimated from a set of signal observations. This element introduces a new dimension and additional complications to the signal modeling problem, which are discussed in Chapter 9. In this chapter we primarily focus on parametric models that replicate the second-order properties (autocorrelation or PSD) of stationary random sequences. If the sequence is Gaussian, the model provides a complete statistical characterization. The characterization of non-Gaussian processes, which requires the use of higher-order moments, is discussed in Chapter 12.

4.1.1 Linear Nonparametric Signal Models

Consider a stable LTI system with impulse response $h(n)$ and input $w(n)$. The output $x(n)$ is given by the convolution summation

$$x(n) = \sum_{k=-\infty}^{\infty} h(k)w(n-k) \quad (4.1.1)$$

which is known as a *nonrecursive* system representation because the output is computed by linearly weighting samples of the input signal.

Linear random signal model. If the input $w(n)$ is a zero-mean white noise process with variance σ_w^2 , autocorrelation $r_w(l) = \sigma_w^2 \delta(l)$, and PSD $R_w(e^{j\omega}) = \sigma_w^2$, $-\pi < \omega \leq \pi$, then from Table 3.2 the autocorrelation, complex PSD, and PSD of the output $x(n)$ are given by, respectively,

$$r_x(l) = \sigma_w^2 \sum_{k=-\infty}^{\infty} h(k)h^*(k-l) = \sigma_w^2 r_h(l) \quad (4.1.2)$$

$$R_x(z) = \sigma_w^2 H(z)H^* \left(\frac{1}{z^*} \right) \quad (4.1.3)$$

$$R_x(e^{j\omega}) = \sigma_w^2 |H(e^{j\omega})|^2 = \sigma_w^2 R_h(e^{j\omega}) \quad (4.1.4)$$

We notice that when the input is a white noise process, the shape of the autocorrelation and the power spectrum (*second-order moments*) of the output signal are completely characterized by the system. We use the term *system-based signal model* to refer to the signal generated by a system with a white noise input. If the system is linear, we use the term *linear random signal model*. In the statistical literature, the resulting model is known as the *general linear process model*. However, we should mention that in some applications it is more appropriate to use a deterministic input with flat spectral envelope or a “white” harmonic process input.

Recursive representation. Suppose now that the inverse system $H_I(n) = 1/H(n)$ is *causal and stable*. If we assume, without any loss of generality, that $h(0) = 1$, then $h_I(n) = \mathcal{Z}^{-1}\{H_I(n)\}$ has $h_I(0) = 1$. Therefore the input $w(n)$ can be obtained by

$$w(n) = x(n) + \sum_{k=1}^{\infty} h_I(k)x(n-k) \quad (4.1.5)$$

Solving for $x(n)$, we obtain the following recursive representation for the output signal

$$x(n) = - \sum_{k=1}^{\infty} h_I(k)x(n-k) + w(n) \quad (4.1.6)$$

We use the term *recursive* representation to emphasize that the present value of the output is obtained by a linear combination of all past output values, plus the present value of the input. By construction the nonrecursive and recursive representations of system $h(n)$ are equivalent; that is, they produce the same output when they are excited by the same input signal.

Innovations representation. If the system $H(z)$ is minimum-phase, then both $h(n)$ and $h_I(n)$ are causal and stable. Hence, the output signal can be expressed nonrecursively by

$$x(n) = \sum_{k=0}^{\infty} h(k)w(n-k) = \sum_{k=-\infty}^n h(n-k)w(k) \quad (4.1.7)$$

or recursively by (4.1.6).

From (4.1.7) we obtain

$$x(n+1) = \sum_{k=-\infty}^n h(n+1-k)w(k) + w(n+1)$$

or by using (4.1.5)

$$x(n+1) = \underbrace{\sum_{k=-\infty}^n h(n+1-k)x(k)}_{\text{past information: linear combination of } x(n), x(n-1), \dots} + \underbrace{w(n+1)}_{\text{new information}} \quad (4.1.8)$$

Careful inspection of (4.1.8) indicates that if the system generating $x(n)$ is minimum-phase, the sample $w(n+1)$ brings all the new information (*innovation*) to be carried by the sample $x(n+1)$. All other information can be predicted from the past samples $x(n), x(n-1), \dots$ of the signal (see Section 6.6). We stress that this interpretation holds only if $H(z)$ is minimum-phase.

The system $H(z)$ generates the signal $x(n)$ by introducing dependence in the white noise input $w(n)$ and is known as the *synthesis* or *coloring* filter. In contrast, the inverse system $H_1(z)$ can be used to recover the input $w(n)$ and is known as the *analysis* or *whitening* filter. In this sense the innovations sequence and the output process are completely equivalent. The synthesis and analysis filters are shown in Figure 4.2.

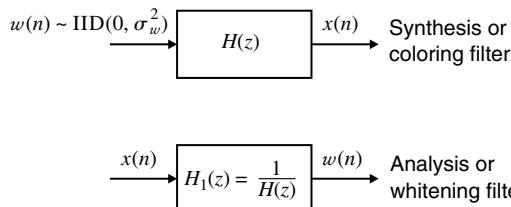


FIGURE 4.2
Synthesis and analysis filters used in innovations representation.

Spectral factorization

Most random processes with a continuous PSD $R_x(e^{j\omega})$ can be generated by exciting a minimum-phase system $H_{\min}(z)$ with white noise. The PSD of the resulting process is given by

$$R_x(e^{j\omega}) = \sigma_w^2 |H_{\min}(e^{j\omega})|^2 \quad (4.1.9)$$

The process of obtaining $H_{\min}(z)$ from $R_x(e^{j\omega})$ or $r_x(l)$ is known as *spectral factorization*.

If the PSD $R_x(e^{j\omega})$ satisfies the Paley-Wiener condition

$$\int_{-\pi}^{\pi} |\ln R_x(e^{j\omega})| d\omega < \infty \quad (4.1.10)$$

then the process $x(n)$ is called *regular* and its complex PSD can be factored as follows (see Section 2.4.4)

$$R_x(z) = \sigma_w^2 H_{\min}(z) H_{\min}^* \left(\frac{1}{z^*} \right) \quad (4.1.11)$$

$$\text{where } \sigma_w^2 = \exp \left\{ \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln[R_x(e^{j\omega})] d\omega \right\} \quad (4.1.12)$$

is the variance of the white noise input and can be interpreted as the *geometric mean* of $R_x(e^{j\omega})$. Consider the inverse Fourier transform of $\ln R_x(e^{j\omega})$:

$$c(k) \triangleq \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln[R_x(e^{j\omega})] e^{jk\omega} d\omega \quad (4.1.13)$$

which is a sequence known as the *cepstrum* of $r_x(l)$. Note that $c(0) = \sigma_w^2$. Thus in the cepstral domain, the multiplicative factors $H_{\min}(z)$ and $H_{\min}^*(1/z^*)$ are now *additively separable* due to the natural logarithm of $R_x(e^{j\omega})$. Define

$$c_+(k) \triangleq \frac{c(0)}{2} + c(k)u(k-1) \quad (4.1.14)$$

$$\text{and } c_-(k) \triangleq \frac{c(0)}{2} + c(k)u(-k-1) \quad (4.1.15)$$

as the positive- and negative-axis projections of $c(k)$, respectively, with $c(0)$ distributed equally between them. Then we obtain

$$h_{\min}(n) = \mathcal{F}^{-1}\{\exp \mathcal{F}[c_+(k)]\} \quad (4.1.16)$$

as the impulse response of the minimum-phase system $H_{\min}(z)$. Similarly,

$$h_{\max}(n) = \mathcal{F}^{-1}\{\exp \mathcal{F}[c_-(k)]\} \quad (4.1.17)$$

is the corresponding maximum-phase system. This completes the spectral factorization procedure for an arbitrary PSD $R_x(e^{j\omega})$, which, in general, is a complicated task. However, it is straightforward if $R_x(z)$ is a rational function, as we discussed in Section 2.4.2.

Spectral flatness measure

The spectral flatness measure (SFM) of a zero-mean process with PSD $R_x(e^{j\omega})$ is defined by (Makhoul 1975)

$$\text{SFM}_x \triangleq \frac{\exp \left\{ \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln[R_x(e^{j\omega})] d\omega \right\}}{\frac{1}{2\pi} \int_{-\pi}^{\pi} R_x(e^{j\omega}) d\omega} = \frac{\sigma_w^2}{\sigma_x^2} \quad (4.1.18)$$

where the second equality follows from (4.1.12). It describes the shape (or more appropriately, flatness) of the PSD by a single number. If $x(n)$ is a white noise process, then $R_x(e^{j\omega}) = \sigma_x^2$ and $\text{SFM}_x = 1$. More specifically, we can show that

$$0 \leq \text{SFM}_x \leq 1 \quad (4.1.19)$$

Observe that the numerator of (4.1.18) is the geometric mean while the denominator is the arithmetic mean of a real-valued, nonnegative continuous waveform $R_x(e^{j\omega})$. Since $x(n)$ is a regular process satisfying (4.1.10), these means are always positive. Furthermore, their ratio, by definition, is never greater than unity and is equal to unity if the waveform is constant. This, then, proves (4.1.19). A detailed proof is given in Jayant and Noll (1984).

When $x(n)$ is obtained by filtering the zero-mean white noise process $w(n)$ through the filter $H(z)$, then the coloring of $R_x(e^{j\omega})$ is due to $H(z)$. In this case, $R_x(e^{j\omega}) = \sigma_w^2 |H(e^{j\omega})|^2$ from (4.1.9), and we obtain

$$\text{SFM}_x = \frac{\sigma_w^2}{\sigma_x^2} = \frac{\sigma_w^2}{\frac{1}{2\pi} \int_{-\pi}^{\pi} \sigma_w^2 |H(e^{j\omega})|^2 d\omega} = \frac{1}{\frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 d\omega} \quad (4.1.20)$$

Thus SFM_x is the inverse of the filter power (or *power transfer factor*) if $h(0)$ is normalized to unity.

4.1.2 Parametric Pole-Zero Signal Models

Parametric models describe a system with a finite number of parameters. The major subject of this chapter is the treatment of parametric models that have rational system functions. To this end, consider a system described by the following linear constant-coefficient difference equation

$$x(n) + \sum_{k=1}^P a_k x(n-k) = \sum_{k=0}^Q d_k w(n-k) \quad (4.1.21)$$

where $w(n)$ and $x(n)$ are the input and output signals, respectively. Taking the z -transform of both sides, we find that the system function is

$$H(z) = \frac{X(z)}{W(z)} = \frac{\sum_{k=0}^Q d_k z^{-k}}{1 + \sum_{k=1}^P a_k z^{-k}} \triangleq \frac{D(z)}{A(z)} \quad (4.1.22)$$

We can express $H(z)$ in terms of the poles and zeros of the system as follows:

$$H(z) = d_0 \frac{\prod_{k=1}^Q (1 - z_k z^{-1})}{\prod_{k=1}^P (1 - p_k z^{-1})} \quad (4.1.23)$$

The system has Q zeros $\{z_k\}$ and P poles $\{p_k\}$ (zeros and poles at $z = 0$ are not considered here). The term d_0 is the system gain. For the rest of the book, we assume that the polynomials $D(z)$ and $A(z)$ do not have any common roots, that is, common poles and zeros have already been canceled.

Types of pole-zero models

There are three cases of interest:

- For $P > 0$ and $Q > 0$, we have a *pole-zero* model, denoted by $\text{PZ}(P, Q)$. If the model is assumed to be causal, its output is given by

$$x(n) = - \sum_{k=1}^P a_k x(n-k) + \sum_{k=0}^Q d_k w(n-k) \quad (4.1.24)$$

- For $P = 0$, we have an *all-zero* model, denoted by $\text{AZ}(Q)$. The input-output difference equation is

$$x(n) = \sum_{k=0}^Q d_k w(n-k) \quad (4.1.25)$$

- For $Q = 0$, we have an *all-pole* model, denoted by $\text{AP}(P)$. The input-output difference equation is

$$x(n) = - \sum_{k=1}^P a_k x(n-k) + d_0 w(n) \quad (4.1.26)$$

If we excite a parametric model with white noise, we obtain a signal whose second-order moments are determined by the parameters of the model. Indeed, from Sections 3.4.2 and 3.4.3, we recall that if $w(n) \sim \text{IID}\{0, \sigma_w^2\}$ with finite variance,[†] then

$$r_x(l) = \sigma_w^2 r_h(l) = \sigma_w^2 h(l) * h^*(-l) \quad (4.1.27)$$

$$R_x(z) = \sigma_w^2 R_h(z) = \sigma_w^2 H(z) H^* \left(\frac{1}{z^*} \right) \quad (4.1.28)$$

$$R_x(e^{j\omega}) = \sigma_w^2 R_h(e^{j\omega}) = \sigma_w^2 |H(e^{j\omega})|^2 \quad (4.1.29)$$

Such signal models are of great practical interest and have special names in the statistical literature:

- The $\text{AZ}(Q)$ is known as the *moving-average* model, denoted by $\text{MA}(Q)$.
- The $\text{AP}(P)$ is known as the *autoregressive* model, denoted by $\text{AR}(P)$.
- The $\text{PZ}(P, Q)$ is known as the *autoregressive moving-average* model, denoted by $\text{ARMA}(P, Q)$.

We specify a parametric signal model by normalizing $d_0 = 1$ and setting the variance of the input to σ_w^2 . The defining set of model parameters is given by $\{a_1, a_2, \dots, a_P, d_1, \dots, d_Q, \sigma_w^2\}$ (see Figure 4.3). An alternative is to set $\sigma_w^2 = 1$ and leave d_0 arbitrary. We stress that these models assume the resulting processes are stationary, which is ensured if the corresponding systems are BIBO stable.

[†]The case of infinite variance is discussed in Chapter 12.

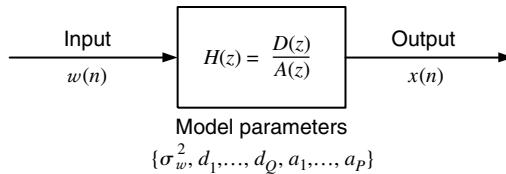


FIGURE 4.3
Block diagram representation of a parametric, rational signal model.

Short-memory behavior

To find the memory behavior of pole-zero models, we investigate the nature of their impulse response. To this end, we recall that for $Q \geq P$, (4.1.23) can be expanded as

$$H(z) = \sum_{j=0}^{Q-P} B_j z^{-j} + \sum_{k=1}^P \frac{A_k}{1 - p_k z^{-1}} \quad (4.1.30)$$

where for simplicity we assume that the model has P distinct poles. The first term in (4.1.30) disappears if $P > Q$. The coefficients B_j can be obtained by long division:

$$A_k = (1 - p_k z^{-1}) H(z)|_{z=p_k} \quad (4.1.31)$$

If the model is causal, taking the inverse z -transform results in an impulse response that is a linear combination of impulses, real exponentials, and damped sinusoids (produced by the combination of complex exponentials)

$$h(n) = \sum_{j=0}^{Q-P} B_j \delta(n-j) + \sum_{k=1}^{P_1} A_k (p_k)^n u(n) + \sum_{i=1}^{P_2} C_i r_i^n \cos(\omega_i n + \phi_i) u(n) \quad (4.1.32)$$

where $p_i = r_i e^{\pm j\omega_i}$ and $P = P_1 + 2P_2$. Recall that $u(n)$ and $\delta(n)$ are the unit step and unit impulse functions, respectively. We note that the memory of any all-pole model decays exponentially with time and that the rate of decay is controlled by the pole closest to the unit circle. The contribution of multiple poles at the same location is treated in Problem 4.1.

Careful inspection of (4.1.32) leads to the following conclusions:

1. For AZ(Q) models, the impulse response has finite duration and, therefore, can have any shape.
2. The impulse response of causal AP(P) and PZ(P, Q) models with single poles consists of a linear combination of damped real exponentials (produced by the real poles) and exponentially damped sinusoids (produced by complex conjugate poles). The rate of decay decreases as the poles move closer to the unit circle and is determined by the pole closest to the unit circle.
3. The model is stable if and only if $h(n)$ is absolutely summable, which, due to (4.1.32), is equivalent to $|p_k| < 1$ for all k . In other words, a causal pole-zero model is BIBO stable if and only if all the poles are inside the unit circle.[†]

We conclude that causal, stable PZ(P, Q) models with $P > 0$ have an exponentially fading memory because their impulse response decays exponentially with time. Therefore, the autocorrelation $r_h(l) = h(l) * h^*(-l)$ also decays exponentially (see Example 4.2.2), and pole-zero models have short memory according to the definition given in Section 3.4.3.

Generation of random signals with rational power spectra

Sample realizations of random sequences with rational power spectra can be easily generated by using the difference equation (4.1.24) and a random number generator. In most applications, we use a Gaussian excitation because the generated sequence will also be Gaussian. For non-Gaussian inputs, it is difficult to predict the type of distribution of the output signal. If, on one hand, we specify the frequency response of the model,

[†]Poles on the unit circle are discussed in Section 4.5.

the coefficients of the difference equation can be obtained by using a digital filter design package. If, on the other hand, the power spectrum or the autocorrelation is given, the coefficients of the model are determined via spectral factorization. If we wish to avoid the transient effects that make some of the initial output samples nonstationary, we should consider the response of the model only after the initial transients have died out.

4.1.3 Mixed Processes and Wold Decomposition

An arbitrary stationary random process can be constructed to possess a continuous PSD $R_x(e^{j\omega})$ and a discrete power spectrum $R_x(k)$. Such processes are called *mixed* processes because the continuous PSD is due to regular processes while the discrete spectrum is due to harmonic (or almost periodic) processes. A further interpretation of mixed processes is that the first part is an *unpredictable* process while the second part is a *predictable* process (in the sense that past samples can be used to exactly determine future samples). This interpretation is due to the Wold decomposition theorem.

THEOREM 4.1 (WOLD DECOMPOSITION). A general stationary random process can be written as a sum

$$x(n) = x_r(n) + x_p(n) \quad (4.1.33)$$

where $x_r(n)$ is a regular process possessing a continuous spectrum and $x_p(n)$ is a predictable process possessing a discrete spectrum. Furthermore, $x_r(n)$ is orthogonal to $x_p(n)$; that is,

$$E\{x_r(n_1)x_p^*(n_2)\} = 0 \quad \text{for all } n_1, n_2 \quad (4.1.34)$$

The proof of this theorem is very involved, but a good approach to it is given in Therrien (1992). Using (4.1.34), the correlation sequence of $x(n)$ in (4.1.33) is given by

$$r_x(l) = r_{x_r}(l) + r_{x_p}(l)$$

from which we obtain the continuous and discrete spectra. As discussed above, the regular process has an innovations representation $w(n)$ that is uncorrelated but *not* independent. For example, $w(n)$ can be the output of an all-pass filter driven by an IID sequence. To determine if this is the case, we need to use higher-order moments (see Section 12.1).

4.2 ALL-POLE MODELS

We start our discussion of linear signal models with all-pole models because they are the easiest to analyze and the most often used in practical applications. We assume an all-pole model of the form

$$H(z) = \frac{d_0}{A(z)} = \frac{d_0}{1 + \sum_{k=1}^P a_k z^{-k}} = \frac{d_0}{\prod_{k=1}^P (1 - p_k z^{-1})} \quad (4.2.1)$$

where d_0 is the system gain and P is the order of the model. The all-pole model can be implemented using either a direct or a lattice structure. The conversion between the two sets of parameters can be done by using the step-up and step-down recursions described in Section 2.5.

4.2.1 Model Properties

In this section, we derive analytic expressions for various properties of the all-pole model, namely, the impulse response, the autocorrelation, and the spectrum. We determine the system-related properties $r_h(l)$ and $R_h(e^{j\omega})$ because the results can be readily applied to obtain the signal model properties for inputs with both continuous and discrete spectra.

Impulse response. The impulse response $h(n)$ can be specified by first rewriting (4.2.1) as

$$H(z) + \sum_{k=1}^P a_k H(z) z^{-k} = d_0$$

and then taking the inverse z -transform to obtain

$$h(n) + \sum_{k=1}^P a_k h(n-k) = d_0 \delta(n) \quad (4.2.2)$$

If the system is causal, then

$$h(n) = - \sum_{k=1}^P a_k h(n-k) + d_0 \delta(n) \quad (4.2.3)$$

If $H(z)$ has all its poles inside the unit circle, then $h(n)$ is a causal, stable sequence and the system is minimum-phase. From (4.2.3) we have

$$h(0) = d_0 \quad (4.2.4)$$

$$h(n) = - \sum_{k=1}^P a_k h(n-k) \quad n > 0 \quad (4.2.5)$$

and owing to causality we have

$$h(n) = 0 \quad n < 0 \quad (4.2.6)$$

Thus, except for the value at $n = 0$, $h(n)$ can be obtained recursively as a linearly weighted summation of its previous values $h(n-1), \dots, h(n-P)$. One can say that $h(n)$ can be *predicted* (with zero error for $n \neq 0$) from the past P values. Thus, the coefficients $\{a_k\}$ are often referred to as *predictor coefficients*. Note that there is a close relationship between all-pole models and linear prediction that will be discussed in Section 4.2.2.

From (4.2.4) and (4.2.5), we can also write the inverse relation

$$a_n = - \frac{h(n)}{h(0)} - \sum_{k=1}^{n-1} a_k \frac{h(n-k)}{h(0)} \quad 1 \leq n \leq P \quad (4.2.7)$$

with $a_0 = 1$. From (4.2.7) and (4.2.4), we conclude that if we are given the first $P+1$ values of the impulse response $h(n)$, $0 \leq n \leq P$, then the parameters of the all-pole filter are completely specified.

Finally, we note that a causal $H(z)$ can be written as a one-sided, infinite polynomial $H(z) = \sum_{n=0}^{\infty} h(n)z^{-n}$. This representation of $H(z)$ implies that any finite-order, all-pole model can be represented equivalently by an infinite number of zeros. In general, a single pole can be represented by an infinite number of zeros, and conversely a single zero can be represented by an infinite number of poles. If the poles are inside the unit circle, so are the corresponding zeros, and vice versa.

EXAMPLE 4.2.1. A single pole at $z = a$ can be represented by

$$H(z) = \frac{1}{1 - az^{-1}} = \sum_{n=0}^{\infty} a^n z^{-n} \quad |a| < 1 \quad (4.2.8)$$

The question is, where are the infinite number of zeros located? To find the answer, let us consider the finite polynomial

$$H_N(z) = \sum_{n=0}^N a^n z^{-n} \quad (4.2.9)$$

where we have truncated $H(z)$ at $n = N$. Thus $H_N(z)$ is a geometric series that can be written in closed form as

$$H_N(z) = \frac{1 - a^{N+1}z^{-(N+1)}}{1 - az^{-1}} \quad (4.2.10)$$

And $H_N(z)$ has a single pole at $z = a$ and $N + 1$ zeros at

$$z_i = ae^{j2\pi i/(N+1)} \quad i = 0, 1, \dots, N \quad (4.2.11)$$

The $N + 1$ zeros are equally distributed on the circle $|z| = a$ with one of the zeros (for $i = 0$) located at $z = a$. But the zero at $z = a$ cancels the pole at the same location. Therefore, $H_N(z)$ has the remaining N zeros:

$$z_i = ae^{j2\pi i(N+1)} \quad i = 1, 2, \dots, N \quad (4.2.12)$$

The transfer function $H(z)$ of the single-pole model is obtained from $H_N(z)$ by letting N go to infinity. In the limit, $H_\infty(z)$ has an infinite number of zeros equally distributed on the circle $|z| = a$; the zeros are everywhere on that circle except at the point $z = a$. Similarly, the denominator from (4.2.8), a polynomial with a single zero at $z = a$, can be written as

$$A(z) = 1 - az^{-1} = \frac{1}{H(z)} = \frac{1}{1 + \sum_{n=1}^{\infty} a^n z^{-n}} \quad |a| < 1 \quad (4.2.13)$$

that is, a single zero can also be represented by an infinite number of poles. In this case, the poles are equally distributed on a circle that passes through the location of the zero; the poles are everywhere on the circle except at the actual location of the zero.

Autocorrelation. The impulse response $h(n)$ of an all-pole model has infinite duration so that its autocorrelation involves an infinite summation, which is not practical to write in closed form except for low-order models. However, the autocorrelation function obeys a recursive relation that relates the autocorrelation values to the model parameters. Multiplying (4.2.2) by $h^*(n - l)$ and summing over all n , we have

$$\sum_{n=-\infty}^{\infty} \sum_{k=0}^P a_k h(n - k) h^*(n - l) = d_0 \sum_{n=-\infty}^{\infty} h^*(n - l) \delta(n) \quad (4.2.14)$$

where $a_0 = 1$. Interchanging the order of summations in the left-hand side, we obtain

$$\sum_{k=0}^P a_k r_h(l - k) = d_0 h^*(-l) \quad -\infty < l < \infty \quad (4.2.15)$$

where $r_h(l)$ is the autocorrelation of $h(n)$. Equation (4.2.15) is true for all l , but because $h(l) = 0$ for $l < 0$, $h(-l) = 0$ for $l > 0$, and we have

$$\sum_{k=0}^P a_k r_h(l - k) = 0 \quad l > 0 \quad (4.2.16)$$

From (4.2.4) and (4.2.15), we also have for $l = 0$,

$$\sum_{k=0}^P a_k r_h(-k) = |d_0|^2 \quad (4.2.17)$$

where we used the fact that $r_h^*(-l) = r_h(l)$. Equation (4.2.16) can be rewritten as

$$r_h(l) = - \sum_{k=1}^P a_k r_h(l - k) \quad l > 0 \quad (4.2.18)$$

which is a recursive relation for $r_h(l)$ in terms of past values of the autocorrelation and $\{a_k\}$. Relation (4.2.18) for $r_h(l)$ is similar to relation (4.2.5) for $h(n)$, but with one important difference: (4.2.5) for $h(n)$ is true for all $n \neq 0$ while (4.2.18) for $r_h(l)$ is true only if $l > 0$; for $l < 0$, $r_h(l)$ obeys (4.2.15).

If we define the *normalized* autocorrelation coefficients as

$$\rho_h(l) = \frac{r_h(l)}{r_h(0)} \quad (4.2.19)$$

then we can divide (4.2.17) by $r_h(0)$ and deduce the following relation for $r_h(0)$

$$r_h(0) = \frac{|d_0|^2}{1 + \sum_{k=1}^P a_k \rho_h(k)} \quad (4.2.20)$$

which is the energy of the output of the all-pole filter when excited by a single impulse.

Autocorrelation in terms of poles. The complex spectrum of the AP(P) model is

$$R_h(z) = H(z)H\left(\frac{1}{z^*}\right) = |d_0|^2 \prod_{k=1}^P \frac{1}{(1 - p_k z^{-1})(1 - p_k z^*)} \quad (4.2.21)$$

Therefore, the autocorrelation sequence can be expressed in terms of the poles by taking the inverse z -transform of $R_h(z)$, that is, $r_h(l) = \mathcal{Z}^{-1}\{R_h(z)\}$. The poles p_k of the minimum-phase model $H(z)$ contribute causal terms in the partial fraction expansion, whereas the poles $1/p_k$ of the nonminimum-phase model $H(1/z^*)$ contribute noncausal terms. This is best illustrated with the following example.

EXAMPLE 4.2.2. Consider the following minimum-phase AP(1) model

$$H(z) = \frac{1}{1 + az^{-1}} \quad -1 < a < 1 \quad (4.2.22)$$

Owing to causality, the ROC of $H(z)$ is $|z| > |a|$. The z -transform

$$H(z^{-1}) = \frac{1}{1 + az} \quad -1 < a < 1 \quad (4.2.23)$$

corresponds to the noncausal sequence $h(-n) = (-a)^{-n}u(-n)$, and its ROC is $|z| < 1/|a|$. Hence,

$$R_h(z) = H(z)H(z^{-1}) = \frac{1}{(1 + az^{-1})(1 + az)} \quad (4.2.24)$$

which corresponds to a two-sided sequence because its ROC, $|a| < |z| < 1/|a|$, is a ring in the z -plane. Using partial fraction expansion, we obtain

$$R_h(z) = \frac{-a}{1 - a^2} \frac{z^{-1}}{1 + az^{-1}} + \frac{1}{1 - a^2} \frac{1}{1 + az} \quad (4.2.25)$$

The pole $p = -a$ corresponds to the causal sequence $[1/(1 - a^2)](-a)^l u(l - 1)$, and the pole $p = -1/a$ to the anticausal sequence $[1/(1 - a^2)](-a)^{-l} u(-l)$. Combining the two terms, we obtain

$$r_h(l) = \frac{1}{1 - a^2} (-a)^{|l|} \quad -\infty < l < \infty \quad (4.2.26)$$

$$\text{or} \quad \rho_h(l) = (-a)^{|l|} \quad -\infty < l < \infty \quad (4.2.27)$$

Note that complex conjugate poles will contribute two-sided damped sinusoidal terms obtained by combining pairs of the form (4.2.27) with $u = p$ and $a = p^v$.

Impulse train excitations. The response of an AP(P) model to a periodic impulse train with period L is periodic with the same period and is given by

$$\begin{aligned}\tilde{h}(n) + \sum_{k=1}^P a_k \tilde{h}(n-k) &= d_0 \sum_{m=-\infty}^{\infty} \delta(n+Lm) \\ &= \begin{cases} d_0 & n+Lm = 0 \\ 0 & n+Lm \neq 0 \end{cases}\end{aligned}\quad (4.2.28)$$

which shows that the prediction error is zero for samples inside the period and d_0 at the beginning of each period. If we multiply both sides of (4.2.28) by $\tilde{h}^*(n-l)$ and sum over a period $0 \leq n \leq L-1$, we obtain

$$\tilde{r}_h(l) + \sum_{k=1}^P a_k \tilde{r}_h(l-k) = \frac{d_0}{L} \tilde{h}^*(-l) \quad \text{all } l \quad (4.2.29)$$

where $\tilde{r}_h(l)$ is the periodic autocorrelation of $\tilde{h}(n)$. Since, in contrast to $h(n)$ in (4.2.15), $\tilde{h}(n)$ is not necessarily zero for $n < 0$, the periodic autocorrelation $\tilde{r}_h(l)$ will not in general obey the linear prediction equation anywhere. Similar results can be obtained for harmonic process excitations.

Model parameters in terms of autocorrelation. Equations (4.2.15) for $l = 0, 1, \dots, P$ comprise $P+1$ equations that relate the $P+1$ parameters of $H(z)$, namely, d_0 and $\{a_k, 1 \leq k \leq P\}$, to the first $P+1$ autocorrelation coefficients $r_h(0), r_h(1), \dots, r_h(P)$. These $P+1$ equations can be written in matrix form as

$$\begin{bmatrix} r_h(0) & r_h^*(1) & \cdots & r_h^*(P) \\ r_h(1) & r_h(0) & \cdots & r_h^*(P-1) \\ \vdots & \vdots & \ddots & \vdots \\ r_h(P) & r_h(P-1) & \cdots & r_h(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_P \end{bmatrix} = \begin{bmatrix} |d_0|^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (4.2.30)$$

If we are given the first $P+1$ autocorrelations, (4.2.30) comprises a system of $P+1$ linear equations, with a Hermitian, Toeplitz matrix that can be solved for d_0 and $\{a_k\}$.

Because of the special structure in (4.2.30), the model parameters are found from the autocorrelations by using the last set of P equations in (4.2.30), followed by the computation of d_0 from the first equation, which is the same as (4.2.17). From (4.2.30), we can write in matrix notation

$$\mathbf{R}_h \mathbf{a} = -\mathbf{r}_h \quad (4.2.31)$$

where \mathbf{R}_h is the autocorrelation matrix, \mathbf{a} is the vector of the model parameters, and \mathbf{r}_h is the vector of autocorrelations. Since $r_x(l) = \sigma_w^2 r_h(l)$, we can also express the model parameters in terms of the autocorrelation $r_x(l)$ of the output process $x(n)$ as follows:

$$\mathbf{R}_x \mathbf{a} = -\mathbf{r}_x \quad (4.2.32)$$

These equations are known as the *Yule-Walker equations* in the statistics literature. In the sequel, we drop the subscript from the autocorrelation sequence or matrix whenever the analysis holds for both the impulse response and the model output.

Because of the Toeplitz structure and the nature of the right-hand side, the linear systems (4.2.31) and (4.2.32) can be solved recursively by using the algorithm of Levinson-Durbin (see Section 7.4). After \mathbf{a} is solved for, the system gain d_0 can be computed from (4.2.17).

Therefore, given $r(0), r(1), \dots, r(P)$, we can completely specify the parameters of the all-pole model by solving a set of linear equations. Below, we will see that the converse is also true: Given the model parameters, we can find the first $P+1$ autocorrelations by

solving a set of linear equations. *This elegant solution of the spectral factorization problem is unique to all-pole models.* In the case in which the model contains zeros ($Q \neq 0$), the spectral factorization problem requires the solution of a nonlinear system of equations.

Autocorrelation in terms of model parameters. If we normalize the autocorrelations in (4.2.31) by dividing throughout by $r(0)$, we obtain the following system of equations

$$\mathbf{P}\mathbf{a} = -\boldsymbol{\rho} \quad (4.2.33)$$

where \mathbf{P} is the normalized autocorrelation matrix and

$$\boldsymbol{\rho} = [\rho(1) \ \rho(2) \ \cdots \ \rho(P)]^H \quad (4.2.34)$$

is the vector of normalized autocorrelations. This set of P equations relates the P model coefficients with the first P (normalized) autocorrelation values. If the poles of the all-pole filter are strictly inside the unit circle, the mapping between the P -dimensional vectors \mathbf{a} and $\boldsymbol{\rho}$ is *unique*. If, in fact, we are given the vector \mathbf{a} , then the normalized autocorrelation vector $\boldsymbol{\rho}$ can be computed from \mathbf{a} by using the set of equations that can be deduced from (4.2.33)

$$\mathbf{A}\boldsymbol{\rho} = -\mathbf{a} \quad (4.2.35)$$

where $\langle \mathbf{A} \rangle_{ij} = a_{i-j} + a_{i+j}$, assuming $a_m = 0$ for $m < 0$ and $m > P$ (see Problem 4.6).

Given the set of coefficients in \mathbf{a} , $\boldsymbol{\rho}$ can be obtained by solving (4.2.35). We will see that, under the assumption of a stable $H(z)$, a solution always exists. Furthermore, there exists a simple, recursive solution that is efficient (see Section 7.5). If, in addition to \mathbf{a} , we are given d_0 , we can evaluate $r(0)$ with (4.2.20) from $\boldsymbol{\rho}$ computed by (4.2.35). Autocorrelation values $r(l)$ for lags $l > P$ are found by using the recursion in (4.2.18) with $r(0), r(1), \dots, r(P)$.

EXAMPLE 4.2.3. For the AP(3) model with real coefficients we have

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r(1) & r(0) & r(1) \\ r(2) & r(1) & r(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = - \begin{bmatrix} r(1) \\ r(2) \\ r(3) \end{bmatrix} \quad (4.2.36)$$

$$d_0^2 = r(0) + a_1r(1) + a_2r(2) + a_3r(3) \quad (4.2.37)$$

Therefore, given $r(0), r(1), r(2), r(3)$, we can find the parameters of the all-pole model by solving (4.2.36) and then substituting into (4.2.37).

Suppose now that instead we are given the model parameters d_0, a_1, a_2, a_3 . If we divide both sides of (4.2.36) by $r(0)$ and solve for the normalized autocorrelations $\rho(1), \rho(2)$, and $\rho(3)$, we obtain

$$\begin{bmatrix} 1 + a_2 & a_3 & 0 \\ a_1 + a_3 & 1 & 0 \\ a_2 & a_1 & 1 \end{bmatrix} \begin{bmatrix} \rho(1) \\ \rho(2) \\ \rho(3) \end{bmatrix} = - \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad (4.2.38)$$

The value of $r(0)$ is obtained from

$$r(0) = \frac{d_0^2}{1 + a_1\rho(1) + a_2\rho(2) + a_3\rho(3)} \quad (4.2.39)$$

If $r(0) = 2$, $r(1) = 1.6$, $r(2) = 1.2$, and $r(3) = 1$, the Toeplitz matrix in (4.2.36) is positive definite because it has positive eigenvalues. Solving the linear system gives $a_1 = -0.9063$, $a_2 = 0.2500$, and $a_3 = -0.1563$. Substituting these values in (4.2.37), we obtain $d_0 = 0.8329$. Using the last two relations, we can recover the autocorrelation from the model parameters.

Correlation matching. All-pole models have the unique distinction that the model parameters are completely specified by the first $P + 1$ autocorrelation coefficients via a set of linear equations. We can write

$$\begin{bmatrix} d_0 \\ \mathbf{a} \end{bmatrix} \leftrightarrow \begin{bmatrix} r(0) \\ \boldsymbol{\rho} \end{bmatrix} \quad (4.2.40)$$

that is, the mapping of the model parameters $\{d_0, a_1, a_2, \dots, a_P\}$ to the autocorrelation coefficients specified by the vector $\{r(0), \rho(1), \dots, \rho(P)\}$ is reversible and unique. This statement implies that given any set of autocorrelation values $r(0), r(1), \dots, r(P)$, we can always find an all-pole model whose first $P + 1$ autocorrelation coefficients are equal to the given autocorrelations. This *correlation matching* of all-pole models is quite remarkable. This property is not shared by all-zero models and is true for pole-zero models only under certain conditions, as we will see in Section 4.4.

Spectrum. The z -transform of the autocorrelation $r(l)$ of $H(z)$ is given by

$$R(z) = H(z)H^* \left(\frac{1}{z^*} \right) \quad (4.2.41)$$

The spectrum is then equal to

$$R(e^{j\omega}) = |H(e^{j\omega})|^2 = \frac{|d_0|^2}{|A(e^{j\omega})|^2} \quad (4.2.42)$$

The right-hand side of (4.2.42) suggests a method for computing the spectrum: First compute $A(e^{j\omega})$ by taking the Fourier transform of the sequence $\{1, a_1, \dots, a_P\}$, then take the squared of the magnitude and divide $|d_0|^2$ by the result. The fast Fourier transform (FFT) can be used to this end by appending the sequence $\{1, a_1, \dots, a_P\}$ with as many zeros as needed to compute the desired number of frequency points.

Partial autocorrelation and lattice structures. We have seen that an AP(P) model is completely described by the first $P + 1$ values of its autocorrelation. However, we cannot determine the order of the model by using the autocorrelation sequence because it has infinite duration. Suppose that we start fitting models of increasing order m , using the autocorrelation sequence of an AP(P) model and the Yule-Walker equations

$$\begin{bmatrix} 1 & \rho^*(1) & \cdots & \rho^*(m-1) \\ \rho(1) & 1 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \rho^*(1) \\ \rho(m-1) & \cdots & \rho(1) & 1 \end{bmatrix} \begin{bmatrix} a_1^{(m)} \\ a_2^{(m)} \\ \vdots \\ a_m^{(m)} \end{bmatrix} = - \begin{bmatrix} \rho^*(1) \\ \rho^*(2) \\ \vdots \\ \rho^*(m) \end{bmatrix} \quad (4.2.43)$$

Since $a_m^{(m)} = 0$ for $m > P$, we can use the sequence $a_m^{(m)}$, $m = 1, 2, \dots$, which is known as the *partial autocorrelation sequence (PACS)*, to determine the order of the all-pole model. Recall from Section 2.5 that

$$a_m^{(m)} = k_m \quad (4.2.44)$$

that is, the PACS is identical to the lattice parameters. A statistical definition and interpretation of the PACS are also given in Section 7.2. The PACS can be defined for any valid (i.e., positive definite) autocorrelation sequence and can be efficiently computed by using the algorithms of Levinson-Durbin and Schur (see Chapter 7).

Furthermore, it has been shown (Burg 1975) that

$$r(0) \prod_{m=1}^P \frac{1 - |k_m|}{1 + |k_m|} \leq R(e^{j\omega}) \leq r(0) \prod_{m=1}^P \frac{1 + |k_m|}{1 - |k_m|} \quad (4.2.45)$$

which indicates that the spectral dynamic range increases if some lattice parameter moves close to 1 or equivalently some pole moves close to the unit circle.

Equivalent model representations. From the previous discussions (see also Chapter 7) we conclude that a minimum-phase AP(P) model can be uniquely described by any one of the following representations:

1. Direct structure: $\{d_0, a_1, a_2, \dots, a_P\}$
2. Lattice structure: $\{d_0, k_1, k_2, \dots, k_P\}$
3. Autocorrelation: $\{r(0), r(1), \dots, r(P)\}$

163

SECTION 4.2
All-Pole Models

where we assume, without loss of generality, that $d_0 > 0$. Note that the minimum-phase property requires that all poles be inside the unit circle or all $|k_m| < 1$ or that \mathbf{R}_{P+1} be positive definite. The transformation from any of the above representations to any other can be done by using the algorithms developed in Section 7.5.

Minimum-phase conditions. As we will show in Section 7.5, if the Toeplitz matrix \mathbf{R}_h (or equivalently \mathbf{R}_x) is positive definite, then $|k_m| < 1$ for all $m = 1, 2, \dots, P$. Therefore, the AP(P) model obtained by solving the Yule-Walker equations is minimum-phase. Therefore, the Yule-Walker equations provide a simple and elegant solution to the spectral factorization problem for all-pole models.

EXAMPLE 4.2.4. The poles of the model obtained in Example 4.2.3 are 0.8316 , $0.0373 + 0.4319i$, and $0.0373 - 0.4319i$. We see that the poles are inside the unit circle and that the autocorrelation sequence is positive definite. If we set $r_h(2) = -1.2$, the autocorrelation becomes negative definite and the obtained model $\mathbf{a} = [1 - 1.222 1.1575]^T$, $d_0 = 2.2271$, is nonminimum-phase.

Pole locations. The poles of $H(z)$ are the zeros $\{p_k\}$ of the polynomial $A(z)$. If the coefficients of $A(z)$ are assumed to be real, the poles are either real or come in complex conjugate pairs. In order for $H(z)$ to be minimum-phase, all poles must be inside the unit circle, that is, $|p_k| < 1$. The model parameters a_k can be written as sums of products of the poles p_k . In particular, it is easy to see that

$$a_1 = - \sum_{k=1}^P p_k \quad (4.2.46)$$

$$a_P = \prod_{k=1}^P (-p_k) \quad (4.2.47)$$

Thus, the first coefficient a_1 is the negative of the sum of the poles, and the last coefficient a_P is the product of the negative of the individual poles. Since $|p_k| < 1$, we must have $|a_P| < 1$ for a minimum-phase polynomial for which $a_0 = 1$. However, note that the reverse is not necessarily true: $|a_P| < 1$ does not guarantee minimum phase. The roots p_k can be computed by using any number of standard root-finding routines.

4.2.2 All-Pole Modeling and Linear Prediction

Consider the AP(P) model

$$x(n) = - \sum_{k=1}^P a_k x(n-k) + w(n) \quad (4.2.48)$$

Now recall from Chapter 1 that the M th-order linear predictor of $x(n)$ and the corresponding prediction error $e(n)$ are

$$\hat{x}(n) = - \sum_{k=1}^M a_k^0 x(n-k) \quad (4.2.49)$$

$$e(n) = x(n) - \hat{x}(n) = x(n) + \sum_{k=1}^M a_k^0 x(n-k) \quad (4.2.50)$$

or

$$x(n) = - \sum_{k=1}^M a_k^0 x(n-k) + e(n) \quad (4.2.51)$$

Notice that if the order of the linear predictor equals the order of the all-pole model ($M = P$) and if $a_k^0 = a_k$, then the prediction error is equal to the excitation of the all-pole model, that is, $e(n) = w(n)$. Since all-pole modeling and FIR linear prediction are closely related, many properties and algorithms developed for one of them can be applied to the other. Linear prediction is extensively studied in Chapters 6 and 7.

4.2.3 Autoregressive Models

Causal all-pole models excited by white noise play a major role in practical applications and are known as *autoregressive (AR)* models. An AR(P) model is defined by the difference equation

$$x(n) = - \sum_{k=1}^P a_k x(n-k) + w(n) \quad (4.2.52)$$

where $\{w(n)\} \sim \text{WN}(0, \sigma_w^2)$. An AR(P) model is valid only if the corresponding AP(P) system is stable. In this case, the output $x(n)$ is a stationary sequence with a mean value of zero. Postmultiplying (4.2.52) by $x^*(n-l)$ and taking the expectation, we obtain the following recursive relation for the autocorrelation:

$$r_x(l) = - \sum_{k=1}^P a_k r_x(l-k) + E\{w(n)x^*(n-l)\} \quad (4.2.53)$$

Similarly, using (4.1.1), we can show that $E\{w(n)x^*(n-l)\} = \sigma_w^2 h^*(-l)$. Thus, we have

$$r_x(l) = - \sum_{k=1}^P a_k r_x(l-k) + \sigma_w^2 h^*(-l) \quad \text{for all } l \quad (4.2.54)$$

The variance of the output signal is

$$\begin{aligned} \sigma_x^2 &= r_x(0) = - \sum_{k=1}^P a_k r_x(k) + \sigma_w^2 \\ \text{or} \quad \sigma_x^2 &= \frac{\sigma_w^2}{1 + \sum_{k=1}^P a_k \rho_x(k)} \end{aligned} \quad (4.2.55)$$

If we substitute $l = 0, 1, \dots, P$ in (4.2.55) and recall that $h(n) = 0$ for $n < 0$, we obtain the following set of Yule-Walker equations:

$$\begin{bmatrix} r_x(0) & r_x(1) & \cdots & r_x(P) \\ r_x^*(1) & r_x(0) & \cdots & r_x(P-1) \\ \vdots & \vdots & \ddots & \vdots \\ r_x^*(P) & r_x^*(P-1) & \cdots & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_P \end{bmatrix} = \begin{bmatrix} \sigma_w^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (4.2.56)$$

Careful inspection of the above equations reveals their similarity to the corresponding relationships developed previously for the AP(P) model. This should be no surprise since the power spectrum of the white noise is flat. However, there is one important difference we should clarify: AP(P) models were specified with a gain d_0 and the parameters $\{a_1, a_2, \dots, a_P\}$, but for AR(P) models we set the gain $d_0 = 1$ and define the model by the

variance of the white excitation σ_w^2 and the parameters $\{a_1, a_2, \dots, a_P\}$. In other words, we incorporate the gain of the model into the power of the input signal. Thus, the power spectrum of the output is $R_x(e^{j\omega}) = \sigma_w^2 |H(e^{j\omega})|^2$. Similar arguments apply to all parametric models driven by white noise. We just rederived some of the relationships to clarify these issues and to provide additional insight into the subject.

4.2.4 Lower-Order Models

In this section, we derive the properties of lower-order all-pole models, namely, first- and second-order models, with real coefficients.

First-order all-pole model: AP(1)

An AP(1) model has a transfer function

$$H(z) = \frac{d_0}{1 + az^{-1}} \quad (4.2.57)$$

with a single pole at $z = -a$ on the real axis. It is clear that $H(z)$ is minimum-phase if

$$-1 < a < 1 \quad (4.2.58)$$

From (4.2.18) with $P = 1$ and $l = 1$, we have

$$a_1 = -\frac{r(1)}{r(0)} = -\rho(1) \quad (4.2.59)$$

Similarly, from (4.2.44) with $m = 1$,

$$a_1^{(1)} = a = -\rho(1) = k_1 \quad (4.2.60)$$

Since from (4.2.4), $h(0) = d_0$, and from (4.2.5) $h(n) = -a_1 h(n-1)$ for $n > 0$, the impulse response of a single-pole filter is given by

$$h(n) = d_0(-a)^n u(n) \quad (4.2.61)$$

The same result can, of course, be obtained by taking the inverse z -transform of $H(z)$.

The autocorrelation is found in a similar fashion. From (4.2.18) and by using the fact that the autocorrelation is an even function,

$$r(l) = r(0)(-a)^{|l|} \quad \text{for all } l \quad (4.2.62)$$

and from (4.2.20)

$$r(0) = \frac{d_0^2}{1 - a^2} = \frac{d_0^2}{1 - k_1^2} \quad (4.2.63)$$

Therefore, if the energy $r(0)$ in the impulse response is set to unity, then the gain must be set to

$$d_0 = \sqrt{1 - k_1^2} \quad r(0) = 1 \quad (4.2.64)$$

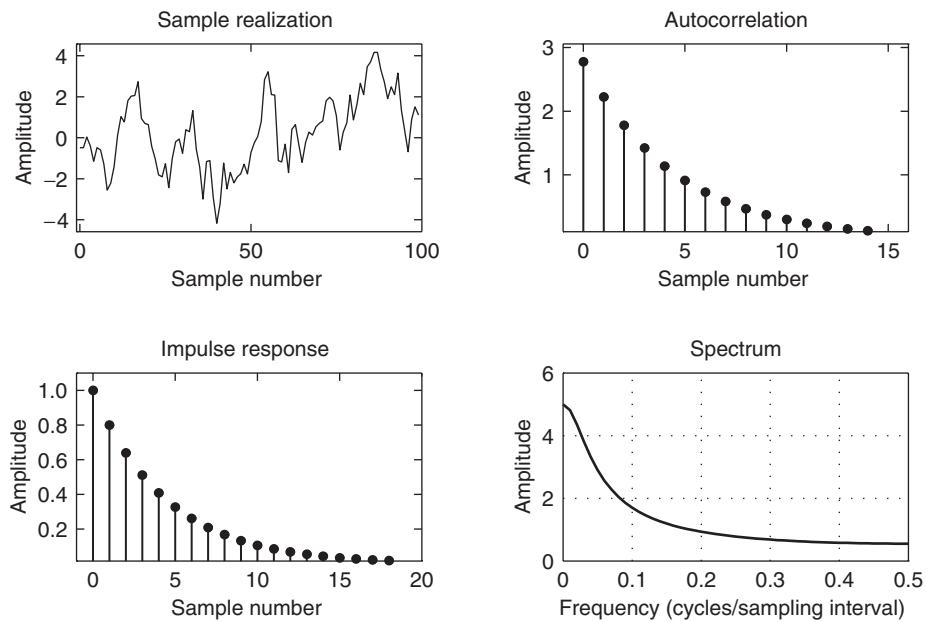
The z -transform of the autocorrelation is then

$$R(z) = \frac{d_0^2}{(1 + az^{-1})(1 + az)} = r(0) \sum_{l=-\infty}^{\infty} (-a)^{|l|} z^{-l} \quad (4.2.65)$$

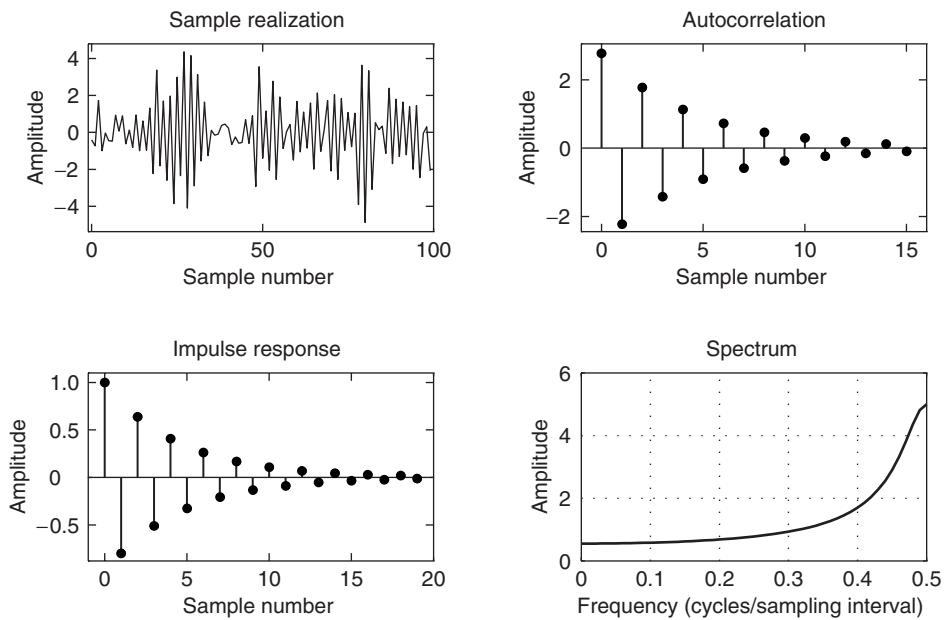
and the spectrum is

$$R(e^{j\omega}) = |H(e^{j\omega})|^2 = \frac{d_0^2}{|1 + ae^{-j\omega}|^2} = \frac{d_0^2}{1 + 2a \cos \omega + a^2} \quad (4.2.66)$$

Figures 4.4 and 4.5 show a typical realization of the output, the impulse response, autocorrelation, and spectrum of two AP(1) models. The sample process realizations were obtained by driving the model with white Gaussian noise of zero mean and unit variance. When the positive pole ($p = -a = 0.8$) is close to the unit circle, successive samples

**FIGURE 4.4**

Sample realization of the output process, impulse response, autocorrelation, and spectrum of an AP(1) model with $a = -0.8$.

**FIGURE 5.4**

Sample realization of the output process, impulse response, autocorrelation, and spectrum of an AP(1) model with $a = 0.8$.

of the output process are similar, as dictated by the slowly decaying autocorrelation and the corresponding low-pass spectrum. In contrast, a negative pole close to the unit circle results in a rapidly oscillating sequence. This is clearly reflected in the alternating sign of the autocorrelation sequence and the associated high-pass spectrum.

Note that a positive real pole is a type of low-pass filter, while a negative real pole has the spectral characteristics of a high-pass filter. (This situation in the digital domain contrasts with that in the corresponding analog domain where a real-axis pole can only have low-pass characteristics.) The discrete-time negative real pole can be thought of as one-half of two conjugate poles at half the sampling frequency. Notice that both spectra are even and have zero slope at $\omega = 0$ and $\omega = \pi$. These propositions are true of the spectra of all parametric models (i.e., pole-zero models) with real coefficients (see Problem 4.13).

Consider now the real-valued AR(1) process $x(n)$ generated by

$$x(n) = -ax(n-1) + w(n) \quad (4.2.67)$$

where $\{w(n)\} \sim \text{WN}(0, \sigma_w^2)$. Using the formula $R_x(z) = \sigma_w^2 H(z)H^*(1/z^*)$ and previous results, we can see that the autocorrelation and the PSD of $x(n)$ are given by

$$r_x(l) = \frac{\sigma_w^2}{1-a^2}(-a)^{|l|}$$

and

$$R_x(e^{j\omega}) = \sigma_w^2 \frac{1-a^2}{1+a^2+2a \cos \omega}$$

respectively. Since $\sigma_x^2 = r_x(0) = \sigma_w^2/(1-a^2)$, the SFM of $x(n)$ is [see (Section 4.1.18)]

$$\text{SFM}_x = \frac{\sigma_w^2}{\sigma_x^2} = 1-a^2 \quad (4.2.68)$$

Clearly, if $a = 0$, then from (4.2.67), $x(n)$ is a white noise process and from (4.2.68), $\text{SFM}_x = 1$. If $a \rightarrow 1$, then $\text{SFM}_x \rightarrow 0$; and in the limit when $a = 1$, the process becomes a random walk process, which is a nonstationary process with linearly increasing variance $E\{x^2(n)\} = n\sigma_w^2$. The correlation matrix is Toeplitz, and it is a rare exception in which eigenvalues and eigenvectors can be described by analytical expressions (Jayant and Noll 1984).

Second-order all-pole model: AP(2)

The system function of an AP(2) model is given by

$$H(z) = \frac{d_0}{1+a_1z^{-1}+a_2z^{-2}} = \frac{d_0}{(1-p_1z^{-1})(1-p_2z^{-1})} \quad (4.2.69)$$

From (4.2.46) and (4.2.47), we have

$$\begin{aligned} a_1 &= -(p_1 + p_2) \\ a_2 &= p_1 p_2 \end{aligned} \quad (4.2.70)$$

Recall that $H(z)$ is minimum-phase if the two poles p_1 and p_2 are inside the unit circle. Under these conditions, a_1 and a_2 lie in a triangular region defined by

$$\begin{aligned} -1 &< a_2 < 1 \\ a_2 - a_1 &> -1 \\ a_2 + a_1 &> -1 \end{aligned} \quad (4.2.71)$$

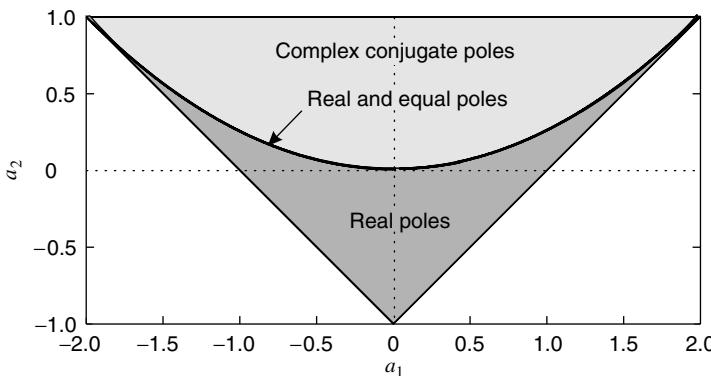
and shown in Figure 4.6. The first condition follows from (4.2.70) since $|p_1| < 1$ and $|p_2| < 1$. The last two conditions can be derived by assuming real roots and setting the larger root to less than 1 and the smaller root to greater than -1 . By adding the last two conditions, we obtain the redundant condition $a_2 > -1$.

Complex roots occur in the region

$$\frac{a_1^2}{4} < a_2 \leq 1 \quad \text{complex poles} \quad (4.2.72)$$

with $a_2 = 1$ resulting in both roots being on the unit circle. Note that, in order to have complex poles, a_2 cannot be negative. If the complex poles are written in polar form

$$p_i = re^{\pm j\theta} \quad 0 \leq r \leq 1 \quad (4.2.73)$$

**FIGURE 4.6**

Minimum-phase region (triangle) for the AP(2) model in the (a_1, a_2) parameter space.

then

$$a_1 = -2r \cos \theta \quad a_2 = r^2 \quad (4.2.74)$$

and

$$H(z) = \frac{d_0}{1 - (2r \cos \theta)z^{-1} + r^2 z^{-2}} \quad \text{complex poles} \quad (4.2.75)$$

Here, r is the radius (magnitude) of the poles, and θ is the angle or normalized frequency of the poles.

Impulse response. The impulse response of an AP(2) model can be written in terms of its two poles by evaluating the inverse z -transform of (4.2.69). The result is

$$h(n) = \frac{d_0}{p_1 - p_2} (p_1^{n+1} - p_2^{n+1}) u(n) \quad (4.2.76)$$

for $p_1 \neq p_2$. Otherwise, for $p_1 = p_2 = p$,

$$h(n) = d_0(n+1)p^n u(n) \quad (4.2.77)$$

In the special case of a complex conjugate pair of poles $p_1 = re^{j\theta}$ and $p_2 = re^{-j\theta}$, Equation (4.2.76) reduces to

$$h(n) = d_0 r^n \frac{\sin[(n+1)\theta]}{\sin \theta} u(n) \quad \text{complex poles} \quad (4.2.78)$$

Since $0 < r < 1$, $h(n)$ is a damped sinusoid of frequency θ .

Autocorrelation. The autocorrelation can also be written in terms of the two poles as

$$r(l) = \frac{d_0^2}{(p_1 - p_2)(1 - p_1 p_2)} \left(\frac{p_1^{l+1}}{1 - p_1^2} - \frac{p_2^{l+1}}{1 - p_2^2} \right) \quad l \geq 0 \quad (4.2.79)$$

from which we can deduce the energy

$$r(0) = \frac{d_0^2(1 + p_1 p_2)}{(1 - p_1 p_2)(1 - p_1^2)(1 - p_2^2)} \quad (4.2.80)$$

For the special case of a complex conjugate pole pair, (4.2.79) can be rewritten as

$$r(l) = \frac{d_0^2 r^l \{ \sin[(l+1)\theta] - r^2 \sin[(l-1)\theta] \}}{[(1 - r^2) \sin \theta](1 - 2r^2 \cos 2\theta + r^4)} \quad l \geq 0 \quad (4.2.81)$$

Then from (4.2.80) we can write an expression for the energy in terms of the polar coordinates of the complex conjugate pole pair

$$r(0) = \frac{d_0^2(1 + r^2)}{(1 - r^2)(1 - 2r^2 \cos 2\theta + r^4)} \quad (4.2.82)$$

The normalized autocorrelation is given by

$$\rho(l) = \frac{r^l \{\sin[(l+1)\theta] - r^2 \sin[(l-1)\theta]\}}{(1+r^2) \sin \theta} \quad l \geq 0 \quad (4.2.83)$$

which can be rewritten as

$$\rho(l) = \frac{1}{\cos \beta} r^l \cos(l\theta - \beta) \quad l \geq 0 \quad (4.2.84)$$

where

$$\tan \beta = \frac{(1-r^2) \cos \theta}{(1+r^2) \sin \theta} \quad (4.2.85)$$

Therefore, $\rho(l)$ is a damped cosine wave with its maximum amplitude at the origin.

Spectrum. By setting the two poles equal to

$$p_1 = r_1 e^{j\theta_1} \quad p_2 = r_2 e^{j\theta_2} \quad (4.2.86)$$

the spectrum of an AP(2) model can be written as

$$R(e^{j\omega}) = \frac{d_0^2}{[1 - 2r_1 \cos(\omega - \theta_1) + r_1^2][1 - 2r_2 \cos(\omega - \theta_2) + r_2^2]} \quad (4.2.87)$$

There are four cases of interest

Pole locations	Peak locations	Type of $R(e^{j\omega})$
$p_1 > 0, p_2 > 0$	$\omega = 0$	Low-pass
$p_1 < 0, p_2 < 0$	$\omega = \pi$	High-pass
$p_1 > 0, p_2 < 0$	$\omega = 0, \omega = \pi$	Stopband
$p_{1,2} = r e^{\pm j\theta}$	$0 < \omega < \pi$	Bandpass

and they depend on the location of the poles on the complex plane.

We concentrate on the fourth case of complex conjugate poles, which is of greatest interest. The other three cases are explored in Problem 4.15. The spectrum is given by

$$R(e^{j\omega}) = \frac{d_0^2}{[1 - 2r \cos(\omega - \theta) + r^2][1 - 2r \cos(\omega + \theta) + r^2]} \quad (4.2.88)$$

The peak of this spectrum can be shown to be located at a frequency ω_c , given by

$$\cos \omega_c = \frac{1+r^2}{2r} \cos \theta \quad (4.2.89)$$

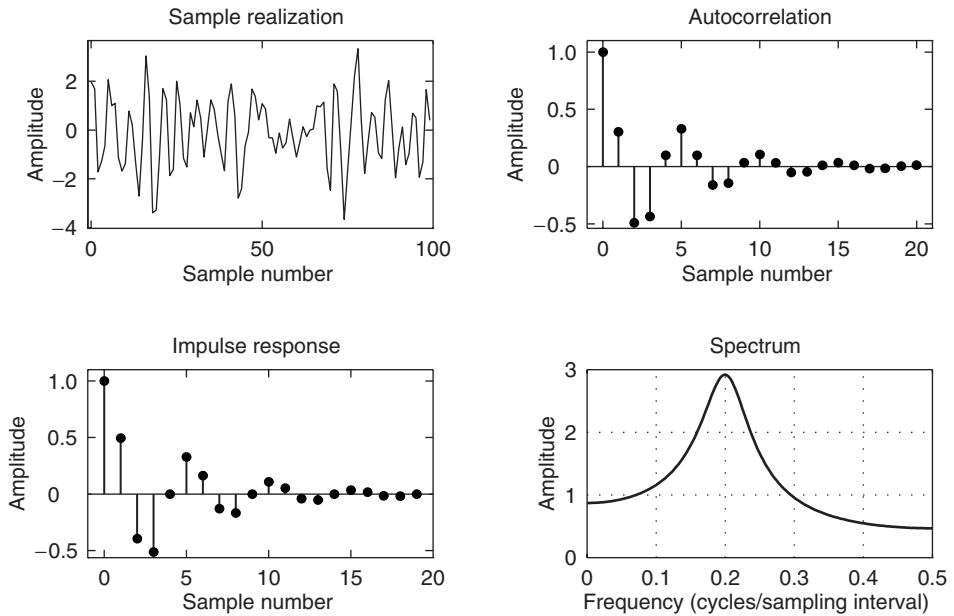
Since $1+r^2 > 2r$ for $r < 1$, and we have

$$\cos \omega_c > \cos \theta \quad (4.2.90)$$

the spectral peak is lower than the pole frequency for $0 < \theta < \pi/2$ and higher than the pole frequency for $\pi/2 < \theta < \pi$.

This behavior is illustrated in Figure 4.7 for an AP(2) model with $a_1 = -0.4944$, $a_2 = 0.64$, and $d_0 = 1$. The model has two complex conjugate poles with $r = 0.8$ and $\theta = \pm 2\pi/5$. The spectrum has a single peak and displays a passband type of behavior. The impulse response is a damped sine wave while the autocorrelation is a damped cosine. The typical realization of the output shows clearly a pseudoperiodic behavior that is explained by the shape of the autocorrelation and the spectrum of the model. We also notice that if the poles are complex conjugates, the autocorrelation has pseudoperiodic behavior.

Equivalent model descriptions. We now write explicit formulas for a_1 and a_2 in terms of the lattice parameters k_1 and k_2 and the autocorrelation coefficients. From the step-up

**FIGURE 4.7**

Sample realization of the output process, impulse response, autocorrelation, and spectrum of an AP(2) model with complex conjugate poles.

and step-down recursions in Section 2.5, we have

$$\begin{aligned} a_1 &= k_1(1 + k_2) \\ a_2 &= k_2 \end{aligned} \quad (4.2.91)$$

and the inverse relations

$$\begin{aligned} k_1 &= \frac{a_1}{1 + a_2} \\ k_2 &= a_2 \end{aligned} \quad (4.2.92)$$

From the Yule-Walker equations (4.2.18), we can write the two equations

$$\begin{aligned} a_1 r(0) + a_2 r(1) &= -r(1) \\ a_1 r(1) + a_2 r(0) &= -r(2) \end{aligned} \quad (4.2.93)$$

which can be solved for a_1 and a_2 in terms of $\rho(1)$ and $\rho(2)$

$$\begin{aligned} a_1 &= -\rho(1) \frac{1 - \rho(2)}{1 - \rho^2(1)} \\ a_2 &= \frac{\rho^2(1) - \rho(2)}{1 - \rho^2(1)} \end{aligned} \quad (4.2.94)$$

or for $\rho(1)$ and $\rho(2)$ in terms of a_1 and a_2

$$\begin{aligned} \rho(1) &= -\frac{a_1}{1 + a_2} \\ \rho(2) &= -a_1 \rho(1) - a_2 = \frac{a_1^2}{1 + a_2} - a_2 \end{aligned} \quad (4.2.95)$$

From the equations above, we can also write the relation and inverse relation between the

coefficients k_1 and k_2 and the normalized autocorrelations $\rho(1)$ and $\rho(2)$ as

$$\begin{aligned} k_1 &= -\rho(1) \\ k_2 &= \frac{\rho^2(1) - \rho(2)}{1 - \rho^2(1)} \end{aligned} \quad (4.2.96)$$

and

$$\begin{aligned} \rho(1) &= -k_1 \\ \rho(2) &= k_1(1 + k_2) - k_2 \end{aligned} \quad (4.2.97)$$

The gain d_0 can also be written in terms of the other coefficients. From (4.2.20), we have

$$d_0^2 = r(0)[1 + a_1\rho(1) + a_2\rho(2)] \quad (4.2.98)$$

which can be shown to be equal to

$$d_0^2 = r(0)(1 - k_1)(1 - k_2) \quad (4.2.99)$$

Minimum-phase conditions. In (4.2.71), we have a set of conditions on a_1 and a_2 so that the AP(2) model is minimum-phase, and Figure 4.6 shows the corresponding *admissible* region for minimum-phase models. Similar relations and regions can be derived for the other types of parameters, as we will show below. In terms of k_1 and k_2 , the AP(2) model is minimum-phase if

$$|k_1| < 1 \quad |k_2| < 1 \quad (4.2.100)$$

This region is depicted in Figure 4.8(a). Shown also is the region that results in complex roots, which is specified by

$$0 < k_2 < 1 \quad (4.2.101)$$

$$k_1^2 < \frac{4k_2}{(1 + k_2)^2} \quad (4.2.102)$$

Because of the correlation matching property of all-pole models, we can find a minimum-phase all-pole model for every positive definite sequence of autocorrelation values. Therefore, the admissible region of autocorrelation values coincides with the positive definite region. The positive definite condition is equivalent to having all the principal minors of the autocorrelation matrix in (4.2.30) be positive definite; that is, the corresponding determinants are positive. For $P = 2$, there are two conditions:

$$\det \begin{bmatrix} 1 & \rho(1) \\ \rho(1) & 1 \end{bmatrix} > 0 \quad \det \begin{bmatrix} 1 & \rho(1) & \rho(2) \\ \rho(1) & 1 & \rho(1) \\ \rho(2) & \rho(1) & 1 \end{bmatrix} > 0 \quad (4.2.103)$$

These two conditions reduce to

$$|\rho(1)| < 1 \quad (4.2.104)$$

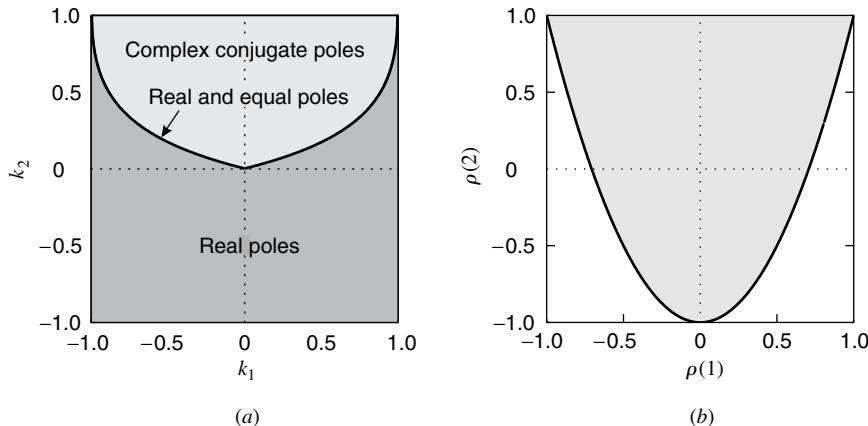
$$2\rho^2(1) - 1 < \rho(2) < 1 \quad (4.2.105)$$

which determine the admissible region shown in Figure 4.8(b). Conditions (4.2.105) can also be derived from (4.2.71) and (4.2.95). The first condition in (4.2.105) is equivalent to

$$\left| \frac{a_1}{1 + a_2} \right| < 1 \quad (4.2.106)$$

which can be shown to be equivalent to the last two conditions in (4.2.71).

It is important to note that the region in Figure 4.8(b) is the admissible region for *any* positive definite autocorrelation, including the autocorrelation of mixed-phase signals. This is reasonable since the autocorrelation does not contain phase information and allows the

**FIGURE 4.8**

Minimum-phase and positive definiteness regions for the AP(2) model in the (a) (k_1, k_2) space and (b) $(\rho(1), \rho(2))$ space.

signal to have minimum- and maximum-phase components. What we are claiming here, however, is that for every autocorrelation sequence in the positive definite region, we can find a minimum-phase all-pole model with the same autocorrelation values. Therefore, for this problem, the positive definite region is identical to the admissible minimum-phase region.

4.3 ALL-ZERO MODELS

In this section, we investigate the properties of the all-zero model. The output of the all-zero model is the weighted average of delayed versions of the input signal

$$x(n) = \sum_{k=0}^Q d_k w(n-k) \quad (4.3.1)$$

where Q is the order of the model. The system function is

$$H(z) = D(z) = \sum_{k=0}^Q d_k z^{-k} \quad (4.3.2)$$

The all-zero model can be implemented by using either a direct or a lattice structure. The conversion between the two sets of parameters can be done by using the step-up and step-down recursions described in Chapter 7 and setting $A(z) = D(z)$. Notice that the same set of parameters can be used to implement either an all-zero or an all-pole model by using a different structure.

4.3.1 Model Properties

We next provide a brief discussion of the properties of the all-zero model.

Impulse response. It can be easily seen that the $AZ(Q)$ model is an FIR system with an impulse response

$$h(n) = \begin{cases} d_n & 0 \leq n \leq Q \\ 0 & \text{elsewhere} \end{cases} \quad (4.3.3)$$

Autocorrelation. The autocorrelation of the impulse response is given by

$$r_h(l) = \sum_{n=-\infty}^{\infty} h(n)h^*(n-l) = \begin{cases} \sum_{k=0}^{Q-l} d_k d_{k+l}^* & 0 \leq l \leq Q \\ 0 & l > Q \end{cases} \quad (4.3.4)$$

and

$$r_h^*(-l) = r_h(l) \quad \text{all } l \quad (4.3.5)$$

We usually set $d_0 = 1$, which implies that

$$r_h(l) = d_l^* + d_1 d_{l+1}^* + \cdots + d_{Q-l} d_Q^* \quad l = 0, 1, \dots, Q \quad (4.3.6)$$

hence, the normalized autocorrelation is

$$\rho_h(l) = \begin{cases} \frac{d_l^* + d_1 d_{l+1}^* + \cdots + d_{Q-l} d_Q^*}{1 + |d_1|^2 + \cdots + |d_Q|^2} & l = 1, 2, \dots, Q \\ 0 & l > Q \end{cases} \quad (4.3.7)$$

We see that the autocorrelation of an AZ(Q) model is zero for lags $|l|$ exceeding the order Q of the model. If $\rho_h(1), \rho_h(2), \dots, \rho_h(Q)$ are known, then the Q equations (4.3.7) can be solved for model parameters d_1, d_2, \dots, d_q . However, unlike the Yule-Walker equations for the AP(P) model, which are linear, Equations (4.3.7) are nonlinear and their solution is quite complicated (see Section 9.3).

Spectrum. The spectrum of the AZ(Q) model is given by

$$R_h(e^{j\omega}) = D(z)D(z^{-1})|_{z=e^{j\omega}} = |D(e^{j\omega})|^2 = \sum_{l=-Q}^Q r_h(l)e^{-j\omega l} \quad (4.3.8)$$

which is basically a trigonometric polynomial.

Impulse train excitations. The response $\tilde{h}(n)$ of the AZ(Q) model to a periodic impulse train with period L is periodic with the same period, and its spectrum is a sampled version of (4.3.8) at multiples of $2\pi/L$ (see Section 2.3.2). Therefore, to recover the autocorrelation $r_h(l)$ and the spectrum $R_h(e^{j\omega})$ from the autocorrelation or spectrum of $\tilde{h}(n)$, we should have $L \geq 2Q + 1$ in order to avoid aliasing in the autocorrelation lag domain. Also, if $L > Q$, the impulse response $h(n)$, $0 \leq n \leq Q$, can be recovered from the response $\tilde{h}(n)$ (no time-domain aliasing) (see Problem 4.24).

Partial autocorrelation and lattice-ladder structures. The PACS of an AZ(Q) model is computed by fitting a series of AP(P) models for $P = 1, 2, \dots$, to the autocorrelation sequence (4.3.7) of the AZ(Q) model. Since the AZ(Q) model is equivalent to an AP(∞) model, the PACS of an all-zero model has infinite extent and behaves as the autocorrelation sequence of an all-pole model. This is illustrated later for the low-order AZ(1) and AZ(2) models.

4.3.2 Moving-Average Models

A moving-average model is an AZ(Q) model with $d_0 = 1$ driven by white noise, that is,

$$x(n) = w(n) + \sum_{k=1}^Q d_k w(n-k) \quad (4.3.9)$$

where $\{w(n)\} \sim \text{WN}(0, \sigma_w^2)$. The output $x(n)$ has zero mean and variance of

$$\sigma_x^2 = \sigma_w^2 \sum_{k=0}^Q |d_k|^2 \quad (4.3.10)$$

The autocorrelation and power spectrum are given by $r_x(l) = \sigma_w^2 r_h(l)$ and $R_x(e^{j\omega}) = \sigma_w^2 |D(e^{j\omega})|^2$, respectively. Clearly, observations that are more than Q samples apart are uncorrelated because the autocorrelation is zero after lag Q .

4.3.3 Lower-Order Models

To familiarize ourselves with all-zero models, we next investigate in detail the properties of the AZ(1) and AZ(2) models with real coefficients.

The first-order all-zero model: AZ(1). For generality, we consider an AZ(1) model whose system function is

$$H(z) = G(1 + d_1 z^{-1}) \quad (4.3.11)$$

The model is stable for any value of d_1 and minimum-phase for $-1 < d_1 < 1$. The autocorrelation is the inverse z -transform of

$$R_h(z) = H(z)H(z^{-1}) = G^2[d_1 z + (1 + d_1^2) + d_1 z^{-1}] \quad (4.3.12)$$

Hence, $r_h(0) = G^2(1 + d_1^2)$, $r_h(1) = r_h(-1) = G^2 d_1$, and $r_h(l) = 0$ elsewhere. Therefore, the normalized autocorrelation is

$$\rho_h(l) = \begin{cases} 1 & l = 0 \\ \frac{d_1}{1 + d_1^2} & l = \pm 1 \\ 0 & |l| \geq 2 \end{cases} \quad (4.3.13)$$

The condition $-1 < d_1 < 1$ implies that $|\rho_h(1)| \leq \frac{1}{2}$ for a minimum-phase model. From $\rho_h(1) = d_1/(1 + d_1^2)$, we obtain the quadratic equation

$$\rho_h(1)d_1^2 - d_1 + \rho_h(1) = 0 \quad (4.3.14)$$

which has the following two roots:

$$d_1 = \frac{1 \pm \sqrt{1 - 4\rho_h^2(1)}}{2\rho_h(1)} \quad (4.3.15)$$

Since the product of the roots is 1, if d_1 is a root, then $1/d_1$ must also be a root. Hence, only one of these two roots can satisfy the minimum-phase condition $-1 < d_1 < 1$.

The spectrum is obtained by setting $z = e^{j\omega}$ in (4.3.12), or from (4.3.8)

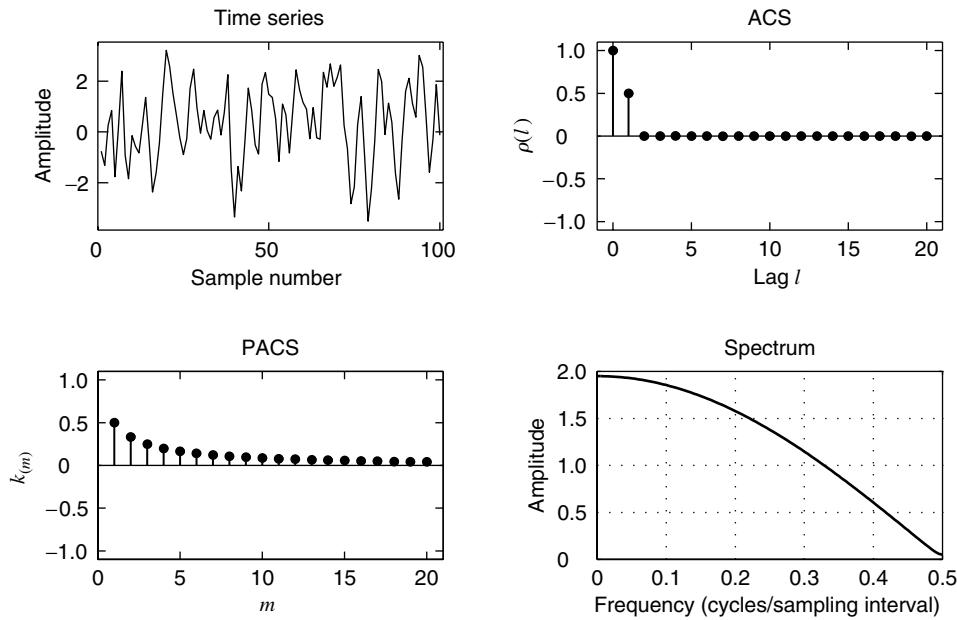
$$R_h(e^{j\omega}) = G^2(1 + d_1^2 + 2d_1 \cos \omega) \quad (4.3.16)$$

The autocorrelation is positive definite if $R_h(e^{j\omega}) > 0$, which holds for all values of d_1 . Note that if $d_1 > 0$, then $\rho_h(1) > 0$ and the spectrum has low-pass behavior (see Figure 4.9), whereas a high-pass spectrum is obtained when $d_1 < 0$ (see Figure 4.10).

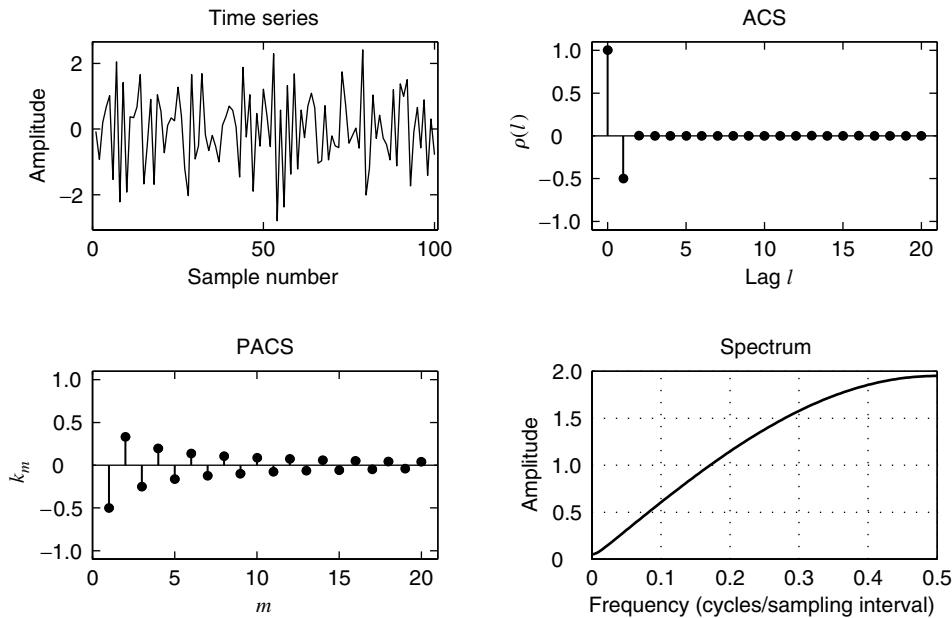
The first lattice parameter of the AZ(1) model is $k_1 = -\rho(1)$. The PACS can be obtained from the Yule-Walker equations by using the autocorrelation sequence (4.3.13). Indeed, after some algebra we obtain

$$k_m = \frac{(-d_1)^m (1 - d_1^2)}{1 - d_1^{2(m+1)}} \quad m = 1, 2, \dots, \infty \quad (4.3.17)$$

(see Problem 4.25). Notice the duality between the ACS and PACS of AP(1) and AZ(1) models.

**FIGURE 4.9**

Sample realization of the output process, ACS, PACS, and spectrum of an AZ(1) model with $d_1 = 0.95$.

**FIGURE 4.10**

Sample realization of the output process, ACS, PACS, and spectrum of an AZ(1) model with $d_1 = -0.95$.

Consider now the MA(1) real-valued process $x(n)$ generated by

$$x(n) = w(n) + bw(n-1)$$

where $\{w(n)\} \sim \text{WN}(0, \sigma_w^2)$. Using $R_x(z) = \sigma_w^2 H(z)H(z^{-1})$, we obtain the PSD function

$$R_x(e^{j\omega}) = \sigma_w^2(1 + b^2 + 2b \cos \omega)$$

which has low-pass (high-pass) characteristics if $0 < b \leq 1$ ($-1 \leq b < 0$). Since $\sigma_x^2 = r_x(0) = \sigma_w^2(1 + b^2)$, we have (see Section 4.1.18)

$$\text{SFM}_x = \frac{\sigma_w^2}{\sigma_x^2} = \frac{1}{1 + b^2} \quad (4.3.18)$$

which is maximum for $b = 0$ (white noise). The correlation matrix is banded (only a number of diagonals close to the main diagonal are nonzero)

$$\mathbf{R}_x = \sigma_w^2(1 + b^2) \begin{bmatrix} 1 & b & 0 & \cdots & 0 \\ b & 1 & b & \cdots & 0 \\ 0 & b & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \quad (4.3.19)$$

and its eigenvalues and eigenvectors are given by $\lambda_k = R_x(e^{j\omega_k})$, $q_n^{(k)} = \sin \omega_k n$, $\omega_k = \pi k / (M + 1)$, where $k = 1, 2, \dots, M$ (see Problem 4.30).

The second-order all-zero model: AZ(2). Now let us consider the second-order all-zero model. The system function of the AZ(2) model is

$$H(z) = G(1 + d_1 z^{-1} + d_2 z^{-2}) \quad (4.3.20)$$

The system is stable for all values of d_1 and d_2 , and minimum-phase [see the discussion for the AP(2) model] if

$$\begin{aligned} -1 &< d_2 < 1 \\ d_2 - d_1 &> -1 \\ d_2 + d_1 &> -1 \end{aligned} \quad (4.3.21)$$

which is a triangular region identical to that shown in Figure 4.6. The normalized autocorrelation and the spectrum are

$$\rho_h(l) = \begin{cases} 1 & l = 0 \\ \frac{d_1(1 + d_2)}{1 + d_1^2 + d_2^2} & l = \pm 1 \\ \frac{d_2}{1 + d_1^2 + d_2^2} & l = \pm 2 \\ 0 & |l| \geq 3 \end{cases} \quad (4.3.22)$$

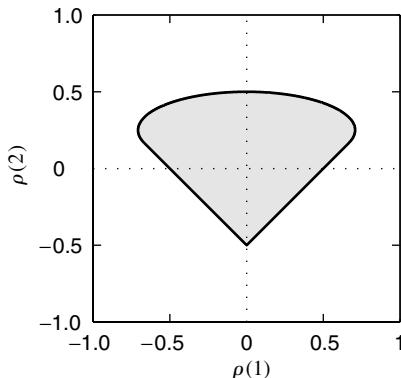
$$\text{and } R_h(e^{j\omega}) = G^2[(1 + d_1^2 + d_2^2) + 2d_1(1 + d_2)\cos\omega + 2d_2\cos 2\omega] \quad (4.3.23)$$

respectively.

The minimum-phase region in the autocorrelation domain is shown in Figure 4.11 and is described by the equations

$$\begin{aligned} \rho(2) + \rho(1) &= -0.5 \\ \rho(2) - \rho(1) &= -0.5 \\ \rho^2(1) &= 4\rho(2)[1 - 2\rho(2)] \end{aligned} \quad (4.3.24)$$

derived in Problem 4.26. The formula for the PACS is quite involved. The important thing is the duality between the ACS and the PACS of AZ(2) and AP(2) models (see Problem 4.27).

**FIGURE 4.11**

Minimum-phase region in the autocorrelation domain for the AZ(2) model.

4.4 POLE-ZERO MODELS

We will focus on causal pole-zero models with a recursive input-output relationship given by

$$x(n) = -\sum_{k=1}^P a_k x(n-k) + \sum_{k=0}^Q d_k w(n-k) \quad (4.4.1)$$

where we assume that $P > 0$ and $Q \geq 1$. The models can be implemented using either direct-form or lattice-ladder structures (Proakis and Manolakis 1996).

4.4.1 Model Properties

In this section, we present some of the basic properties of pole-zero models.

Impulse response. The impulse response of a causal pole-zero model can be written in recursive form from (4.4.1) as

$$h(n) = -\sum_{k=1}^P a_k h(n-k) + d_n \quad n \geq 0 \quad (4.4.2)$$

where

$$d_n = 0 \quad n > Q$$

and $h(n) = 0$ for $n < 0$. Clearly, this formula is useful if the model is stable. From (4.4.2), it is clear that

$$h(n) = -\sum_{k=1}^P a_k h(n-k) \quad n > Q \quad (4.4.3)$$

so that the impulse response obeys the linear prediction equation for $n > Q$. Thus if we are given $h(n)$, $0 \leq n \leq P + Q$, we can compute $\{a_k\}$ from (4.4.3) by using the P equations specified by $Q + 1 \leq n \leq Q + P$. Then we can compute $\{d_k\}$ from (4.4.2), using $0 \leq n \leq Q$. Therefore, the first $P + Q + 1$ values of the impulse response completely specify the pole-zero model.

If the model is minimum-phase, the impulse response of the inverse model $h_I(n) = \mathcal{Z}^{-1}\{A(z)/D(z)\}$, $d_0 = 1$ can be computed in a similar manner.

Autocorrelation. The complex spectrum of $H(z)$ is given by

$$R_h(z) = H(z)H^*\left(\frac{1}{z^*}\right) = \frac{D(z)D^*(1/z^*)}{A(z)A^*(1/z^*)} \triangleq \frac{R_d(z)}{R_a(z)} \quad (4.4.4)$$

where $R_d(z)$ and $R_a(z)$ are both finite two-sided polynomials. In a manner similar to the

all-pole case, we can write a recursive relation between the autocorrelation, impulse response, and parameters of the model. Indeed, from (4.4.4) we obtain

$$A(z)R_h(z) = D(z)H^*\left(\frac{1}{z^*}\right) \quad (4.4.5)$$

Taking the inverse z -transform of (4.4.5) and noting that the inverse z -transform of $H^*(1/z^*)$ is $h^*(-n)$, we have

$$\sum_{k=0}^P a_k r_h(l-k) = \sum_{k=0}^Q d_k h^*(k-l) \quad \text{for all } l \quad (4.4.6)$$

Since $h(n)$ is causal, we see that the right-hand side of (4.4.6) is zero for $l > Q$:

$$\sum_{k=0}^P a_k r_h(l-k) = 0 \quad l > Q \quad (4.4.7)$$

Therefore, the autocorrelation of a pole-zero model obeys the linear prediction equation for $l > Q$.

Because the impulse response $h(n)$ is a function of a_k and d_k , the set of equations in (4.4.6) is nonlinear in terms of parameters a_k and d_k . However, (4.4.7) is linear in a_k ; therefore, we can compute $\{a_k\}$ from (4.4.7), using the set of equations for $l = Q + 1, \dots, Q + P$, which can be written in matrix form as

$$\begin{bmatrix} r_h(Q) & r_h(Q-1) & \cdots & r_h(Q-P+1) \\ r_h(Q+1) & r_h(Q) & \cdots & r_h(Q-P+2) \\ \vdots & \vdots & \ddots & \vdots \\ r_h(Q+P-1) & r_h(Q+P-2) & \cdots & r_h(Q) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_P \end{bmatrix} = - \begin{bmatrix} r_h(Q+1) \\ r_h(Q+2) \\ \vdots \\ r_h(Q+P) \end{bmatrix} \quad (4.4.8)$$

or

$$\bar{\mathbf{R}}_h \mathbf{a} = -\bar{\mathbf{r}}_h \quad (4.4.9)$$

Here, $\bar{\mathbf{R}}_h$ is a non-Hermitian Toeplitz matrix, and the linear system (4.4.8) can be solved by using the algorithm of Trench (Trench 1964; Carayannis et al. 1981).

Even after we solve for \mathbf{a} , (4.4.6) continues to be nonlinear in d_k . To compute d_k , we use (4.4.4) to find $R_d(z)$

$$R_d(z) = R_a(z)R_h(z) \quad (4.4.10)$$

where the coefficients of $R_a(z)$ are given by

$$r_a(l) = \sum_{k=k_1}^{k=k_2} a_k a_{k+|l|}^* \quad -P \leq l \leq P, \quad k_1 = \begin{cases} 0, & l \geq 0 \\ -l, & l < 0 \end{cases}, \quad k_2 = \begin{cases} P-l, & l \geq 0 \\ P, & l < 0 \end{cases} \quad (4.4.11)$$

From (4.4.10), $r_d(l)$ is the convolution of $r_a(l)$ with $r_h(l)$, given by

$$r_d(l) = \sum_{k=-P}^P r_a(k)r_h(l-k) \quad (4.4.12)$$

If $r(l)$ was originally the autocorrelation of a PZ(P, Q) model, then $r_d(l)$ in (4.4.12) will be zero for $|l| > Q$. Since $R_d(z)$ is specified, it can be factored into the product of two polynomials $D(z)$ and $D^*(1/z^*)$, where $D(z)$ is minimum-phase, as shown in Section 2.4.

Therefore, we have seen that, given the values of the autocorrelation $r_h(l)$ of a PZ(P, Q) model in the range $0 \leq l \leq P + Q$, we can compute the values of the parameters $\{a_k\}$ and $\{d_k\}$ such that $H(z)$ is minimum-phase. Now, given the parameters of a pole-zero model, we can compute its autocorrelation as follows. Equation (4.4.4) can be written as

$$R_h(z) = R_a^{-1}(z)R_d(z) \quad (4.4.13)$$

where $R_a^{-1}(z)$ is the spectrum of the all-pole model $1/A(z)$, that is, $1/R_a(z)$. The coefficients of $R_a^{-1}(z)$ can be computed from $\{a_k\}$ by using (4.2.20) and (4.2.18). The coefficients of $R_d(z)$ are computed from (4.3.8). Then $R_h(z)$ is the convolution of the two autocorrelations thus computed, which is equivalent to multiplying the two polynomials in (4.4.13) and equating equal powers of z on both sides of the equation. Since $R_d(z)$ is finite, the summations used to obtain the coefficients of $R_h(z)$ are also finite.

EXAMPLE 4.4.1. Consider a signal that has autocorrelation values of $r_h(0) = 19$, $r_h(1) = 9$, $r_h(2) = -5$, and $r_h(3) = -7$. The parameters of the PZ(2, 1) model are found in the following manner. First form the equation from (4.4.8)

$$\begin{bmatrix} 9 & 19 \\ -5 & 9 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$$

which yields $a_1 = -\frac{1}{2}$, $a_2 = \frac{1}{2}$. Then we compute the coefficients from (4.4.11), $r_a(0) = \frac{3}{2}$, $r_a(\pm 1) = -\frac{3}{4}$, and $r_a(\pm 2) = \frac{1}{2}$. Computing the convolution in (4.4.12) for $l \leq Q = 1$, we obtain the following polynomial:

$$R_d(z) = 4z + 10 + 4z^{-1} = 4 \left(1 + \frac{z^{-1}}{2}\right)(z + 2)$$

Therefore, $D(z)$ is obtained by taking the causal part, that is, $D(z) = 2[1 + z^{-1}/(2)]$, and $d_1 = \frac{1}{2}$.

Spectrum. The spectrum of $H(z)$ is given by

$$R_h(e^{j\omega}) = |H(e^{j\omega})|^2 = \frac{|D(e^{j\omega})|^2}{|A(e^{j\omega})|^2} \quad (4.4.14)$$

Therefore, $R_h(e^{j\omega})$ can be obtained by dividing the spectrum of $D(z)$ by the spectrum of $A(z)$. Again, the FFT can be used to advantage in computing the numerator and denominator of (4.4.14). If the spectrum $R_h(e^{j\omega})$ of a PZ(P, Q) model is given, then the parameters of the (minimum-phase) model can be recovered by first computing the autocorrelation $r_h(l)$ as the inverse Fourier transform of $R_h(e^{j\omega})$ and then using the procedure outlined in the previous section to compute the sets of coefficients $\{a_k\}$ and $\{d_k\}$.

Partial autocorrelation and lattice-ladder structures. Since a PZ(P, Q) model is equivalent to an AP(∞) model, its PACS has infinite extent and behaves, after a certain lag, as the PACS of an all-zero model.

4.4.2 Autoregressive Moving-Average Models

The autoregressive moving-average model is a PZ(P, Q) model driven by white noise and is denoted by ARMA(P, Q). Again, we set $d_0 = 1$ and incorporate the gain into the variance (power) of the white noise excitation. Hence, a causal ARMA(P, Q) model is defined by

$$x(n) = -\sum_{k=1}^P a_k x(n-k) + w(n) + \sum_{k=1}^Q d_k w(n-k) \quad (4.4.15)$$

where $\{w(n)\} \sim \text{WN}(0, \sigma_w^2)$. The ARMA(P, Q) model parameters are $\{\sigma_w^2, a_1, \dots, a_P, d_1, \dots, d_Q\}$. The output has zero mean and variance of

$$\sigma_x^2 = -\sum_{k=1}^P a_k r_x(k) + \sigma_w^2 [1 + \sum_{k=1}^Q d_k h(k)] \quad (4.4.16)$$

where $h(n)$ is the impulse response of the model. The presence of $h(n)$ in (4.4.16) makes the dependence of σ_x^2 on the model parameters highly nonlinear. The autocorrelation of

$x(n)$ is given by

$$\sum_{k=0}^P a_k r_x(l-k) = \sigma_w^2 \left[1 + \sum_{k=1}^Q d_k h(k-l) \right] \quad \text{for all } l \quad (4.4.17)$$

and the power spectrum by

$$R_x(e^{j\omega}) = \sigma_w^2 \frac{|D(e^{j\omega})|^2}{|A(e^{j\omega})|^2} \quad (4.4.18)$$

The significance of ARMA(P, Q) models is that they can provide more accurate representations than AR or MA models with the same number of parameters. *The ARMA model is able to combine the spectral peak matching of the AR model with the ability of the MA model to place nulls in the spectrum.*

4.4.3 The First-Order Pole-Zero Model: PZ(1, 1)

Consider the PZ(1, 1) model with the following system function

$$H(z) = G \frac{1 + d_1 z^{-1}}{1 + a_1 z^{-1}} \quad (4.4.19)$$

where d_1 and a_1 are real coefficients. The model is minimum-phase if

$$\begin{aligned} -1 < d_1 < 1 \\ -1 < a_1 < 1 \end{aligned} \quad (4.4.20)$$

which correspond to the rectangular region shown in Figure 4.12(a).

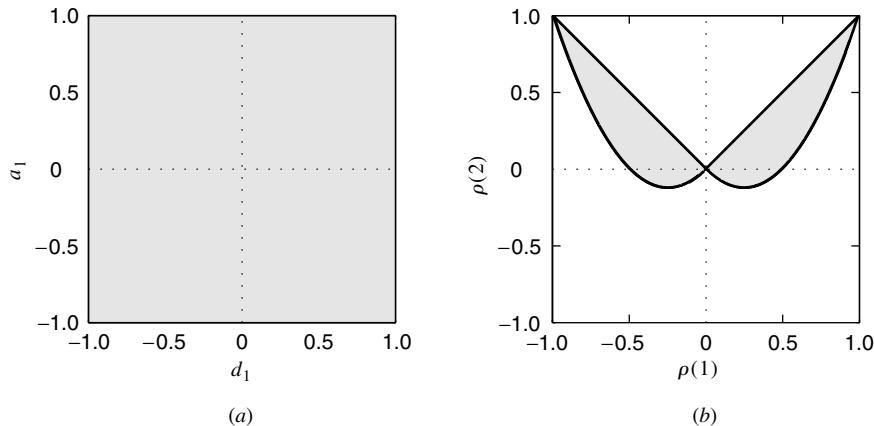


FIGURE 4.12

Minimum-phase and positive definiteness regions for the PZ(1, 1) model in the (a) (d_1, a_1) space and (b) $(\rho(1), \rho(2))$ space.

For the minimum-phase case, the impulse responses of the direct and the inverse models are

$$h(n) = \mathcal{Z}^{-1}\{H(z)\} = \begin{cases} 0 & n < 0 \\ G & n = 0 \\ G(-a_1)^{n-1}(d_1 - a_1) & n > 0 \end{cases} \quad (4.4.21)$$

$$\text{and } h_I(n) = \mathcal{Z}^{-1} \left\{ \frac{1}{H(z)} \right\} = \begin{cases} 0 & n < 0 \\ G & n = 0 \\ G(-d_1)^{n-1}(a_1 - d_1) & n > 0 \end{cases} \quad (4.4.22)$$

respectively. We note that as the pole $p = -a_1$ gets closer to the unit circle, the impulse response decays more slowly and the model has “longer memory.” The zero $z = -d_1$ controls the impulse response of the inverse model in a similar way. The PZ(1, 1) model is equivalent to the AZ(∞) model

$$x(n) = Gw(n) + G \sum_{k=1}^{\infty} h(k)w(n-k) \quad (4.4.23)$$

or the AP(∞) model

$$x(n) = - \sum_{k=1}^{\infty} h_I(k)x(n-k) + Gw(n) \quad (4.4.24)$$

If we wish to approximate the PZ(1, 1) model with a finite-order AZ(Q) model, the order Q required to achieve a certain accuracy increases as the pole moves closer to the unit circle. Likewise, in the case of an AP(P) approximation, better fits to the PZ(P, Q) model require an increased order P as the zero moves closer to the unit circle.

To determine the autocorrelation, we recall from (4.4.6) that for a causal model

$$r_h(l) = -a_1 r_h(l-1) + Gh(-l) + Gd_1 h(1-l) \quad \text{all } l \quad (4.4.25)$$

or

$$\begin{aligned} r_h(0) &= -a_1 r_h(1) + G + Gd_1(d_1 - a_1) \\ r_h(1) &= -a_1 r_h(0) + Gd_1 \\ r_h(l) &= -a_1 r_h(l-1) \quad l \geq 2 \end{aligned} \quad (4.4.26)$$

Solving the first two equations for $r_h(0)$ and $r_h(1)$, we obtain

$$r_h(0) = G \frac{1 + d_1^2 - 2a_1 d_1}{1 - a_1^2} \quad (4.4.27)$$

and

$$r_h(1) = G \frac{(d_1 - a_1)(1 - a_1 d_1)}{1 - a_1^2} \quad (4.4.28)$$

The normalized autocorrelation is given by

$$\rho_h(1) = \frac{(d_1 - a_1)(1 - a_1 d_1)}{1 + d_1^2 - 2a_1 d_1} \quad (4.4.29)$$

and

$$\rho_h(l) = (-a_1)^{l-1} \rho_h(l-1) \quad l \geq 2 \quad (4.4.30)$$

Note that given $\rho_h(1)$ and $\rho_h(2)$, we have a nonlinear system of equations that must be solved to obtain a_1 and d_1 . By using Equations (4.4.20), (4.4.29), and (4.4.30), it can be shown (see Problem 4.28) that the PZ(1, 1) is minimum-phase if the ACS satisfies the conditions

$$\begin{aligned} |\rho(2)| &< |\rho(1)| \\ \rho(2) &> \rho(1)[2\rho(1) + 1] \quad \rho(1) < 0 \\ \rho(2) &> \rho(1)[2\rho(1) - 1] \quad \rho(1) > 0 \end{aligned} \quad (4.4.31)$$

which correspond to the admissible region shown in Figure 4.12(b).

4.4.4 Summary and Dualities

Table 4.1 summarizes the key properties of all-zero, all-pole, and pole-zero models. These properties help to identify models for empirical discrete-time signals. Furthermore, the table shows the duality between AZ and AP models. More specifically, we see that

1. An invertible AZ(Q) model is equivalent to an AP(∞) model. Thus, it has a finite-extent autocorrelation and an infinite-extent partial autocorrelation.
2. A stable AP(P) model is equivalent to an AZ(∞) model. Thus, it has an infinite-extent autocorrelation and a finite-extent partial autocorrelation.
3. The autocorrelation of an AZ(Q) model behaves as the partial autocorrelation of an AP(P) model, and vice versa.
4. The spectra of an AP(P) model and an AZ(Q) model are related through an inverse relationship.

TABLE 4.1
Summary of all-pole, all-zero, and pole-zero model properties

Model	AP(P)	AZ(Q)	PZ(P, Q)
Input-output description	$x(n) + \sum_{k=1}^P a_k x(n-k) = w(n)$	$x(n) = d_0 w(n) + \sum_{k=1}^Q d_k w(n-k)$	$x(n) + \sum_{k=1}^P a_k x(n-k) = d_0 w(n) + \sum_{k=1}^Q d_k w(n-k)$
System function	$H(z) = \frac{1}{A(z)} = \frac{d_0}{1 + \sum_{k=1}^P a_k z^{-k}}$	$H(z) = D(z) = d_0 + \sum_{k=1}^Q d_k z^{-k}$	$H(z) = \frac{D(z)}{A(z)}$
Recursive representation	Finite summation	Infinite summation	Infinite summation
Nonrecursive representation	Infinite summation	Finite summation	Infinite summation
Stability conditions	Poles inside unit circle	Always	Poles inside unit circle
Invertibility conditions	Always	Zeros inside unit circle	Zeros inside unit circle
Autocorrelation sequence	Infinite duration (damped exponentials and/or sine waves)	Finite duration	Infinite duration (damped exponentials and/or sine waves after $Q - P$ lags)
	Tails off	Cuts off	Tails off
Partial autocorrelation	Finite duration	Infinite duration (damped exponentials and/or sine waves)	Infinite duration (dominated by damped exponentials and/or sine waves after $Q - P$ lags)
	Cuts off	Tails off	Tails off
Spectrum	Good peak matching	Good “notch” matching	Good peak and valley matching

These dualities and properties have been shown and illustrated for low-order models in the previous sections.

4.5 MODELS WITH POLES ON THE UNIT CIRCLE

In this section, we show that by restricting some poles to being on the unit circle, we obtain models that are useful for modeling certain types of nonstationary behavior.

Pole-zero models with poles on the unit circle are unstable. Hence, if we drive them with stationary white noise, the generated process is nonstationary. However, as we will see in the sequel, placing a small number of real poles at $z = 1$ or complex conjugate poles at $z_k = e^{\pm j\theta_k}$ provides a class of models useful for modeling certain types of nonstationary behavior. The system function of a pole-zero model with d poles at $z = 1$, denoted as PZ(P, d, Q), is

$$H(z) = \frac{D(z)}{A(z)} \frac{1}{(1 - z^{-1})^d} \quad (4.5.1)$$

and can be viewed as $\text{PZ}(P, Q)$ model, $D(z)/A(z)$, followed by a d th-order accumulator. The accumulator $y(n) = y(n-1) + x(n)$ has the system function $1/(1 - z^{-1})$ and can be thought of as a discrete-time integrator. The presence of the unit poles makes the $\text{PZ}(P, d, Q)$ model non-minimum-phase. Since the model is unstable, we cannot use the convolution summation to represent it because, in practice, only finite-order approximations are possible. This can be easily seen if we recall that the impulse response of the model $\text{PZ}(0, d, 0)$ equals $u(n)$ for $d = 1$ and $(n+1)u(n)$ for $d = 2$. However, if $D(z)/A(z)$ is minimum-phase, the inverse model $H_I(z) = 1/H(z)$ is stable, and we can use the recursive form (see Section 4.1) to represent the model. Indeed, we always use this representation when we apply this model in practice.

The spectrum of the $\text{PZ}(0, d, 0)$ model is

$$R_d(e^{j\omega}) = \frac{1}{[2 \sin(\omega/2)]^{2d}} \quad (4.5.2)$$

and since $R_d(0) = \sum_{l=-\infty}^{\infty} r_d(l) = \infty$, the autocorrelation does not exist.

In the case of complex conjugate poles, the term $(1 - z^{-1})^d$ in (4.5.1) is replaced by $(1 - 2 \cos \theta_k z^{-1} + z^{-2})^d$, that is,

$$H(z) = \frac{D(z)}{A(z)} \frac{1}{(1 - 2 \cos \theta_k z^{-1} + z^{-2})^d} \quad (4.5.3)$$

The second term is basically a cascade of AP(2) models with complex conjugate poles on the unit circle. This model exhibits strong periodicity in its impulse response, and its “resonance-like” spectrum diverges at $\omega = \theta_k$.

With regard to the partial autocorrelation, we recall that the presence of poles on the unit circle results in some lattice parameters taking on the values ± 1 .

EXAMPLE 4.5.1. Consider the following causal $\text{PZ}(1, 1, 1)$ model

$$H(z) = \frac{1 + d_1 z^{-1}}{1 + a_1 z^{-1}} \frac{1}{1 - z^{-1}} = \frac{1 + d_1 z^{-1}}{1 - (1 - a_1)z^{-1} - a_1 z^{-2}} \quad (4.5.4)$$

with $-1 < a_1 < 1$ and $-1 < d_1 < 1$.

The difference equation representation of the model uses previous values of the output and the present and previous values of the input. It is given by

$$y(n) = (1 - a_1)y(n-1) + a_1y(n-2) + x(n) + d_1x(n-1) \quad (4.5.5)$$

To express the output in terms of the present and previous values of the input (nonrecursive representation), we find the impulse response of the model

$$h(n) = \mathcal{Z}^{-1}\{H(z)\} = A_1 u(n) + A_2(-a_1)^n u(n) \quad (4.5.6)$$

where $A_1 = (1 + d_1)/(1 + a_1)$ and $A_2 = (a_1 - d_1)/(1 + a_1)$. Note that the model is unstable, and it cannot be approximated by an FIR system because $h(n) \rightarrow A_1 u(n)$ as $n \rightarrow \infty$.

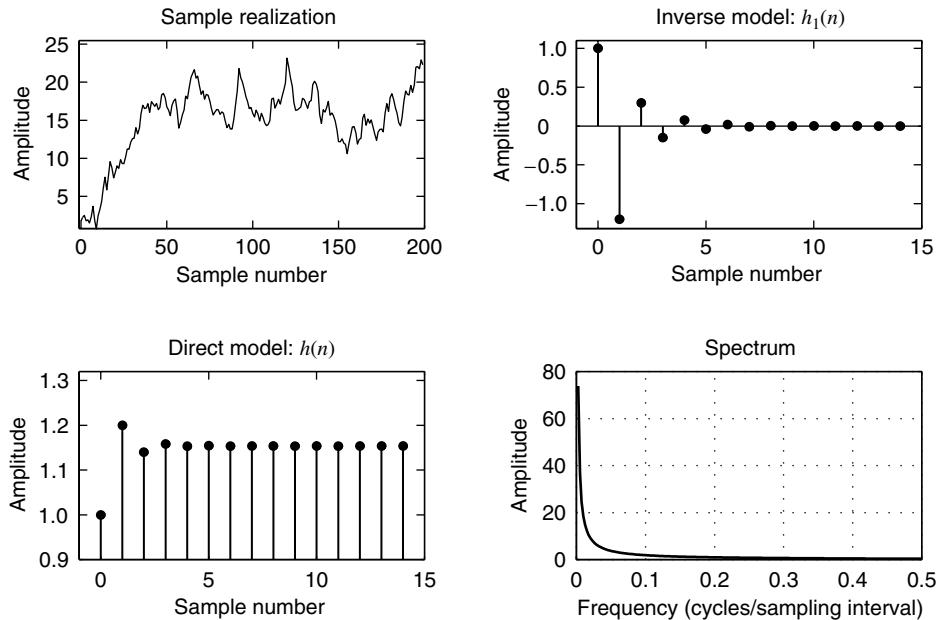
Finally, we can express the output as a weighted sum of previous outputs and the present input, using the impulse response of the inverse model $G(z) = 1/H(z)$

$$h_I(n) = \mathcal{Z}^{-1}\{H_I(z)\} = B_1 \delta(n) + B_2 \delta(n-1) + B_3(-d_1)^n u(n) \quad (4.5.7)$$

where $B_1 = (a_1 - d_1 + a_1 d_1)/d_1^2$, $B_2 = -a_1/d_1$, and $B_3 = (-a_1 + d_1 - a_1 d_1 + d_1^2)/d_1^2$. Since $-1 < d_1 < 1$, the sequence $h_I(n)$ decays at a rate governed by the value of d_1 . If $h_I(n) \simeq 0$ for $n \geq p_d$, the recursive formula

$$y(n) = - \sum_{k=1}^{p_d} h_I(k) y(n-k) + x(n) \quad (4.5.8)$$

provides a good representation of the $\text{PZ}(1, 1, 1)$ model. For example, if $a_1 = 0.3$ and $d_1 = 0.5$, we find that $|h_I(n)| \leq 0.0001$ for $n \geq 12$, which means that the current value of the model output can be computed with sufficient accuracy from the 12 most recent values of signal $y(n)$. This is illustrated in Figure 4.13, which also shows a realization of the output process if the model is driven by white Gaussian noise.

**FIGURE 4.13**

Sample realization of the output process, impulse response, impulse response of the inverse model, and spectrum of a PZ(1, 1, 1) model with $a_1 = 0.3$, $d_1 = 0.5$, and $d = 1$. The value $R(e^{j0}) = \infty$ is not plotted.

Autoregressive integrated moving-average models. In Section 3.3.2 we discussed discrete-time random signals with stationary increments. Clearly, driving a PZ(P, d, Q) model with white noise generates a random signal whose d th difference is a stationary ARMA(P, Q) process. Such time series are known in the statistical literature as *autoregressive integrated moving-average models*, denoted ARIMA(P, d, Q). They are useful in modeling signals with certain *stochastic trends* (e.g., random changes in the level and slope of the signal). Indeed, many empirical signals (e.g., infrared background measurements and stock prices) exhibit this type of behavior (see Figure 1.6). Notice that the ARIMA(0, 1, 0) process, that is, $x(n) = x(n - 1) + w(n)$, where $\{w(n)\} \sim WN(0, \sigma_w^2)$, is the discrete-time equivalent of the *random walk* or *Brownian motion* process (Papoulis 1991).

When the unit poles are complex conjugate, the model is known as a *harmonic PZ model*. This model produces random sequences that exhibit “random periodic behavior” and are known as *seasonal time series* in the statistical literature. Such signals repeat themselves cycle by cycle, but there is some randomness in both the length and the pattern of each cycle. The identification and estimation of ARIMA and seasonal models and their applications can be found in Box, Jenkins, and Reinsel (1994); Brockwell and Davis (1991); and Hamilton (1994).

4.6 CEPSTRUM OF POLE-ZERO MODELS

In this section we determine the cepstrum of pole-zero models and its properties, and we develop algorithms to convert between direct structure model parameters and cepstral coefficients. The cepstrum has been proved a valuable tool in speech coding and recognition applications and has been extensively studied in the corresponding literature (Rabiner and Schafer 1978; Rabiner and Juang 1993; Furui 1989). For simplicity, we consider models with real coefficients.

The cepstrum of the impulse response $h(n)$ of a pole-zero model is the inverse z -transform of

$$\log H(z) = \log D(z) - \log A(z) \quad (4.6.1)$$

$$= \log d_0 + \sum_{i=1}^Q \log (1 - z_i z^{-1}) - \sum_{i=1}^P \log (1 - p_i z^{-1}) \quad (4.6.2)$$

where $\{z_i\}$ and $\{p_i\}$ are the zeros and poles of $H(z)$, respectively. If we assume that $H(z)$ is minimum-phase and use the power series expansion

$$\log (1 - \alpha z^{-1}) = - \sum_{n=1}^{\infty} \frac{\alpha^n}{n} z^{-n} \quad |z| > |\alpha|$$

we find that the cepstrum $c(n)$ is given by

$$c(n) = \begin{cases} 0 & n < 0 \\ \log d_0 & n = 0 \\ \frac{1}{n} \left(\sum_{i=1}^P p_i^n - \sum_{i=1}^Q z_i^n \right) & n > 0 \end{cases} \quad (4.6.3)$$

Since the poles and zeros are assumed to be inside the unit circle, (4.6.3) implies that $c(n)$ is bounded by

$$-\frac{P+Q}{n} \leq c(n) \leq \frac{P+Q}{n} \quad (4.6.4)$$

with equality if and only if all the roots are appropriately at $z = 1$ or $z = -1$.

If $H(z)$ is minimum-phase, then there exists a unique mapping between the cepstrum and the impulse response, given by the recursive relations (Oppenheim and Schafer 1989)

$$\begin{aligned} c(0) &= \log h(0) = \log d_0 \\ c(n) &= \frac{h(n)}{h(0)} - \frac{1}{n} \sum_{m=0}^{n-1} m c(m) \frac{h(n-m)}{h(0)} \quad n > 0 \end{aligned} \quad (4.6.5)$$

and

$$\begin{aligned} h(0) &= e^{c(0)} \\ h(n) &= h(0)c(n) + \frac{1}{n} \sum_{m=0}^{n-1} m c(m)h(n-m) \quad n > 0 \end{aligned} \quad (4.6.6)$$

where we have assumed $d_0 > 0$ without loss of generality. Therefore, given the cepstrum $c(n)$ in the range $0 \leq n \leq P+Q$, we can completely recover the parameters of the pole-zero model as follows. From (4.6.6) we can compute $h(n)$, $0 \leq n \leq P+Q$, and from (4.4.2) and (4.4.3) we can recover $\{a_k\}$ and $\{d_k\}$.

4.6.2 All-Pole Models

The cepstrum of a minimum-phase all-pole model is given by (4.6.2) and (4.6.3) with $Q = 0$. Since $H(z)$ is minimum-phase, the cepstrum $c(n)$ of $1/A(z)$ is simply the negative of the cepstrum of $A(z)$, which can be written in terms of a_k (see also Problem 4.34). As a result, the cepstrum can be obtained from the direct-form coefficients by using the following

recursion

$$c(n) = \begin{cases} -a_n - \frac{1}{n} \sum_{k=1}^{n-1} (n-k) a_k c(n-k) & 1 \leq n \leq P \\ -\frac{1}{n} \sum_{k=1}^P (n-k) a_k c(n-k) & n > P \end{cases} \quad (4.6.7)$$

The inverse relation is

$$a_n = -c(n) - \frac{1}{n} \sum_{k=1}^{n-1} (n-k) a_k c(n-k) \quad n > 0 \quad (4.6.8)$$

which shows that the first P cepstral coefficients completely determine the model parameters (Furui 1981).

From (4.6.7) it is evident that the cepstrum generally decays as $1/n$. Therefore, it may be desirable sometimes to consider

$$c'(n) = nc(n) \quad (4.6.9)$$

which is known as the *ramp cepstrum* since it is obtained by multiplying the cepstrum by a ramp function. From (4.6.9) and (4.6.4), we note that the ramp cepstrum of an AP(P) model is bounded by

$$|c'(n)| \leq P \quad n > 0 \quad (4.6.10)$$

with equality if and only if all the poles are at $z = 1$ or $z = -1$. Also $c'(n)$ is equal to the negative of the inverse z -transform of the derivative of $\log H(z)$. From the preceding equations, we can write

$$c'(n) = -na_n - \sum_{k=1}^{n-1} a_k c'(n-k) \quad 1 \leq n \leq P \quad (4.6.11)$$

$$c'(n) = -\sum_{k=1}^P a_k c'(n-k) \quad n > P \quad (4.6.12)$$

$$\text{and } a_n = \frac{1}{n} \left[c'(n) + \sum_{k=1}^{n-1} a_k c'(n-k) \right] \quad n > 0 \quad (4.6.13)$$

It is evident that the first P values of $c'(n)$, $1 \leq n \leq P$, completely specify the model coefficients. However, since $c'(0) = 0$, the information about the gain d_0 is lost in the ramp cepstrum. Equation (4.6.12) for $n > P$ is reminiscent of similar equations for the impulse response in (4.2.5) and the autocorrelation in (4.2.18), with the major difference that for the ramp cepstrum the relation is only true for $n > P$, while for the impulse response and the autocorrelation, the relations are true for $n > 0$ and $k > 0$, respectively.

Since $R(z) = H(z)H(z^{-1})$, we have

$$\log R(z) = \log H(z) + \log H(z^{-1}) \quad (4.6.14)$$

and if $c_r(n)$ is the real cepstrum of $R(e^{j\omega})$, we conclude that

$$c_r(n) = c(n) + c(-n) \quad (4.6.15)$$

For minimum-phase $H(z)$, $c(n) = 0$ for $n < 0$. Therefore,

$$c_r(n) = \begin{cases} c(-n) & n < 0 \\ 2c(0) & n = 0 \\ c(n) & n > 0 \end{cases} \quad (4.6.16)$$

and

$$c(n) = \begin{cases} 0 & n < 0 \\ \frac{c_r(0)}{2} & n = 0 \\ c_r(n) & n > 0 \end{cases} \quad (4.6.17)$$

In other words, the cepstrum $c(n)$ can be obtained simply by taking the inverse Fourier transform of $\log R(e^{j\omega})$ to obtain $c_r(n)$ and then applying (4.6.17).

EXAMPLE 4.6.1. From (4.6.7) we find that the cepstrum of the AP(1) model is given by

$$c(n) = \begin{cases} 0 & n < 0 \\ \log d_0 & n = 0 \\ \frac{1}{n}(-a)^n & n > 0 \end{cases} \quad (4.6.18)$$

From (4.2.18) with $P = 1$ and $k = 1$, we have $a_1^{(1)} = -r(1)/r(0) = k_1$; and from (4.6.7) we have $a_1 = -c(1)$. These results are summarized below:

$$a_1^{(1)} = a = -\rho(1) = k_1 = -c(1) \quad (4.6.19)$$

The fact that $\rho(1) = c(1)$ here is peculiar to a single-pole spectrum and is not true in general for arbitrary spectra. And $\rho(1)$ is the integral of a cosine-weighted spectrum while $c(1)$ is the integral of a cosine-weighted log spectrum.

EXAMPLE 4.6.2. From (4.6.7), the cepstrum for an AP(2) model is equal to

$$c(n) = \begin{cases} 0 & n < 0 \\ \log d_0 & n = 0 \\ \frac{1}{n}(p_1^n + p_2^n) & n > 0 \end{cases} \quad (4.6.20)$$

For a complex conjugate pole pair, we have

$$c(n) = \frac{2}{n} r^n \cos n\theta \quad n > 0 \quad (4.6.21)$$

where $p_{1,2} = r \exp(\pm j\theta)$. Therefore, the cepstrum of a damped sine wave is a damped cosine wave. The cepstrum and autocorrelation are similar in that they are both damped cosines, but the cepstrum has an additional $1/n$ weighting. From (4.6.7) and (4.6.8) we can relate the model parameters and the cepstral coefficients:

$$\begin{aligned} a_1 &= -c(1) \\ a_2 &= -c(2) + \frac{1}{2}c^2(1) \end{aligned} \quad (4.6.22)$$

and

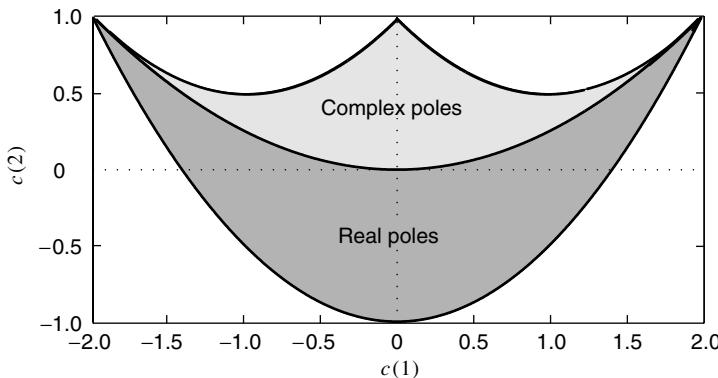
$$\begin{aligned} c(1) &= -a_1 \\ c(2) &= -a_2 + \frac{1}{2}a_1^2 \end{aligned} \quad (4.6.23)$$

Using (4.2.71) and the relations for the cepstrum, we can derive the conditions on the cepstrum for $H(z)$ to be minimum-phase:

$$\begin{aligned} c(2) &> \frac{c^2(1)}{2} - 1 \\ c(2) &< \frac{c^2(1)}{2} - c(1) + 1 \\ c(2) &< \frac{c^2(1)}{2} + c(1) + 1 \end{aligned} \quad (4.6.24)$$

The corresponding admissible region is shown in Figure 4.14. The region corresponding to complex roots is given by

$$\frac{1}{2}c^2(1) - 1 < c(2) < \frac{c^2(1)}{4} \quad (4.6.25)$$

**FIGURE 4.14**

Minimum-phase region of the AP(2) model in the cepstral domain.

In comparing Figures 4.6, 4.8, and 4.14, we note that the admissible regions for the PACS and ACS are convex while that for the cepstral coefficients is not. (A region is convex if a straight line drawn between any two points in the region lies completely in the region.) In general, the PACS and the ACS span regions or spaces that are convex. The admissible region in Figure 4.14 for the model coefficients is also convex. However, for $P > 2$ the admissible regions for the model coefficients are not convex, in general.

Cepstral distance. A measure of the difference between two signals, which has many applications in speech coding and recognition, is the distance between their log spectra (Rabiner and Juang 1993). It is known as the *cepstral distance* and is defined as

$$\text{CD} \triangleq \frac{1}{2\pi} \int_{-\pi}^{\pi} |\log R_1(e^{j\omega}) - \log R_2(e^{j\omega})|^2 d\omega \quad (4.6.26)$$

$$= \sum_{n=-\infty}^{\infty} [c_1(n) - c_2(n)]^2 \quad (4.6.27)$$

where $c_1(n)$ and $c_2(n)$ are the cepstral coefficients of $R_1(e^{j\omega})$ and $R_2(e^{j\omega})$, respectively (see Problem 4.36). Since for minimum-phase sequences the cepstrum decays fast, the summation (4.6.27) can be computed with sufficient accuracy using a small number of terms, usually 20 to 30. For minimum-phase all-pole models, which are mostly used in speech processing, the cepstral coefficients are efficiently computed using the recursion (4.6.7).

4.6.3 All-Zero Models

The cepstrum of a minimum-phase all-zero model is given by (4.6.2) and (4.6.3) with $P = 0$. The cepstrum corresponding to a minimum-phase AZ(Q) model is related to its real cepstrum by

$$c(n) = \begin{cases} 0 & n < 0 \\ \frac{c_r(n)}{2} & n = 0 \\ c_r(n) & n > 0 \end{cases} \quad (4.6.28)$$

Since we found $c(n)$, the coefficients of a minimum-phase AZ(Q) model $D(z)$ can be

$$d_k = \begin{cases} e^c d_0 & k = 0 \\ c(k) d_0 + \frac{1}{k} \sum_{m=0}^{k-1} m c(m) d_{k-m} & 1 \leq k \leq Q \end{cases} \quad (4.6.29)$$

This procedure for finding a minimum-phase polynomial $D(z)$ from the autocorrelation consists in first computing the cepstrum from the log spectrum, then applying (4.6.28) and the recursion (4.6.29) to compute the coefficients d_k . This approach to the spectral factorization of AZ(Q) models is preferable because finding the roots of $R(z)$ for large Q may be cumbersome.

Mixed pole-zero model representations. In the previous sections we saw that the $P + Q + 1$ parameters of the minimum-phase PZ(P, Q) model can be represented equivalently and uniquely by $P + Q + 1$ values of the impulse response, the autocorrelation, or the cepstrum. A question arises as to whether PZ(P, Q) can be represented uniquely by a mixture of representations, as long as the total number of representative values is $P + Q + 1$. For example, could we have a unique representation that consists of, say, Q autocorrelation values and $P + 1$ impulse response values, or some other mixture? The answer to this question has not been explored in general; the relevant equations are sufficiently nonlinear that a totally different approach would appear to be needed to solve the general problem.

4.7 SUMMARY

In this chapter we introduced the class of pole-zero signal models and discussed their properties. Each model consists of two components: an excitation source and a system. In our treatment, we emphasized that the properties of a signal model are shaped by the properties of both components; and we tried, whenever possible, to attribute each property to its originator. Thus, for uncorrelated random inputs, which by definition are the excitations for ARMA models, the second-order moments of the signal model and its minimum-phase characteristics are completely determined by the system. For excitations with line spectra, properties such as minimum phase are meaningful only when they are attributed to the underlying system. If the goal is to model a signal with a line PSD, the most appropriate approach is to use a harmonic process.

We provided a detailed description of the autocorrelation, power spectrum density, partial correlation, and cepstral properties of all AZ, AP, and PZ models for the general case and for first- and second-order models. An understanding of these properties is very important for model selection in practical applications.

PROBLEMS

- 4.1** Show that a second-order pole p_i contributes the term $np_i^n u(n)$ and a third-order pole the terms $np_i^n u(n) + n^2 p_i^n u(n)$ to the impulse response of a causal PZ model. The general case is discussed in Oppenheim et al. (1997).
- 4.2** Consider a zero-mean random sequence $x(n)$ with PSD

$$R_x(e^{j\omega}) = \frac{5 + 3 \cos \omega}{17 + 8 \cos \omega}$$

- (a) Determine the innovations representation of the process $x(n)$.
(b) Find the autocorrelation sequence $r_x(l)$.

- 4.3** We want to generate samples of a Gaussian process with autocorrelation $r_x(l) = (\frac{1}{2})^{|l|} + (-\frac{1}{2})^{|l|}$ for all l .
- Find the difference equation that generates the process $x(n)$ when excited by $w(n) \sim \text{WGN}(0, 1)$.
 - Generate $N = 1000$ samples of the process and estimate the pdf, using the histogram and the normalized autocorrelation $\rho_x(l)$ using $\hat{\rho}_x(l)$ [see Equation (1.2.1)].
 - Check the validity of the model by plotting on the same graph (i) the true and estimated pdf of $x(n)$ and (ii) the true and estimated autocorrelation.
- 4.4** Compute and compare the autocorrelations of the following processes:
- $x_1(n) = w(n) + 0.3w(n - 1) - 0.4w(n - 2)$ and
 - $x_2(n) = w(n) - 1.2w(n - 1) - 1.6w(n - 2)$ where $w(n) \sim \text{WGN}(0, 1)$.
- Explain your findings.
- 4.5** Compute and plot the impulse response and the magnitude response of the systems $H(z)$ and $H_N(z)$ in Example 4.2.1 for $a = 0.7, 0.95$ and $N = 8, 16, 64$. Investigate how well the all-zero systems approximate the single-pole system.
- 4.6** Prove Equation (4.2.35) by writing explicitly Equation (4.2.33) and rearranging terms. Then show that the coefficient matrix \mathbf{A} can be written as the sum of a triangular Toeplitz matrix and a triangular Hankel matrix (recall that a matrix \mathbf{H} is Hankel if the matrix $\mathbf{J}\mathbf{H}\mathbf{J}^H$ is Toeplitz).
- 4.7** Use the Yule-Walker equations to determine the autocorrelation and partial autocorrelation coefficients of the following AR models, assuming that $w(n) \sim \text{WN}(0, 1)$.
- $x(n) = 0.5x(n - 1) + w(n)$.
 - $x(n) = 1.5x(n - 1) - 0.6x(n - 2) + w(n)$.
- What is the variance σ_x^2 of the resulting process?
- 4.8** Given the AR process $x(n) = x(n - 1) - 0.5x(n - 2) + w(n)$, complete the following tasks.
- Determine $\rho_x(1)$.
 - Using $\rho_x(0)$ and $\rho_x(1)$, compute $\{\rho_x(l)\}_{l=0}^{15}$ by the corresponding difference equation.
 - Plot $\rho_x(l)$ and use the resulting graph to estimate its period.
 - Compare the period obtained in part (c) with the value obtained using the PSD of the model.
(Hint: Use the frequency of the PSD peak.)
- 4.9** Given the parameters d_0, a_1, a_2 , and a_3 of an AP(3) model, compute its ACS analytically and verify your results, using the values in Example 4.2.3. (*Hint: Use Cramer's rule.*)
- 4.10** Consider the following AP(3) model: $x(n) = 0.98x(n - 3) + w(n)$, where $w(n) \sim \text{WGN}(0, 1)$.
- Plot the PSD of $x(n)$ and check if the obtained process is going to exhibit a pseudoperiodic behavior.
 - Generate and plot 100 samples of the process. Does the graph support the conclusion of part (a)? If yes, what is the period?
 - Compute and plot the PSD of the process $y(n) = \frac{1}{3}[x(n - 1) + x(n) + x(n + 1)]$.
 - Repeat part (b) and explain the difference between the behavior of processes $x(n)$ and $y(n)$.
- 4.11** Consider the following AR(2) models: (i) $x(n) = 0.6x(n - 1) + 0.3x(n - 2) + w(n)$ and (ii) $x(n) = 0.8x(n - 1) - 0.5x(n - 2) + w(n)$, where $w(n) \sim \text{WGN}(0, 1)$.
- Find the general expression for the normalized autocorrelation sequence $\rho(l)$, and determine σ_x^2 .
 - Plot $\{\rho(l)\}_{l=0}^{15}$ and check if the models exhibit pseudoperiodic behavior.
 - Justify your answer in part (b) by plotting the PSD of the two models.
- 4.12** (a) Derive the formulas that express the PACS of an AP(3) model in terms of its ACS, using the Yule-Walker equations and Cramer's rule.

- (b) Use the obtained formulas to compute the PACS of the AP(3) model in Example 4.2.3.
 (c) Check the results in part (b) by recomputing the PACS, using the algorithm of Levinson-Durbin.

4.13 Show that the spectrum of any PZ model with real coefficients has zero slope at $\omega = 0$ and $\omega = \pi$.

4.14 Derive Equations (4.2.71) describing the minimum-phase region of the AP(2) model, starting from the conditions

- (a) $|p_1| < 1, |p_2| < 1$ and
 (b) $|k_1| < 1, |k_2| < 1$.

4.15 (a) Show that the spectrum of an AP(2) model with real poles can be obtained by the cascade connection of two AP(1) models with real coefficients.
 (b) Compute and plot the impulse response, ACS, PACS, and spectrum of the AP models with $p_1 = 0.6, p_2 = -0.9$, and $p_1 = p_2 = 0.9$.

4.16 Prove Equation (4.2.89) and demonstrate its validity by plotting the spectrum (4.2.88) for various values of r and θ .

4.17 Prove that if the AP(P) model $A(z)$ is minimum-phase, then

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} \log \frac{1}{|A(e^{j\omega})|^2} d\omega = 0$$

4.18 (a) Prove Equations (4.2.101) and (4.2.102) and recreate the plot in Figure 4.8(a).
 (b) Determine and plot the regions corresponding to complex and real poles in the autocorrelation domain by recreating Figure 4.8(b).

4.19 Consider an AR(2) process $x(n)$ with $d_0 = 1, a_1 = -1.6454, a_2 = 0.9025$, and $w(n) \sim \text{WGN}(0, 1)$.

- (a) Generate 100 samples of the process and use them to estimate the ACS $\hat{\rho}_x(l)$, using Equation (1.2.1).
 (b) Plot and compare the estimated and theoretical ACS values for $0 \leq l \leq 10$.
 (c) Use the estimated values of $\hat{\rho}_x(l)$ and the Yule-Walker equations to estimate the parameters of the model. Compare the estimated with the true values, and comment on the accuracy of the approach.
 (d) Use the estimated parameters to compute the PSD of the process. Plot and compare the estimated and true PSDs of the process.
 (e) Compute and compare the estimated with the true PACS.

4.20 Find a minimum-phase model with autocorrelation $\rho(0) = 1, \rho(\pm 1) = 0.25$, and $\rho(l) = 0$ for $|l| \geq 2$.

4.21 Consider the MA(2) model $x(n) = w(n) - 0.1w(n-1) + 0.2w(n-2)$.

- (a) Is the process $x(n)$ stationary? Why?
 (b) Is the model minimum-phase? Why?
 (c) Determine the autocorrelation and partial autocorrelation of the process.

4.22 Consider the following ARMA models: (i) $x(n) = 0.6x(n-1) + w(n) - 0.9w(n-1)$ and (ii) $x(n) = 1.4x(n-1) - 0.6x(n-2) + w(n) - 0.8w(n-1)$.

- (a) Find a general expression for the autocorrelation $\rho(l)$.
 (b) Compute the partial autocorrelation k_m for $m = 1, 2, 3$.
 (c) Generate 100 samples from each process, and use them to estimate $\{\hat{\rho}(l)\}_0^{20}$ using Equation (1.2.1).
 (d) Use $\hat{\rho}(l)$ to estimate $\{\hat{k}_m\}_1^{20}$.
 (e) Plot and compare the estimates with the theoretically obtained values.

4.23 Determine the coefficients of a PZ(2, 1) model with autocorrelation values $r_h(0) = 19$, $r_h(1) = 9$, $r_h(2) = -5$, and $r_h(3) = -7$.

- 4.24** (a) Show that the impulse response of an AZ(Q) model can be recovered from its response $\tilde{h}(n)$ to a periodic train with period L if $L > Q$.
 (b) Show that the ACS of an AZ(Q) model can be recovered from the ACS or spectrum of $\tilde{h}(n)$ if $L \geq 2Q + 1$.

4.25 Prove Equation (4.3.17) and illustrate its validity by computing the PACS of the model $H(z) = 1 - 0.8z^{-1}$.

4.26 Prove Equations (4.3.24) that describe the minimum-phase region of the AZ(2) model.

4.27 Consider an AZ(2) model with $d_0 = 2$ and zeros $z_{1,2} = 0.95e^{\pm j\pi/3}$.

- (a) Compute and plot $N = 100$ output samples by exciting the model with the process $w(n) \sim \text{WGN}(0, 1)$.
 (b) Compute and plot the ACS, PACS, and spectrum of the model.
 (c) Repeat parts (a) and (b) by assuming that we have an AP(2) model with poles at $p_{1,2} = 0.95e^{\pm j\pi/3}$.
 (d) Investigate the duality between the ACS and PACS of the two models.

4.28 Prove Equations (4.4.31) and use them to reproduce the plot shown in Figure 4.12(b). Indicate which equation corresponds to each curve.

4.29 Determine the spectral flatness measure of the following processes:

- (a) $x(n) = a_1x(n-1) + a_2x(n-2) + w(n)$ and
 (b) $x(n) = w(n) + b_1w(n-1) + b_2w(n-2)$, where $w(n)$ is a white noise sequence.

4.30 Consider a zero-mean wide-sense stationary (WSS) process $x(n)$ with PSD $R_x(e^{j\omega})$ and an $M \times M$ correlation matrix with eigenvalues $\{\lambda_k\}_1^M$. Szegő's theorem (Grenander and Szegő 1958) states that if $g(\cdot)$ is a continuous function, then

$$\lim_{M \rightarrow \infty} \frac{g(\lambda_1) + g(\lambda_2) + \cdots + g(\lambda_M)}{M} = \frac{1}{2\pi} \int_{-\pi}^{\pi} g[R_x(e^{j\omega})] d\omega$$

Using this theorem, show that

$$\lim_{M \rightarrow \infty} (\det \mathbf{R}_x)^{1/M} = \exp \left\{ \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln[R_x(e^{j\omega})] d\omega \right\}$$

4.31 Consider two linear random processes with system functions

$$(i) \quad H(z) = \frac{1 - 0.81z^{-1} - 0.4z^{-2}}{(1 - z^{-1})^2} \quad \text{and} \quad (ii) \quad H(z) = \frac{1 - 0.5z^{-1}}{1 - z^{-1}}$$

- (a) Find a difference equation that leads to a numerically stable simulation of each process.
 (b) Generate and plot 100 samples from each process, and look for indications of nonstationarity in the obtained records.
 (c) Compute and plot the second difference of (i) and the first difference of (ii). Comment about the stationarity of the obtained records.

4.32 Generate and plot 100 samples for each of the linear processes with system functions

$$(a) \quad H(z) = \frac{1}{(1 - z^{-1})(1 - 0.9z^{-1})}$$

$$(b) \quad H(z) = \frac{1 - 0.5z^{-1}}{(1 - z^{-1})(1 - 0.9z^{-1})}$$

and then estimate and examine the values of the ACS $\{\hat{\rho}(l)\}_0^{20}$ and the PACS $\{\hat{k}_m\}_1^{20}$.

4.33 Consider the process $y(n) = d_0 + d_1 n + d_2 n^2 + x(n)$, where $x(n)$ is a stationary process with known autocorrelation $r_x(l)$.

- (a) Show that the process $y^{(2)}(n)$ obtained by passing $y(n)$ through the filter $H(z) = (1 - z^{-1})^2$ is stationary.
- (b) Express the autocorrelation $r_y^{(2)}(l)$ of $y^{(2)}(n)$ in terms of $r_x(l)$. *Note:* This process is used in practice to remove quadratic trends from data before further analysis.

4.34 Prove Equation (4.6.7), which computes the cepstrum of an AP model from its coefficients.

4.35 Consider a minimum-phase AZ(Q) model $D(z) = \sum_{k=0}^Q d_k z^{-k}$ with complex cepstrum $c(k)$. We create another AZ model with coefficients $\tilde{d}_k = \alpha^k d_k$ and complex cepstrum $\tilde{c}(k)$.

- (a) If $0 < \alpha < 1$, find the relation between $\tilde{c}(k)$ and $c(k)$.
- (b) Choose α so that the new model has no minimum phase.
- (c) Choose α so that the new model has a maximum phase.

4.36 Prove Equation (4.6.27), which determines the cepstral distance in the frequency and time domains.

Nonparametric Power Spectrum Estimation

The essence of frequency analysis is the representation of a signal as a superposition of sinusoidal components. In theory, the exact form of this decomposition (spectrum) depends on the assumed signal model. In Chapters 2 and 3 we discussed the mathematical tools required to define and compute the spectrum of signals described by deterministic and stochastic models, respectively. In practical applications, where only a finite segment of a signal is available, we cannot obtain a complete description of the adopted signal model. Therefore, we can only compute an approximation (estimate) of the spectrum of the adopted signal model (“true” or theoretical spectrum). The quality of the estimated spectrum depends on

- How well the assumed signal model represents the data.
- What values we assign to the unavailable signal samples.
- Which spectrum estimation method we use.

Clearly, meaningful application of spectrum estimation in practical problems requires sufficient a priori information, understanding of the signal generation process, knowledge of theoretical concepts, and experience.

In this chapter we discuss the most widely used correlation and spectrum estimation methods, as well as their properties, implementation, and application to practical problems. We discuss only *nonparametric* techniques that do *not* assume a particular functional form, but allow the form of the estimator to be determined *entirely* by the data. These methods are based on the discrete Fourier transform of either the signal segment or its autocorrelation sequence. In contrast, parametric methods assume that the available signal segment has been generated by a specific parametric model (e.g., a pole-zero or harmonic model). Since the choice of an inappropriate signal model will lead to erroneous results, the successful application of parametric techniques, without sufficient a priori information, is very difficult in practice. These methods are discussed in Chapter 9.

We begin this chapter with an introductory discussion on the purpose of, and the DSP approach to, spectrum estimation. We explore various errors involved in the estimation of finite-length data records (i.e., based on partial information). We also outline conventional techniques for deterministic signals, using concepts developed in Chapter 2. Also in Section 3.6, we presented important concepts and results from the estimation theory that are used extensively in this chapter. Section 5.3 is the main section of this chapter in which we discuss various nonparametric approaches to the power spectrum estimation of stationary random signals. This analysis is extended to joint stationary (bivariate) random signals for the computation of the cross-spectrum in Section 5.4. The computation of auto and cross-spectra using Thomson’s multiple windows (or multitapers) is discussed in Section

5.5. Finally, in Section 5.6 we summarize important topics and concepts from this chapter. A classification of the various spectral estimation methods that are discussed in this book is provided in Figure 5.1.

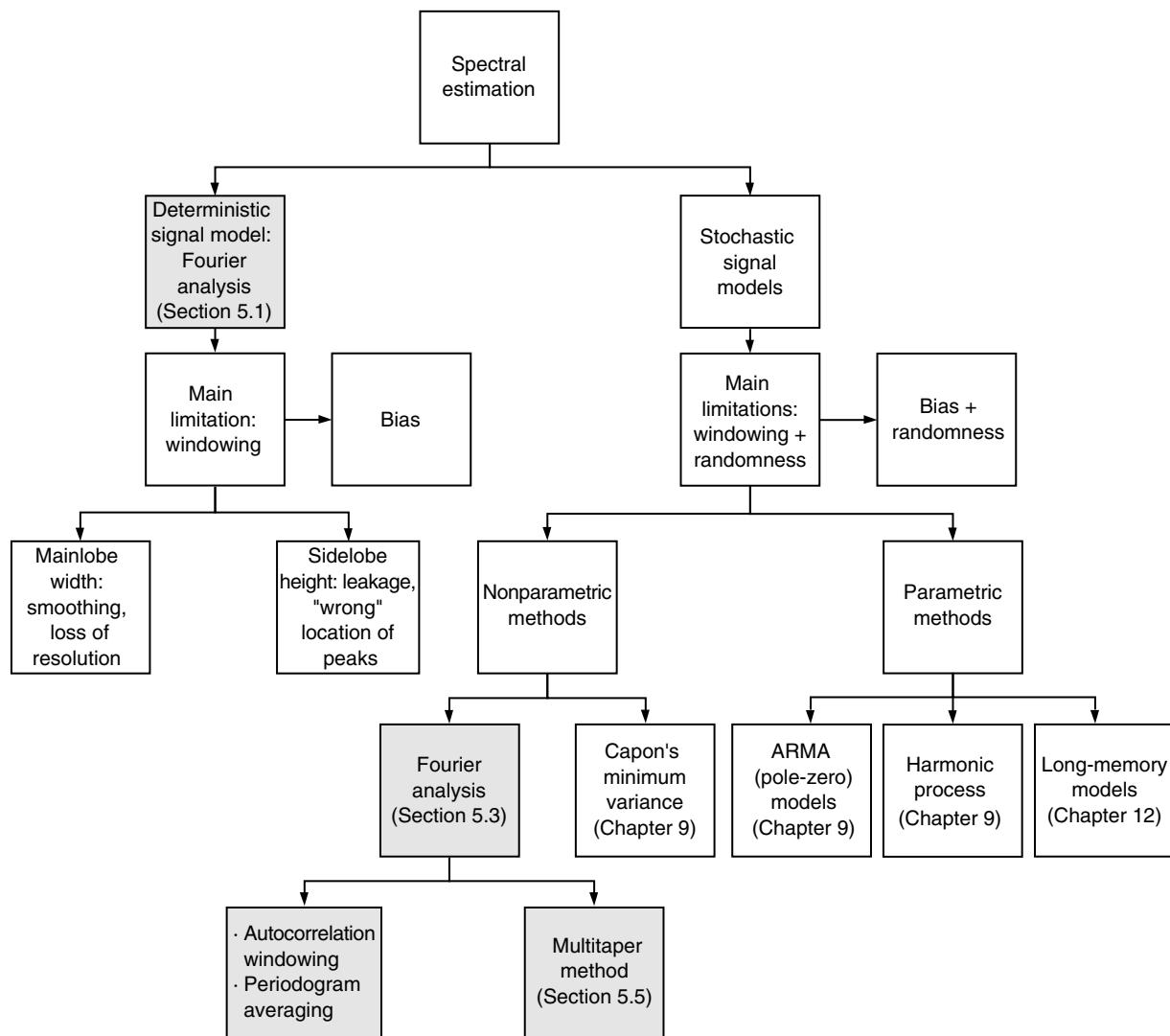


FIGURE 5.1

Classification of various spectrum estimation methods.

5.1 SPECTRAL ANALYSIS OF DETERMINISTIC SIGNALS

If we adopt a deterministic signal model, the mathematical tools for spectral analysis are the Fourier series and the Fourier transforms summarized in Section 2.2.1. It should be stressed at this point that applying any of these tools requires that the signal values in the entire time interval from $-\infty$ to $+\infty$ be available. If it is known a priori that a signal is periodic, then only one period is needed. The rationale for defining and studying various spectra for deterministic signals is threefold. First, we note that every realization (or sample function) of a stochastic process is a deterministic function. Thus we can use the Fourier series and transforms to compute a spectrum for stationary processes. Second, deterministic functions

and sequences are used in many aspects of the study of stationary processes, for example, the autocorrelation sequence, which is a deterministic sequence. Third, the various spectra that can be defined for deterministic signals can be used to summarize important features of stationary processes.

Most practical applications of spectrum estimation involve continuous-time signals. For example, in speech analysis we use spectrum estimation to determine the pitch of the glottal excitation and the formants of the vocal tract (Rabiner and Schafer 1978). In electroencephalography, we use spectrum estimation to study sleep disorders and the effect of medication on the functioning of the brain (Duffy, Iyer, and Surwill 1989). Another application is in Doppler radar, where the frequency shift between the transmitted and the received waveform is used to determine the radial velocity of the target (Levanon 1988).

The numerical computation of the spectrum of a continuous-time signal involves three steps:

1. Sampling the continuous-time signal to obtain a sequence of samples.
2. Collecting a finite number of contiguous samples (data segment or block) to use for the computation of the spectrum. This operation, which usually includes weighting of the signal samples, is known as *windowing*, or *tapering*.
3. Computing the values of the spectrum at the desired set of frequencies. This step is usually implemented using some efficient implementation of the DFT.

The above processing steps, which are necessary for DFT-based spectrum estimation, are shown in Figure 5.2. The continuous-time signal is first processed through a low-pass (antialiasing) filter and then sampled to obtain a discrete-time signal. Data samples of frame length N with *frame overlap* N_0 are selected and then conditioned using a window. Finally, a suitable-length DFT of the windowed data is taken as an estimate of its spectrum, which is then analyzed. In this section, we discuss in detail the effects of each of these operations on the accuracy of the computed spectrum. The understanding of the implications of these effects is very important in all practical applications of spectrum estimation.

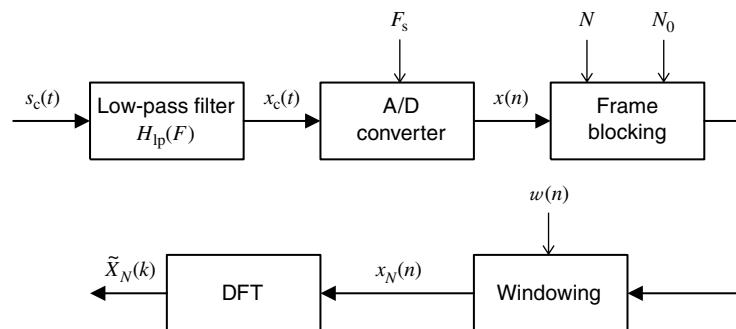


FIGURE 5.2
 DFT-based Fourier analysis system for continuous-time signals.

5.1.1 Effect of Signal Sampling

The continuous-time signal $s_c(t)$, whose spectrum we seek to estimate, is first passed through a low-pass filter, also known as an *antialiasing* filter $H_{lp}(F)$, in order to minimize the aliasing error after sampling. The antialiased signal $x_c(t)$ is then sampled through an analog-to-digital converter[†] (ADC) to produce the discrete-time sequence $x(n)$, that is,

$$x(n) = x_c(t)|_{t=n/F_s} \quad (5.1.1)$$

[†]We will ignore the quantization of discrete-time signals as discussed in Chapter 2.

From the sampling theorem in Section 2.2.2, we have

$$X(e^{j2\pi F/F_s}) = F_s \sum_{l=-\infty}^{\infty} X_c(F - lF_s) \quad (5.1.2)$$

where $X_c(F) = H_{lp}(F)S_c(F)$. We note that the spectrum of the discrete-time signal $x(n)$ is a periodic replication of $X_c(F)$. Overlapping of the replicas $X_c(F - lF_s)$ results in aliasing. Since any practical antialiasing filter does not have infinite attenuation in the stopband, some nonzero overlap of frequencies higher than $F_s/2$ should be expected within the band of frequencies of interest in $x(n)$. These aliased frequencies give rise to the aliasing error, which, in any practical signal, is unavoidable. It can be made negligible by a properly designed antialiasing filter $H_{lp}(F)$.

5.1.2 Windowing, Periodic Extension, and Extrapolation

In practice, we compute the spectrum of a signal by using a finite-duration segment. The reason is threefold:

1. The spectral composition of the signal changes with time. or
2. We have only a finite set of data at our disposal. or
3. We wish to keep the computational complexity to an acceptable level.

Therefore, it is necessary to partition $x(n)$ into blocks (or *frames*) of data prior to processing. This operation is called *frame blocking*, and it is characterized by two parameters: the *length* of frame N and the *overlap* between frames N_0 (see Figure 5.2). Therefore, the central problem in practical frequency analysis can be stated as follows:

Determine the spectrum of a signal $x(n)$, $-\infty < n < \infty$, from its values in a finite interval $0 \leq n \leq N - 1$, that is, from a finite-duration segment.

Since $x(n)$ is unknown for $n < 0$ and $n \geq N$, we cannot say, without having sufficient a priori information, whether the signal is periodic or aperiodic. If we can reasonably assume that the signal is periodic with fundamental period N , we can easily determine its spectrum by computing its Fourier series, using the DFT (see Section 2.2.1).

However, in most practical applications, we cannot make this assumption because the available block of data could be either part of the period of a periodic signal or a segment from an aperiodic signal. In such cases, the spectrum of the signal *cannot* be determined without *assigning* values to the signal samples outside the available interval. There are three ways to deal with this issue:

1. *Periodic extension*. We assume that $x(n)$ is periodic with period N , that is, $x(n) = x(n + N)$ for all n , and we compute its Fourier series, using the DFT.
2. *Windowing*. We assume that the signal is zero outside the interval of observation, that is, $x(n) = 0$ for $n < 0$ and $n \geq N$. This is equivalent to multiplying the signal with the rectangular window

$$w_R(n) \triangleq \begin{cases} 1 & 0 \leq n \leq N - 1 \\ 0 & \text{elsewhere} \end{cases} \quad (5.1.3)$$

The resulting sequence is aperiodic, and its spectrum is obtained by the discrete-time Fourier transform (DTFT).

3. *Extrapolation*. We use a priori information about the signal to extrapolate (i.e., determine its values for $n < 0$ and $n \geq N$) outside the available interval and then determine its spectrum by using the DTFT.

Periodic extension and windowing can be considered the simplest forms of extrapolation. It should be obvious that a successful extrapolation results in better spectrum estimates

than periodic extension or windowing. Periodic extension is a straightforward application of the DFT, whereas extrapolation requires some form of a sophisticated signal model. As we shall see, most of the signal modeling techniques discussed in this book result in some kind of extrapolation. We first discuss, in the next section, the effect of spectrum sampling as imposed by the application of DFT (and its side effect—the periodic extension) before we provide a detailed analysis of the effect of windowing.

5.1.3 Effect of Spectrum Sampling

In many real-time spectrum analyzers, as illustrated in Figure 5.2, the spectrum is computed (after signal conditioning) by using the DFT. From Section 2.2.3, we note that this computation samples the continuous spectrum at equispaced frequencies. Theoretically, if the number of DFT samples is greater than or equal to the frame length N , then the exact continuous spectrum (based on the given frame) can be obtained by using the frequency-domain reconstruction (Oppenheim and Schafer 1989; Proakis and Manolakis 1996). This reconstruction, which requires a periodic sinc function [defined in (5.1.9)], is not a practical function to implement, especially in real-time applications. Hence a simple linear interpolation is used for plotting or display purposes. This linear interpolation can lead to misleading results even though the computed DFT sample values are correct. It is possible that there may not be a DFT sample precisely at a frequency where a peak of the DTFT is located. In other words, the DFT spectrum misses this peak, and the resulting linearly interpolated spectrum provides the wrong location and height of the DTFT spectrum peak. This error can be made smaller by sampling the DTFT spectrum at a finer grid, that is, by increasing the size of the DFT. The denser spectrum sampling is implemented by an operation called *zero padding* and is discussed later in this section.

Another effect of the application of DFT for spectrum calculations is the periodic extension of the sequence in the time domain. From our discussion in Section 2.2.3, it follows that the N -point DFT

$$\tilde{X}(k) = \sum_{n=0}^{N-1} x(n)e^{-j(2\pi/N)kn} \quad (5.1.4)$$

is periodic with period N . This should be expected given the relationship of the DFT to the Fourier transform or the Fourier series of discrete-time signals, which are periodic in ω with period 2π . A careful look at the inverse DFT

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k)e^{j(2\pi/N)kn} \quad (5.1.5)$$

reveals that $x(n)$ is also periodic with period N . This is a somewhat surprising result since no assumption about the signal $x(n)$ outside the interval $0 \leq n \leq N - 1$ has been made. However, this periodicity in the time domain can be easily justified by recalling that sampling in the time domain results in a periodicity in the frequency domain, and vice versa.

To understand these effects of spectrum sampling, consider the following example in which a continuous-time sinusoidal signal is sampled and then is truncated by a rectangular window before its DFT is performed.

EXAMPLE 5.1.1. A continuous-time signal $x_c(t) = 2 \cos 2\pi t$ is sampled with a sampling frequency of $F_s = 1/T = 10$ samples per second, to obtain the sequence $x(n)$. It is windowed by an N -point rectangular window $w_R(n)$ to obtain the sequence $x_N(n)$. Determine and plot $|\tilde{X}_N(k)|$, the magnitude of the DFT of $x_N(n)$, for (a) $N = 10$ and (b) $N = 15$. Comment on the shapes of these plots.

Solution. The discrete-time signal $x(n)$ is a sampled version of $x_c(t)$ and is given by

$$x(n) = x_c(t = nT) = 2 \cos \frac{2\pi n}{F_s} = 2 \cos 0.2\pi n \quad T = 0.1 \text{ s}$$

Then, $x(n)$ is a periodic sequence with fundamental period $N = 10$.

- For $N = 10$, we obtain $x_N(n) = 2 \cos 0.4\pi n, 0 \leq n \leq 9$, which contains one period of $x(n)$. The periodic extension of $x_N(n)$ and the magnitude plot of its DFT are shown in the top row of Figure 5.3. For comparison, the DTFT $X_N(e^{j\omega})$ of $x_N(n)$ is also superimposed on the DFT samples. We observe that the DFT has only two nonzero samples, which together constitute the correct frequency of the analog signal $x_c(t)$. The DTFT has a mainlobe and several sidelobes due to the windowing effect. However, the DFT samples the sidelobes at their zero values, as illustrated in the DFT plot. Another explanation for this behavior is that since the samples in $x_N(n)$ for $N = 10$ constitute one full period of $\cos 0.4\pi n$, the 10-point periodic extension of $x_N(n)$, shown in the top left graph of Figure 5.3, results in the original sinusoidal sequences $x(n)$. Thus what the DFT “sees” is the exact sampled signal $x_c(t)$. In this case, the choice of N is a desirable one.
- For $N = 15$, we obtain $x_N(n) = 2 \cos 0.4\pi n, 0 \leq n \leq 14$, which contains $1\frac{1}{2}$ periods of $x(n)$. The periodic extension of $x_N(n)$ and the magnitude plot of its DFT are shown in the bottom row of Figure 5.3. Once again for comparison, the DTFT $X_N(e^{j\omega})$ of $x_N(n)$ is superimposed on the DFT samples. In this case, the DFT plot looks markedly different

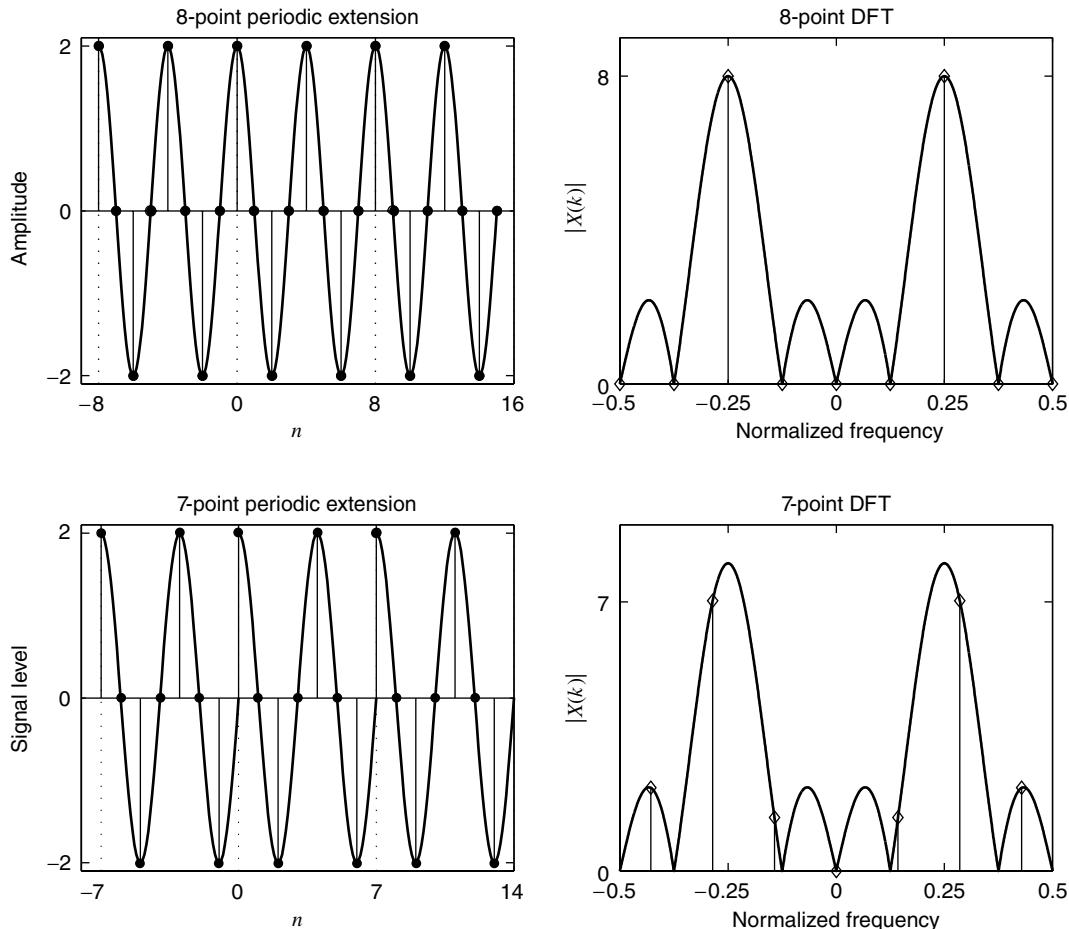


FIGURE 5.3

Effect of window length L on the DFT spectrum shape.

from that for $N = 10$ although the DTFT plot appears to be similar. In this case, the DFT does not sample two peaks at the exact frequencies; hence if the resulting DFT samples are joined by the linear interpolation, then we will get a misleading result. Since the sequence $x_N(n)$ does not contain full periods of $\cos 0.4\pi n$, the periodic extension of $x_N(n)$ contains discontinuities at $n = lN$, $l = 0, \pm 1, \pm 2, \dots$, as shown in the bottom left graph of Figure 5.3. This discontinuity results in higher-order harmonics in the DFT values. The DTFT plot also has mainlobes and sidelobes, but the DFT samples these sidelobes at nonzero values. Therefore, the length of the window is an important consideration in spectrum estimation. The sidelobes are the source of the problem of leakage that gives rise to bias in the spectral values, as we will see in the following section. The suppression of the sidelobes is controlled by the window shape, which is another important consideration in spectrum estimation.

A quantitative description of the above interpretations and arguments related to the capacities and limitations of the DFT is offered by the following result (see Proakis and Manolakis 1996).

THEOREM 5.1 (DFT SAMPLING THEOREM). Let $x_c(t)$, $-\infty < t < \infty$, be a continuous-time signal with Fourier transform $X_c(F)$, $-\infty < F < \infty$. Then, the N -point sequences $\{Tx_p(n), 0 \leq n \leq N - 1\}$ and $\{\tilde{X}_p(k), 0 \leq k \leq N - 1\}$ form an N -point DFT pair, that is,

$$x_p(n) \triangleq \sum_{m=-\infty}^{\infty} x_c(nT - mNT) \xrightarrow[N]{\text{DFT}} \tilde{X}_p(k) \triangleq F_s \sum_{l=-\infty}^{\infty} X_c\left(k \frac{F_s}{N} - lF_s\right) \quad (5.1.6)$$

where $F_s = 1/T$ is the sampling frequency.

Proof. The proof is explored in Problem 5.1.

Thus, given a continuous-time signal $x_c(t)$ and its spectrum $X_c(F)$, we can create a DFT pair by sampling and aliasing in the time and frequency domains. Obviously, this DFT pair provides a “faithful” description of $x_c(t)$ and $X_c(F)$ if both the time-domain aliasing and the frequency-domain aliasing are insignificant. The meaning of relation (5.1.6) is graphically illustrated in Figure 5.4. In this figure, we show the time-domain signals in the left column and their Fourier transforms in the right column. The top row contains continuous-time signals, which are shown as nonperiodic and of infinite extent in both domains, since many real-world signals exhibit this behavior. The middle row contains the sampled version of the continuous-time signal and its periodic Fourier transform (the nonperiodic transform is shown as a dashed curve). Clearly, aliasing in the frequency domain is evident. Finally, the bottom row shows the sampled (periodic) Fourier transform and its corresponding time-domain periodic sequence. Again, aliasing in the time domain should be expected. Thus we have sampled and periodic signals in both domains with the certainty of aliasing one domain and the possibility in both domains. This figure should be recalled any time we use the DFT for the analysis of sampled signals.

Zero padding

The N -point DFT values of an N -point sequence $x(n)$ are samples of the DTFT $X(e^{j\omega})$, as discussed in Chapter 2. These samples can be used to reconstruct the DTFT $X(e^{j\omega})$ by using the periodic sinc interpolating function. Alternatively, one can obtain more (i.e., dense) samples of the DTFT by computing a larger N_{FFT} -point DFT of $x(n)$, where $N_{\text{FFT}} \gg N$. Since the number of samples of $x(n)$ is fixed, the only way we can treat $x(n)$ as an N_{FFT} -point sequence is by appending $N_{\text{FFT}} - N$ zeros to it. This procedure is called the *zero padding* operation, and it is used for many purposes including the augmentation of the sequence length so that a power-of-2 FFT algorithm can be used. In spectrum estimation, zero padding is primarily used to provide a *better-looking* plot of the spectrum of a finite-length sequence. This is shown in Figure 5.5 where the magnitude of an N_{FFT} -point DFT of the eight-point sequence $x(n) = \cos(2\pi n/4)$ is plotted for $N_{\text{FFT}} = 8, 16, 32$, and 64 . The DTFT magnitude $|X(e^{j\omega})|$ is also shown for comparison. It can be seen that as more zeros

are appended (by increasing N_{FFT}), the resulting larger-point DFT provides more closely spaced samples of the DTFT, thus giving a better-looking plot. Note, however, that the zero padding *does not increase* the resolution of the spectrum; that is, there are no new peaks and valleys in the display, just a better display of the available information. This type of plot is called a *high-density spectrum*. For a *high-resolution spectrum*, we have to collect more information by increasing N . The DTFT plots shown in Figures 5.3 and 5.5 were obtained by using a very large amount of zero padding.

5.1.4 Effects of Windowing: Leakage and Loss of Resolution

To see the effect of the window on the spectrum of an arbitrary deterministic signal $x(n)$, defined over the entire range $-\infty < n < \infty$, we notice that the available data record can be expressed as

$$x_N(n) = x(n)w_R(n) \quad (5.1.7)$$

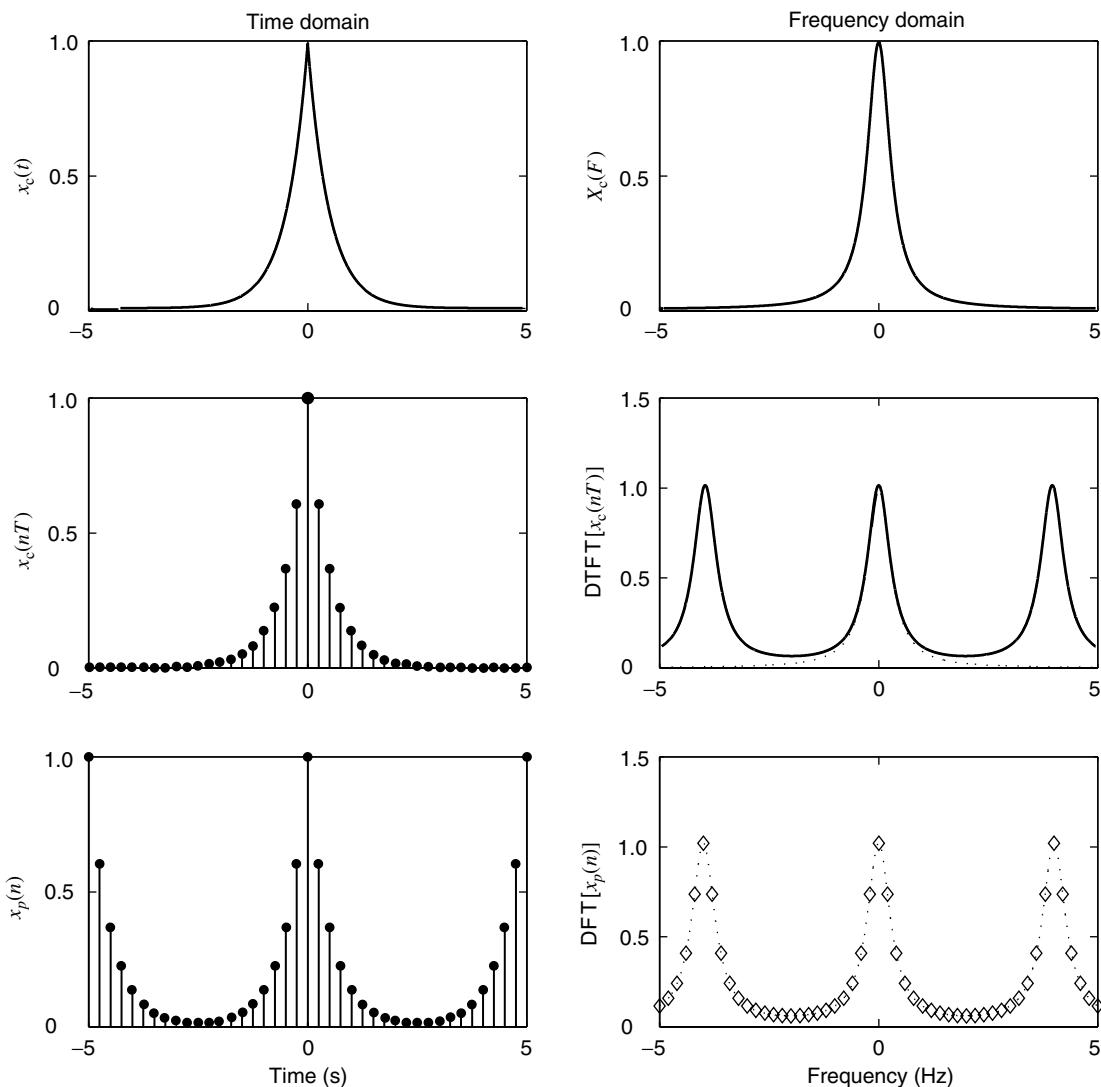
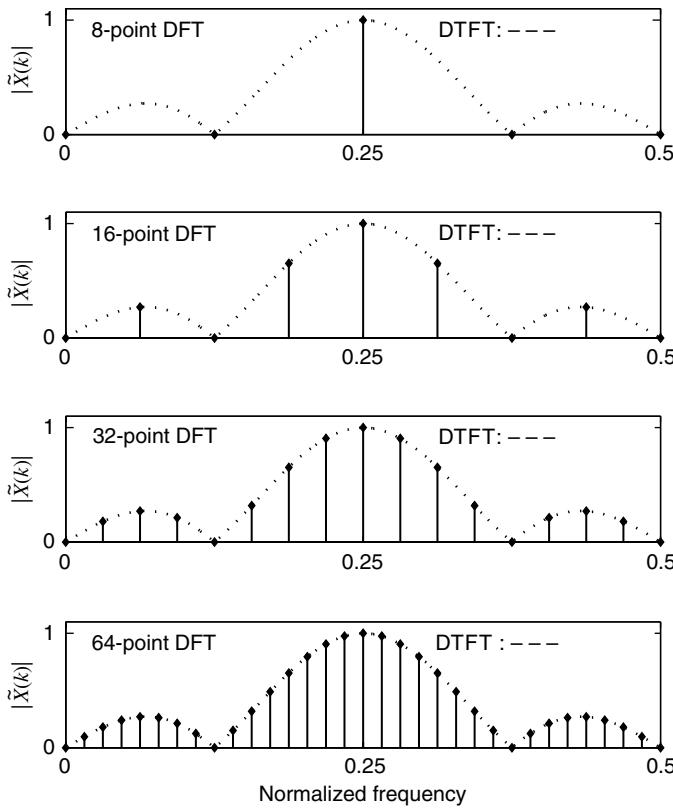


FIGURE 5.4

Graphical illustration of the DFT sampling theorem.

**FIGURE 5.5**

Effect of zero padding.

where $w_R(n)$ is the rectangular window defined in (5.1.3). Thus, a finite segment of the signal can be thought of as a product of the actual signal $x(n)$ and a *data window* $w(n)$. In (5.1.7), $w(n) = w_R(n)$, but $w(n)$ can be any arbitrary finite-duration sequence. The Fourier transform of $x_N(n)$ is

$$X_N(e^{j\omega}) = X(e^{j\omega}) \otimes W(e^{j\omega}) \triangleq \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\theta}) W(e^{j(\omega-\theta)}) d\theta \quad (5.1.8)$$

that is, $X_N(e^{j\omega})$ equals the periodic convolution of the actual Fourier transform with the Fourier transform $W(e^{j\omega})$ of the data window. For the rectangular window, $W(e^{j\omega}) = W_R(e^{j\omega})$, where

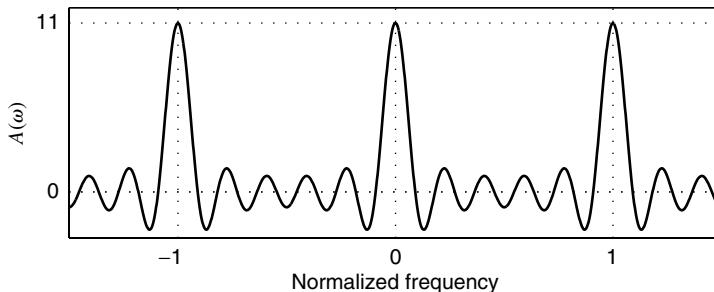
$$W_R(e^{j\omega}) = \left[\frac{\sin(\omega N/2)}{\sin(\omega/2)} \right] e^{-j\omega(N-1)/2} \triangleq A(\omega) e^{-j\omega(N-1)/2} \quad (5.1.9)$$

The function $A(\omega)$ is a periodic function in ω with fundamental period equal to 2π and is called a *periodic sinc* function. Figure 5.6 shows three periods of $A(\omega)$ for $N = 11$. We note that $W_R(e^{j\omega})$ consists of a mainlobe (ML).

$$W_{ML}(e^{j\omega}) = \begin{cases} W_R(e^{j\omega}) & |\omega| < \frac{2\pi}{N} \\ 0 & \frac{2\pi}{N} < |\omega| \leq \pi \end{cases} \quad (5.1.10)$$

and the sidelobes $W_{SL}(e^{j\omega}) = W_R(e^{j\omega}) - W_{ML}(e^{j\omega})$. Thus, (5.1.8) can be written as

$$X_N(e^{j\omega}) = X(e^{j\omega}) \otimes W_{ML}(e^{j\omega}) + X(e^{j\omega}) \otimes W_{SL}(e^{j\omega}) \quad (5.1.11)$$

**FIGURE 5.6**

Plot of $A(\omega) = \sin(\omega N/2)/\sin(\omega/2)$ for $N = 11$.

The first convolution in (5.1.11) smoothes rapid variations and suppresses narrow peaks in $X(e^{j\omega})$, whereas the second convolution introduces ripples in smooth regions of $X(e^{j\omega})$ and can create “false” peaks. Therefore, the spectrum we observe is the convolution of the actual spectrum with the Fourier transform of the data window. The only way to improve the estimate is to increase the window length N or to choose another window shape. For the rectangular window, increasing N results in a narrower mainlobe, and the distortion is reduced. As $N \rightarrow \infty$, $W_R(e^{j\omega})$ tends to an impulse train with period 2π and $X_N(e^{j\omega})$ tends to $X(e^{j\omega})$, as expected. Since in practice the value of N is always finite, the only way to improve the estimate $X_N(e^{j\omega})$ is by properly choosing the shape of the window $w(n)$. The only restriction on $w(n)$ is that it be of finite duration.

It is known that any time-limited sequence $w(n)$ has a Fourier transform $W(e^{j\omega})$ that is nonzero except at a finite number of frequencies. Thus, from (5.1.8) we see that the estimated value $X_N(e^{j\omega_0})$ is computed by using all values of $X(e^{j\omega})$ weighted by $W(e^{j(\omega_0-\theta)})$. The contribution of the sinusoidal components with frequencies $\omega \neq \omega_0$ to the value $X_N(e^{j\omega_0})$ introduces an error known as *leakage*. As the name suggests, energy from one frequency range “leaks” into another, giving the wrong impression of stronger or weaker frequency components.

To illustrate the effect of the window shape and duration on the estimated spectrum, consider the signal

$$x(n) = \cos 0.35\pi n + \cos 0.4\pi n + 0.25 \cos 0.8\pi n \quad (5.1.12)$$

which has a line spectrum with lines at frequencies $\omega_1 = 0.35\pi$, $\omega_2 = 0.4\pi$, and $\omega_3 = 0.8\pi$. This line spectrum (normalized so that the magnitude is between 0 and 1) is shown in the top graph of Figure 5.7 over $0 \leq \omega \leq \pi$. The spectrum $X_N(e^{j\omega})$ of $x_N(n)$ using the rectangular window is given by

$$\begin{aligned} X_N(e^{j\omega}) = & \frac{1}{2}[W(e^{j(\omega+\omega_1)}) + W(e^{j(\omega-\omega_1)}) + W(e^{j(\omega+\omega_2)}) \\ & + W(e^{j(\omega-\omega_2)}) + 0.25W(e^{j(\omega+\omega_3)}) + 0.25W(e^{j(\omega-\omega_3)})] \end{aligned} \quad (5.1.13)$$

The second and the third plots in Figure 5.7 show 2048-point DFTs of $x_N(n)$ for a rectangular data window with $N = 21$ and $N = 81$. We note that the ability to pick out peaks (*resolvability*) depends on the duration $N - 1$ of the data window.[†] To resolve two spectral lines at $\omega = \omega_1$ and $\omega = \omega_2$ using a rectangular window, we should have the difference $|\omega_1 - \omega_2|$ greater than the mainlobe width $\Delta\omega$, which is approximately equal to $2\pi/(N - 1)$, in radians per sampling interval, from the plot of $A(\omega)$ in Figure 5.6, that is,

$$|\omega_1 - \omega_2| > \Delta\omega \approx \frac{2\pi}{N - 1} \quad \text{or} \quad N > \frac{2\pi}{|\omega_1 - \omega_2|} + 1$$

[†]Since there are N samples in a data window, the number of intervals or durations is $N - 1$.

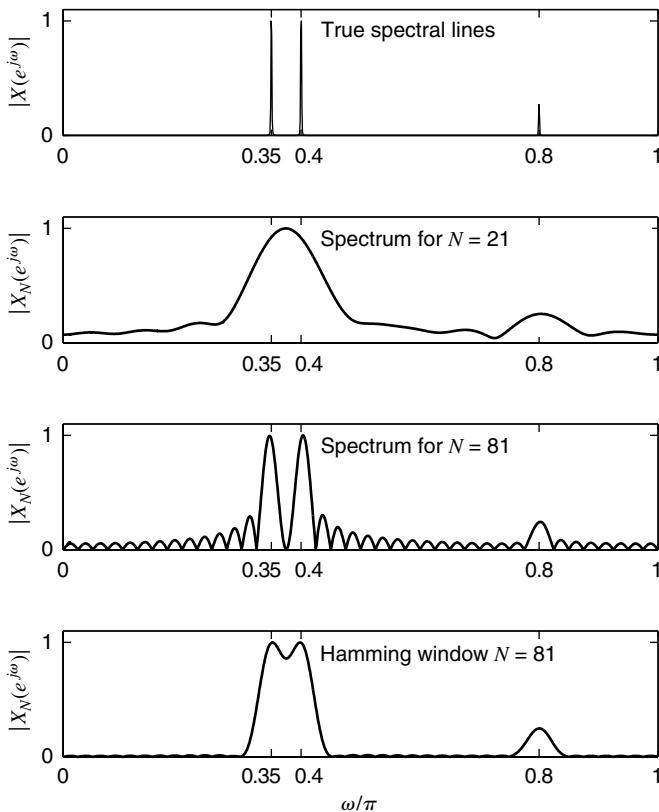


FIGURE 5.7
 Spectrum of three sinusoids using rectangular and Hamming windows.

For a rectangular window of length N , the exact value of $\Delta\omega$ is equal to $1.81\pi/(N - 1)$. If N is too small, the two peaks at $\omega = 0.35\pi$ and $\omega = 0.4\pi$ are fused into one, as shown in the $N = 21$ plot. When $N = 81$, the corresponding plot shows a resolvable separation; however, the peaks have shifted somewhat from their true locations. This is called *bias*, and it is a direct result of the leakage from sidelobes. In both cases, the peak at $\omega = 0.8\pi$ can be distinguished easily (but also has a bias).

Another important observation is that the sidelobes of the data window introduce false peaks. For a rectangular window, the peak sidelobe level is 13 dB below zero, which is not a good attenuation. Thus these false peaks have values that are comparable to that of the true peak at $\omega = 0.8\pi$, as shown in Figure 5.7. These peaks can be minimized by reducing the amplitudes of the sidelobes. The rectangular window cannot help in this regard because of Gibb's well-known phenomenon associated with it. We need a different window shape. However, any window other than the rectangular window has a wider mainlobe; hence this reduction can be achieved only at the expense of the resolution. To illustrate this, consider the Hamming (Hm) data window, given by

$$w_{\text{Hm}}(n) = \begin{cases} 0.54 - 0.46 \cos \frac{2\pi n}{N-1} & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases} \quad (5.1.14)$$

with the approximate width of the mainlobe equal to $8\pi/(N - 1)$ and the exact mainlobe width equal to $6.27\pi/(N - 1)$. The peak sidelobe level is 43 dB below zero, which is

considerably better than that of the rectangular window. The Hamming window is obtained by using the `hamming(N)` function in MATLAB.

The bottom plot in Figure 5.7 shows the 2048-point DFT of the signal $x_N(n)$ for a Hamming window with $N = 81$. Now the peak at $\omega = 0.8\pi$ is more prominent than before, and the sidelobes are almost suppressed. Note also that since the mainlobe width of the Hamming window is wider, the peaks have a wider base—so much so that the first two frequencies are barely recognized. We can correct this problem by choosing a larger window length. This interplay between the shape and the duration of a window function is one of the important issues and, as we will see in Section 5.3, produces similar effects in the spectral analysis of random signals.

Some useful windows

The design of windows for spectral analysis applications has drawn a lot of attention and is examined in detail in Harris (1978). We have already discussed two windows, namely, the rectangular and the Hamming window. Another useful window in spectrum analysis is due to Hann and is mistakenly known as the Hanning window. There are several such windows with varying degrees of tradeoff between resolution (mainlobe width) and leakage (peak sidelobe level). These windows are known as *fixed* windows since each provides a fixed amount of leakage that is independent of the length N . Unlike fixed windows, there are windows that contain a design parameter that can be used to trade between resolution and leakage. Two such windows are the Kaiser window and the Dolph-Chebyshev window, which are widely used in spectrum estimation. Figure 5.8 shows the time-domain window functions and their corresponding frequency-domain log-magnitude plots in decibels for these five windows. The important properties such as peak sidelobe level and mainlobe width of these windows are compared in Table 5.1.

TABLE 5.1

Comparison of properties of commonly used windows. Each window is assumed to be of length N .

Window type	Peak sidelobe level (dB)	Approximate mainlobe width	Exact mainlobe width
Rectangular	-13	$\frac{4\pi}{N-1}$	$\frac{1.81\pi}{N-1}$
Hanning	-32	$\frac{8\pi}{N-1}$	$\frac{5.01\pi}{N-1}$
Hamming	-43	$\frac{8\pi}{N-1}$	$\frac{6.27\pi}{N-1}$
Kaiser	$-A$	—	$\frac{A-8}{2.285N-1}$
Dolph-Chebyshev	$-A$	—	$\cos^{-1} \left[\left(\cosh \frac{\cosh^{-1} 10^{A/20}}{N-1} \right)^{-1} \right]$

Hanning window. This window is given by the function

$$w_{Hn}(n) = \begin{cases} 0.5 - 0.5 \cos \frac{2\pi n}{N-1} & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases} \quad (5.1.15)$$

which is a raised cosine function. The peak sidelobe level is 32 dB below zero, and the approximate mainlobe width is $8\pi/(N-1)$ while the exact mainlobe width is $5.01\pi/(N-1)$. In MATLAB this window function is obtained through the function `hamming(N)`.

Kaiser window. This window function is due to J. F. Kaiser and is given by

$$w_K(n) = \begin{cases} \frac{I_0\left\{\beta\sqrt{1-[1-2n/(N-1)]^2}\right\}}{I_0(\beta)} & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases} \quad (5.1.16)$$

where $I_0(\cdot)$ is the modified zero-order Bessel function of the first kind and β is a window shape parameter that can be chosen to obtain various peak sidelobe levels and the

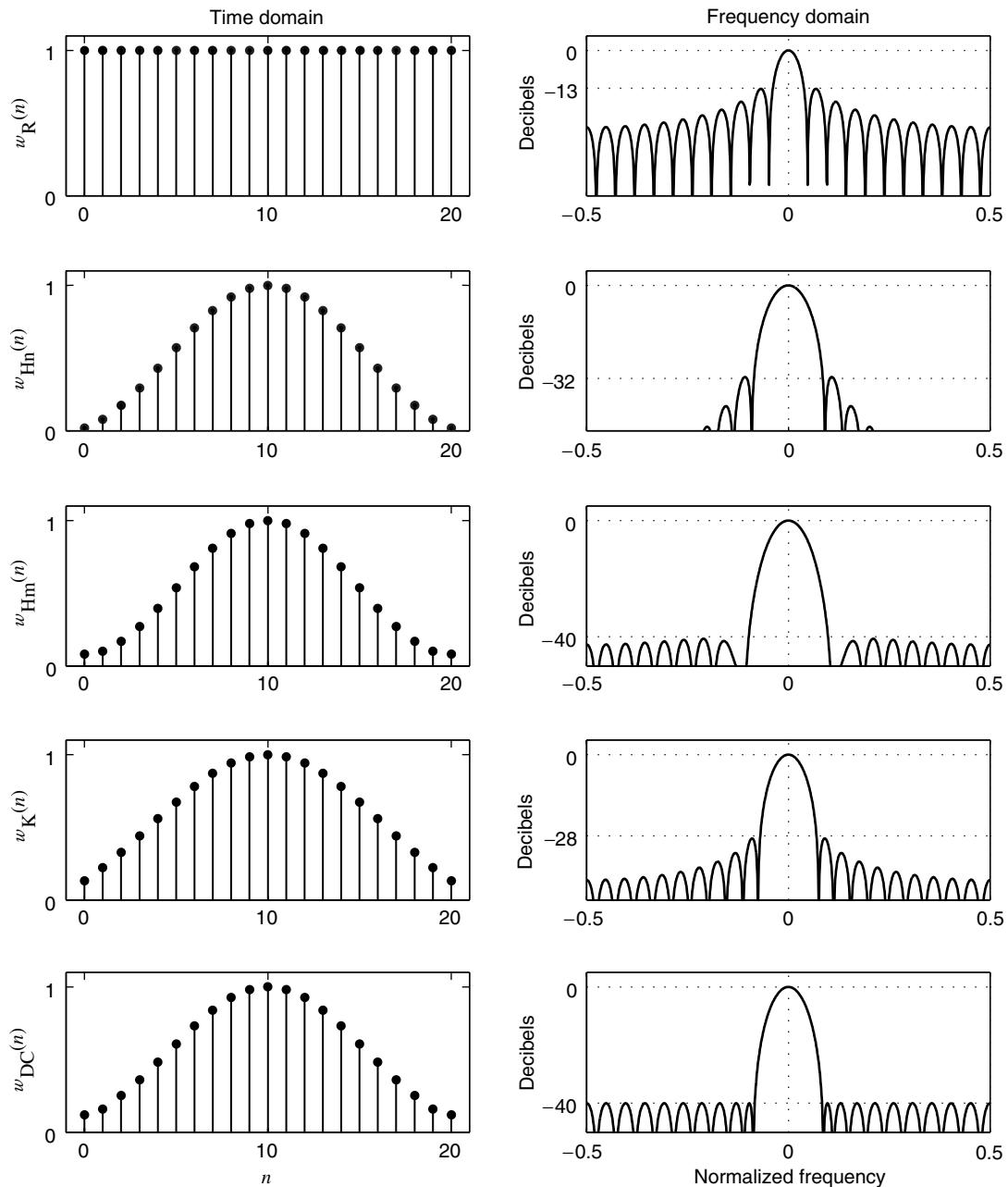


FIGURE 5.8

Time-domain window functions and their frequency-domain characteristics for rectangular, Hanning, Hamming, Kaiser, and Dolph-Chebyshev windows.

corresponding mainlobe widths. Clearly, $\beta = 0$ results in the rectangular window while $\beta > 0$ results in lower sidelobe leakage at the expense of a wider mainlobe. Kaiser has developed approximate design equations for β . Given a peak sidelobe level of A dB below the peak value, the approximate value of β is given by

$$\beta \simeq \begin{cases} 0 & A \leq 21 \\ 0.5842(A - 21)^{0.4} + 0.07886(A - 21) & 21 < A \leq 50 \\ 0.1102(A - 8.7) & A > 50 \end{cases} \quad (5.1.17)$$

Furthermore, to achieve the given values of the peak sidelobe level of A and the mainlobe width $\Delta\omega$, the length N must satisfy

$$\Delta\omega = \frac{A - 8}{2.285(N - 1)} \quad (5.1.18)$$

In MATLAB this window is given by the function `kaiser(N, beta)`.

Dolph-Chebyshev window. This window is characterized by the property that the peak sidelobe levels are constant; that is, it has an “equiripple” behavior. The window $w_{DC}(n)$ is obtained as the inverse DFT of the Chebyshev polynomial evaluated at N equally spaced frequencies around the unit circle. The details of this window function computation are available in Harris (1978). The parameters of the Dolph-Chebyshev window are the constant sidelobe level A in decibels, the window length N , and the mainlobe width $\Delta\omega$. However, only two of the three parameters can be independently specified. In spectrum estimation, parameters N and A are generally specified. Then $\Delta\omega$ is given by

$$\Delta\omega = \cos^{-1} \left[\left(\cosh \frac{\cosh^{-1} 10^{A/20}}{N - 1} \right)^{-1} \right] \quad (5.1.19)$$

In MATLAB this window is obtained through the function `chebwin(N, A)`.

To illustrate the usefulness of these windows, consider the same signal containing three frequencies given in (5.1.12). Figure 5.9 shows the spectrum of $x_N(n)$ using the

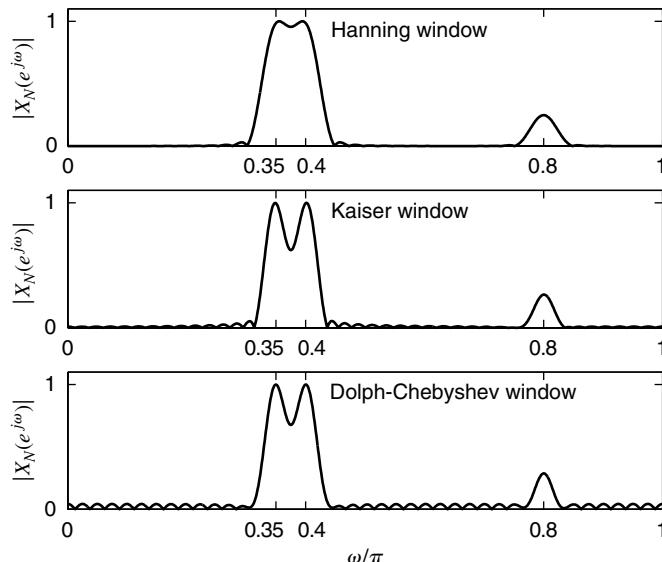


FIGURE 5.9

Spectrum of three sinusoids using Hanning, Kaiser, and Chebyshev windows.

Hanning, Kaiser, and Chebyshev windows for length $N = 81$. The Kaiser and Chebyshev window parameters are adjusted so that the peak sidelobe level is 40 dB or below. Clearly, these windows have suppressed sidelobes considerably compared to that of the rectangular window but the main peaks are wider with negligible bias. The two peaks in the Hanning window spectrum are barely resolved because the mainlobe width of this window is much wider than that of the rectangular window. The Chebyshev window spectrum has uniform sidelobes while the Kaiser window spectrum shows decreasing sidelobes away from the mainlobes.

5.1.5 Summary

In conclusion, the frequency analysis of deterministic signals requires a careful study of three important steps. First, the continuous-time signal $x_c(t)$ is sampled to obtain samples $x(n)$ that are collected into blocks or frames. The frames are “conditioned” to minimize certain errors by multiplying by a window sequence $w(n)$ of length N . Finally the windowed frames $x_N(n)$ are transformed to the frequency domain using the DFT. The resulting DFT spectrum $\tilde{X}_N(k)$ is a *faithful* replica of the actual spectrum $X_c(F)$ if the following errors are sufficiently small.

Aliasing error. This is an error due to the sampling operation. If the sampling rate is sufficiently high and if the antialiasing filter is properly designed so that most of the frequencies of interest are represented in $x(n)$, then this error can be made smaller. However, a certain amount of aliasing should be expected. The sampling principle and aliasing are discussed in Section 2.2.2.

Errors due to finite-length window. There are several errors such as resolution loss, bias, and leakage that are attributed to the windowing operation. Therefore, a careful design of the window function and its length is necessary to minimize these errors. These topics were discussed in Section 5.1.4. In Table 5.1 we summarize key properties of five windows discussed in this section that are useful for spectrum estimation.

Spectrum reconstruction error. The DFT spectrum $\tilde{X}_N(k)$ is a number sequence that must be reconstructed into a continuous function for the purpose of plotting. A practical choice for this reconstruction is the first-order polynomial interpolation. This reconstruction error can be made smaller (and in fact comparable to the screen resolution) by choosing a large number of frequency samples, which can be achieved by the *zero padding* operation in the DFT. It was discussed in Section 5.1.3.

With the understanding of frequency analysis concepts developed in this section, we are now ready to tackle the problem of spectral analysis of stationary random signals. From Chapter 3, we recognize that the true spectral values can only be obtained as estimates. This requires some understanding of key concepts from estimation theory, which is developed in Section 3.6.

5.2 ESTIMATION OF THE AUTOCORRELATION OF STATIONARY RANDOM SIGNALS

The second-order moments of a stationary random sequence—that is, the mean value μ_x , the autocorrelation sequence $r_x(l)$, and the PSD $R_x(e^{j\omega})$ —play a crucial role in signal analysis and signal modeling. In this section, we discuss the estimation of the autocorrelation sequence $r_x(l)$ using a finite data record $\{x(n)\}_0^{N-1}$ of the process.

For a stationary process $x(n)$, the most widely used estimator of $r_x(l)$ is given by the *sample autocorrelation sequence*

$$\hat{r}_x(l) \triangleq \begin{cases} \frac{1}{N} \sum_{n=0}^{N-l-1} x(n+l)x^*(n) & 0 \leq l \leq N-1 \\ \hat{r}_x^*(-l) & -(N-1) \leq l < 0 \\ 0 & \text{elsewhere} \end{cases} \quad (5.2.1)$$

or, equivalently,

$$\hat{r}_x(l) \triangleq \begin{cases} \frac{1}{N} \sum_{n=l}^{N-1} x(n)x^*(n-l) & 0 \leq l \leq N-1 \\ \hat{r}_x^*(-l) & -(N-1) \leq l < 0 \\ 0 & \text{elsewhere} \end{cases} \quad (5.2.2)$$

which is a random sequence. Note that without further information beyond the observed data $\{x(n)\}_{n=0}^{N-1}$, it is not possible to provide reasonable estimates of $r_x(l)$ for $|l| \geq N$. Even for lag values $|l|$ close to N , the correlation estimates are unreliable since very few $x(n+|l|)x(n)$ pairs are used. A good rule of thumb provided by Box and Jenkins (1976) is that N should be at least 50 and that $|l| \leq N/4$. The sample autocorrelation $\hat{r}_x(l)$ given in (5.2.1) has a desirable property that for each $l \geq 1$, the sample autocorrelation matrix

$$\hat{\mathbf{R}}_x = \begin{bmatrix} \hat{r}_x(0) & \hat{r}_x^*(1) & \cdots & \hat{r}_x^*(N-1) \\ \hat{r}_x(1) & \hat{r}_x(0) & \cdots & \hat{r}_x^*(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{r}_x(N-1) & \hat{r}_x(N-2) & \cdots & \hat{r}_x(0) \end{bmatrix} \quad (5.2.3)$$

is nonnegative definite (see Section 3.5.1). This property is explored in Problem 5.5. MATLAB provides functions to compute the correlation matrix $\hat{\mathbf{R}}_x$ (for example, `corr`), given the data $\{x(n)\}_{n=0}^{N-1}$; however, the book toolbox function `rx = autoc(x, L)` computes $\hat{r}_x(l)$ according to (5.2.1) very efficiently.

The estimate of covariance $\gamma_x(l)$ from the data record $\{x(n)\}_{n=0}^{N-1}$ is given by the sample autocovariance sequence

$$\hat{\gamma}_x(l) = \begin{cases} \frac{1}{N} \sum_{n=0}^{N-l-1} [x(n+l) - \hat{\mu}_x][x^*(n) - \hat{\mu}_x^*] & 0 \leq l \leq N-1 \\ \hat{\gamma}_x^*(-l) & -(N-1) \leq l < 0 \\ 0 & \text{elsewhere} \end{cases} \quad (5.2.4)$$

so that the corresponding autocovariance matrix $\hat{\mathbf{I}}_x$ is nonnegative definite. Similarly, the sample autocorrelation coefficient sequence $\hat{\rho}_x(l)$ is given by

$$\hat{\rho}_x(l) = \frac{\hat{\gamma}_x(l)}{\hat{\sigma}_x^2} \quad (5.2.5)$$

In the rest of this section, we assume that $x(n)$ is a zero-mean process and hence $\hat{r}_x(l) = \hat{\gamma}_x(l)$, so that we can discuss the autocorrelation estimate in detail.

To determine the statistical quality of this estimator, we now consider its mean and variance.

Mean of $\hat{r}_x(l)$. We first note that (5.2.1) can be written as

$$\hat{r}_x(l) = \frac{1}{N} \sum_{n=-\infty}^{\infty} x(n+l)w(n+l)x^*(n)w(n) \quad |l| \geq 0 \quad (5.2.6)$$

where $w(n) = w_R(n) = \begin{cases} 1 & 0 \leq n \leq N-1 \\ 0 & \text{elsewhere} \end{cases}$ (5.2.7)

is the rectangular window. The expected value of $\hat{r}_x(l)$ is

$$E\{\hat{r}_x(l)\} = \frac{1}{N} \sum_{n=-\infty}^{\infty} E\{x(n+l)x^*(n)\}w(n+l)w(n) \quad l \geq 0$$

and

$$E\{\hat{r}_x(-l)\} = E\{\hat{r}_x^*(-l)\} \quad -l \leq 0$$

Therefore

$$E\{\hat{r}_x(l)\} = \frac{1}{N} r_x(l) r_w(l) \quad (5.2.8)$$

where

$$r_w(l) = w(l) * w(-l) = \sum_{n=-\infty}^{\infty} w(n)w(n+l) \quad (5.2.9)$$

is the autocorrelation of the window sequence. For the rectangular window

$$r_w(l) = w_B(n) \triangleq \begin{cases} N - |l| & |l| \leq N-1 \\ 0 & \text{elsewhere} \end{cases} \quad (5.2.10)$$

which is the unnormalized triangular or Bartlett window. Thus

$$E\{\hat{r}_x(l)\} = \frac{1}{N} r_x(l) w_B(n) = r_x(l) \left(1 - \frac{|l|}{N}\right) w_R(n) \quad (5.2.11)$$

Therefore, we conclude that the relation (5.2.1) provides a *biased* estimate of $r_x(l)$ because the expected value of $\hat{r}_x(l)$ from (5.2.11) is not equal to the true autocorrelation $r_x(l)$. However, $\hat{r}_x(l)$ is an *asymptotically unbiased* estimator since if $N \rightarrow \infty$, $E\{\hat{r}_x(l)\} \rightarrow r_x(l)$. Clearly, the bias is small if $\hat{r}_x(l)$ is evaluated for $|l| \leq L$, where L is the maximum desired lag and $L \ll N$.

Variance of $\hat{r}_x(l)$. An approximate expression for the covariance of $\hat{r}_x(l)$ is given by Jenkins and Watts (1968)

$$\text{cov}\{\hat{r}_x(l_1), \hat{r}_x(l_2)\} \simeq \frac{1}{N} \sum_{l=-\infty}^{\infty} [r_x(l)r_x(l+l_2-l_1) + r_x(l+l_2)r_x(l-l_1)] \quad (5.2.12)$$

This indicates that successive values of $\hat{r}_x(l)$ may be highly correlated and that $\hat{r}_x(l)$ may fail to die out even if it is expected to. This makes the interpretation of autocorrelation graphs quite challenging because we do not know whether the variation is real or statistical.

The variance of $\hat{r}_x(l)$, which can be obtained by setting $l_1 = l_2$ in (5.2.12), tends to zero as $N \rightarrow \infty$. Thus, $\hat{r}_x(l)$ provides a good estimate of $r_x(l)$ if the lag $|l|$ is much smaller than N . However, as $|l|$ approaches N , fewer and fewer samples of $x(n)$ are used to evaluate $\hat{r}_x(l)$. As a result, the estimate $\hat{r}_x(l)$ becomes worse and its variance increases.

Nonnegative definiteness of $\hat{r}_x(l)$. An alternative estimator for the autocorrelation sequence is given by

$$\check{r}_x(l) = \begin{cases} \frac{1}{N-l} \sum_{n=0}^{N-l-1} x(n+l)x^*(n) & 0 \leq l \leq L < N \\ \check{r}_x^*(-l) & -N < -L \leq l < 0 \\ 0 & \text{elsewhere} \end{cases} \quad (5.2.13)$$

Although this estimator is unbiased, it is not used in spectral estimation because of its negative definiteness. In contrast, the estimator $\hat{r}_x(l)$ from (5.2.1) is nonnegative definite,

and any spectral estimates based on it do not have any negative values. Furthermore, the estimator $\hat{r}_x(l)$ has smaller variance and mean square error than the estimator $\check{r}_x(l)$ (Jenkins and Watts 1968). Thus, in this book we use the estimator $\hat{r}_x(l)$ defined in (5.2.1).

5.3 ESTIMATION OF THE POWER SPECTRUM OF STATIONARY RANDOM SIGNALS

From a practical point of view, most stationary random processes have continuous spectra. However, harmonic processes (i.e., processes with line spectra) appear in several applications either alone or in mixed spectra (a mixture of continuous and line spectra). We first discuss the estimation of continuous spectra in detail. The estimation of line spectra is considered in Chapter 9.

The power spectral density of a zero-mean stationary stochastic process was defined in (3.3.39) as

$$R_x(e^{j\omega}) \triangleq \sum_{l=-\infty}^{\infty} r_x(l)e^{-j\omega l} \quad (5.3.1)$$

assuming that the autocorrelation sequence $r_x(l)$ is absolutely summable. We will deal with the problem of estimating the power spectrum $R_x(e^{j\omega})$ of a stationary process $x(n)$ from a finite record of observations $\{x(n)\}_0^{N-1}$ of a single realization. The ideal goal is to devise an estimate that will faithfully characterize the power-versus-frequency distribution of the stochastic process (i.e., all the sequences of the ensemble) using only a segment of a single realization. For this to be possible, the estimate should typically involve some kind of averaging among several realizations or along a single realization.

In some practical applications (e.g., interferometry), it is possible to directly measure the autocorrelation $r_x(l)$, $|l| \leq L < N$ with great accuracy. In this case, the spectrum estimation problem can be treated as a deterministic one, as described in Section 5.1. We will focus on the “stochastic” version of the problem, where $R_x(e^{j\omega})$ is estimated from the available data $\{x(n)\}_0^{N-1}$. A natural estimate of $R_x(e^{j\omega})$, suggested by (5.3.1), is to estimate $r_x(l)$ from the available data and then transform it by using (5.3.1).

5.3.1 Power Spectrum Estimation Using the Periodogram

The periodogram is an estimator of the power spectrum, introduced by Schuster (1898) in his efforts to search for hidden periodicities in solar sunspot data. The *periodogram* of the data segment $\{x(n)\}_0^{N-1}$ is defined by

$$\hat{R}_x(e^{j\omega}) \triangleq \frac{1}{N} \left| \sum_{n=0}^{N-1} v(n)e^{-j\omega n} \right|^2 = \frac{1}{N} |V(e^{j\omega})|^2 \quad (5.3.2)$$

where $V(e^{j\omega})$ is the DTFT of the windowed sequence

$$v(n) = x(n)w(n) \quad 0 \leq n \leq N - 1 \quad (5.3.3)$$

The above definition of the periodogram stems from Parseval’s relation (2.2.10) on the power of a signal. The window $w(n)$, which has length N , is known as the *data window*. Usually, the term *periodogram* is used when $w(n)$ is a rectangular window. In contrast, the term *modified periodogram* is used to stress the use of nonrectangular windows. The values of the periodogram at the discrete set of frequencies $\{\omega_k = 2\pi k/N\}_0^{N-1}$ can be calculated by

$$\tilde{\hat{R}}_x(k) \triangleq \hat{R}_x(e^{j2\pi k/N}) = \frac{1}{N} |\tilde{V}(k)|^2 \quad k = 0, 1, \dots, N - 1 \quad (5.3.4)$$

where $\tilde{V}(k)$ is the N -point DFT of the windowed segment $v(n)$. In MATLAB, the modified periodogram computation is implemented by using the function

```
Rx = psd(x,Nfft,Fs,window(N), 'none');
```

where *window* is the name of any MATLAB-provided window function (e.g., hamming); *Nfft* is the size of the DFT, which is chosen to be larger than N to obtain a high-density spectrum (see zero padding in Section 5.1.1); and *Fs* is the sampling frequency, which is used for plotting purposes. If the window boxcar is used, then we obtain the periodogram estimate.

The periodogram can be expressed in terms of the autocorrelation estimate $\hat{r}_v(l)$ of the windowed sequence $v(n)$ as (see Problem 5.9)

$$\hat{R}_x(e^{j\omega}) = \sum_{l=-(N-1)}^{N-1} \hat{r}_v(l) e^{-j\omega l} \quad (5.3.5)$$

which shows that $\hat{R}_x(e^{j\omega})$ is a “natural” estimate of the power spectrum. From (5.3.2) it follows that $\hat{R}_x(e^{j\omega})$ is nonnegative for all frequencies ω . This results from the fact that the autocorrelation sequence $\hat{r}(l)$, $0 \leq |l| \leq N - 1$, is nonnegative definite. If we use the estimate $\check{r}_x(l)$ from (5.2.13) in (5.3.5) instead of $\hat{r}_x(l)$, the obtained periodogram may assume negative values, which implies that $\check{r}_x(l)$ is not guaranteed to be nonnegative definite.

The inverse Fourier transform of $\hat{R}_x(e^{j\omega})$ provides the estimated autocorrelation $\hat{r}_v(l)$, that is,

$$\hat{r}_v(l) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \hat{R}_x(e^{j\omega}) e^{j\omega l} d\omega \quad (5.3.6)$$

because $\hat{r}_v(l)$ and $\hat{R}_x(e^{j\omega})$ form a DTFT pair. Using (5.3.6) and (5.2.1) for $l = 0$, we have

$$\hat{r}_v(0) = \frac{1}{N} \sum_{n=0}^{N-1} |v(n)|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} \hat{R}_x(e^{j\omega}) d\omega \quad (5.3.7)$$

Thus, the periodogram $\hat{R}_x(e^{j\omega})$ shows how the power of the segment $\{v(n)\}_0^{N-1}$, which provides an estimate of the variance of the process $x(n)$, is distributed as a function of frequency.

Filter bank interpretation. The above assertion that the periodogram describes a distribution of power as a function of frequency can be interpreted in a different way, in which the power estimate over a narrow frequency band is attributed to the output power of a narrow-bandpass filter. This leads to the well-known *filter bank interpretation* of the periodogram. To develop this interpretation, consider the basic (unwindowed) periodogram estimator $\hat{R}_x(e^{j\omega})$ in (5.3.2), evaluated at a frequency $\omega_k \triangleq k\Delta\omega \triangleq 2\pi k/N$, which can be expressed as

$$\begin{aligned} \hat{R}_x(e^{j\omega_k}) &= \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n) e^{-j\omega_k n} \right|^2 = \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n) e^{j2\pi k - j\omega_k n} \right|^2 \\ &= \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n) e^{j\omega_k(N-n)} \right|^2 \quad \text{since } \omega_k N = 2\pi k \quad (5.3.8) \\ &= \frac{1}{N} \left| \sum_{m=0}^{N-1} x(N-m) e^{j\omega_k m} \right|^2 \end{aligned}$$

Clearly, the term inside the absolute value sign in (5.3.8) can be interpreted as a convolution of $x(n)$ and $e^{j\omega_k n}$, evaluated at $n = N$. Define

$$h_k(n) \triangleq \begin{cases} \frac{1}{N} e^{j\omega_k n} & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases} \quad (5.3.9)$$

as the impulse response of a linear system whose frequency response is given by

$$\begin{aligned} H_k(e^{j\omega}) &= \mathcal{F}[h_k(n)] = \frac{1}{N} \sum_{n=0}^{N-1} e^{j\omega_k n} e^{-j\omega n} \\ &= \frac{1}{N} \sum_{n=0}^{N-1} e^{-j(\omega - \omega_k)n} = \frac{1}{N} \frac{e^{-jN(\omega - \omega_k)} - 1}{e^{-j(\omega - \omega_k)} - 1} \\ &= \frac{1}{N} \frac{\sin[N(\omega - \omega_k)/2]}{\sin[(\omega - \omega_k)/2]} e^{-j(N-1)(\omega - \omega_k)/2} \end{aligned} \quad (5.3.10)$$

which is a linear-phase, narrow-bandpass filter centered at $\omega = \omega_k$. The 3-dB bandwidth of this filter is proportional to $2\pi/N$ rad per sampling interval (or $1/N$ cycles per sampling interval). A plot of the magnitude response $|H_k(e^{j\omega})|$, for $\omega_k = \pi/2$ and $N = 50$, is shown in Figure 5.10, which evidently shows the narrowband nature of the filter.

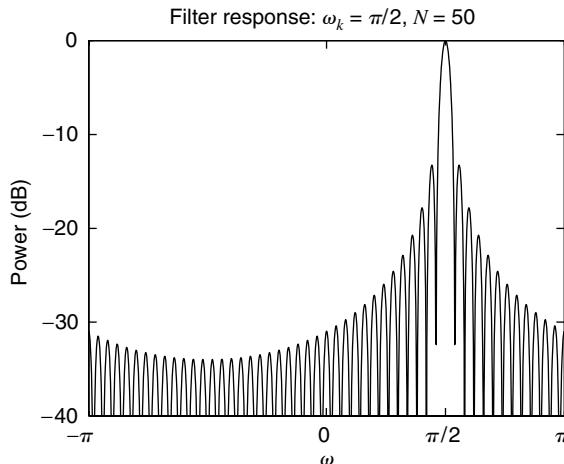


FIGURE 5.10

The magnitude of the frequency response of the narrow-bandpass filter for $\omega_k = \pi/2$ and $N = 50$.

Continuing, we also define the output of the filter $h_k(n)$ by $y_k(n)$, that is,

$$y_k(n) \triangleq h_k(n) * x(n) = \frac{1}{N} \sum_{m=0}^{N-1} x(n-m) e^{j\omega_k m} \quad (5.3.11)$$

Then (5.3.8) can be written as

$$\hat{R}_x(e^{j\omega_k}) = N |y_k(N)|^2 \quad (5.3.12)$$

Now consider the average power in $y_k(n)$, which can be evaluated using the spectral density as [see (3.3.45) and (3.4.22)]

$$\begin{aligned} E\{|y_k(n)|^2\} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} R_x(e^{j\omega}) |H_k(e^{j\omega})|^2 d\omega \\ &\approx \frac{\Delta\omega}{2\pi} R_x(e^{j\omega_k}) = \frac{1}{N} R_x(e^{j\omega_k}) \end{aligned} \quad (5.3.13)$$

since $H_k(e^{j\omega})$ is a narrowband filter. If we estimate the average power $E\{|y_k(n)|^2\}$ using one sample $y_k(N)$, then from (5.3.13) the estimated spectral density is the periodogram given

by (5.3.12), which says that the k th DFT sample of the periodogram [see (5.3.4)] is given by the average power of a *single* N th output sample of the ω_k -centered narrow-bandpass filter. Now imagine one such filter for each ω_k , $k = 0, \dots, N - 1$, frequencies. Thus we have a bank of filters, each tuned to the discrete frequency (based on the data record length), providing the periodogram estimates every N samples. This filter bank is inherently built into the periodogram and hence need not be explicitly implemented. The block diagram of this filter bank approach to the periodogram computation is shown in Figure 5.11.

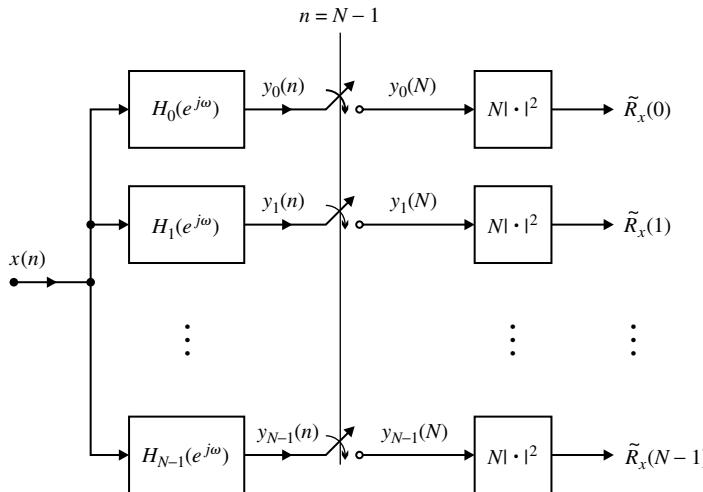


FIGURE 5.11

The filter bank approach to the periodogram computation.

In Section 5.1, we observed that the periodogram of a deterministic signal approaches the true energy spectrum as the number of observations $N \rightarrow \infty$. To see how the power spectrum of random signals is related to the number observations, we consider the following example.

EXAMPLE 5.3.1 (PERIODOGGRAM OF A SIMULATED WHITE NOISE SEQUENCE). Let $x(n)$ be a stationary white Gaussian noise with zero-mean and unit variance. The theoretical spectrum of $x(n)$ is

$$R_x(e^{j\omega}) = \sigma_x^2 = 1 \quad -\pi < \omega \leq \pi$$

To study the periodogram estimate, 50 different N -point records of $x(n)$ were generated using a pseudorandom number generator. The periodogram $\hat{R}_x(e^{j\omega})$ of each record was computed for $\omega = \omega_k = 2\pi k/1024$, $k = 0, 1, \dots, 512$, that is, with $N_{FFT} = 1024$, from the available data using (5.3.4) for $N = 32, 128$, and 256 . These results in the form of periodogram overlays (a Monte Carlo simulation) and their averages are shown in Figure 5.12. We notice that $\hat{R}_x(e^{j\omega})$ fluctuates so erratically that it is impossible to conclude from its observation that the signal has a flat spectrum. Furthermore, the size of the fluctuations (as seen from the ensemble average) is not reduced by increasing the segment length N . In this sense, we should not expect the periodogram $\hat{R}_x(e^{j\omega})$ to converge to the true spectrum $R_x(e^{j\omega})$ in some statistical sense as $N \rightarrow \infty$. Since $R_x(e^{j\omega})$ is constant over frequency, the fluctuations of $\hat{R}_x(e^{j\omega})$ can be characterized by their mean, variance, and mean square error over frequency for each N and are given in Table 5.2. It can be seen that although the mean value tends to 1 (true value), the standard deviation is not reduced as N increases. In fact, it is close to 1; that is, it is of the order of the size of the quantity to be estimated. This illustrates that the periodogram is not a good estimate of the power spectrum.

Since for each value of ω , $\hat{R}_x(e^{j\omega})$ is a random variable, the erratic behavior of the periodogram estimator, which is illustrated in Figure 5.12, can be explained by considering its mean, covariance, and variance.

TABLE 5.2
Performance of periodogram for white Gaussian noise signal in Example 5.3.1.

N	32	128	256
$\hat{E}[R_x(e^{j\omega_k})]$	0.7829	0.8954	0.9963
$\hat{\text{var}}[R_x(e^{j\omega_k})]$	0.7232	1.0635	1.1762
MSE	0.7689	1.07244	1.1739

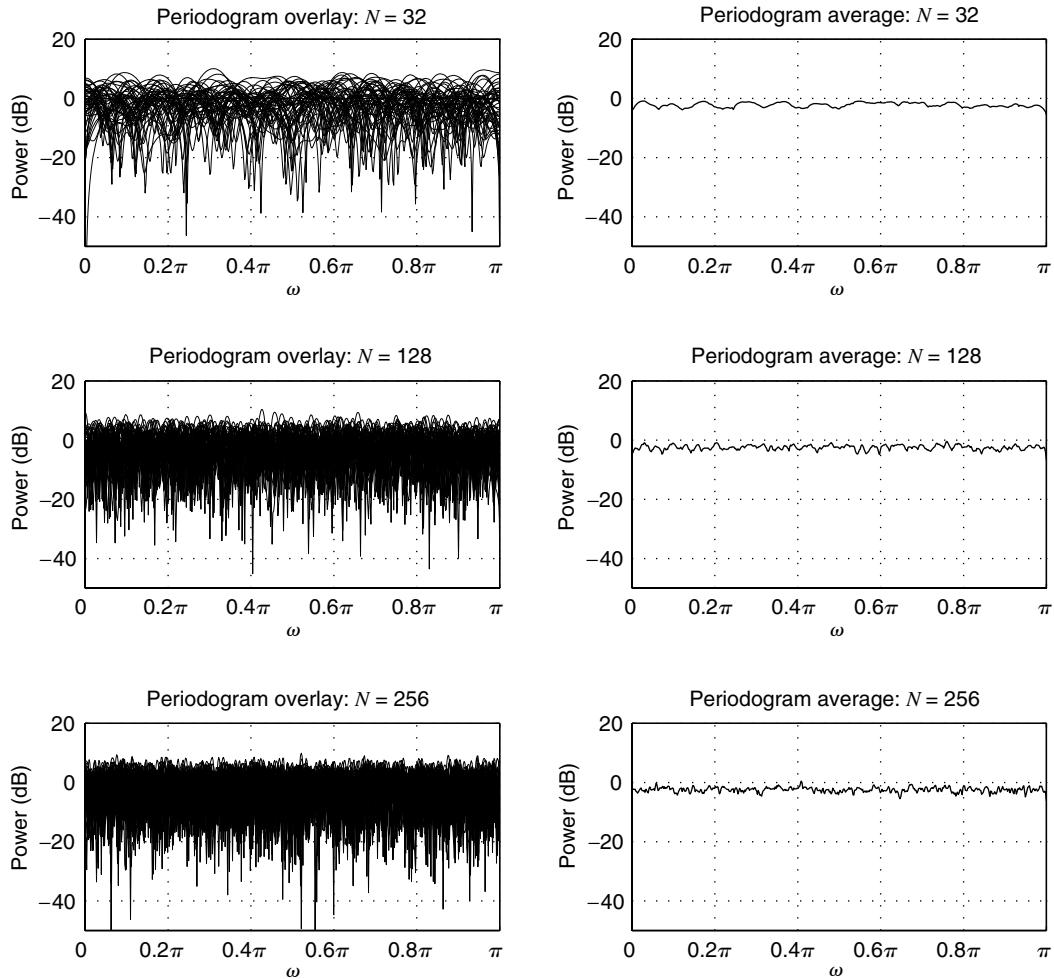


FIGURE 5.12
 Periodograms of white Gaussian noise in Example 5.3.1.

Mean of $\hat{R}_x(e^{j\omega})$. Taking the mathematical expectation of (5.3.5) and using (5.2.8), we obtain

$$E\{\hat{R}_x(e^{j\omega})\} = \sum_{l=-(N-1)}^{N-1} E\{\hat{r}_v(l)\}e^{-j\omega l} = \frac{1}{N} \sum_{l=-(N-1)}^{N-1} r_x(l)r_w(l)e^{-j\omega l} \quad (5.3.14)$$

Since $E\{\hat{R}_x(e^{j\omega})\} \neq R_x(e^{j\omega})$, the periodogram is a biased estimate of the true power spectrum $R_x(e^{j\omega})$.

Equation (5.3.14) can be interpreted in the frequency domain as a periodic convolution. Indeed, using the frequency domain convolution theorem, we have

$$E\{\hat{R}_x(e^{j\omega})\} = \frac{1}{2\pi N} \int_{-\pi}^{\pi} R_x(e^{j\theta}) R_w(e^{j(\omega-\theta)}) d\theta \quad (5.3.15)$$

where

$$R_w(e^{j\omega}) = |W(e^{j\omega})|^2 \quad (5.3.16)$$

is the spectrum of the window. Thus, the expected value of the periodogram is obtained by convolving the true spectrum $R_x(e^{j\omega})$ with the spectrum $R_w(e^{j\omega})$ of the window. This is equivalent to windowing the true autocorrelation $r_x(l)$ with the *correlation* or *lag* window $r_w(l) = w(l) * w(-l)$, where $w(n)$ is the data window.

To understand the implications of (5.3.15), consider the rectangular data window (5.2.7). Using (5.2.11), we see that (5.3.14) becomes

$$E\{\hat{R}_x(e^{j\omega})\} = \sum_{l=-(N-1)}^{N-1} \left(1 - \frac{|l|}{N}\right) r_x(l) e^{-j\omega l} \quad (5.3.17)$$

For nonperiodic autocorrelations, the value of $r_x(l)$ becomes negligible for large values of $|l|$. Hence, as the record length N increases, the term $(1 - |l|/N) \rightarrow 1$ for all l , which implies that

$$\lim_{N \rightarrow \infty} E\{\hat{R}_x(e^{j\omega})\} = R_x(e^{j\omega}) \quad (5.3.18)$$

that is, the periodogram is an *asymptotically unbiased* estimator of $R_x(e^{j\omega})$. In the frequency domain, we obtain

$$R_w(e^{j\omega}) = \mathcal{F}\{w_R(l) * w_R(-l)\} = |W_R(e^{j\omega})|^2 = \left[\frac{\sin(\omega N/2)}{\sin(\omega/2)} \right]^2 \quad (5.3.19)$$

where

$$W_R(e^{j\omega}) = e^{-j\omega(N-1)/2} \frac{\sin(\omega N/2)}{\sin(\omega/2)} \quad (5.3.20)$$

is the Fourier transform of the rectangular window. The spectrum $R_w(e^{j\omega})$, in (5.3.19), of the correlation window $r_w(l)$ approaches a periodic impulse train as the window length increases.[†] As a result, $E\{\hat{R}_x(e^{j\omega})\}$ approaches the true power spectrum $R_x(e^{j\omega})$ as N approaches ∞ .

The result (5.3.18) holds for any window that satisfies the following two conditions:

1. The window is normalized such that

$$\sum_{n=0}^{N-1} |w(n)|^2 = N \quad (5.3.21)$$

This condition is obtained by noting that, for asymptotic unbiasedness, we want $R_w(e^{j\omega})/N$ in (5.3.15) to be an approximation of an impulse in the frequency domain. Since the area under the impulse function is unity, using (5.3.16) and Parseval's theorem, we have

$$\frac{1}{2\pi N} \int_{-\pi}^{\pi} |W(e^{j\omega})|^2 d\omega = \frac{1}{N} \sum_{n=0}^{N-1} |w(n)|^2 = 1 \quad (5.3.22)$$

2. The width of the mainlobe of the spectrum $R_w(e^{j\omega})$ of the correlation window decreases as $1/N$. This condition guarantees that the area under $R_w(e^{j\omega})$ is concentrated at the origin as N becomes large. For more precise conditions see Brockwell and Davis (1991).

[†]This spectrum is sometimes referred to as the *Fejer kernel*.

The bias is introduced by the sidelobes of the correlation window through leakage, as illustrated in Section 5.1. Therefore, we can reduce the bias by using the modified periodogram and a “better” window. Bias can be avoided if either $N = \infty$, in which case the spectrum of the window is a periodic train of impulses, or $R_x(e^{j\omega}) = \sigma_x^2$, that is, $x(n)$ has a flat power spectrum. Thus, for white noise, $\hat{R}_x(e^{j\omega})$ is unbiased for all N . This fact was apparent in Example 5.3.1 and is very important for practical applications. In the following example, we illustrate that the bias becomes worse as the dynamic range of the spectrum increases.

EXAMPLE 5.3.2 (BIAS AND LEAKAGE PROPERTIES OF THE PERIODOGRAM). Consider an AR(2) process with

$$\mathbf{a}_2 = [1 \ -0.75 \ 0.5]^T \quad d_0 = 1 \quad (5.3.23)$$

and an AR(4) process with

$$\mathbf{a}_4 = [1 \ -2.7607 \ 3.8106 \ -2.6535 \ 0.9238]^T \quad d_0 = 1 \quad (5.3.24)$$

where $w(n) \sim WN(0, 1)$. Both processes have been used extensively in the literature for power spectrum estimation studies (Percival and Walden 1993). Their power spectrum is given by (see Chapter 4)

$$R_x(e^{j\omega}) = \frac{\sigma_w^2 d_0}{|A(e^{j\omega})|^2} = \frac{\sigma_w^2}{\left| \sum_{k=0}^p a_k e^{j\omega k} \right|^2} \quad (5.3.25)$$

For simulation purposes, $N = 1024$ samples of each process were generated. The sample realizations and the shapes of the two power spectra in (5.3.25) are shown in Figure 5.13. The dynamic range of the two spectra, that is, $\max_{\omega} R_x(e^{j\omega}) / \min_{\omega} R_x(e^{j\omega})$, is about 15 and 65 dB, respectively.

From the sample realizations, periodograms and modified periodograms, based on the Hanning window, were computed by using (5.3.4) at $N_{FFT} = 1024$ frequencies. These are shown in Figure 5.14. The periodograms for the AR(2) and AR(4) processes, respectively, are shown in the

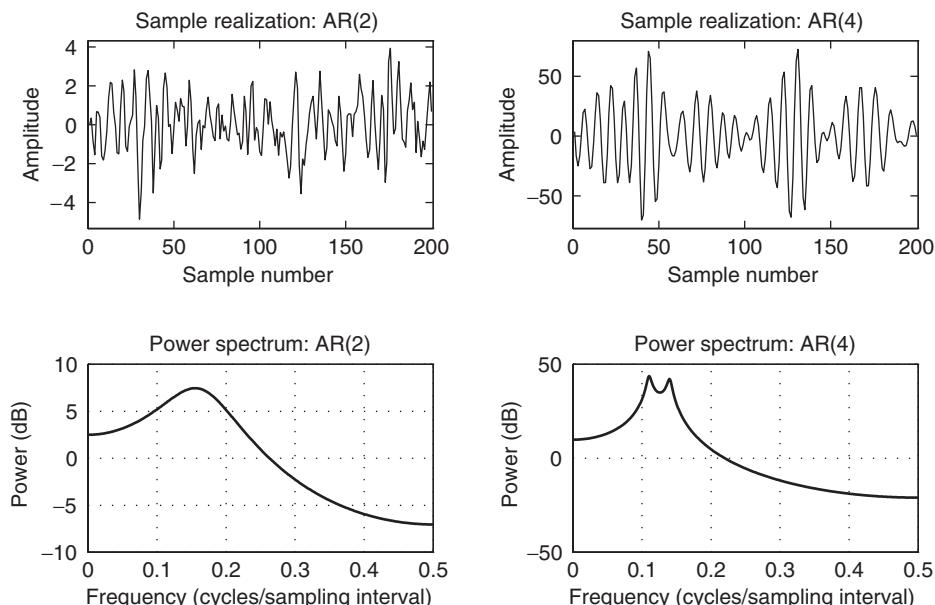


FIGURE 5.13

Sample realizations and power spectra of the AR(2) and AR(4) processes used in Example 5.3.2.

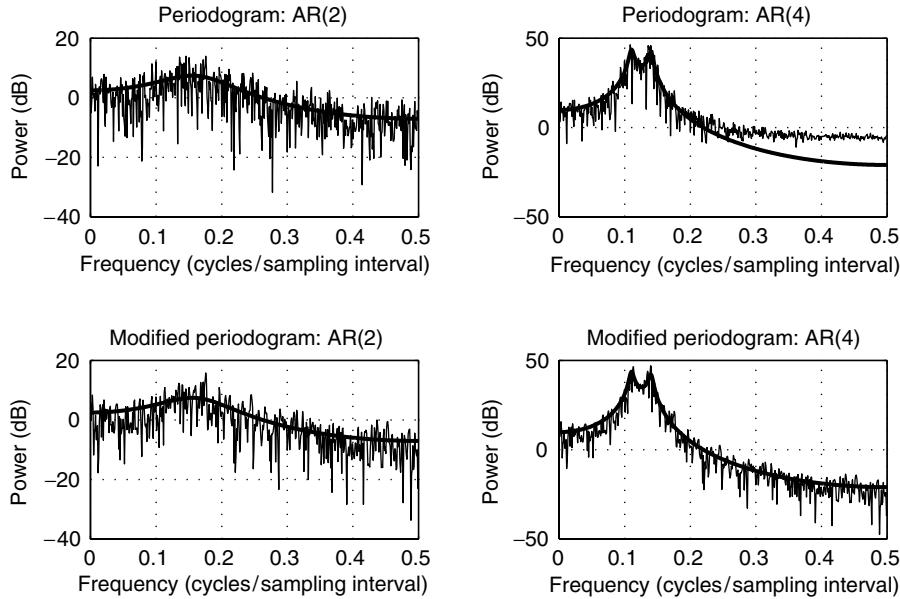
**FIGURE 5.14**

Illustration of properties of periodogram as a power spectrum estimator.

top row while the modified periodograms for the same processes are shown in the bottom row. These plots illustrate that the periodogram is a biased estimator of the power spectrum. In the case of the AR(2) process, since the spectrum has a small dynamic range (15 dB), the bias in the periodogram estimate is not obvious; furthermore, the windowing in the modified periodogram did not show much improvement. On the other hand, the AR(4) spectrum has a large dynamic range, and hence the bias is clearly visible at high frequencies. This bias is clearly reduced by windowing of the data in the modified periodogram. In both cases, the random fluctuations are not reduced by the data windowing operation.

EXAMPLE 5.3.3 (FREQUENCY RESOLUTION PROPERTY OF THE PERIODOGRAM). Consider two unit-amplitude sinusoids observed in unit variance white noise. Let

$$x(n) = \cos(0.35\pi n + \phi_1) + \cos(0.4\pi n + \phi_2) + v(n)$$

where ϕ_1 and ϕ_2 are jointly independent random variables uniformly distributed over $[-\pi, \pi]$ and $v(n)$ is a unit-variance white noise. Since two frequencies, 0.35π and 0.4π , are close, we will need (see Table 5.1)

$$N - 1 > \frac{1.81\pi}{0.4\pi - 0.35\pi} \quad \text{or} \quad N > 37$$

To obtain a periodogram ensemble, 50 realizations of $x(n)$ for $N = 32$ and $N = 64$ were generated, and their periodograms were computed. The plots of these periodogram overlays and the corresponding ensemble average for $N = 32$ and $N = 64$ are shown in Figure 5.15. For $N = 32$, frequencies in the periodogram cannot be resolved, as expected; but for $N = 64$ it is possible to separate the two sinusoids with ease. Note that the modified periodogram (i.e., data windowing) will not help since windowing increases smoothing and smearing of peaks.

The case of nonzero mean. In the periodogram method of spectrum analysis in this section, we assumed that the random signal has zero mean. If a random signal has nonzero mean, it should be estimated using (3.6.20) and then removed from the signal prior to computing its periodogram. This is because the power spectrum of a nonzero mean signal has an impulse at the zero frequency. If this mean is relatively large, then because of the leakage inherent in the periodogram, this mean will obscure low-amplitude, low-frequency

components of the spectrum. Even though the estimate is not an exact value, its removal often provides better estimates, especially at low frequencies.

Covariance of $\hat{R}_x(e^{j\omega})$. Obtaining an expression for the covariance of the periodogram is a rather complicated process. However, it has been shown (Jenkins and Watts 1968) that

$$\begin{aligned} \text{cov}\{\hat{R}_x(e^{j\omega_1}), \hat{R}_x(e^{j\omega_2})\} &\simeq R_x(e^{j\omega_1})R_x(e^{j\omega_2}) \left(\left\{ \frac{\sin[(\omega_1 + \omega_2)N/2]}{N \sin[(\omega_1 + \omega_2)/2]} \right\}^2 \right. \\ &\quad \left. + \left\{ \frac{\sin[(\omega_1 - \omega_2)N/2]}{N \sin[(\omega_1 - \omega_2)/2]} \right\}^2 \right) \end{aligned} \quad (5.3.26)$$

This expression applies to stationary random signals with zero mean and Gaussian probability density. The approximation becomes exact if the signal has a flat spectrum (white noise). Although this approximation deteriorates for non-Gaussian probability densities, the qualitative results that one can draw from this approximation appear to hold for a rather broad range of densities.

From (5.3.26), for $\omega_1 = (2\pi/N)k_1$ and $\omega_2 = (2\pi/N)k_2$ with k_1, k_2 integers, we have

$$\text{cov}\{\hat{R}_x(e^{j\omega_1})\hat{R}_x(e^{j\omega_2})\} \simeq 0 \quad \text{for } k_1 \neq k_2 \quad (5.3.27)$$

Thus, values of the periodogram spaced in frequency by integer multiples of $2\pi/N$ are approximately uncorrelated. As the record length N increases, these uncorrelated periodogram samples come closer together, and hence the rate of fluctuations in the periodogram increases. This explains the results in Figure 5.12.

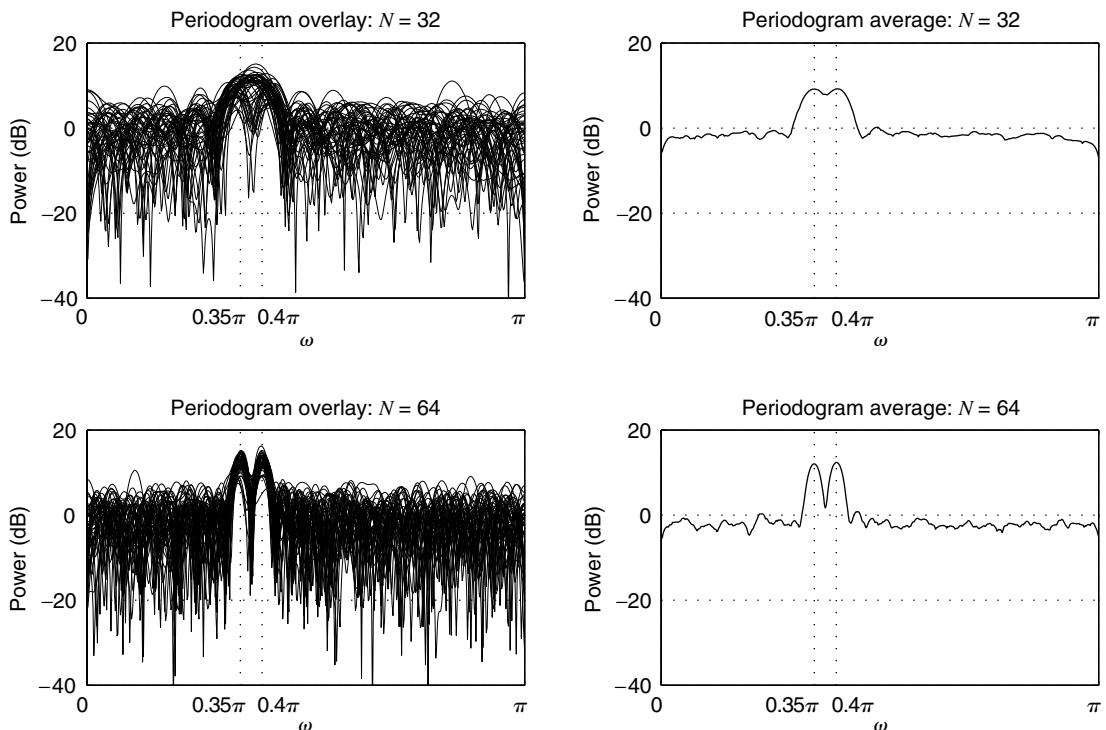


FIGURE 5.15

Illustration of the frequency resolution property of the periodogram in Example 5.3.3.

Variance of $\hat{R}_x(e^{j\omega})$. The variance of the periodogram at a particular frequency $\omega = \omega_1 = \omega_2$ can be obtained from (5.3.26)

$$\text{var}\{\hat{R}_x(e^{j\omega})\} \simeq R_x^2(e^{j\omega}) \left[1 + \left(\frac{\sin \omega N}{N \sin \omega} \right)^2 \right] \quad (5.3.28)$$

For large values of N , the variance of $\hat{R}_x(e^{j\omega})$ can be approximated by

$$\text{var}\{\hat{R}_x(e^{j\omega})\} \simeq \begin{cases} R_x^2(e^{j\omega}) & 0 < \omega < \pi \\ 2R_x^2(e^{j\omega}) & \omega = 0, \pi \end{cases} \quad (5.3.29)$$

This result is crucial, because it shows that the variance of the periodogram (estimate) remains at the level of $R_x^2(e^{j\omega})$ (quantity to be estimated), independent of the record length N used. Furthermore, since the variance does not tend to zero as $N \rightarrow \infty$, the periodogram is not a consistent estimator; that is, its distribution does not tend to cluster more closely around the true spectrum as N increases.[†]

This behavior was illustrated in Example 5.3.1. The variance of $\hat{R}_x(e^{j\omega_k})$ fails to decrease as N increases because the number of periodogram values $\hat{R}_x(e^{j\omega_k})$, $k = 0, 1, \dots, N - 1$, is always equal to the length N of the data record.

EXAMPLE 5.3.4 (COMPARISON OF PERIODOGRAM AND MODIFIED PERIODOGRAM).

Consider the case of three sinusoids discussed in Section 5.1.4. In particular, we assume that these sinusoids are observed in white noise with

$$x(n) = \cos(0.35\pi n + \phi_1) + \cos(0.4\pi n + \phi_2) + 0.25 \cos(0.8\pi n + \phi_3) + v(n)$$

where ϕ_1, ϕ_2 , and ϕ_3 are jointly independent random variables uniformly distributed over $[-\pi, \pi]$ and $v(n)$ is a unit-variance white noise. An ensemble of 50 realizations of $x(n)$ was generated using $N = 128$. The periodograms and the Hamming window-based modified periodograms of these realizations were computed, and the results are shown in Figure 5.16. The top row of the figure contains periodogram overlays and the corresponding ensemble average for the unwindowed periodogram, and the bottom row shows the same for the modified periodogram. Spurious peaks (especially near the two close frequencies) in the periodogram have been suppressed by the data windowing operation in the modified periodogram; hence the peak corresponding to 0.8π is sufficiently enhanced. This enhancement is clearly at the expense of the frequency resolution (or smearing of the true peaks), which is to be expected. The overall variance of the noise floor is still not reduced.

Failure of the periodogram

To conclude, we note that the periodogram in its “basic form” is a very poor estimator of the power spectrum function. The failure of the periodogram when applied to random signals is uniquely pointed out in Jenkins and Watts (1968, p. 213):

The basic reason why Fourier analysis breaks down when applied to time series is that it is based on the assumption of *fixed* amplitudes, frequencies and phases. Time series, on the other hand, are characterized by *random* changes of frequencies, amplitudes and phases. Therefore it is not surprising that Fourier methods need to be adapted to account for the random nature of a time series.

The attempt at improving the periodogram by windowing the available data, that is, by using the modified periodogram in Example 5.3.4, showed that the presence and the length of the window had no effect on the variance. The major problems with the periodogram lie in its variance, which is on the order of $R_x^2(e^{j\omega})$, as well as in its erratic behavior. Thus, to obtain a better estimator, we should reduce its variance; that is, we should “smooth” the periodogram.

[†]The definition of the PSD by $R_x(e^{j\omega}) = \lim_{N \rightarrow \infty} \hat{R}_x(e^{j\omega})$ is not valid because even if $\lim_{N \rightarrow \infty} E\{\hat{R}_x(e^{j\omega})\} = R_x(e^{j\omega})$, the variance of $\hat{R}_x(e^{j\omega})$ does not tend to zero as $N \rightarrow \infty$ (Papoulis 1991).

From the previous discussion, it follows that the sequence $\tilde{\hat{R}}_x(k)$, $k = 0, 1, \dots, N - 1$, of the harmonic periodogram components can be reasonably assumed to be a sequence of uncorrelated random variables. Furthermore, it is well known that the variance of the sum of K uncorrelated random variables with the same variance is $1/K$ times the variance of one of these individual random variables. This suggests two ways of reducing the variance, which also lead to smoother spectral estimators:

- Average contiguous values of the periodogram.
- Average periodograms obtained from multiple data segments.

It should be apparent that owing to stationarity, the two approaches should provide comparable results under similar circumstances.

5.3.2 Power Spectrum Estimation by Smoothing a Single Periodogram— The Blackman-Tukey Method

The idea of reducing the variance of the periodogram through smoothing using a moving-average filter was first proposed by Daniel (1946). The estimator proposed by Daniel is a zero-phase moving-average filter, given by

$$\hat{R}_x^{(\text{PS})}(e^{j\omega_k}) \triangleq \frac{1}{2M+1} \sum_{j=-M}^M \hat{R}_x(e^{j\omega_{k-j}}) \triangleq \sum_{j=-M}^M W(e^{j\omega_j}) \hat{R}_x(e^{j\omega_{k-j}}) \quad (5.3.30)$$

where $\omega_k = (2\pi/N)k$, $k = 0, 1, \dots, N - 1$, $W(e^{j\omega_j}) \triangleq 1/(2M + 1)$, and the superscript (PS) denotes periodogram smoothing. Since the samples of the periodogram are approxi-

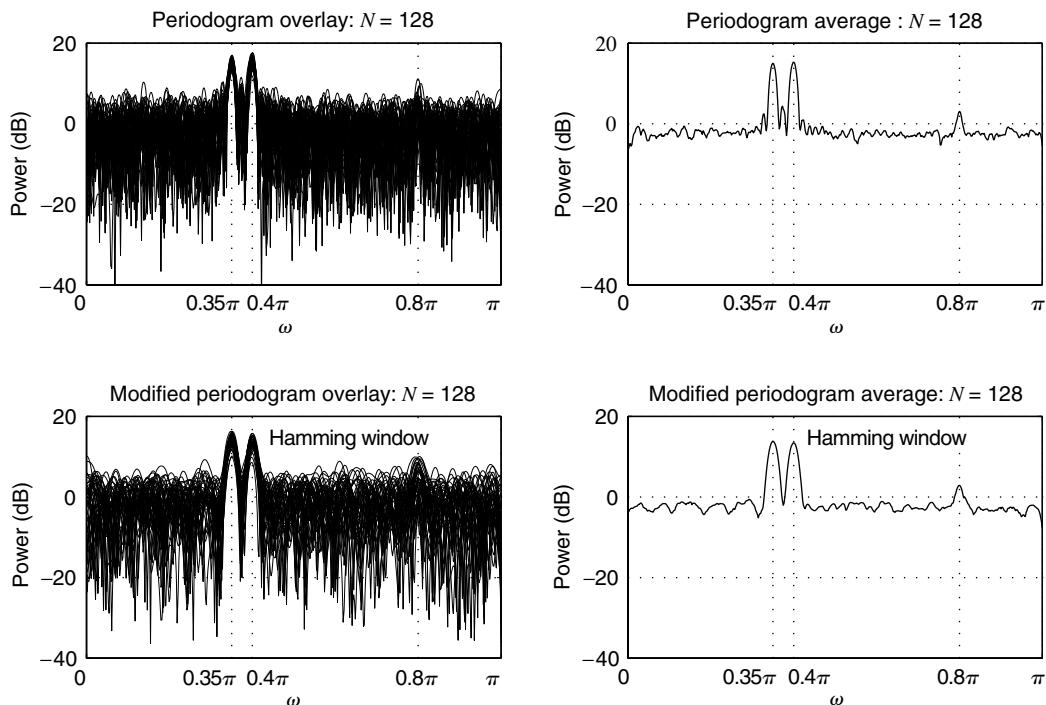


FIGURE 5.16

Comparison of periodogram and modified periodogram in Example 5.3.4.

$$\text{var}\{\hat{R}_x^{(\text{PS})}(e^{j\omega_k})\} \simeq \frac{1}{2M+1} \text{var}\{\hat{R}_x(e^{j\omega_k})\} \quad (5.3.31)$$

that is, averaging $2M + 1$ consecutive spectral lines reduces the variance by a factor of $2M + 1$. The quantity $\Delta\omega \approx (2\pi/N)(2M + 1)$ determines the frequency resolution, since any peaks within the $\Delta\omega$ range are smoothed over the entire interval $\Delta\omega$ into a single peak and cannot be resolved. Thus, increasing M reduces the variance (resulting in a smoother spectrum estimate), at the expense of spectral resolution. This is the fundamental tradeoff in practical spectral analysis.

Blackman-Tukey approach

The discrete moving average in (5.3.30) is computed in the frequency domain. We now introduce a better and simpler way to smooth the periodogram by operating on the estimated autocorrelation sequence. To this end, we note that the continuous frequency equivalent of the discrete convolution formula (5.3.30) is the periodic convolution

$$\hat{R}_x^{(\text{PS})}(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \hat{R}_x(e^{j(\omega-\theta)}) W_a(e^{j\theta}) d\theta = \hat{R}_x(e^{j\omega}) \otimes W_a(e^{j\omega}) \quad (5.3.32)$$

where $W_a(e^{j\omega})$ is a periodic function of ω with period 2π , given by

$$W_a(e^{j\omega}) = \begin{cases} \frac{1}{\Delta\omega} & |\omega| < \frac{\Delta\omega}{2} \\ 0 & \frac{\Delta\omega}{2} \leq \omega \leq \pi \end{cases} \quad (5.3.33)$$

By using the convolution theorem, (5.3.32) can be written as

$$\hat{R}_x^{(\text{PS})}(e^{j\omega}) = \sum_{l=-(L-1)}^{L-1} \hat{r}_x(l) w_a(l) e^{-j\omega l} \quad (5.3.34)$$

where $w_a(l)$ is the inverse Fourier transform of $W_a(e^{j\omega})$ and $L < N$. As we have already mentioned, the window $w_a(l)$ is known as the correlation or lag window.[†] The correlation window corresponding to (5.3.33) is

$$w_a(l) = \frac{\sin(l\Delta\omega/2)}{\pi l} \quad -\infty < l < \infty \quad (5.3.35)$$

Since $w_a(l)$ has infinite duration, its truncation at $|l| = L \leq N$ creates ripples in $W_a(e^{j\omega})$ (Gibbs effect). To avoid this problem, we use correlation windows with finite duration, that is, $w_a(l) = 0$ for $|l| > L \leq N$. For real sequences, where $\hat{r}_x(l)$ is real and even, $w_a(l)$ [and hence $W_a(e^{j\omega})$] should be real and even. Given that $\hat{R}_x(e^{j\omega})$ is nonnegative, a sufficient (but not necessary) condition that $\hat{R}_x^{(\text{PS})}(e^{j\omega})$ be nonnegative is that $W_a(e^{j\omega}) \geq 0$ for all ω . This condition holds for the Bartlett (triangular) and Parzen (see Problem 5.11) windows, but it does not hold for the Hamming, Hanning, or Kaiser window.

Thus, we note that smoothing the periodogram $\hat{R}_x(e^{j\omega})$ by convolving it with the spectrum $W_a(e^{j\omega}) = \mathcal{F}\{w_a(l)\}$ is equivalent to windowing the autocorrelation estimate $\hat{r}_x(l)$ with the correlation window $w_a(l)$. This approach to power spectrum estimation, which was introduced by Blackman and Tukey (1959), involves the following steps:

[†]The term *spectral window* is quite often used for $W_a(e^{j\omega}) = \mathcal{F}\{w_a(l)\}$, the Fourier transform of the correlation window. However, this term is misleading because $W_a(e^{j\omega})$ is essentially a frequency-domain impulse response. We use the term *correlation window* for $w_a(l)$ and the term *Fourier transform of the correlation window* for $W_a(e^{j\omega})$.

1. Estimate the autocorrelation sequence from the unwindowed data.
2. Window the obtained autocorrelation samples.
3. Compute the DTFT of the windowed autocorrelation as given in (5.3.34).

A pictorial comparison between the theoretical [i.e., using (5.3.32)] and the above practical computation of power spectrum using the single-periodogram smoothing is shown in Figure 5.17.

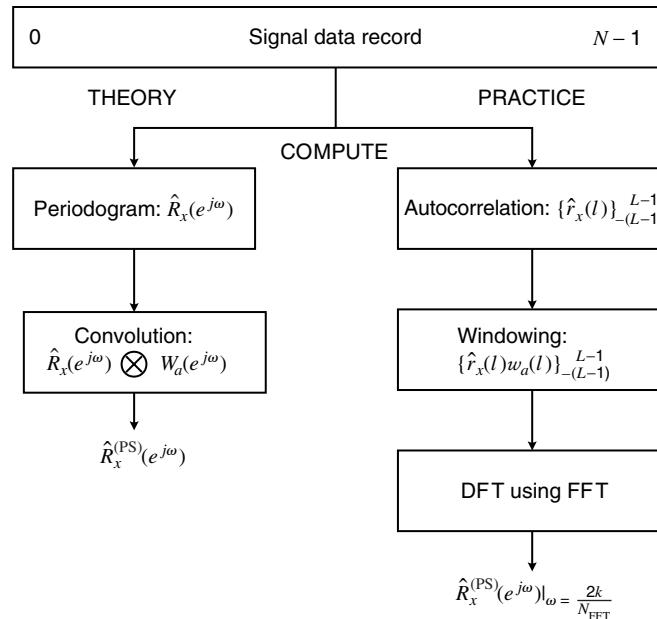


FIGURE 5.17

Comparison of the theory and practice of the Blackman-Tukey method.

The resolution of the Blackman-Tukey power spectrum estimator is determined by the duration $2L - 1$ of the correlation window. For most correlation windows, the resolution is measured by the 3-dB bandwidth of the mainlobe, which is on the order of $2\pi/L$ rad per sampling interval.

The statistical quality of the Blackman-Tukey estimate $\hat{R}_x^{(PS)}(e^{j\omega})$ can be evaluated by examining its mean, covariance, and variance.

Mean of $\hat{R}_x^{(PS)}(e^{j\omega})$. The expected value of the smoothed periodogram $\hat{R}_x^{(PS)}(e^{j\omega})$ can be obtained by using (5.3.34) and (5.2.11). Indeed, we have

$$\begin{aligned}
 E\{\hat{R}_x^{(PS)}(e^{j\omega})\} &= \sum_{l=-(L-1)}^{L-1} E\{\hat{r}_x(l)\} w_a(l) e^{-j\omega l} \\
 &= \sum_{l=-(L-1)}^{L-1} r_x(l) \left(1 - \frac{|l|}{N}\right) w_a(l) e^{-j\omega l}
 \end{aligned} \tag{5.3.36}$$

or, using the frequency convolution theorem, we have

$$E\{\hat{R}_x^{(PS)}(e^{j\omega})\} = R_x(e^{j\omega}) \otimes W_B(e^{j\omega}) \otimes W_a(e^{j\omega}) \tag{5.3.37}$$

where $W_B(e^{j\omega}) = \mathcal{F} \left\{ \left(1 - \frac{|l|}{N} \right) w_R(n) \right\} = \frac{1}{N} \left[\frac{\sin(\omega N/2)}{\sin(\omega/2)} \right]^2$ (5.3.38)

is the Fourier transform of the Bartlett window. Since $E\{\hat{R}_x^{(PS)}(e^{j\omega})\} \neq R_x(e^{j\omega})$, $\hat{R}_x^{(PS)}(e^{j\omega})$ is a *biased* estimate of $R_x(e^{j\omega})$.

For $L \ll N$, $(1 - |l|/N) \simeq 1$ and hence we obtain

$$\begin{aligned} E\{\hat{R}_x^{(PS)}(e^{j\omega})\} &= \sum_{l=-(L-1)}^{L-1} r_x(l) \left(1 - \frac{|l|}{N} \right) w_a(l) e^{-j\omega l} \\ &\simeq R_x(e^{j\omega}) \otimes W_a(e^{j\omega}) \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} R_x(e^{j\theta}) W_a(e^{j(\omega-\theta)}) d\theta \end{aligned} \quad (5.3.39)$$

If L is sufficiently large, the correlation window $w_a(l)$ consists of a narrow mainlobe. If $R_x(e^{j\omega})$ can be assumed to be constant within the mainlobe, we have

$$E\{\hat{R}_x^{(PS)}(e^{j\omega})\} \simeq R_x(e^{j\omega}) \frac{1}{2\pi} \int_{-\pi}^{\pi} W_a(e^{j(\omega-\theta)}) d\theta$$

which implies that $\hat{R}_x^{(PS)}(e^{j\omega})$ is *asymptotically unbiased* if

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} W_a(e^{j\omega}) d\omega = w_a(0) = 1 \quad (5.3.40)$$

that is, if the spectrum of the correlation window has unit area. Under this condition, if both L and N tend to infinity, then $W_a(e^{j\omega})$ and $W_B(e^{j\omega})$ become periodic impulse trains and the convolution (5.3.37) reproduces $R_x(e^{j\omega})$.

Covariance of $\hat{R}_x^{(PS)}(e^{j\omega})$. The following approximation

$$\text{cov}\{\hat{R}_x^{(PS)}(e^{j\omega_1}), \hat{R}_x^{(PS)}(e^{j\omega_2})\} \simeq \frac{1}{2\pi N} \int_{-\pi}^{\pi} R_x^2(e^{j\theta}) W_a(e^{j(\omega_1-\theta)}) W_a(e^{j(\omega_2-\theta)}) d\theta \quad (5.3.41)$$

derived in Jenkins and Watts (1968), holds under the assumptions that (1) N is sufficiently large that $W_B(e^{j\omega})$ behaves as a periodic impulse train and (2) L is sufficiently large that $W_a(e^{j\omega})$ is sufficiently narrow that the product $W_a(e^{j(\omega_1+\theta)}) W_a(e^{j(\omega_2-\theta)})$ is negligible. Hence, the covariance increases proportionally to the width of $W_a(e^{j\omega})$, and the amount of overlap between the windows $W_a(e^{j(\omega_1-\theta)})$ (centered at ω_1) and $W_a(e^{j(\omega_2-\theta)})$ (centered at ω_2) increases.

Variance of $\hat{R}_x^{(PS)}(e^{j\omega})$. When $\omega = \omega_1 = \omega_2$, (5.3.41) gives

$$\text{var}\{\hat{R}_x^{(PS)}(e^{j\omega})\} \simeq \frac{1}{2\pi N} \int_{-\pi}^{\pi} R_x^2(e^{j\omega}) W_a^2(e^{j(\omega-\theta)}) d\theta \quad (5.3.42)$$

If $R_x(e^{j\omega})$ is smooth within the width of $W_a(e^{j\omega})$, then

$$\text{var}\{\hat{R}_x^{(PS)}(e^{j\omega})\} \simeq R_x^2(e^{j\omega}) \frac{1}{2\pi N} \int_{-\pi}^{\pi} W_a^2(e^{j\omega}) d\omega \quad (5.3.43)$$

or $\text{var}\{\hat{R}_x^{(PS)}(e^{j\omega})\} \simeq \frac{E_w}{N} R_x^2(e^{j\omega}) \quad 0 < \omega < \pi$ (5.3.44)

where $E_w = \frac{1}{2\pi} \int_{-\pi}^{\pi} W_a^2(e^{j\omega}) d\omega = \sum_{l=-(L-1)}^{L-1} w_a^2(l)$ (5.3.45)

is the energy of the correlation window. From (5.3.29) and (5.3.44) we have

$$\frac{\text{var}\{\hat{R}_x^{(\text{PS})}(e^{j\omega})\}}{\text{var}\{\hat{R}_x(e^{j\omega})\}} \simeq \frac{E_w}{N} \quad 0 < \omega < \pi \quad (5.3.46)$$

which is known as the *variance reduction factor* or *variance ratio* and provides the reduction in variance attained by smoothing the periodogram.

In the beginning of this section, we explained the variance reduction in terms of frequency-domain averaging. An alternative explanation can be provided by considering the windowing of the estimated autocorrelation. As discussed in Section 5.2, the variance of the autocorrelation estimate increases as $|l|$ approaches N because fewer and fewer samples are used to compute the estimate. Since every value of $\hat{r}_x(l)$ affects the value of $\hat{R}_x(\omega)$ at all frequencies, the less reliable values affect the quality of the periodogram everywhere. Thus, we can reduce the variance of the periodogram by minimizing the contribution of autocorrelation terms with large variance, that is, with lags close to N , by proper windowing.

As we have already stressed, there is a tradeoff between resolution and variance. For the variance to be small, we must choose a window that contains a small amount of energy E_w . Since $|w_a(l)| \leq 1$, we have $E_w \leq 2L$. Thus, to reduce the variance, we must have $L \ll N$. The bias of $\hat{R}_x^{(\text{PS})}(e^{j\omega})$ is directly related to the resolution, which is determined by the mainlobe width of the window, which in turn is proportional to $1/L$. Hence, to reduce the bias, $W_a(e^{j\omega})$ should have a narrow mainlobe that demands a large L . The requirements for high resolution (small bias) and low variance can be simultaneously satisfied only if N is sufficiently large. The variance reduction for some commonly used windows is examined in Problem 5.12. Empirical evidence suggests that use of the Parzen window is a reasonable choice.

Confidence intervals. In the interpretation of spectral estimates, it is important to know whether the spectral details are real or are due to statistical fluctuations. Such information is provided by the confidence intervals (Chapter 3). When the spectrum is plotted on a logarithmic scale, the $(1 - \alpha) \times 100$ percent confidence interval is constant at every frequency, and it is given by (Koopmans 1974)

$$\left(10 \log \hat{R}_x^{(\text{PS})}(e^{j\omega}) - 10 \log \frac{\chi_v^2(1 - \alpha/2)}{v}, 10 \log \hat{R}_x^{(\text{PS})}(e^{j\omega}) + 10 \log \frac{v}{\chi_v^2(\alpha/2)} \right) \quad (5.3.47)$$

where

$$v = \frac{2N}{\sum_{l=-(L-1)}^L w_a^2(l)} \quad (5.3.48)$$

is the degrees of freedom of a χ_v^2 distribution.

Computation of $\hat{R}_x^{(\text{PS})}(e^{j\omega})$ using the DFT. In practice, the Blackman-Tukey power spectrum estimator is computed by using an N -point DFT as follows:

1. Estimate the autocorrelation $r_x(l)$, using the formula

$$\hat{r}_x(l) = \hat{r}_x^*(-l) = \frac{1}{N} \sum_{n=0}^{N+l-1} x(n+l)x^*(n) \quad l = 0, 1, \dots, L-1 \quad (5.3.49)$$

For $L > 100$, indirect computation of $\hat{r}_x(l)$ by using DFT techniques is usually more efficient (see Problem 5.13).

$$f(l) = \begin{cases} \hat{r}_x(l)w_a(l) & 0 \leq l \leq L-1 \\ 0 & L \leq l \leq N-L \\ \hat{r}_x^*(N-l)w_a(N-l) & N-L+1 \leq l \leq N-1 \end{cases} \quad (5.3.50)$$

3. Compute the power spectrum estimate

$$\hat{R}_x^{(\text{PS})}(e^{j\omega})|_{\omega=(2\pi/N)k} = F(k) = \text{DFT}\{f(l)\} \quad 0 \leq k \leq N-1 \quad (5.3.51)$$

as the N -point DFT of the sequence $f(l)$.

MATLAB does not provide a direct function to implement the Blackman-Tukey method. However, such a function can be easily constructed by using built-in MATLAB functions and the above approach. The book toolbox function

```
Rx = bt_psd(x,Nfft>window,L);
```

implements the above algorithm in which $window$ is any available MATLAB window and $Nfft$ is chosen to be larger than N to obtain a high-density spectrum.

EXAMPLE 5.3.5 (BLACKMAN-TUKEY METHOD). Consider the spectrum estimation of three sinusoids in white noise given in Example 5.3.4, that is,

$$x(n) = \cos(0.35\pi n + \phi_1) + \cos(0.4\pi n + \phi_2) + 0.25 \cos(0.8\pi n + \phi_3) + v(n) \quad (5.3.52)$$

where ϕ_1 , ϕ_2 , and ϕ_3 are jointly independent random variables uniformly distributed over $[-\pi, \pi]$ and $v(n)$ is a unit-variance white noise. An ensemble of 50 realizations of $x(n)$ was generated using $N = 512$. The autocorrelations of these realizations were estimated up to lag $L = 64$, 128, and 256. These autocorrelations were windowed using the Bartlett window, and then their 1024-point DFT was computed as the spectrum estimate. The results are shown in Figure 5.18. The top row of the figure contains estimate overlays and the corresponding ensemble average for $L = 64$, the middle row for $L = 128$, and the bottom row for $L = 256$. Several observations can be made from these plots. First, the variance in the estimate has considerably reduced over the periodogram estimate. Second, the lower the lag distance L , the lower the variance and the resolution (i.e., the higher the smoothing of the peaks). This observation is consistent with our discussion above about the effect of L on the quality of estimates. Finally, all the frequencies including the one at 0.8π are clearly distinguishable, something that the basic periodogram could not achieve.

5.3.3 Power Spectrum Estimation by Averaging Multiple Periodograms—The Welch-Bartlett Method

As mentioned in Section 5.3.1, in general, the variance of the sum of K IID random variables is $1/K$ times the variance of each of the random variables. Thus, to reduce the variance of the periodogram, we could average the periodograms from K different realizations of a stationary random signal. However, in most practical applications, only a single realization is available. In this case, we can subdivide the existing record $\{x(n), 0 \leq n \leq N-1\}$ into K (possibly overlapping) smaller segments as follows:

$$x_i(n) = x(iD + n)w(n) \quad 0 \leq n \leq L-1, 0 \leq i \leq K-1 \quad (5.3.53)$$

where $w(n)$ is a window of duration L and D is an *offset* distance. If $D < L$, the segments overlap; and for $D = L$, the segments are contiguous. The periodogram of the i th segment is

$$\hat{R}_{x,i}(e^{j\omega}) \triangleq \frac{1}{L} |X_i(e^{j\omega})|^2 = \frac{1}{L} \left| \sum_{n=0}^{L-1} x_i(n)e^{-j\omega n} \right|^2 \quad (5.3.54)$$

We remind the reader that the window $w(n)$ in (5.3.53) is called a data window because it is applied directly to the data, in contrast to a correlation window that is applied to the autocorrelation sequence [see (5.3.34)]. Notice that there is no need for the data window to have an even shape or for its Fourier transform to be nonnegative. The purpose of using the data window is to control spectral leakage.

The spectrum estimate $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ is obtained by averaging K periodograms as follows:

$$\hat{R}_x^{(\text{PA})}(e^{j\omega}) \triangleq \frac{1}{K} \sum_{i=0}^{K-1} \hat{R}_{x,i}(e^{j\omega}) = \frac{1}{KL} \sum_{i=0}^{K-1} |X_i(e^{j\omega})|^2 \quad (5.3.55)$$

where the superscript (PA) denotes periodogram averaging. To determine the bias and variance of $\hat{R}_x^{(\text{PA})}(e^{j\omega})$, we let $D = L$ so that the segments do not overlap. The so-computed estimate $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ is known as the *Bartlett* estimate. We also assume that $r_x(l)$ is very small for $|l| > L$. This implies that the signal segments can be assumed to be approximately uncorrelated. To show that the simple periodogram averaging in Bartlett's method reduces the periodogram variance, we consider the following example.

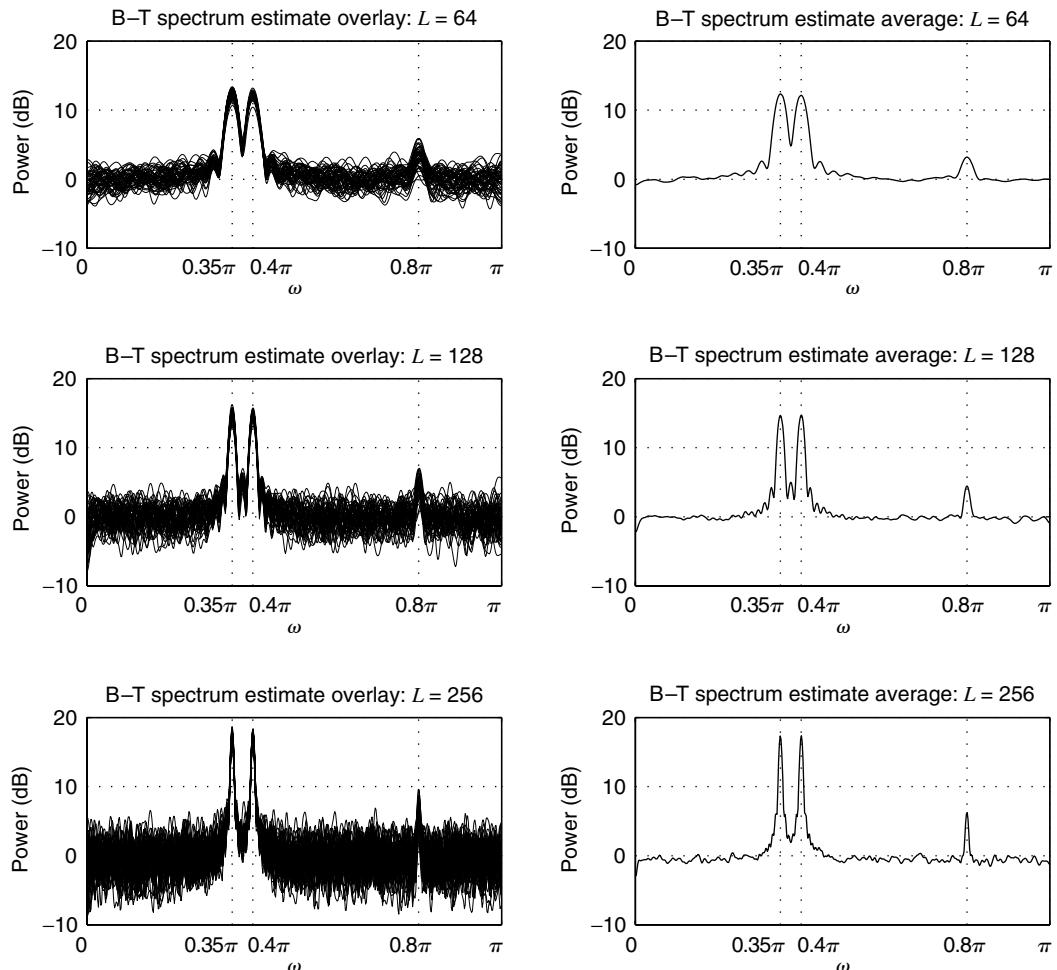


FIGURE 5.18

Spectrum estimation of three sinusoids in white noise using the Blackman-Tukey method in Example 5.3.5.

EXAMPLE 5.3.6 (PERIODOGRAM AVERAGING). Let $x(n)$ be a stationary white Gaussian noise with zero mean and unit variance. The theoretical spectrum of $x(n)$ is

$$R_x(e^{j\omega}) = \sigma_x^2 = 1 \quad -\pi < \omega \leq \pi$$

An ensemble of 50 different 512-point records of $x(n)$ was generated using a pseudorandom number generator. The Bartlett estimate of each record was computed for $K = 1$ (i.e., the basic periodogram), $K = 4$ (or $L = 128$), and $K = 8$ (or $L = 64$). The results in the form of estimate overlays and averages are shown in Figure 5.19. The effect of periodogram averaging is clearly evident.

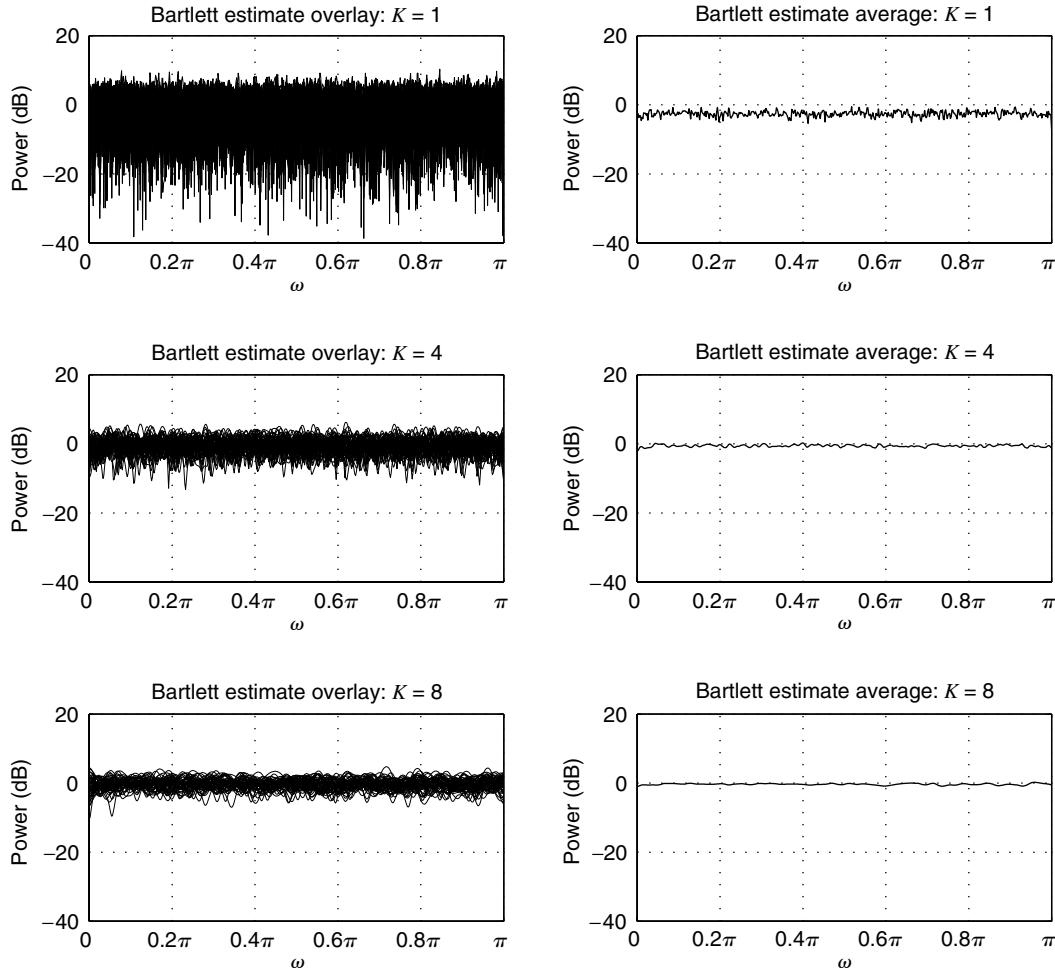


FIGURE 5.19

Spectral estimation of white noise using Bartlett's method in Example 5.3.6.

Mean of $\hat{R}_x^{(\text{PA})}(e^{j\omega})$. The mean value of $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ is

$$E\{\hat{R}_x^{(\text{PA})}(e^{j\omega})\} = \frac{1}{K} \sum_{i=0}^{K-1} E\{\hat{R}_{x,i}(e^{j\omega})\} = E\{\hat{R}_x(e^{j\omega})\} \quad (5.3.56)$$

where we have assumed that $E\{\hat{R}_{x,i}(e^{j\omega})\} = E\{\hat{R}_x(e^{j\omega})\}$ because of the stationarity assumption. From (5.3.56) and (5.3.15), we have

$$E\{\hat{R}_x^{(\text{PA})}(e^{j\omega})\} = E\{\hat{R}_x(e^{j\omega})\} = \frac{1}{2\pi L} \int_{-\pi}^{\pi} R_x(e^{j\theta}) R_w(e^{j(\omega-\theta)}) d\theta \quad (5.3.57)$$

where $R_w(e^{j\omega})$ is the spectrum of the data window $w(n)$. Hence, $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ is a biased estimate of $R_x(e^{j\omega})$. However, if the data window is normalized such that

$$\sum_{n=0}^{L-1} w^2(n) = L \quad (5.3.58)$$

the estimate $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ becomes asymptotically unbiased [see the discussion following equation (5.3.15)].

Variance of $\hat{R}_x^{(\text{PA})}(e^{j\omega})$. The variance of $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ is

$$\text{var}\{\hat{R}_x^{(\text{PA})}(e^{j\omega})\} = \frac{1}{K} \text{var}\{\hat{R}_x(e^{j\omega})\} \quad (5.3.59)$$

(assuming segments are independent) or using (5.3.29) gives

$$\text{var}\{\hat{R}_x^{(\text{PA})}(e^{j\omega})\} \simeq \frac{1}{K} R_x^2(e^{j\omega}) \quad (5.3.60)$$

Clearly, as K increases, the variance tends to zero. Thus, $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ provides an asymptotically unbiased and consistent estimate of $R_x(e^{j\omega})$. If N is fixed and $N = KL$, we see that increasing K to reduce the variance (or equivalently obtain a smoother estimate) results in a decrease in L , that is, a reduction in resolution (or equivalently an increase in bias).

When $w(n)$ in (5.3.53) is the rectangular window of duration L , the square of its Fourier transform is equal to the Fourier transform of the triangular sequence $w_T(n) \triangleq L - |l|$, $|l| < L$, which when combined with the $1/L$ factor in (5.3.57), results in the Bartlett window

$$w_B(l) = \begin{cases} 1 - \frac{|l|}{L} & |l| < L \\ 0 & \text{elsewhere} \end{cases} \quad (5.3.61)$$

$$\text{with } W_B(e^{j\omega}) = \frac{1}{L} \left[\frac{\sin(\omega L/2)}{\sin(\omega/2)} \right]^2 \quad (5.3.62)$$

This special case of averaging multiple nonoverlapping periodograms was introduced by Bartlett (1953).

The method has been extended to modified overlapping periodograms by Welch (1970), who has shown that the shape of the window does not affect the variance formula (5.3.59). Welch showed that overlapping the segments by 50 percent reduces the variance by about a factor of 2, owing to doubling the number of segments. More overlap does not result in additional reduction of variance because the data segments become less and less independent. Clearly, the nonoverlapping segments can be uncorrelated only for white noise signals. However, the data segments can be considered approximately uncorrelated if they do not have sharp spectral peaks or if their autocorrelations decay fast.

Thus, the variance reduction factor for the spectral estimator $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ is

$$\frac{\text{var}\{\hat{R}_x^{(\text{PA})}(e^{j\omega})\}}{\text{var}\{\hat{R}_x(e^{j\omega})\}} \simeq \frac{1}{K} \quad 0 < \omega < \pi \quad (5.3.63)$$

and is reduced by a factor of 2 for 50 percent overlap.

Confidence intervals. The $(1 - \alpha) \times 100$ percent confidence interval on a logarithmic scale may be shown to be (Jenkins and Watts 1968)

$$\left(10 \log \hat{R}_x^{(\text{PA})}(e^{j\omega}) - 10 \log \frac{\chi_{2K}^2(1 - \alpha/2)}{2K}, 10 \log \hat{R}_x^{(\text{PA})}(e^{j\omega}) + 10 \log \frac{2K}{\chi_{2K}^2(\alpha/2)} \right) \quad (5.3.64)$$

where χ_{2K}^2 is a chi-squared distribution with $2K$ degrees of freedom.

Computation of $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ using the DFT. In practice, to compute $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ at L equally spaced frequencies $\omega_k = 2\pi k/L$, $0 \leq k \leq L - 1$, the method of periodogram averaging can be easily and efficiently implemented by using the DFT as follows (we have assumed that L is even):

1. Segment data $\{x(n)\}_0^{N-1}$ into K segments of length L , each offset by D duration using

$$\bar{x}_i(n) = x(iD + n) \quad 0 \leq i \leq K - 1, 0 \leq n \leq L - 1 \quad (5.3.65)$$

If $D = L$, there is no overlap; and if $D = L/2$, the overlap is 50 percent.

2. Window each segment, using data window $w(n)$

$$x_i(n) = \bar{x}_i(n)w(n) = x(iD + n)w(n) \quad 0 \leq i \leq K - 1, 0 \leq n \leq L - 1 \quad (5.3.66)$$

3. Compute the N -point DFTs $X_i(k)$ of the segments $x_i(n)$, $0 \leq i \leq K - 1$,

$$\tilde{X}_i(k) = \sum_{n=0}^{L-1} x_i(n)e^{-j(2\pi/L)kn} \quad 0 \leq k \leq L - 1, 0 \leq i \leq K - 1 \quad (5.3.67)$$

4. Accumulate the squares $|\tilde{X}_i(k)|^2$

$$\tilde{S}_i(k) \triangleq \sum_{i=0}^{K-1} |\tilde{X}_i(k)|^2 \quad 0 \leq k \leq L/2 \quad (5.3.68)$$

5. Finally, normalize by KL to obtain the estimate $\hat{R}_x^{(\text{PA})}(k)$:

$$\hat{R}_x^{(\text{PA})}(k) = \frac{1}{KL} \sum_{i=0}^{K-1} \tilde{S}_i(k) \quad 0 \leq k \leq N/2 \quad (5.3.69)$$

At this point we emphasize that the spectrum estimate $\hat{R}_x^{(\text{PA})}(k)$ is always nonnegative. A pictorial description of this computational algorithm is shown in Figure 5.20. A more efficient way to compute $\hat{R}_x^{(\text{PA})}(k)$ is examined in Problem 5.14.

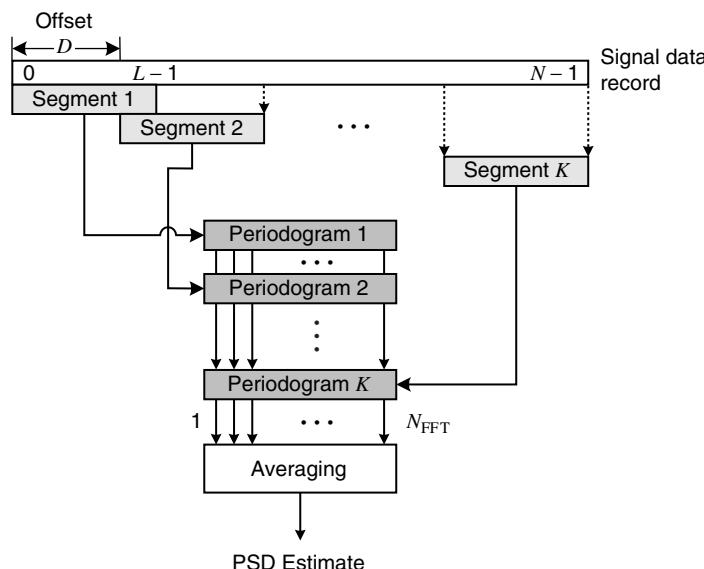


FIGURE 5.20

Pictorial description of the Welch-Bartlett method.

In MATLAB the Welch-Bartlett method is implemented by using the function

```
Rx = psd(x,Nfft,Fs,window(L),Noverlap,'none');
```

where *window* is the name of any MATLAB-provided window function (e.g., *hamming*); *Nfft* is the size of the DFT, which is chosen to be larger than *L* to obtain a high-density spectrum; *Fs* is the sampling frequency, which is used for plotting purposes; and *Noverlap* specifies the number of overlapping samples. If the *boxcar* window is used along with *Noverlap*=0, then we obtain Bartlett's method of periodogram averaging. (Note that *Noverlap* is different from the offset parameter *D* given above.) If *Noverlap*=*L*/2 is used, then we obtain Welch's averaged periodogram method with 50 percent overlap.

A biased estimate $\hat{r}_x(l)$, $|l| < L$, of the autocorrelation sequence of $x(n)$ can be obtained by taking the inverse *N*-point DFT of $\hat{R}_x^{(PA)}(k)$ if $N \geq 2L - 1$. Since only samples of the continuous spectrum $\hat{R}_x^{(PA)}(e^{j\omega})$ are available, the obtained autocorrelation sequence $\hat{r}_x^{(PA)}(l)$ is an aliased version of the true autocorrelation $r_x(l)$ of the signal $x(n)$ (see Problem 5.15).

EXAMPLE 5.3.7 (BARTLETT'S METHOD). Consider again the spectrum estimation of three sinusoids in white noise given in Example 5.3.4, that is,

$$x(n) = \cos(0.35\pi n + \phi_1) + \cos(0.4\pi n + \phi_2) + 0.25 \cos(0.8\pi n + \phi_3) + v(n) \quad (5.3.70)$$

where ϕ_1 , ϕ_2 , and ϕ_3 are jointly independent random variables uniformly distributed over $[-\pi, \pi]$ and $v(n)$ is a unit-variance white noise. An ensemble of 50 realizations of $x(n)$ was generated using $N = 512$. The Bartlett estimate of each ensemble was computed for $K = 1$ (i.e., the basic periodogram), $K = 4$ (or $L = 128$), and $K = 8$ (or $L = 64$). The results in the form of estimate overlays and averages are shown in Figure 5.21. Observe that the variance in the estimate has consistently reduced over the periodogram estimate as the number of averaging segments has increased. However, this reduction has come at the price of broadening of the spectral peaks. Since no window is used, the sidelobes are very prominent even for the $K = 8$ segment. Thus confidence in the $\omega = 0.8\pi$ spectral line is not very high for the $K = 8$ case.

EXAMPLE 5.3.8 (WELCH'S METHOD). Consider Welch's method for the random process in the above example for $N = 512$, 50 percent overlap, and a Hamming window. Three different values for L were considered: $L = 256$ (3 segments), $L = 128$ (7 segments), and $L = 64$ (15 segments). The estimate overlays and averages are shown in Figure 5.22. In comparing these results with those in Figure 5.21, note that the windowing has considerably reduced the spurious peaks in the spectra but has also further smoothed the peaks. Thus the peak at 0.8π is recognizable with high confidence, but the separation of two close peaks is not so clear for $L = 64$. However, the $L = 128$ case provides the best balance between separation and detection. On comparing the Blackman-Tukey (Figure 5.18) and Welch estimates, we observe that the results are comparable in terms of variance reduction and smoothing aspects.

5.3.4 Some Practical Considerations and Examples

The periodogram and its modified version, which is the basic tool involved in the estimation of the power spectrum of stationary signals, can be computed either *directly* from the signal samples $\{x(n)\}_0^{N-1}$ using the DTFT formula

$$\hat{R}_x(e^{j\omega}) = \frac{1}{N} \left| \sum_{n=0}^{N-1} w(n)x(n)e^{-j\omega n} \right|^2 \quad (5.3.71)$$

or *indirectly* using the autocorrelation sequence

$$\hat{R}_x(e^{j\omega}) = \sum_{l=-(N-1)}^{N-1} \hat{r}_x(l)e^{-j\omega l} \quad (5.3.72)$$

where $\hat{r}_x(l)$ is the estimated autocorrelation of the windowed segment $\{w(n)x(n)\}_0^{N-1}$. The periodogram $\hat{R}_x(e^{j\omega})$ provides an unacceptable estimate of the power spectrum because

1. it has a bias that depends on the length N and the shape of the data window $w(n)$ and
2. its variance is equal to the true spectrum $R_x(e^{j\omega})$.

Given a data segment of fixed duration N , there is no way to reduce the bias, or equivalently to increase the resolution, because it depends on the length and the shape of the window. However, we can reduce the variance either by averaging the single periodogram of the data (method of Blackman-Tukey) or by averaging multiple periodograms obtained by partitioning the available record into smaller overlapping segments (method of Bartlett-Welch).

The method of Blackman-Tukey is based on the following modification of the indirect periodogram formula

$$\hat{R}_x^{(\text{PS})}(e^{j\omega}) = \sum_{l=-(L-1)}^{L-1} \hat{r}_x(l) w_a(l) e^{-j\omega l} \quad (5.3.73)$$

which basically involves windowing of the estimated autocorrelation (5.2.1) with a proper

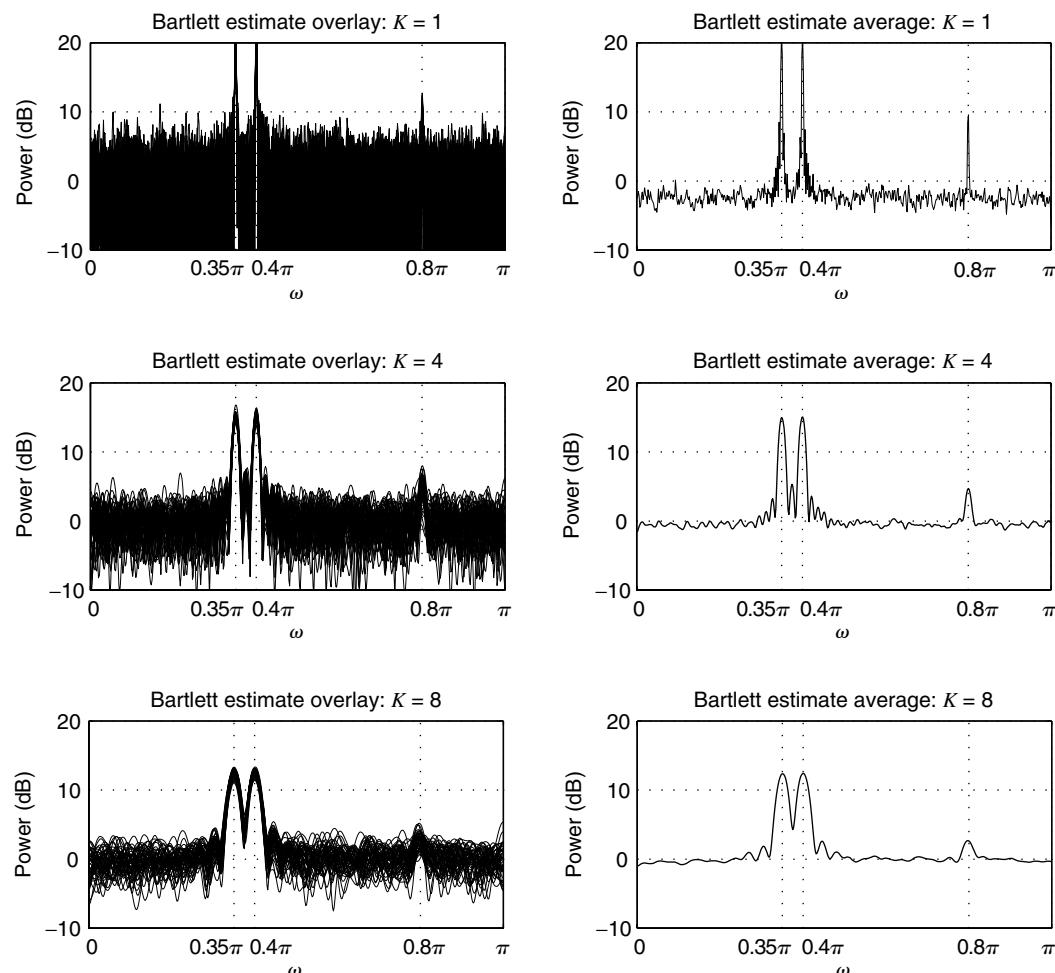


FIGURE 5.21

Estimation of three sinusoids in white noise using Bartlett's method in Example 5.3.7.

correlation window. Using only the first $L \ll N$ more-reliable values of the autocorrelation sequence reduces the variance of the spectrum estimate by a factor of approximately L/N . However, at the same time, this reduces the resolution from about $1/N$ to about $1/L$. The recommended range for L is between $0.1N$ and $0.2N$.

The method of Bartlett-Welch is based on partitioning the available data record into windowed overlapping segments of length L , computing their periodograms by using the direct formula (5.3.71), and then averaging the resulting periodograms to compute the estimate

$$\hat{R}_x^{(\text{PA})}(e^{j\omega}) = \frac{1}{KL} \left| \sum_{n=0}^{L-1} x_i(n) e^{-j\omega n} \right|^2 \quad (5.3.74)$$

whose resolution is reduced to approximately $1/L$ and whose variance is reduced by a factor of about $1/K$, where K is the number of segments.

The reduction in resolution and variance of the Blackman-Tukey estimate is achieved by “averaging” the values of the spectrum at consecutive frequency bins by windowing the estimated autocorrelation sequence. In the Bartlett-Welch method, the same effect is achieved by averaging the values of multiple shorter periodograms at the same frequency

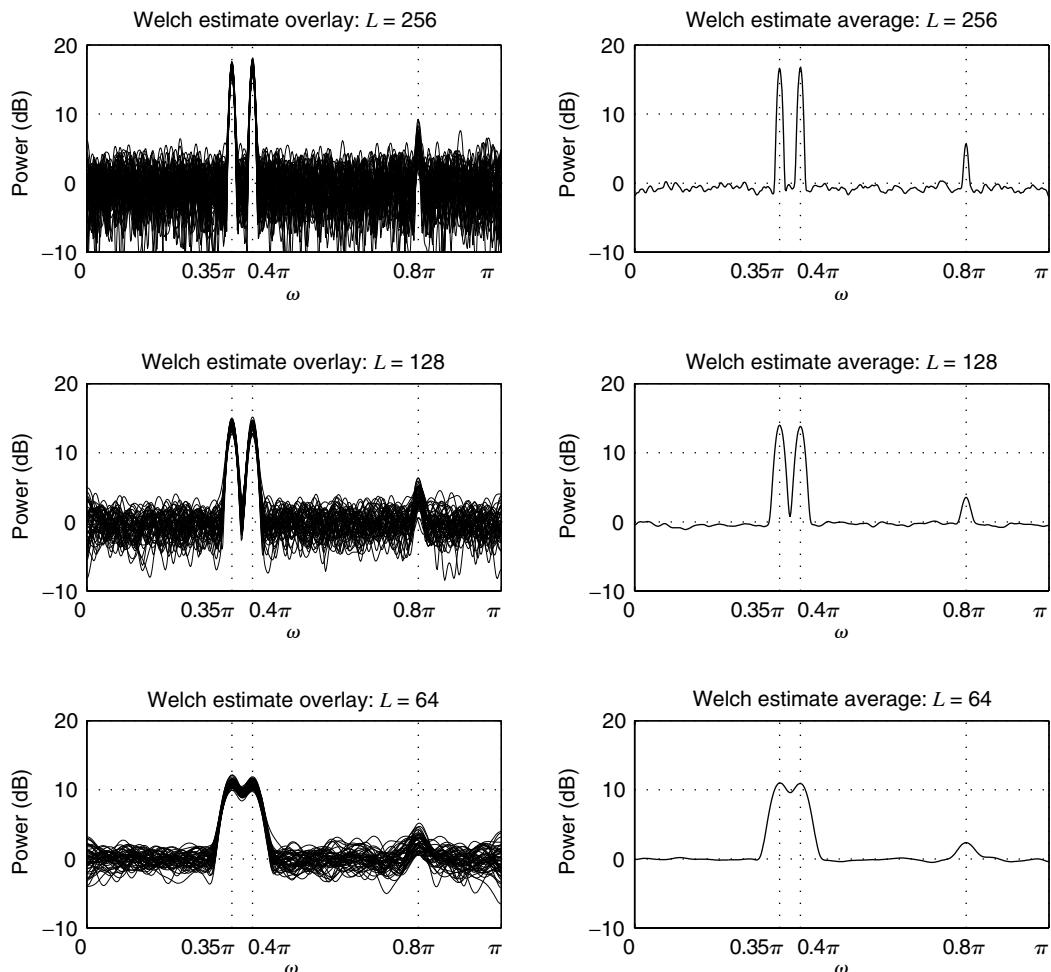


FIGURE 5.22

Estimation of three sinusoids in white noise using Welch's method in Example 5.3.8.

bin. The PSD estimation methods and their properties are summarized in Table 5.3. The multitaper spectrum estimation method given in the last column of Table 5.3 is discussed in Section 5.5.

TABLE 5.3
Comparison of PSD estimation methods.

	Periodogram $\hat{R}_x(e^{j\omega})$	Single-periodogram smoothing (Blackman-Tukey): $\hat{R}_x^{(PS)}(e^{j\omega})$	Multiple-periodogram averaging (Bartlett-Welch): $\hat{R}_x^{(PA)}(e^{j\omega})$	Multitaper (Thomson): $\hat{R}_x^{(MT)}(e^{j\omega})$
Description of the method	Compute DFT of data record	Compute DFT of windowed autocorrelation estimate (see Figure 5.17)	Split record into K segments and average their modified periodograms (see Figure 5.20)	Window data record using K orthonormal tapers and average their periodograms (see Figure 5.30)
Basic idea	Natural estimator of $R_x(e^{j\omega})$; the error $ r_x(l) - \hat{r}_x(l) $ is large for large $ l $	Local smoothing of $\hat{R}_x(e^{j\omega})$ by weighting $\hat{r}_x(l)$ with a lag window $w_a(l)$	Overlap data records to create more segments; window segments to reduce bias; average periodograms to reduce variance	For properly designed orthogonal tapers, periodograms are independent at each frequency. Hence averaging reduces variance
Bias	Severe for small N ; negligible for large N	Asymptotically unbiased	Asymptotically unbiased	Negligible for properly designed tapers
Resolution	$\propto \frac{1}{N}$	$\propto \frac{1}{L}$, L is maximum lag	$\propto \frac{1}{L}$ L is segment length	$\propto \frac{1}{N}$
Variance	Unacceptable: about $R_x^2(e^{j\omega})$ for all N	$R_x^2(e^{j\omega}) \times \frac{E_w}{N}$	$\frac{R_x^2(e^{j\omega})}{K}$ K is number of segments	$\frac{R_x^2(e^{j\omega})}{K}$ K is number of tapers

EXAMPLE 5.3.9 (COMPARISON OF BLACKMAN-TUKEY AND WELCH-BARTLETT METHODS).

Figure 5.23 illustrates the properties of the power spectrum estimators based on autocorrelation windowing and periodogram averaging using the AR(4) model (5.3.24). The top plots show the power spectrum of the process. The left column plots show the power spectrum obtained by windowing the data with a Hanning window and the autocorrelation with a Parzen window of length $L = 64, 128$, and 256 . We notice that as the length of the window increases, the resolution decreases and the variance increases. We see a similar behavior with the method of averaged periodograms as the segment length L increases from 64 to 256 . Clearly, both methods give comparable results if their parameters are chosen properly.

Example of ocean wave data. To apply spectrum estimation techniques discussed in this chapter to real data, we will use two real-valued time series that are obtained by recording the height of ocean waves as a function of time, as measured by two wave gages of different designs. These two series are shown in Figure 5.24. The top graph shows the wire wave gage data while the bottom graph shows the infrared wave gage data. The frequency responses of these gages are such that—mainly because of its inertia—frequencies higher than 1 Hz cannot be reliably measured. The frequency range between 0.2 and 1 Hz is also important because the rate at which the spectrum decreases has a physical model associated with it. Both series were collected at a rate of 30 samples per second. There are 4096 samples in each series.[†] We will also use these data to study joint signal analysis in the next section.

[†]These data were collected by A. Jessup, Applied Physics Laboratory, University of Washington. It was obtained from StatLib, a statistical archive maintained by Carnegie Mellon University.

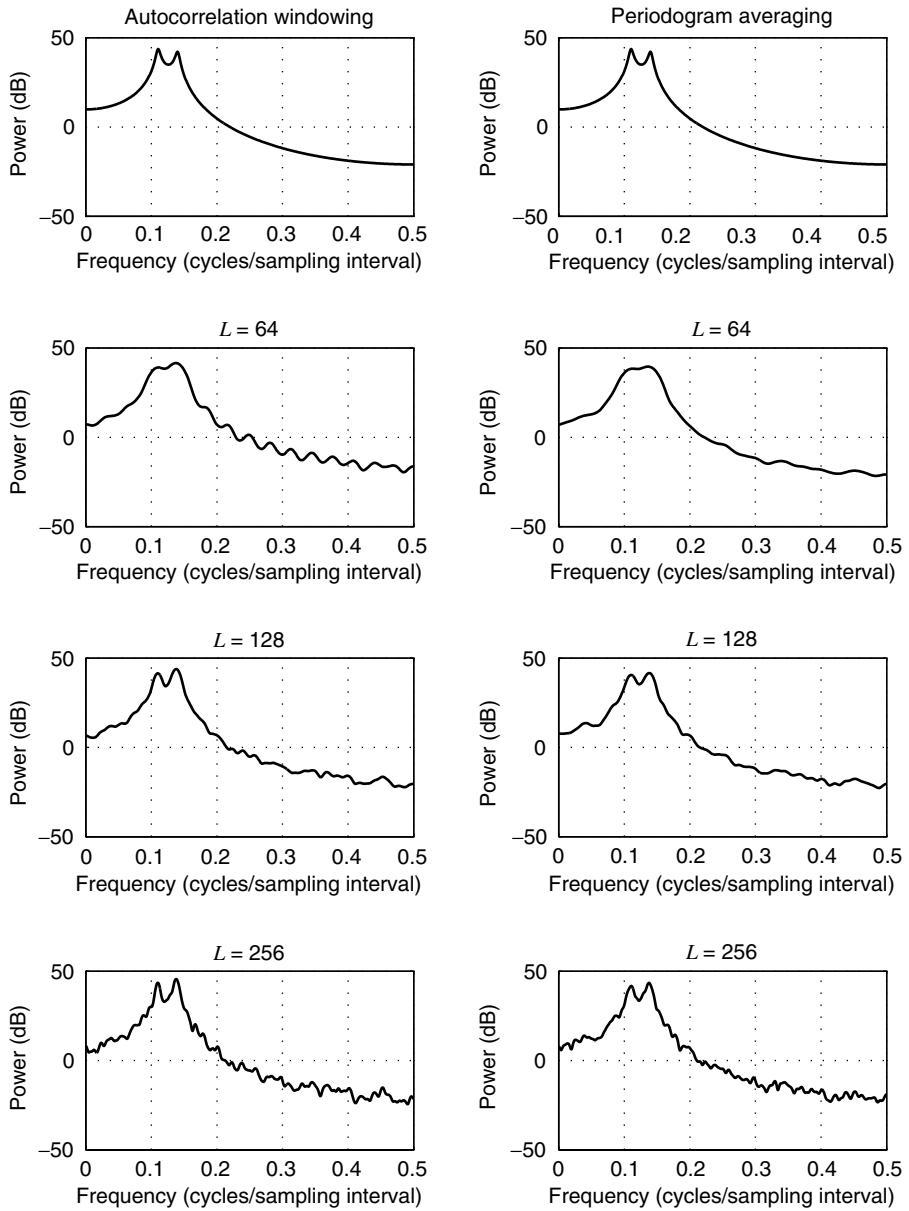
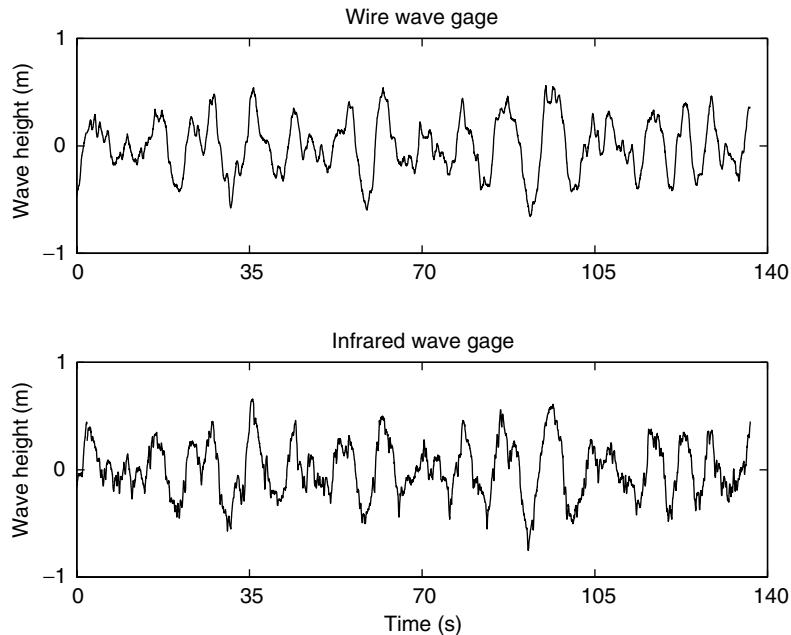
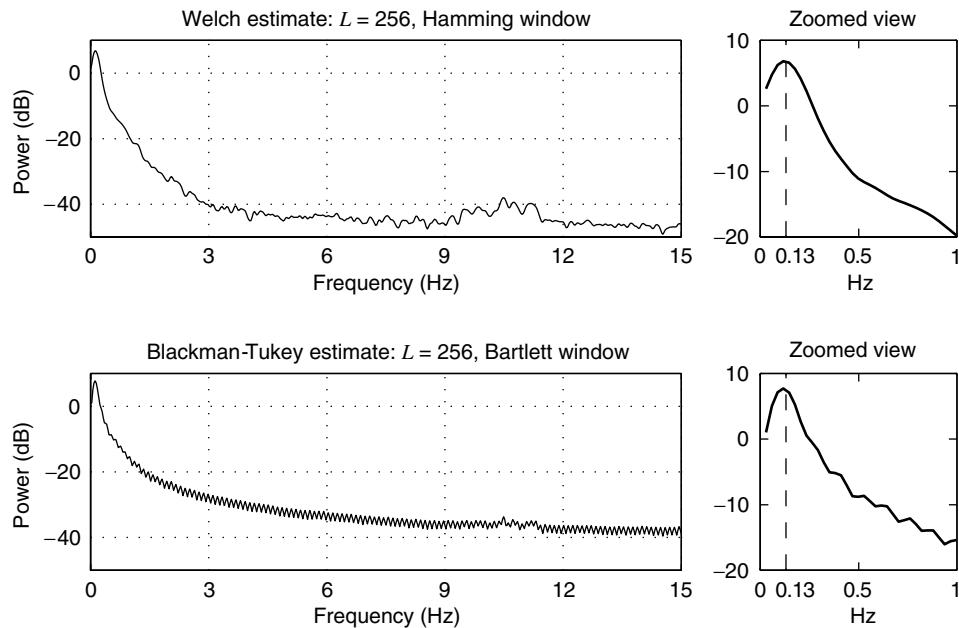
**FIGURE 5.23**

Illustration of the properties of the power spectrum estimators using autocorrelation windowing (left column) and periodogram averaging (right column) in Example 5.3.9.

EXAMPLE 5.3.10 (ANALYSIS OF THE OCEAN WAVE DATA). Figure 5.25 depicts the periodogram averaging and smoothing estimates of the wire wave gage data. The top row of plots shows the Welch estimate using a Hamming window, $L = 256$, and 50 percent overlap between segments. The bottom row shows the Blackman-Tukey estimate using a Bartlett window and a lag length of $L = 256$. In both cases, a zoomed view of the plots between 0 and 1 Hz is shown in the right column to obtain a better view of the spectra. Both spectral estimates provide a similar spectral behavior, especially over the frequency range of 0 to 1 Hz. Furthermore, both show a broad, low-frequency peak at 0.13 Hz, corresponding to a period of about 8 s. The dominant features of the time series thus can be attributed to this peak and other features in the 0- to 0.2-Hz range. The shape of the spectrum between 0.2 and 1 Hz is a decaying exponential and is consistent with the physical model. Similar results were obtained for the infrared wave gauge data.

**FIGURE 5.24**

Display of ocean wave data.

**FIGURE 5.25**

Spectrum estimation of the ocean wave data using the Welch and Blackman-Tukey methods.

5.4 JOINT SIGNAL ANALYSIS

Until now, we discussed estimation techniques for the computation of the power spectrum of one random process $x(n)$, which is also known as *univariate* spectral estimation. In many practical applications, we have two jointly stationary random processes and we wish to study the correlation between them. The analysis and computation of this correlation

and the associated spectral quantities are similar to those of univariate estimation and are called *bivariate* spectrum estimation. In this section, we provide a brief overview of this joint signal analysis.

Let $x(n)$ and $y(n)$ be two zero-mean, jointly stationary random processes with power spectra $R_x(e^{j\omega})$ and $R_y(e^{j\omega})$, respectively. Then from (3.3.61), the cross-power spectral density of $x(n)$ and $y(n)$ is given by

$$R_{xy}(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r_{xy}(l)e^{-j\omega l} \quad (5.4.1)$$

where $r_{xy}(l)$ is the cross-correlation sequence between $x(n)$ and $y(n)$. The cross-spectral density $R_{xy}(e^{j\omega})$ is, in general, a complex-valued function that is difficult to interpret or plot in its complex form. Therefore, we need to express it by using real-valued functions that are easier to deal with. It is customary to express the conjugate of $R_{xy}(e^{j\omega})$ in terms of its real and imaginary components, that is,

$$R_{xy}(e^{j\omega}) = C_{xy}(\omega) - jQ_{xy}(\omega) \quad (5.4.2)$$

where

$$C_{xy}(\omega) \triangleq \operatorname{Re}[R_{xy}(e^{j\omega})] \quad (5.4.3)$$

is called the *cospectrum* and

$$Q_{xy}(\omega) \triangleq \operatorname{Im}[R_{xy}^*(e^{j\omega})] = -\operatorname{Im}[R_{xy}(e^{j\omega})] \quad (5.4.4)$$

is called the *quadrature spectrum*. Alternately, the most popular approach is to express $R_{xy}(e^{j\omega})$ in terms of its magnitude and angle components, that is,

$$R_{xy}(e^{j\omega}) = A_{xy}(\omega) \exp[j\Phi_{xy}(\omega)] \quad (5.4.5)$$

where

$$A_{xy}(\omega) = |R_{xy}(e^{j\omega})| = \sqrt{C_{xy}^2(\omega) + Q_{xy}^2(\omega)} \quad (5.4.6)$$

and

$$\Phi_{xy}(\omega) = \angle R_{xy}(e^{j\omega}) = \tan^{-1}\{-Q_{xy}(\omega)/C_{xy}(\omega)\} \quad (5.4.7)$$

The magnitude $A_{xy}(\omega)$ is called the *cross-amplitude spectrum*, and the angle $\Phi_{xy}(\omega)$ is called the *phase spectrum*. All these derived functions are real-valued and hence can be examined graphically. However, the phase spectrum has the 2π ambiguity in its computation, which makes its interpretation somewhat problematic.

From (3.3.64) the normalized cross-spectrum, called the *complex coherence function*, is given by

$$\mathcal{G}_{xy}(\omega) = \frac{R_{xy}(e^{j\omega})}{\sqrt{R_x(e^{j\omega})R_y(e^{j\omega})}} \quad (5.4.8)$$

which is a complex-valued frequency-domain correlation coefficient that measures the correlation between the random amplitudes of the complex exponentials with frequency ω in the spectral representations of $x(n)$ and $y(n)$. Hence to interpret this coefficient, its magnitude $|\mathcal{G}_{xy}(\omega)|$ is computed, which is referred to as the *coherency spectrum*. Recall that in Chapter 3, we called $|\mathcal{G}_{xy}(\omega)|^2$ the *magnitude-squared coherence (MSC)*. Clearly, $0 \leq |\mathcal{G}_{xy}(\omega)| \leq 1$. Since the coherency spectrum captures the amplitude spectrum but completely ignores the phase spectrum, in practice, the coherency and the phase spectrum are useful real-valued summaries of the cross-spectrum.

5.4.1 Estimation of Cross Power Spectrum

Now we apply the techniques developed in Section 5.3 to the problem of estimating the cross-spectrum and its associated real-valued functions. Let $\{x(n), y(n)\}_0^{N-1}$ be the data

record available for estimation. By using the periodogram (5.3.5) as a guide, the estimator for $R_{xy}(e^{j\omega})$ is the *cross-periodogram* given by

$$\hat{R}_{xy}(e^{j\omega}) \triangleq \sum_{l=-(N-1)}^{N-1} \hat{r}_{xy}(l) e^{-j\omega l} \quad (5.4.9)$$

$$\text{where } \hat{r}_{xy}(l) = \begin{cases} \frac{1}{N} \sum_{n=0}^{N-l-1} x(n+l)y^*(n) & 0 \leq l \leq N-1 \\ \frac{1}{N} \sum_{n=0}^{N+l-1} x(n)y^*(n-l) & -(N-1) \leq l \leq -1 \\ 0 & l \leq -N \text{ or } l \geq N \end{cases} \quad (5.4.10)$$

In analogy to (5.3.2), the cross-periodogram can also be written as

$$\hat{R}_{xy}(e^{j\omega}) = \frac{1}{N} \left[\sum_{n=0}^{N-1} x(n)e^{-j\omega n} \right] \left[\sum_{n=0}^{N-1} y(n)e^{-j\omega n} \right]^* \quad (5.4.11)$$

Once again, it can be shown that the bias and variance properties of the cross-periodogram are as poor as those of the periodogram. Another disturbing result of these periodograms is that from (5.4.11) and (5.3.2), we obtain

$$|\hat{R}_{xy}(e^{j\omega})|^2 = \left(\frac{1}{N} \right)^2 \left| \sum_{n=0}^{N-1} x(n)e^{-j\omega n} \right|^2 \left| \sum_{n=0}^{N-1} y(n)e^{-j\omega n} \right|^2 = \hat{R}_x(e^{j\omega}) \hat{R}_y(e^{j\omega})$$

which implies that if we estimate the MSC from the “raw” autoperiodograms as well as cross-periodograms, then the result is always unity for all frequencies. This seemingly unreasonable result is due to the fact that the frequency-domain correlation coefficient at each frequency ω is estimated by using only one single pair of observations from the two signals. Therefore, a reasonable amount of smoothing in the periodogram is necessary to reduce the inherent variability of the cross-spectrum and to improve the accuracy of the estimated coherency. This variance reduction can be achieved by straightforward extensions of various techniques discussed in Section 5.3 for power spectra. These methods include periodogram smoothing across frequencies and the various modified periodogram averaging techniques. In practice, Welch’s approach to modified periodogram averaging, based on overlapped segments, is preferred owing to its superior performance. For illustration purposes, we describe Welch’s approach in a brief fashion.

In this approach, we subdivide the existing data records $\{x(n), y(n); 0 \leq n \leq N-1\}$ into K overlapping smaller segments of length L as follows:

$$\begin{aligned} x_i(n) &= x(iD + n)w(n) & 0 \leq n \leq L-1, 0 \leq i \leq K-1 \\ y_i(n) &= y(iD + n)w(n) \end{aligned} \quad (5.4.12)$$

where $w(n)$ is a data window of length L and $D = L/2$ for 50 percent overlap. The cross-periodogram of the i th segment is given by

$$\hat{R}_i(e^{j\omega}) = \frac{1}{L} X_i(e^{j\omega}) Y_i^*(e^{j\omega}) = \frac{1}{L} \left[\sum_{n=0}^{L-1} x_i(n)e^{-j\omega n} \right] \left[\sum_{n=0}^{L-1} y_i(n)e^{-j\omega n} \right]^* \quad (5.4.13)$$

Finally, the smoothed cross-spectrum $\hat{R}_{xy}^{(\text{PA})}(e^{j\omega})$ is obtained by averaging K cross-periodograms as follows:

$$\hat{R}_{xy}^{(\text{PA})}(e^{j\omega}) = \frac{1}{K} \sum_{i=0}^{K-1} \hat{R}_i(e^{j\omega}) = \frac{1}{KL} \sum_{i=0}^{K-1} X_i(e^{j\omega}) Y_i^*(e^{j\omega}) \quad (5.4.14)$$

Similar to (5.3.51), the DFT computation of $\hat{R}_{xy}^{(\text{PA})}(e^{j\omega})$ is given by

$$\hat{\tilde{R}}_{xy}^{(\text{PA})}(k) = \frac{1}{KL} \sum_{i=0}^{K-1} \left[\sum_{n=0}^{L-1} x_i(n) e^{-j2\pi kn/N} \right] \left[\sum_{n=0}^{L-1} y_i(n) e^{-j2\pi kn/N} \right]^* \quad (5.4.15)$$

where $0 \leq k \leq N - 1$, $N > L$.

Estimation of cospectra and quadrature spectra. Once the cross-spectrum $R_{xy}(e^{j\omega})$ has been estimated, we can compute the estimates of all the associated real-valued spectra by replacing $R_{xy}(e^{j\omega})$ with its estimate $\hat{R}_{xy}^{(\text{PA})}(e^{j\omega})$ in the definitions of these functions. To estimate the cospectrum, we use

$$\hat{C}_{xy}^{(\text{PA})}(\omega) = \text{Re}[\hat{R}_{xy}^{(\text{PA})}(e^{j\omega})] = \text{Re} \left[\frac{1}{KL} \sum_{i=0}^{K-1} X_i(e^{j\omega}) Y_i^*(e^{j\omega}) \right] \quad (5.4.16)$$

and to estimate the quadrature spectrum, we use

$$\hat{Q}_{xy}^{(\text{PA})}(\omega) = -\text{Im}[\hat{R}_{xy}^{(\text{PA})}(e^{j\omega})] = -\text{Im} \left[\frac{1}{KL} \sum_{i=0}^{K-1} X_i(e^{j\omega}) Y_i^*(e^{j\omega}) \right] \quad (5.4.17)$$

The analyses of bias, variance, and covariance of these estimates are similar in complexity to those of the autocorrelation spectral estimates, and the details can be found in Goodman (1957) and Jenkins and Watts (1968).

Estimation of cross-amplitude and phase spectra. Following the definitions in (5.4.6) and (5.4.7), we may estimate the cross-amplitude spectrum $A_{xy}(\omega)$ and the phase spectrum $\Phi_{xy}(\omega)$ between the random processes $x(n)$ and $y(n)$ by

$$\hat{A}_{xy}^{(\text{PA})}(\omega) = \sqrt{[\hat{C}_{xy}^{(\text{PA})}(\omega)]^2 + [\hat{Q}_{xy}^{(\text{PA})}(\omega)]^2} \quad (5.4.18)$$

and

$$\hat{\Phi}_{xy}^{(\text{PA})}(\omega) = \tan^{-1}\{-\hat{Q}_{xy}^{(\text{PA})}(\omega)/\hat{C}_{xy}^{(\text{PA})}(\omega)\} \quad (5.4.19)$$

where the estimates $\hat{C}_{xy}^{(\text{PA})}(e^{j\omega})$ and $\hat{Q}_{xy}^{(\text{PA})}(e^{j\omega})$ are given by (5.4.16) and (5.4.17), respectively. Since the cross-amplitude and phase spectral estimates are nonlinear functions of the cospectral and quadrature spectral estimates, their analysis in terms of bias, variance, and covariance is much more complicated. Once again, the details are available in Jenkins and Watts (1968).

Estimation of coherency spectrum. The coherency spectrum is given by the magnitude of the complex coherence $\hat{\mathcal{G}}_{xy}(\omega)$. Replacing $R_{xy}(e^{j\omega})$, $R_x(e^{j\omega})$, and $R_y(e^{j\omega})$ by their estimates in (5.4.8), we see the estimate for the coherency spectrum is given by

$$|\hat{\mathcal{G}}_{xy}^{(\text{PA})}(\omega)| = \frac{|\hat{R}_{xy}^{(\text{PA})}(e^{j\omega})|}{\sqrt{\hat{R}_x^{(\text{PA})}(e^{j\omega}) \hat{R}_y^{(\text{PA})}(e^{j\omega})}} = \left\{ \frac{[\hat{C}_{xy}^{(\text{PA})}(\omega)]^2 + [\hat{Q}_{xy}^{(\text{PA})}(\omega)]^2}{\hat{R}_x^{(\text{PA})}(e^{j\omega}) \hat{R}_y^{(\text{PA})}(e^{j\omega})} \right\}^{1/2} \quad (5.4.20)$$

with bias and variance properties similar to those of the cross-amplitude spectrum.

In MATLAB the function

```
Rxy=csd(x,y,Nfft,Fs,window(L),Noverlap);
```

is available, which is similar to the `psd` function described in Section 5.3.3. It estimates the cross-spectral density of signal vectors x and y by using Welch's method. The `window` parameter specifies a window function, `Fs` is the sampling frequency for plotting purposes,

`Nfft` is the size of the FFT used, and `Noverlap` specifies the number of overlapping samples. The function

```
cohere(x,y,Nfft,Fs,window(L),Noverlap);
```

estimates the coherency spectrum between two vectors `x` and `y`. Its values are between 0 and 1.

5.4.2 Estimation of Frequency Response Functions

When random processes $x(n)$ and $y(n)$ are the input and output of some physical system, the bivariate spectral estimation techniques discussed in this section can be used to estimate the system characteristics, namely, its frequency response. Problems of this kind arise in many applications including communications, industrial control, and biomedical signal processing. In communications applications, we need to characterize a channel over which signals are transmitted. In this situation, a known training signal is transmitted, and the channel response is recorded. By using the statistics of these two signals, it is possible to estimate channel characteristics within a reasonable accuracy. In the industrial applications such as a gas furnace, the classical methods using step (or sinusoidal) inputs may be inappropriate because of large disturbances generated within the system. Hence, it is necessary to use statistical methods that take into account noise generated in the system.

From Chapter 3, we know that if $x(n)$ and $y(n)$ are input and output signals of an LTI system characterized by the impulse response $h(n)$, then

$$y(n) = h(n) * x(n) \quad (5.4.21)$$

The impulse response $h(n)$, in principle, can be computed through the deconvolution operation. However, deconvolution is not always computationally feasible. If the input and output processes are jointly stationary, then from Chapter 3 we know that the cross-correlation between these two processes is given by

$$r_{yx}(l) = h(l) * r_x(l) \quad (5.4.22)$$

and the cross-spectrum is given by

$$R_{yx}(e^{j\omega}) = H(e^{j\omega})R_x(e^{j\omega}) \quad (5.4.23)$$

or

$$H(e^{j\omega}) = \frac{R_{yx}(e^{j\omega})}{R_x(e^{j\omega})} \quad (5.4.24)$$

Hence, if we can estimate the auto power spectrum and cross power spectrum with reasonable accuracy, then we can determine the frequency response of the system.

Consider next an LTI system with additive output noise,[†] as shown in Figure 5.26. This model situation applies to many practical problems where the input measurements $x(n)$ are essentially without noise while the output measurements $y(n)$ can be modeled by the sum of the ideal response $y_o(n)$ due to $x(n)$ and an additive noise $v(n)$, which is statistically independent of $x(n)$. If we observe the input $x(n)$ and the ideal output $y_o(n)$, the frequency response can be obtained by

$$H(e^{j\omega}) = \frac{R_{y_o x}(e^{j\omega})}{R_x(e^{j\omega})} \quad (5.4.25)$$

where all signals are assumed stationary with zero mean (see Section 5.3.1). Since $x(n)$ and $v(n)$ are independent, we can easily show that

$$R_{y_o x}(e^{j\omega}) = R_{y x}(e^{j\omega}) \quad (5.4.26)$$

[†] More general situations involving both additive input noise and additive output noise are discussed in Bendat and Piersol (1980).

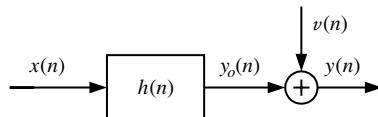


FIGURE 5.26
 Input-output LTI system model with output noise.

and

$$R_y(e^{j\omega}) = R_{y_o}(e^{j\omega}) + R_v(e^{j\omega}) \quad (5.4.27)$$

where

$$R_{y_o}(e^{j\omega}) = |H(e^{j\omega})|^2 R_x(e^{j\omega}) \quad (5.4.28)$$

is the ideal output PSD produced by the input. From (5.4.25) and (5.4.26), we have

$$H(e^{j\omega}) = \frac{R_{yx}(e^{j\omega})}{R_x(e^{j\omega})} \quad (5.4.29)$$

which shows that we can determine the frequency response by using the cross power spectral density between the noisy output and the input signals. Given a finite record of input-output data $\{x(n), y(n)\}_0^{N-1}$, we estimate $\hat{R}_{yx}^{(\text{PA})}(e^{j\omega_k})$ and $\hat{R}_x^{(\text{PA})}(e^{j\omega_k})$ by using one of the previously discussed methods and then estimate $H(e^{j\omega})$ at a set of equidistant frequencies $\{\omega_k = 2\pi k/K\}_0^{K-1}$, that is,

$$\hat{H}(e^{j\omega_k}) = \frac{\hat{R}_{yx}^{(\text{PA})}(e^{j\omega_k})}{\hat{R}_x^{(\text{PA})}(e^{j\omega_k})} \quad (5.4.30)$$

The coherence function, which measures the linear correlation between two signals $x(n)$ and $y(n)$ in the frequency domain, is given by

$$\mathcal{G}_{xy}^2(\omega) = \frac{|R_{xy}(e^{j\omega})|^2}{R_x(e^{j\omega})R_y(e^{j\omega})} \quad (5.4.31)$$

and satisfies the inequality $0 \leq \mathcal{G}_{xy}^2(\omega) \leq 1$ (see Section 3.3.6). If $R_{xy}(e^{j\omega}) = 0$ for all ω , then $\mathcal{G}_{xy}^2(\omega) = 0$. On the other hand, if $y(n) = h(n) * x(n)$, then $\mathcal{G}_{xy}^2(\omega) = 1$ because $R_y(e^{j\omega}) = |H(e^{j\omega})|^2 R_x(e^{j\omega})$ and $R_{xy}(e^{j\omega}) = H^*(e^{j\omega})R_x(e^{j\omega})$. Furthermore, we can show that the coherence function is invariant under linear transformations. Indeed, if $x_1(n) = h_1(n)*x(n)$ and $y_1(n) = h_2(n)*y(n)$, then $\mathcal{G}_{xy}^2(\omega) = \mathcal{G}_{x_1 y_1}^2(\omega)$ (see Problem 5.16). To avoid delta function behavior at $\omega = 0$, we should remove the mean value from the data before we compute $\mathcal{G}_{xy}^2(\omega)$. Also $R_x(e^{j\omega})R_y(e^{j\omega}) > 0$ to avoid division by 0.

In practice, the coherence function is usually greater than 0 and less than 1. This may result from one or more of the following reasons (Bendat and Piersol 1980):

1. Excessive measurement noise.
2. Significant resolution bias in the spectral estimates.
3. The system relating $y(n)$ to $x(n)$ is nonlinear.
4. The output $y(n)$ is *not* produced exclusively by the input $x(n)$.

Using (5.4.28), (5.4.25), $R_{xy_o}(e^{j\omega}) = H^*(e^{j\omega})R_x(e^{j\omega})$, and (5.4.31), we obtain

$$R_{y_o}(e^{j\omega}) = \mathcal{G}_{xy}^2(\omega)R_y(e^{j\omega}) \quad (5.4.32)$$

which is known as the *coherent output PSD*. Combining the last equation with (5.4.27), we have

$$R_v(e^{j\omega}) = [1 - \mathcal{G}_{xy}^2(\omega)]R_y(e^{j\omega}) \quad (5.4.33)$$

which can be interpreted as the part of the output PSD that cannot be produced from the input by using linear operations.

which shows that $\mathcal{G}_{xy}^2(\omega) \rightarrow 1$ as $R_v(e^{j\omega})/R_y(e^{j\omega}) \rightarrow 0$ and $\mathcal{G}_{xy}^2(\omega) \rightarrow 0$ as $R_v(e^{j\omega})/R_y(e^{j\omega}) \rightarrow 1$. Typically, the coherence function between input and output measurements reveals the presence of errors and helps to identify their origin and magnitude. Therefore, the coherence function provides a useful tool for evaluating the accuracy of frequency response estimates.

In MATLAB the function

```
H = tfe(x,y,Nfft,fs,window(L),Noverlap)
```

is available that estimates the transfer function of the system with input signal x and output y using Welch's method. The `window` parameter specifies a window function, `fs` is the sampling frequency for plotting purposes, `Nfft` is the size of the FFT used, and `Noverlap` specifies the number of overlapping samples.

We next provide two examples that illustrate some of the problems that may arise when we estimate frequency response functions by using input and output measurements.

EXAMPLE 5.4.1. Consider the AP(4) system

$$H(z) = \frac{1}{1 - 2.7607z^{-1} + 3.8106z^{-2} - 2.6535z^{-3} + 0.9238z^{-4}}$$

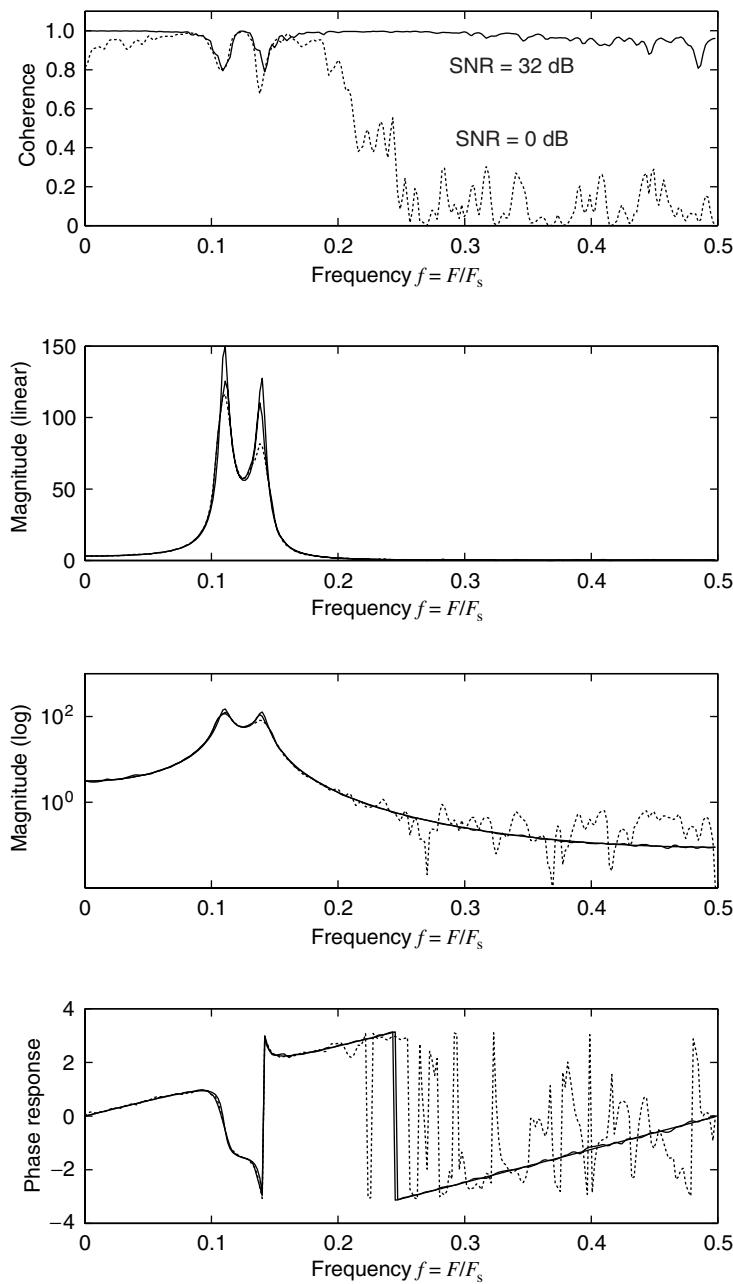
discussed in Example 5.3.2. The input is white Gaussian noise, and the output of this system is corrupted by additive white Gaussian noise, as shown in Figure 5.27. We wish to estimate the frequency response of the system from a set of measurements $\{x(n), y(n)\}_0^{N-1}$. Since the input is white, when the output signal-to-noise ratio (SNR) is very high, we can estimate the magnitude response of the system by computing the PSD of the output signal. However, to compute the phase response or a more accurate estimate of the magnitude response, we should use the joint measurements of the input and output signals, as explained above.

Figure 5.27 shows estimates of the MSC function, magnitude response functions (in linear and log scales), and phase response functions for two different levels of output SNR: 32 and 0 dB. When SNR = 32 dB, we note that $|\mathcal{G}_{xy}(\omega)|$ is near unity at almost all frequencies, as we theoretically expect for ideal LTI input-output relations. The estimated magnitude and phase responses are almost identical to the theoretical ones with the exception at the two sharp peaks of $|H(e^{j\omega})|$. Since the SNR is high, the two notches in $|\mathcal{G}_{xy}(\omega)|$ at the same frequencies suggest a bias error due to the lack of sufficient frequency resolution. When SNR = 0 dB, we see that $|\mathcal{G}_{xy}(\omega)|$ falls very sharply for frequencies above 0.2 cycle per sampling interval. We notice that the presence of noise increases the random errors in the estimates of magnitude and phase response in this frequency region, and the bias error in the peaks of the magnitude response. Finally, we note that the uncertainty fluctuations in $|\mathcal{G}_{xy}(\omega)|$ increase as $|\mathcal{G}_{xy}(\omega)| \rightarrow 0$, as predicted by the formula

$$\frac{\text{std}[|\hat{\mathcal{G}}_{xy}^{(\text{PA})}(\omega)|]}{|\mathcal{G}_{xy}(\omega)|} = \sqrt{2} \frac{1 - |\hat{\mathcal{G}}_{xy}^{(\text{PA})}(\omega)|^2}{|\mathcal{G}_{xy}^{(\text{PA})}(\omega)|\sqrt{K}} \quad (5.4.35)$$

where $\text{std}(\cdot)$ means standard deviation and K is the number of averaged segments (Bendat and Piersol 1980).

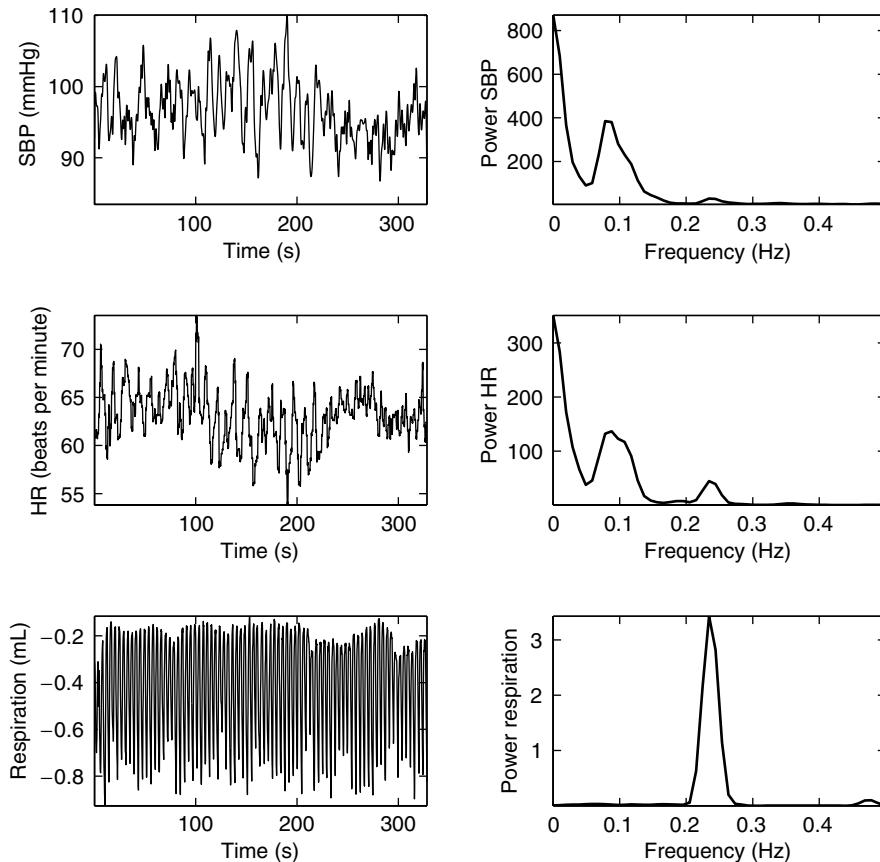
EXAMPLE 5.4.2. In this example we illustrate the use of frequency response estimation to study the effect of respiration and blood pressure on heart rate. Figure 5.28 shows the systolic blood pressure (mmHg), heart rate (beats per minute), and the respiration (mL) signals with their corresponding PSD functions (Grossman 1998). The sampling frequency is $F_s = 5$ Hz, and the PSDs were estimated using the method of averaged periodograms with 50 percent overlap. Note the corresponding quasiperiodic oscillations of blood pressure and heart rate occurring approximately every 12 s (0.08 Hz). Close inspection of the heart rate time series will also reveal

**FIGURE 5.27**

Estimated coherence, magnitude response, and phase response for the AP(4) system. The solid lines show the ideal magnitude and phase responses.

another rhythm corresponding to the respiratory period (about 4.3 s, or 0.23 Hz). These rhythms reflect nervous system mechanisms that control the activity of the heart and the circulation under most circumstances.

The left column of Figure 5.29 shows the coherence, magnitude response, and phase response between respiration as input and heart rate as output. Heart rate fluctuates clearly at the respiratory frequency (here at 0.23 Hz); this is indicated by the large amount of heart rate power and the high degree of coherence at the respiratory frequency. Heart function is largely controlled

**FIGURE 5.28**

Continuous systolic blood pressure (SBP), heart rate (HR), and respiration of a young man during quiet upright tilt and their estimated PSDs.

by two branches of the autonomic nervous system, the parasympathetic and sympathetic. Frequency analysis of cardiovascular signals may improve our understanding of the manner in which these two branches interact under varied circumstances. Heart rate fluctuations at the respiratory frequency (termed *respiratory sinus arrhythmia*) are primarily mediated by the parasympathetic branch of the autonomic nervous system. Increases in respiratory sinus arrhythmia indicate enhanced parasympathetic influence upon the heart. Sympathetic oscillations of heart rate occur only at slower frequencies (below 0.10 Hz) owing to the more sluggish frequency response characteristics of the sympathetic branch of the autonomic nervous system.

The right column of Figure 5.29 shows the coherence, magnitude response, and phase response between systolic blood pressure as input and heart rate as output. Coherent oscillations among cardiac and blood pressure signals can often be discerned in a frequency band with a typical center frequency of 0.10 Hz (usual range, 0.07 to 0.12 Hz). This phenomenon has been tied to the cardiovascular baroreflex system, which involves baroreceptors, that is, bodies of cells in the carotid arteries and aorta that are sensitive to stretch. When blood pressure is increased, these baroreceptors fire proportionally to stretch and pressure changes, sending commands via the brain to the heart and circulatory system. This baroreflex system is the only known physiological system acting to buffer rapid and extreme surges or falls in blood pressure. Increased baroreceptor stretch, for example, slows the heart rate by means of increased parasympathetic activity; decreased baroreceptor stretch will elicit cardiovascular sympathetic activation that will speed the heart and constrict arterial vessels. Thus pressure drops due to a decrease in flow. The 0.10-Hz blood pressure oscillations (see PSD in Figure 5.28) are sympathetic in origin and are produced by periodic sympathetic constriction of arterial blood vessels.

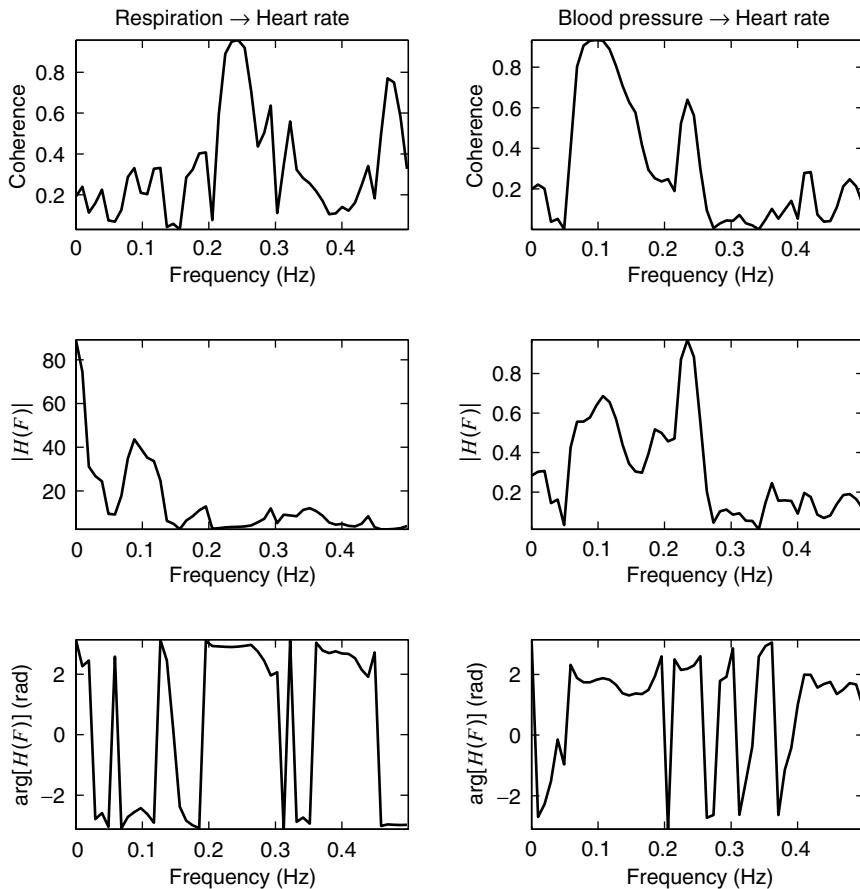


FIGURE 5.29

Coherence, magnitude response, and phase response between respiration as input and heart rate as output, and systolic blood pressure as input and heart rate as output.

5.5 MULTITAPER POWER SPECTRUM ESTIMATION

Tapering is another name for the data windowing operation in the time domain. The periodogram estimate of the power spectrum, discussed in Section 5.3, is an operation on a data record $\{x(n)\}_{n=0}^{N-1}$. One interpretation of this finite-duration data record is that it is obtained by truncating an infinite-duration process $x(n)$ with a rectangular window (or taper). Since bias and variance properties of the periodogram estimate are unacceptable, methods for bias and variance reduction were developed either by smoothing estimates in the frequency domain (using lag windows) or by averaging periodograms computed over several short segments (data windows). Since these window functions (other than the rectangular one) typically taper the response toward both ends of the data record, windows are also referred to as *tapers*.

In 1982, Thomson suggested an alternate approach for producing a “direct” (or “raw”) periodogram-based spectral estimator. In this method, rather than use a single rectangular data taper as in the periodogram estimate, several data tapers are used on the same data record to compute several modified periodograms. These modified periodograms are then averaged (with or without weighting) to produce the multitaper spectral estimate. The central premise of this multitaper approach is that if the data tapers are properly designed orthogonal functions, then, under mild conditions, the spectral estimates would be independent of each other at every frequency. Thus, averaging would reduce the variance while proper design of

full-length windows would reduce bias and loss of resolution. Thomson suggested windows based on discrete prolate spheroidal sequences (DPSSs) that form an orthonormal set, although any other orthogonal set with desirable properties can also be used. This DPSS set is also known as the set of *Slepian* tapers. The multitaper method is different in spirit from the other methods in that it does not seek to produce highly smoothed spectra. Detailed discussions of the multitaper approach are given in Thomson (1982) and in Percival and Walden (1993). In this section, we provide a brief sketch of the algorithm.

5.5.1 Estimation of Auto Power Spectrum

Given a data record $\{x(n)\}_{n=0}^{N-1}$ of length N , consider a set of K data tapers $\{w_k(n); 0 \leq n \leq N-1, 0 \leq k \leq K-1\}$. These tapers are assumed to be orthonormal, that is,

$$\sum_{n=0}^{N-1} w_k(n) w_l(n) = \begin{cases} 1 & k = l \\ 0 & k \neq l \end{cases} \quad (5.5.1)$$

Let $\hat{R}_{k,x}(e^{j\omega})$ be the periodogram estimator based on k th taper. Then, similar to (5.3.2), we obtain

$$\hat{R}_{k,x}(e^{j\omega}) = \frac{1}{N} \left| \sum_{n=0}^{N-1} w_k(n) x(n) e^{-j\omega n} \right|^2 \quad (5.5.2)$$

The simple averaged multitaper (MT) estimator is then defined by

$$\hat{R}_x^{(\text{MT})}(e^{j\omega}) = \frac{1}{K} \sum_{k=0}^{K-1} \hat{R}_{k,x}(e^{j\omega}) \quad (5.5.3)$$

A pictorial description of this multitaper algorithm is shown in Figure 5.30. Another approach, suggested by Thomson, is to apply adaptive weights (both frequency- and data-dependent) prior to averaging to protect against the biasing degradations of different tapers.

In either case, the multitaper estimator is an average of direct spectral estimators (called eigenspectra by Thomson) employing an orthonormal set of tapers. Thomson (1982) showed that under mild conditions, the orthonormality of the tapers results in an approximate independence of each individual $\hat{R}_{k,x}(e^{j\omega})$ at every frequency ω . This approximate independence further implies that the equivalent degrees of freedom for $\hat{R}_x^{(\text{MT})}(e^{j\omega})$ are equal to twice the number of data tapers. This increase in degrees of freedom is enough to shrink the width of the 95 percent confidence interval for $\hat{R}_x^{(\text{MT})}(e^{j\omega})$ and to reduce the variability to the point at which the overall shape of the spectrum is easily recognizable even though the spectrum is not highly smoothed.

Clearly, the success of this approach lies in the selection of K orthonormal tapers. To understand the rationale behind the selection of these tapers, consider the bias or mean of $\hat{R}_{k,x}(e^{j\omega})$. Following (5.3.15), we obtain

$$E\{\hat{R}_{k,x}(e^{j\omega})\} = \frac{1}{2\pi N} \int_{-\pi}^{\pi} R_x(e^{j\theta}) R_{k,w}(e^{j(\omega-\theta)}) d\theta \quad (5.5.4)$$

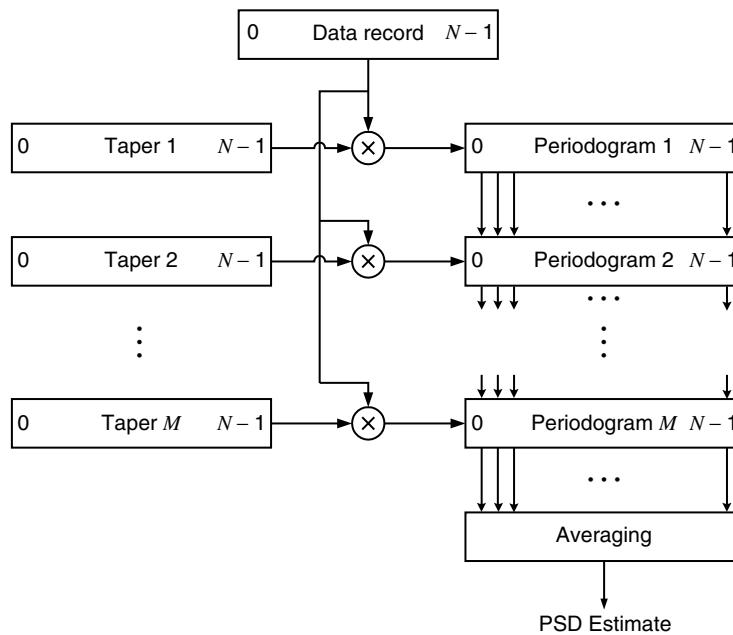
where $R_{k,w}(e^{j\omega}) = \mathcal{F}\{w_k(n) * w_k(-n)\} = |W_k(e^{j\omega})|^2$

It follows, then, from (5.5.3) that

$$E\{\hat{R}_x^{(\text{MT})}(e^{j\omega})\} = \frac{1}{2\pi N} \int_{-\pi}^{\pi} R_x(e^{j\theta}) \bar{R}_w(e^{j(\omega-\theta)}) d\theta \quad (5.5.6)$$

where

$$\bar{R}_w(e^{j\omega}) \triangleq \frac{1}{K} \sum_{k=0}^{K-1} |W_k(e^{j\omega})|^2 \quad (5.5.7)$$

**FIGURE 5.30**

A pictorial description of the multitaper approach to power spectrum estimation.

The function $\bar{R}_w(e^{j\omega})$ is the spectral window of the averaged multitaper estimator, which is obtained by averaging spectra of the individual tapers. Hence, for $\bar{R}_w(e^{j\omega})$ to produce a good leakage-free estimate $\hat{R}_x^{(MT)}(e^{j\omega})$, all K spectral windows must provide good protection against leakage. Therefore, each taper must have low sidelobe levels. Furthermore, the averaging of K individual periodograms also reduces the overall variance of $\hat{R}_x^{(MT)}(e^{j\omega})$. The reduction in variance is possible if the $\hat{R}_{k,x}(e^{j\omega})$ are pairwise uncorrelated with common variance, in which case the variance reduces by a factor of $1/K$.

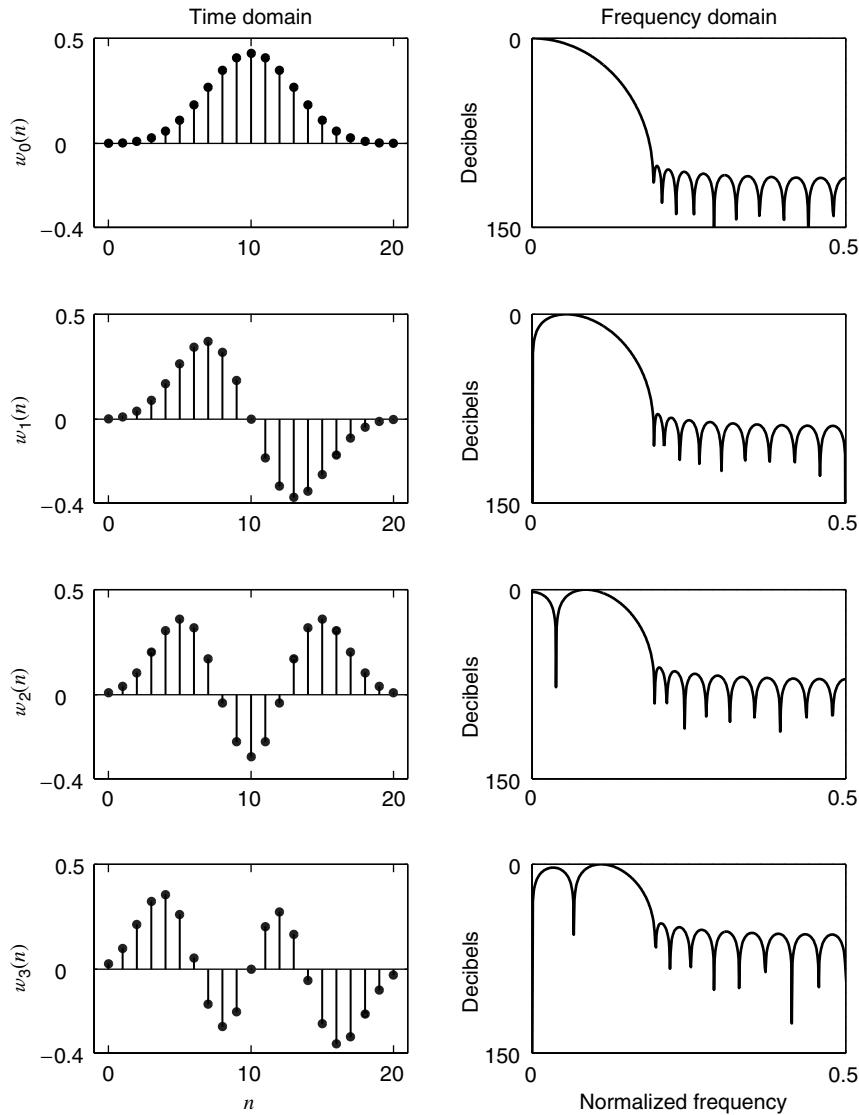
Thus, we need K orthonormal data tapers such that each one provides a good protection against leakage and such that the resulting individual spectral estimates are nearly uncorrelated. One such set is obtained by using DPSS with parameter W and of orders $k = 0, \dots, K - 1$, where K is chosen to be less than or equal to the number $2W$ (called the *Shannon number*, which is also a fixed-resolution bandwidth). The design of these sequences is discussed in detail in Thomson (1982) and in Percival and Walden (1993). In MATLAB these tapers are generated by using the `[w]=dpss(L,W)` function, where L is the length of $2W$ tapers computed in matrix w .

The first four 21-point DPSS tapers with $W = 4$ and their Fourier transforms are shown in Figure 5.31 while the next four DPSS tapers are shown in Figure 5.32. It can be seen that higher-order tapers assume both positive and negative values. The zeroth-order taper (like other windows) heavily attenuates data values near $n = 0$ and $n = L$. The higher-order tapers successively give greater weights to these values to the point that tapers for $k \geq K$ have very poor bias properties and hence are not used. This behavior is quite evident in the frequency domain where as the taper order increases, mainlobe width and sidelobe attenuation decrease. The multitapering approach can be interpreted as a technique in which higher-order tapers capture information that is “lost” when only the first taper is used.

In MATLAB the function

$$[Pxx, Pxxc, F] = PMTM(x, W, Nfft, Fs)$$

estimates the power spectrum of the data vector x in the array Pxx , using the multitaper approach. The function uses DPSS tapers with parameter w and adaptive weighted averaging

**FIGURE 5.31**

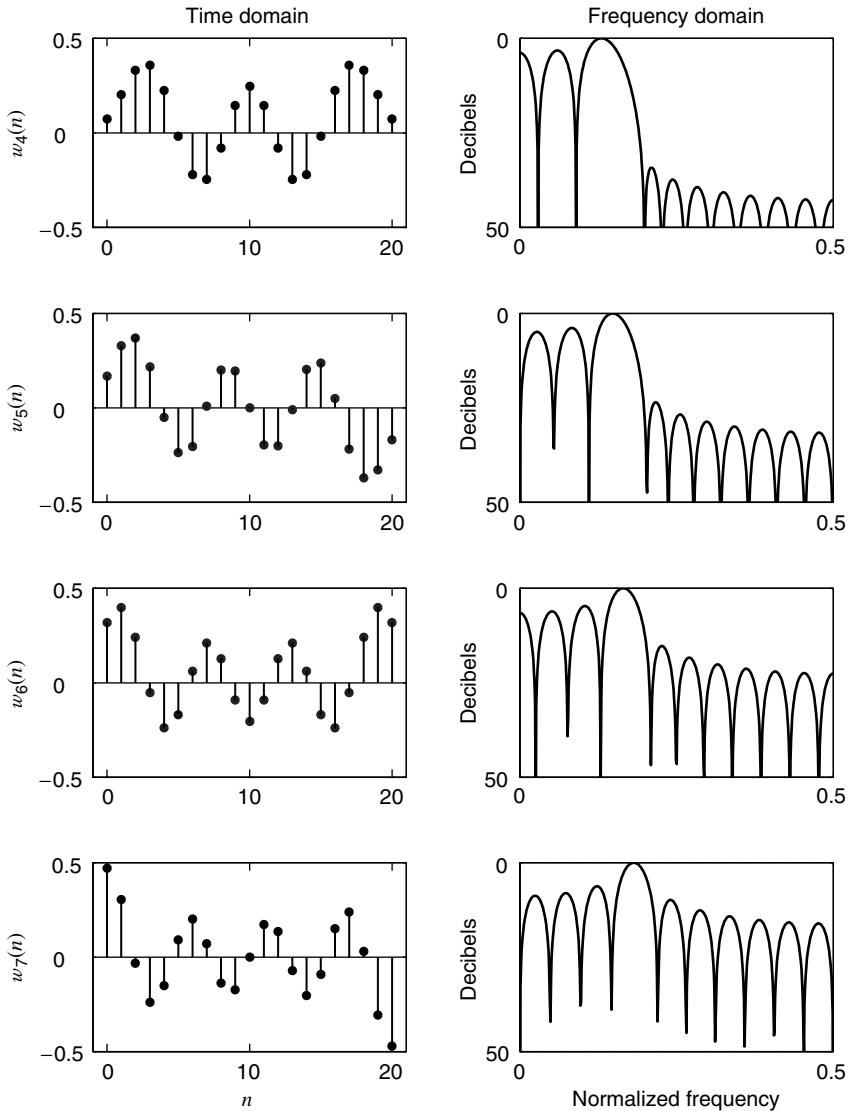
DPSS data tapers for $k = 0, 1, 2, 3$ in the time and frequency domains.

as the default method. The 95 percent confidence interval is available in `Pxxc`. The size of the DFT used is `Nfft`, the sampling frequency is `Fs`, and the frequency values are returned in the vector `F`.

Another much simpler set of orthonormal tapers was suggested by Reidel and Siderenko (1995). This particular set contains harmonically related sinusoidal tapers. One important aspect of multitapering is to reduce the periodogram variance without reducing resolution caused by smoothing across frequencies. If the spectrum is changing slowly across the band so that sidelobe bias is not severe (recall the argument given for the unbiasedness of the periodogram for the white noise process), then sine tapers can reduce the variance. The k th taper in this set of $k = 0, 1, \dots, N - 1$ tapers is given by

$$w_k(n) = \sqrt{\frac{2}{N+1}} \sin \frac{\pi(k+1)(n+1)}{N+1} \quad n = 0, 1, \dots, N-1 \quad (5.5.8)$$

where the amplitude term on the right is a normalization factor that ensures orthonormality of the tapers. These sine tapers have much narrower mainlobe but also much higher sidelobes

**FIGURE 5.32**

DPSS data tapers for $k = 4, 5, 6, 7$ in the time and frequency domains.

(recall the rectangular window) than the DPSS tapers. Thus they achieve a smaller bias due to smoothing by the mainlobe than the DPSS tapers, but at the expense of sidelobe suppression. Clearly this performance is acceptable if the spectrum is varying slowly. Owing to their simple nature, these tapers can be analyzed analytically, and it can be shown that (Reidel and Siderenko 1995) the k th sinusoidal taper has its spectral energy concentrated in the frequency bands

$$\frac{\pi k}{N+1} \leq |\omega| \leq \frac{\pi(k+2)}{N+1} \quad k = 0, 1, \dots, N-1 \quad (5.5.9)$$

If the first $K < N$ tapers are used, then the multitaper estimator has the spectral window concentrated in the band

$$\left[-\frac{K+1}{N+1}, \frac{K+1}{N+1} \right] \quad (5.5.10)$$

A summary of the multitaper algorithm performance and its comparison with other PSD estimation methods are given in Table 5.3.

EXAMPLE 5.5.1 (THREE SINUSOIDS IN WHITE NOISE). Consider the random process $x(n)$ containing three sinusoids in white noise discussed earlier, that is,

$$x(n) = \cos(0.35\pi n + \phi_1) + \cos(0.4\pi n + \phi_2) + 0.25 \cos(0.8\pi n + \phi_3) + v(n)$$

Fifty realizations of $x(n)$, $0 \leq n \leq N - 1$, were processed using the PMTM function to obtain multitaper spectrum estimates for $K = 3, 5$, and 7 Slepian tapers. The results are shown in Figure 5.33 in the form of overlays and averages. Several interesting observations and comparisons with the previous methods can be made. The number of tapers used in the estimation determines the variance and the smearing of the spectrum. When fewer tapers are used, the peaks are sharper and narrower but the noise variance is larger. After increasing the number of tapers, the variance is decreased but the peaks become wider. When these estimates are compared with those from Welch's method, an interesting feature can be noticed. The broadening of the peaks is not just at the base but is present along the entire length of the peak. Therefore, even with seven tapers, peaks are distinguishable. This feature is due to the bandwidth of the average spectral window due to K tapers.

EXAMPLE 5.5.2 (OCEAN WAVE DATA). Consider the wire gage wave data of Figure 5.24. The multitaper estimate $\hat{R}_x^{(\text{MT})}(e^{j\omega})$ of these 4096-point data is obtained using the PMTM function in which the parameter W is set to 4. The results are shown in Figure 5.34. In this graph, the middle solid is the spectral estimate in decibels while the upper and lower solid curves are the upper

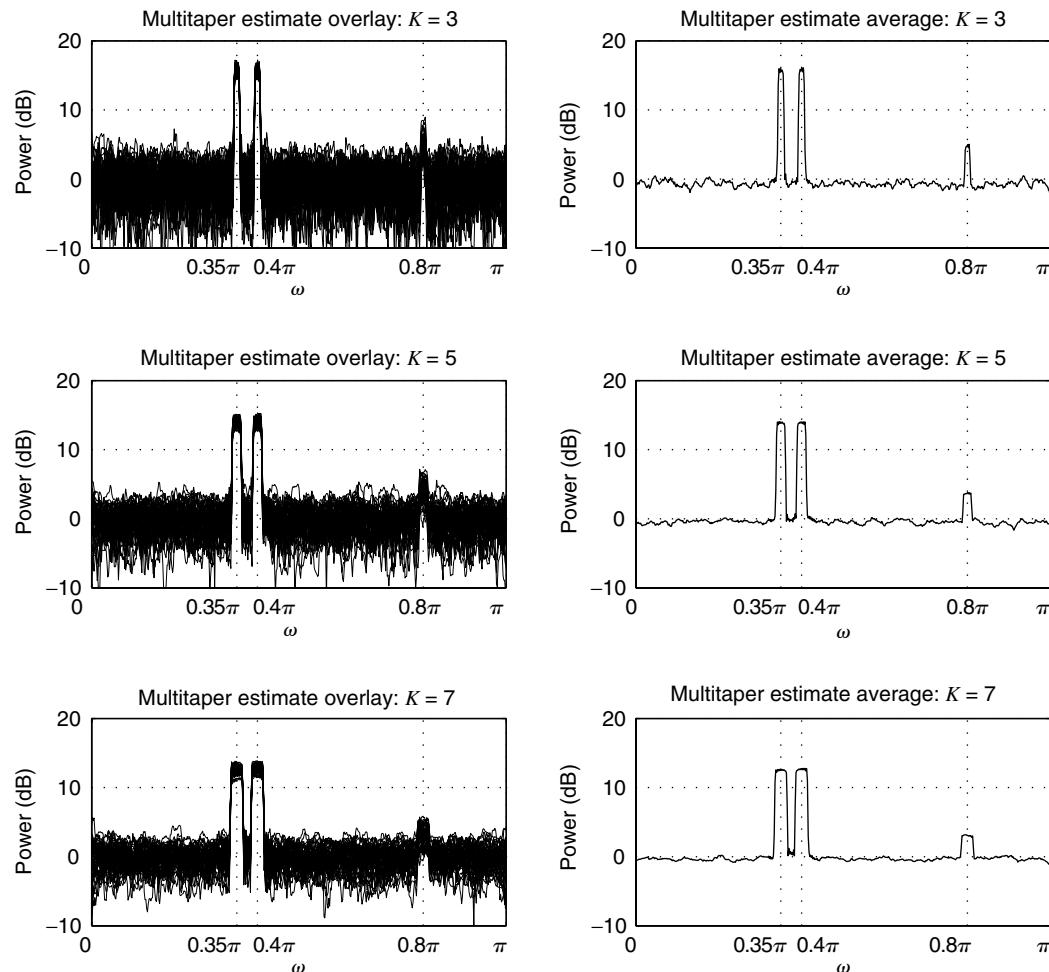


FIGURE 5.33

Spectrum estimation of three sinusoids in white noise using the multitaper method in Example 5.5.1.

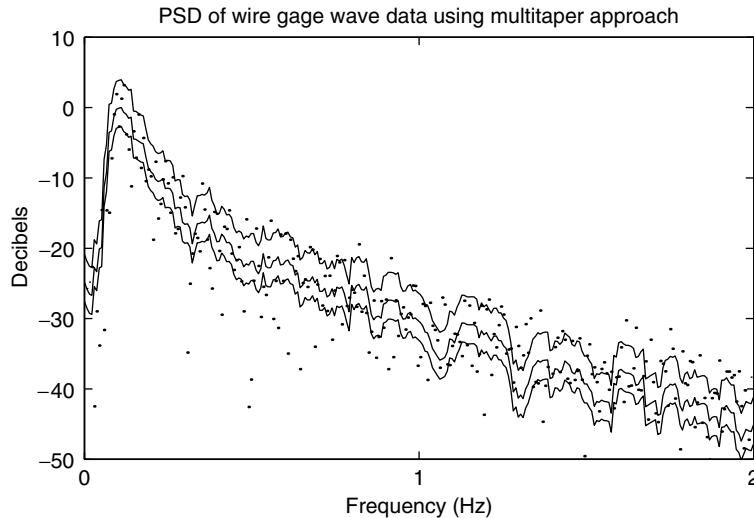


FIGURE 5.34

Spectrum estimation of the wire gage wave data using the multitaper method in Example 5.5.2.

and lower limits of the 95 percent confidence interval for a fixed frequency. For comparison purposes, the “raw” periodogram estimate is also shown as small dots. Clearly, the periodogram has a large variability that is reduced in the multitaper estimate. At the same time, the multitaper estimate is not smooth, but its variability is small enough to follow the shape of the overall structure.

5.5.2 Estimation of Cross Power Spectrum

The multitapering approach can also be extended to the estimation of the cross power spectrum. Following (5.4.11), the multitaper estimator of the cross power spectrum is given by

$$\hat{R}_{xy}^{(\text{MT})}(e^{j\omega}) = \frac{1}{KL_{xy}} \sum_{k=0}^{K-1} \left[\sum_{n=0}^{L_{xy}-1} w_k(n)x(n)e^{-j\omega n} \right] \left[\sum_{n=0}^{L_{xy}-1} w_k(n)y(n)e^{-j\omega n} \right]^* \quad (5.5.11)$$

where $w_k(n)$ is the k th-order data taper of length L_{xy} and a fixed-resolution bandwidth of $2W$. As with the auto power spectrum, the use of multitaper averaging reduces the variability of the cross-periodogram $R_{xy}(e^{j\omega})$. Once again, the number of equivalent degrees of freedom for $\hat{R}_{xy}(e^{j\omega})$ is equal to $2K$.

The real-valued functions associated with the cross power spectrum can also be estimated by using the multitaper approach in a similar fashion. The cospectrum and the quadrature spectrum are given by

$$\hat{C}_{xy}^{(\text{MT})}(\omega) = \text{Re}[\hat{R}_{xy}^{(\text{MT})}(e^{j\omega})] \quad \text{and} \quad \hat{Q}_{xy}^{(\text{MT})}(\omega) = -\text{Im}[\hat{R}_{xy}^{(\text{MT})}(e^{j\omega})] \quad (5.5.12)$$

while the cross-amplitude spectrum and the phase spectrum are given by

$$\hat{A}_{xy}^{(\text{MT})}(\omega) = \sqrt{[\hat{C}_{xy}^{(\text{MT})}(\omega)]^2 + [\hat{Q}_{xy}^{(\text{MT})}(\omega)]^2} \quad \text{and} \quad \hat{\Phi}_{xy}^{(\text{MT})}(\omega) = \tan^{-1} \left[-\frac{\hat{Q}_{xy}^{(\text{MT})}(\omega)}{\hat{C}_{xy}^{(\text{MT})}(\omega)} \right] \quad (5.5.13)$$

Finally, the coherency spectrum is given by

$$|\hat{G}_{xy}^{(\text{MT})}(\omega)| = \left\{ \frac{[\hat{C}_{xy}^{(\text{MT})}(\omega)]^2 + [\hat{Q}_{xy}^{(\text{MT})}(\omega)]^2}{\hat{R}_x^{(\text{MT})}(e^{j\omega})\hat{R}_y^{(\text{MT})}(e^{j\omega})} \right\}^{1/2} \quad (5.5.14)$$

MATLAB does not provide a function for cross power spectrum estimation using the multitaper approach. However, by using the DPSS function, it is relatively straightforward to implement the simple averaging method of (5.5.11).

EXAMPLE 5.5.3. Again consider the wire gage and the infrared gage wave data of Figure 5.24. The multitaper estimate of the cross power spectrum $\hat{R}_{xy}^{(MT)}(e^{j\omega})$ of these two 4096-point sequences is obtained by using (5.5.11) in which the parameter W is set to 4. Figure 5.35 shows plots of the estimates of the auto power spectra of the two data sets in solid lines. The cross power spectrum of the two signals is shown with a dotted line. It is interesting to note that the two auto power spectra agree almost perfectly over the band up to 0.3 Hz and then reasonably well up to 0.9 Hz, beyond which point the spectrum due to the infrared gage is consistently higher due to high-frequency noise inherent in the measurements. The cross power spectrum agrees with the two auto power spectra at low frequencies up to 0.2 Hz. Figure 5.36 contains two graphs; the

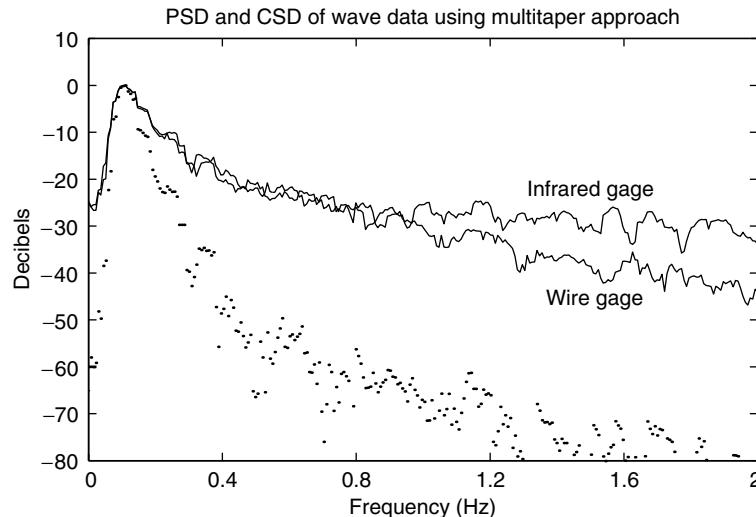


FIGURE 5.35

Cross power spectrum estimation of the wave data using the multitaper approach.

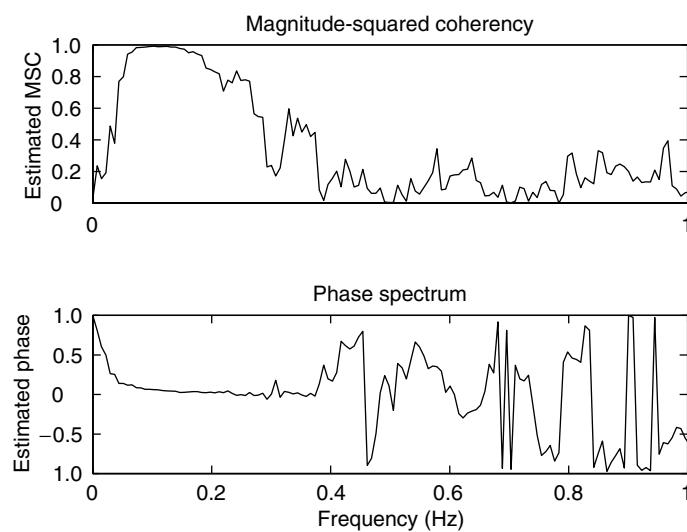


FIGURE 5.36

Coherency and phase spectrum of the wave data using the multitaper approach.

upper graph is for the MSC while the lower one is for the phase spectrum. Consistent with our observation of the cross power spectrum in Figure 5.36, the MSC is almost one over these lower frequencies. The phase spectrum is almost a linear function over the range over which the two auto power spectra agree. Thus, the multitaper approach provides estimates that agree with the conventional techniques.

5.6 SUMMARY

In this chapter, we presented many different nonparametric methods for estimating the power spectrum of a wide-sense stationary random process. Nonparametric methods do not depend on any particular model of the process but use estimators that are determined entirely by the data. Therefore, one has to be very careful about the data and the interpretation of results based on them.

We began by revisiting the topic of frequency analysis of deterministic signals. Since the spectrum estimation of random processes is based on the Fourier transformation of data, the purpose of this discussion was to identify and study errors associated with the practical implementation. In this regard, three problems—the sampling of the continuous signal, windowing of the sampled data, and the sampling of the spectrum—were isolated and discussed in detail. Some useful data windows and their characteristics were also given. This background was necessary to understand more complex spectrum estimation methods and their results.

An important topic of autocorrelation estimation was considered next. Although this discussion was not directly related to spectrum estimation, its inclusion was appropriate since one important method (i.e., that of Blackman and Tukey) was based on this estimation. The statistical properties of the estimator and its implementation completed this topic.

The major part of this chapter was devoted to the section on the auto power spectrum estimation. The classical approach was to develop an estimator from the Fourier transform of the given values of the process. This was called the periodogram method, and it resulted in a natural PSD estimator as a Fourier transform of an autocorrelation estimate. Unfortunately, the statistical analysis of the periodogram showed that it was not an unbiased estimator or a consistent estimator; that is, its variability did not decrease with increasing data record length. The modification of the periodogram using the data window lessened the spectral leakage and improved the unbiasedness but did not decrease the variance. Several examples were given to verify these aspects.

To improve the statistical performance of the simple periodogram, we then looked at several possible improvements to the basic technique. Two main directions emerged for reducing the variance: periodogram smoothing and periodogram averaging. These approaches produced consistent and asymptotically unbiased estimates. The periodogram smoothing was obtained by applying the lag window to the autocorrelation estimate and then Fourier-transforming it. This method was due to Blackman and Tukey, and results of its mean and variance were given. The periodogram averaging was done by segmenting the data to obtain several records, followed by windowing to reduce spectral leakage, and finally by averaging their periodograms to reduce variance. This was the well-known Welch-Bartlett method, and the results of its statistical analysis were also given. Finally, implementations based on the DFT and MATLAB were given for both methods along with several examples to illustrate the performance of their estimates. These nonparametric methods were further extended to estimate the cross power spectrum, coherence functions, and transfer function.

Finally, we presented a newer nonparametric technique for auto power spectrum and cross power spectrum that was based on applying several data windows or tapers to the data followed by averaging of the resulting modified periodograms. The basic principle behind this method was that if the tapers are orthonormal and properly designed (to reduce leakage), then the resulting periodograms can be considered to be independent at each frequency and

hence their average would reduce the variance. Two orthogonal sets of data taper, namely, the Slepian and sinusoidal, were provided. The implementation using MATLAB was given, and examples were given to complete the chapter.

PROBLEMS

- 5.1** Let $x_c(t)$, $-\infty < t < \infty$, be a continuous-time signal with Fourier transform $X_c(F)$, $-\infty < F < \infty$, and let $x(n)$ be obtained by sampling $x_c(t)$ every T per sampling interval with its DTFT $X(e^{j\omega})$.

(a) Show that the DTFT $X(e^{j\omega})$ is given by

$$X(e^{j\omega}) = F_s \sum_{l=-\infty}^{\infty} X_c(fF_s - lF_s) \quad \omega = 2\pi f \quad F_s = \frac{1}{T}$$

(b) Let $\tilde{X}_p(k)$ be obtained by sampling $X(e^{j\omega})$ every $2\pi/N$ rad per sampling interval, that is,

$$\tilde{X}_p(k) = X(e^{j2\pi k/N}) = F_s \sum_{l=-\infty}^{\infty} X_c\left(\frac{kF_s}{N} - lF_s\right)$$

Then show that inverse DFT(\tilde{X}_p) is given by

$$x_p(n) \triangleq \text{IDFT}(\tilde{X}_p) = x_p(n) \triangleq \sum_{m=-\infty}^{\infty} x_c(nT - mNT)$$

- 5.2** MATLAB provides two functions to generate triangular windows, namely, `bartlett` and `triang`. These two functions actually generate two slightly different coefficients.

- (a) Use `bartlett` to generate $N = 11$, 31 , and 51 length windows $w_B(n)$, and plot their samples, using the `stem` function.
- (b) Compute the DTFTs $W_B(e^{j\omega})$, and plot their magnitudes over $[-\pi, \pi]$. Determine experimentally the width of the mainlobe as a function of N . Repeat part (a) using the `triang` function. How are the lengths and the mainlobe widths different in this case? Which window function is an appropriate one in terms of nonzero samples?
- (c) Determine the length of the `bartlett` window that has the same mainlobe width as that of a 51 -point rectangular window.

- 5.3** Sidelobes of the window transform contribute to the spectral leakage due to the frequency-domain convolution. One measure of this leakage is the maximum sidelobe height, which generally occurs at the first sidelobe for all windows except the Dolph-Chebyshev window.

- (a) For simple windows such as the rectangular, Hanning, or Hamming window, the maximum sidelobe height is independent of window length N . Choose $N = 11$, 31 , and 51 , and determine the maximum sidelobe height in decibels for the above windows.
- (b) For the Kaiser window, the maximum sidelobe height is controlled by the shape parameter β and is proportional to $\beta / \sinh \beta$. Using several values of β and N , verify the relationship between β and the maximum sidelobe height.
- (c) Determine the value of β that gives the maximum sidelobe height nearly the same as that of the Hamming window of the same length. Compare the mainlobe widths and the window coefficients of these two windows.
- (d) For the Dolph-Chebyshev window, all sidelobes have the same height A in decibels. For $A = 40$, 50 , and 60 dB, determine the 3-dB mainlobe widths for $N = 31$ length window.

- 5.4** Let $x(n)$ be given by

$$y(n) = \cos \omega_1 n + \cos (\omega_2 n + \phi) \quad \text{and} \quad x(n) = y(n)w(n)$$

where $w(n)$ is a length- N data window. The $|X(e^{j\omega})|^2$ is computed using MATLAB and is plotted over $[0, \pi]$.

- (a) Let $w(n)$ be a rectangular window. For $\omega_1 = 0.25\pi$ and $\omega_2 = 0.3\pi$, determine the minimum length N so that the two frequencies in the $|X(e^{j\omega})|^2$ plot are barely separable for any arbitrary $\phi \in [-\pi, \pi]$. (You may want to consider the worst possible value of ϕ or experiment, using several values of ϕ .)
- (b) Repeat part (a) for a Hamming window.
- (c) Repeat part (a) for a Blackman window.

- 5.5** In this problem we will prove that the autocorrelation matrix $\hat{\mathbf{R}}_x$ given in (5.2.3), in which the sample correlations are defined by (5.2.1), is a nonnegative definite matrix, that is,

$$\mathbf{x}^H \hat{\mathbf{R}}_x \mathbf{x} \geq 0 \quad \text{for every } \mathbf{x} \geq \mathbf{0}$$

- (a) Show that $\hat{\mathbf{R}}_x$ can be decomposed into the product $\mathbf{X}^H \mathbf{X}$, where \mathbf{X} is called a data matrix. Determine the form of \mathbf{X} .
- (b) Using the above decomposition, now prove that $\mathbf{x}^H \hat{\mathbf{R}}_x \mathbf{x} \geq 0$, for every $\mathbf{x} \geq \mathbf{0}$.

- 5.6** An alternative autocorrelation estimate $\check{r}_x(l)$ is given in (5.2.13) and is repeated below.

$$\check{r}_x(l) = \begin{cases} \frac{1}{N-l} \sum_{n=0}^{N-l-1} x(n+l)x^*(n) & 0 \leq l \leq L < N \\ \check{r}_x^*(-l) & -N < -L \leq l < 0 \\ 0 & \text{elsewhere} \end{cases}$$

- (a) Show that the mean of $\check{r}_x(l)$ is equal to $r_x(l)$ and an approximate expression for the variance of $\check{r}_x(l)$.
- (b) Show that the mean of the corresponding periodogram [that is, $\check{R}_x(e^{j\omega}) \triangleq \mathcal{F}[\check{r}_x(l)]$] is given by

$$E\{\check{R}_x(e^{j\omega})\} = \frac{1}{2\pi} \int_{-\pi}^{\pi} R_x(e^{j\varphi}) W_R(e^{j\omega-\varphi}) d\varphi$$

where $W_R(e^{j\omega})$ is the DTFT of the rectangular window and is sometimes called the *Dirichlet kernel*.

- 5.7** Consider the above unbiased autocorrelation estimator $\check{r}_x(l)$ of a zero-mean white Gaussian process with variance σ_x^2 .

- (a) Determine the variance of $\check{r}_x(l)$. Compute its limiting value as $l \rightarrow \infty$.
- (b) Repeat part (a) for the biased estimator $\hat{r}_x(l)$. Comment on any differences in the results.

- 5.8** Show that the autocorrelation matrix $\check{\mathbf{R}}_x$ formed by using $\check{r}_x(l)$ is not nonnegative definite, that is,

$$\mathbf{x}^H \check{\mathbf{R}}_x \mathbf{x} < 0 \quad \text{for some } \mathbf{x} \geq \mathbf{0}$$

- 5.9** In this problem, we will show that the periodogram $\hat{R}_x(e^{j\omega})$ can also be expressed as a DTFT of the autocorrelation estimate $\hat{r}_x(l)$ given in (5.2.1).

- (a) Let $v(n) = x(n)w_R(n)$, where $w_R(n)$ is a rectangular window of length N . Show that

$$\hat{r}_x(l) = \frac{1}{N} v(l) * v^*(-l) \tag{P.1}$$

- (b) Take the DTFT of (P.1) to show that

$$\hat{R}_x(e^{j\omega}) = \sum_{l=-N+1}^{N-1} \hat{r}_x(l) e^{-j\omega l}$$

- 5.10** Consider the following simple windows over $0 \leq n \leq N-1$: rectangular, Bartlett, Hanning, and Hamming.

- (a) Determine analytically the DTFT of each of the above windows.
- (b) Sketch the magnitude of these Fourier transforms for $N = 31$.
- (c) Verify your sketches by performing a numerical computation of the DTFT using MATLAB.

$$w_P(l) \triangleq \begin{cases} 1 - 6\left(\frac{l}{L}\right)^2 + 6\left(\frac{l}{L}\right)^3 & 0 \leq |l| \leq \frac{L}{2} \\ 2\left(1 - \frac{l}{L}\right)^3 & \frac{L}{2} < |l| < L \\ 0 & \text{elsewhere} \end{cases} \quad (\text{P.2})$$

(a) Show that its DTFT is given by

$$W_P(e^{j\omega}) \simeq \left[\frac{\sin(\omega L/4)}{\sin(\omega/4)} \right]^4 \geq 0 \quad (\text{P.3})$$

Hence using the Parzen window as a correlation window always produces nonnegative spectrum estimates.

- (b) Using MATLAB, compute and plot the time-domain window $w_P(l)$ and its frequency-domain response $W_P(e^{j\omega})$ for $L = 5, 10$, and 20 .
- (c) From the frequency-domain plots in part (b) experimentally determine the 3-dB mainlobe width $\Delta\omega$ as a function of L .

5.12 The variance reduction ratio of a correlation window $w_a(l)$ is defined as

$$\frac{\text{var}\{\hat{R}_x^{(\text{PS})}(e^{j\omega})\}}{\text{var}\{\hat{R}_x(e^{j\omega})\}} \simeq \frac{E_w}{N} \quad 0 < \omega < \pi$$

where $E_w = \frac{1}{2\pi} \int_{-\pi}^{\pi} W_a^2(e^{j\omega}) d\omega = \sum_{l=-(L-1)}^{L-1} w_a^2(l)$

- (a) Using MATLAB, compute and plot E_w as a function of L for the following windows: rectangular, Bartlett, Hanning, Hamming, and Parzen.
- (b) Using your computations above, show that for $L \gg 1$, the variance reduction ratio for each window is given by the formula in the following table.

Window name	Variance reduction factor
Rectangular	$2L/N$
Bartlett	$0.667L/N$
Hanning	$0.75L/N$
Hamming	$0.7948L/N$
Parzen	$0.539L/N$

5.13 For $L > 100$, the direct computation of $\hat{r}_x(l)$ using (5.3.49) is time-consuming; hence an indirect computation using the DFT can be more efficient. This computation is implemented by the following steps:

- Given the sequence $\{x(n)\}_{n=0}^{N-1}$, pad enough zeros to make it a $(2N - 1)$ -point sequence.
- Compute the N_{FFT} -point FFT of $x(n)$ to obtain $\tilde{X}(k)$, where N_{FFT} is equal to the next power-of-2 number that is greater than or equal to $2N - 1$.
- Compute $1/N|\tilde{X}(k)|^2$ to obtain $\tilde{R}(k)$.
- Compute the N_{FFT} -point IFFT of $\tilde{R}(k)$ to obtain $\hat{r}_x(l)$.

Develop a MATLAB function $rx = \text{autocfft}(x, L)$ which computes $\hat{r}_x(l)$, over $-L \leq l \leq L$. Compare this function with the `autoc` function discussed in the chapter in terms of the execution time for $L \geq 100$.

5.14 The Welch-Bartlett estimate $\hat{R}_x^{(\text{PA})}(k)$ is given by

$$\hat{R}_x^{(\text{PA})}(k) = \frac{1}{KL} \sum_{i=0}^{K-1} |X_i(k)|^2$$

If $x(n)$ is real-valued, then the sum in the above expression can be evaluated more efficiently. Let K be an even number. Then we will combine two real-valued sequences into one complex-valued sequence and compute one FFT, which will reduce the overall computations. Specifically, let

$$g_r(n) \triangleq x_{2r}(n) + jx_{2r+1}(n) \quad n = 0, 1, \dots, L-1, r = 0, 1, \dots, \frac{K}{2} - 1$$

Then the L -point DFT of $g_r(n)$ is given by

$$\tilde{G}_r(k) = \tilde{X}_{2r}(k) + j\tilde{X}_{2r+1}(k) \quad k = 0, 1, \dots, L-1, r = 0, 1, \dots, \frac{K}{2} - 1$$

(a) Show that

$$|\tilde{G}_r(k)|^2 + |\tilde{G}_r(L-k)|^2 = 2[|\tilde{X}_{2r}(k)|^2 + |\tilde{X}_{2r+1}(k)|^2] \quad k, r = 0, \dots, \frac{K}{2} - 1$$

(b) Determine the resulting expression for $\hat{R}_x^{(\text{PA})}(k)$ in terms of $\tilde{G}(k)$.

(c) What changes are necessary if K is an odd number? Provide detailed steps for this case.

5.15 Since $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ is a PSD estimate, one can determine autocorrelation estimate $\hat{r}_x^{(\text{PA})}(l)$ from Welch's method as

$$\hat{r}_x^{(\text{PA})}(l) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \hat{R}_x^{(\text{PA})}(e^{j\omega}) e^{j\omega l} d\omega \quad (\text{P.4})$$

Let $\tilde{\hat{R}}_x^{(\text{PA})}(k)$ be the samples of $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ according to

$$\tilde{\hat{R}}_x^{(\text{PA})}(k) \triangleq \hat{R}_x^{(\text{PA})}(e^{j2\pi k/N_{\text{FFT}}}) \quad 0 \leq k \leq N_{\text{FFT}} - 1$$

(a) Show that the IDFT $\tilde{\hat{r}}_x^{(\text{PA})}(l)$ of $\tilde{\hat{R}}_x^{(\text{PA})}(k)$ is an aliased version of the autocorrelation estimate $\hat{r}_x^{(\text{PA})}(l)$.

(b) If the length of the overlapping data segment in Welch's method is L , how should N_{FFT} be chosen to avoid aliasing in $\tilde{\hat{r}}_x^{(\text{PA})}(l)$?

5.16 Show that the coherence function $\mathcal{G}_{xy}^2(\omega)$ is invariant under linear transformation, that is, if $x_1(n) = h_1(n) * x(n)$ and $y_1(n) = h_2(n) * y(n)$, then

$$\mathcal{G}_{xy}^2(\omega) = \mathcal{G}_{x_1 y_2}^2(\omega)$$

5.17 Bartlett's method is a special case of Welch's method in which nonoverlapping sections of length L are used without windowing in the periodogram averaging operation.

(a) Show that the i th periodogram in this method can be expressed as

$$\hat{R}_{x,i}(e^{j\omega}) = \sum_{l=-L}^L \hat{r}_{x,i}(l) w_B(l) e^{-j\omega l} \quad (\text{P.5})$$

where $w_B(l)$ is a $(2L-1)$ -length Bartlett window.

(b) Let $\mathbf{u}(e^{j\omega}) \triangleq [1 \quad e^{j\omega} \quad \dots \quad e^{j(L-1)\omega}]^T$. Show that $\hat{R}_{x,i}(e^{j\omega})$ in (P.5) can be expressed as a quadratic product

$$\hat{R}_{x,i}(e^{j\omega}) = \frac{1}{L} \mathbf{u}^H(e^{j\omega}) \hat{\mathbf{R}}_{x,i} \mathbf{u}(e^{j\omega}) \quad (\text{P.6})$$

where $\hat{\mathbf{R}}_{x,i}$ is the autocorrelation matrix of $\hat{r}_{x,i}(l)$ values.

(c) Finally, show that the Bartlett estimate is given by

$$\hat{R}_x^{(\text{B})}(e^{j\omega}) = \frac{1}{KL} \sum_{i=1}^K \mathbf{u}^H(e^{j\omega}) \hat{\mathbf{R}}_{x,i} \mathbf{u}(e^{j\omega}) \quad (\text{P.7})$$

5.18 In this problem, we will explore a spectral estimation technique that uses combined data and correlation weighting (Carter and Nuttall 1980). In this technique, the following steps are performed:

- Given $\{x(n)\}_{n=0}^{N-1}$, compute the Welch-Bartlett estimate $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ by choosing the appropriate values of L and D .
- Compute the autocorrelation estimate $\hat{r}_x^{(\text{PA})}(l)$, $-L \leq l \leq L$, using the approach described in Problem 5.15.
- Window $\hat{r}_x^{(\text{PA})}(l)$, using a lag window $w_a(l)$ to obtain $\hat{r}_x^{(\text{CN})}(l) \triangleq \hat{r}_x^{(\text{PA})}(l)w_a(l)$.
- Finally, compute the DTFT of $\hat{r}_x^{(\text{CN})}(l)$ to obtain the new spectrum estimate $\hat{R}_x^{(\text{CN})}(e^{j\omega})$.
 - (a) Determine the bias of $\hat{R}_x^{(\text{CN})}(e^{j\omega})$.
 - (b) Comment on the effect of additional windowing on the variance and resolution of the estimate.
 - (c) Implement this technique in MATLAB, and compute spectral estimates of the process containing three sinusoids in white noise, which was discussed in the chapter. Experiment with various values of L and with different windows. Compare your results to those given for the Welch-Bartlett and Blackman-Tukey methods.

5.19 Explain why we use the scaling factor

$$\sum_{n=0}^{L-1} w^2(n)$$

which is the energy of the data window in the Welch-Bartlett method.

5.20 Consider the basic periodogram estimator $\hat{R}_x(e^{j\omega})$ at the zero frequency, that is, at $\omega = 0$.

- (a) Show that

$$\hat{R}_x(e^{j0}) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n)e^{j0} \right|^2 = \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n) \right|^2$$

- (b) If $x(n)$ is a real-valued, zero-mean white Gaussian process with variance σ_x^2 , determine the mean and variance of $\hat{R}_x(e^{j0})$.
 (c) Determine if $\hat{R}_x(e^{j0})$ is a consistent estimator by evaluating the variance as $N \rightarrow \infty$.

5.21 Consider Bartlett's method for estimating $R_x(e^{j0})$ using $L = 1$; that is, we use nonoverlapping segments of single samples. The periodogram of one sample $x(n)$ is simply $|x(n)|^2$. Thus we have

$$\hat{R}_x^{(\text{B})}(e^{j0}) = \frac{1}{N} \sum_{n=0}^{N-1} \hat{R}_{x,n}(e^{j0}) = \frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2$$

Again assume that $x(n)$ is a real-valued white Gaussian process with variance σ_x^2 .

- (a) Determine the mean and variance of $\hat{R}_x^{(\text{B})}(e^{j0})$.
 (b) Compare the above result with those in Problem 5.20. Comment on any differences.

5.22 One desirable property of lag or correlation windows is that their Fourier transforms are non-negative.

- (a) Formulate a procedure to generate a symmetric lag window of length $2L + 1$ with nonnegative Fourier transform.
 (b) Using the Hanning window as a prototype in the above procedure, determine and plot a 31-length lag window. Also plot its Fourier transform.

5.23 Consider the following random process

$$x(n) = \sum_{k=1}^4 A_k \sin(\omega_k n + \phi_k) + v(n)$$

$$\text{where } \begin{array}{llll} A_1 = 1 & A_2 = 0.5 & A_3 = 0.5 & A_4 = 0.25 \\ \omega_1 = 0.1\pi & \omega_2 = 0.6\pi & \omega_3 = 0.65\pi & \omega_4 = 0.8\pi \end{array}$$

and the phases $\{\phi_i\}_{i=1}^4$ are IID random variables uniformly distributed over $[-\pi, \pi]$. Generate 50 realizations of $x(n)$ for $0 \leq n \leq 256$. Let $v(n)$ be WN(0, 1).

- (a) Compute the Blackman-Tukey estimates for $L = 32, 64$, and 128 , using the Bartlett lag window. Plot your results, using overlay and averaged estimates. Comment on your plots.
- (b) Repeat part (a), using the Parzen window.
- (c) Provide a qualitative comparison between the above two sets of plots.

5.24 Consider the random process given in Problem 5.23.

- (a) Compute the Bartlett estimate, using $L = 16, 32$, and 64 . Plot your results, using overlay and averaged estimates. Comment on your plots.
- (b) Compute the Welch estimate, using 50 percent overlap, Hamming window, and $L = 16, 32$, and 64 . Plot your results, using overlay and averaged estimates. Comment on your plots.
- (c) Provide a qualitative comparison between the above two sets of plots.

5.25 Consider the random process given in Problem 5.23.

- (a) Compute the multitaper spectrum estimate, using $K = 3, 5$, and 7 Slepian tapers. Plot your results, using overlay and averaged estimates. Comment on your plots.
- (b) Make a qualitative comparison between the above plots and those obtained in Problems 5.23 and 5.24.

5.26 Generate 1000 samples of an AR(1) process using $a = -0.9$. Determine its theoretical PSD.

- (a) Determine and plot the periodogram of the process along with the true spectrum. Comment on the plots.
- (b) Compute the Blackman-Tukey estimates for $L = 10, 20, 50$, and 100 . Plot these estimates along with the true spectrum. Comment on your results.
- (c) Compute the Welch estimates for 50 percent overlap, Hamming window, and $L = 10, 20, 50$, and 100 . Plot these estimates along with the true spectrum. Comment on your results.

5.27 Generate 1000 samples of an AR(1) process using $a = 0.9$. Determine its theoretical PSD.

- (a) Determine and plot the periodogram of the process along with the true spectrum. Comment on the plots.
- (b) Compute the Blackman-Tukey estimates for $L = 10, 20, 50$, and 100 . Plot these estimates along with the true spectrum. Comment on your results.
- (c) Compute the Welch estimates for 50 percent overlap, Hamming window, and $L = 10, 20, 50$, and 100 . Plot these estimates along with the true spectrum. Comment on your results.

5.28 Multitaper estimation technique requires a properly designed orthonormal set of tapers for the desired performance. One set discussed in the chapter was that of harmonically related sinusoids given in (5.5.8).

- (a) Design a MATLAB function `[tapers] = sine_tapers(N, K)` that generates $K < N$ sinusoidal tapers of length N .
- (b) Using the above function, compute and plot the Fourier transform magnitudes of the first 5 tapers of length 51.

5.29 Design a MATLAB function `Pxx = psd_sinetaper(x, K)` that determines the multitaper estimates using the sine tapers.

- (a) Apply the function `psd_sinetaper` to the AR(1) process given in Problem 5.26, and compare its performance.
- (b) Apply the function `psd_sinetaper` to the AR(1) process given in Problem 5.27, and compare its performance.

Optimum Linear Filters

In this chapter, we present the theory and application of optimum linear filters and predictors. We concentrate on linear filters that are optimum in the sense of minimizing the mean square error (MSE). The minimum MSE (MMSE) criterion leads to a theory of linear filtering that is elegant and simple, involves only second-order statistics, and is useful in many practical applications. The optimum filter designed for a given set of second-order moments can be used for any realizations of stochastic processes with the same moments.

We start with the general theory of linear MMSE estimators and their computation, using the triangular decomposition of Hermitian positive definite matrices. Then we apply the general theory to the design of optimum FIR filters and linear predictors for both nonstationary and stationary processes (Wiener filters). We continue with the design of nonparametric (impulse response) and parametric (pole-zero) optimum IIR filters and predictors for stationary processes. Then we present the design of optimum filters for inverse system modeling, blind deconvolution, and their application to equalization of data communication channels. We conclude with a concise introduction to optimum matched filters and eigenfilters that maximize the output SNR. These signal processing methods find extensive applications in digital communication, radar, and sonar systems.

6.1 OPTIMUM SIGNAL ESTIMATION

As we discussed in Chapter 1, the solution of many problems of practical interest depends on the ability to accurately estimate the value $y(n)$ of a signal (*desired response*) by using a set of values (*observations* or *data*) from another related signal or signals. Successful estimation is possible if there is significant statistical dependence or correlation between the signals involved in the particular application. For example, in the linear prediction problem we use the M past samples $x(n - 1), x(n - 2), \dots, x(n - M)$ of a signal to estimate the current sample $x(n)$. The echo canceler in Figure 1.17 uses the transmitted signal to form a replica of the received echo. The radar signal processor in Figure 1.27 uses the signals $x_k(n)$ for $1 \leq k \leq M$ received by the linear antenna array to estimate the value of the signal $y(n)$ received from the direction of interest. Although the signals in these and other similar applications have different physical origins, the mathematical formulations of the underlying signal processing problems are very similar.

In array signal processing, the data are obtained by using M different sensors. The situation is simpler for filtering applications, because the data are obtained by delaying a single discrete-time signal; that is, we have $x_k(n) = x(n + 1 - k)$, $1 \leq k \leq M$ (see Figure 6.1). Further simplifications are possible in linear prediction, where both the desired response and the data are time samples of the same signal, for example, $y(n) = x(n)$ and

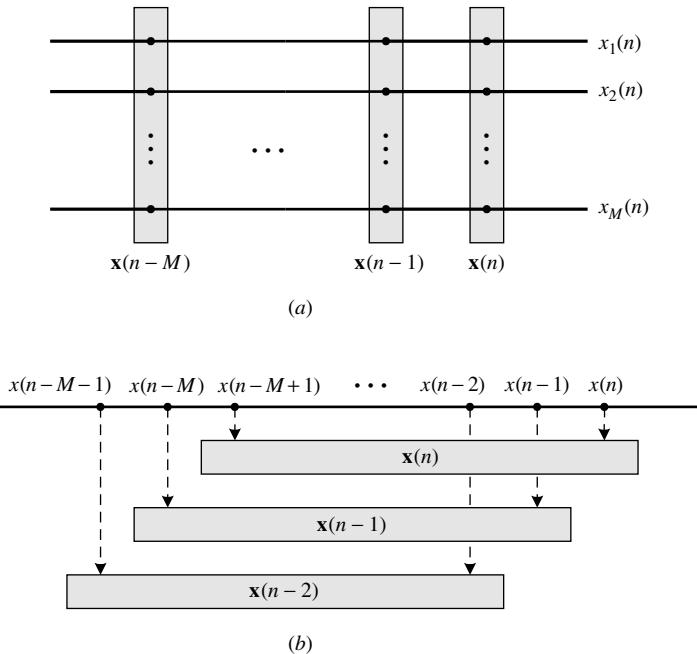
**FIGURE 6.1**

Illustration of the data vectors for (a) array processing (multiple sensors) and (b) FIR filtering or prediction (single sensor) applications.

$x_k(n) = x(n - k)$, $1 \leq k \leq M$. As a result, the design and implementation of optimum filters and predictors are simpler than those for an optimum array processor.

Since array processing problems are the most general ones, we will formulate and solve the following estimation problem: Given a set of data $x_k(n)$ for $1 \leq k \leq M$, determine an estimate $\hat{y}(n)$, of the desired response $y(n)$, using the rule (*estimator*)

$$\hat{y}(n) \triangleq H\{x_k(n), 1 \leq k \leq M\} \quad (6.1.1)$$

which, in general, is a nonlinear function of the data. When $x_k(n) = x(n + 1 - k)$, the estimator takes on the form of a discrete-time filter that can be linear or nonlinear, time-invariant or time-varying, and with a finite- or infinite-duration impulse response. Linear filters can be implemented using any direct, parallel, cascade, or lattice-ladder structure (see Section 2.5 and Proakis and Manolakis 1996).

The difference between the estimated response $\hat{y}(n)$ and the desired response $y(n)$, that is,

$$e(n) \triangleq y(n) - \hat{y}(n) \quad (6.1.2)$$

is known as the *error signal*. We want to find an estimator whose output approximates the desired response as closely as possible according to a certain performance criterion. We use the term *optimum estimator* or *optimum signal processor* to refer to such an estimator. We stress that *optimum* is not used as a synonym for *best*; it simply means the best under the given set of assumptions and conditions. If either the criterion of performance or the assumptions about the statistics of the processed signals change, the corresponding optimum filter will change as well. Therefore, an optimum estimator designed for a certain performance metric and set of assumptions may perform poorly according to some other criterion or if the actual statistics of the processed signals differ from the ones used in the design. For this reason, the sensitivity of the performance to deviations from the assumed statistics is very important in practical applications of optimum estimators.

Therefore, the design of an optimum estimator involves the following steps:

1. Selection of a computational structure with well-defined parameters for the implementation of the estimator.
2. Selection of a criterion of performance or cost function that measures the performance of the estimator under some assumptions about the statistical properties of the signals to be processed.
3. Optimization of the performance criterion to determine the parameters of the optimum estimator.
4. Evaluation of the optimum value of the performance criterion to determine whether the optimum estimator satisfies the design specifications.

263

SECTION 6.1
Optimum Signal Estimation

Many practical applications (e.g., speech, audio, and image coding) require subjective criteria that are difficult to express mathematically. Thus, we focus on criteria of performance that (1) only depend on the estimation error $e(n)$, (2) provide a sufficient measure of the user satisfaction, and (3) lead to a mathematically tractable problem. We generally select a criterion of performance by compromising between these objectives.

Since, in most applications, negative and positive errors are equally harmful, we should choose a criterion that weights both negative and positive errors equally. Choices that satisfy this requirement include the absolute value of the error $|e(n)|$, or the squared error $|e(n)|^2$, or some other power of $|e(n)|$ (see Figure 6.2). The emphasis put on different values of the error is a key factor when we choose a criterion of performance. For example, the squared-error criterion emphasizes the effect of large errors much more than the absolute error criterion. Thus, the squared-error criterion is more sensitive to outliers (occasional large values) than the absolute error criterion is.

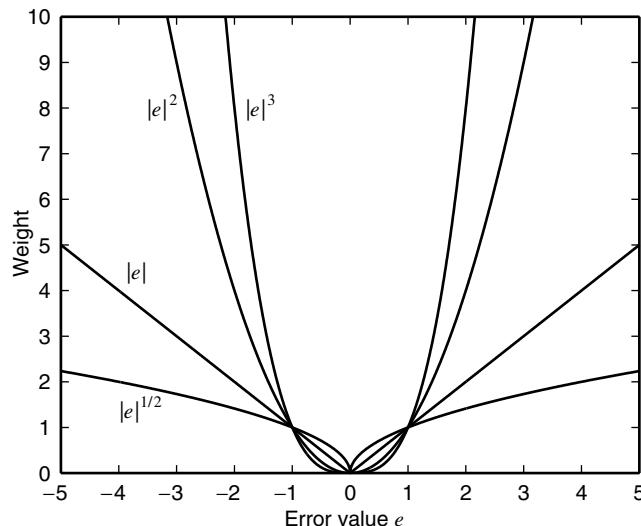


FIGURE 6.2

Graphical illustration of various error-weighting functions.

To develop a mathematical theory that will help to design and analyze the performance of optimum estimators, we assume that the desired response and the data are realizations of stochastic processes. Furthermore, although in practice the estimator operates on specific realizations of the input and desired response signals, we wish to design an estimator with good performance across all members of the ensemble, that is, an estimator that “works

well on average.” Since, at any fixed time n , the quantities $y(n)$, $x_k(n)$ for $1 \leq k \leq M$, and $e(n)$ are random variables, we should choose a criterion that involves the ensemble or time averaging of some function of $|e(n)|$. Here is a short list of potential criteria of performance:

1. The mean square error criterion

$$P(n) \triangleq E\{|e(n)|^2\} \quad (6.1.3)$$

which leads, in general, to a nonlinear optimum estimator.

2. The mean α th-order error criterion $E\{|e(n)|^\alpha\}$, $\alpha \neq 2$. Using a lower- or higher-order moment of the absolute error is more appropriate for certain types of non-Gaussian statistics than the MSE (Stuck 1978).
3. The sum of squared errors (SSE)

$$E(n_i, n_f) \triangleq \sum_{n=n_i}^{n_f} |e(n)|^2 \quad (6.1.4)$$

which, if it is divided by $n_f - n_i + 1$, provides an estimate of the MSE.

The MSE criterion (6.1.3) and the SSE criterion (6.1.4) are the most widely used because they (1) are mathematically tractable, (2) lead to the design of useful systems for practical applications, and (3) can serve as a yardstick for evaluating estimators designed with other criteria (e.g., signal-to-noise ratio, maximum likelihood). In most practical applications, we use linear estimators, which further simplifies their design and evaluation.

Mean square estimation is a rather vast field that was originally developed by Gauss in the nineteenth century. The current theories of estimation and optimum filtering started with the pioneering work of Wiener and Kolmogorov that was later extended by Kalman, Bucy, and others. Some interesting historical reviews are given in Kailath (1974) and Sorenson (1970).

6.2 LINEAR MEAN SQUARE ERROR ESTIMATION

In this section, we develop the theory of linear MSE estimation. We concentrate on linear estimators for various reasons, including mathematical simplicity and ease of implementation. The problem can be stated as follows:

Design an estimator that provides an estimate $\hat{y}(n)$ of the desired response $y(n)$ using a *linear* combination of the data $x_k(n)$ for $1 \leq k \leq M$, such that the MSE $E\{|y(n) - \hat{y}(n)|^2\}$ is minimized.

More specifically, the linear estimator is defined by

$$\hat{y}(n) \triangleq \sum_{k=1}^M c_k^*(n)x_k(n) \quad (6.2.1)$$

and the goal is to determine the coefficients $c_k(n)$ for $1 \leq k \leq M$ such that the MSE (6.1.3) is minimized. In general, a new set of optimum coefficients should be computed for each time instant n . Since we assume that the desired response and the data are realizations of stochastic processes, the quantities $y(n)$, $x_1(n), \dots, x_M(n)$ are random variables at any fixed time n . For convenience, we formulate and solve the estimation problem at a fixed time instant n . Thus, we drop the time index n and restate the problem as follows:

Estimate a random variable y (the desired response) from a set of related random variables x_1, x_2, \dots, x_M (data) using the linear estimator

$$\hat{y} \triangleq \sum_{k=1}^M c_k^* x_k = \mathbf{c}^H \mathbf{x} \quad (6.2.2)$$

where $\mathbf{x} = [x_1 \ x_2 \ \cdots \ x_M]^T$ (6.2.3)

is the *input data vector* and

$$\mathbf{c} = [c_1 \ c_2 \ \cdots \ c_M]^T \quad (6.2.4)$$

is the *parameter or coefficient vector* of the estimator.

Unless otherwise stated, *all* random variables are assumed to have *zero-mean* values. The number M of data components used is called the *order* of the estimator. The linear estimator (6.2.2) is represented graphically as shown in Figure 6.3 and involves a computational structure known as the *linear combiner*. The MSE

$$P \triangleq E\{|e|^2\} \quad (6.2.5)$$

where

$$e \triangleq y - \hat{y} \quad (6.2.6)$$

is a function of the parameters c_k . Minimization of (6.2.5) with respect to parameters c_k leads to a linear estimator \mathbf{c}_o that is optimum in the MSE sense. The parameter vector \mathbf{c}_o is known as the *linear MMSE (LMMSE) estimator* and \hat{y}_o as the LMMSE estimate.

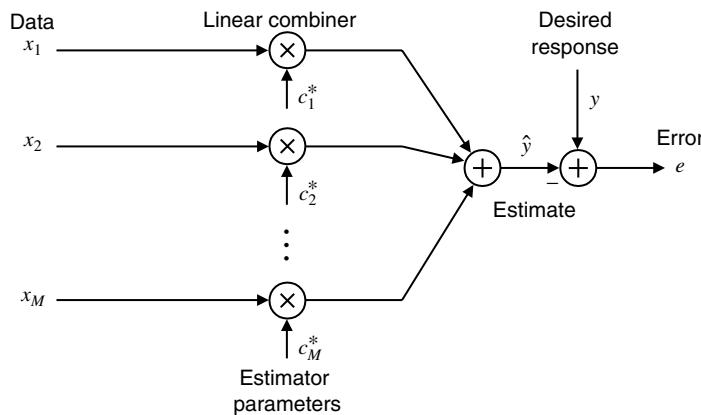


FIGURE 6.3

Block diagram representation of the linear estimator.

6.2.1 Error Performance Surface

To determine the linear MMSE estimator, we seek the value of the parameter vector \mathbf{c} that minimizes the function (6.2.5). To this end, we want to express the MSE as a function of the parameter vector \mathbf{c} and to understand the nature of this dependence.

By using (6.2.5), (6.2.6), (6.2.2), and the linearity property of the expectation operator, the MSE is given by

$$\begin{aligned} P(\mathbf{c}) &= E\{|e|^2\} = E\{(y - \mathbf{c}^H \mathbf{x})(y^* - \mathbf{x}^H \mathbf{c})\} \\ &= E\{|y|^2\} - \mathbf{c}^H E\{\mathbf{x}y^*\} - E\{y\mathbf{x}^H\}\mathbf{c} + \mathbf{c}^H E\{\mathbf{x}\mathbf{x}^H\}\mathbf{c} \end{aligned}$$

or more compactly,

$$P(\mathbf{c}) = P_y - \mathbf{c}^H \mathbf{d} - \mathbf{d}^H \mathbf{c} + \mathbf{c}^H \mathbf{R} \mathbf{c} \quad (6.2.7)$$

where

$$P_y \triangleq E\{|y|^2\} \quad (6.2.8)$$

is the power of the desired response,

$$\mathbf{d} \triangleq E\{\mathbf{x}y^*\} \quad (6.2.9)$$

is the cross-correlation vector between the data vector \mathbf{x} and the desired response y , and

$$\mathbf{R} \triangleq E\{\mathbf{x}\mathbf{x}^H\} \quad (6.2.10)$$

is the correlation matrix of the data vector \mathbf{x} . The matrix \mathbf{R} is guaranteed to be Hermitian and nonnegative definite (see Section 3.4.4).

The function $P(\mathbf{c})$ is known as the *error performance surface* of the estimator. Equation (6.2.7) shows that the MSE $P(\mathbf{c})$ (1) depends *only* on the second-order moments of the desired response and the data and (2) is a quadratic function of the estimator coefficients and represents an $(M + 1)$ -dimensional surface with M degrees of freedom. We will see that if \mathbf{R} is positive definite, then the quadratic function $P(\mathbf{c})$ is bowl-shaped and has a unique minimum that corresponds to the optimum parameters. The next example illustrates this fact for the second-order case.

EXAMPLE 6.2.1. If $M = 2$ and the random variables y, x_1 , and x_2 are real-valued, the MSE is

$$P(c_1, c_2) = P_y - 2d_1c_1 - 2d_2c_2 + r_{11}c_1^2 + 2r_{12}c_1c_2 + r_{22}c_2^2$$

because $r_{12} = r_{21}$. And $P(c_1, c_2)$ is a second-order function of coefficients c_1 and c_2 , and Figure 6.4 shows two plots of the function $P(c_1, c_2)$ that are quite different in appearance. The surface in Figure 6.4(a) looks like a bowl and has a unique extremum that is a minimum. The values for the error surface parameters are $P_y = 0.5, r_{11} = r_{22} = 4.5, r_{12} = r_{21} = -0.1545, d_1 = -0.5$, and $d_2 = -0.1545$. On the other hand, in Figure 6.4(b), we have a saddle point that is neither a

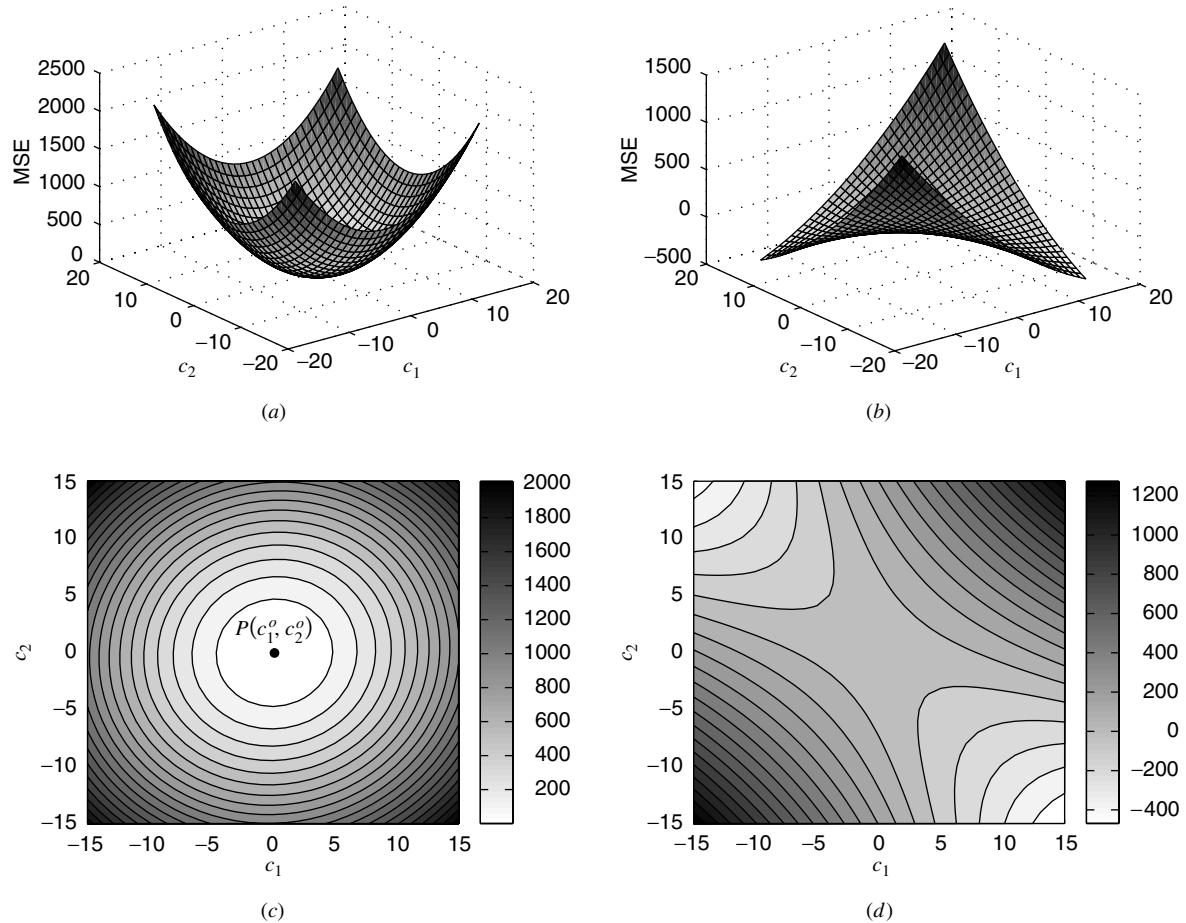


FIGURE 6.4

Representative surface and contour plots for positive definite and negative definite quadratic error performance surfaces.

minimum nor a maximum (here only the matrix elements have changed to $r_{11} = r_{22} = 1, r_{12} = r_{21} = 2$). If we cut the surfaces with planes parallel to the (c_1, c_2) plane, we obtain *contours* of constant MSE that are shown in Figure 6.4(c) and (d). In conclusion, the error performance surface is bowl-shaped and has a unique minimum only if the matrix \mathbf{R} is positive definite (the determinants of the two matrices are 20.23 and -3 , respectively). Only in this case can we obtain an estimator that minimizes the MSE, and the contours are concentric ellipses whose center corresponds to the optimum estimator. The bottom of the bowl is determined by setting the partial derivatives with respect to the unknown parameters to zero, that is,

$$\begin{aligned}\frac{\partial P(c_1, c_2)}{\partial c_1} &= 0 \quad \text{which results in} \quad r_{11}c_1^o + r_{12}c_2^o = d_1 \\ \frac{\partial P(c_1, c_2)}{\partial c_2} &= 0 \quad \text{which results in} \quad r_{12}c_1^o + r_{22}c_2^o = d_2\end{aligned}$$

This is a linear system of two equations with two unknowns whose solution provides the coefficients c_1^o and c_2^o that minimize the MSE function $P(c_1, c_2)$.

When the optimum filter is specified by a rational system function, the error performance surface may be nonquadratic. This is illustrated in the following example.

EXAMPLE 6.2.2. Suppose that we wish to estimate the real-valued output $y(n)$ of the “unknown” system (see Figure 6.5)

$$G(z) = \frac{0.05 - 0.4z^{-1}}{1 - 1.1314z^{-1} + 0.25z^{-2}}$$

using the pole-zero filter

$$H(z) = \frac{b}{1 - az^{-1}}$$

by minimizing the MSE $E\{e^2(n)\}$ (Johnson and Larimore 1977). The input signal $x(n)$ is white noise with zero mean and variance σ_x^2 . The MSE is given by

$$E\{e^2(n)\} = E\{[y(n) - \hat{y}(n)]^2\} = E\{y^2(n)\} - 2E\{y(n)\hat{y}(n)\} + E\{\hat{y}^2(n)\}$$

and is a function of parameters b and a . Since the impulse response $h(n) = ba^n u(n)$ of the optimum filter has infinite duration, we cannot use (6.2.7) to compute $E\{e^2(n)\}$ and to plot the error surface. The three components of $E\{e^2(n)\}$ can be evaluated as follows, using Parseval’s theorem: The power of the desired response

$$E\{y^2(n)\} = \sigma_x^2 \sum_{n=0}^{\infty} g^2(n) = \frac{\sigma_x^2}{2\pi j} \oint G(z)G(z^{-1})z^{-1} dz \triangleq \sigma_x^2 \sigma_g^2$$

is constant and can be computed either numerically by using the first M “nonzero” samples of $g(n)$ or analytically by evaluating the integral using the residue theorem. The power of the optimum filter output is

$$E\{\hat{y}^2(n)\} = E\{x^2(n)\} \sum_{n=0}^{\infty} h^2(n) = \frac{\sigma_x^2}{2\pi j} \oint H(z)H(z^{-1})z^{-1} dz = \sigma_x^2 \frac{b^2}{1 - a^2}$$

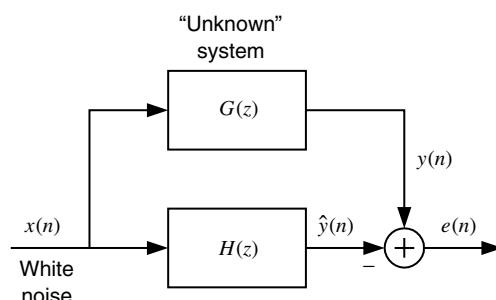


FIGURE 6.5

Identification of an “unknown” system using an optimum filter.

which is a function of parameters b and a . The middle term is

$$\begin{aligned} E\{y(n)\hat{y}(n)\} &= E\left\{\sum_{k=0}^{\infty} g(k)x(n-k) \sum_{m=0}^{\infty} h(m)x(n-m)\right\} \\ &= \sigma_x^2 \sum_{k=0}^{\infty} g(k)h(k) = \frac{\sigma_x^2}{2\pi j} \oint G(z)H(z^{-1})z^{-1} dz = bG(z)|_{z^{-1}=a} \end{aligned}$$

because $E\{x(n-k)x(n-m)\} = \sigma_x^2 \delta(m-k)$. For convenience we compute the normalized MSE

$$P(b, a) \triangleq \frac{E\{e^2(n)\}}{\sigma_g^2} = \sigma_x^2 - \frac{2b}{\sigma_g^2} G(z) \Big|_{z^{-1}=a} + \frac{\sigma_x^2}{\sigma_g^2} \frac{b^2}{1-a^2}$$

whose surface and contour plots are shown in Figure 6.6. We note that the resulting error performance surface is bimodal with a global minimum $P = 0.277$ at $(b, a) = (-0.311, 0.906)$ and a local minimum $P = 0.976$ at $(b, a) = (0.114, -0.519)$. As a result, the determination of the optimum filter requires the use of nonlinear optimization techniques with all associated drawbacks.

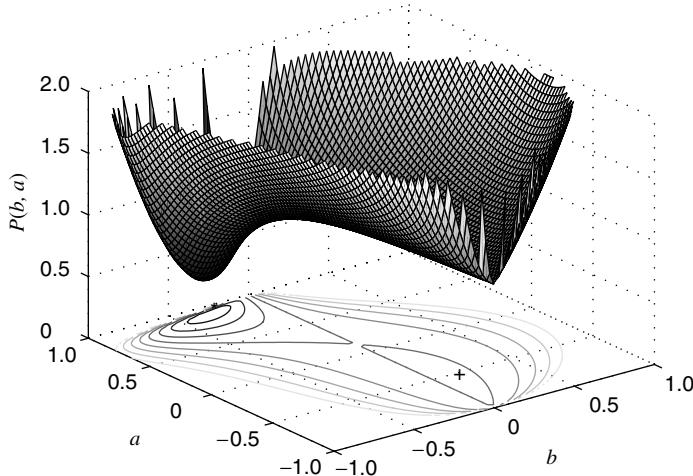


FIGURE 6.6

Illustration of the nonquadratic form of the error performance surface of a pole-zero optimum filter specified by the coefficients of its difference equation.

6.2.2 Derivation of the Linear MMSE Estimator

The approach in Example 6.2.1 can be generalized to obtain the necessary and sufficient conditions that determine the linear MMSE estimator.[†] Here, we present a simpler matrix-based approach that is sufficient for the scope of this chapter.

We first notice that we can put (6.2.7) into the form of a “perfect square” as

$$P(\mathbf{c}) = P_y - \mathbf{d}^H \mathbf{R}^{-1} \mathbf{d} + (\mathbf{R}\mathbf{c} - \mathbf{d})^H \mathbf{R}^{-1} (\mathbf{R}\mathbf{c} - \mathbf{d}) \quad (6.2.11)$$

where only the third term depends on \mathbf{c} . If \mathbf{R} is positive definite, the inverse matrix \mathbf{R}^{-1} exists

[†]For complex-valued random variables, there are some complications that should be taken into account because $|e|^2$ is not an *analytic* function. This topic is discussed in Appendix B.

and is positive definite; that is, $\mathbf{z}^H \mathbf{R}^{-1} \mathbf{z} > 0$ for all $\mathbf{z} \neq \mathbf{0}$. Therefore, if \mathbf{R} is positive definite, the term $\mathbf{d}^H \mathbf{R}^{-1} \mathbf{d} > 0$ decreases the cost function by an amount determined exclusively by the second-order moments. In contrast, the term $(\mathbf{R}\mathbf{c} - \mathbf{d})^H \mathbf{R}^{-1} (\mathbf{R}\mathbf{c} - \mathbf{d}) > 0$ increases the cost function depending on the choice of the estimator parameters. Thus, the best estimator is obtained by setting $\mathbf{R}\mathbf{c} - \mathbf{d} = \mathbf{0}$.

Therefore, the *necessary and sufficient* conditions that determine the linear MMSE estimator \mathbf{c}_o are

$$\mathbf{R}\mathbf{c}_o = \mathbf{d} \quad (6.2.12)$$

and

$$\mathbf{R} \text{ is positive definite} \quad (6.2.13)$$

In greater detail, (6.2.12) can be written as

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1M} \\ r_{21} & r_{22} & \cdots & r_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ r_{M1} & r_{M2} & \cdots & r_{MM} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_M \end{bmatrix} \quad (6.2.14)$$

where

$$r_{ij} \triangleq E\{x_i x_j^*\} = r_{ji}^* \quad (6.2.15)$$

and

$$d_i \triangleq E\{x_i y^*\} \quad (6.2.16)$$

and are known as the set of *normal equations*. The invertibility of the correlation matrix \mathbf{R} —and hence the existence of the optimum estimator—is guaranteed if \mathbf{R} is positive definite. In theory, \mathbf{R} is guaranteed to be nonnegative definite, but in physical applications it will almost always be positive definite. The normal equations can be solved by using any general-purpose routine for a set of linear equations.

Using (6.2.11) and (6.2.12), we find that the MMSE P_o is

$$P_o = P_y - \mathbf{d}^H \mathbf{R}^{-1} \mathbf{d} = P_y - \mathbf{d}^H \mathbf{c}_o \quad (6.2.17)$$

where we can easily show that the term $\mathbf{d}^H \mathbf{c}_o$ is equal to $E\{|\hat{y}_o|^2\}$, the power of the optimum estimate. If \mathbf{x} and y are uncorrelated ($\mathbf{d} = \mathbf{0}$), we have the worst situation ($P_o = P_y$) because there is no linear estimator that can reduce the MSE. If $\mathbf{d} \neq \mathbf{0}$, there is always going to be some reduction in the MSE owing to the correlation between the data vector \mathbf{x} and the desired response y , assuming that \mathbf{R} is positive definite. The best situation corresponds to $\hat{y} = y$, which gives $P_o = 0$. Thus, for comparison purposes, we use the *normalized MSE*

$$\mathcal{E} \triangleq \frac{P_o}{P_y} = 1 - \frac{P_{\hat{y}_o}}{P_y} \quad (6.2.18)$$

because it is bounded between 0 and 1, that is,

$$0 \leq \mathcal{E} \leq 1 \quad (6.2.19)$$

If $\tilde{\mathbf{c}}$ is the deviation from the optimum vector \mathbf{c}_o , that is, if $\mathbf{c} = \mathbf{c}_o + \tilde{\mathbf{c}}$, then substituting into (6.2.11) and using (6.2.17), we obtain

$$P(\mathbf{c}_o + \tilde{\mathbf{c}}) = P(\mathbf{c}_o) + \tilde{\mathbf{c}}^H \mathbf{R} \tilde{\mathbf{c}} \quad (6.2.20)$$

Equation (6.2.20) shows that if \mathbf{R} is positive definite, any deviation $\tilde{\mathbf{c}}$ from the optimum vector \mathbf{c}_o increases the MSE by an amount $\tilde{\mathbf{c}}^H \mathbf{R} \tilde{\mathbf{c}} > 0$, which is known as the *excess MSE*, that is,

$$\text{Excess MSE} \triangleq P(\mathbf{c}_o + \tilde{\mathbf{c}}) - P(\mathbf{c}_o) = \tilde{\mathbf{c}}^H \mathbf{R} \tilde{\mathbf{c}} \quad (6.2.21)$$

We emphasize that the excess MSE depends only on the input correlation matrix and not on the desired response. This fact has important implications because any deviation from the optimum can be detected by monitoring the MSE.

For nonzero-mean random variables, we use the estimator $\hat{y} \triangleq c_0 + \mathbf{c}^H \mathbf{x}$. The elements of \mathbf{R} and \mathbf{d} are replaced by the corresponding covariances and $c_0 = E\{y\} - \mathbf{c}^H E\{\mathbf{x}\}$ (see Problem 6.1). In the sequel, unless otherwise explicitly stated, we assume that all random variables have zero mean or have been reduced to zero mean by replacing y by $y - E\{y\}$ and \mathbf{x} by $\mathbf{x} - E\{\mathbf{x}\}$.

6.2.3 Principal-Component Analysis of the Optimum Linear Estimator

The properties of optimum linear estimators and their error performance surfaces depend on the correlation matrix \mathbf{R} . We can learn a lot about the nature of the optimum estimator if we express \mathbf{R} in terms of its eigenvalues and eigenvectors. Indeed, from Section 3.4.4, we have

$$\mathbf{R} = \mathbf{Q} \Lambda \mathbf{Q}^H = \sum_{i=1}^M \lambda_i \mathbf{q}_i \mathbf{q}_i^H \quad \text{and} \quad \Lambda = \mathbf{Q}^H \mathbf{R} \mathbf{Q} \quad (6.2.22)$$

where $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_M\}$ (6.2.23)

are the eigenvalues of \mathbf{R} , assumed to be distinct, and

$$\mathbf{Q} = [\mathbf{q}_1 \ \mathbf{q}_2 \ \cdots \ \mathbf{q}_M] \quad (6.2.24)$$

are the eigenvectors of \mathbf{R} . The *modal* matrix \mathbf{Q} is unitary, that is,

$$\mathbf{Q}^H \mathbf{Q} = \mathbf{I} \quad (6.2.25)$$

which implies that $\mathbf{Q}^{-1} = \mathbf{Q}^H$. The relationship (6.2.22) between \mathbf{R} and Λ is known as a *similarity transformation*.

In general, the multiplication of a vector by a matrix changes both the length and the direction of the vector. We define a coordinate transformation of the optimum parameter vector by

$$\mathbf{c}'_o \triangleq \mathbf{Q}^H \mathbf{c}_o \quad \text{or} \quad \mathbf{c}_o \triangleq \mathbf{Q} \mathbf{c}'_o \quad (6.2.26)$$

Since $\|\mathbf{c}_o\| = (\mathbf{Q} \mathbf{c}'_o)^H \mathbf{Q} \mathbf{c}'_o = \mathbf{c}'_o^H \mathbf{Q}^H \mathbf{Q} \mathbf{c}'_o = \|\mathbf{c}'_o\|$ (6.2.27)

the transformation (6.2.26) changes the direction of the transformed vector but not its length.

If we substitute (6.2.22) into the normal equations (6.2.12), we obtain

$$\mathbf{Q} \Lambda \mathbf{Q}^H \mathbf{c}_o = \mathbf{d} \quad \text{or} \quad \Lambda \mathbf{Q}^H \mathbf{c}_o = \mathbf{Q}^H \mathbf{d}$$

which results in

$$\Lambda \mathbf{c}'_o = \mathbf{d}' \quad (6.2.28)$$

where $\mathbf{d}' \triangleq \mathbf{Q}^H \mathbf{d}$ or $\mathbf{d} \triangleq \mathbf{Q} \mathbf{d}'$ (6.2.29)

is the transformed “decoupled” cross-correlation vector.

Because Λ is diagonal, the set of M equations (6.2.28) can be written as

$$\lambda_i c'_{o,i} = d'_i \quad 1 \leq i \leq M \quad (6.2.30)$$

where $c'_{o,i}$ and d'_i are the components of \mathbf{c}'_o and \mathbf{d}' , respectively. This is an uncoupled set of M first-order equations. If $\lambda_i \neq 0$, then

$$c'_{o,i} = \frac{d'_i}{\lambda_i} \quad 1 \leq i \leq M \quad (6.2.31)$$

and if $\lambda_i = 0$, the value of $c'_{o,i}$ is indeterminate.

The MMSE becomes

271

SECTION 6.2

Linear Mean Square Error
Estimation

$$\begin{aligned} P_o &= P_y - \mathbf{d}^H \mathbf{c}_o \\ &= P_y - (\mathbf{Q}\mathbf{d}')^H \mathbf{Q}\mathbf{c}'_o = P_y - \mathbf{d}'^H \mathbf{c}'_o \\ &= P_y - \sum_{i=1}^M d_i'^* c'_{o,i} = P_y - \sum_{i=1}^M \frac{|d'_i|^2}{\lambda_i} \end{aligned} \quad (6.2.32)$$

which shows how the eigenvalues and the decoupled cross-correlations affect the performance of the optimum filter. The advantage of (6.2.31) and (6.2.32) is that we can study the behavior of each parameter of the optimum estimator independently of all the remaining ones.

To appreciate the significance of the principal-component transformation, we will discuss the error surface of a second-order estimator. However, all the results can be easily generalized to estimators of order M , whose error performance surface exists in a space of $M + 1$ dimensions. Figure 6.7 shows the contours of constant MSE for a positive definite, second-order error surface. The contours are concentric ellipses centered at the tip of the optimum vector \mathbf{c}_o . We define a new coordinate system with origin at \mathbf{c}_o and axes determined by the major axis $\tilde{\mathbf{v}}_1$ and the minor axis $\tilde{\mathbf{v}}_2$ of the ellipses. The two axes are orthogonal, and the resulting system is known as the *principal coordinate system*. The transformation from the “old” system to the “new” system is done in two steps:

$$\begin{aligned} \text{Translation: } \tilde{\mathbf{c}} &= \mathbf{c} - \mathbf{c}_o \\ \text{Rotation: } \tilde{\mathbf{v}} &= \mathbf{Q}^H \tilde{\mathbf{c}} \end{aligned} \quad (6.2.33)$$

where the rotation changes the axes of the space to match the axes of the ellipsoid. The excess MSE (6.2.21) becomes

$$\Delta P(\tilde{\mathbf{v}}) = \tilde{\mathbf{c}}^H \mathbf{R} \tilde{\mathbf{c}} = \tilde{\mathbf{c}}^H \mathbf{Q} \Lambda \mathbf{Q}^H \tilde{\mathbf{c}} = \tilde{\mathbf{v}}^H \Lambda \tilde{\mathbf{v}} = \sum_{i=1}^M \lambda_i |\tilde{v}_i|^2 \quad (6.2.34)$$

which shows that the penalty paid for the deviation of a parameter from its optimum value is proportional to the corresponding eigenvalue. Clearly, changes in uncoupled parameters (which correspond to $\lambda_i = 0$) do not affect the excess MSE.

Using (6.2.22), we have

$$\mathbf{c}_o = \mathbf{R}^{-1} \mathbf{d} = \mathbf{Q} \Lambda^{-1} \mathbf{Q}^H \mathbf{d} = \sum_{i=1}^M \frac{\mathbf{q}_i^H \mathbf{d}}{\lambda_i} \mathbf{q}_i = \sum_{i=1}^M \frac{d'_i}{\lambda_i} \mathbf{q}_i \quad (6.2.35)$$

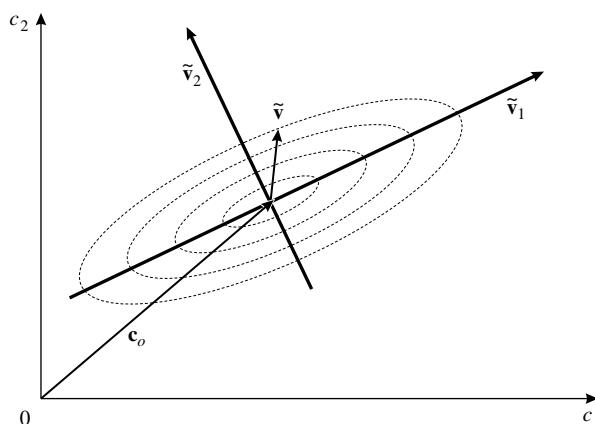


FIGURE 6.7
Contours of constant MSE and principal-component axes for a second-order quadratic error surface.

and the optimum estimate can be written as

$$\hat{y}_o = \mathbf{c}_o^H \mathbf{x} = \sum_{i=1}^M \frac{d'_i}{\lambda_i} (\mathbf{q}_i^H \mathbf{x}) \quad (6.2.36)$$

which leads to the representation of the optimum estimator shown in Figure 6.8. The eigen-filters \mathbf{q}_i decorrelate the data vector \mathbf{x} into its principal components, which are weighted and added to produce the optimum estimate.

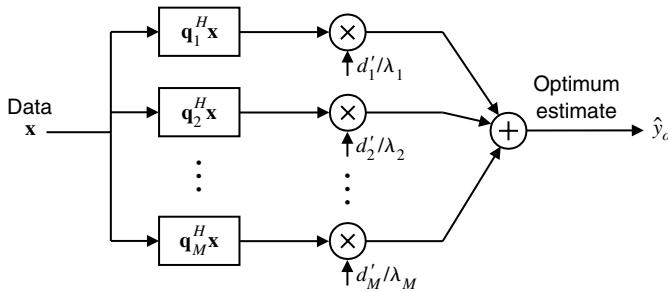


FIGURE 6.8

Principal-components representation of the optimum linear estimator.

6.2.4 Geometric Interpretations and the Principle of Orthogonality

It is convenient and pedagogic to think of random variables with zero mean value and finite variance as vectors in an abstract vector space with an inner product (i.e., a Hilbert space) defined by their correlation

$$\langle x, y \rangle \triangleq E\{xy^*\} \quad (6.2.37)$$

and the length of a vector by

$$\|x\|^2 \triangleq \langle x, x \rangle = E\{|x|^2\} < \infty \quad (6.2.38)$$

From the definition of the correlation coefficient in Section 3.2.1 and the above definitions, we obtain

$$|\langle x, y \rangle|^2 \leq \|x\| \|y\| \quad (6.2.39)$$

which is known as the Cauchy-Schwartz inequality. Two random variables are *orthogonal*, denoted by $x \perp y$, if

$$\langle x, y \rangle = E\{xy^*\} = 0 \quad (6.2.40)$$

which implies they are uncorrelated since they have zero mean.

This geometric viewpoint offers an illuminating and intuitive interpretation for many aspects of MSE estimation that we will find very useful. Indeed, using (6.2.9), (6.2.10), and (6.2.12), we have

$$E\{\mathbf{x}e_o^*\} = E\{\mathbf{x}(y^* - \mathbf{x}^H \mathbf{c}_o)\} = E\{\mathbf{x}y^*\} - E\{\mathbf{x}\mathbf{x}^H\}\mathbf{c}_o = \mathbf{d} - \mathbf{R}\mathbf{c}_o = \mathbf{0}$$

$$\text{Therefore } E\{\mathbf{x}e_o^*\} = \mathbf{0} \quad (6.2.41)$$

$$\text{or } E\{x_m e_o^*\} = 0 \quad \text{for } 1 \leq m \leq M \quad (6.2.42)$$

that is, the *estimation error is orthogonal to the data used for the estimation*. Equations

(6.2.41), or equivalently (6.2.42), are known as the *orthogonality principle* and are widely used in linear MMSE estimation.

To illustrate the use of the orthogonality principle, we note that any linear combination $c_1^*x_1 + \dots + c_M^*x_M$ lies in the subspace defined by the vectors[†] x_1, \dots, x_M . Therefore, the estimate \hat{y} that minimizes the squared length of the error vector e , that is, the MSE, is determined by the foot of the perpendicular from the tip of the vector y to the “plane” defined by vectors x_1, \dots, x_M . This is illustrated in Figure 6.9 for $M = 2$. Since e_o is perpendicular to every vector in the plane, we have $x_m \perp e_o$, $1 \leq m \leq M$, which leads to the orthogonality principle (6.2.42). Conversely, we can start with the orthogonality principle (6.2.41) and derive the normal equations. This interpretation has led to the name normal equations for (6.2.12). We will see several times that the concept of orthogonality has many important theoretical and practical implications. As an illustration, we apply the Pythagorean theorem to the orthogonal triangle formed by vectors \hat{y}_o , e_o , and y , in Figure 6.9, to obtain

$$\|y\|^2 = \|\hat{y}_o\|^2 + \|e_o\|^2$$

or

$$E\{|y|^2\} = E\{|\hat{y}_o|^2\} + E\{|e_o|^2\} \quad (6.2.43)$$

which decomposes the power of the desired response into two components, one that is correlated to the data and one that is uncorrelated to the data.

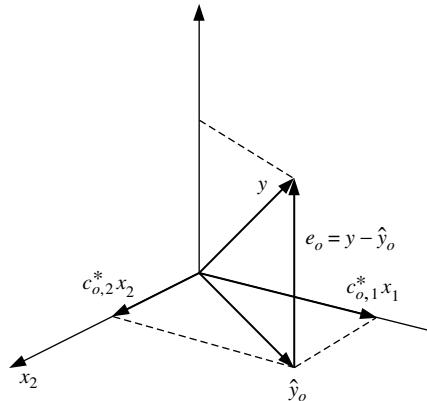


FIGURE 6.9
Pictorial illustration of the orthogonality principle. For random vectors orthogonality holds on the “average.”

6.2.5 Summary and Further Properties

We next summarize, for emphasis and future reference, some important properties of optimum, in the MMSE sense, linear estimators.

1. Equations (6.2.12) and (6.2.17) show that the optimum estimator and the MMSE depend *only* on the second-order moments of the desired response and the data. The dependence on the second-order moments is a consequence of both the linearity of the estimator and the use of the MSE criterion.
2. The error performance surface of the optimum estimator is a quadratic function of its coefficients. If the data correlation matrix is positive definite, this function has a unique minimum that determines the optimum set of coefficients. The surface can be visualized as a bowl, and the optimum estimator corresponds to the bottom of the bowl.

[†]We should be careful to avoid confusing vector random variables, that is, vectors whose components are random variables, and random variables interpreted as vectors in the abstract vector space defined by Equations (6.2.37) to (6.2.39).

3. If the data correlation matrix \mathbf{R} is positive definite, any deviation from the optimum increases the MMSE according to (6.2.21). The resulting excess MSE depends on \mathbf{R} only. This property is very useful in the design of adaptive filters.
4. When the estimator operates with the optimum set of coefficients, the error e_o is uncorrelated (orthogonal) to both the data x_1, x_2, \dots, x_M and the optimum estimate \hat{y}_o . This property is very useful if we want to monitor the performance of an optimum estimator in practice and is used also to design adaptive filters.
5. The MMSE, the optimum estimator, and the optimum estimate can be expressed in terms of the eigenvalues and eigenvectors of the data correlation matrix. See (6.2.32), (6.2.35), and (6.2.36).
6. The general (unconstrained) estimator

$$\hat{y} \triangleq h(\mathbf{x}) = h(x_1, x_2, \dots, x_M)$$

that minimizes the MSE

$$P = E\{|y - h(\mathbf{x})|^2\}$$

with respect to $h(\mathbf{x})$ is given by the *mean of the conditional density*, that is,

$$\hat{y}_o \triangleq h_o(\mathbf{x}) = E\{y|\mathbf{x}\} = \int_{-\infty}^{\infty} y p_y(y|\mathbf{x}) dy$$

and clearly is a nonlinear function of x_1, \dots, x_M . If the desired response and the data are jointly Gaussian, the linear MMSE estimator is the best in the MMSE sense; that is, we *cannot* find a nonlinear estimator that produces an estimate with smaller MMSE (Papoulis 1991).

6.3 SOLUTION OF THE NORMAL EQUATIONS

In this section, we present a numerical method for the solution of the normal equations and the computation of the minimum error, using a slight modification of the Cholesky decomposition of Hermitian positive definite matrices known as the *lower-diagonal-upper decomposition*, or LDL^H decomposition for short.

Hermitian positive definite matrices can be uniquely decomposed into the product of a lower triangular and a diagonal and an upper triangular matrix as

$$\mathbf{R} = \mathbf{L} \mathbf{D} \mathbf{L}^H \quad (6.3.1)$$

where \mathbf{L} is a unit lower-triangular matrix

$$\mathbf{L} \triangleq \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{10} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{M-1,0} & l_{M-1,1} & \cdots & 1 \end{bmatrix} \quad (6.3.2)$$

and

$$\mathbf{D} = \text{diag}\{\xi_1, \xi_2, \dots, \xi_M\} \quad (6.3.3)$$

is a diagonal matrix with strictly real, positive elements. When the decomposition (6.3.1) is known, we can solve the normal equations

$$\mathbf{R}\mathbf{c}_o = \mathbf{L}\mathbf{D}(\mathbf{L}^H\mathbf{c}_o) = \mathbf{d} \quad (6.3.4)$$

by solving the lower triangular system

$$\mathbf{L}\mathbf{D}\mathbf{k} \triangleq \mathbf{d} \quad (6.3.5)$$

for the intermediate vector \mathbf{k} and the upper triangular system

$$\mathbf{L}^H\mathbf{c}_o = \mathbf{k} \quad (6.3.6)$$

for the optimum estimator \mathbf{c}_o . The advantage is that the solution of triangular systems of equations is trivial.

We next provide a constructive proof of the LDL^H decomposition by example and illustrate its application to the solution of the normal equations for $M = 4$. The generalization to an arbitrary order is straightforward and is given in Section 7.1.4.

EXAMPLE 6.3.1. Writing the decomposition (6.3.1) explicitly for $M = 4$, we have

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{10} & 1 & 0 & 0 \\ l_{20} & l_{21} & 1 & 0 \\ l_{30} & l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} \xi_1 & 0 & 0 & 0 \\ 0 & \xi_2 & 0 & 0 \\ 0 & 0 & \xi_3 & 0 \\ 0 & 0 & 0 & \xi_4 \end{bmatrix} \begin{bmatrix} 1 & l_{10}^* & l_{20}^* & l_{30}^* \\ 0 & 1 & l_{21}^* & l_{31}^* \\ 0 & 0 & 1 & l_{32}^* \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.3.7)$$

where $r_{ij} = r_{ji}^*$ and $\xi_i > 0$, by assumption. If we perform the matrix multiplications on the right-hand side of (6.3.7) and equate the matrix elements on the left and right sides, we obtain

$$\begin{aligned} r_{11} &= \xi_1 & \Rightarrow \quad \xi_1 &= r_{11} \\ r_{21} &= \xi_1 l_{10} & \Rightarrow \quad l_{10} &= \frac{r_{21}}{\xi_1} \\ r_{22} &= \xi_1 |l_{10}|^2 + \xi_2 & \Rightarrow \quad \xi_2 &= r_{22} - \xi_1 |l_{10}|^2 \\ r_{31} &= \xi_1 l_{20} & \Rightarrow \quad l_{20} &= \frac{r_{31}}{\xi_1} \\ r_{32} &= \xi_1 l_{20} l_{10}^* + \xi_2 l_{21} & \Rightarrow \quad l_{21} &= \frac{r_{32} - \xi_1 l_{20} l_{10}^*}{\xi_2} \\ r_{33} &= \xi_1 |l_{20}|^2 + \xi_2 |l_{21}|^2 + \xi_3 & \Rightarrow \quad \xi_3 &= r_{33} - \xi_1 |l_{20}|^2 - \xi_2 |l_{21}|^2 \\ r_{41} &= \xi_1 l_{30} & \Rightarrow \quad l_{30} &= \frac{r_{41}}{\xi_1} \\ r_{42} &= \xi_1 l_{30} l_{10}^* + \xi_2 l_{31} & \Rightarrow \quad l_{31} &= \frac{r_{42} - \xi_1 l_{30} l_{10}^*}{\xi_2} \\ r_{43} &= \xi_1 l_{30} l_{20}^* + \xi_2 l_{31} l_{21}^* + \xi_3 l_{32} & \Rightarrow \quad l_{32} &= \frac{r_{43} - \xi_1 l_{30} l_{20}^* - \xi_2 l_{31} l_{21}^*}{\xi_3} \\ r_{44} &= \xi_1 |l_{30}|^2 + \xi_2 |l_{31}|^2 + \xi_3 |l_{32}|^2 + \xi_4 & \Rightarrow \quad \xi_4 &= r_{44} - \xi_1 |l_{30}|^2 - \xi_2 |l_{31}|^2 - \xi_3 |l_{32}|^2 \end{aligned} \quad (6.3.8)$$

which provides a row-by-row computation of the elements of the LDL^H decomposition. We note that the computation of the next row does not change the already computed rows.

The lower unit triangular system in (6.3.5) becomes

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{10} & 1 & 0 & 0 \\ l_{20} & l_{21} & 1 & 0 \\ l_{30} & l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} \xi_1 k_1 \\ \xi_2 k_2 \\ \xi_3 k_3 \\ \xi_4 k_4 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} \quad (6.3.9)$$

and can be solved by forward substitution, starting with the first equation. Indeed, we obtain

$$\begin{aligned} \xi_1 k_1 &= d_1 \Rightarrow k_1 = \frac{d_1}{\xi_1} \\ l_{10} \xi_1 k_1 + \xi_2 k_2 &= d_2 \Rightarrow k_2 = \frac{d_2 - l_{10} \xi_1 k_1}{\xi_2} \\ l_{20} \xi_1 k_1 + l_{21} \xi_2 k_2 + \xi_3 k_3 &= d_3 \Rightarrow k_3 = \frac{d_3 - l_{20} \xi_1 k_1 - l_{21} \xi_2 k_2}{\xi_3} \\ l_{30} \xi_1 k_1 + l_{31} \xi_2 k_2 + l_{32} \xi_3 k_3 + \xi_4 k_4 &= d_4 \Rightarrow k_4 = \frac{d_4 - l_{30} \xi_1 k_1 - l_{31} \xi_2 k_2 - l_{32} \xi_3 k_3}{\xi_4} \end{aligned} \quad (6.3.10)$$

which compute the coefficients k_i in “forward” order. Then, the optimum estimator is obtained by solving the upper unit triangular system in (6.3.6) by backward substitution, starting from the last equation. Indeed, we have

$$\begin{bmatrix} 1 & l_{10}^* & l_{20}^* & l_{30}^* \\ 0 & 1 & l_{21}^* & l_{31}^* \\ 0 & 0 & 1 & l_{32}^* \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1^{(4)} \\ c_2^{(4)} \\ c_3^{(4)} \\ c_4^{(4)} \end{bmatrix} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \end{bmatrix} \Rightarrow \begin{aligned} c_4^{(4)} &= k_4 \\ c_3^{(4)} &= k_3 - l_{32}^* c_4 \\ c_2^{(4)} &= k_2 - l_{21}^* c_3 - l_{31}^* c_4 \\ c_1^{(4)} &= k_1 - l_{10}^* c_2 - l_{20}^* c_3 - l_{30}^* c_4 \end{aligned} \quad (6.3.11)$$

that is, the coefficients of the optimum estimator are computed in “backward” order. As a result of this backward substitution, computing one more coefficient for the optimum estimator changes *all* the previously computed coefficients. Indeed, the coefficients of the third-order estimator are

$$\begin{bmatrix} 1 & l_{10}^* & l_{20}^* \\ 0 & 1 & l_{21}^* \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1^{(3)} \\ c_2^{(3)} \\ c_3^{(3)} \end{bmatrix} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} \Rightarrow \begin{aligned} c_3^{(3)} &= k_3 \\ c_2^{(3)} &= k_2 - l_{21}^* c_3^{(3)} \\ c_1^{(3)} &= k_1 - l_{10}^* c_2^{(3)} - l_{20}^* c_3^{(3)} \end{aligned} \quad (6.3.12)$$

which are different from the first three coefficients of the fourth-order estimator.

Careful inspection of the formulas for r_{11}, r_{22}, r_{33} , and r_{44} shows that the diagonal elements of \mathbf{R} provide an upper bound for the elements of \mathbf{L} and \mathbf{D} , which is the reason for the good numerical properties of the LDL^H decomposition algorithm. The general formulas for the row-by-row computation of the triangular decomposition, forward substitution, and backward substitution are given in Table 6.1 and can be easily derived by generalizing the results of the previous example. The triangular decomposition requires $M^3/6$ operations, and the solution of each triangular system requires $M(M + 1)/2 \approx M^2/2$ operations.

TABLE 6.1
Solution of normal equations using triangular decomposition.

For $i = 1, 2, \dots, M$ and for $j = 0, 1, \dots, i - 1$,

$$l_{ij} = \frac{1}{\xi_i} \left(r_{i+1,j+1} - \sum_{m=0}^{j-1} \xi_{m+1} l_{im} l_{jm}^* \right) \quad (\text{not executed when } i = M)$$

$$\xi_i = r_{ii} - \sum_{m=1}^{i-1} \xi_m |l_{i-1,m-1}|^2$$

For $i = 1, 2, \dots, M$,

$$k_i = \frac{d_i}{\xi_i} - \sum_{m=0}^{i-2} l_{i-1,m} k_{m+1}$$

For $i = M, M - 1, \dots, 1$,

$$c_i = k_i - \sum_{m=i+1}^M l_{m-1,i-1}^* c_m$$

The decomposition (6.3.1) leads to an interesting and practical formula for the computation of the MMSE *without* using the optimum estimator coefficients. Indeed, using (6.2.17), (6.3.6), and (6.3.1), we obtain

$$P_o = P_y - \mathbf{c}_o^H \mathbf{R} \mathbf{c}_o = P_y - \mathbf{k}^H \mathbf{L}^{-1} \mathbf{R} (\mathbf{L}^{-1})^H \mathbf{k} = P_y - \mathbf{k}^H \mathbf{D} \mathbf{k} \quad (6.3.13)$$

or in scalar form

$$P_o = P_y - \sum_{i=1}^M \xi_i |k_i|^2 \quad (6.3.14)$$

since \mathbf{D} is diagonal. Equation (6.3.14) shows that because $\xi_i > 0$, *increasing the order of the filter can only reduce the minimum error and hence leads to a better estimate*. Another important application of (6.3.14) is in the computation of arbitrary positive definite quadratic forms. Such problems arise in various statistical applications, such as detection and hypothesis testing, involving the correlation matrix of Gaussian processes (McDonough and Whalen 1995).

Since the determinant of a unit lower triangular matrix equals 1, from (6.3.1) we obtain

$$\det \mathbf{R} = (\det L)(\det D)(\det L^T) \prod_{i=1}^M \xi_i \quad (6.3.15)$$

which shows that if \mathbf{R} is positive definite, $\xi_i > 0$ for all i , and vice versa.

The triangular decomposition of symmetric, positive definite matrices is numerically stable. The function `[L,D]=ldlt(R)` implements the first part of the algorithm in Table 6.1, and it fails only if matrix \mathbf{R} is *not* positive definite. Therefore, it can be used as an efficient test to find out whether a symmetric matrix is positive definite. The function `[co, Po]=lduneqs(L, D, d)` computes the MMSE estimator using the last formula in Table 6.1 and the corresponding MMSE using (6.3.14).

To summarize, linear MMSE estimation involves the following computational steps

- | | |
|---|--|
| 1. $\mathbf{R} = E\{\mathbf{x}\mathbf{x}^H\}$, $\mathbf{d} = E\{\mathbf{xy}^*\}$ | Normal equations $\mathbf{R}\mathbf{c}_o = \mathbf{d}$ |
| 2. $\mathbf{R} = \mathbf{LDL}^H$ | Triangular decomposition |
| 3. $\mathbf{LDk} = \mathbf{d}$ | Forward substitution $\rightarrow \mathbf{k}$ |
| 4. $\mathbf{L}^H \mathbf{c}_o = \mathbf{k}$ | Backward substitution $\rightarrow \mathbf{c}_o$ |
| 5. $P_o = P_y - \mathbf{k}^H \mathbf{D} \mathbf{k}$ | MMSE computation |
| 6. $e = y - \mathbf{c}_o^H \mathbf{x}$ | Computation of residuals |
- (6.3.16)

The vector \mathbf{k} can also be obtained using the LDL^H decomposition of an augmented correlation matrix. To this end, consider the augmented vector

$$\bar{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ y \end{bmatrix} \quad (6.3.17)$$

and its correlation matrix

$$\bar{\mathbf{R}} = E\{\bar{\mathbf{x}}\bar{\mathbf{x}}^H\} = \begin{bmatrix} E\{\mathbf{x}\mathbf{x}^H\} & E\{\mathbf{xy}^*\} \\ E\{y\mathbf{x}^H\} & E\{|y|^2\} \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{d} \\ \mathbf{d}^H & P_y \end{bmatrix} \quad (6.3.18)$$

We can easily show that the LDL^H decomposition of $\bar{\mathbf{R}}$ is

$$\bar{\mathbf{R}} = \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{k}^H & 1 \end{bmatrix} \begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0}^H & P_o \end{bmatrix} \begin{bmatrix} \mathbf{L}^H & \mathbf{k}^H \\ \mathbf{0}^H & 1 \end{bmatrix} \quad (6.3.19)$$

which provides the MMSE P_o and the quantities \mathbf{L} and \mathbf{k} required to obtain the optimum estimator \mathbf{c}_o by solving $\mathbf{L}^H \mathbf{c}_o = \mathbf{k}$.

EXAMPLE 6.3.2. Compute, using the LDL^H method, the optimum estimator and the MMSE specified by the following second-order moments:

$$\mathbf{R} = \begin{bmatrix} 1 & 3 & 2 & 4 \\ 3 & 12 & 18 & 21 \\ 2 & 18 & 54 & 48 \\ 4 & 21 & 48 & 55 \end{bmatrix} \quad \mathbf{d} = \begin{bmatrix} 1 \\ 2 \\ 1.5 \\ 4 \end{bmatrix} \quad \text{and} \quad P_y = 100$$

Solution. We first compute the triangular factors

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 2 & 4 & 1 & 0 \\ 4 & 3 & 2 & 1 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

using (6.3.8), and the vector \mathbf{k}

$$\mathbf{k} = [1 \ -\frac{1}{3} \ 1.75 \ -1]^T$$

using (6.3.9). Then we determine the optimum estimator

$$\mathbf{c} = [34.5 \ -12\frac{1}{3} \ 3.75 \ -1]^T$$

by solving the triangular system (6.3.11). The corresponding MMSE

$$P_o = 88.5$$

can be evaluated by using either (6.2.17) or (6.3.14). The reader can easily verify that the LDL^H decomposition of $\bar{\mathbf{R}}$ provides the elements of \mathbf{L} , \mathbf{k} , and P_o .

Since the diagonal elements ξ_k are positive, the matrix

$$\mathcal{L} \triangleq \mathbf{L}\mathbf{D}^{1/2} \quad (6.3.20)$$

is lower triangular with positive diagonal elements. Then (6.3.1) can be written as

$$\mathbf{R} = \mathcal{L}\mathcal{L}^H \quad (6.3.21)$$

which is known as the *Cholesky decomposition* of \mathbf{R} (Golub and Van Loan 1996). The computation of \mathcal{L} requires $M^3/6$ multiplications and additions and M square roots and can be done by using the function `L=chol(R)'`. The function `[L,D]=ldltchol(R)` computes the LDL^H decomposition using the function `chol`.

6.4 OPTIMUM FINITE IMPULSE RESPONSE FILTERS

In the previous section, we presented the theory of general linear MMSE estimators [see Figure 6.1(a)]. In this section, we apply these results to the design of optimum linear filters, that is, filters whose performance is the best possible when measured according to the MMSE criterion [see Figure 6.1(b)]. The general formulation of the optimum filtering problem is shown in Figure 6.10. The optimum filter forms an estimate $\hat{y}(n)$ of the desired response $y(n)$ by using samples from a related input signal $x(n)$. The theory of optimum filters was developed by Wiener (1942) in continuous time and Kolmogorov (1939) in discrete time. Levinson (1947) reformulated the theory for FIR filters and stationary processes and developed an efficient algorithm for the solution of the normal equations that exploits the Toeplitz structure of the autocorrelation matrix \mathbf{R} (see Section 7.4). For this reason, linear MMSE filters are often referred to as Wiener filters.

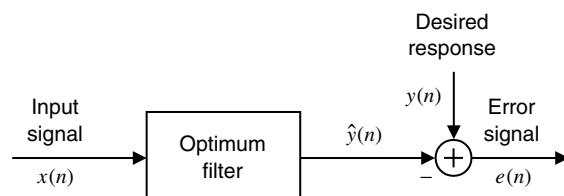


FIGURE 6.10
Block diagram representation of the optimum filtering problem.

We consider a linear FIR filter specified by its impulse response $h(n, k)$. The output of the filter is determined by the superposition summation

$$\hat{y}(n) = \sum_{k=0}^{M-1} h(n, k)x(n-k) \quad (6.4.1)$$

$$\triangleq \sum_{k=1}^M c_k^*(n)x(n-k+1) \triangleq \mathbf{c}^H(n)\mathbf{x}(n) \quad (6.4.2)$$

where

$$\mathbf{c}(n) \triangleq [c_1(n) \ c_2(n) \ \cdots \ c_M(n)]^T \quad (6.4.3)$$

and

$$\mathbf{x}(n) \triangleq [x(n) \ x(n-1) \ \cdots \ x(n-M+1)]^T \quad (6.4.4)$$

are the filter *coefficient vector*[†] and the *input data vector*, respectively. Equation (6.4.1) becomes a convolution if $h(n, k)$ does not depend on n , that is, when the filter is time-invariant. The objective is to find the coefficient vector that minimizes the MSE $E\{|e(n)|^2\}$.

We prefer FIR over IIR filters because (1) any stable IIR filter can be approximated to any desirable degree by an FIR filter and (2) optimum FIR filters are easily obtained by solving a linear system of equations.

6.4.1 Design and Properties

To determine the optimum FIR filter $\mathbf{c}_o(n)$, we note that at every time instant n , the optimum filter is the linear MMSE estimator of the desired response $y(n)$ based on the data $\mathbf{x}(n)$. Since for any fixed n the quantities $y(n), x(n), \dots, x(n-M+1)$ are random variables, we can determine the optimum filter either from (6.2.12) by replacing \mathbf{x} by $\mathbf{x}(n)$, y by $y(n)$, and \mathbf{c}_o by $\mathbf{c}_o(n)$; or by applying the orthogonality principle (6.2.41). Indeed, using (6.2.41), (6.1.2), and (6.4.2), we have

$$E\{\mathbf{x}(n)[y^*(n) - \mathbf{x}^H(n)\mathbf{c}_o(n)]\} = \mathbf{0} \quad (6.4.5)$$

which leads to the following set of normal equations

$$\mathbf{R}(n)\mathbf{c}_o(n) = \mathbf{d}(n) \quad (6.4.6)$$

where

$$\mathbf{R}(n) \triangleq E\{\mathbf{x}(n)\mathbf{x}^H(n)\} \quad (6.4.7)$$

is the correlation matrix of the input data vector and

$$\mathbf{d}(n) \triangleq E\{\mathbf{x}(n)y^*(n)\} \quad (6.4.8)$$

is the cross-correlation vector between the desired response and the input data vector, that is, the input values stored currently in the filter memory and used by the filter to estimate the desired response. We see that, at every time n , the coefficients of the optimum filter are obtained as the solution of a linear system of equations. The filter $\mathbf{c}_o(n)$ is optimum if and only if the Hermitian matrix $\mathbf{R}(n)$ is positive definite.

To find the MMSE, we can use either (6.2.17) or the orthogonality principle (6.2.41). Using the orthogonality principle, we have

$$\begin{aligned} P_o(n) &= E\{e_o(n)[y^*(n) - \mathbf{x}^H(n)\mathbf{c}_o(n)]\} \\ &= E\{e_o(n)y^*(n)\} \quad \text{due to orthogonality} \\ &= E\{[y(n) - \mathbf{x}^H(n)\mathbf{c}_o(n)]y^*(n)\} \end{aligned}$$

[†]We define $c_{k+1}(n) \triangleq h^*(n, k)$, $0 \leq k \leq M-1$ in order to comply with the definition $\mathbf{R}(n) \triangleq E\{\mathbf{x}(n)\mathbf{x}^H(n)\}$ of the correlation matrix.

which can be written as

$$P_o(n) = P_y(n) - \mathbf{d}^H(n)\mathbf{c}_o(n) \quad (6.4.9)$$

The first term

$$P_y(n) \triangleq E\{|y(n)|^2\} \quad (6.4.10)$$

is the power of the desired response signal and represents the MSE in the absence of filtering. The second term $\mathbf{d}^H(n)\mathbf{c}_o(n)$ is the reduction in the MSE that is obtained by using the optimum filter.

In many practical applications, we need to know the performance of the optimum filter in terms of MSE reduction prior to computing the coefficients of the filter. Then we can decide if it is preferable to (1) use an optimum filter (assuming we can design one), (2) use a simpler suboptimum filter with adequate performance, or (3) not use a filter at all. Hence, the performance of the optimum filter can serve as a yardstick for other competing methods.

The optimum filter consists of (1) a linear system solver that determines the optimum set of coefficients from the normal equations formed, using the known second-order moments, and (2) a discrete-time filter that computes the estimate $\hat{y}(n)$ (see Figure 6.11). The solution of (6.4.6) can be obtained by using standard linear system solution techniques. In MATLAB, we solve (6.4.6) by `copt=R\d` and compute the MMSE by `Popt=Py-dot(conj(d),copt)`. The optimum filter is implemented by `yest=filter(copt,1,x)`. We emphasize that the optimum filter only needs the input signal for its operation, that is, to form the estimate of $y(n)$; the desired response, if it is available, may be used for other purposes.

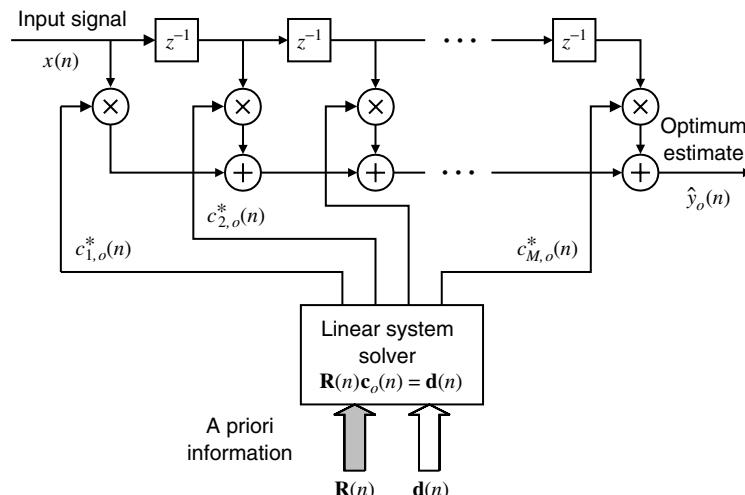


FIGURE 6.11

Design and implementation of a time-varying optimum FIR filter.

Conventional frequency-selective filters are designed to shape the spectrum of the input signal within a specific frequency band in which it operates. In this sense, these filters are effective only if the components of interest in the input signal have their energy concentrated within *nonoverlapping* bands. To design the filters, we need to know the limits of these bands, not the values of the sequences to be filtered. Note that such filters do not depend on the values of the data (values of the samples) to be filtered; that is, they are *not* data-adaptive. In contrast, optimum filters are designed using the second-order moments of the processed signals and have the same effect on all classes of signals with the same second-order moments. Optimum filters are effective even if the signals of interest have

overlapping spectra. Although the actual data values also do not affect optimum filters, that is, they are also not data-adaptive, these filters are optimized to the statistics of the data and thus provide superior performance when judged by the statistical criterion.

The dependence of the optimum filter *only* on the second-order moments is a consequence of the linearity of the filter and the use of the MSE criterion. Phase information about the input signal or non-second-order moments of the input and desired response processes is not needed; even if the moments are known, they are not used by the filter. Such information is useful only if we employ a nonlinear filter or use another criterion of performance.

The error performance surface of the optimum direct-form FIR filter is a quadratic function of its impulse response. If the input correlation matrix is positive definite, this function has a unique minimum that determines the optimum set of coefficients. The surface can be visualized as a bowl, and the optimum filter corresponds to the bottom of the bowl. The bottom is moving if the processes are nonstationary and fixed if they are stationary. In general, the shape of the error performance surface depends on the criterion of performance and the structure of the filter. Note that the use of another criterion of performance or another filter structure may lead to error performance surfaces with multiple local minima or saddle points.

6.4.2 Optimum FIR Filters for Stationary Processes

Further simplifications and additional insight into the operation of optimum linear filters are possible when the input and desired response stochastic processes are *jointly wide-sense stationary*. In this case, the correlation matrix of the input data and the cross-correlation vector do *not* depend on the time index n . Therefore, the optimum filter and the MMSE are *time-invariant* (i.e., they are independent of the time index n) and are determined by

$$\mathbf{R}\mathbf{c}_o = \mathbf{d} \quad (6.4.11)$$

and

$$P_o = P_y - \mathbf{d}^H \mathbf{c}_o \quad (6.4.12)$$

Owing to stationarity, the autocorrelation matrix is

$$\mathbf{R} \triangleq \begin{bmatrix} r_x(0) & r_x(1) & \cdots & r_x(M-1) \\ r_x^*(1) & r_x(0) & \cdots & r_x(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_x^*(M-1) & r_x^*(M-2) & \cdots & r_x(0) \end{bmatrix} \quad (6.4.13)$$

determined by the autocorrelation $r_x(l) = E\{x(n)x^*(n-l)\}$ of the input signal. The cross-correlation vector between the desired response and the input data vector is

$$\mathbf{d} \triangleq [d_1 \ d_2 \ \cdots \ d_M]^T \triangleq [r_{yx}^*(0) \ r_{yx}^*(1) \ \cdots \ r_{yx}^*(M-1)]^T \quad (6.4.14)$$

and P_y is the power of the desired response. For stationary processes, the matrix \mathbf{R} is Toeplitz and positive definite unless the components of the data vector are linearly dependent.

Since the optimum filter is time-invariant, it is implemented by using convolution

$$\hat{y}_o(n) = \sum_{k=0}^{M-1} h_o(k) x(n-k) \quad (6.4.15)$$

where $h_o(n) = c_{o,n+1}^*$ is the impulse response of the optimum filter.

Using (6.4.13), (6.4.14), $h_o(n) = c_{o,n+1}^*$, and $r(l) = r^*(-l)$, we can write the normal equations (6.4.11) more explicitly as

$$\sum_{k=0}^{M-1} h_o(k) r_{*}(m-k) = r_{yx}(m) \quad 0 \leq m \leq M-1 \quad (6.4.16)$$

which is the discrete-time counterpart of the *Wiener-Hopf* integral equation, and its solution determines the impulse response of the optimum filter. We notice that the cross-correlation between the input signal and the desired response (right-hand side) is equal to the convolution between the autocorrelation of the input signal and the optimum filter (left-hand side). Thus, to obtain the optimum filter, we need to solve a convolution equation.

The MMSE is given by

$$P_o = P_y - \sum_{k=0}^{M-1} h_o(k) r_{yx}^*(k) \quad (6.4.17)$$

which is obtained by substituting (6.4.14) into (6.4.12). Table 6.2 summarizes the information required for the design of an optimum (in the MMSE sense) linear time-invariant filter, the Wiener-Hopf equations that define the filter, and the resulting MMSE.

TABLE 6.2
Specification of optimum linear filters for stationary signals. The limits 0 and $M - 1$ on the summations can be replaced by any values M_1 and M_2 .

Filter and Error Definitions	$e(n) \triangleq y(n) - \sum_{k=0}^{M-1} h(k)x(n-k)$
Criterion of Performance	$P \triangleq E\{ e(n) ^2\} \rightarrow \text{minimum}$
Wiener-Hopf Equations	$\sum_{k=0}^{M-1} h_o(k)r_x(m-k) = r_{yx}(m), 0 \leq m \leq M-1$
Minimum MSE	$P_o = P_y - \sum_{k=0}^{M-1} h_o(k)r_{yx}^*(k)$
Second-Order Statistics	$r_x(l) = E\{x(n)x^*(n-l)\}, P_y = E\{ y(n) ^2\}$ $r_{yx}(l) = E\{y(n)x^*(n-l)\}$

To summarize, for nonstationary processes $\mathbf{R}(n)$ is Hermitian and nonnegative definite, and the optimum filter $\mathbf{h}_o(n)$ is time-varying. For stationary processes, \mathbf{R} is Hermitian, non-negative definite, and Toeplitz, and the optimum filter is time-invariant. A Toeplitz autocorrelation matrix is positive definite if the power spectrum of the input satisfies $R_x(e^{j\omega}) > 0$ for all frequencies ω . In both cases, the filter is used for all realizations of the processes. If $M = \infty$, we have a causal IIR optimum filter determined by an infinite-order linear system of equations that can only be solved in the stationary case by using analytical techniques (see Section 6.6).

EXAMPLE 6.4.1. Consider a harmonic random process

$$y(n) = A \cos(\omega_0 n + \phi)$$

with fixed, but unknown, amplitude and frequency, and random phase ϕ , uniformly distributed on the interval from 0 to 2π . This process is corrupted by additive white Gaussian noise $v(n) \sim N(0, \sigma_v^2)$ that is uncorrelated with $y(n)$. The resulting signal $x(n) = y(n) + v(n)$ is available to the user for processing. Design an optimum FIR filter to remove the corrupting noise $v(n)$ from the observed signal $x(n)$.

Solution. The input of the optimum filter is $x(n)$, and the desired response is $y(n)$. The signal $y(n)$ is obviously unavailable, but to design the filter, we only need the second-order moments $r_x(l)$ and $r_{yx}(l)$. We first note that since $y(n)$ and $v(n)$ are uncorrelated, the autocorrelation of

$$r_x(l) = r_y(l) + r_v(l) = \frac{1}{2}A^2 \cos \omega_0 l + \sigma_v^2 \delta(l)$$

where $r_y(l) = \frac{1}{2}A^2 \cos \omega_0 l$ is the autocorrelation of $y(n)$. The cross-correlation between the desired response $y(n)$ and the input signal $x(n)$ is

$$r_{yx}(l) = E\{y(n)[y(n-l) + v(n-l)]\} = r_y(l)$$

Therefore, the autocorrelation matrix \mathbf{R} is symmetric Toeplitz and is determined by the elements $r(0), r(1), \dots, r(M-1)$ of its first row. The right-hand side of the Wiener-Hopf equations is $\mathbf{d} = [r_y(0) \ r_y(1) \ \dots \ r_y(M-1)]^T$. If we know $r_y(l)$ and σ_v^2 , we can numerically determine the optimum filter and the MMSE from (6.4.11) and (6.4.12). For example, suppose that $A = 0.5$, $f_0 = \omega_0/(2\pi) = 0.05$, and $\sigma_v^2 = 0.5$. The input signal-to-noise ratio (SNR) is

$$\text{SNR}_I = 10 \log \frac{A^2/2}{\sigma_v^2} = -6.02 \text{ dB}$$

The *processing gain (PG)*, defined as the ratio of signal-to-noise ratios at the output and input of a signal processing system

$$\text{PG} \triangleq \frac{\text{SNR}_O}{\text{SNR}_I}$$

provides another useful measure of performance.

The first problem we encounter is how to choose the order M of the filter. In the absence of any a priori information, we compute \mathbf{h}_o and P_o^h for $1 \leq M \leq M_{\max} = 50$ and PG and plot both results in Figure 6.12. We see that an $M = 20$ order filter provides satisfactory performance. Figure 6.13 shows a realization of the corrupted and filtered signals. Another useful approach to evaluate how well the optimum filter enhances a harmonic signal is to compute the spectra of the input and output signals and the frequency response of the optimum filter. These are shown in Figure 6.14, where we see that the optimum filter has a sharp bandpass about frequency f_0 , as expected (for details see Problem 6.5).

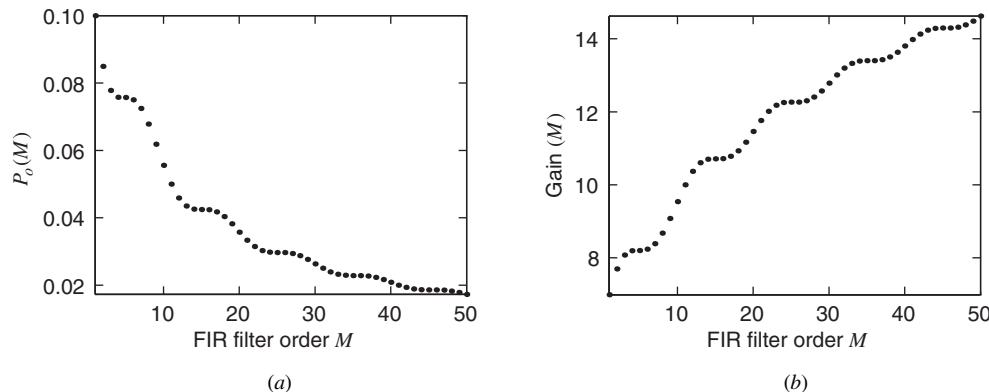
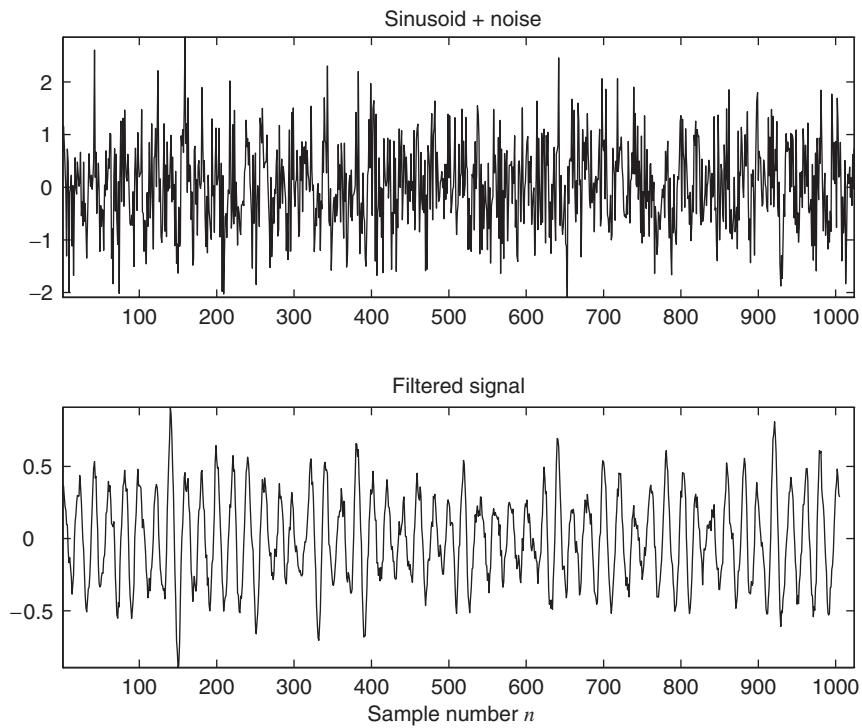
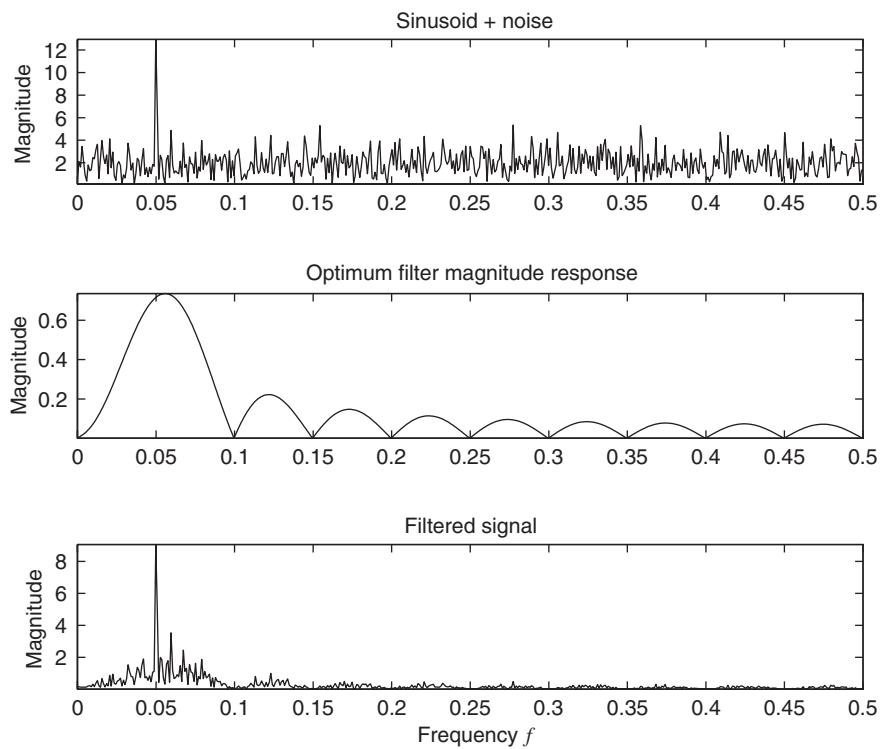


FIGURE 6.12
Plots of (a) the MMSE and (b) the processing gain as a function of the filter order M .

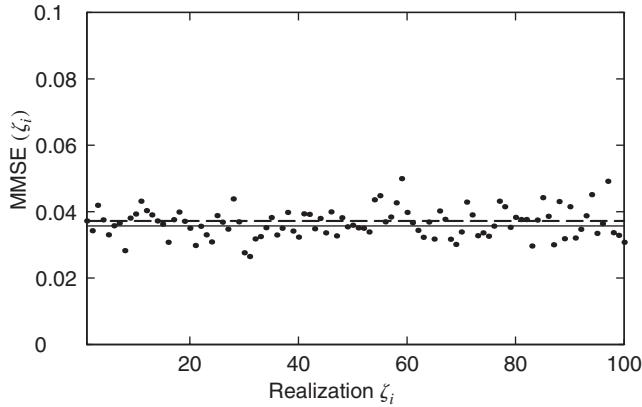
To illustrate the meaning of the estimator's optimality, we will use a Monte Carlo simulation. Thus, we generate $K = 100$ realizations of the sequence $x(\zeta_i, n)$, $0 \leq n \leq N-1$ ($N = 1000$); we compute the output sequence $\hat{y}(\zeta_i, n)$, using (6.4.15); and then the error sequence $e(\zeta_i, n) = y(\zeta_i, n) - \hat{y}(\zeta_i, n)$ and its variance $\hat{P}(\zeta_i)$. Figure 6.15 shows a plot of $\hat{P}(\zeta_i)$, $1 \leq \zeta_i \leq K$. We

**FIGURE 6.13**

Example of the noise-corrupted and filtered sinusoidal signals.

**FIGURE 6.14**

PSD of the input signal, magnitude response of the optimum filter, and PSD of the output signal.

**FIGURE 6.15**

Results of Monte Carlo simulation of the optimum filter. The solid line corresponds to the MMSE and the dashed line to the average of $\hat{P}(\xi_i)$ values.

notice that although the filter performs better or worse than the optimum in particular cases, on average its performance is close to the theoretically predicted one. This is exactly the meaning of the MMSE criterion: *optimum performance on the average (in the MMSE sense)*.

For a certain realization, the optimum filter may not perform as well as some other linear filters; however, on average, it performs better than any other linear filter of the same order when all possible realizations of $x(n)$ and $y(n)$ are considered.

6.4.3 Frequency-Domain Interpretations

We will now investigate the performance of the optimum filter, for stationary processes, in the frequency domain. Using (6.2.7), (6.4.13), and (6.4.14), we can easily show that the MSE of an FIR filter $h(n)$ is given by

$$P = E\{|e(n)|^2\} = r_y(0) - \sum_{k=0}^{M-1} h(k)r_{yx}^*(k) - \sum_{k=0}^{M-1} h^*(k)r_{yx}(k) + \sum_{k=0}^{M-1} \sum_{l=0}^{M-1} h(k)r(l-k)h^*(l) \quad (6.4.18)$$

The frequency response function of the FIR filter is

$$H(e^{j\omega}) \triangleq \sum_{k=0}^{M-1} h(k)e^{-jk\omega} \quad (6.4.19)$$

Using Parseval's theorem,

$$\sum_{n=-\infty}^{\infty} x_1(n)x_2^*(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X_1(e^{j\omega})X_2^*(e^{j\omega}) d\omega \quad (6.4.20)$$

we can show that the MSE (6.4.18) can be expressed in the frequency domain as

$$P = r_y(0) - \frac{1}{2\pi} \int_{-\pi}^{\pi} [H(e^{j\omega})R_{yx}^*(e^{j\omega}) + H^*(e^{j\omega})R_{yx}(e^{j\omega}) - H(e^{j\omega})H^*(e^{j\omega})R_x(e^{j\omega})] d\omega \quad (6.4.21)$$

where $R_x(e^{j\omega})$ is the PSD of $x(n)$ and $R_{yx}(e^{j\omega})$ is the cross-PSD of $y(n)$ and $x(n)$ (see Problem 6.10). This formula holds for both FIR and IIR filters.

If we minimize (6.4.21) with respect to $H(e^{j\omega})$, we obtain the system function of the optimum filter and the MMSE. However, we leave this for Problem 6.11 and instead express

(6.4.17) in the frequency domain by using (6.4.20). Indeed, we have

$$\begin{aligned} P_o &= r_y(0) - \frac{1}{2\pi} \int_{-\pi}^{\pi} H_o(e^{j\omega}) R_{yx}^*(e^{j\omega}) d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} [R_y(e^{j\omega}) - H_o(e^{j\omega}) R_{yx}^*(e^{j\omega})] d\omega \end{aligned} \quad (6.4.22)$$

where $H_o(e^{j\omega})$ is the frequency response of the optimum filter. The above equation holds for any filter, FIR or IIR, as long as we use the proper limits to compute the summation in (6.4.19).

We will now obtain a formula for the MMSE that holds only for IIR filters whose impulse response extends from $-\infty$ to ∞ . In this case, (6.4.16) is a convolution equation that holds for $-\infty < m < \infty$. Using the convolution theorem of the Fourier transform, we obtain

$$H_o(e^{j\omega}) = \frac{R_{yx}(e^{j\omega})}{R_x(e^{j\omega})} \quad (6.4.23)$$

which, we again stress, holds for noncausal IIR filters *only*. Substituting into (6.4.22), we obtain

$$\begin{aligned} P_o &= \frac{1}{2\pi} \int_{-\pi}^{\pi} [1 - \frac{|R_{yx}(e^{j\omega})|^2}{R_y(e^{j\omega}) R_x(e^{j\omega})}] R_y(e^{j\omega}) d\omega \\ \text{or} \quad P_o &= \frac{1}{2\pi} \int_{-\pi}^{\pi} [1 - \mathcal{G}_{yx}(e^{j\omega})] R_y(e^{j\omega}) d\omega \end{aligned} \quad (6.4.24)$$

where $\mathcal{G}_{yx}(e^{j\omega})$ is the coherence function between $x(n)$ and $y(n)$.

This important equation indicates that the performance of the optimum filter depends on the coherence between the input and desired response processes. As we recall from Section 5.4, the coherence is a measure of both the noise disturbing the observations and the relative linearity between $x(n)$ and $y(n)$. The optimum filter can reduce the MMSE at a certain band only if there is significant coherence, that is, $\mathcal{G}_{yx}(e^{j\omega}) \simeq 1$. Thus, the optimum filter $H_o(z)$ constitutes the best, in the MMSE sense, linear relationship between the stochastic processes $x(n)$ and $y(n)$. These interpretations apply to causal IIR and FIR optimum filters, even if (6.4.23) and (6.4.24) only hold approximately in these cases (see Section 6.6).

6.5 LINEAR PREDICTION

Linear prediction plays a prominent role in many theoretical, computational, and practical areas of signal processing and deals with the problem of estimating or predicting the value $x(n)$ of a signal at the time instant $n = n_0$, by using a set of other samples from the *same* signal. Although linear prediction is a subject useful in itself, its importance in signal processing is also due, as we will see later, to its use in the development of fast algorithms for optimum filtering and its relation to all-pole signal modeling.

6.5.1 Linear Signal Estimation

Suppose that we are given a set of values $x(n), x(n-1), \dots, x(n-M)$ of a stochastic process and we wish to estimate the value of $x(n-i)$, using a linear combination of the remaining samples. The resulting estimate and the corresponding estimation error are given

by

$$\hat{x}(n-i) \triangleq - \sum_{\substack{k=0 \\ k \neq i}}^M c_k^*(n)x(n-k) \quad (6.5.1)$$

and

$$e^{(i)}(n) \triangleq x(n-i) - \hat{x}(n-i) = \sum_{k=0}^M c_k^*(n)x(n-k) \quad \text{with } c_i(n) \triangleq 1 \quad (6.5.2)$$

where $c_k(n)$ are the coefficients of the estimator as a function of discrete-time index n . The process is illustrated in Figure 6.16.

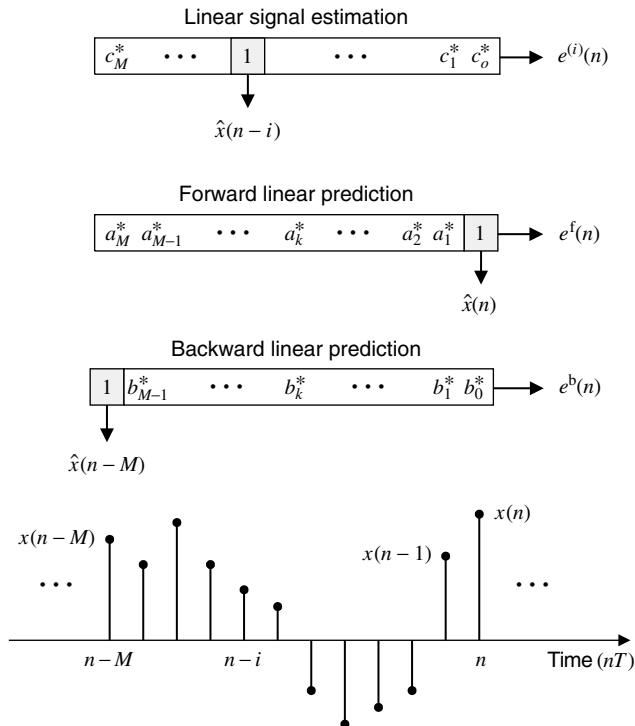


FIGURE 6.16

Illustration showing the samples, estimates, and errors used in linear signal estimation, forward linear prediction, and backward linear prediction.

To determine the MMSE signal estimator, we partition (6.5.2) as

$$\begin{aligned} e^{(i)}(n) &= \sum_{k=0}^{i-1} c_k^*(n)x(n-k) + x(n-i) + \sum_{k=i+1}^M c_k^*(n)x(n-k) \\ &\triangleq \mathbf{c}_1^H(n)\mathbf{x}_1(n) + x(n-i) + \mathbf{c}_2^H(n)\mathbf{x}_2(n) \\ &\triangleq [\tilde{\mathbf{c}}^{(i)}(n)]^H \bar{\mathbf{x}}(n) \end{aligned} \quad (6.5.3)$$

where the partitions of the coefficient and data vectors, around the i th component, are easily defined from the context. To obtain the normal equations and the MMSE for the optimum

linear signal estimator, we note that

$$\text{Desired response} = x(n - i) \quad \text{data vector} = \begin{bmatrix} \mathbf{x}_1(n) \\ \mathbf{x}_2(n) \end{bmatrix}$$

Using (6.4.6) and (6.4.9) or the orthogonality principle, we have

$$\begin{bmatrix} \mathbf{R}_{11}(n) & \mathbf{R}_{12}(n) \\ \mathbf{R}_{12}^T(n) & \mathbf{R}_{22}(n) \end{bmatrix} \begin{bmatrix} \mathbf{c}_1(n) \\ \mathbf{c}_2(n) \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_1(n) \\ \mathbf{r}_2(n) \end{bmatrix} \quad (6.5.4)$$

or more compactly[†]

$$\mathbf{R}^{(i)}(n)\mathbf{c}_o^{(i)}(n) = -\mathbf{d}^{(i)}(n) \quad (6.5.5)$$

$$\text{and} \quad P_o^{(i)}(n) = P_x(n - i) + \mathbf{r}_1^H(n)\mathbf{c}_1(n) + \mathbf{r}_2^H(n)\mathbf{c}_2(n) \quad (6.5.6)$$

where for $j, k = 1, 2$

$$\mathbf{R}_{jk}(n) \triangleq E\{\mathbf{x}_j(n)\mathbf{x}_k^H(n)\} \quad (6.5.7)$$

$$\mathbf{r}_j(n) \triangleq E\{\mathbf{x}_j(n)x^*(n - i)\} \quad (6.5.8)$$

$$P_x(n) = E\{|x(n)|^2\} \quad (6.5.9)$$

For various reasons, to be seen later, we will combine (6.5.4) and (6.5.6) into a single equation. To this end, we note that the correlation matrix of the extended vector

$$\bar{\mathbf{x}}(n) = \begin{bmatrix} \mathbf{x}_1(n) \\ x(n - i) \\ \mathbf{x}_2(n) \end{bmatrix} \quad (6.5.10)$$

can be partitioned as

$$\bar{\mathbf{R}}(n) = E\{\bar{\mathbf{x}}(n)\bar{\mathbf{x}}^H(n)\} = \begin{bmatrix} \mathbf{R}_{11}(n) & \mathbf{r}_1(n) & \mathbf{R}_{12}(n) \\ \mathbf{r}_1^H(n) & P_x(n - i) & \mathbf{r}_2^H(n) \\ \mathbf{R}_{12}^H(n) & \mathbf{r}_2(n) & \mathbf{R}_{22}(n) \end{bmatrix} \quad (6.5.11)$$

with respect to its i th row and i th column. Using (6.5.4), (6.5.6), and (6.5.11), we obtain

$$\bar{\mathbf{R}}(n)\bar{\mathbf{c}}_o^{(i)}(n) = \begin{bmatrix} \mathbf{0} \\ P_o^{(i)}(n) \\ \mathbf{0} \end{bmatrix} \leftarrow i\text{th row} \quad (6.5.12)$$

which completely determines the linear signal estimator $\mathbf{c}^{(i)}(n)$ and the MMSE $P_o^{(i)}(n)$.

If $M = 2L$ and $i = L$, we have a *symmetric linear smoother* $\bar{\mathbf{c}}(n)$ that produces an estimate of the middle sample by using the L past and the L future samples. The above formulation suggests an easy procedure for the computation of the linear signal estimator for any value of i , which is outlined in Table 6.3 and implemented by the function `[ci,Pi]=olsigest(R,i)`. We next discuss two types of linear signal estimation that are of special interest and have their own dedicated notation.

6.5.2 Forward Linear Prediction

One-step *forward linear prediction* (*FLP*) involves the estimation or prediction of the value $x(n)$ of a stochastic process by using a linear combination of the past samples $x(n - 1), \dots, x(n - M)$ (see Figure 6.16). We should stress that in signal processing applications

[†]The minus sign on the right-hand side of the normal equations is the result of arbitrarily setting the coefficient $c_i(n) \triangleq 1$.

Steps for the computation of optimum signal estimators.

-
1. Determine the matrix $\bar{\mathbf{R}}(n)$ of the extended data vector $\bar{\mathbf{x}}(n)$.
 2. Create the $M \times M$ submatrix $\mathbf{R}^{(i)}(n)$ of $\bar{\mathbf{R}}(n)$ by removing its i th row and its i th column.
 3. Create the $M \times 1$ vector $\mathbf{d}^{(i)}(n)$ by extracting the i th column $\bar{\mathbf{d}}^{(i)}(n)$ of $\bar{\mathbf{R}}(n)$ and removing its i th element.
 4. Solve the linear system $\mathbf{R}^{(i)}(n)\mathbf{c}_o^{(i)}(n) = -\mathbf{d}^{(i)}(n)$ to obtain $\mathbf{c}_o^{(i)}(n)$.
 5. Compute the MMSE $P_o^{(i)}(n) = [\bar{\mathbf{d}}^{(i)}(n)]^H \bar{\mathbf{c}}_o^{(i)}(n)$.
-

of linear prediction, what is important is the ability to obtain a good estimate of a sample, pretending that it is unknown, instead of forecasting the future. Thus, the term *prediction* is used more with signal estimation than forecasting in mind. The forward predictor is a linear signal estimator with $i = 0$ and is denoted by

$$\begin{aligned} e^f(n) &\triangleq x(n) + \sum_{k=1}^M a_k^*(n)x(n-k) \\ &= x(n) + \mathbf{a}^H(n)\mathbf{x}(n-1) \end{aligned} \quad (6.5.13)$$

where

$$\mathbf{a}(n) \triangleq [a_1(n) \ a_2(n) \ \cdots \ a_M(n)]^T \quad (6.5.14)$$

is known as the *forward linear predictor* and $a_k(n)$ with $a_0(n) \triangleq 1$ as the FLP error filter. To obtain the normal equations and the MMSE for the optimum FLP, we note that for $i = 0$, (6.5.11) can be written as

$$\bar{\mathbf{R}}(n) = \begin{bmatrix} P_x(n) & \mathbf{r}^{fH}(n) \\ \mathbf{r}^f(n) & \mathbf{R}(n-1) \end{bmatrix} \quad (6.5.15)$$

where

$$\mathbf{R}(n) = E\{\mathbf{x}(n)\mathbf{x}^H(n)\} \quad (6.5.16)$$

and

$$\mathbf{r}^f(n) = E\{\mathbf{x}(n-1)x^*(n)\} \quad (6.5.17)$$

Therefore, (6.5.5) and (6.5.6) give

$$\mathbf{R}(n-1)\mathbf{a}_o(n) = -\mathbf{r}^f(n) \quad (6.5.18)$$

and

$$P_o^f(n) = P_x(n) + \mathbf{r}^{fH}(n)\mathbf{a}_o(n) \quad (6.5.19)$$

or

$$\bar{\mathbf{R}}(n) \begin{bmatrix} 1 \\ \mathbf{a}_o(n) \end{bmatrix} = \begin{bmatrix} P_o^f(n) \\ \mathbf{0} \end{bmatrix} \quad (6.5.20)$$

which completely specifies the FLP parameters.

6.5.3 Backward Linear Prediction

In this case, we want to estimate the sample $x(n-M)$ in terms of the future samples $x(n), x(n-1), \dots, x(n-M+1)$ (see Figure 6.16). The term *backward linear prediction* (BLP) is not accurate but is used since it is an established convention. A more appropriate name might be *postdiction* or *hind sight*. The BLP is basically a linear signal estimator with $i = M$ and is denoted by

$$\begin{aligned} e^b(n) &\triangleq \sum_{k=0}^{M-1} b_k^*(n)x(n-k) + x(n-M) \\ &= \mathbf{b}^H(n)\mathbf{x}(n) + x(n-M) \end{aligned} \quad (6.5.21)$$

where

$$\mathbf{b}(n) \triangleq [b_0(n) \ b_1(n) \ \cdots \ b_{M-1}(n)]^T \quad (6.5.22)$$

is the BLP and $b_k(n)$ with $b_M(n) \triangleq 1$ is the *backward prediction error filter (BPEF)*. For $i = M$, (6.5.11) gives

$$\bar{\mathbf{R}}(n) = \begin{bmatrix} \mathbf{R}(n) & \mathbf{r}^b(n) \\ \mathbf{r}^{bH}(n) & P_x(n - M) \end{bmatrix} \quad (6.5.23)$$

where $\mathbf{r}^b(n) \triangleq E\{\mathbf{x}(n)x^*(n - M)\}$ (6.5.24)

The optimum backward linear predictor is specified by

$$\mathbf{R}(n)\mathbf{b}_o(n) = -\mathbf{r}^b(n) \quad (6.5.25)$$

and the MMSE is

$$P_o^b(n) = P_x(n - M) + \mathbf{r}^{bH}(n)\mathbf{b}_o(n) \quad (6.5.26)$$

and can be put in a single equation as

$$\bar{\mathbf{R}}(n) \begin{bmatrix} \mathbf{b}_o(n) \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ P_o^b(n) \end{bmatrix} \quad (6.5.27)$$

In Table 6.4, we summarize the definitions and design equations for optimum FIR filtering and prediction. Using the entries in this table, we can easily obtain the normal equations and the MMSE for the FLP and BLP from those of the optimum filter.

TABLE 6.4
Summary of the design equations for optimum FIR filtering and prediction.

	Optimum filter	FLP	BLP
Input data vector	$\mathbf{x}(n)$	$\mathbf{x}(n - 1)$	$\mathbf{x}(n)$
Desired response	$y(n)$	$x(n)$	$x(n - M)$
Coefficient vector	$\mathbf{h}(n)$	$\mathbf{a}(n)$	$\mathbf{b}(n)$
Estimation error	$e(n) = y(n) - \mathbf{c}^H(n)\mathbf{x}(n)$	$e^f(n) = x(n) + \mathbf{a}^H(n)\mathbf{x}(n - 1)$	$e^b(n) = x(n - M) + \mathbf{b}^H(n)\mathbf{x}(n)$
Normal equations	$\mathbf{R}(n)\mathbf{c}_o(n) = \mathbf{d}(n)$	$\mathbf{R}(n - 1)\mathbf{a}_o(n) = -\mathbf{r}^f(n)$	$\mathbf{R}(n)\mathbf{b}_o(n) = -\mathbf{r}^b(n)$
MMSE	$P_o^c(n) = P_y(n) - \mathbf{c}_o^H(n)\mathbf{d}(n)$	$P_o^f(n) = P_x(n) + \mathbf{a}^H(n)\mathbf{r}_o^f(n)$	$P_o^b(n) = P_x(n - M) + \mathbf{b}^H(n)\mathbf{r}_o^b(n)$
Required moments	$\mathbf{R}(n) = E\{\mathbf{x}(n)\mathbf{x}^H(n)\}$ $\mathbf{d}(n) = E\{\mathbf{x}(n)y^*(n)\}$	$\mathbf{r}^f(n) = E\{\mathbf{x}(n - 1)x^*(n)\}$	$\mathbf{r}^b(n) = E\{\mathbf{x}(n)x^*(n - M)\}$
Stationary processes	$\mathbf{R}\mathbf{c}_o = \mathbf{d}$, \mathbf{R} is Toeplitz	$\mathbf{R}\mathbf{a}_o = -\mathbf{r}^*$	$\mathbf{R}\mathbf{b}_o = -\mathbf{J}\mathbf{r} \Rightarrow \mathbf{b}_o = \mathbf{J}\mathbf{a}_o^*$

6.5.4 Stationary Processes

If the process $x(n)$ is stationary, then the correlation matrix $\bar{\mathbf{R}}(n)$ does *not* depend on the time n and it is Toeplitz

$$\bar{\mathbf{R}} = \begin{bmatrix} r(0) & r(1) & \cdots & r(M) \\ r^*(1) & r(0) & \cdots & r(M-1) \\ \vdots & \vdots & \ddots & \vdots \\ r^*(M) & r^*(M-1) & \cdots & r(0) \end{bmatrix} \quad (6.5.28)$$

Therefore, all the resulting linear MMSE signal estimators are *time-invariant*. If we define

where $r(l) = E\{x(n)x^*(n-l)\}$, we can easily see that the cross-correlation vectors for the FLP and the BLP are

$$\mathbf{r}^f = [r(1) \ r(2) \ \cdots \ r(M)]^T \quad (6.5.29)$$

and

$$\mathbf{r}^b = E\{\mathbf{x}(n)x^*(n-M)\} = \mathbf{J}\mathbf{r} \quad (6.5.31)$$

where

$$\mathbf{J} = \begin{bmatrix} 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \end{bmatrix}, \quad \mathbf{J}^H \mathbf{J} = \mathbf{J} \mathbf{J}^H = \mathbf{I} \quad (6.5.32)$$

is the exchange matrix that simply reverses the order of the vector elements. Therefore,

$$\mathbf{R}\mathbf{a}_o = -\mathbf{r}^* \quad (6.5.33)$$

$$P_o^f = r(0) + \mathbf{r}^H \mathbf{a}_o \quad (6.5.34)$$

$$\mathbf{R}\mathbf{b}_o = -\mathbf{J}\mathbf{r} \quad (6.5.35)$$

$$P_o^b = r(0) + \mathbf{r}^H \mathbf{J}\mathbf{b}_o \quad (6.5.36)$$

where the Toeplitz matrix \mathbf{R} is obtained from $\bar{\mathbf{R}}$ by deleting the last column and row. Using the centrosymmetry property of symmetric Toeplitz matrices

$$\mathbf{R}\mathbf{J} = \mathbf{J}\mathbf{R}^* \quad (6.5.37)$$

and (6.5.33), we have

$$\mathbf{J}\mathbf{R}^* \mathbf{a}_o^* = -\mathbf{J}\mathbf{r} \quad \text{or} \quad \mathbf{R}\mathbf{J}\mathbf{a}_o^* = -\mathbf{J}\mathbf{r} \quad (6.5.38)$$

Comparing the last equation with (6.5.35), we have

$$\mathbf{b}_o = \mathbf{J}\mathbf{a}_o^* \quad (6.5.39)$$

that is, the BLP coefficient vector is the reverse of the conjugated FLP coefficient vector. Furthermore, from (6.5.34), (6.5.36), and (6.5.39), we have

$$P_o \triangleq P_o^f = P_o^b \quad (6.5.40)$$

that is, the forward and backward prediction error powers are equal.

This remarkable symmetry between the MMSE forward and backward linear predictors holds for stationary processes but disappears for nonstationary processes. Also, we do not have such a symmetry if a criterion other than the MMSE is used and the process to be predicted is non-Gaussian (Weiss 1975; Lawrence 1991).

EXAMPLE 6.5.1. To illustrate the basic ideas in FLP, BLP, and linear smoothing, we consider the second-order estimators for stationary processes.

The augmented equations for the first-order FLP are ($r(o)$ is always real)

$$\begin{bmatrix} r(0) & r(1) \\ r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \end{bmatrix} = \begin{bmatrix} P_1^f \\ 0 \end{bmatrix}$$

and they can be solved by using Cramer's rule. Indeed, we obtain

$$a_0^{(1)} = \frac{\det \begin{bmatrix} P_1^f & r(1) \\ 0 & r(0) \end{bmatrix}}{\det \mathbf{R}_2} = \frac{r(0)P_1^f}{\det \mathbf{R}_2} = 1 \Rightarrow P_1^f = \frac{\det \mathbf{R}_2}{\det \mathbf{R}_1} = \frac{r^2(0) - |r(1)|^2}{r(0)}$$

and

$$a_1^{(1)} = \frac{\det \begin{bmatrix} r(0) & P_1^f \\ r^*(1) & 0 \end{bmatrix}}{\det \mathbf{R}_2} = \frac{-P_1^f r^*(1)}{\det \mathbf{R}_2} = -\frac{r^*(1)}{r(0)}$$

for the MMSE and the FLP. For the second-order case we have

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r^*(1) & r(0) & r(1) \\ r^*(2) & r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} = \begin{bmatrix} P_2^f \\ 0 \\ 0 \end{bmatrix}$$

whose solution is

$$a_0^{(2)} = \frac{P_2^f \det \mathbf{R}_2}{\det \mathbf{R}_3} = 1 \Rightarrow P_2^f = \frac{\det \mathbf{R}_3}{\det \mathbf{R}_2}$$

$$a_1^{(2)} = \frac{-P_2^f \det \begin{bmatrix} r^*(1) & r(1) \\ r^*(2) & r(0) \end{bmatrix}}{\det \mathbf{R}_3} = \frac{-\det \begin{bmatrix} r^*(1) & r(1) \\ r^*(2) & r(0) \end{bmatrix}}{\det \mathbf{R}_2} = \frac{r(1)r^*(2) - r(0)r^*(1)}{r^2(0) - |r(1)|^2}$$

and

$$a_2^{(2)} = \frac{P_2^f \det \begin{bmatrix} r^*(1) & r(0) \\ r^*(2) & r^*(1) \end{bmatrix}}{\det \mathbf{R}_3} = \frac{\det \begin{bmatrix} r^*(1) & r(0) \\ r^*(2) & r^*(1) \end{bmatrix}}{\det \mathbf{R}_2} = \frac{[r^*(1)]^2 - r(0)r^*(2)}{r^2(0) - |r(1)|^2}$$

Similarly, for the BLP

$$\begin{bmatrix} r(0) & r(1) \\ r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} b_0^{(1)} \\ b_1^{(1)} \end{bmatrix} = \begin{bmatrix} 0 \\ P_1^b \end{bmatrix}$$

where $b_1^{(1)} = 1$, we obtain

$$P_1^b = \frac{\det \mathbf{R}_2}{\det \mathbf{R}_1} \quad \text{and} \quad b_0^{(1)} = -\frac{r(1)}{r(0)}$$

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r^*(1) & r(0) & r(1) \\ r^*(2) & r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} b_0^{(2)} \\ b_1^{(2)} \\ b_2^{(2)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ P_2^b \end{bmatrix}$$

$$P_2^b = \frac{\det \mathbf{R}_3}{\det \mathbf{R}_2} \quad b_1^{(2)} = \frac{r^*(1)r(2) - r(0)r(1)}{r^2(0) - |r(1)|^2} \quad b_0^{(2)} = \frac{r^2(1) - r(0)r(2)}{r^2(0) - |r(1)|^2}$$

We note that

$$P_1^f = P_1^b \quad a_1^{(1)} = b_0^{(1)*}$$

and

$$P_2^f = P_2^b \quad a_1^{(2)*} = b_1^{(2)*} \quad a_2^{(2)*} = b_0^{(2)*}$$

which is a result of the stationarity of $x(n)$ or equivalently of the Toeplitz structure of \mathbf{R}_m .

For the linear signal estimator, we have

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r^*(1) & r(0) & r(1) \\ r^*(2) & r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} c_0^{(2)} \\ c_1^{(2)} \\ c_2^{(2)} \end{bmatrix} = \begin{bmatrix} 0 \\ P_2 \\ 0 \end{bmatrix}$$

with $c_1^{(2)} = 1$. Using Cramer's rule, we obtain

$$P_2 = \frac{\det \mathbf{R}_3}{\det \mathbf{R}_3^{(2)}}$$

$$c_0^{(2)} = \frac{-P_2 \det \begin{bmatrix} r(1) & r(2) \\ r^*(1) & r(0) \end{bmatrix}}{\det \mathbf{R}_3} = -\frac{\det \begin{bmatrix} r(1) & r(2) \\ r^*(1) & r(0) \end{bmatrix}}{\det \mathbf{R}_3^{(2)}} = \frac{r^*(1)r(2) - r(0)r(1)}{r^2(0) - |r(1)|^2}$$

$$c_2^{(2)} = \frac{-P_2 \det \begin{bmatrix} r(0) & r(1) \\ r^*(2) & r^*(1) \end{bmatrix}}{\det \mathbf{R}_3} = -\frac{\det \begin{bmatrix} r(0) & r(1) \\ r^*(2) & r^*(1) \end{bmatrix}}{\det \mathbf{R}_3^{(2)}} = \frac{r(1)r^*(2) - r(0)r^*(1)}{r^2(0) - |r(1)|^2}$$

from which we see that $c_0^{(2)} = c_2^{(2)*}$; that is, we have a linear phase estimator.

6.5.5 Properties

Linear signal estimators and predictors have some interesting properties that we discuss next.

PROPERTY 6.5.1. If the process $x(n)$ is stationary, then the symmetric, linear smoother has linear phase.

Proof. Using the centrosymmetry property $\bar{\mathbf{R}}\mathbf{J} = \mathbf{J}\bar{\mathbf{R}}^*$ and (6.5.12) for $M = 2L, i = L$, we obtain

$$\bar{\mathbf{c}} = \mathbf{J}\bar{\mathbf{c}}^* \quad (6.5.41)$$

that is, the symmetric, linear smoother has even symmetry and, therefore, has linear phase (see Problem 6.12).

PROPERTY 6.5.2. If the process $x(n)$ is stationary, the forward prediction error filter (PEF) $1, a_1, a_2, \dots, a_M$ is minimum-phase and the backward PEF $b_0, b_1, \dots, b_{M-1}, 1$ is maximum-phase.

Proof. The system function of the M th-order forward PEF can be factored as

$$A(z) = 1 + \sum_{k=1}^M a_k^* z^{-k} = G(z)(1 - qz^{-1})$$

where q is a zero of $A(z)$ and

$$G(z) = 1 + \sum_{k=1}^{M-1} g_k z^{-k}$$

is an $(M-1)$ st-order filter. The filter $A(z)$ can be implemented as the cascade connection of the filters $G(z)$ and $1 - qz^{-1}$ (see Figure 6.17). The output $s(n)$ of $G(z)$ is

$$s(n) = x(n) + g_1 x(n-1) + \cdots + g_{M-1} x(n-M+1)$$

and it is easy to see that

$$E\{s(n-1)e^f(n)\} = 0 \quad (6.5.42)$$

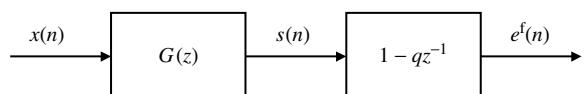


FIGURE 6.17
The prediction error filter with one zero factored out.

because $E\{x(n-k)e^f(n)\} = 0$ for $1 \leq k \leq M$. Since the output of the second filter can be expressed as

$$e^f(n) = s(n) - qs(n-1)$$

we have

$$E\{s(n-1)e^f(n)\} = E\{s(n-1)s^*(n)\} - q^*E\{s(n-1)s^*(n-1)\} = 0$$

which implies that

$$q = \frac{r_s(-1)}{r_s(0)} \Rightarrow |q| \leq 1$$

because q is equal to the normalized autocorrelation of $s(n)$. If the process $x(n)$ is not predictable, that is, $E\{|e^f(n)|^2\} \neq 0$, we have

$$\begin{aligned} E\{|e^f(n)|^2\} &= E\{e^f(n)[s^*(n) - q^*s^*(n-1)]\} \\ &= E\{e^f(n)s^*(n)\} \quad \text{due to (6.5.42)} \\ &= E\{[s(n) - qs(n-1)]s^*(n)\} \\ &= r_s(0)(1 - |q|^2) \neq 0 \end{aligned}$$

which implies that

$$|q| < 1$$

that is, the zero q of the forward PEF filter is strictly inside the unit circle. Repeating this process, we can show that all zeros of $A(z)$ are inside the unit circle; that is, $A(z)$ is minimum-phase. This proof was presented in Vaidyanathan et al. (1996). The property $\mathbf{b} = \mathbf{J}\mathbf{a}^*$ is equivalent to

$$B(z) = z^{-M} A^* \left(\frac{1}{z^*} \right)$$

which implies that $B(z)$ is a maximum-phase filter (see Section 2.4).

PROPERTY 6.5.3. The forward and backward prediction error filters can be expressed in terms of the eigenvalues $\bar{\lambda}_i$ and the eigenvectors $\bar{\mathbf{q}}_i$ of the correlation matrix $\bar{\mathbf{R}}(n)$ as follows

$$\begin{bmatrix} 1 \\ \mathbf{a}_o(n) \end{bmatrix} = P_o^f(n) \sum_{i=1}^{M+1} \frac{1}{\bar{\lambda}_i} \bar{\mathbf{q}}_i \bar{q}_{i,1}^* \quad (6.5.43)$$

$$\text{and} \quad \begin{bmatrix} \mathbf{b}_o(n) \\ 1 \end{bmatrix} = P_o^b(n) \sum_{i=1}^{M+1} \frac{1}{\bar{\lambda}_i} \bar{\mathbf{q}}_i \bar{q}_{i,M+1}^* \quad (6.5.44)$$

where $\bar{q}_{i,1}$ and $\bar{q}_{i,M+1}$ are the first and last components of $\bar{\mathbf{q}}_i$. The first equation of (6.5.43) and the last equation in (6.5.44) can be solved to provide the MMSEs $P_o^f(n)$ and $P_o^b(n)$, respectively.

Proof. See Problem 6.13.

PROPERTY 6.5.4. Let $\bar{\mathbf{R}}^{-1}(n)$ be the inverse of the correlation matrix $\bar{\mathbf{R}}(n)$. Then, the inverse of the i th element of the i th column of $\bar{\mathbf{R}}^{-1}(n)$ is equal to the MMSE $P^{(i)}(n)$, and the i th column normalized by the i th element is equal to $\mathbf{c}^{(i)}(n)$.

Proof. See Problem 6.14.

PROPERTY 6.5.5. The MMSE prediction errors can be expressed as

$$P_o^f(n) = \frac{\det \bar{\mathbf{R}}(n)}{\det \mathbf{R}(n-1)} \quad P_o^b(n) = \frac{\det \bar{\mathbf{R}}(n)}{\det \mathbf{R}(n)} \quad (6.5.45)$$

Proof. Problem 6.17.

The previous concepts are illustrated in the following example.

EXAMPLE 6.5.2. A random sequence $x(n)$ is generated by passing the white Gaussian noise process $w(n) \sim WN(0, 1)$ through the filter

$$x(n) = w(n) + \frac{1}{2}w(n-1)$$

Determine the second-order FLP, BLP, and symmetric linear signal smoother.

Solution. The complex power spectrum is

$$R(z) = H(z)H(z^{-1}) = (1 + \frac{1}{2}z^{-1})(1 + \frac{1}{2}z) = \frac{1}{2}z + \frac{5}{4} + \frac{1}{2}z^{-1}$$

Therefore, the autocorrelation sequence is equal to $r(0) = \frac{5}{4}$, $r(\pm 1) = \frac{1}{2}$, $r(l) = 0$ for $|l| \geq 2$. Since the power spectrum $R(e^{j\omega}) = \frac{5}{4} + \cos \omega > 0$ for all ω , the autocorrelation matrix is positive definite. The same is true of any principal submatrix. To determine the second-order linear signal estimators, we start with the matrix

$$\bar{\mathbf{R}} = \begin{bmatrix} \frac{5}{4} & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{5}{4} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{5}{4} \end{bmatrix}$$

and follow the procedure outlined in Section 6.5.1 or use the formulas in Table 6.3. The results are

Forward linear prediction ($i = 0$): $\{a_k\} \rightarrow \{1, -0.476, 0.190\}$ $P_o^f = 1.0119$

Symmetric linear smoothing ($i = 1$): $\{c_k\} \rightarrow \{-0.4, 1, -0.4\}$ $P_o^s = 0.8500$

Backward linear prediction ($i = 2$): $\{b_k\} \rightarrow \{0.190, -0.476, 1\}$ $P_o^b = 1.0119$

The inverse of the correlation matrix $\bar{\mathbf{R}}$ is

$$\bar{\mathbf{R}}^{-1} = \begin{bmatrix} 0.9882 & -0.4706 & 0.1882 \\ -0.4706 & 1.1765 & -0.4706 \\ 0.1882 & -0.4706 & 0.9882 \end{bmatrix}$$

and we see that dividing the first, second, and third columns by 0.9882, 1.1765, and 0.9882 provides the forward PEF, the symmetric linear smoothing filter, and the backward PEF, respectively. The inverses of the diagonal elements provide the MMSEs P_o^f , P_o^s , and P_o^b . The reader can easily see, by computing the zeros of the corresponding system functions, that the FLP is minimum-phase, the BLP is maximum-phase, and the symmetric linear smoother is mixed-phase. It is interesting to note that the smoother performs better than either of the predictors.

6.6 OPTIMUM INFINITE IMPULSE RESPONSE FILTERS

So far we have dealt with optimum FIR filters and predictors for nonstationary and stationary processes. In this section, we consider the design of optimum IIR filters for stationary stochastic processes. For nonstationary processes, the theory becomes very complicated. The Wiener-Hopf equations for optimum IIR filters are the same for FIR filters; only the limits in the convolution summation and the range of values for which the normal equations hold are different. Both are determined by the limits of summation in the filter convolution equation. We can easily see from (6.4.16) and (6.4.17), or by applying the orthogonality principle (6.2.41), that the optimum IIR filter

$$\hat{y}(n) = \sum_k h_o(k)x(n-k) \quad (6.6.1)$$

is specified by the Wiener-Hopf equations

$$\sum_k h_o(k)r_x(m-k) = r_{yx}(m) \quad (6.6.2)$$

and the MMSE is given by

$$P_o = r_y(0) - \sum_k h_o(k) r_{yx}^*(k) \quad (6.6.3)$$

where $r_x(l)$ is the autocorrelation of the input stochastic process $x(n)$ and $r_{yx}(l)$ is the cross-correlation between the desired response process $y(n)$ and $x(n)$. We assume that the processes $x(n)$ and $y(n)$ are jointly wide-sense stationary with zero mean values.

The range of summation in the above equations includes all the nonzero coefficients of the impulse response of the filter. The range of k in (6.6.1) determines the number of unknowns and the number of equations, that is, the range of m . For IIR filters, we have an infinite number of equations and unknowns, and thus only analytical solutions for (6.6.2) are possible. The key to analytical solutions is that the left-hand side of (6.6.2) can be expressed as the convolution of $h_o(m)$ with $r_x(m)$, that is,

$$h_o(m) * r_x(m) = r_{yx}(m) \quad (6.6.4)$$

which is a *convolutional equation* that can be solved by using the z -transform. The complexity of the solution depends on the range of m .

The formula for the MMSE is the same for any filter, either FIR or IIR. Indeed, using Parseval's theorem and (6.6.3), we obtain

$$P_o = r_y(0) - \frac{1}{2\pi j} \oint_C H_o(z) R_{yx}^* \left(\frac{1}{z^*} \right) z^{-1} dz \quad (6.6.5)$$

where $H_o(z)$ is the system function of the optimum filter and $R_{yx}(z) = \mathcal{Z}\{r_{yx}(l)\}$. The power P_y can be computed by

$$P_y = r_y(0) = \frac{1}{2\pi j} \oint_C R_y(z) z^{-1} dz \quad (6.6.6)$$

where $R_y(z) = \mathcal{Z}\{r_y(l)\}$. Combining (6.6.5) with (6.6.6), we obtain

$$P_o = \frac{1}{2\pi j} \oint_C [R_y(z) - H_o(z) R_{yx}^* \left(\frac{1}{z^*} \right)] z^{-1} dz \quad (6.6.7)$$

which expresses the MMSE in terms of z -transforms. To obtain the MMSE in the frequency domain, we replace z by $e^{j\omega}$. For example, (6.6.5) becomes

$$P_o = r_y(0) - \frac{1}{2\pi} \int_{-\pi}^{\pi} H_o(e^{j\omega}) R_{yx}^*(e^{j\omega}) d\omega$$

where $H_o(e^{j\omega})$ is the frequency response of the optimum filter.

6.6.1 Noncausal IIR Filters

For the noncausal IIR filter

$$\hat{y}(n) = \sum_{k=-\infty}^{\infty} h_{nc}(k) x(n-k) \quad (6.6.8)$$

the range of the Wiener-Hopf equations (6.6.2) is $-\infty < m < \infty$ and can be easily solved by using the convolution property of the z -transform. This gives

$$H_{nc}(z) R_x(z) = R_{yx}(z)$$

$$\text{or} \quad H_{nc}(z) = \frac{R_{yx}(z)}{R_x(z)} \quad (6.6.9)$$

where $H_{nc}(z)$ is the system function of the optimum filter, $R_x(z)$ is the complex PSD of $x(n)$, and $R_{yx}(z)$ is the complex cross-PSD between $y(n)$ and $x(n)$.

For the causal IIR filter

$$\hat{y}(n) = \sum_{k=0}^{\infty} h_c(k)x(n-k) \quad (6.6.10)$$

the Wiener-Hopf equations (6.6.2) hold only for m in the range $0 \leq m < \infty$. Since the sequence $r_y(m)$ can be expressed as the convolution of $h_o(m)$ and $r_x(m)$ only for $m \geq 0$, we cannot solve (6.6.2) using the z -transform. However, a simple solution is possible using the spectral factorization theorem.[†] This approach was introduced for continuous-time processes in Bode and Shannon (1950) and Zadeh and Ragazzini (1950). It is based on the following two observations:

1. The solution of the Wiener-Hopf equations is trivial if the input is white.
2. Any regular process can be transformed to an equivalent white process.

White input processes. We first note that if the process $x(n)$ is white noise, the solution of the Wiener-Hopf equations is trivial. Indeed, if

$$r_x(l) = \sigma_x^2 \delta(l)$$

Then Equation (6.6.4) gives

$$h_c(m) * \delta(m) = \frac{r_{yx}(m)}{\sigma_x^2} \quad 0 \leq m < \infty$$

which implies that

$$h_c(m) = \begin{cases} \frac{1}{\sigma_x^2} r_{yx}(m) & 0 \leq m < \infty \\ 0 & m < 0 \end{cases} \quad (6.6.11)$$

because the filter is causal. The system function of the optimum filter is given by

$$H_c(z) = \frac{1}{\sigma_x^2} [R_{yx}(z)]_+ \quad (6.6.12)$$

where $[R_{yx}(z)]_+ \triangleq \sum_{l=0}^{\infty} r_{yx}(l)z^{-l}$ (6.6.13)

is the one-sided z -transform of the two-sided sequence $r_{yx}(l)$. The MMSE is given by

$$P_c = r_y(0) - \frac{1}{\sigma_x^2} \sum_{k=0}^{\infty} |r_{yx}(k)|^2 \quad (6.6.14)$$

which follows from (6.6.3) and (6.6.11).

Regular input processes. The PSD of a regular process can be factored as

$$R_x(z) = \sigma_x^2 H_x(z) H_x^* \left(\frac{1}{z^*} \right) \quad (6.6.15)$$

where $H_x(z)$ is the innovations filter (see Section 4.1). The innovations process

$$w(n) = x(n) - \sum_{k=1}^{\infty} h_x(k)w(n-k) \quad (6.6.16)$$

[†] An analogous matrix-based approach is extensively used in Chapter 7 for the design and implementation of optimum FIR filters.

is white and *linearly equivalent* to the input process $x(n)$. Therefore, linear estimation of $y(n)$ based on $x(n)$ is equivalent to linear estimation of $y(n)$ based on $w(n)$. The optimum filter that estimates $y(n)$ from $x(n)$ is obtained by cascading the whitening filter $1/H_x(z)$ with the optimum filter that estimates $y(n)$ from $w(n)$ (see Figure 6.18). Since $w(n)$ is white, the optimum filter for estimating $y(n)$ from $w(n)$ is

$$H'_c(z) = \frac{1}{\sigma_x^2} [R_{yw}(z)]_+ \quad (6.6.17)$$

where $[R_{yw}(z)]_+$ is the one-sided z -transform of $r_{yw}(l)$. To express $H'_c(z)$ in terms of $R_{yx}(z)$, we need the relationship between $R_{yw}(z)$ and $R_{yx}(z)$. From

$$x(n) = \sum_{k=0}^{\infty} h_x(k)w(n-k)$$

if we recall that $r_{yx}(l) = r_{yw}(l) * h_x^*(-l)$, we obtain

$$\begin{aligned} E\{y(n)x^*(n-l)\} &= \sum_{k=0}^{\infty} h_x^*(k)E\{y(n)w^*(n-l-k)\} \\ \text{or} \quad r_{yx}(l) &= \sum_{k=0}^{\infty} h_x^*(k)r_{yw}(l+k) \end{aligned} \quad (6.6.18)$$

Taking the z -transform of the above equation leads to

$$R_{yw}(z) = \frac{R_{yx}(z)}{H_x^*(1/z^*)} \quad (6.6.19)$$

which, combined with (6.6.17), gives

$$H'_c(z) = \frac{1}{\sigma_x^2} \left[\frac{R_{yx}(z)}{H_x^*(1/z^*)} \right]_+ \quad (6.6.20)$$

which is the causal optimum filter for the estimation of $y(n)$ from $w(n)$. The optimum filter for estimating $y(n)$ from $x(n)$ is

$$H_c(z) = \frac{1}{\sigma_x^2 H_x(z)} \left[\frac{R_{yx}(z)}{H_x^*(1/z^*)} \right]_+ \quad (6.6.21)$$

which is causal since it is the cascade connection of two causal filters [see Figure 6.19(a)].

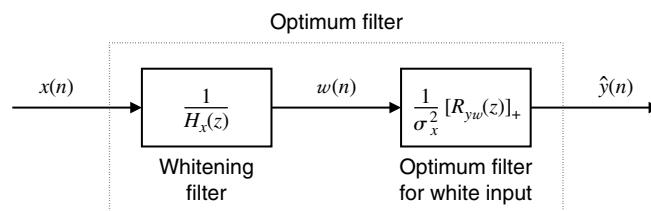


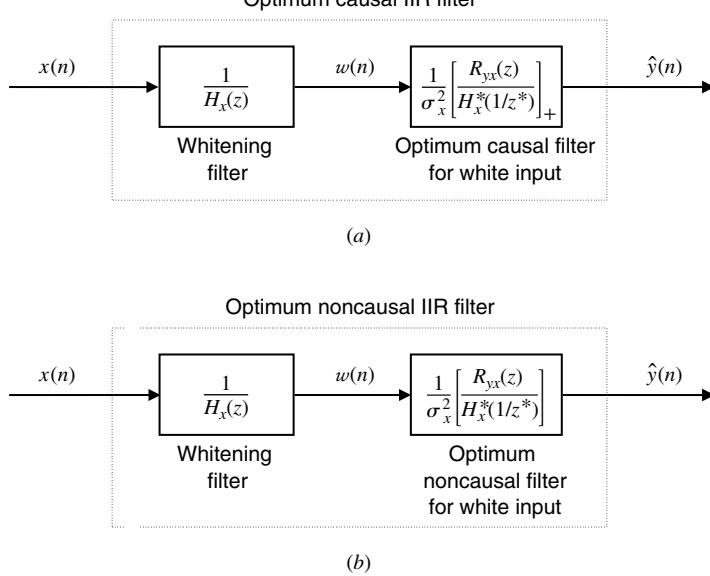
FIGURE 6.18

Optimum causal IIR filter design by the spectral factorization method.

The MMSE from (6.6.3) can also be expressed as

$$P_c = r_y(0) - \frac{1}{\sigma_x^2} \sum_{k=0}^{\infty} |r_{yw}(k)|^2 \quad (6.6.22)$$

which shows that the MMSE decreases as we increase the order of the filter. Table 6.5 summarizes the equations required for the design of optimum FIR and IIR filters.

**FIGURE 6.19**

Comparison of causal and noncausal IIR optimum filters.

TABLE 6.5
Design of FIR and IIR optimum filters for stationary processes.

Filter type	Solution	Required quantities
FIR	$e(n) = y(n) - \mathbf{c}_o^H \mathbf{x}(n)$ $\mathbf{c}_o = \mathbf{R}^{-1} \mathbf{d}$ $P_o = r_y(0) - \mathbf{d}^H \mathbf{c}_o$	$\mathbf{R} = [r_x(m-k)], \mathbf{d} = [r_{yx}(m)]$ $0 \leq k, m \leq M-1, M = \text{finite}$
Noncausal IIR	$H_{nc}(z) = \frac{R_{yx}(z)}{R_x(z)}$ $P_{nc} = r_y(0) - \sum_{k=-\infty}^{\infty} h_{nc}(k)r_{yx}^*(k)$	$R_x(z) = \mathcal{Z}\{r_x(l)\}$ $R_{yx}(z) = \mathcal{Z}\{r_{xy}^*(l)\}$
Causal IIR	$H_c(z) = \frac{1}{\sigma_x^2 H_x(z)} \left[\frac{R_{yx}(z)}{H_x^*(1/z^*)} \right]_+$ $P_c = r_y(0) - \sum_{k=0}^{\infty} h_{nc}(k)r_{yx}^*(k)$	$R_x(z) = \sigma_x^2 H_x(z) H_x^*(1/z^*)$ $R_{yx}(z) = \mathcal{Z}\{r_{xy}(l)\}$

Finally, since the equation for the noncausal IIR filter can be written as

$$H_{nc}(z) = \frac{1}{\sigma_x^2 H_x(z)} \frac{R_{yx}(z)}{H_x^*(1/z^*)} \quad (6.6.23)$$

we see that the only difference from the causal filter is that the noncausal filter includes both the causal and noncausal parts of $R_{yx}(z)/H_x(z^{-1})$ [see Figure 6.19(b)]. By using the innovations process $w(n)$, the MMSE can be expressed as

$$P_{nc} = r_y(0) - \frac{1}{\sigma_x^2} \sum_{k=-\infty}^{\infty} |r_{yw}(k)|^2 \quad (6.6.24)$$

and is known as the *irreducible MMSE* because it is the best performance that can be achieved by a linear filter. Indeed, since $|r_{yw}(k)| \geq 0$, every coefficient we add to the optimum filter can help to reduce the MMSE.

6.6.3 Filtering of Additive Noise

To illustrate the optimum filtering theory developed above, we consider the problem of estimating a “useful” or desired signal $y(n)$ that is corrupted by additive noise $v(n)$. The goal is to find an optimum filter that extracts the signal $y(n)$ from the noisy observations

$$x(n) = y(n) + v(n) \quad (6.6.25)$$

given that $y(n)$ and $v(n)$ are uncorrelated processes with known autocorrelation sequences $r_y(l)$ and $r_v(l)$.

To design the optimum filter, we need the autocorrelation $r_x(l)$ of the input signal $x(n)$ and the cross-correlation $r_{yx}(l)$ between the desired response $y(n)$ and the input signal $x(n)$. Using (6.6.25), we find

$$r_x(l) = E\{x(n)x^*(n-l)\} = r_y(l) + r_v(l) \quad (6.6.26)$$

and

$$r_{yx}(l) = E\{y(n)x^*(n-l)\} = r_y(l) \quad (6.6.27)$$

because $y(n)$ and $v(n)$ are uncorrelated.

The design of optimum IIR filters requires the functions $R_x(z)$ and $R_{yx}(z)$. Taking the z -transform of (6.6.26) and (6.6.27), we obtain

$$R_x(z) = R_y(z) + R_v(z) \quad (6.6.28)$$

and

$$R_{yx}(z) = R_y(z) \quad (6.6.29)$$

The noncausal optimum filter is given by

$$H_{nc}(z) = \frac{R_{yx}(z)}{R_x(z)} = \frac{R_y(z)}{R_y(z) + R_v(z)} \quad (6.6.30)$$

which for $z = e^{j\omega}$ shows that, for those values of ω for which $|R_y(e^{j\omega})| \gg |R_v(e^{j\omega})|$, that is, for high SNR, we have $|H_{nc}(e^{j\omega})| \approx 1$. In contrast, if $|R_y(e^{j\omega})| \ll |R_v(e^{j\omega})|$, that is, for low SNR, we have $|H_{nc}(e^{j\omega})| \approx 0$. Thus, the optimum filter “passes” its input in bands with high SNR and attenuates it in bands with low SNR, as we would expect intuitively.

Substituting (6.6.30) into (6.6.7), we obtain for real-valued signals

$$P_{nc} = \frac{1}{2\pi j} \oint_C \frac{R_y(z)R_v(z)}{R_y(z) + R_v(z)} z^{-1} dz \quad (6.6.31)$$

which provides an expression for the MMSE that does not require knowledge of the optimum filter.

We next illustrate the design of optimum filters for the reduction of additive noise with a detailed numerical example.

EXAMPLE 6.6.1. In this example we illustrate the design of an optimum IIR filter to extract a random signal with known autocorrelation sequence

$$r_y(l) = \alpha^{|l|} \quad -1 < \alpha < 1 \quad (6.6.32)$$

which is corrupted by additive white noise with autocorrelation

$$r_v(l) = \sigma_v^2 \delta(l) \quad (6.6.33)$$

The processes $y(n)$ and $v(n)$ are uncorrelated.

Required statistical moments. The input to the filter is the signal $x(n) = y(n) + v(n)$ and the desired response, the signal $y(n)$. The first step in the design is to determine the required second-order moments, that is, the autocorrelation of the input process and the cross-correlation between input and desired response. Substituting into (6.6.26) and (6.6.27), we have

$$r_x(l) = \alpha^{|l|} + \sigma_v^2 \delta(l) \quad (6.6.34)$$

and

$$r_{yx}(l) = \alpha^{|l|} \quad (6.6.35)$$

To simplify the derivations and deal with “nice, round” numbers, we choose $\alpha = 0.8$ and $\sigma_v^2 = 1$. Then the complex power spectral densities of $y(n)$, $v(n)$, and $x(n)$ are

$$R_y(z) = \frac{\left(\frac{3}{5}\right)^2}{(1 - \frac{4}{5}z^{-1})(1 - \frac{4}{5}z)} \quad \frac{4}{5} < |z| < \frac{5}{4} \quad (6.6.36)$$

$$R_v(z) = \sigma_v^2 = 1 \quad (6.6.37)$$

and

$$R_x(z) = \frac{8}{5} \frac{(1 - \frac{1}{2}z^{-1})(1 - \frac{1}{2}z)}{(1 - \frac{4}{5}z^{-1})(1 - \frac{4}{5}z)} \quad (6.6.38)$$

respectively.

Noncausal filter. Using (6.6.9), (6.6.29), (6.6.36), and (6.6.38), we obtain

$$H_{nc}(z) = \frac{R_{yx}(z)}{R_x(z)} = \frac{9}{40} \frac{1}{(1 - \frac{1}{2}z^{-1})(1 - \frac{1}{2}z)} \quad \frac{1}{2} < |z| < 2$$

Evaluating the inverse the z -transform we have

$$h_{nc}(n) = \frac{3}{10} \left(\frac{1}{2}\right)^{|n|} \quad -\infty < n < \infty$$

which clearly corresponds to a noncausal filter. From (6.6.3), the MMSE is

$$P_{nc} = 1 - \frac{3}{10} \sum_{k=-\infty}^{\infty} \left(\frac{1}{2}\right)^{|k|} \left(\frac{4}{5}\right)^{|k|} = \frac{3}{10} \quad (6.6.39)$$

and provides the irreducible MMSE.

Causal filter. To find the optimum causal filter, we need to perform the spectral factorization

$$R_x(z) = \sigma_x^2 H_x(z) H_x(z^{-1})$$

which is provided by (6.6.38) with

$$\sigma_x^2 = \frac{8}{5} \quad (6.6.40)$$

and

$$H_x(z) = \frac{1 - \frac{1}{2}z^{-1}}{1 - \frac{4}{5}z^{-1}} \quad (6.6.41)$$

Thus,

$$R_{yw}(z) = \frac{R_{yx}(z)}{H_x(z^{-1})} = \frac{0.36}{(1 - \frac{4}{5}z^{-1})(1 - \frac{1}{2}z)} = \frac{0.6}{1 - \frac{4}{5}z^{-1}} + \frac{0.3z}{1 - \frac{1}{2}z} \quad (6.6.42)$$

where the first term (causal) converges for $|z| > \frac{4}{5}$ and the second term (noncausal) converges for $|z| < 2$. Hence, taking the causal part

$$\left[\frac{R_{yx}(z)}{H_x(z^{-1})} \right]_+ = \frac{\frac{3}{5}}{1 - \frac{4}{5}z^{-1}}$$

and substituting into (6.6.21), we obtain the causal optimum filter

$$H_c(z) = \frac{5}{8} \left(\frac{1 - \frac{4}{5}z^{-1}}{1 - \frac{1}{2}z^{-1}} \frac{\frac{3}{5}}{1 - \frac{4}{5}z^{-1}} \right) = \frac{3}{8} \left(\frac{1}{1 - \frac{1}{2}z^{-1}} \right) \quad |z| < \frac{1}{2} \quad (6.6.43)$$

The impulse response is

$$h_c(n) = \frac{3}{8} \left(\frac{1}{2}\right)^n u(n)$$

which corresponds to a causal and stable IIR filter. The MMSE is

$$P_c = r_y(0) - \sum_{k=0}^{\infty} h_c(k) r_{yx}(k) = 1 - \frac{3}{8} \sum_{k=0}^{\infty} \left(\frac{1}{2}\right)^k \left(\frac{4}{5}\right)^k = \frac{3}{8} \quad (6.6.44)$$

which is, as expected, larger than P_{nc} .

From (6.6.43), we see that the optimum causal filter is a first-order recursive filter that can be implemented by the difference equation

$$\hat{y}(n) = \frac{1}{2}\hat{y}(n-1) + \frac{3}{8}x(n)$$

In general, this is possible only when $H_c(z)$ is a rational function.

Computation of MMSE using the innovation. We next illustrate how to find the MMSE by using the cross-correlation sequence $r_{yw}(l)$. From (6.6.42), we obtain

$$r_{yw}(l) = \begin{cases} \frac{3}{5}(\frac{4}{5})^l & l \geq 0 \\ \frac{3}{5}2^l & l < 0 \end{cases} \quad (6.6.45)$$

which, in conjunction with (6.6.22) and (6.6.24), gives

$$P_c = r_y(0) - \frac{1}{\sigma_x^2} \sum_{k=0}^{\infty} r_{yw}^2(k) = 1 - \frac{5}{8}(\frac{3}{5})^2 \sum_{k=0}^{\infty} (\frac{4}{5})^{2k} = \frac{3}{8}$$

$$\text{and } P_{nc} = r_y(0) - \frac{1}{\sigma_x^2} \left[\sum_{k=0}^{\infty} r_{yw}^2(k) - \sum_{k=-\infty}^{-1} r_{yw}^2(k) \right] = \frac{3}{10}$$

which agree with (6.6.44) and (6.6.39).

Noncausal smoothing filter. Suppose now that we want to estimate the value $y(n+D)$ of the desired response from the data $x(n)$, $-\infty < n < \infty$. Since

$$E\{y(n+D)x(n-l)\} = r_{yx}(n+D) \quad (6.6.46)$$

$$\text{and } \mathcal{Z}\{r_{yx}(n+D)\} = z^D R_{yx}(z) \quad (6.6.47)$$

the noncausal Wiener smoothing filter is

$$H_{nc}^D(z) = \frac{z^D R_{yx}(z)}{R_x(z)} = \frac{z^D R_y(z)}{R_x(z)} = z^D H_{nc}(z) \quad (6.6.48)$$

$$h_{nc}^D(n) = h_{nc}(n+D) \quad (6.6.49)$$

The MMSE is

$$P_{nc}^D = r_y(0) - \sum_{k=-\infty}^{\infty} h_{nc}(k+D)r_{yx}(k+D) = P_{nc} \quad (6.6.50)$$

which is independent of the time shift D .

Causal prediction filter. We estimate the value $y(n+D)$ ($D > 0$) of the desired response using the data $x(k)$, $-\infty < k \leq n$. The whitening part of the causal prediction filter does not depend on $y(n)$ and is still given by (6.6.41). The coloring part depends on $y(n+D)$ and is given by $R'_{yw}(z) = z^D R_{yw}(z)$ or $r'_{yw}(l) = r_{yw}(l+D)$. Taking into consideration that $D > 0$, we can show (see Problem 6.31) that the system function and the impulse response of the causal Wiener predictor are

$$H_c^{[D]}(z) = \frac{5}{8} \left(\frac{1 - \frac{4}{5}z^{-1}}{1 - \frac{1}{2}z^{-1}} \right) \left[\frac{\frac{3}{5}(\frac{4}{5})^D}{1 - \frac{4}{5}z^{-1}} \right] = \frac{\frac{3}{8}(\frac{4}{5})^D}{1 - \frac{1}{2}z^{-1}} \quad (6.6.51)$$

$$\text{and } h_c^{[D]}(n) = \frac{3}{8}(\frac{4}{5})^D (\frac{1}{2})^n u(n) \quad (6.6.52)$$

respectively. This shows that as $D \rightarrow \infty$, the impulse response $h_c^{[D]}(n) \rightarrow 0$, which is consistent with our intuition that the prediction is less and less reliable. The MMSE is

$$P_c^{[D]} = 1 - \frac{3}{8}(\frac{4}{5})^{2D} \sum_{k=0}^{\infty} (\frac{2}{5})^k = 1 - \frac{5}{8}(\frac{4}{5})^{2D} \quad (6.6.53)$$

and $P_c^{[D]} \rightarrow r_y(0) = 1$ as $D \rightarrow \infty$, which agrees with our earlier observation. For $D = 2$, the MMSE is $P_c^{[2]} = 93/125 = 0.7440 > P_c$, as expected.

Causal smoothing filter. To estimate the value $y(n+D)$ ($D < 0$) of the desired response using the data $x(n)$, $-\infty < k \leq n$, we need a smoothing Wiener filter. The derivation, which is straightforward but somewhat involved, is left for Problem 6.32. The system function of the optimum smoothing filter is

$$H_c^{[D]}(z) = \frac{3}{8} \left(\frac{z^D}{1 - \frac{1}{2}z^{-1}} + \frac{2^D \sum_{l=0}^{-D-1} 2^l z^{-l}}{1 - \frac{1}{2}z^{-1}} - \frac{4}{5} \frac{2^D \sum_{l=0}^{-D-1} 2^l z^{-l-1}}{1 - \frac{1}{2}z^{-1}} \right) \quad (6.6.54)$$

where $D < 0$. To find the impulse response for $D = -2$, we invert (6.6.54). This gives

$$h_c^{[-2]}(k) = \frac{3}{32} \delta(k) + \frac{51}{320} \delta(k-1) + \frac{39}{128} (\frac{1}{2})^{k-2} u(k-2) \quad (6.6.55)$$

and if we express $r_{yx}(k-2)$ in a similar form, we can compute the MMSE

$$P_c^{[-2]} = 1 - \frac{3}{50} - \frac{51}{400} - (\frac{39}{128}) \frac{5}{3} = \frac{39}{128} = 0.3047 \quad (6.6.56)$$

which is less than $P_c = 0.375$. This should be expected since the smoothing Wiener filter uses more information than the Wiener filter (i.e., when $D = 0$). In fact it can be shown that

$$\lim_{D \rightarrow -\infty} P_c^{[D]} = P_{nc} \quad \text{and} \quad \lim_{D \rightarrow -\infty} h_c^{[D]}(n) = h_{nc}(n) \quad (6.6.57)$$

which is illustrated in Figure 6.20 (Problem 6.22). Figure 6.21 shows the impulse responses of the various optimum IIR filters designed in this example. Interestingly, all are obtained by shifting and truncating the impulse response of the optimum noncausal IIR filter.

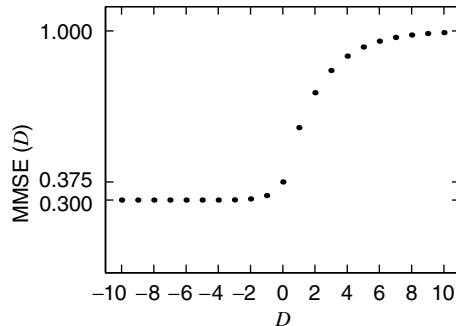


FIGURE 6.20
MMSE as a function of the time shift D .

FIR filter. The M th-order FIR filter is obtained by solving the linear system

$$\mathbf{Rh} = \mathbf{d}$$

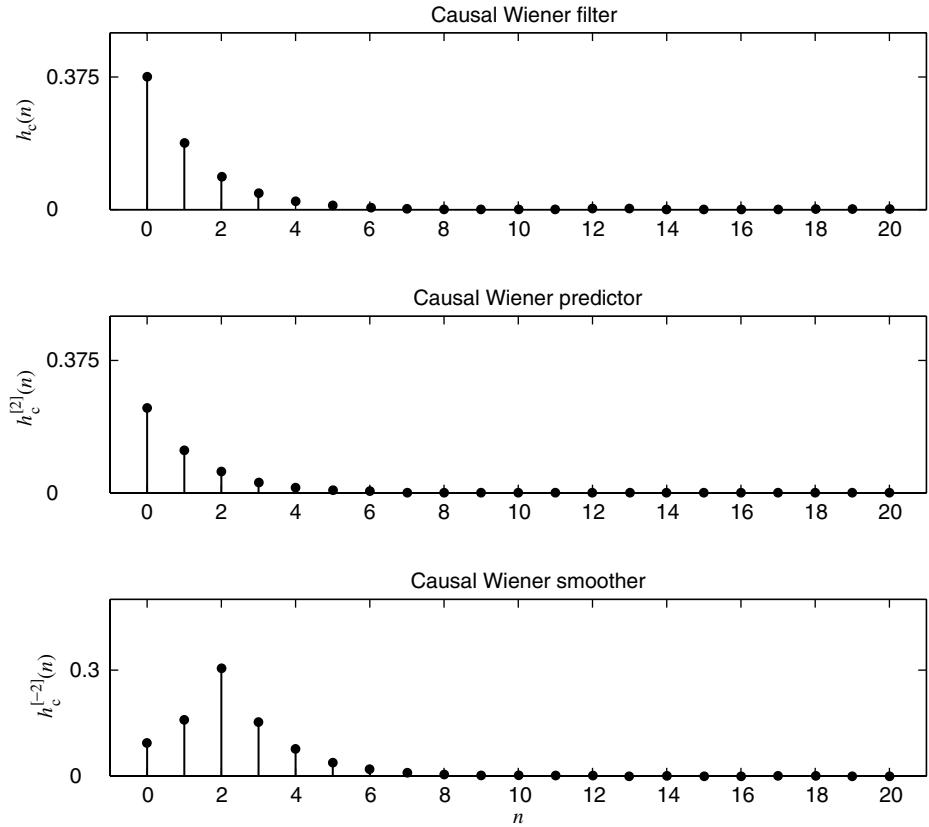
where $\mathbf{R} = \text{Toeplitz}(1 + \sigma_v^2, \alpha, \dots, \alpha^{M-1})$

and $\mathbf{d} = [1 \ \alpha \ \dots \ \alpha^{M-1}]^T$

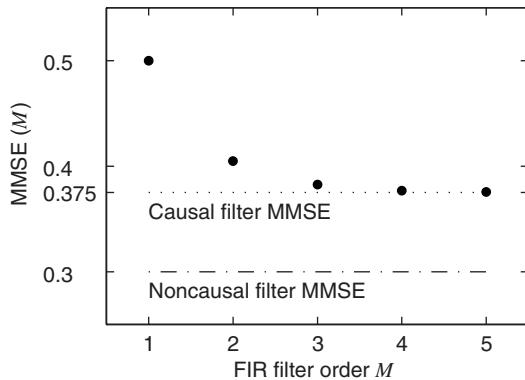
The MMSE is

$$P_o = r_y(0) - \sum_{k=0}^{M-1} h_o(k) r_{yx}(k)$$

and is shown in Figure 6.22 as a function of the order M together with P_c and P_{nc} . We notice that an optimum FIR filter of order $M = 4$ provides satisfactory performance. This can be explained by noting that the impulse response of the causal optimum IIR filter is negligible for $n > 4$.

**FIGURE 6.21**

Impulse response of optimum filters for pure filtering, prediction, and smoothing.

**FIGURE 6.22**

MMSE as a function of the optimum FIR filter order M .

6.6.4 Linear Prediction Using the Infinite Past—Whitening

The one-step forward IIR linear predictor is a causal IIR optimum filter with desired response $y(n) \triangleq x(n+1)$. The prediction error is

$$e^f(n+1) = x(n+1) - \sum_{k=0}^{\infty} h_{lp}(k)x(n-k) \quad (6.6.58)$$

where

$$H_{lp}(z) = \sum_{k=0}^{\infty} h_{lp}(k)z^{-k} \quad (6.6.59)$$

is the system function of the optimum predictor. Since $y(n) = x(n + 1)$, we have $r_{yx}(l) = r_x(l + 1)$ and $R_{yx}(z) = zR_x(z)$. Hence, the optimum predictor is

$$H_{lp}(z) = \frac{1}{\sigma_x^2 H_x(z)} \left[\frac{z\sigma_x^2 H_x(z)H_x(z^{-1})}{H_x(z^{-1})} \right]_+ = \frac{[zH_x(z)]_+}{H_x(z)} = \frac{zH_x(z) - z}{H_x(z)}$$

and the prediction error filter (PEF) is

$$H_{PEF}(z) = \frac{E^f(z)}{X(z)} = 1 - z^{-1}H_{lp}(z) = \frac{1}{H_x(z)} \quad (6.6.60)$$

that is, *the one-step IIR linear predictor of a regular process is identical to the whitening filter of the process*. Therefore, the prediction error process is white, and the prediction error filter is minimum-phase. We will see that the efficient solution of optimum filtering problems includes as a prerequisite the solution of a linear prediction problem. Furthermore, algorithms for linear prediction provide a convenient way to perform spectral factorization in practice.

The MMSE is

$$\begin{aligned} P_o^f &= \frac{1}{2\pi j} \oint_C \left\{ R_x(z) - z \left[1 - \frac{1}{H_x(z)} \right] z^{-1} R_x^* \left(\frac{1}{z^*} \right) \right\} z^{-1} dz \\ &= \frac{1}{2\pi j} \oint_C R_x(z) \frac{1}{H_x(z)} z^{-1} dz \\ &= \sigma_x^2 \frac{1}{2\pi j} \oint_C H_x^* \left(\frac{1}{z^*} \right) z^{-1} dz = \sigma_x^2 \end{aligned} \quad (6.6.61)$$

because

$$\frac{1}{2\pi j} \oint_C H_x^* \left(\frac{1}{z^*} \right) z^{-1} dz = h_x(0) = 1$$

From Section 2.4.4 and (6.6.61) we have

$$P_o^f = \sigma_x^2 = \exp \left[\frac{1}{2\pi} \int_{-\pi}^{\pi} \ln R_x(e^{j\omega}) d\omega \right] \quad (6.6.62)$$

which is known as the *Kolmogorov-Szegö formula*.

We can easily see that the D -step predictor ($D > 0$) is given by

$$H_D(z) = \frac{[z^D H_x(z)]_+}{H_x(z)} = \frac{1}{H_x(z)} \sum_{k=D}^{\infty} h_x(k) z^{-k+D} \quad (6.6.63)$$

but is not guaranteed to be minimum-phase for $D \neq 1$.

EXAMPLE 6.6.2. Consider a minimum-phase AR(2) process

$$x(n) = a_1 x(n-1) + a_2 x(n-2) + w(n)$$

where $w(n) \sim \text{WN}(0, \sigma_w^2)$. The complex PSD of the process is

$$R_x(z) = \frac{\sigma_x^2}{A(z)A(z^{-1})} \triangleq \sigma_x^2 H_x(z)H_x(z^{-1})$$

where $A(z) \triangleq 1 - a_1 z^{-1} - a_2 z^{-2}$ and $\sigma_x^2 = \sigma_w^2$. The one-step forward predictor is given by

$$H_{lp}(z) = z - \frac{z}{H_x(z)} = z - zA(z) = a_1 + a_2 z^{-1}$$

or

$$\hat{x}(n+1) = a_1 x(n) + a_2 x(n-1)$$

as should be expected because the present value of the process depends only on the past two values. Since the excitation $w(n)$ is white and cannot be predicted from the present or previous values of the signal $x(n)$, it is equal to the prediction error $e^f(n)$. Therefore, $\sigma_{e^f}^2 = \sigma_w^2$, as expected from (6.6.62). This shows that the MMSE of the one-step linear predictor depends on the SFM of the process $x(n)$. It is maximum for a white noise process, which is clearly unpredictable.

Predictable processes. A random process $x(n)$ is said to be (exactly) *predictable* if $P_e = E\{|e^f(n)|^2\} = 0$. We next show that a process $x(n)$ is predictable if and only if its PSD consists of impulses, that is,

$$R_x(e^{j\omega}) = \sum_k A_k \delta(\omega - \omega_k) \quad (6.6.64)$$

or in other words, $x(n)$ is a harmonic process. For this reason harmonic processes are also known as *deterministic* processes. From (6.6.60) we have

$$P_e = E\{|e^f(n)|^2\} = \int_{-\pi}^{\pi} |H_{PEF}(e^{j\omega})|^2 R_x(e^{j\omega}) d\omega \quad (6.6.65)$$

where $H_{PEF}(e^{j\omega})$ is the frequency response of the prediction error filter. Since $R_x(e^{j\omega}) \geq 0$, the integral in (6.6.65) is zero if and only if $|H_{PEF}(e^{j\omega})|^2 R_x(e^{j\omega}) = 0$. This is possible only if $R_x(e^{j\omega})$ is a linear combination of impulses, as in (6.6.64), and $e^{j\omega_k}$ are the zeros of $H_{PEF}(z)$ on the unit circle (Papoulis 1985).

From the Wold decomposition theorem (see Section 4.1.3) we know that every random process can be decomposed into two components that are mutually orthogonal: (1) a regular component with continuous PSD that can be modeled as the response of a minimum-phase system to white noise and (2) a predictable process that can be exactly predicted from a linear combination of past values. This component has a line PSD and is essentially a harmonic process. A complete discussion of this subject can be found in Papoulis (1985, 1991) and Therrien (1992).

6.7 INVERSE FILTERING AND DECONVOLUTION

In many practical applications, a signal of interest passes through a distorting system whose output may be corrupted by additive noise. When the distorting system is linear and time-invariant, the observed signal is the convolution of the desired input with the impulse response of the system. Since in most cases we deal with linear and time-invariant systems, the terms *filtering* and *convolution* are often used interchangeably.

Deconvolution is the process of retrieving the *unknown* input of a *known* system by using its observed output. If the system is also unknown, which is more common in practical applications, we have a problem of *blind deconvolution*. The term *blind deconvolution* was introduced in Stockham et al. (1975) for a method used to restore old records. Other applications include estimation of the vocal tract in speech processing, equalization of communication channels, deconvolution of seismic data for the elimination of multiple reflections, and image restoration.

The basic problem is illustrated in Figure 6.23. The output of the unknown LTI system $G(z)$, which is assumed BIBO stable, is given by

$$x(n) = \sum_{k=-\infty}^{\infty} g(k)w(n-k) \quad (6.7.1)$$

where $w(n) \sim \text{IID}(0, \sigma_w^2)$ is a white noise sequence. Suppose that we observe the output $x(n)$ and that we wish to recover the input signal $w(n)$, and possibly the system $G(z)$, using the output signal and some statistical information about the input.

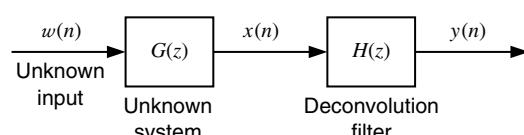


FIGURE 6.23
Basic blind deconvolution model.

If we know the system $G(z)$, the inverse system $H(z)$ is obtained by noticing that perfect retrieval of the input is possible if

$$h(n) * g(n) * w(n) = b_0 w(n - n_0) \quad (6.7.2)$$

where b_0 and n_0 are constants. From (6.7.2), we have $h(n) * g(n) = b_0 \delta(n - n_0)$, or equivalently

$$H(z) = b_0 \frac{z^{-n_0}}{G(z)} \quad (6.7.3)$$

which provides the system function of the inverse system. The input can be recovered by convolving the output with the inverse system $H(z)$. Therefore, the terms *inverse filtering* and *deconvolution* are equivalent for LTI systems.

There are three approaches for blind deconvolution:

- Identify the system $G(z)$, design its inverse system $H(z)$, and then compute the input $w(n)$.
- Identify directly the inverse $H(z) = 1/G(z)$ of the system, and then determine the input $w(n)$.
- Estimate directly the input $w(n)$ from the output $x(n)$.

Any of the above approaches requires either directly or indirectly the estimation of both the magnitude response $|G(e^{j\omega})|$ and the phase response $\angle G(e^{j\omega})$ of the unknown system. In practice, the problem becomes more complicated because the output $x(n)$ is usually corrupted by additive noise. If this noise is uncorrelated with the input signal and the required second-order moments are available, we show how to design an optimum inverse filter that provides an optimum estimate of the input in the presence of noise. In Section 6.8 we apply these results to the design of optimum equalizers for data transmission systems. The main blind identification and deconvolution problem, in which only statistical information about the output is known, is discussed in Chapter 12.

We now discuss the design of optimum inverse filters for linearly distorted signals observed in the presence of additive output noise. The typical configuration is shown in Figure 6.24. Ideally, we would like the optimum filter to restore the distorted signal $x(n)$ to its original value $y(n)$. However, the ability of the optimum filter to attain ideal performance is limited by three factors. First, there is additive noise $v(n)$ at the output of the system. Second, if the physical system $G(z)$ is causal, its output $s(n)$ is delayed with respect to the input, and we may need some delay z^{-D} to improve the performance of the system. When $G(z)$ is a non-minimum-phase system, the inverse system is either noncausal or unstable and should be approximated by a causal and stable filter. Third, the inverse system may be IIR and should be approximated by an FIR filter.

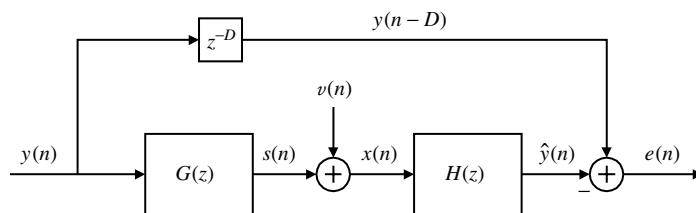


FIGURE 6.24

Typical configuration for optimum inverse system modeling.

The optimum inverse filter is the noncausal Wiener filter

$$H_{nc}(z) = \frac{z^{-D} R_{yx}(z)}{R_x(z)} \quad (6.7.4)$$

where the term z^{-D} appears because the desired response is $y_D(n) \triangleq y(n - D)$. Since $y(n)$

and $v(n)$ are uncorrelated, we have

$$R_{yx}(z) = R_{ys}(z) \quad (6.7.5)$$

and

$$R_x(z) = G(z)G^*\left(\frac{1}{z^*}\right)R_y(z) + R_v(z) \quad (6.7.6)$$

The cross-correlation between $y(n)$ and $s(n)$

$$R_{ys}(z) = G^*\left(\frac{1}{z^*}\right)R_y(z) \quad (6.7.7)$$

is obtained by using Equation (6.6.18). Therefore, the optimum inverse filter is

$$H_{nc}(z) = \frac{z^{-D}G^*(1/z^*)R_y(z)}{G(z)G^*(1/z^*)R_y(z) + R_v(z)} \quad (6.7.8)$$

which, in the absence of noise, becomes

$$H_{nc}(z) = \frac{z^{-D}}{G(z)} \quad (6.7.9)$$

as expected. The behavior of the optimum inverse system is illustrated in the following example.

EXAMPLE 6.7.1. Let the system $G(z)$ be an all-zero non-minimum-phase system given by

$$G(z) = \frac{1}{5}(-3z + 7 - 2z^{-1}) = -\frac{3}{5}(1 - \frac{1}{3}z^{-1})(z - 2)$$

Then the inverse system is given by

$$H(z) = G^{-1}(z) = \frac{5}{-3z + 7 - 2z^{-1}} = \frac{1}{1 - \frac{1}{3}z^{-1}} - \frac{1}{1 - 2z^{-1}}$$

which is stable if the ROC is $-\frac{1}{3} < |z| < 2$. Therefore, the impulse response of the inverse system is

$$h(n) = \begin{cases} (\frac{1}{3})^n & n \geq 0 \\ 2^n & n < 0 \end{cases}$$

which is noncausal and stable.

Following the discussion given in this section, we want to design an optimum inverse system given that $G(z)$ is driven by a white noise sequence $y(n)$ and that the additive noise $v(n)$ is white, that is, $R_y(z) = \sigma_y^2$ and $R_v(z) = \sigma_v^2$. From (6.7.8), the optimum noncausal inverse filter is given by

$$H_{nc}(z) = \frac{z^{-D}}{G(z) + [1/G(z^{-1})](\sigma_v^2/\sigma_y^2)}$$

which can be computed by assuming suitable values for variances σ_y^2 and σ_v^2 . Note that if $\sigma_v^2 \ll \sigma_y^2$, that is, for very large SNR, we obtain (6.7.9).

A more interesting case occurs when the optimum inverse filter is FIR, which can be easily implemented. To design this FIR filter, we will need the autocorrelation $r_x(l)$ and the cross-correlation $r_{y_Dx}(l)$, where $y_D(n) = y(n - D)$ is the delayed system input sequence. Since

$$R_x(z) = \sigma_y^2 G(z)G(z^{-1}) + \sigma_v^2$$

and

$$R_{y_Dx}(l) = \sigma_y^2 z^{-D} G(z^{-1})$$

we have (see Section 3.4.1)

$$r_x(l) = g(l) * g(-l) * r_y(l) + r_v(l) = \sigma_y^2 [g(l) * g(-l)] + \sigma_v^2 \delta(l)$$

and

$$r_{y_Dx}(l) = g(-l) * r_y(l - D) = \sigma_y^2 g(-l + D)$$

respectively. Now we can determine the optimum FIR filter \mathbf{h}_D of length M by constructing an $M \times M$ Toeplitz matrix \mathbf{R} from $r_x(l)$ and an $M \times 1$ vector \mathbf{d} from $r_{y_D}(l)$ and then solving

$$\mathbf{Rh}_D = \mathbf{d}$$

for various values of D . We can then plot the MMSE as a function of D to determine the best value of D (and the corresponding FIR filter) which will give the smallest MMSE. For example, if $\sigma_y^2 = 1$, $\sigma_v^2 = 0.1$, and $M = 10$, the correlation functions are

$$r_x(l) = \begin{bmatrix} \frac{6}{25}, -\frac{7}{5}, \frac{129}{50}, -\frac{7}{5}, \frac{6}{25} \\ \uparrow l=0 \end{bmatrix} \quad \text{and} \quad r_{yDx}(l) = \begin{bmatrix} -\frac{2}{5}, \frac{7}{5}, -\frac{3}{5} \\ \uparrow l=D \end{bmatrix}$$

The resulting MMSE as a function of D is shown in Figure 6.25, which indicates that the best value of D is approximately $M/2$. Finally, plots of impulse responses of the inverse system are shown in Figure 6.26. The first plot shows the noncausal $h(n)$, the second plot shows the causal

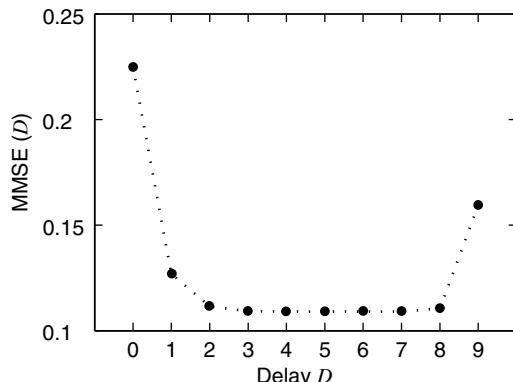


FIGURE 6.25
The inverse filtering MMSE as a function of delay D .

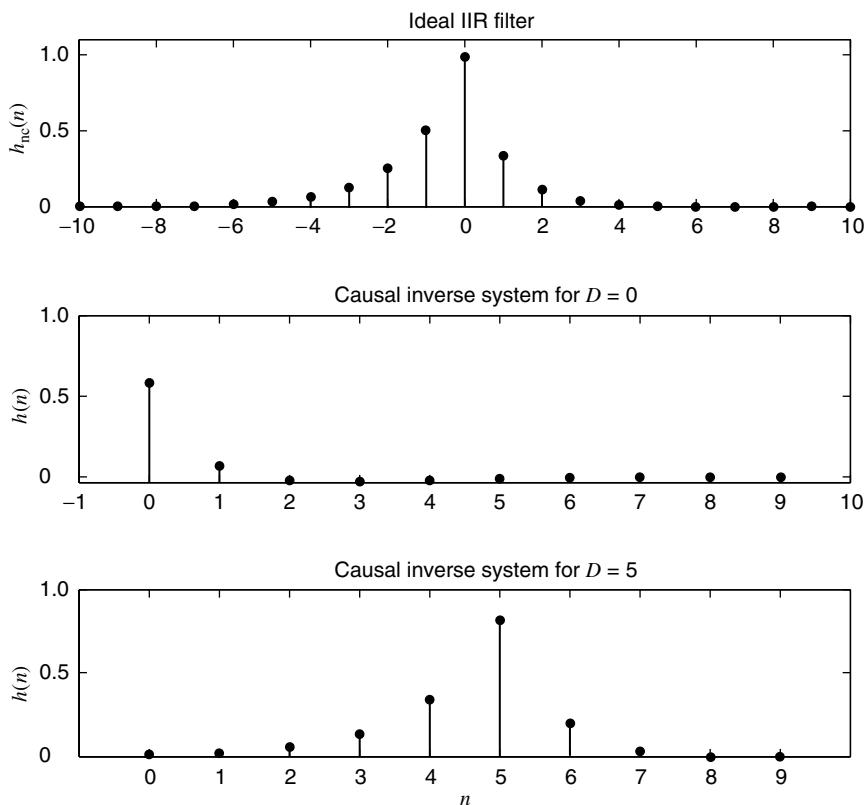


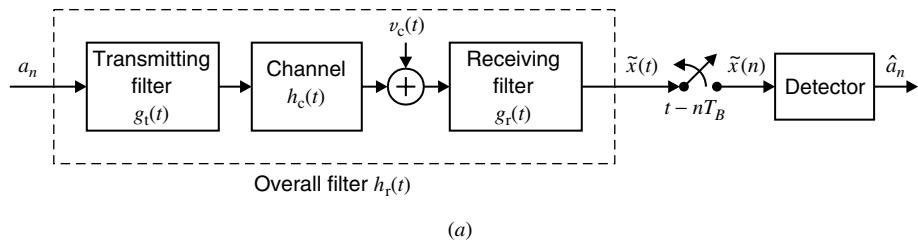
FIGURE 6.26
Impulse responses of optimum inverse filters.

FIR system $h_0(n)$ for $D = 0$, and the third plot shows the causal FIR system $h_D(n)$ for $D = 5$. It is clear that the optimum delayed FIR inverse filter for $D \simeq M/2$ closely matches the impulse response of the inverse filter $h(n)$.

6.8 CHANNEL EQUALIZATION IN DATA TRANSMISSION SYSTEMS

The performance of data transmission systems through channels that can be approximated by linear systems is limited by factors such as finite bandwidth, intersymbol interference, and thermal noise (see Section 1.4). Typical examples include telephone lines, microwave line-of-sight radio links, satellite channels, and underwater acoustic channels. When the channel frequency response deviates from the ideal of flat magnitude and linear phase, both (left and right) tails of a transmitted pulse will interfere with neighboring pulses. Hence, the value of a sample taken at the center of a pulse will contain components from the tails of the other pulses. The distortion caused by the overlapping tails is known as *intersymbol interference (ISI)*, and it can lead to erroneous decisions that increase the probability of error. For band-limited channels with low background noise (e.g., voice band telephone channel), ISI is the main performance limitation for high-speed data transmission. In radio and undersea channels, ISI is the result of multipath propagation (Siller 1984).

Intersymbol interference occurs in all pulse modulation systems, including frequency-shift keying (FSK), phase-shift keying (PSK), and quadrature amplitude modulation (QAM). However, to simplify the presentation, we consider a baseband pulse amplitude modulation (PAM) system. This does not result in any loss of generality because we can obtain an equivalent baseband model for any linear modulation scheme (Proakis 1996). We consider the K -ary ($K = 2^L$) PAM communication system shown in Figure 6.27(a). The binary



(a)

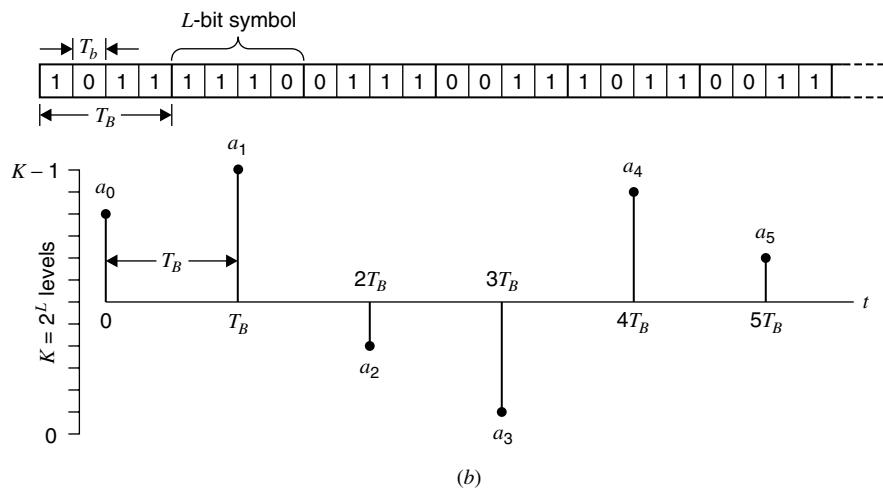


FIGURE 6.27

(a) Baseband pulse amplitude modulation data transmission system model and (b) input symbol sequence a_n .

input sequence is subdivided into L -bit blocks, or *symbols*, and each symbol is mapped to one of the K amplitude levels, as shown in Figure 6.27(b). The interval T_B is called the symbol or *baud interval* while the interval T_b is called the *bit interval*. The quantity $\mathcal{R}_B = 1/T_B$ is known as the *baud rate*, and the quantity $\mathcal{R}_b = L\mathcal{R}_B$ is the *bit rate*.

The resulting symbol sequence $\{a_n\}$ modulates the transmitted pulse $g_t(t)$. For analysis purposes, the symbol sequence $\{a_n\}$ can be represented by an equivalent continuous-time signal using an impulse train, that is,

$$\{a_n\}_{-\infty}^{\infty} \Leftrightarrow \sum_{n=-\infty}^{\infty} a_n \delta(t - nT_B) \quad (6.8.1)$$

The modulated pulses are transmitted over the channel represented by the impulse response $h_c(t)$ and the additive noise $v_c(t)$. The received signal is filtered by the receiving filter $g_r(t)$ to obtain $\tilde{x}(t)$. Using (6.8.1), the signal $\tilde{x}(t)$ at the output of the receiving filter is given by

$$\begin{aligned} \tilde{x}(t) &= \sum_{k=-\infty}^{\infty} a_k \{ \delta(t - kT_B) * g_t(t) * h_c(t) * g_r(t) \} + v_c(t) * g_r(t) \\ &\triangleq \sum_{k=-\infty}^{\infty} a_k \tilde{h}_r(t - kT_B) \tilde{v}(t) \end{aligned} \quad (6.8.2)$$

where

$$\tilde{h}_r(t) \triangleq g_t(t) * h_c(t) * g_r(t) \quad (6.8.3)$$

is the impulse response of the combined system of transmitting filter, channel, and receiving filter, and

$$\tilde{v}(t) \triangleq g_r(t) * v_c(t) \quad (6.8.4)$$

is the additive noise at the output of the receiving filter.

6.8.1 Nyquist's Criterion for Zero ISI

If we sample the received signal $x(t)$ at the time instant $t_0 + nT_B$, we obtain

$$\begin{aligned} \tilde{x}(t_0 + nT_B) &= \sum_{k=-\infty}^{\infty} a_k \tilde{h}_r(t_0 + nT_B - kT_B) + \tilde{v}(t_0 + nT_B) \\ &= a_n \tilde{h}_r(t_0) + \sum_{\substack{k=-\infty \\ k \neq n}}^{\infty} a_k \tilde{h}_r(t_0 + nT_B - kT_B) + \tilde{v}(t_0 + nT_B) \end{aligned} \quad (6.8.5)$$

where t_0 accounts for the channel delay and the sampler phase. The first term in (6.8.5) is the desired signal term while the third term is the noise term. The middle term in (6.8.5) represents the ISI, and it will be zero if and only if

$$\tilde{h}_r(t_0 + nT_B - kT_B) = 0 \quad n \neq k \quad (6.8.6)$$

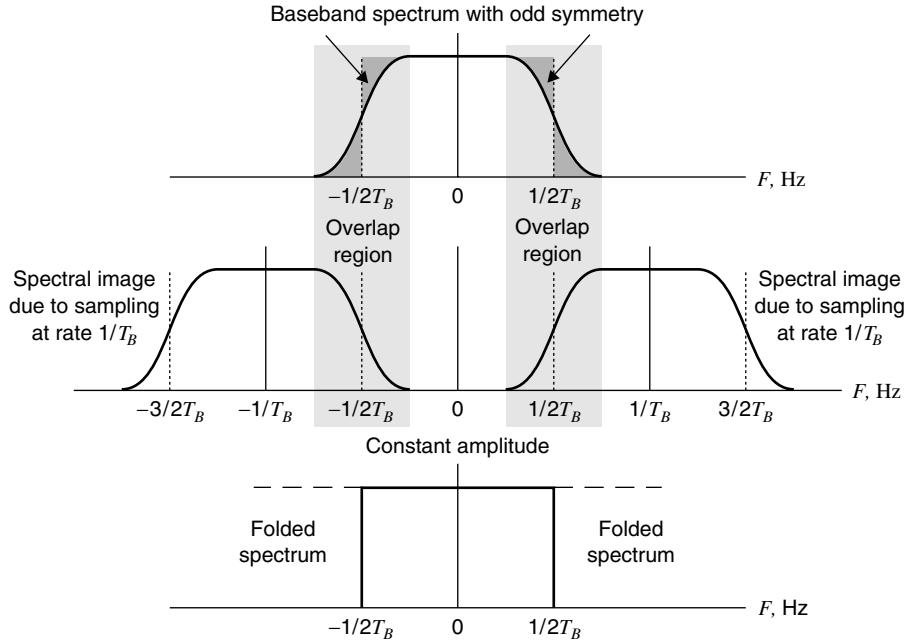
As was first shown by Nyquist (Gitlin, Hayes, and Weinstein 1992), a time-domain pulse $\tilde{h}_r(t)$ will have zero crossings once every T_B s, that is,

$$\tilde{h}_r(nT_B) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases} \quad (6.8.7)$$

if its Fourier transform satisfies the condition

$$\sum_{l=-\infty}^{\infty} \tilde{H}_r \left(F + \frac{l}{T_B} \right) = T_B \quad (6.8.8)$$

This condition is known as the *Nyquist criterion for zero ISI* and its basic meaning is illustrated in Figure 6.28.

**FIGURE 6.28**

Frequency-domain Nyquist criterion for zero ISI.

A pulse shape that satisfies (6.8.8) and that is widely used in practice is of the *raised cosine family*

$$\tilde{h}_{\text{rc}}(t) = \frac{\sin(\pi t/T_B)}{\pi t/T_B} \frac{\cos(\pi\alpha t/T_B)}{1 - 4\alpha^2 t^2/T_B^2} \quad (6.8.9)$$

where $0 \leq \alpha \leq 1$ is known as the *rolloff factor*. This pulse and its Fourier transform for $\alpha = 0, 0.5$, and 1 are shown in Figure 6.29. The choice of $\alpha = 0$ reduces $\tilde{h}_{\text{rc}}(t)$ to the unrealizable sinc pulse and $\mathcal{R}_B = 1/T_B$, whereas for $\alpha = 1$ the symbol rate is $\mathcal{R}_B = 1/(2T_B)$. In practice, we can see the effect of ISI and the noise if we display the received signal on the vertical axis of an oscilloscope and set the horizontal sweep rate at $1/T_B$. The resulting display is known as *eye pattern* because it resembles the human eye. The closing of the eye increases with the increase in ISI.

6.8.2 Equivalent Discrete-Time Channel Model

Referring to Figure 6.27(a), we note that the input to the data transmission system is a discrete-time sequence $\{a_n\}$ at the symbol rate $1/T_B$ symbols per second, and the input to the detector is also a discrete-time sequence $\tilde{x}(nT_B)$ at the symbol rate. Thus the overall system between the input symbols and the equalizer can be modeled as a discrete-time channel model for further analysis. From (6.8.2), after sampling at the symbol rate, we obtain

$$\tilde{x}(nT_B) = \sum_{k=-\infty}^{\infty} a_k \tilde{h}_r(nT_B - kT_B) + \tilde{v}(nT_B) \quad (6.8.10)$$

where $\tilde{h}_r(t)$ is given in (6.8.3) and $\tilde{v}(t)$ is given in (6.8.4). The first term in (6.8.10) can be interpreted as a discrete-time IIR filter with impulse response[†] $\tilde{h}_r(n) \triangleq h_r(nT_B)$ with input

[†]Here we have abused the notation to avoid a new symbol.

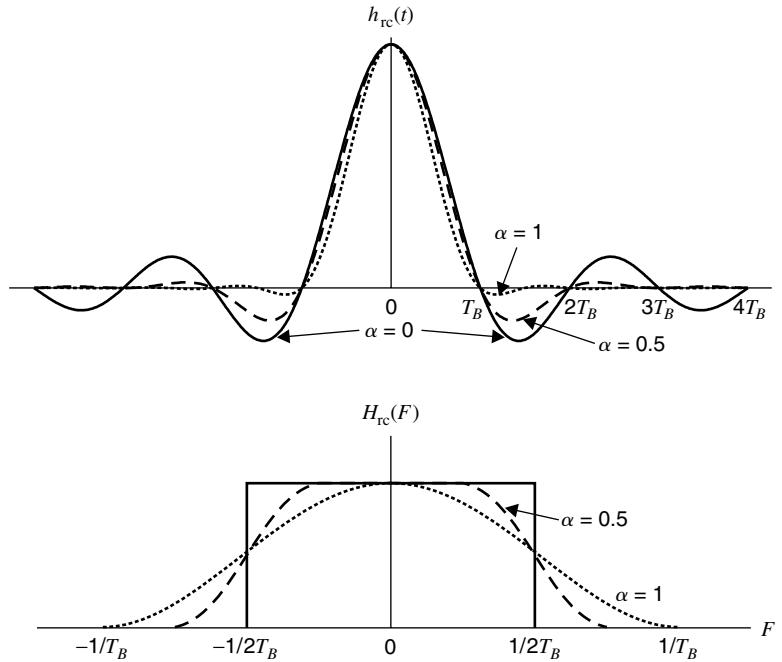


FIGURE 6.29
 Pulses with a raised cosine spectrum.

a_k . In a practical data transmission system, it is not unreasonable to assume that $\tilde{h}_r(n) = 0$ for $|n| \geq L$, where L is some arbitrary positive integer. Then we obtain

$$\begin{aligned} \tilde{x}(n) &= \sum_{k=-L}^L a_k \tilde{h}_r(n-k) + \tilde{v}(n) \\ \tilde{x}(n) &\triangleq \tilde{x}(nT_B) \quad \tilde{v}(n) \triangleq \tilde{v}(nT_B) \end{aligned} \quad (6.8.11)$$

which is an FIR filter of length $2L + 1$, shown in Figure 6.30.

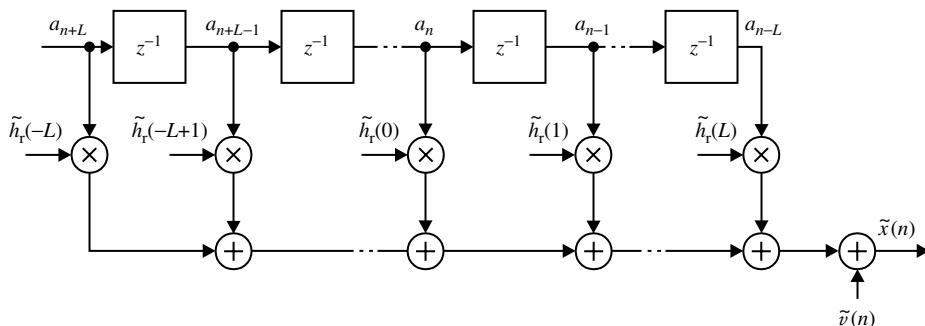


FIGURE 6.30
 Equivalent discrete-time model of data transmission system with ISI.

There is one difficulty with this model. If we assume that the additive channel noise $v_c(t)$ is zero-mean white, then the equivalent noise sequence $\tilde{v}(n)$ is not white. This can be seen from the definition of $\tilde{v}(t)$ in (6.8.4). Thus the autocorrelation of $\tilde{v}(n)$ is given by

$$r_{\tilde{v}}(l) = \sigma_v^2 r_{g_r}(l) \quad (6.8.12)$$

where σ_v^2 is the variance of the samples of $v_c(t)$ and $r_{g_r}(l)$ is the sampled autocorrelation of

$g_r(t)$. This nonwhiteness of $\tilde{v}(t)$ poses a problem in the subsequent design and performance evaluation of equalizers. Therefore, in practice, it is necessary to whiten this noise by designing a whitening filter and placing it after the sampler in Figure 6.27(a). The whitening filter is designed by using spectral factorization of $\mathcal{Z}[r_{gr}(l)]$. Let

$$R_{gr}(z) = \mathcal{Z}[r_{gr}(l)] = R_{gr}^+(z)R_{gr}^-(z) \quad (6.8.13)$$

where $R_{gr}^+(z)$ is the minimum-phase factor and $R_{gr}^-(z)$ is the maximum-phase factor. Choosing

$$W(z) \triangleq \frac{1}{R_{gr}^+(z)} \quad (6.8.14)$$

as a causal, stable, and recursive filter and applying the sampled sequence $\tilde{x}(n)$ to this filter, we obtain

$$x(n) \triangleq w(n) * \tilde{x}(n) = \sum_{k=0}^{\infty} a_k h_r(n-k) + v(n) \quad (6.8.15)$$

where $h_r(n) \triangleq \tilde{h}_r(n) * w(n)$ (6.8.16)

and $v(n) \triangleq w(n) * \tilde{v}(n)$ (6.8.17)

The spectral density of $v(n)$, from (6.8.12), (6.8.13), and (6.8.14), is given by

$$R_v(z) = R_w(z)R_{\tilde{v}}(z) = \frac{1}{R_{gr}^+(z)R_{gr}^-(z)}\sigma_v^2 R_{gr}^+(z)R_{gr}^-(z) = \sigma_v^2 \quad (6.8.18)$$

which means that $v(n)$ is a white sequence. Once again, assuming that $h_r(n) = 0, n > L$, where L is an arbitrary positive integer, we obtain an equivalent discrete-time channel model with white noise

$$x(n) = \sum_{k=0}^L a_k h_r(n-k) + v(n) \quad (6.8.19)$$

This equivalent model is shown in Figure 6.31. An example to illustrate the use of this model in the design and analysis of an equalizer is given in the next section.

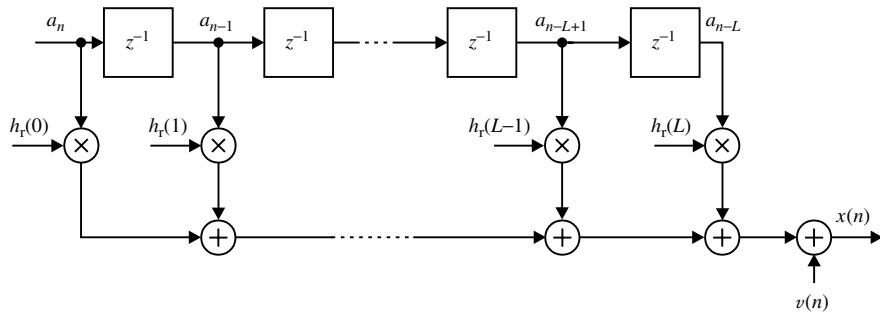


FIGURE 6.31

Equivalent discrete-time model of data transmission system with ISI and WGN.

6.8.3 Linear Equalizers

If we know the characteristics of the channel, that is, the magnitude response $|H_c(F)|$ and the phase response $\angle H_c(F)$, we can design optimum transmitting and receiving filters that will maximize the SNR and will result in zero ISI at the sampling instant. However, in practice we have to deal with channels whose characteristics are either unknown (dial-up telephone channels) or time-varying (ionospheric radio channels). In this case, we usually

use a receiver that consists of a fixed filter $g_r(t)$ and an adjustable *linear equalizer*, as shown in Figure 6.32. The response of the fixed filter either is matched to the transmitted pulse or is designed as a compromise equalizer for an “average” channel typical of the given application. In principle, to eliminate the ISI, we should design the equalizer so that the overall pulse shape satisfies Nyquist’s criterion (6.8.6) or (6.8.8).

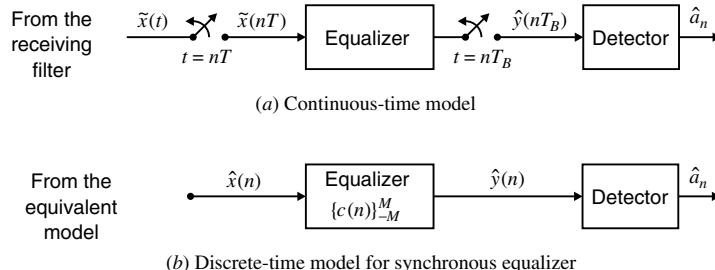


FIGURE 6.32

Equalizer-based receiver model.

The most widely used equalizers are implemented using digital FIR filters. To this end, as shown in Figure 6.32(a), we sample the received signal $\tilde{x}(t)$ periodically at times $t = t_0 + nT$, where t_0 is the sampling phase and T is the sampling period. The sampling period should be less or equal to the symbol interval T_B because the output of the equalizer should be sampled once every symbol interval (the case $T > T_B$ creates aliasing). For digital implementation T should be chosen as a rational fraction of the symbol interval, that is, $T = L_1 T_B / L_2$, with $L_1 \leq L_2$ (typical choices are $T = T_B$, $T = T_B/2$, or $T = 2T_B/3$). If the sampling interval $T = T_B$, we have a *synchronous* or *symbol equalizer (SE)*[†] and if $T < T_B$ a *fractionally spaced equalizer (FSE)*.[‡] The output of the equalizer is quantized to obtain the decision \hat{a}_n .

The goal of the equalizer is to determine the coefficients $\{c_k\}_{-M}^M$ so as to minimize the ISI according to some criterion of performance. The most meaningful criterion for data transmission is the average probability of error. However, this criterion is a nonlinear function of the equalizer coefficients, and its minimization is extremely difficult.

We next discuss two criteria that are used in practical applications. For this discussion we assume a synchronous equalizer, that is, $T = T_B$. The FSE is discussed in Chapter 12. For the synchronous equalizer, the equivalent discrete-time model given in Figure 6.31 is applicable in which the input is $x(n)$, given by

$$x(n) = \sum_{l=0}^L a_l h_r(n-l) + v(n) \quad (6.8.20)$$

The output of the equalizer is given by

$$\hat{y}(n) = \sum_{k=-M}^M c^*(k) x(n-k) \triangleq \mathbf{c}^H \mathbf{x}(n) \quad (6.8.21)$$

where

$$\mathbf{c} = [c(-M) \cdots c(0) \cdots c(M)]^T \quad (6.8.22)$$

$$\mathbf{x}(n) = [x(n+M) \cdots x(n) \cdots x(n-M)]^T \quad (6.8.23)$$

This equalizer model is shown in Figure 6.32(b).

[†]Also known as a baud-spaced equalizer (BSE).

[‡]The most significant difference between SE and FSE is that by properly choosing T we can completely avoid aliasing at the input of the FSE. Thus, the FSE can provide better compensation for timing phase and asymmetries in the channel response without noise enhancement (Qureshi 1985).

6.8.4 Zero-Forcing Equalizers

Zero-forcing (zf) equalization (Lucky, Saltz, and Weldon 1968) requires that the response of the equalizer to the combined pulse $\tilde{h}_r(t)$ satisfy the Nyquist criterion (6.8.7). For the FIR equalizer in (6.8.21), in the absence of noise we have

$$\sum_{k=-M}^M c_{\text{zf}}(k)h_r(n-k) = \begin{cases} 1 & n=0 \\ 0 & n=\pm 1, \pm 2, \dots, \pm M \end{cases} \quad (6.8.24)$$

which is a linear system of equations whose solution provides the required coefficients. The zero-forcing equalizer does not completely eliminate the ISI because it has finite duration. If $M = \infty$, Equation (6.8.24) becomes a convolution equation that can be solved by using the z -transform. The solution is

$$C_{\text{zf}}(z) = \frac{1}{H_r(z)} \quad (6.8.25)$$

where $H_r(z)$ is the z -transform of $h_r(n)$. Thus, the zero-forcing equalizer is an inverse filter that inverts the frequency-folded (aliased) response of the overall channel. When M is finite, then it is generally impossible to eliminate the ISI at the output of the equalizer because there are only $2M + 1$ adjustable parameters to force zero ISI outside of $[-M, M]$. Then the equalizer design problem reverts to minimizing the *peak distortion*

$$D \triangleq \sum_{n \neq 0} \left| \sum_{k=-M}^M c_{\text{zf}}(k)h_r(n-k) \right| \quad (6.8.26)$$

This distortion function can be shown to be a convex function (Lucky 1965), and its minimization, in general, is difficult to obtain except when the input ISI is less than 100 percent (i.e., the eye pattern is open). This minimization and the determination of $\{c_{\text{zf}}\}$ can be obtained by using the steepest descent algorithm, which is discussed in Chapter 10.

Zero-forcing equalizers have two drawbacks: (1) They ignore the presence of noise and therefore amplify the noise appearing near the spectral nulls of $H_r(e^{j\omega})$, and (2) they minimize the peak distortion or *worst-case ISI* only when the eye is open. For these reasons they are not currently used for bad channels or high-speed modems (Qureshi 1985). The above two drawbacks are eliminated if the equalizers are designed using the MSE criterion.

6.8.5 Minimum MSE Equalizers

It has been shown (Saltzberg 1968) that the error rate $\Pr\{\hat{a}_n \neq a_n\}$ decreases monotonically with the MSE defined by

$$\text{MSE} = E\{|e(n)|^2\} \quad (6.8.27)$$

where $e(n) = y(n) - \hat{y}(n) = a_n - \hat{y}(n)$ (6.8.28)

is the difference between the desired response $y(n) \triangleq a_n$ and the actual response $\hat{y}(n)$ given in (6.8.21). Therefore, if we minimize the MSE in (6.8.27), we take into consideration both the ISI and the noise at the output of the equalizer. For $M = \infty$, following the arguments similar to those leading to (6.8.25), the minimum MSE equalizer is specified by

$$C_{\text{MSE}}(z) = \frac{H_r^*(1/z^*)}{H_r(z)H_r^*(1/z^*) + \sigma_v^2} \quad (6.8.29)$$

where σ_v^2 is the variance of the sampled channel noise $v_c(kT_B)$. Clearly, (6.8.29) reduces to the zero-forcing equalizer if $\sigma_v^2 = 0$. Also (6.8.29) is the classical Wiener filter. For finite M , the minimum MSE equalizer is specified by

$$\mathbf{R}\mathbf{c}_o = \mathbf{d} \quad (6.8.30)$$

where $\mathbf{R} = E\{\mathbf{x}(n)\mathbf{x}^H(n)\}$ and $\mathbf{d} = E\{a_n^*\mathbf{x}(n)\}$. The data sequence $y(n) = a_n$ is assumed to be white with zero mean and power $P_a = E\{|a_n|^2\}$, and uncorrelated with the additive channel noise. Under these assumptions, the elements of the correlation matrix \mathbf{R} and the cross-correlation vector \mathbf{d} are given by

$$\begin{aligned} r_{ij} &\triangleq E\{x(n-i)x^*(n-j)\} \\ &= P_a \sum_m h_r(m-i)h_r^*(m-j) + \sigma_v^2 \delta_{ij} \quad -M \leq i, j \leq M \end{aligned} \quad (6.8.32)$$

$$\text{and } d_i \triangleq E\{x(n-i)y^*(n)\} = P_a h_r(-i) \quad -M \leq i, j \leq M \quad (6.8.33)$$

that is, in terms of the overall (equivalent) channel response $h_r(n)$ and the noise power σ_v^2 . We hasten to stress that matrix \mathbf{R} is Toeplitz if $T = T_B$; otherwise, for $T \neq T_B$, matrix \mathbf{R} is Hermitian but not Toeplitz.

Since MSE equalizers, in contrast to zero-forcing equalizers, take into account both the statistical properties of the noise and the ISI, they are more robust to both noise and large amounts of ISI.

EXAMPLE 6.8.1. Consider the model of the data communication system shown in Figure 6.33. The input symbol sequence $\{a(n)\}$ is a Bernoulli sequence $\{\pm 1\}$, with $\Pr\{1\} = \Pr\{-1\} = 0.5$. The channel (including the receiving and whitening filter) is modeled as

$$h(n) = \begin{cases} 0.5 \left[1 + \cos \frac{2\pi(n-2)}{W} \right] & n = 1, 2, 3 \\ 0 & \text{otherwise} \end{cases} \quad (6.8.34)$$

where W controls the amount of amplitude distortion introduced by the channel. The channel impulse response values are (the arrow denotes the sample at $n = 0$)

$$h(n) = \left\{ \uparrow 0, 0.5 \left(1 + \cos \frac{2\pi}{W} \right), 1, 0.5 \left(1 + \cos \frac{2\pi}{W} \right), 0 \right\} \quad (6.8.35)$$

which is a symmetric channel, and its frequency response is

$$H(e^{j\omega}) = e^{-j2\omega} \left[1 + \left(1 + \cos \frac{2\pi}{W} \right) \cos \omega \right]$$

The channel noise $v(n)$ is modeled as white Gaussian noise (WGN) with zero mean and variance σ_v^2 . The equalizer is an 11-tap FIR filter whose optimum tap weights $\{c(n)\}$ are obtained using either optimum filter theory (nonadaptive approach) or adaptive algorithms that will be described in Chapter 10. The input to the equalizer is

$$x(n) = s(n) + v(n) = h(n) * a(n) + v(n) \quad (6.8.36)$$

where $s(n)$ represents the distorted pulse sequence. The output of the equalizer is $\hat{y}(n)$, which is an estimate of $a(n)$. In practical modem implementations, the equalizer is initially designed using a training sequence that is known to the receiver. It is shown in Figure 6.33 as the sequence $y(n)$. It is reasonable to introduce a delay D in the training sequence to account for delays introduced in the channel and in the equalizer; that is, $y(n) = a(n-D)$ during the training phase. The error sequence $e(n)$ is further used to design the equalizer $c(n)$. The aim of this example is to study the effect of the delay D and to determine its optimum value for proper operation.

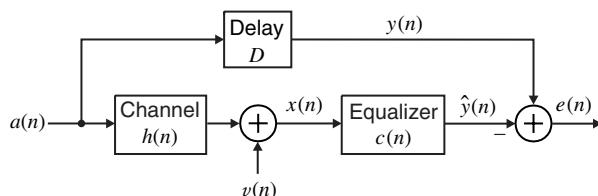


FIGURE 6.33
Data communication model
used in Example 6.8.1.

To obtain an optimum equalizer $c(n)$, we will need the autocorrelation matrix \mathbf{R}_x of the input sequence $x(n)$ and the cross-correlation vector \mathbf{d} between $x(n)$ and $y(n)$. Consider the autocorrelation $r_x(l)$ of $x(n)$. From (6.8.36), assuming real-valued quantities, we obtain

$$r_x(l) = E\{[s(n) + v(n)][s(n-l) + v(n-l)]\} = r_s(l) + \sigma_v^2 \delta(l) \quad (6.8.37)$$

where we have assumed that $s(n)$ and $v(n)$ are uncorrelated. Since $s(n)$ is a convolution between $\{a_n\}$ and $h(n)$, the autocorrelation $r_s(l)$ is given by

$$r_s(l) = r_a(l) * r_h(l) = r_h(l) \quad (6.8.38)$$

where $r_a(l) = \delta(l)$ since $\{a(n)\}$ is a Bernoulli sequence, and $r_h(l)$ is the autocorrelation of the channel response $h(n)$ and is given by

$$r_h(l) = h(l) * h(-l)$$

Using the symmetric channel response values in (6.8.35), we find that the autocorrelation $r_x(l)$ in (6.8.37) is given by

$$\begin{aligned} r_x(0) &= h^2(1) + h^2(2) + h^2(3) + \sigma_v^2 = 1 + 0.5 \left(1 + \cos \frac{2\pi}{W}\right)^2 + \sigma_v^2 \\ r_x(\pm 1) &= h(1)h(2) + h(2)h(3) = 1 + \cos \frac{2\pi}{W} \\ r_x(\pm 2) &= h(1)h(3) = 0.25 \left(1 + \cos \frac{2\pi}{W}\right)^2 \\ r_x(l) &= 0 \quad |l| \geq 3 \end{aligned} \quad (6.8.39)$$

Since the equalizer is an 11-tap FIR filter, the autocorrelation matrix \mathbf{R}_x is an 11×11 matrix. However, owing to few nonzero values of $r_x(l)$ in (6.8.39), it is also a *quintdiagonal* matrix with the main diagonal containing $r_x(0)$ and two upper and lower non-zero diagonals. The cross-correlation between $x(n)$ and $y(n) = a(n - D)$ is given by

$$\begin{aligned} d(l) &= E\{a(n - D)x(n - l)\} = E\{a(n - D)[s(n - l) + v(n - l)]\} \\ &= E\{a(n - D)s(n - l)\} + E\{a(n - D)v(n - l)\} \\ &= E\{a(n - D)[h(n - l) * a(n - l)]\} \\ &= h(D - l) * r_a(D - l) = h(D - l) \end{aligned} \quad (6.8.40)$$

where we have used (6.8.36). The last step follows from the fact that $r_a(l) = \delta(l)$. Using the channel impulse response values in (6.8.35), we obtain

$$\begin{aligned} D = 0 \quad d(l) &= h(-l) = 0 \quad l \geq 0 \\ D = 1 \quad d(l) &= h(1 - l) \Rightarrow d(0) = h(1) \quad d(l) = 0 \quad l > 0 \\ D = 2 \quad d(l) &= h(2 - l) \Rightarrow d(0) = h(2) \quad d(1) = h(1) \quad d(l) = 0 \quad l > 1 \\ &\vdots \quad \vdots \\ D = 7 \quad d(l) &= h(7 - l) \Rightarrow d(4) = h(3) \quad d(5) = h(2) \quad d(6) = h(1) \\ &d(l) = 0 \quad \text{elsewhere} \end{aligned} \quad (6.8.41)$$

Remarks. There are some interesting observations that we can make from (6.8.41) in which the delay D turns the estimation problem into a filtering, prediction, or smoothing problem.

1. When $D = 0$, we have a filtering case. The cross-correlation vector $\mathbf{d} = \mathbf{0}$, hence the equalizer taps are all zeros. This means that if we do not provide any delay in the system, the cross-correlation is zero and equalization is not possible because $\mathbf{c}_o = \mathbf{0}$.
2. When $D = 1$, we have a one-step prediction case.
3. When $D \geq 2$, we have a smoothing filter, which provides better performance. When $D = 7$, we note that the vector \mathbf{d} is symmetric [with respect to the middle sample $d(5)$] and hence we should expect the best performance because the channel is also symmetric. We can also show that $D = 7$ is the optimum delay for this example (see Problem 6.40). However, this should not be a surprise since $h(n)$ is symmetric about $n = 2$, and if we make the equalizer symmetric about $n = 5$, then the channel input $a(n)$ is delayed by $D = 5 + 2 = 7$.

Figure 6.34 shows the channel impulse response $h(n)$ and the equalizer $c(n)$ for $D = 7$, $\sigma_v^2 = 0.001$, and $W = 2.9$ and $W = 3.1$.

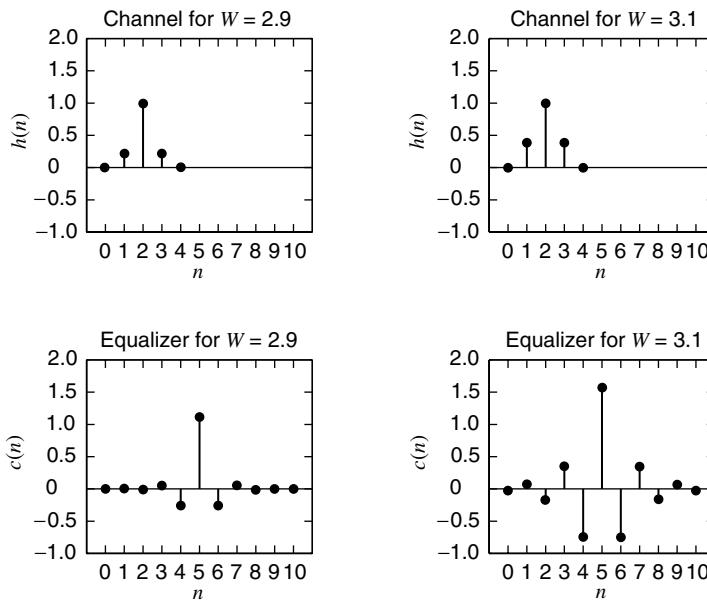


FIGURE 6.34

Channel impulse response $h(n)$ and the equalizer $c(n)$ for $D = 7$, $\sigma_v^2 = 0.001$, and $W = 2.9$ and $W = 3.1$.

6.9 MATCHED FILTERS AND EIGENFILTERS

In this section we discuss the design of optimum filters that maximize the output signal-to-noise power ratio. Such filters are used to detect signals in additive noise in many applications, including digital communications and radar. First we discuss the case of a known deterministic signal in noise, and then we extend the results to the case of a random signal in noise.

Suppose that the observations obtained by sampling the output of a single sensor at M instances, or M sensors at the same instant, are arranged in a vector $\mathbf{x}(n)$. Furthermore, we assume that the available signal $\mathbf{x}(n)$ consists of a desired signal $\mathbf{s}(n)$ plus an additive noise plus interference signal $\mathbf{v}(n)$, that is,

$$\mathbf{x}(n) = \mathbf{s}(n) + \mathbf{v}(n) \quad (6.9.1)$$

where $\mathbf{s}(n)$ can be one of two things. It can be a *deterministic* signal of the form $\mathbf{s}(n) = \alpha \mathbf{s}_0$, where \mathbf{s}_0 is the completely known shape of $\mathbf{s}(n)$ and α is a complex random variable with power $P_\alpha = E\{|\alpha|^2\}$. The argument $\angle \alpha$ provides the unknown initial phase, and the modulus $|\alpha|$, the amplitude of the signal, respectively. It can also be a *random* signal with known correlation matrix $\mathbf{R}_s(n)$. The signals $\mathbf{s}(n)$ and $\mathbf{v}(n)$ are assumed to be uncorrelated with zero means.

The output of a linear processor (combiner or FIR filter) with coefficients $\{c_k^*\}_1^M$ is

$$y(n) = \mathbf{c}^H \mathbf{x}(n) = \mathbf{c}^H \mathbf{s}(n) + \mathbf{c}^H \mathbf{v}(n) \quad (6.9.2)$$

$$\text{and its power } P_y(n) = E\{|y(n)|^2\} = E\{\mathbf{c}^H \mathbf{x}(n) \mathbf{x}^H(n) \mathbf{c}\} = \mathbf{c}^H \mathbf{R}_x(n) \mathbf{c} \quad (6.9.3)$$

is a quadratic function of the filter coefficients.

The output noise power is

$$P_v(n) = E\{|\mathbf{c}^H \mathbf{v}(n)|^2\} = E\{\mathbf{c}^H \mathbf{v}(n) \mathbf{v}^H(n) \mathbf{c}\} = \mathbf{c}^H \mathbf{R}_v(n) \mathbf{c} \quad (6.9.4)$$

where $\mathbf{R}_v(n)$ is the noise correlation matrix. The determination of the output SNR, and hence the subsequent optimization, depends on the nature of the signal $\mathbf{s}(n)$.

6.9.1 Deterministic Signal in Noise

In the deterministic signal case, the power of the signal is

$$P_s(n) = E\{|\alpha \mathbf{c}^H \mathbf{s}_0|^2\} = P_\alpha |\mathbf{c}^H \mathbf{s}_0|^2 \quad (6.9.5)$$

and therefore the output SNR can be written as

$$\text{SNR}(\mathbf{c}) = P_\alpha \frac{|\mathbf{c}^H \mathbf{s}_0|^2}{\mathbf{c}^H \mathbf{R}_v(n) \mathbf{c}} \quad (6.9.6)$$

White noise case. If the correlation matrix of the additive noise is given by $\mathbf{R}_v(n) = P_v \mathbf{I}$, the SNR becomes

$$\text{SNR}(\mathbf{c}) = \frac{P_\alpha}{P_v} \frac{|\mathbf{c}^H \mathbf{s}_0|^2}{\mathbf{c}^H \mathbf{c}} \quad (6.9.7)$$

which simplifies the maximization process. Indeed, from the Cauchy-Schwartz inequality

$$\mathbf{c}^H \mathbf{s}_0 \leq (\mathbf{c}^H \mathbf{c})^{1/2} (\mathbf{s}_0^H \mathbf{s}_0)^{1/2} \quad (6.9.8)$$

we conclude that the SNR in (6.9.7) attains its maximum value

$$\text{SNR}_{\max} = \frac{P_\alpha}{P_v} \mathbf{s}_0^H \mathbf{s}_0 \quad (6.9.9)$$

if the optimum filter \mathbf{c}_o is chosen as

$$\mathbf{c}_o = \kappa \mathbf{s}_0 \quad (6.9.10)$$

that is, when the filter is a scaled *replica* of the known signal shape. This property resulted in the term *matched filter*, which is widely used in communications and radar applications.[†] We note that if a vector \mathbf{c}_o maximizes the SNR (6.9.7), then any constant κ times \mathbf{c}_o maximizes the SNR as well. Therefore, we can choose this constant in any way we want. In this section, we choose κ_o so that $\mathbf{c}_o^H \mathbf{s}_0 = 1$.

Colored noise case. Using the Cholesky decomposition $\mathbf{R}_v = \mathbf{L}_v \mathbf{L}_v^H$ of the noise correlation matrix, we can write the SNR in (6.9.6) as

$$\text{SNR}(\mathbf{c}) = P_\alpha \frac{|(\mathbf{L}_v^H \mathbf{c})^H (\mathbf{L}_v^{-1} \mathbf{s}_0)|^2}{(\mathbf{L}_v^H \mathbf{c})^H (\mathbf{L}_v^H \mathbf{c})} \quad (6.9.11)$$

which, according to the Cauchy-Schwartz inequality, attains its maximum

$$\text{SNR}_{\max} = P_\alpha \|\mathbf{L}_v^{-1} \mathbf{s}_0\|^2 = P_\alpha \mathbf{s}_0^H \mathbf{R}_v^{-1} \mathbf{s}_0 \quad (6.9.12)$$

when the optimum filter satisfies $\mathbf{L}_v^H \mathbf{c}_o = \kappa \mathbf{L}_v^{-1} \mathbf{s}_0$, or equivalently

$$\mathbf{c}_o = \kappa \mathbf{R}_v^{-1} \mathbf{s}_0 \quad (6.9.13)$$

which provides the optimum matched filter for color additive noise. Again, the optimum filter can be scaled in any desirable way. We choose $\mathbf{c}_o^H \mathbf{s}_0 = 1$ which implies $\kappa = (\mathbf{s}_0^H \mathbf{R}_v^{-1} \mathbf{s}_0)^{-1}$.

If we pass the observed signal through the preprocessor \mathbf{L}_v^{-1} , we obtain a signal $\mathbf{L}_v^{-1} \mathbf{s}$ in additive white noise $\tilde{\mathbf{v}} = \mathbf{L}_v^{-1} \mathbf{v}$ because $E\{\tilde{\mathbf{v}} \tilde{\mathbf{v}}^H\} = E\{\mathbf{L}_v^{-1} \mathbf{v} \mathbf{v}^H \mathbf{L}_v^{-H}\} = \mathbf{I}$. Therefore, the optimum matched filter in additive color noise is the cascade of a whitening filter followed by a matched filter for white noise (compare with a similar decomposition for the optimum

[†]We note that the matched filter \mathbf{c}_o in (6.9.10) is not a complex conjugate reversed version of the signal \mathbf{s} . This happens when we define the matched filter as a convolution that involves a reversal of the impulse response (Therrien 1992).

Wiener filter in Figure 6.19). The application of the optimum matched filter is discussed in Section 11.3, which provides a more detailed treatment.

EXAMPLE 6.9.1. Consider a finite-duration deterministic signal $s(n) = a^n$, $0 \leq n \leq M - 1$, corrupted by additive noise $v(n)$ with autocorrelation sequence $r_v(l) = \sigma_0^2 \rho^{|l|}/(1 - \rho^2)$. We determine and plot the impulse response of an M th-order matched filter for $a = 0.6$, $M = 8$, $\sigma_0^2 = 0.25$, and (a) $\rho = 0.1$ and (b) $\rho = -0.8$. We first note that the signal vector is $\mathbf{s} = [1 \ a \ a^2 \ \dots \ a^7]^T$ and that the noise correlation matrix \mathbf{R}_v is Toeplitz with first row $[r_v(0) \ r_v(1) \ \dots \ r_v(7)]$. The optimum matched filters are determined by $\mathbf{c} = \mathbf{R}_v^{-1} \mathbf{s}_0$ and are shown in Figure 6.35. We notice that for $\rho = 0.1$ the matched filter looks like the signal because the correlation between the samples of the interference is very small; that is, the additive noise is close to white. For $\rho = -0.8$ the correlation increases, and the shape of the optimum filter differs more from the shape of the signal. However, as a result of the increased noise correlation, the optimum SNR increases.

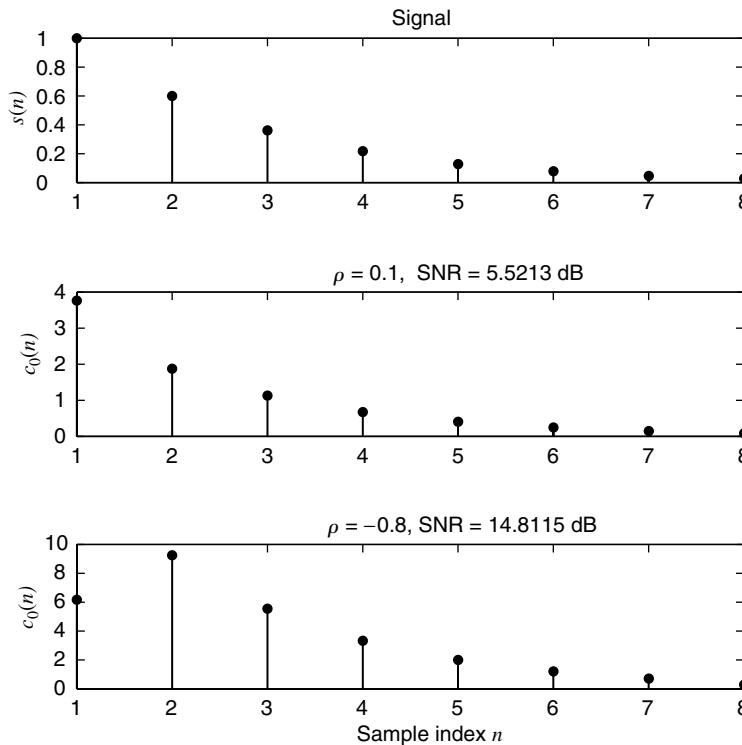


FIGURE 6.35
 Signal and impulse responses of the optimum matched filter that maximizes the SNR in the presence of additive color noise.

6.9.2 Random Signal in Noise

In the case of a random signal with known correlation matrix \mathbf{R}_s , the SNR is

$$\text{SNR}(\mathbf{c}) = \frac{\mathbf{c}^H \mathbf{R}_s \mathbf{c}}{\mathbf{c}^H \mathbf{R}_v \mathbf{c}} \quad (6.9.14)$$

that is, the ratio of two quadratic forms. We again distinguish two cases.

White noise case. If the correlation matrix of the noise is given by $\mathbf{R}_v = P_v \mathbf{I}$, we have

$$\text{SNR}(\mathbf{c}) = \frac{1}{P_v} \frac{\mathbf{c}^H \mathbf{R}_s \mathbf{c}}{\mathbf{c}^H \mathbf{c}} \quad (6.9.15)$$

which has the form of Rayleigh's quotient (Strang 1980; Leon 1998). By using the innovations transformation $\tilde{\mathbf{c}} = \mathbf{Q}^H \mathbf{c}$, where the unitary matrix \mathbf{Q} is obtained from the eigendecomposition $\mathbf{R}_s = \mathbf{Q} \Lambda \mathbf{Q}^H$, the SNR can be expressed as

$$\text{SNR}(\mathbf{c}) = \frac{1}{P_v} \frac{\tilde{\mathbf{c}}^H \Lambda \tilde{\mathbf{c}}}{\tilde{\mathbf{c}}^H \tilde{\mathbf{c}}} = \frac{1}{P_v} \frac{\lambda_1 |\tilde{c}_1|^2 + \cdots + \lambda_M |\tilde{c}_M|^2}{|\tilde{c}_1|^2 + \cdots + |\tilde{c}_M|^2} \quad (6.9.16)$$

where $0 \leq \lambda_1 \leq \cdots \leq \lambda_M$ are the eigenvalues of the signal correlation matrix. The SNR is maximized if we choose $\tilde{c}_M = 1$ and $\tilde{c}_1 = \cdots = \tilde{c}_{M-1} = 0$ and is minimized if we choose $\tilde{c}_1 = 1$ and $\tilde{c}_2 = \cdots = \tilde{c}_M = 0$. Therefore, for any positive definite matrix \mathbf{R}_s , we have

$$\lambda_{\min} \leq \frac{\mathbf{c}^H \mathbf{R}_s \mathbf{c}}{\mathbf{c}^H \mathbf{c}} \leq \lambda_{\max} \quad (6.9.17)$$

which is known as *Rayleigh's quotient* (Strang 1980). This implies that the optimum filter $\mathbf{c} = \mathbf{Q}\tilde{\mathbf{c}}$ is the eigenvector corresponding to the maximum eigenvalue of \mathbf{R}_s , that is,

$$\mathbf{c} = \mathbf{q}_{\max} \quad (6.9.18)$$

and provides a maximum SNR

$$\text{SNR}_{\max} = \frac{\lambda_{\max}}{P_v} \quad (6.9.19)$$

where $\lambda_{\max} = \lambda_M$. The obtained optimum filter is sometimes known as an *eigenfilter* (Makhoul 1981). The following example provides a geometric interpretation of these results for a second-order filter.

EXAMPLE 6.9.2. Suppose that the signal correlation matrix \mathbf{R}_s is given by (see Example 3.5.1)

$$\mathbf{R} \triangleq \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1-\rho & 0 \\ 0 & 1+\rho \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}^H = \mathbf{Q} \Lambda \mathbf{Q}^H$$

where $\rho = 0.81$. To obtain a geometric interpretation, we fix $\mathbf{c}^H \mathbf{c} = 1$ and try to maximize the numerator $\mathbf{c}^H \mathbf{R} \mathbf{c} > 0$ (we assume that \mathbf{R} is positive definite). The relation $c_1^2 + c_2^2 = 1$ represents a circle in the (c_1, c_2) plane. The plot can be easily obtained by using the parametric description $c_1 = \cos \phi$ and $c_2 = \sin \phi$. To obtain the plot of $\mathbf{c}^H \mathbf{R} \mathbf{c} = 1$, we note that

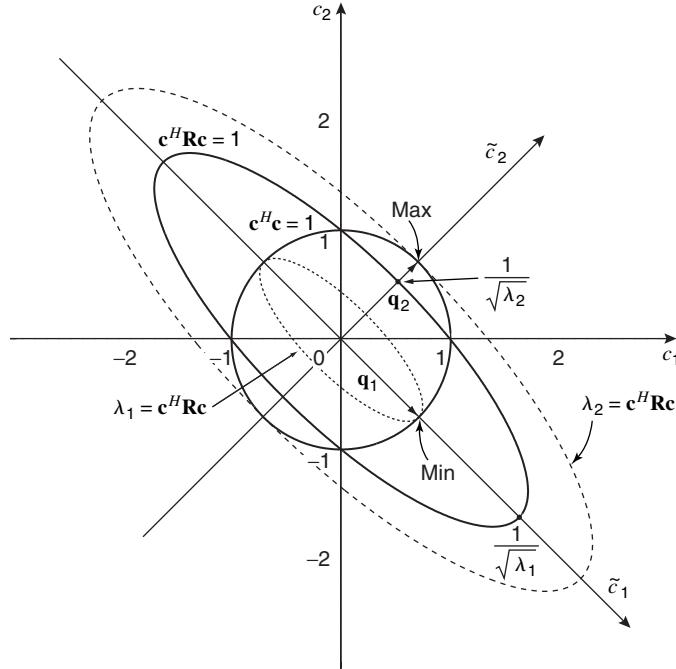
$$\mathbf{c}^H \mathbf{R} \mathbf{c} = \mathbf{c}^H \mathbf{Q} \Lambda \mathbf{Q}^H \mathbf{c} = \tilde{\mathbf{c}}^H \Lambda \tilde{\mathbf{c}} = \lambda_1^2 \tilde{c}_1^2 + \lambda_2^2 \tilde{c}_2^2 = 1$$

where $\tilde{\mathbf{c}} \triangleq \mathbf{Q}^H \mathbf{c}$. To plot $\lambda_1^2 \tilde{c}_1^2 + \lambda_2^2 \tilde{c}_2^2 = 1$, we use the parametric description $\tilde{c}_1 = \cos \phi / \sqrt{\lambda_1}$ and $\tilde{c}_2 = \sin \phi / \sqrt{\lambda_2}$. The result is an ellipse in the $(\tilde{c}_1, \tilde{c}_2)$ plane. For $\tilde{c}_2 = 0$ we have $\tilde{c}_1 = 1/\sqrt{\lambda_1}$, and for $\tilde{c}_1 = 0$ we have $\tilde{c}_2 = 1/\sqrt{\lambda_2}$. Since $\lambda_1 < \lambda_2$, $2/\sqrt{\lambda_1}$ provides the length of the major axis determined by the eigenvector $\mathbf{q}_1 = [1 \ -1]^T / \sqrt{2}$. Similarly, $2/\sqrt{\lambda_2}$ provides the length of the minor axis determined by the eigenvector $\mathbf{q}_2 = [1 \ 1]^T / \sqrt{2}$. The coordinates of the ellipse in the (c_1, c_2) plane are obtained by the rotation transformation $\mathbf{c} = \mathbf{Q}\tilde{\mathbf{c}}$. The resulting circle and ellipse are shown in Figure 6.36. The maximum value of $\mathbf{c}^H \mathbf{R} \mathbf{c} = \lambda_1 \tilde{c}_1^2 + \lambda_2 \tilde{c}_2^2$ on the circle $\tilde{c}_1^2 + \tilde{c}_2^2 = 1$ is obtained for $\tilde{c}_1 = 0$ and $\tilde{c}_2 = 1$, that is, at the endpoint of eigenvector \mathbf{q}_2 , and is equal to the largest eigenvalue λ_2 . Similarly, the minimum is λ_1 and is obtained at the tip of eigenvector \mathbf{q}_1 (see Figure 6.36). Therefore, the optimum filter is $\mathbf{c} = \mathbf{q}_2$ and the maximum SNR is λ_2 / P_v .

Colored noise case. Using the Cholesky decomposition $\mathbf{R}_v = \mathbf{L}_v \mathbf{L}_v^H$ of the noise correlation matrix, we process the observed signal with the transformation \mathbf{L}_v^{-1} , that is, we obtain

$$\begin{aligned} \mathbf{x}_v(n) &\triangleq \mathbf{L}_v^{-1} \mathbf{x}(n) = \mathbf{L}_v^{-1} \mathbf{s}(n) + \mathbf{L}_v^{-1} \mathbf{v}(n) \\ &= \mathbf{s}_v(n) + \tilde{\mathbf{v}}(n) \end{aligned} \quad (6.9.20)$$

where $\tilde{\mathbf{v}}(n)$ is white noise with $E\{\tilde{\mathbf{v}}(n)\tilde{\mathbf{v}}^H(n)\} = \mathbf{I}$ and $E\{\mathbf{s}_v(n)\mathbf{s}_v^H(n)\} = \mathbf{L}_v^{-1} \mathbf{R}_s \mathbf{L}_v^{-H}$. Therefore, the optimum matched filter is determined by the eigenvector corresponding to the maximum eigenvalue of matrix $\mathbf{L}_v^{-1} \mathbf{R}_s \mathbf{L}_v^{-H}$, that is, the correlation matrix of the transformed signal $\mathbf{s}_v(n)$.

**FIGURE 6.36**

Geometric interpretation of the optimization process for the derivation of the optimum eigenfilter using isopower contours for $\lambda_1 < \lambda_2$.

The problem can also be solved by using the simultaneous diagonalization of the signal and noise correlation matrices \mathbf{R}_s and \mathbf{R}_v , respectively. Starting with the decomposition $\mathbf{R}_v = \mathbf{Q}_v \Lambda_v \mathbf{Q}_v^H$, we compute the isotropic transformation

$$\begin{aligned} \mathbf{x}_v(n) &\triangleq \Lambda_v^{-1/2} \mathbf{Q}_v^H \mathbf{x}(n) \\ &= \Lambda_v^{-1/2} \mathbf{Q}_v^H \mathbf{s}(n) + \Lambda_v^{-1/2} \mathbf{Q}_v^H \mathbf{v}(n) \triangleq \tilde{\mathbf{s}}(n) + \tilde{\mathbf{v}}(n) \end{aligned} \quad (6.9.21)$$

where $E\{\tilde{\mathbf{v}}(n)\tilde{\mathbf{v}}^H(n)\} = \mathbf{I}$ and $E\{\tilde{\mathbf{s}}(n)\tilde{\mathbf{s}}^H(n)\} = \Lambda_v^{-1/2} \mathbf{Q}_v^H \mathbf{R}_s \mathbf{Q}_v \Lambda_v^{-1/2} \triangleq \mathbf{R}_{\tilde{s}}$. Since the noise vector is white, the optimum matched filter is determined by the eigenvector corresponding to the maximum eigenvalue of matrix $\mathbf{R}_{\tilde{s}}$.

Finally, if $\mathbf{R}_{\tilde{s}} = \mathbf{Q}_{\tilde{s}} \Lambda_{\tilde{s}} \mathbf{Q}_{\tilde{s}}^H$, the transformation

$$\mathbf{x}_{vs}(n) \triangleq \mathbf{Q}_{\tilde{s}}^H \mathbf{x}_v(n) = \mathbf{Q}_{\tilde{s}}^H \tilde{\mathbf{s}}(n) + \mathbf{Q}_{\tilde{s}}^H \tilde{\mathbf{v}}(n) \triangleq \bar{\mathbf{s}}(n) + \bar{\mathbf{v}}(n) \quad (6.9.22)$$

results in new signal and noise vectors with correlation matrices

$$E\{\bar{\mathbf{s}}(n)\bar{\mathbf{s}}^H(n)\} = \mathbf{Q}_{\tilde{s}}^H \mathbf{R}_{\tilde{s}} \mathbf{Q}_{\tilde{s}} = \Lambda_{\tilde{s}} \quad (6.9.23)$$

$$E\{\bar{\mathbf{v}}(n)\bar{\mathbf{v}}^H(n)\} = \mathbf{Q}_{\tilde{s}}^H \mathbf{I} \mathbf{Q}_{\tilde{s}} = \mathbf{I} \quad (6.9.24)$$

Therefore, the transformation matrix

$$\mathbf{Q} \triangleq \mathbf{Q}_{\tilde{s}}^H \Lambda_v^{-1/2} \mathbf{Q}_v^H \quad (6.9.25)$$

diagonalizes matrices \mathbf{R}_s and \mathbf{R}_v simultaneously (Fukunaga 1990).

The maximization of (6.9.14) can also be obtained by whitening the signal, that is, by using the Cholesky decomposition $\mathbf{R}_s = \mathbf{L}_s \mathbf{L}_s^H$ of the signal correlation matrix. Indeed, using the transformation $\tilde{\mathbf{c}} \triangleq \mathbf{L}_s^H \mathbf{c}$, we have

$$\text{SNR}(\mathbf{c}) = \frac{\mathbf{c}^H \mathbf{R}_s \mathbf{c}}{\mathbf{c}^H \mathbf{R}_v \mathbf{c}} = \frac{\mathbf{c}^H \mathbf{L}_s \mathbf{L}_s^H \mathbf{c}}{\mathbf{c}^H \mathbf{L}_s \mathbf{L}_s^{-1} \mathbf{R}_v \mathbf{L}_s^{-H} \mathbf{L}_s^H \mathbf{c}} = \frac{\tilde{\mathbf{c}}^H \tilde{\mathbf{c}}}{\tilde{\mathbf{c}}^H \mathbf{L}_s^{-1} \mathbf{R}_v \mathbf{L}_s^{-H} \tilde{\mathbf{c}}} \quad (6.9.26)$$

which attains its maximum when $\tilde{\mathbf{c}}$ is equal to the eigenvector corresponding to the minimum eigenvalue of matrix $\mathbf{L}_s^{-1} \mathbf{R}_v \mathbf{L}_s^{-H}$. This approach has been used to obtain optimum moving-target indicator filters for radar applications (Hsiao 1974).

EXAMPLE 6.9.3. The basic problem in many radar detection systems is the separation of a useful signal from colored noise or interference background. In several cases the signal is a point target (i.e., it can be modeled as a unit impulse) or is random with a flat PSD, that is, $\mathbf{R}_s = P_a \mathbf{I}$. Suppose that the background is colored with correlation $r_v(i, j) = \rho^{(i-j)^2}$, $1 \leq i, j \leq M$, which leads to a Toeplitz correlation matrix \mathbf{R}_v . We determine and compare three filters for interference rejection. The first is a matched filter that maximizes the SNR

$$\text{SNR}(\mathbf{c}) = \frac{P_a \mathbf{c}^H \mathbf{c}}{\mathbf{c}^H \mathbf{R}_v \mathbf{c}} \quad (6.9.27)$$

by setting \mathbf{c} equal to the eigenvector corresponding to the minimum eigenvalue of \mathbf{R}_v . The second approach is based on the method of linear prediction. Indeed, if we assume that the interference $v_k(n)$ is much stronger than the useful signal $s_k(n)$, we can obtain an estimate $\hat{v}_1(n)$ of $v_1(n)$ using the observed samples $\{x_k(n)\}_2^M$ and then subtract $\hat{v}_1(n)$ from $x_1(n)$ to cancel the interference. The Wiener filter with desired response $y(n) = v_1(n)$ and input data $\{x_k(n)\}_2^M$ is

$$\hat{y}(n) = - \sum_{k=1}^{M-1} a_k^* x_{k+1}(n) \triangleq -\mathbf{a}^H \tilde{\mathbf{x}}(n)$$

and is specified by the normal equations

$$\tilde{\mathbf{R}}_x \mathbf{a} = -\tilde{\mathbf{d}}$$

and the MMSE

$$P_f^f = E\{|v_1|^2\} + \tilde{\mathbf{d}}^H \mathbf{a}$$

$$\text{where } \langle \tilde{\mathbf{R}}_x \rangle_{ij} = E\{x_{i+1}(n)x_{j+1}^*(n)\} \simeq E\{v_{i+1}(n)v_{j+1}^*(n)\}$$

$$\text{and } \tilde{d}_i = E\{v_1 x_{i+1}^*(n)\} \simeq E\{v_1(n)v_{i+1}^*(n)\}$$

because the interference is assumed much stronger than the signal. Using the last four equations, we obtain

$$\mathbf{R}_v \begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} E^f \\ \mathbf{0} \end{bmatrix} \quad (6.9.28)$$

which corresponds to the forward linear prediction error (LPE) filter discussed in Section 6.5.2. Finally, for the sake of comparison, we consider the binomial filters $H_M(z) = (1 - z^{-1})^M$ that are widely used in radar systems for the elimination of stationary (i.e., nonmoving) clutter. Figure 6.37 shows the magnitude response of the three filters for $\rho = 0.9$ and $M = 4$. We emphasize that

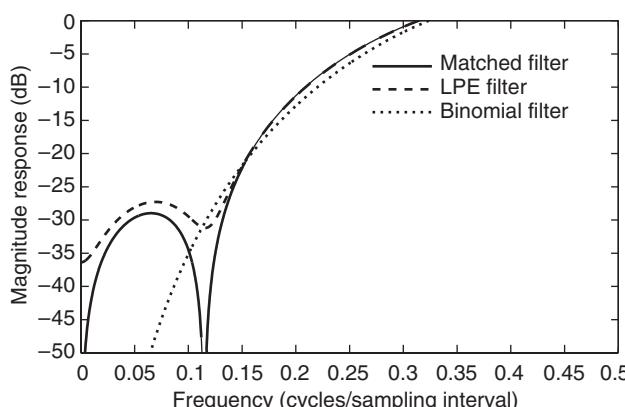


FIGURE 6.37
Comparison of frequency responses of matched filter, prediction error filter, and binomial interference rejection filter.

6.10 SUMMARY

In this chapter, we discussed the theory and application of optimum linear filters designed by minimizing the MSE criterion of performance. Our goal was to explain the characteristics of each criterion, emphasize when its use made sense, and illustrate its meaning in the context of practical applications.

We started with linear processors that formed an estimate of the desired response by combining a set of different signals (data) and showed that the parameters of the optimum processor can be obtained by solving a linear system of equations (normal equations). The matrix and the right-hand side vector of the normal equations are completely specified by the second-order moments of the input data and the desired response. Next, we used the developed theory to design optimum FIR filters, linear signal estimators, and linear predictors.

We emphasized the case of stationary stochastic processes and showed that the resulting optimum estimators are time-invariant. Therefore, we need to design only one optimum filter that can be used to process all realizations of the underlying stochastic processes. Although another filter may perform better for some realizations, that is, the estimated MSE is smaller than the MMSE, on average (i.e., when we consider all possible realizations), the optimum filter is the best.

We showed that the performance of optimum linear filters improves as we increase the number of filter coefficients. Therefore, the noncausal IIR filter provides the best possible performance and can be used as a yardstick to assess other filters. Because IIR filters involve an infinite number of parameters, their design involves linear equations with an infinite number of unknowns. For stationary processes, these equations take the form of a convolution equation that can be solved using z -transform techniques. If we use a pole-zero structure, the normal equations become nonlinear and the design of the optimum filter is complicated by the presence of multiple local minima.

Then we discussed the design of optimum filters for inverse system modeling and blind deconvolution, and we provided a detailed discussion of their use in the important practical application of channel equalization for data transmission systems.

Finally, we provided a concise introduction to the design of optimum matched filters and eigenfilters that maximize the output SNR and find applications for the detection of signals in digital communication and radar systems.

PROBLEMS

- 6.1** Let \mathbf{x} be a random vector with mean $E\{\mathbf{x}\}$. Show that the linear MMSE estimate \hat{y} of a random variable y using the data vector \mathbf{x} is given by $\hat{y} = y_o + \mathbf{c}^H \mathbf{x}$, where $y_o = E\{y\} - \mathbf{c}^H E\{\mathbf{x}\}$, $\mathbf{c} = \mathbf{R}^{-1} \mathbf{d}$, $\mathbf{R} = E\{\mathbf{x}\mathbf{x}^H\}$, and $\mathbf{d} = E\{\mathbf{x}y^*\}$.
- 6.2** Consider an optimum FIR filter specified by the input correlation matrix $\mathbf{R} = \text{Toeplitz } \{1, \frac{1}{4}\}$ and cross-correlation vector $\mathbf{d} = [1 \ \frac{1}{2}]^T$.
 - (a) Determine the optimum impulse response \mathbf{c}_o and the MMSE P_o .
 - (b) Express \mathbf{c}_o and P_o in terms of the eigenvalues and eigenvectors of \mathbf{R} .
- 6.3** Repeat Problem 6.2 for a third-order optimum FIR filter.

- 6.4** A process $y(n)$ with the autocorrelation $r_y(l) = a^{|l|}$, $-1 < a < 1$, is corrupted by additive, uncorrelated white noise $v(n)$ with variance σ_v^2 . To reduce the noise in the observed process $x(n) = y(n) + v(n)$, we use a first-order Wiener filter.

- (a) Express the coefficients $c_{o,1}$ and $c_{o,2}$ and the MMSE P_o in terms of parameters a and σ_v^2 .
- (b) Compute and plot the PSD of $x(n)$ and the magnitude response $|C_o(e^{j\omega})|$ of the filter when $\sigma_v^2 = 2$, for both $a = 0.8$ and $a = -0.8$, and compare the results.
- (c) Compute and plot the processing gain of the filter for $a = -0.9, -0.8, -0.7, \dots, 0.9$ as a function of a and comment on the results.

- 6.5** Consider the harmonic process $y(n)$ and its noise observation $x(n)$ given in Example 6.4.1.

- (a) Show that $r_y(l) = \frac{1}{2}A^2 \cos \omega_0 l$.
- (b) Write a Matlab function $h = \text{opt_fir}(A, f_0, \text{var_v}, M)$ to design an M th-order optimum FIR filter impulse response $h(n)$. Use the `toeplitz` function from MATLAB to generate correlation matrix \mathbf{R} .
- (c) Determine the impulse response of a 20th-order optimum FIR filter for $A = 0.5$, $f_0 = 0.05$, and $\sigma_v^2 = 0.5$.
- (d) Using MATLAB, determine and plot the magnitude response of the above-designed filter, and verify your results with those given in Example 6.4.1.

- 6.6** Consider a “desired” signal $s(n)$ generated by the process $s(n) = -0.8w(n-1) + w(n)$, where $w(n) \sim \text{WN}(0, \sigma_w^2)$. This signal is passed through the causal system $H(z) = 1 - 0.9z^{-1}$ whose output $y(n)$ is corrupted by additive white noise $v(n) \sim \text{WN}(0, \sigma_v^2)$. The processes $w(n)$ and $v(n)$ are uncorrelated with $\sigma_w^2 = 0.3$ and $\sigma_v^2 = 0.1$.

- (a) Design a second-order optimum FIR filter that estimates $s(n)$ from the signal $x(n) = y(n) + v(n)$ and determine \mathbf{c}_o and P_o .
- (b) Plot the error performance surface, and verify that it is quadratic and that the optimum filter points to its minimum.
- (c) Repeat part (a) for a third-order filter, and see whether there is any improvement.

- 6.7** Repeat Problem 6.6, assuming that the desired signal is generated by $s(n) = -0.8s(n-1) + w(n)$.

- 6.8** Repeat Problem 6.6, assuming that $H(z) = 1$.

- 6.9** A stationary process $x(n)$ is generated by the difference equation $x(n) = \rho x(n-1) + w(n)$, where $w(n) \sim \text{WN}(0, \sigma_w^2)$.

- (a) Show that the correlation matrix of $x(n)$ is given by

$$\mathbf{R}_x = \frac{\sigma_w^2}{1 - \rho^2} \text{Toeplitz}\{1, \rho, \rho^2, \dots, \rho^{M-1}\}$$

- (b) Show that the M th-order FLP is given by $a_1^{(M)} = -\rho$, $a_k^{(M)} = 0$ for $k > 1$ and the MMSE is $P_M^f = \sigma_w^2$.

- 6.10** Using Parseval’s theorem, show that (6.4.18) can be written as (6.4.21) in the frequency domain.

- 6.11** By differentiating (6.4.21) with respect to $H(e^{j\omega})$, derive the frequency response function $H_o(e^{j\omega})$ of the optimum filter in terms of $R_{yx}(e^{j\omega})$ and $R_x(e^{j\omega})$.

- 6.12** A conjugate symmetric linear smoother is obtained from (6.5.12) when $M = 2L$ and $i = L$. If the process $x(n)$ is stationary, then, using $\bar{\mathbf{R}}\mathbf{J} = \mathbf{J}\bar{\mathbf{R}}^*$, show that $\bar{\mathbf{c}} = \mathbf{J}\bar{\mathbf{c}}^*$.

- 6.13** Let $\bar{\mathbf{Q}}$ and $\bar{\Lambda}$ be the matrices from the eigendecomposition of $\bar{\mathbf{R}}$, that is, $\bar{\mathbf{R}} = \bar{\mathbf{Q}}\bar{\Lambda}\bar{\mathbf{Q}}^H$.

- (a) Substitute \mathbf{R} into (6.5.20) and (6.5.27) to prove (6.5.43) and (6.5.44).

- (b) Generalize the above result for a j th-order linear signal estimator $\mathbf{c}^{(j)}(n)$; that is, prove that

$$\mathbf{c}^{(j)}(n) = P_o^{(j)}(n) \sum_{i=1}^{M+1} \frac{1}{\bar{\lambda}_i} \bar{\mathbf{q}}_i \bar{q}_{i,j}$$

6.14 Let $\tilde{\mathbf{R}}(n)$ be the inverse of the correlation matrix $\bar{\mathbf{R}}(n)$ given in (6.5.11).

- (a) Using (6.5.12), show that the diagonal elements of $\tilde{\mathbf{R}}(n)$ are given by

$$\langle \tilde{\mathbf{R}}(n) \rangle_{i,i} = \frac{1}{P^{(i)}(n)} \quad 1 \leq i \leq M+1$$

- (b) Furthermore, show that

$$\mathbf{c}^{(i)}(n) = \frac{\tilde{\mathbf{r}}_i(n)}{\langle \tilde{\mathbf{R}}(n) \rangle_{i,i}} \quad 1 \leq i \leq M+1$$

where $\tilde{\mathbf{r}}_i(n)$ is the i -th column of $\tilde{\mathbf{R}}(n)$.

6.15 The first five samples of the autocorrelation sequence of a signal $x(n)$ are $r(0) = 1, r(1) = 0.8, r(2) = 0.6, r(3) = 0.4$, and $r(4) = 0.3$. Compute the FLP, the BLP, the optimum symmetric smoother, and the corresponding MMSE (a) by using the normal equations method and (b) by using the inverse of the normal equations matrix.

6.16 For the symmetric, Toeplitz autocorrelation matrix $\mathbf{R} = \text{Toeplitz}\{r(0), r(1), r(2)\} = r(0) \times \text{Toeplitz}\{1, \rho_1, \rho_2\}$ with $\mathbf{R} = \mathbf{L}\mathbf{D}\mathbf{L}^H$ and $\mathbf{D} = \text{diag}\{\xi_1, \xi_2, \xi_3\}$, the following conditions are equivalent:

- \mathbf{R} is positive definite.
- $\xi_i > 0$ for $1 \leq i \leq 3$.
- $|k_i| < 1$ for $1 \leq i \leq 3$.

Determine the values of ρ_1 and ρ_2 for which \mathbf{R} is positive definite, and plot the corresponding area in the (ρ_1, ρ_2) plane.

6.17 Prove the first equation in (6.5.45) by rearranging the FLP normal equations in terms of the unknowns $P_o^f(n), a_1(n), \dots, a_M(n)$ and then solve for $P_o^f(n)$, using Cramer's rule. Repeat the procedure for the second equation.

6.18 Consider the signal $x(n) = y(n) + v(n)$, where $y(n)$ is a useful random signal corrupted by noise $v(n)$. The processes $y(n)$ and $v(n)$ are uncorrelated with PSDs

$$R_y(e^{j\omega}) = \begin{cases} 1 & 0 \leq |\omega| \leq \frac{\pi}{2} \\ 0 & \frac{\pi}{2} < |\omega| \leq \pi \end{cases}$$

and

$$R_v(e^{j\omega}) = \begin{cases} 1 & \frac{\pi}{4} \leq |\omega| \leq \frac{\pi}{2} \\ 0 & 0 \leq |\omega| < \frac{\pi}{4} \text{ and } \frac{\pi}{2} < |\omega| \leq \pi \end{cases}$$

respectively. (a) Determine the optimum IIR filter and find the MMSE. (b) Determine a third-order optimum FIR filter and the corresponding MMSE. (c) Determine the noncausal optimum FIR filter defined by

$$\hat{y}(n) = h(-1)x(n+1) + h(0)x(n) + h(1)x(n-1)$$

and the corresponding MMSE.

6.19 Consider the ARMA(1, 1) process $x(n) = 0.8x(n-1) + w(n) + 0.5w(n-1)$, where $w(n) \sim \text{WGN}(0, 1)$. (a) Determine the coefficients and the MMSE of (1) the one-step ahead FLP $\hat{x}(n) = a_1x(n-1) + a_2x(n-2)$ and (2) the two-step ahead FLP $\hat{x}(n+1) = a_1x(n-1) + a_2x(n-2)$. (b) Check if the obtained prediction error filters are minimum-phase, and explain your findings.

6.20 Consider a random signal $x(n) = s(n) + v(n)$, where $v(n) \sim \text{WGN}(0, 1)$ and $s(n)$ is the AR(1) process $s(n) = 0.9s(n - 1) + w(n)$, where $w(n) \sim \text{WGN}(0, 0.64)$. The signals $s(n)$ and $v(n)$ are uncorrelated. (a) Determine and plot the autocorrelation $r_s(l)$ and the PSD $R_s(e^{j\omega})$ of $s(n)$.

- (b) Design a second-order optimum FIR filter to estimate $s(n)$ from $x(n)$. What is the MMSE?
 (c) Design an optimum IIR filter to estimate $s(n)$ from $x(n)$. What is the MMSE?

6.21 A useful signal $s(n)$ with PSD $R_s(z) = [(1 - 0.9z^{-1})(1 - 0.9z)]^{-1}$ is corrupted by additive uncorrelated noise $v(n) \sim \text{WN}(0, \sigma^2)$. (a) The resulting signal $x(n) = s(n) + v(n)$ is passed through a causal filter with system function $H(z) = (1 - 0.8z^{-1})^{-1}$. Determine (1) the SNR at the input, (2) the SNR at the output, and (3) the processing gain, that is, the improvement in SNR. (b) Determine the causal optimum filter and compare its performance with that of the filter in (a).

6.22 A useful signal $s(n)$ with PSD $R_s(z) = 0.36[(1 - 0.8z^{-1})(1 - 0.8z)]^{-1}$ is corrupted by additive uncorrelated noise $v(n) \sim \text{WN}(0, 1)$. Determine the optimum noncausal and causal IIR filters, and compare their performance by examining the MMSE and their magnitude response. Hint: Plot the magnitude responses on the same graph with the PSDs of signal and noise.

6.23 Consider a process with PSD $R_x(z) = \sigma^2 H_x(z)H_x(z^{-1})$. Determine the D -step ahead linear predictor, and show that the MMSE is given by $P^{(D)} = \sigma^2 \sum_{n=0}^{D-1} |h_x|^2(n)$. Check your results by using the PSD $R_x(z) = (1 - a^2)[(1 - az^{-1})(1 - az)]^{-1}$.

6.24 Let $x(n) = s(n) + v(n)$ with $R_v(z) = 1$, $R_{sv}(z) = 0$, and

$$R_s(z) = \frac{0.75}{(1 - 0.5z^{-1})(1 - 0.5z)}$$

Determine the optimum filters for the estimation of $s(n)$ and $s(n - 2)$ from $\{x(k)\}_{-\infty}^n$ and the corresponding MMSEs.

6.25 For the random signal with PSD

$$R_x(z) = \frac{(1 - 0.2z^{-1})(1 - 0.2z)}{(1 - 0.9z^{-1})(1 - 0.9z)}$$

determine the optimum two-step ahead linear predictor and the corresponding MMSE.

6.26 Repeat Problem 6.25 for

$$R_x(z) = \frac{1}{(1 - 0.2z^{-1})(1 - 0.2z)(1 - 0.9z^{-1})(1 - 0.9z)}$$

6.27 Let $x(n) = s(n) + v(n)$ with $v(n) \sim \text{WN}(0, 1)$ and $s(n) = 0.6s(n - 1) + w(n)$, where $w(n) \sim \text{WN}(0, 0.82)$. The processes $s(n)$ and $v(n)$ are uncorrelated. Determine the optimum filters for the estimation of $s(n)$, $s(n + 2)$, and $s(n - 2)$ from $\{x(k)\}_{-\infty}^n$ and the corresponding MMSEs.

6.28 Repeat Problem 6.27 for $R_s(z) = [(1 - 0.5z^{-1})(1 - 0.5z)]^{-1}$, $R_v(z) = 5$, and $R_{sv}(z) = 0$.

6.29 Consider the random sequence $x(n)$ generated in Example 6.5.2

$$x(n) = w(n) + \frac{1}{2}w(n - 1)$$

where $w(n)$ is $\text{WN}(0, 1)$. Generate $K = 100$ sample functions $\{w_k(n)\}_{n=0}^N$, $k = 1, \dots, K$ of $w(n)$, in order to generate K sample functions $\{x_k(n)\}_{n=0}^N$, $k = 1, \dots, K$ of $x(n)$.

- (a) Use the second-order FLP a_k to obtain predictions $\{\hat{x}_k^f(n)\}_{n=2}^N$ of $x_k(n)$, for $k = 1, \dots, K$. Then determine the average error

$$\hat{P}^f = \frac{1}{N-1} \sum_{n=2}^N |x_k(n) - \hat{x}_k^f(n)|^2 \quad k = 1, \dots, K$$

and plot it as a function of k . Compare it with P_o^f .

- (b) Use the second-order BLP b_k to obtain predictions $\{\hat{x}_k^b(n)\}_{n=0}^{N-2}$, $k = 1, \dots, K$ of $x_k(n)$. Then determine the average error

$$\hat{P}^b = \frac{1}{N-1} \sum_{n=0}^{N-2} |x_k(n) - \hat{x}_k^b(n)|^2 \quad k = 1, \dots, K$$

and plot it as a function of k . Compare it with P_o^b .

- (c) Use the second-order symmetric linear smoother c_k to obtain smooth estimates $\{\hat{x}_k^s(n)\}_{n=0}^{N-2}$ of $x_k(n)$ for $k = 1, \dots, K$. Determine the average error

$$\hat{P}^s = \frac{1}{N-1} \sum_{n=1}^{N-1} |x_k(n) - \hat{x}_k^s(n)|^2 \quad k = 1, \dots, K$$

and plot it as a function of k . Compare it with P_o^s .

- 6.30** Let $x(n) = y(n) + v(n)$ be a wide-sense stationary process. The linear, symmetric smoothing filter estimator of $y(n)$ is given by

$$\hat{y}(n) = \sum_{k=-L}^L c_k^s x(n-k)$$

- (a) Determine the normal equations for the optimum MMSE filter.
(b) Show that the smoothing filter c_o^s has linear phase.
(c) Use the Lagrange multiplier method to determine the MMSE M th-order estimator $\hat{y}(n) = \mathbf{c}^H \mathbf{x}(n)$, where $M = 2L + 1$, when the filter vector \mathbf{c} is constrained to be conjugate symmetric, that is, $\mathbf{c} = \mathbf{J}\mathbf{c}^*$. Compare the results with those obtained in part (a).

- 6.31** Consider the causal prediction filter discussed in Example 6.6.1. To determine $H_c^{[D]}(z)$, first compute the causal part of the z -transform $[R'_{yw}(z)]_+$. Next compute $H_c^{[D]}(z)$ by using (6.6.21).

- (a) Determine $h_c^{[D]}(n)$.
(b) Using the above $h_c^{[D]}(n)$, show that

$$P_c^{[D]} = 1 - \frac{5}{8}(\frac{4}{5})^{2D}$$

- 6.32** Consider the causal smoothing filter discussed in Example 6.6.1.

- (a) Using $[r'_{yw}(l)]_+ = r_{yw}(l+D)u(l)$, $D < 0$, show that $[r'_{yw}(l)]_+$ can be put in the form

$$[r'_{yw}(l)]_+ = \frac{3}{5}(\frac{4}{5})^{l+D}u(l+D) + \frac{3}{5}(2^{l+D})[u(l) - u(l+D)] \quad D < 0$$

- (b) Hence, show that $[R'_{yw}(z)]_+$ is given by

$$[R'_{yw}(z)]_+ = \frac{3}{5} \frac{z^D}{1 - \frac{4}{5}z^{-1}} + \frac{3}{5}(2^D) \sum_{l=0}^{-D-1} 2^l z^{-l}$$

- (c) Finally using (6.6.21), prove (6.6.54).

- 6.33** In this problem, we will prove (6.6.57)

- (a) Starting with (6.6.42), show that $[R'_{yw}(z)]_+$ can also be put in the form

$$[R'_{yw}(z)]_+ = \frac{3}{5} \left(\frac{z^D}{1 - \frac{4}{5}z^{-1}} + \frac{2^D - z^D}{1 - 2z^{-1}} \right)$$

- (b) Now, using (6.6.21), show that

$$H_c^{[D]}(z) = \frac{3}{8} \left[\frac{2^D(1 - \frac{4}{5}z^{-1}) + \frac{3}{5}z^{D-1}}{(1 - \frac{4}{5}z^{-1})(1 - 2z^{-1})} \right]$$

hence, show that

$$\lim_{D \rightarrow -\infty} H_c^{[D]}(z) = \frac{9}{40} \left[\frac{z^D}{(1 - \frac{4}{5}z^{-1})(1 - 2z^{-1})} \right] = z^D H_{nc}(z)$$

(c) Finally, show that $\lim_{D \rightarrow \infty} P_c^{[D]} = P_{nc}$.

- 6.34** Consider the block diagram of a simple communication system shown in Figure 6.38. The information resides in the signal $s(n)$ produced by exciting the system $H_1(z) = 1/(1 + 0.95z^{-1})$ with the process $w(n) \sim \text{WGN}(0, 0.3)$. The signal $s(n)$ propagates through the channel $H_2(z) = 1/(1 - 0.85z^{-1})$, and is corrupted by the additive noise process $v(n) \sim \text{WGN}(0, 0.1)$, which is uncorrelated with $w(n)$. (a) Determine a second-order optimum FIR filter ($M = 2$) that estimates the signal $s(n)$ from the received signal $x(n) = z(n) + v(n)$. What is the corresponding MMSE P_o ? (b) Plot the error performance surface and verify that the optimum filter corresponds to the bottom of the “bowl.” (c) Use a Monte Carlo simulation (100 realizations with a 1000-sample length each) to verify the theoretically obtained MMSE in part (a). (d) Repeat part (a) for $M = 3$ and check if there is any improvement. Hint: To compute the autocorrelation of $z(n)$, notice that the output of $H_1(z)H_2(z)$ is an AR(2) process.

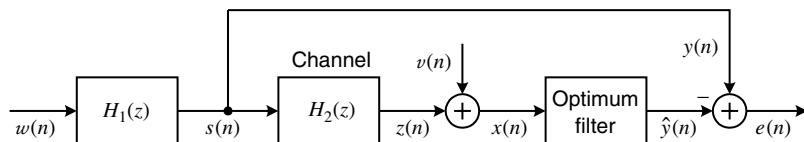


FIGURE 6.38

Block diagram of simple communication system used in Problem 6.34.

- 6.35** Write a program to reproduce the results shown in Figure 6.35 of Example 6.9.1. (a) Produce plots for $\rho = 0.1, -0.8, 0.8$. (b) Repeat part (a) for $M = 16$. Compare the plots obtained in (a) and (b) and justify any similarities or differences.

- 6.36** Write a program to reproduce the plot shown in Figure 6.36 of Example 6.9.2. Repeat for $\rho = -0.81$ and explain the similarities and differences between the two plots.

- 6.37** In this problem we study in greater detail the interference rejection filters discussed in Example 6.9.3. (a) Shows that SNRs for the matched filter and FLP filter are given by

<i>M</i>	Matched filter	FLP filter
2	$\frac{1}{1 - \rho}$	$\frac{1 + \rho^2}{1 - \rho^2}$
3	$\frac{2}{2 + \rho^4(1 - \sqrt{1 + 8\rho^{-6}})}$	$\frac{1 + \rho^2 + 3\rho^4 + \rho^6}{(\rho^2 - 1)(\rho^4 - 1)}$

and check the results numerically. (b) Compute and plot the SNRs and compare the performance of both filters for $M = 2, 3, 4$ and $\rho = 0.6, 0.8, 0.9, 0.95, 0.99$, and 0.995 . For what values of ρ and M do the two methods give similar results? Explain your conclusions. (c) Plot the magnitude response of the matched, FLP, and binomial filters for $M = 3$ and $\rho = 0.9$. Why does the optimum matched filter always have some nulls in its frequency response?

- 6.38** Determine the matched filter for the deterministic pulse $s(n) = \cos \omega_0 n$ for $0 \leq n \leq M - 1$ and zero elsewhere when the noise is (a) white with variance σ_v^2 and (b) colored with autocorrelation $r_v(l) = \sigma_v^2 \rho^{|l|}/(1 - \rho^2)$, $-1 < \rho < 1$. Plot the frequency response of the filter and superimpose

it on the noise PSD, for $\omega_0 = \pi/6$, $M = 12$, $\sigma_v^2 = 1$, and $\rho = 0.9$. Explain the shape of the obtained response. (c) Study the effect of the SNR in part (a) by varying the value of σ_v^2 . (d) Study the effect of the noise correlation in part (c) by varying the value of ρ .

- 6.39** Consider the equalization experiment in Example 6.8.1 with $M = 11$ and $D = 7$. (a) Compute and plot the magnitude response $|H(e^{j\omega})|$ of the channel and $|C_o(e^{j\omega})|$ of the optimum equalizer for $W = 2.9, 3.1, 3.3$, and 3.5 and comment upon the results. (b) For the same values of W , compute the spectral dynamic range $|H(e^{j\omega})|_{\max}/|H(e^{j\omega})|_{\min}$ of the channel and the eigenvalue spread $\lambda_{\max}/\lambda_{\min}$ of the $M \times M$ input correlation matrix. Explain how the variation in one affects the other.
- 6.40** In this problem we clarify some of the properties of the MSE equalizer discussed in Example 6.8.1. (a) Compute and plot the MMSE P_o as a function of M , and recommend how to choose a “reasonable” value. (b) Compute and plot P_o as a function of the delay D for $0 \leq D \leq 11$. What is the best value of D ? (c) Study the effect of input SNR upon P_o for $M = 11$ and $D = 7$ by fixing $\sigma_y^2 = 1$ and varying σ_v^2 .
- 6.41** In this problem we formulate the design of optimum linear signal estimators (LSE) using a constrained optimization framework. To this end we consider the estimator $e(n) = c_0^*x(n) + \dots + c_M^*x(n - M) \triangleq \mathbf{c}^H \mathbf{x}(n)$ and we wish to minimize the output power $E\{|e(n)|^2\} = \mathbf{c}^H \mathbf{R} \mathbf{c}$. To prevent the trivial solution $\mathbf{c} = \mathbf{0}$ we need to impose some constraint on the filter coefficients and use Lagrange multipliers to determine the minimum. Let \mathbf{u}_i be an $M \times 1$ vector with one at the i th position and zeros elsewhere. (a) Show that minimizing $\mathbf{c}^H \mathbf{R} \mathbf{c}$ under the linear constraint $\mathbf{u}_i^T \mathbf{c} = 1$ provides the following estimators: FLP if $i = 0$, BLP if $i = M$, and linear smoother if $i \neq 0, M$. (b) Determine the appropriate set of constraints for the L -steps ahead linear predictor, defined by $c_0 = 1$ and $\{c_k = 0\}_1^{L-1}$, and solve the corresponding constrained optimization problem. Verify your answer by obtaining the normal equations using the orthogonality principle. (c) Determine the optimum linear estimator by minimizing $\mathbf{c}^H \mathbf{R} \mathbf{c}$ under the quadratic constraints $\mathbf{c}^H \mathbf{c} = 1$ and $\mathbf{c}^H \mathbf{W} \mathbf{c} = 1$ (\mathbf{W} is a positive definite matrix) which impose a constraint on the length of the filter vector.

Algorithms and Structures for Optimum Linear Filters

The design and application of optimum filters involves (1) the solution of the normal equations to determine the optimum set of coefficients, (2) the evaluation of the cost function to determine whether the obtained parameters satisfy the design requirements, and (3) the implementation of the optimum filter, that is, the computation of its output that provides the estimate of the desired response.

The normal equations can be solved by using any general-purpose routine for linear simultaneous equations. However, there are several important reasons to study the normal equations in greater detail in order to develop efficient, special-purpose algorithms for their solution. First, the throughput of several real-time applications can only be served with serial or parallel algorithms that are obtained by exploiting the special structure (e.g., Toeplitz) of the correlation matrix. Second, sometimes we can develop order-recursive algorithms that help us to choose the correct filter order or to stop the algorithm before the manifestation of numerical problems. Third, some algorithms lead to intermediate sets of parameters that have physical meaning, provide easy tests for important properties (e.g., minimum phase), or are useful in special applications (e.g., data compression). Finally, sometimes there is a link between the algorithm for the solution of the normal equations and the structure for the implementation of the optimum filter.

In this chapter, we present different algorithms for the solution of the normal equations, the computation of the minimum mean square error (MMSE), and the implementation of the optimum filter. We start in Section 7.1 with a discussion of some results from matrix algebra that are useful for the development of order-recursive algorithms and introduce an algorithm for the order-recursive computation of the LDL^H decomposition, the MMSE, and the optimum estimate in the general case. In Section 7.2, we present some interesting interpretations for the various introduced algorithmic quantities and procedures that provide additional insight into the optimum filtering problem.

The only assumption we have made so far is that we know the required second-order statistics; hence, the results apply to any linear estimation problem: array processing, filtering, and prediction of nonstationary or stationary processes. In the sequel, we impose additional constraints on the input data vector and show how to exploit them in order to simplify the general algorithms and structures or specify new ones. In Section 7.3, we explore the shift invariance of the input data vector to develop a time-varying lattice-ladder structure for the optimum filter. However, to derive an order-recursive algorithm for the computation of either the direct or lattice-ladder structure parameters of the optimum time-varying filter, we need an analytical description of the changing second-order statistics of the nonstationary input process. Recall that in the simplest case of stationary processes, the correlation matrix is constant and Toeplitz. As a result, the optimum FIR filters and predictors are time-invariant, and their direct or lattice-ladder structure parameters can be computed (only once) using efficient, order-recursive algorithms due to Levinson and Durbin (Section 7.4) or Schür (Section 7.6). Section 7.5 provides a derivation of the lattice-ladder structures for

optimum filtering and prediction, their structural and statistical properties, and algorithms for transformations between the various sets of parameters. Section 7.7 deals with efficient, order-recursive algorithms for the triangularization and inversion of Toeplitz matrices.

The chapter concludes with Section 7.8 which provides a concise introduction to the Kalman filtering algorithm. The Kalman filter provides a recursive solution to the minimum MSE filtering problem when the input stochastic process is described by a known state space model. This is possible because the state space model leads to a recursive formula for the updating of the required second-order moments.

7.1 FUNDAMENTALS OF ORDER-RECURSIVE ALGORITHMS

In Section 6.3, we introduced a method to solve the normal equations and compute the MMSE using the LDL^H decomposition. The optimum estimate is computed as a sum of products using a linear combiner supplied with the optimum coefficients and the input data. The key characteristic of this approach is that the order of the estimator should be fixed initially, and in case we choose a different order, we have to repeat *all* the computations. Such computational methods are known as *fixed-order algorithms*.

When the order of the estimator becomes a design variable, we need to modify our notation to take this into account. For example, the m th-order estimator $\mathbf{c}_m(n)$ is obtained by minimizing $E\{|e_m(n)|^2\}$, where

$$e_m(n) \triangleq y(n) - \hat{y}_m(n) \quad (7.1.1)$$

$$\hat{y}_m(n) \triangleq \mathbf{c}_m^H(n) \mathbf{x}_m(n) \quad (7.1.2)$$

$$\mathbf{c}_m(n) \triangleq [c_1^{(m)}(n) \ c_2^{(m)}(n) \ \cdots \ c_m^{(m)}(n)]^T \quad (7.1.3)$$

$$\mathbf{x}_m(n) \triangleq [x_1(n) \ x_2(n) \ \cdots \ x_m(n)]^T \quad (7.1.4)$$

In general, we use the subscript m to denote the order of a matrix or vector and the superscript m to emphasize that a scalar is a component of an $m \times 1$ vector. We note that these quantities are functions of time n , but sometimes we do not explicitly show this dependence for the sake of simplicity.

If the m th-order estimator $\mathbf{c}_m(n)$ has been computed by solving the normal equations, it seems to be a waste of computational power to start from scratch to compute the $(m + 1)$ st-order estimator $\mathbf{c}_{m+1}(n)$. Thus, we would like to arrange the computations so that the results for order m , that is, $\mathbf{c}_m(n)$ or $\hat{y}_m(n)$, can be used to compute the estimates for order $m + 1$, that is, $\mathbf{c}_{m+1}(n)$ or $\hat{y}_{m+1}(n)$. The resulting procedures are called *order-recursive algorithms* or *order-updating relations*. Similarly, procedures that compute $\mathbf{c}_m(n + 1)$ from $\mathbf{c}_m(n)$ or $\hat{y}_m(n + 1)$ from $\hat{y}_m(n)$ are called *time-recursive algorithms* or *time-updating relations*. Combined order and time updates are also possible. All these updates play a central role in the design and implementation of many optimum and adaptive filters.

In this section, we derive order-recursive algorithms for the computation of the LDL^H decomposition, the MMSE, and the MMSE optimal estimate. We also show that there is no order-recursive algorithm for the computation of the estimator parameters.

7.1.1 Matrix Partitioning and Optimum Nesting

We start by introducing some notation that is useful for the discussion of order-recursive algorithms.[†] Notice that if the order of the estimator increases from m to $m + 1$, then the input data vector is augmented with one additional observation x_{m+1} . We use the notation

[†] All quantities in Sections 7.1 and 7.2 are functions of the time index n . However, for notational simplicity we do not explicitly show this dependence.

$\mathbf{x}_{m+1}^{[m]}$ to denote the vector that consists of the first m components and $\mathbf{x}_{m+1}^{[m]}$ for the last m components of vector \mathbf{x}_{m+1} . The same notation can be generalized to matrices. The $m \times m$ matrix $\mathbf{R}_{m+1}^{[m]}$, obtained by the intersection of the first m rows and columns of \mathbf{R}_{m+1} , is known as the m th-order *leading principal submatrix* of \mathbf{R}_{m+1} . In other words, if r_{ij} are the elements of \mathbf{R}_{m+1} , then the elements of $\mathbf{R}_{m+1}^{[m]}$ are r_{ij} , $1 \leq i, j \leq m$. Similarly, $\mathbf{R}_{m+1}^{[m]}$ denotes the matrix obtained by the intersection of the last m rows and columns of \mathbf{R}_{m+1} . For example, if $m = 3$ we obtain

$$\mathbf{R}_4 = \begin{bmatrix} & & & \mathbf{R}_4^{[3]} \\ & r_{11} & r_{12} & r_{13} & r_{14} \\ & r_{21} & \boxed{r_{22}} & r_{23} & r_{24} \\ & r_{31} & r_{32} & \boxed{r_{33}} & r_{34} \\ & r_{41} & r_{42} & r_{43} & r_{44} \\ & & & \mathbf{R}_4^{[3]} & \end{bmatrix} \quad (7.1.5)$$

which illustrates the *upper left corner* and *lower right corner* partitionings of matrix \mathbf{R}_4 .

Since $\mathbf{x}_{m+1}^{[m]} = \mathbf{x}_m$, we can easily see that the correlation matrix can be partitioned as

$$\mathbf{R}_{m+1} = E \left\{ \begin{bmatrix} \mathbf{x}_m \\ \mathbf{x}_{m+1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_m^H & \mathbf{x}_{m+1}^* \end{bmatrix} \right\} = \begin{bmatrix} \mathbf{R}_m & \mathbf{r}_m^b \\ \mathbf{r}_m^{bH} & \rho_m^b \end{bmatrix} \quad (7.1.6)$$

where

$$\mathbf{r}_m^b \triangleq E\{\mathbf{x}_m \mathbf{x}_{m+1}^*\} \quad (7.1.7)$$

and

$$\rho_m^b \triangleq E\{|\mathbf{x}_{m+1}|^2\} \quad (7.1.8)$$

The result

$$\mathbf{x}_{m+1}^{[m]} = \mathbf{x}_m \Rightarrow \mathbf{R}_m = \mathbf{R}_{m+1}^{[m]} \quad (7.1.9)$$

is known as the *optimum nesting property* and is instrumental in the development of order-recursive algorithms. Similarly, we can show that $\mathbf{x}_{m+1}^{[m]} = \mathbf{x}_m$ implies

$$\mathbf{d}_{m+1} = E\{\mathbf{x}_{m+1} y^*\} = E \left\{ \begin{bmatrix} \mathbf{x}_m \\ \mathbf{x}_{m+1} \end{bmatrix} y^* \right\} = \begin{bmatrix} \mathbf{d}_m \\ d_{m+1} \end{bmatrix} \quad (7.1.10)$$

or

$$\mathbf{x}_{m+1}^{[m]} = \mathbf{x}_m \Rightarrow \mathbf{d}_m = \mathbf{d}_{m+1}^{[m]} \quad (7.1.11)$$

that is, the right-hand side of the normal equations also has the optimum nesting property.

Since (7.1.9) and (7.1.11) hold for all $1 \leq m \leq M$, the correlation matrix \mathbf{R}_M and the cross-correlation vector \mathbf{d}_M contain the information for the computation of all the optimum estimators \mathbf{c}_m for $1 \leq m \leq M$.

7.1.2 Inversion of Partitioned Hermitian Matrices

Suppose now that we know the inverse \mathbf{R}_m^{-1} of the leading principal submatrix $\mathbf{R}_{m+1}^{[m]} = \mathbf{R}_m$ of matrix \mathbf{R}_{m+1} and we wish to use it to compute \mathbf{R}_{m+1}^{-1} without having to repeat all the work. Since the inverse \mathbf{Q}_{m+1} of the Hermitian matrix \mathbf{R}_{m+1} is also Hermitian, it can be partitioned as

$$\mathbf{Q}_{m+1} = \begin{bmatrix} \mathbf{Q}_m & \mathbf{q}_m \\ \mathbf{q}_m^H & q_m \end{bmatrix} \quad (7.1.12)$$

Using (7.1.6), we obtain

$$\mathbf{R}_{m+1} \mathbf{Q}_{m+1} = \begin{bmatrix} \mathbf{R}_m & \mathbf{r}_m^b \\ \mathbf{r}_m^{bH} & \rho_m^b \end{bmatrix} \begin{bmatrix} \mathbf{Q}_m & \mathbf{q}_m \\ \mathbf{q}_m^H & q_m \end{bmatrix} = \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_m \\ \mathbf{0}_m^H & 1 \end{bmatrix} \quad (7.1.13)$$

After performing the matrix multiplication, we get

$$\mathbf{R}_m \mathbf{Q}_m + \mathbf{r}_m^b \mathbf{q}_m^H = \mathbf{I}_m \quad (7.1.14)$$

$$\mathbf{r}_m^{bH} \mathbf{Q}_m + \rho_m^b \mathbf{q}_m^H = \mathbf{0}_m^H \quad (7.1.15)$$

$$\mathbf{R}_m \mathbf{q}_m + \mathbf{r}_m^b q_m = \mathbf{0}_m \quad (7.1.16)$$

$$\mathbf{r}_m^{bH} \mathbf{q}_m + \rho_m^b q_m = 1 \quad (7.1.17)$$

where $\mathbf{0}_m$ is the $m \times 1$ zero vector. If matrix \mathbf{R}_m is invertible, we can solve (7.1.16) for \mathbf{q}_m

$$\mathbf{q}_m = -\mathbf{R}_m^{-1} \mathbf{r}_m^b q_m \quad (7.1.18)$$

and then substitute into (7.1.17) to obtain q_m as

$$q_m = \frac{1}{\rho_m^b - \mathbf{r}_m^{bH} \mathbf{R}_m^{-1} \mathbf{r}_m^b} \quad (7.1.19)$$

assuming that the scalar quantity $\rho_m^b - \mathbf{r}_m^{bH} \mathbf{R}_m^{-1} \mathbf{r}_m^b \neq 0$. Substituting (7.1.19) into (7.1.18), we obtain

$$\mathbf{q}_m = \frac{-\mathbf{R}_m^{-1} \mathbf{r}_m^b}{\rho_m^b - \mathbf{r}_m^{bH} \mathbf{R}_m^{-1} \mathbf{r}_m^b} \quad (7.1.20)$$

which, in conjunction with (7.1.14), yields

$$\mathbf{Q}_m = \mathbf{R}_m^{-1} - \mathbf{R}_m^{-1} \mathbf{r}_m^b \mathbf{q}_m^H = \mathbf{R}_m^{-1} + \frac{\mathbf{R}_m^{-1} \mathbf{r}_m^b (\mathbf{R}_m^{-1} \mathbf{r}_m^b)^H}{\rho_m^b - \mathbf{r}_m^{bH} \mathbf{R}_m^{-1} \mathbf{r}_m^b} \quad (7.1.21)$$

We note that (7.1.19) through (7.1.21) express the parts of the inverse matrix \mathbf{Q}_{m+1} in terms of known quantities. For our purposes, we express the above equations in a more convenient form, using the quantities

$$\mathbf{b}_m \triangleq [b_0^{(m)} \ b_1^{(m)} \ \cdots \ b_{m-1}^{(m)}]^T \triangleq -\mathbf{R}_m^{-1} \mathbf{r}_m^b \quad (7.1.22)$$

$$\text{and } \alpha_m^b \triangleq \rho_m^b - \mathbf{r}_m^{bH} \mathbf{R}_m^{-1} \mathbf{r}_m^b = \rho_m^b + \mathbf{r}_m^{bH} \mathbf{b}_m \quad (7.1.23)$$

Thus, if matrix \mathbf{R}_m is invertible and $\alpha_m^b \neq 0$, combining (7.1.13) with (7.1.19) through (7.1.23), we obtain

$$\mathbf{R}_{m+1}^{-1} = \begin{bmatrix} \mathbf{R}_m & \mathbf{r}_m^b \\ \mathbf{r}_m^{bH} & \rho_m^b \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{R}_m^{-1} & \mathbf{0}_m \\ \mathbf{0}_m^H & 0 \end{bmatrix} + \frac{1}{\alpha_m^b} \begin{bmatrix} \mathbf{b}_m \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{b}_m^H & 1 \end{bmatrix} \quad (7.1.24)$$

which determines \mathbf{R}_{m+1}^{-1} from \mathbf{R}_m^{-1} by using a simple rank-one modification known as the *matrix inversion by partitioning lemma* (Noble and Daniel 1988).

Another useful expression for α_m^b is

$$\alpha_m^b = \frac{\det \mathbf{R}_{m+1}}{\det \mathbf{R}_m} \quad (7.1.25)$$

which reinforces the importance of the quantity α_m^b for the invertibility of matrix \mathbf{R}_{m+1} (see Problem 7.1).

EXAMPLE 7.1.1. Given the matrix

$$\mathbf{R}_3 = \left[\begin{array}{cc|c} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \hline \frac{1}{3} & \frac{1}{2} & 1 \end{array} \right] = \begin{bmatrix} \mathbf{R}_2 & \mathbf{r}_2^b \\ \mathbf{r}_2^{bH} & \rho_2^b \end{bmatrix}$$

and the inverse matrix

$$\mathbf{R}_2^{-1} = \left[\begin{array}{cc} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{array} \right]^{-1} = \frac{1}{3} \begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix}$$

compute matrix \mathbf{R}_3^{-1} , using the matrix inversion by partitioning lemma.

Solution. To determine \mathbf{R}_3^{-1} from the order-updating formula (7.1.24), we first compute

$$\mathbf{b}_2 = -\mathbf{R}_2^{-1}\mathbf{r}_2^b = -\frac{1}{3} \begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} \frac{1}{3} \\ \frac{1}{2} \end{bmatrix} = -\frac{1}{9} \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

and

$$\alpha_2^b = \rho_2^b + \mathbf{r}_2^{bH}\mathbf{b}_2 = 1 - \frac{1}{9} \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \frac{20}{27}$$

using (7.1.22) and (7.1.23). Then we compute

$$\mathbf{R}_3^{-1} = \frac{1}{3} \begin{bmatrix} 4 & -2 & 0 \\ -2 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \frac{27}{20} \begin{bmatrix} -\frac{1}{9} \\ -\frac{4}{9} \\ 1 \end{bmatrix} \begin{bmatrix} -\frac{1}{9} & -\frac{4}{9} & 1 \end{bmatrix} = \frac{1}{20} \begin{bmatrix} 27 & -12 & -3 \\ -12 & 32 & -12 \\ -3 & -12 & 27 \end{bmatrix}$$

using (7.1.24). The reader can easily verify the above calculations using MATLAB.

Following a similar approach, we can show (see Problem 7.2) that the inverse of the lower right corner partitioned matrix \mathbf{R}_{m+1} can be expressed as

$$\mathbf{R}_{m+1}^{-1} \triangleq \begin{bmatrix} \rho_m^f & \mathbf{r}_m^{fH} \\ \mathbf{r}_m^f & \mathbf{R}_m^f \end{bmatrix}^{-1} = \begin{bmatrix} 0 & \mathbf{0}_m^H \\ \mathbf{0}_m & (\mathbf{R}_m^f)^{-1} \end{bmatrix} + \frac{1}{\alpha_m^f} \begin{bmatrix} 1 \\ \mathbf{a}_m^H \end{bmatrix} \begin{bmatrix} 1 & \mathbf{a}_m^H \end{bmatrix} \quad (7.1.26)$$

where

$$\mathbf{a}_m \triangleq [a_1^{(m)} \ a_2^{(m)} \ \cdots \ a_m^{(m)}]^T \triangleq -(\mathbf{R}_m^f)^{-1}\mathbf{r}_m^f \quad (7.1.27)$$

$$\alpha_m^f \triangleq \rho_m^f - \mathbf{r}_m^{fH}(\mathbf{R}_m^f)^{-1}\mathbf{r}_m^f = \rho_m^f + \mathbf{r}_m^{fH}\mathbf{a}_m = \frac{\det \mathbf{R}_{m+1}}{\det \mathbf{R}_m^f} \quad (7.1.28)$$

and the relationship (7.1.26) exists if matrix \mathbf{R}_m^f is invertible and $\alpha_m^f \neq 0$. A similar set of formulas can be obtained for arbitrary matrices (see Problem 7.3).

Interpretations. The vector \mathbf{b}_m , defined by (7.1.22), is the MMSE estimator of observation x_{m+1} from data vector \mathbf{x}_m . Indeed, if

$$e_m^b = x_{m+1} - \hat{x}_{m+1} = x_{m+1} + \mathbf{b}_m^H \mathbf{x}_m \quad (7.1.29)$$

we can show, using the orthogonality principle $E\{\mathbf{x}_m e_m^{b*}\} = \mathbf{0}$, that \mathbf{b}_m results in the MMSE given by

$$P_m^b = \rho_m^b + \mathbf{b}_m^H \mathbf{r}_m^b = \alpha_m^b \quad (7.1.30)$$

Similarly, we can show that \mathbf{a}_m , defined by (7.1.27), is the optimum estimator of x_1 based on $\tilde{\mathbf{x}}_m \triangleq [x_2 \ x_3 \ \cdots \ x_{m+1}]^T$. By using the orthogonality principle, $E\{\mathbf{x}_m e_m^{f*}\} = \mathbf{0}$, the MMSE is

$$P_m^f = \rho_m^f + \mathbf{r}_m^{fH} \mathbf{a}_m = \alpha_m^f \quad (7.1.31)$$

If $\mathbf{x}_{m+1} = [x(n) \ x(n-1) \ \cdots \ x(n-m)]^T$, then \mathbf{b}_m provides the *backward linear predictor* (BLP) and \mathbf{a}_m the *forward linear predictor* (FLP) of the process $x(n)$ from Section 6.5. For convenience, we always use this terminology even if, strictly speaking, the linear prediction interpretation is not applicable.

7.1.3 Levinson Recursion for the Optimum Estimator

We now illustrate how to use (7.1.24) to express the optimum estimator \mathbf{c}_{m+1} in terms of the estimator \mathbf{c}_m . Indeed, using (7.1.24), (7.1.10), and the normal equations $\mathbf{R}_m \mathbf{c}_m = \mathbf{d}_m$,

we have

$$\begin{aligned}\mathbf{c}_{m+1} &= \mathbf{R}_{m+1}^{-1} \mathbf{d}_{m+1} \\ &= \begin{bmatrix} \mathbf{R}_m^{-1} & \mathbf{0}_m \\ \mathbf{0}_m^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{d}_m \\ d_{m+1} \end{bmatrix} + \frac{1}{\alpha_m^b} \begin{bmatrix} \mathbf{b}_m \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{b}_m^H & 1 \end{bmatrix} \begin{bmatrix} \mathbf{d}_m \\ d_{m+1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_m^{-1} \mathbf{d}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m \\ 1 \end{bmatrix} \frac{\mathbf{b}_m^H \mathbf{d}_m + d_{m+1}}{\alpha_m^b}\end{aligned}$$

or more concisely

$$\mathbf{c}_{m+1} = \begin{bmatrix} \mathbf{c}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m \\ 1 \end{bmatrix} k_m^c \quad (7.1.32)$$

where the quantities

$$k_m^c \triangleq \frac{\beta_m^c}{\alpha_m^b} \quad (7.1.33)$$

and

$$\beta_m^c \triangleq \mathbf{b}_m^H \mathbf{d}_m + d_{m+1} \quad (7.1.34)$$

contain the “new information” d_{m+1} (the new component of \mathbf{d}_{m+1}). By using (7.1.22) and $\mathbf{R}_m \mathbf{c}_m = \mathbf{d}_m$, alternatively β_m^c can be written as

$$\beta_m^c = -\mathbf{r}_m^{bH} \mathbf{c}_m + d_{m+1} \quad (7.1.35)$$

We will use the term *Levinson recursion* for the order-updating relation (7.1.32) because a similar recursion was introduced as part of the celebrated algorithm due to Levinson (see Section 7.3). However, we stress that even though (7.1.32) is order-recursive, the parameter vector \mathbf{c}_{m+1} does not have the optimum nesting property, that is, $\mathbf{c}_{m+1}^{[m]} \neq \mathbf{c}_m$.

Clearly, if we know the vector \mathbf{b}_m , we can determine \mathbf{c}_{m+1} , using (7.1.32); however, its practical utility depends on how easily we can obtain the vector \mathbf{b}_m . In general, \mathbf{b}_m requires the solution of an $m \times m$ linear system of equations, and the computational savings compared to direct solution of the $(m + 1)$ st-order normal equations is insignificant. For the Levinson recursion to be useful, we need an order recursion for vector \mathbf{b}_m . Since matrix \mathbf{R}_{m+1} has the optimum nesting property, we need to check whether the same is true for the right-hand side vector in $\mathbf{R}_{m+1} \mathbf{b}_{m+1} = -\mathbf{r}_{m+1}^b$. From the definition $\mathbf{r}_m^b \triangleq E\{\mathbf{x}_m \mathbf{x}_{m+1}^*\}$, we can easily see that $\mathbf{r}_{m+1}^{b[m]} \neq \mathbf{r}_m^b$ and $\mathbf{r}_{m+1}^{b[m]} \neq \mathbf{r}_m^b$. Hence, in general, we cannot find a Levinson recursion for vector \mathbf{b}_m . This is possible only in optimum filtering problems in which the input data vector $\mathbf{x}_m(n)$ has a shift-invariance structure (see Section 7.3).

EXAMPLE 7.1.2. Use the Levinson recursion to determine the optimum linear estimator \mathbf{c}_3 specified by the matrix

$$\mathbf{R}_3 = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 1 \end{bmatrix}$$

in Example 7.1.1 and the cross-correlation vector

$$\mathbf{d}_3 = [1 \ 2 \ 4]^T$$

Solution. For $m = 1$ we have $r_{11} c_1^{(1)} = d_1$, which gives $c_1^{(1)} = 1$. Also, from (7.1.32) and (7.1.34) we obtain $k_0^c = c_1^{(1)} = 1$ and $\beta_0^c = d_1 = 1$. Finally, from $k_0^c = \beta_0^c / \alpha_0^b$, we get $\alpha_0^b = 1$.

To obtain \mathbf{c}_2 , we need $b_1^{(1)}$, k_1^c , β_1^c , and α_1 . We have

$$r_{11}b_1^{(1)} = -r_1^b \Rightarrow b_1^{(1)} = -\frac{1}{2} = -\frac{1}{2}$$

$$\beta_1^c = b_1^{(1)}d_1 + d_2 = -\frac{1}{2}(1) + 2 = \frac{3}{2}$$

$$\alpha_1^b = \rho_1^b + r_1^b b_1^{(1)} = 1 + \frac{1}{2}(-\frac{1}{2}) = \frac{3}{4}$$

$$k_1^c = \frac{\beta_1^c}{\alpha_1^b} = 2$$

and therefore

$$\mathbf{c}_2 = \begin{bmatrix} \mathbf{c}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 \\ 1 \end{bmatrix} k_1^c = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{2} \\ 1 \end{bmatrix} 2 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

To determine \mathbf{c}_3 , we need \mathbf{b}_2 , β_2^c , and α_2^b . To obtain \mathbf{b}_2 , we solve the linear system

$$\mathbf{R}_2 \mathbf{b}_2 = -\mathbf{r}_2^b \quad \text{or} \quad \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} b_1^{(2)} \\ b_2^{(2)} \end{bmatrix} = -\begin{bmatrix} \frac{1}{3} \\ \frac{1}{2} \end{bmatrix} \Rightarrow \mathbf{b}_2 = -\frac{1}{9} \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

and then compute

$$\beta_2^c = \mathbf{b}_2^T \mathbf{d}_2 + d_3 = -\frac{1}{9} [1 \quad 4] \begin{bmatrix} 1 \\ 2 \end{bmatrix} + 4 = 3$$

$$\alpha_2^b = \rho_2^b + \mathbf{r}_2^b \mathbf{b}_2 = 1 + \left[\frac{1}{3} \quad \frac{1}{2} \right] \begin{bmatrix} 1 \\ 4 \end{bmatrix} \left(-\frac{1}{9} \right) = \frac{20}{27}$$

$$k_2^c = \frac{\beta_2^c}{\alpha_2^b} = \frac{81}{20}$$

The desired solution \mathbf{c}_3 is obtained by using the Levinson recursion

$$\mathbf{c}_3 = \begin{bmatrix} \mathbf{c}_2 \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_2 \\ 1 \end{bmatrix} k_2^c \Rightarrow \mathbf{c}_3 = \begin{bmatrix} 0 \\ 2 \end{bmatrix} + \begin{bmatrix} -\frac{1}{9} \\ -\frac{4}{9} \end{bmatrix} \frac{81}{20} = \frac{1}{20} \begin{bmatrix} -9 \\ 4 \\ 81 \end{bmatrix}$$

which agrees with the solution obtained by solving $\mathbf{R}_3 \mathbf{c}_3 = \mathbf{d}_3$ using the function `c3=R3\d3`. We can also solve this linear system by developing an algorithm using the lower partitioning (7.1.26) as discussed in Problem 7.4.

Matrix inversion and the linear system solution for $m = 1$ are trivial (scalar division only). If \mathbf{R}_M is strictly positive definite, that is, $\mathbf{R}_m = \mathbf{R}_M^{[m]}$ is positive definite for all $1 \leq m \leq M$, the inverse matrices \mathbf{R}_m^{-1} and the solutions of $\mathbf{R}_m \mathbf{c}_m = \mathbf{d}_m$, $2 \leq m \leq M$, can be determined using (7.1.22) and the Levinson recursion (7.1.32) for $m = 1, 2, \dots, M-1$. However, in practice using the LDL^H provides a better method for performing these computations.

7.1.4 Order-Recursive Computation of the LDL^H Decomposition

We start by showing that the LDL^H decomposition can be computed in an order-recursive manner. The procedure is developed as part of a formal proof of the LDL^H decomposition using induction.

For $M = 1$, the matrix \mathbf{R}_1 is a positive number r_{11} and can be written uniquely in the form $r_{11} = 1 \cdot \xi_1 \cdot 1 > 0$. As we increment the order m , the $(m+1)$ st-order principal

submatrix of \mathbf{R}_m can be partitioned as in (7.1.6). By the induction hypothesis, there are unique matrices \mathbf{L}_m and \mathbf{D}_m such that

$$\mathbf{R}_m = \mathbf{L}_m \mathbf{D}_m \mathbf{L}_m^H \quad (7.1.36)$$

We next form the matrices

$$\mathbf{L}_{m+1} = \begin{bmatrix} \mathbf{L}_m & \mathbf{0} \\ \mathbf{l}_m^H & 1 \end{bmatrix} \quad \mathbf{D}_{m+1} = \begin{bmatrix} \mathbf{D}_m & \mathbf{0} \\ \mathbf{0}^H & \xi_{m+1} \end{bmatrix} \quad (7.1.37)$$

and try to determine the vector \mathbf{l}_m and the positive number ξ_{m+1} so that

$$\mathbf{R}_{m+1} = \mathbf{L}_{m+1} \mathbf{D}_{m+1} \mathbf{L}_{m+1}^H \quad (7.1.38)$$

Using (7.1.6) and (7.1.36) through (7.1.38), we see that

$$(\mathbf{L}_m \mathbf{D}_m) \mathbf{l}_m = \mathbf{r}_m^b \quad (7.1.39)$$

$$\rho_m^b = \mathbf{l}_m^H \mathbf{D}_m \mathbf{l}_m + \xi_{m+1}, \quad \xi_{m+1} > 0 \quad (7.1.40)$$

$$\text{Since } \det \mathbf{R}_m = \det \mathbf{L}_m \det \mathbf{D}_m \det \mathbf{L}_m^H = \xi_1 \xi_2 \cdots \xi_m > 0 \quad (7.1.41)$$

then $\det \mathbf{L}_m \mathbf{D}_m \neq 0$ and (7.1.39) has a unique solution \mathbf{l}_m . Finally, from (7.1.41) we obtain $\xi_{m+1} = \det \mathbf{R}_{m+1} / \det \mathbf{R}_m$, and therefore $\xi_{m+1} > 0$ because \mathbf{R}_{m+1} is positive definite. Hence, ξ_{m+1} is uniquely computed from (7.1.41), which completes the proof.

Because the triangular matrix \mathbf{L}_m is generated row by row using (7.1.39) and because the diagonal elements of matrix \mathbf{D}_m are computed sequentially using (7.1.40), both matrices have the optimum nesting property, that is, $\mathbf{L}_m = \mathbf{L}^{[m]}$, $\mathbf{D}_m = \mathbf{D}^{[m]}$. The optimum filter \mathbf{c}_m is then computed by solving

$$\mathbf{L}_m \mathbf{D}_m \mathbf{k}_m \triangleq \mathbf{d}_m \quad (7.1.42)$$

$$\mathbf{L}_m^H \mathbf{c}_m = \mathbf{k}_m \quad (7.1.43)$$

Using (7.1.42), we can easily see that \mathbf{k}_m has the optimum nesting property, that is, $\mathbf{k}_m = \mathbf{k}^{[m]}$ for $1 \leq m \leq M$. This is a consequence of the lower triangular form of \mathbf{L}_m . The computation of \mathbf{L}_m , \mathbf{D}_m , and \mathbf{k}_m can be done in a simple, order-recursive manner, which is all that is needed to compute \mathbf{c}_m for $1 \leq m \leq M$. However, the optimum estimator does not have the optimum nesting property, that is, $\mathbf{c}_{m+1}^{[m]} \neq \mathbf{c}_m$, because of the backward substitution involved in the solution of the upper triangular system (7.1.43) (see Example 6.3.1).

Using (7.1.42) and (7.1.43), we can write the MMSE for the m th-order linear estimator as

$$P_m = P_y - \mathbf{c}_m^H \mathbf{d}_m = P_y - \mathbf{k}_m^H \mathbf{D}_m \mathbf{k}_m \quad (7.1.44)$$

which, owing to the optimum nesting property of \mathbf{D}_m and \mathbf{k}_m , leads to

$$P_m = P_{m-1} - \xi_m |k_m|^2 \quad (7.1.45)$$

which is initialized with $P_0 = P_y$. Equation (7.1.45) provides an *order-recursive algorithm* for the computation of the MMSE.

7.1.5 Order-Recursive Computation of the Optimum Estimate

The computation of the optimum linear estimate $\hat{y}_m = \mathbf{c}_m^H \mathbf{x}_m$, using a linear combiner, requires m multiplications and $m - 1$ additions. Therefore, if we want to compute \hat{y}_m , for $1 \leq m \leq M$, we need M linear combiners and hence $M(M + 1)/2$ operations.

We next provide an alternative, more efficient order-recursive implementation that exploits the triangular decomposition of \mathbf{R}_{m+1} . We first notice that using (7.1.43), we obtain

$$\hat{y}_m = \mathbf{c}_m^H \mathbf{x}_m = (\mathbf{k}_m^H \mathbf{L}_m^{-1}) \mathbf{x}_m = \mathbf{k}_m^H (\mathbf{L}_m^{-1} \mathbf{x}_m) \quad (7.1.46)$$

Next, we define vector \mathbf{w}_m as

$$\mathbf{L}_m \mathbf{w}_m \triangleq \mathbf{x}_m \quad (7.1.47)$$

which can be found by using forward substitution in order to solve the triangular system. Therefore, we obtain

$$\hat{y}_m = \mathbf{k}_m^H \mathbf{w}_m = \sum_{i=1}^m k_i^* w_i \quad (7.1.48)$$

which provides the estimate \hat{y}_m in terms of \mathbf{k}_m and \mathbf{w}_m , that is, without using the estimator vector \mathbf{c}_m . Hence, if the ultimate goal is the computation of \hat{y}_m we do not need to compute the estimator \mathbf{c}_m .

For an order-recursive algorithm to be possible, the vector \mathbf{w}_m must have the optimum nesting property, that is, $\mathbf{w}_m = \mathbf{w}_{m+1}^{[m]}$. Indeed, using (7.1.37) and the matrix inversion by partitioning lemma for nonsymmetric matrices (see Problem 7.3), we obtain

$$\mathbf{L}_{m+1}^{-1} = \begin{bmatrix} \mathbf{L}_m & \mathbf{0} \\ \mathbf{l}_m^H & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{L}_m^{-1} & \mathbf{0} \\ \mathbf{v}_m^H & 1 \end{bmatrix}$$

where $\mathbf{v}_m = -\mathbf{L}_m^{-H} \mathbf{l}_m = -(\mathbf{L}_m^H)^{-1} \mathbf{D}_m^{-1} \mathbf{L}_m^{-1} \mathbf{r}_m^b = -\mathbf{R}_m^{-1} \mathbf{r}_m^b = \mathbf{b}_m$

due to (7.1.22). Therefore,

$$\mathbf{w}_{m+1} = \mathbf{L}_{m+1}^{-1} \mathbf{x}_{m+1} = \begin{bmatrix} \mathbf{L}_m^{-1} & \mathbf{0} \\ \mathbf{b}_m^H & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_m \\ x_{m+1} \end{bmatrix} = \begin{bmatrix} \mathbf{w}_m \\ w_{m+1} \end{bmatrix} \quad (7.1.49)$$

where

$$w_{m+1} = \mathbf{b}_m^H \mathbf{x}_m + x_{m+1} = e_m^b \quad (7.1.50)$$

from (7.1.29). In this case, we can derive order-recursive algorithms for the computation of \hat{y}_m and e_m , for all $1 \leq m \leq M$. Indeed, using (7.1.48) and (7.1.49), we obtain

$$\hat{y}_m = \hat{y}_{m-1} + k_m^* w_m \quad (7.1.51)$$

with $\hat{y}_0 = 0$. From (7.1.51) and $e_m = y - \hat{y}_m$, we have

$$e_m = e_{m-1} - k_m^* w_m \quad (7.1.52)$$

for $m = 1, 2, \dots, M$ with $e_0 = y$. The quantity w_m can be computed in an order-recursive manner by solving (7.1.47) using forward substitution. Indeed, from the m th row of (7.1.47) we obtain

$$w_m = x_m - \sum_{i=1}^{m-1} l_{i-1}^{(m-1)} w_i \quad (7.1.53)$$

which provides a recursive computation of w_m for $m = 1, 2, \dots, M$. To comply with the order-oriented notation, we use $l_{i-1}^{(m-1)}$ instead of $l_{m-1,i-1}$. Depending on the application, we use either (7.1.51) or (7.1.52).

For MMSE estimation, all the quantities are functions of the time index n , and therefore, the triangular decomposition of \mathbf{R}_m and the recursions (7.1.51) through (7.1.53) should be repeated for every new set of observations $y(n)$ and $\mathbf{x}(n)$.

EXAMPLE 7.1.3. A linear estimator is specified by the correlation matrix \mathbf{R}_4 and the cross-correlation vector \mathbf{d}_4 in Example 6.3.2. Compute the estimates \hat{y}_m , $1 \leq m \leq 4$, if the input data vector is given by $\mathbf{x}_4 = [1 \ 2 \ 1 \ -1]^T$.

Solution. Using the triangular factor \mathbf{L}_4 and the vector \mathbf{k}_4 found in Example 6.3.2 and (7.1.53), we find

$$\mathbf{w}_4 = [1 \ -1 \ 3 \ -8]^T$$

and $\hat{y}_1 = 1 \quad \hat{y}_2 = \frac{4}{3} \quad \hat{y}_3 = 6.6 \quad \hat{y}_4 = 14.6$

which the reader can verify by computing \mathbf{c}_m and $\hat{y}_m = \mathbf{c}_m^T \mathbf{x}_m$, $1 \leq m \leq 4$.

If we compute the matrix

$$\mathbf{B}_{m+1} \triangleq \mathbf{L}_{m+1}^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ b_0^{(1)} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b_0^{(m)} & b_1^{(m)} & \cdots & 1 \end{bmatrix} \quad (7.1.54)$$

then (7.1.49) can be written as

$$\mathbf{w}_{m+1} = \mathbf{e}_{m+1}^b = \mathbf{B}_{m+1} \mathbf{x}_{m+1} \quad (7.1.55)$$

where

$$\mathbf{e}_{m+1}^b \triangleq [e_0^b \ e_1^b \ \cdots \ e_m^b]^T \quad (7.1.56)$$

is the BLP error vector. From (7.1.22), we can easily see that the rows of \mathbf{B}_{m+1} are formed by the optimum estimators \mathbf{b}_m of x_{m+1} from \mathbf{x}_m . Note that the elements of matrix \mathbf{B}_{m+1} are denoted by using the order-oriented notation $b_i^{(m)}$ introduced in Section 7.1 rather than the conventional b_{mi} matrix notation. Equation (7.1.55) provides an alternative computation of \mathbf{w}_{m+1} as a matrix-vector multiplication. Each component of \mathbf{w}_{m+1} can be computed independently, and hence in parallel, by the formula

$$w_j = x_j + \sum_{i=1}^{j-1} b_{i-1}^{(j-1)*} x_i \quad 1 \leq j \leq m \quad (7.1.57)$$

which, in contrast to (7.1.53), is nonrecursive. Using (7.1.57) and (7.1.51), we can derive the order-recursive MMSE estimator implementation shown in Figure 7.1.

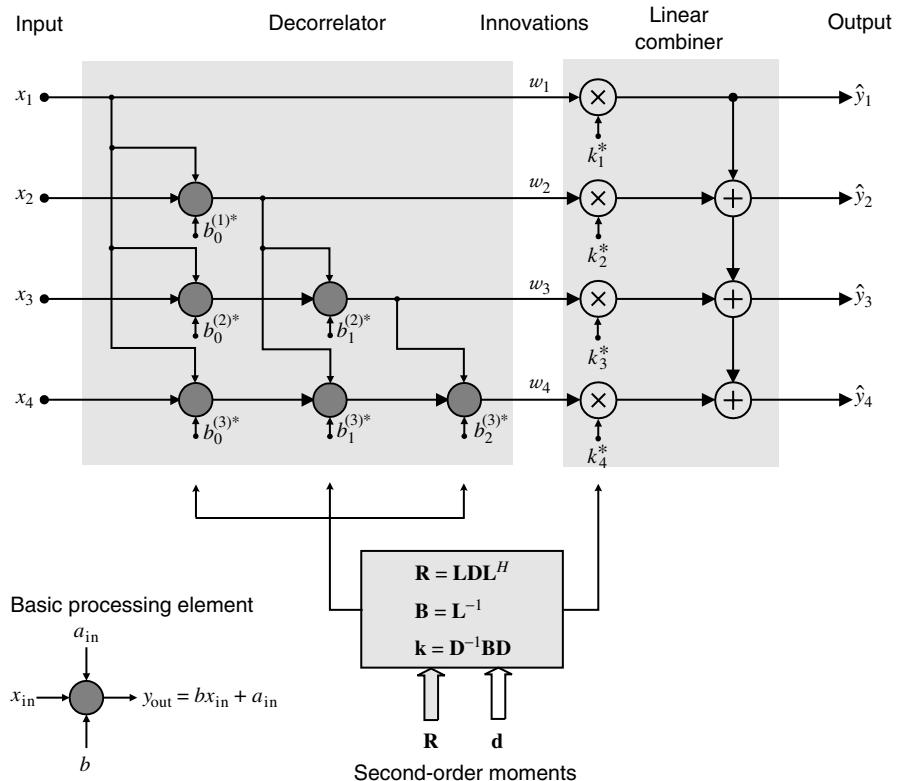


FIGURE 7.1

Orthogonal order-recursive structure for linear MMSE estimation.

Finally, we notice that matrix \mathbf{B}_m provides the UDU^H decomposition of the inverse correlation matrix \mathbf{R}_m . Indeed, from (7.1.36) we obtain

$$\mathbf{R}_m^{-1} = (\mathbf{L}_m^H)^{-1} \mathbf{D}_m^{-1} \mathbf{L}_m^{-1} = \mathbf{B}_m^H \mathbf{D}_m^{-1} \mathbf{B}_m \quad (7.1.58)$$

because inversion and transposition are interchangeable and the UDU^H decomposition is unique. This formula provides a practical method to compute the inverse of the correlation matrix by using the LDL^H decomposition because computing the inverse of a triangular matrix is simple (see Problem 7.5).

7.2 INTERPRETATIONS OF ALGORITHMIC QUANTITIES

We next show that various intermediate quantities that appear in the linear MMSE estimation algorithms have physical and statistical interpretations that, besides their intellectual value, facilitate better understanding of the operation, performance, and numerical properties of the algorithms.

7.2.1 Innovations and Backward Prediction

The correlation matrix of \mathbf{w}_m is

$$E\{\mathbf{w}_m \mathbf{w}_m^H\} = \mathbf{L}_m^{-1} E\{\mathbf{x}_m \mathbf{x}_m^H\} \mathbf{L}_m^{-H} = \mathbf{D}_m \quad (7.2.1)$$

where we have used (7.1.47) and the triangular decomposition (7.1.36). Therefore, the components of \mathbf{w}_m are uncorrelated, random variables with variances

$$\xi_i = E\{|w_i|^2\} \quad (7.2.2)$$

since $\xi_i \geq 0$. Furthermore, the two sets of random variables $\{w_1, w_2, \dots, w_M\}$ and $\{x_1, x_2, \dots, x_M\}$ are *linearly equivalent* because they can be obtained from each other through the linear transformation (7.1.47). This transformation removes all the redundant correlation among the components of \mathbf{x} and is known as a *decorrelation* or *whitening* operation (see Section 3.5.2). Because the random variables w_i are uncorrelated, each of them adds “new information” or innovation. In this sense, $\{w_1, w_2, \dots, w_m\}$ is the *innovations representation* of the random variables $\{x_1, x_2, \dots, x_m\}$. Because $\mathbf{x}_m = \mathbf{L}_m \mathbf{w}_m$, the random vector $\mathbf{w}_m = \mathbf{e}_m^b$ is the *innovations representation*, and \mathbf{x}_m and \mathbf{w}_m are *linearly equivalent* as well, (see Section 3.5).

The cross-correlation matrix between \mathbf{x}_m and \mathbf{w}_m is

$$E\{\mathbf{x}_m \mathbf{w}_m^H\} = E\{\mathbf{L}_m \mathbf{w}_m \mathbf{w}_m^H\} = \mathbf{L}_m \mathbf{D}_m \quad (7.2.3)$$

which shows that, owing to the lower triangular form of \mathbf{L}_m , $E\{x_i w_j^*\} = 0$ for $j > i$. We will see in Section 7.6 that these factors are related to the gapped functions and the algorithm of Schür.

Furthermore, since $e_m^b = w_{m+1}$, from (7.1.50) we have

$$P_m^b = \xi_{m+1} = E\{|w_{m+1}|^2\}$$

which also can be shown algebraically by using (7.1.41), (7.1.40), and (7.1.30). Indeed, we have

$$\xi_{m+1} = \frac{\det \mathbf{R}_{m+1}}{\det \mathbf{R}_m} = \rho_m^b - \mathbf{l}_m^H \mathbf{D}_m \mathbf{l}_m = \rho_m^b - \mathbf{r}_m^{bH} \mathbf{R}_m^{-1} \mathbf{r}_m^b = P_m^b \quad (7.2.4)$$

and, therefore,

$$\mathbf{D}_m = \text{diag}\{P_0^b, P_1^b, \dots, P_{m-1}^b\} \quad (7.2.5)$$

7.2.2 Partial Correlation

In general, the random variables $y, x_1, \dots, x_m, x_{m+1}$ are correlated. The correlation between y and x_{m+1} , after the influence from the components of the vector \mathbf{x}_m has been removed, is known as *partial correlation*. To remove the correlation due to \mathbf{x}_m , we extract from y and x_{m+1} the components that can be predicted from \mathbf{x}_m . The remaining correlation is from the estimation errors e_m and e_m^b , which are both uncorrelated with \mathbf{x}_m because of the orthogonality principle. Therefore, the partial correlation of y and x_{m+1} is

$$\begin{aligned} \text{PARCOR}(y; x_{m+1}) &\triangleq E\{e_m e_m^{b*}\} = E\{(y - \mathbf{c}_m^H \mathbf{x}_m) e_m^{b*}\} \\ &= E\{ye_m^{b*}\} = E\{y(x_{m+1}^* + \mathbf{x}_m^H \mathbf{b}_m)\} \\ &= E\{yx_{m+1}^*\} + E\{y\mathbf{x}_m^H\}\mathbf{b}_m \\ &= d_{m+1}^* + \mathbf{d}_m^H \mathbf{b}_m \triangleq \beta_m^{c*} \end{aligned} \quad (7.2.6)$$

where we have used the orthogonality principle $E\{\mathbf{x}_m e_m^{b*}\} = \mathbf{0}$ and (7.1.10), (7.1.50), and (7.1.34).

The partial correlation $\text{PARCOR}(y; x_{m+1})$ is also related to the parameters k_m obtained from the LDL^H decomposition. Indeed, from (7.1.42) and (7.1.54), we obtain the relation

$$\mathbf{k}_{m+1} = \mathbf{D}_{m+1}^{-1} \mathbf{B}_{m+1} \mathbf{d}_{m+1} \quad (7.2.7)$$

whose last row is

$$k_{m+1} = \frac{\mathbf{b}_m^H \mathbf{d}_m + d_{m+1}}{\xi_{m+1}} = \frac{\beta_m^c}{P_m^b} = k_m^c \quad (7.2.8)$$

owing to (7.2.4) and (7.2.6).

EXAMPLE 7.2.1. The LDL^H decomposition of matrix \mathbf{R}_3 in Example 7.1.2 is given by

$$\mathbf{L} = \left[\begin{array}{cc|c} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \hline \frac{1}{3} & \frac{4}{9} & 1 \end{array} \right] \quad \mathbf{D} = \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & \frac{3}{4} & 0 \\ 0 & 0 & \frac{20}{27} \end{array} \right]$$

and can be found by using the function $[L, D] = \text{ldlt}(R)$. Comparison with the results obtained in Example 7.1.2 shows that the rows of the matrix

$$\mathbf{L}^{-1} = \left[\begin{array}{ccc} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ -\frac{1}{9} & -\frac{4}{9} & 1 \end{array} \right]$$

provide the elements of the backward predictors, whereas the diagonal elements of \mathbf{D} are equal to the scalars α_m . Using (7.2.7), we obtain $\mathbf{k} = [1 \ 2 \ \frac{81}{20}]^T$ whose elements are the quantities k_0^c, k_1^c , and k_2^c computed in Example 7.1.2 using the Levinson recursion.

7.2.3 Order Decomposition of the Optimum Estimate

The equation $\hat{y}_{m+1} = \hat{y}_m + k_{m+1}^* w_{m+1}$, with $k_{m+1} = \beta_m^c / P_m^b = k_m^c$, shows that the improvement in the estimate when we include one more observation x_{m+1} , that is, when we increase the order by 1, is proportional to the innovation w_{m+1} contained in x_{m+1} . The innovation is the part of x_{m+1} that cannot be linearly estimated from the already used data \mathbf{x}_m . The term w_{m+1} is scaled by the ratio of the partial correlation between y and the “new” observation x_{m+1} and the power of the innovation P_m^b .

Thus, the computation of the $(m+1)$ st-order estimate of y based on $\mathbf{x}_{m+1} = [\mathbf{x}_m^T \ x_{m+1}]$ can be reduced to two m th-order estimation problems: the estimation of y based on \mathbf{x}_m and the estimation of the new observation x_{m+1} based on \mathbf{x}_m . This decomposition of linear estimation problems into smaller ones has very important applications to the development of efficient algorithms and structures for MMSE estimation.

We use the term *direct* for the implementation of the MMSE linear combiner as a sum of products, involving the optimum parameters $c_i^{(m)}$, $1 \leq i \leq m$, to emphasize the direct use of these coefficients. Because the random variables w_i used in the implementation of Figure 7.1 are orthogonal, that is, $\langle w_i, w_j \rangle = 0$ for $i \neq j$, we refer to this implementation as the *orthogonal implementation* or the *orthogonal structure*. These two structures appear in every type of linear MMSE estimation problem, and their particular form depends on the specifics of the problem and the associated second-order moments. In this sense, they play a prominent role in linear MMSE estimation in general, and in this book in particular.

We conclude our discussion with the following important observations:

1. The direct implementation combines correlated, that is, redundant information, and it is *not* order-recursive because increasing the order of the estimator *destroys* the optimality of the existing coefficients. Again, the reason is that the direct-form optimum filter coefficients do not possess the optimal nesting property.
2. The orthogonal implementation consists of a decorrelator and a linear combiner. The estimator combines the innovations of the data (nonredundant information) and is order-recursive because it does not use the optimum coefficient vector. Hence, increasing the order of the estimator preserves the optimality of the existing lower-order part. The resulting structure is modular such that each additional term improves the estimate by an amount proportional to the included innovation w_m .
3. Using the vector interpretation of random variables, the transformation $\tilde{\mathbf{x}}_m = \mathbf{F}_m \mathbf{x}_m$ is just a change of basis. The choice $\mathbf{F}_m = \mathbf{L}_m^{-1}$ converts from the *oblique* set $\{x_1, x_2, \dots, x_m\}$ to the *orthogonal* basis $\{w_1, w_2, \dots, w_m\}$. The advantage of working with orthogonal bases is that adding new components does not affect the optimality of previous ones.
4. The LDL^H decomposition for random vectors is the matrix equivalent of the spectral factorization theorem for discrete-time, stationary, stochastic processes. Both approaches facilitate the design and implementation of optimum FIR and IIR filters (see Sections 6.4 and 6.6).

7.2.4 Gram-Schmidt Orthogonalization

We next combine the geometric interpretation of the random variables with the Gram-Schmidt procedure used in linear algebra. The Gram-Schmidt procedure produces the innovations $\{w_1, w_2, \dots, w_m\}$ by orthogonalizing the original set $\{x_1, x_2, \dots, x_m\}$.

We start by choosing w_1 to be in the direction of x_1 , that is,

$$w_1 = x_1$$

The next “vector” w_2 should be orthogonal to w_1 . To determine w_2 , we subtract from x_2 its component along w_1 [see Figure 7.2(a)], that is,

$$w_2 = x_2 - l_0^{(1)} w_1$$

where $l_0^{(1)}$ is obtained from the condition $w_2 \perp w_1$ as follows:

$$\langle w_2, w_1 \rangle = \langle x_2, w_1 \rangle - l_0^{(1)} \langle w_1, w_1 \rangle = 0$$

or

$$l_0^{(1)} = \frac{\langle x_2, w_1 \rangle}{\langle w_1, w_1 \rangle}$$

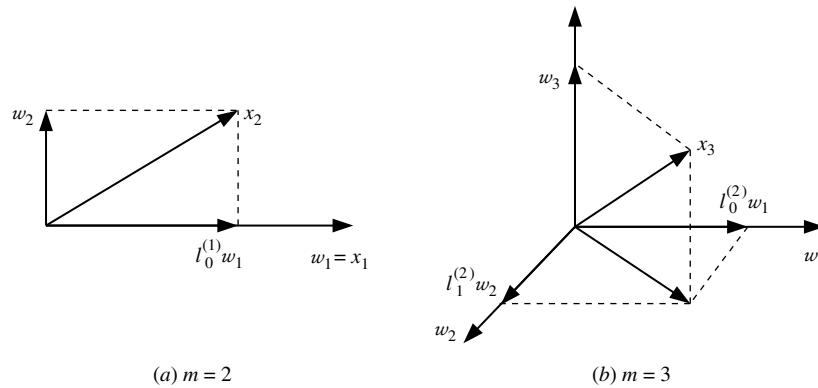
**FIGURE 7.2**

Illustration of the Gram-Schmidt orthogonalization process.

Similarly, to determine w_3 , we subtract from x_3 its components along w_1 and w_2 , that is,

$$w_3 = x_3 - l_0^{(2)}w_1 - l_1^{(2)}w_2$$

as illustrated in Figure 7.2(b). Using the conditions $w_3 \perp w_1$ and $w_3 \perp w_2$, we can easily see that

$$l_0^{(2)} = \frac{\langle x_3, w_1 \rangle}{\langle w_1, w_1 \rangle} \quad l_1^{(2)} = \frac{\langle x_3, w_2 \rangle}{\langle w_2, w_2 \rangle}$$

This approach leads to the following *classical Gram-Schmidt algorithm*:

- Define $w_1 = x_1$.
- For $2 \leq m \leq M$, compute

$$w_m = x_m - l_0^{(m-1)}w_1 - \dots - l_{m-2}^{(m-1)}w_{m-1} \quad (7.2.9)$$

$$\text{where } l_i^{(m-1)} = \frac{\langle x_{m-1}, w_i \rangle}{\langle w_i, w_i \rangle} \quad (7.2.10)$$

assuming that $\langle w_i, w_i \rangle \neq 0$.

From the derivation of the algorithm it should be clear that the sets $\{x_1, \dots, x_m\}$ and $\{w_1, \dots, w_m\}$ are linearly equivalent for $m = 1, 2, \dots, M$. Using (7.2.11), we obtain

$$\mathbf{x}_m = \mathbf{L}_m \mathbf{w}_m \quad (7.2.11)$$

$$\text{where } \mathbf{L}_m \triangleq \begin{bmatrix} 1 & 0 & \dots & 0 \\ l_0^{(1)} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_0^{(m-1)} & l_1^{(m-1)} & \dots & 1 \end{bmatrix} \quad (7.2.12)$$

is a unit lower triangular matrix. Since, by construction, the components of \mathbf{w}_m are uncorrelated, its correlation matrix \mathbf{D}_m is diagonal with elements $\xi_i = E\{|w_i|^2\}$. Using (7.2.11), we obtain

$$\mathbf{R}_m = E\{\mathbf{x}_m \mathbf{x}_m^H\} = \mathbf{L}_m E\{\mathbf{w}_m \mathbf{w}_m^H\} \mathbf{L}_m^H = \mathbf{L}_m \mathbf{D}_m \mathbf{L}_m^H \quad (7.2.13)$$

which is precisely the unique LDL^H decomposition of the correlation matrix \mathbf{R}_m . Therefore, the *Gram-Schmidt orthogonalization of the data vector \mathbf{x}_m provides an alternative approach to obtain the LDL^H decomposition of its correlation matrix $\mathbf{R}_m = E\{\mathbf{x}_m \mathbf{x}_m^H\}$* .

The key difference between a linear combiner and an FIR filter is the nature of the input data vector. The input data vector for FIR filters consists of *consecutive samples* from the *same* discrete-time stochastic process, that is,

$$\mathbf{x}_m(n) = [x(n) \ x(n-1) \ \cdots \ x(n-m+1)]^T \quad (7.3.1)$$

instead of samples from m different processes $x_i(n)$. This *shift invariance* of the input data vector allows for the development of simpler, order-recursive algorithms and structures for optimum FIR filtering and prediction compared to those for general linear estimation. Furthermore, the quest for order-recursive algorithms leads to a natural, elegant, and unavoidable interconnection between optimum filtering and the BLP and FLP problems.

We start with the following upper and lower partitioning of the input data vector

$$\mathbf{x}_{m+1}(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-m+1) \\ x(n-m) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_m(n) \\ x(n-m) \end{bmatrix} = \begin{bmatrix} x(n) \\ \mathbf{x}_m(n-1) \end{bmatrix} \quad (7.3.2)$$

which shows that $\mathbf{x}_{m+1}^{[m]}(n)$ and $\mathbf{x}_{m+1}^{[m]}(n)$ are simply shifted versions (by one sample delay) of the same vector $\mathbf{x}_m(n)$. The shift invariance of $\mathbf{x}_{m+1}(n)$ results in an analogous shift invariance for the correlation matrix $\mathbf{R}_{m+1}(n) = E\{\mathbf{x}_{m+1}(n)\mathbf{x}_{m+1}^H(n)\}$. Indeed, we can easily show that the upper-lower partitioning of the correlation matrix is

$$\mathbf{R}_{m+1}(n) = \begin{bmatrix} \mathbf{R}_m(n) & \mathbf{r}_m^b(n) \\ \mathbf{r}_m^{bH}(n) & P_x(n-m) \end{bmatrix} \quad (7.3.3)$$

and the lower-upper partitioning is

$$\mathbf{R}_{m+1}(n) = \begin{bmatrix} P_x(n) & \mathbf{r}_m^{fH}(n) \\ \mathbf{r}_m^f(n) & \mathbf{R}_m(n-1) \end{bmatrix} \quad (7.3.4)$$

where

$$\mathbf{r}_m^b(n) = E\{\mathbf{x}_m(n)x^*(n-m)\} \quad (7.3.5)$$

$$\mathbf{r}_m^f(n) = E\{\mathbf{x}_m(n-1)x^*(n)\} \quad (7.3.6)$$

$$P_x(n) = E\{|x(n)|^2\} \quad (7.3.7)$$

We note that, in contrast to the general case (7.1.5) where the matrix $\mathbf{R}_m^f(n) = \mathbf{R}_{m+1}^{[m]}(n)$ is *unrelated* to $\mathbf{R}_m(n)$, here the matrix $\mathbf{R}_{m+1}^{[m]}(n) = \mathbf{R}_m(n-1)$. This is a by-product of the shift-invariance property of the input data vector and takes the development of order-recursive algorithms one step further. We begin our pursuit of an order-recursive algorithm with the development of a Levinson order recursion for the optimum FIR filter coefficients.

7.3.1 Order-Recursive Computation of the Optimum Filter

Suppose that at time n we have already computed the optimum FIR filter $\mathbf{c}_m(n)$ specified by

$$\mathbf{c}_m(n) = \mathbf{R}_m^{-1}(n)\mathbf{d}_m(n) \quad (7.3.8)$$

and the MMSE is

$$P_m^c(n) = P_y(n) - \mathbf{d}_m^H(n)\mathbf{c}_m(n) \quad (7.3.9)$$

where

$$\mathbf{d}_m(n) = E\{\mathbf{x}_m(n)y^*(n)\} \quad (7.3.10)$$

We wish to compute the optimum filter

$$\mathbf{c}_{m+1}(n) = \mathbf{R}_{m+1}^{-1}(n)\mathbf{c}_{m+1}(n)$$

by modifying $\mathbf{c}_m(n)$ using an order-recursive algorithm. From (7.3.3), we see that matrix $\mathbf{R}_{m+1}(n)$ has the optimum nesting property. Using the upper partitioning in (7.3.2), we obtain

$$\mathbf{d}_{m+1}(n) = E \left\{ \begin{bmatrix} \mathbf{x}_m(n) \\ x(n-m) \end{bmatrix} y^*(n) \right\} = \begin{bmatrix} \mathbf{d}_m(n) \\ d_{m+1}(n) \end{bmatrix} \quad (7.3.11)$$

which shows that $\mathbf{d}_{m+1}(n)$ also has the optimum nesting property. Therefore, we can develop a Levinson order recursion using the upper left matrix inversion by partitioning lemma

$$\mathbf{R}_{m+1}^{-1}(n) = \begin{bmatrix} \mathbf{R}_m^{-1}(n) & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \frac{1}{P_m^b(n)} \begin{bmatrix} \mathbf{b}_m(n) \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{b}_m^H(n) & 1 \end{bmatrix} \quad (7.3.12)$$

where

$$\mathbf{b}_m(n) = -\mathbf{R}_m^{-1}(n)\mathbf{r}_m^b(n) \quad (7.3.13)$$

is the optimum BLP, and

$$P_m^b(n) = \frac{\det \mathbf{R}_{m+1}(n)}{\det \mathbf{R}_m(n)} = P_x(n-m) + \mathbf{r}_m^{bH}(n)\mathbf{b}_m(n) \quad (7.3.14)$$

is the corresponding MMSE. Equations (7.3.12) through (7.3.14) follow easily from (7.1.22), (7.1.23), and (7.1.24). It is interesting to note that $\mathbf{b}_m(n)$ is the optimum estimator for the additional observation $x(n-m)$ used by the optimum filter $\mathbf{c}_{m+1}(n)$. Substituting (7.3.11) and (7.3.12) into (7.3.8), we obtain

$$\mathbf{c}_{m+1}(n) = \begin{bmatrix} \mathbf{c}_m(n) \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m(n) \\ 1 \end{bmatrix} k_m^c(n) \quad (7.3.15)$$

where

$$k_m^c(n) \triangleq \frac{\beta_m^c(n)}{P_m^b(n)} \quad (7.3.16)$$

and

$$\beta_m^c(n) \triangleq \mathbf{b}_m^H(n)\mathbf{d}_m(n) + d_{m+1}(n) \quad (7.3.17)$$

Thus, if we know the BLP $\mathbf{b}_m(n)$, we can determine $\mathbf{c}_{m+1}(n)$ by using the Levinson recursion in (7.3.15).

Levinson recursion for the backward predictor. For the order recursion in (7.3.15) to be useful, we need an order recursion for the BLP $\mathbf{b}_m(n)$. This is possible if the linear systems

$$\begin{aligned} \mathbf{R}_m(n)\mathbf{b}_m(n) &= -\mathbf{r}_m^b(n) \\ \mathbf{R}_{m+1}(n)\mathbf{b}_{m+1}(n) &= -\mathbf{r}_{m+1}^b(n) \end{aligned} \quad (7.3.18)$$

are nested. Since the matrices are nested [see (7.3.3)], we check whether the right-hand side vectors are nested. We can easily see that no optimum nesting is possible if we use the upper partitioning in (7.3.2). However, if we use the lower-upper partitioning, we obtain

$$\mathbf{r}_{m+1}^b(n) = E \left\{ \begin{bmatrix} x(n) \\ \mathbf{x}_m(n-1) \end{bmatrix} x^*(n-m-1) \right\} \triangleq \begin{bmatrix} r_{m+1}^b(n) \\ \mathbf{r}_m^b(n-1) \end{bmatrix} \quad (7.3.19)$$

which provides a partitioning that includes the wanted vector $\mathbf{r}_m^b(n)$ delayed by one sample as a result of the shift invariance of $\mathbf{x}_m(n)$. To explore this partitioning, we use the lower-upper corner matrix inversion by partitioning lemma

$$\mathbf{R}_{m+1}^{-1}(n) = \begin{bmatrix} 0 & \mathbf{0}^H \\ \mathbf{0} & \mathbf{R}_m^{-1}(n-1) \end{bmatrix} + \frac{1}{P_f(n)} \begin{bmatrix} 1 \\ \mathbf{a}_m(n) \end{bmatrix} \begin{bmatrix} 1 & \mathbf{a}_m^H(n) \end{bmatrix} \quad (7.3.20)$$

where

$$\mathbf{a}_m(n) \triangleq -\mathbf{R}_m^{-1}(n-1)\mathbf{r}_m^f(n) \quad (7.3.21)$$

is the optimum FLP and

$$P_m^f(n) = \frac{\det \mathbf{R}_{m+1}(n)}{\det \mathbf{R}_m(n-1)} = P_x(n) + \mathbf{r}_m^{fH}(n)\mathbf{a}_m(n) \quad (7.3.22)$$

is the forward linear prediction MMSE. Equations (7.3.20) through (7.3.22) follow easily from (7.1.26) through (7.1.28). Substituting (7.3.20) and (7.3.19) into

$$\mathbf{b}_{m+1}(n) = -\mathbf{R}_{m+1}^{-1}(n)\mathbf{r}_{m+1}^b(n)$$

we obtain the recursion

$$\mathbf{b}_{m+1}(n) = \begin{bmatrix} 0 \\ \mathbf{b}_m(n-1) \end{bmatrix} + \begin{bmatrix} 1 \\ \mathbf{a}_m(n) \end{bmatrix} k_m^b(n) \quad (7.3.23)$$

where

$$k_m^b(n) \triangleq -\frac{\beta_m^b(n)}{P_m^f(n)} \quad (7.3.24)$$

and

$$\beta_m^b(n) \triangleq \mathbf{r}_{m+1}^b(n) + \mathbf{a}_m^H(n)\mathbf{r}_m^b(n-1) \quad (7.3.25)$$

To proceed with the development of the order-recursive algorithm, we clearly need an order recursion for the optimum FLP $\mathbf{a}_m(n)$.

Levinson recursion for the forward predictor. Following a similar procedure for the Levinson recursion of the BLP, we can derive the Levinson recursion for the FLP. If we use the upper-lower partitioning in (7.3.2), we obtain

$$\mathbf{r}_{m+1}^f(n) = E\{\mathbf{x}_{m+1}(n-1)x^*(n)\} = \begin{bmatrix} \mathbf{r}_m^f(n) \\ r_{m+1}^f(n) \end{bmatrix} \quad (7.3.26)$$

which in conjunction with (7.3.12) and (7.3.21) leads to the following order recursion

$$\mathbf{a}_{m+1}(n) = \begin{bmatrix} \mathbf{a}_m(n) \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m(n-1) \\ 1 \end{bmatrix} k_m^f(n) \quad (7.3.27)$$

where

$$k_m^f(n) \triangleq -\frac{\beta_m^f(n)}{P_m^b(n-1)} \quad (7.3.28)$$

and

$$\beta_m^f(n) \triangleq \mathbf{b}_m^H(n-1)\mathbf{r}_m^f(n) + r_{m+1}^f(n) \quad (7.3.29)$$

Is an order-recursive algorithm feasible? For $m = 1$, we have a scalar equation $r_{11}(n)c_1^{(1)}(n) = d_1(n)$ whose solution is $c_1^{(1)}(n) = d_1(n)/r_{11}(n)$. Using the Levinson order recursions for $m = 1, 2, \dots, M-1$, we can find $\mathbf{c}_M(n)$ if the quantities $\mathbf{b}_m(n-1)$ and $P_m^b(n-1)$, $1 \leq m < M$, required by (7.3.27) and (7.3.28) are known. The lack of this information prevents the development of a complete order-recursive algorithm for the solution of the normal equations for optimum FIR filtering or prediction. The need for time updates arises because each order update requires both the upper left corner and the lower right corner partitionings

$$\mathbf{R}_{m+1}(n) = \begin{bmatrix} \mathbf{R}_m(n) & \times \\ \times & \times \end{bmatrix} = \begin{bmatrix} \times & \times \\ \times & \mathbf{R}_m(n-1) \end{bmatrix}$$

of matrix \mathbf{R}_{m+1} . The presence of $\mathbf{R}_m(n-1)$, which is a result of the nonstationarity of the input signal, creates the need for a time updating of $\mathbf{b}_m(n)$. This is possible only for certain types of nonstationarity that can be described by simple relations between $\mathbf{R}_m(n)$ and $\mathbf{R}_m(n-1)$. The simplest case occurs for stationary processes where $\mathbf{R}_m(n) = \mathbf{R}_m(n-1) = \mathbf{R}_m$. Another very useful case occurs for nonstationary processes generated by linear state-space models, which results in the Kalman filtering algorithm (see Section 7.8).

Partial correlation interpretation. The partial correlation between $y(n)$ and $x(n-m)$, after the influence of the intermediate samples $x(n), x(n-1), \dots, x(n-m+1)$ has been removed, is

$$E\{e_m^b(n)e_m^*(n)\} = \mathbf{b}_m^H(n)\mathbf{d}_m(n) + d_{m+1}(n) = \beta_m^c(n) \quad (7.3.30)$$

which is obtained by working as in the derivation of (7.2.6). It can be shown, following a procedure similar to that leading to (7.2.8), that the $k_m(n)$ parameters in the Levinson recursions can be obtained from

$$\begin{aligned} \mathbf{R}_m(n) &= \mathbf{L}_m(n)\mathbf{D}_m(n)\mathbf{L}_m^H(n) \\ \mathbf{L}_m(n)\mathbf{D}_m(n)\mathbf{k}_m^c(n) &= \mathbf{d}_m(n) \\ \mathbf{L}_m(n)\mathbf{D}_m(n)\mathbf{k}_m^f(n) &= \mathbf{r}_m^b(n) \\ \mathbf{L}_m(n-1)\mathbf{D}_m(n-1)\mathbf{k}_m^b(n) &= \mathbf{r}_m^f(n) \end{aligned} \quad (7.3.31)$$

that is, as a by-product of the LDL^H decomposition.

Similarly, if we consider the sequence $x(n), x(n-1), \dots, x(n-m), x(n-m-1)$, we can show that the partial correlation between $x(n)$ and $x(n-m-1)$ is given by (see Problem 7.6)

$$E\{e_m^b(n-1)e_m^{f*}(n)\} = r_{m+1}^f(n) + \mathbf{b}_m^H(n-1)\mathbf{r}_m^f(n) = \beta_m^f(n) \quad (7.3.32)$$

Because $r_{m+1}^f(n) = r_{m+1}^{b*}(n)$, we have the following simplification

$$\begin{aligned} \beta_m^f(n) &= \mathbf{b}_m^H(n-1)\mathbf{R}_m(n-1)\mathbf{R}_m^{-1}(n-1)\mathbf{r}_m^f(n) + r_{m+1}^f(n) \\ &= \mathbf{r}_m^{bH}(n-1)\mathbf{a}_m(n) + r_{m+1}^{b*}(n) = \beta_m^{b*}(n) \end{aligned}$$

which is known as *Burg's lemma* (Burg 1975). In order to simplify the notation, we define

$$\beta_m(n) \triangleq \beta_m^f(n) = \beta_m^{b*}(n) \quad (7.3.33)$$

Using (7.3.24), (7.3.28), and (7.3.30), we obtain

$$k_m^b(n)k_m^f(n) = \frac{|\beta_m(n)|^2}{P_m^f(n)P_m^b(n-1)} = \frac{|E\{e_m^b(n-1)e_m^{f*}(n)\}|^2}{E\{|e_m^f(n)|^2\}E\{|e_m^b(n-1)|^2\}} \quad (7.3.34)$$

which implies that

$$0 \leq k_m^f(n)k_m^b(n) \leq 1 \quad (7.3.35)$$

because the last term in (7.3.34) is the squared magnitude of the correlation coefficient of the random variables $e_m^f(n)$ and $e_m^b(n-1)$.

Order recursions for the MMSEs. Using the Levinson order recursions, we can obtain order-recursive formulas for the computation of $P_m^f(n)$, $P_m^b(n)$, and $P_m^c(n)$. Indeed, using (7.3.26), (7.3.27), and (7.3.29), we have

$$\begin{aligned} P_{m+1}^f(n) &= P_x(n) + \mathbf{r}_{m+1}^{fH}(n)\mathbf{a}_{m+1}(n) \\ &= P_x(n) + [\mathbf{r}_m^{fH}(n)r_{m+1}^{f*}(n)] \left\{ \begin{bmatrix} \mathbf{a}_m(n) \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m(n-1) \\ 1 \end{bmatrix} k_m^f(n) \right\} \\ &= P_x(n) + \mathbf{r}_m^{fH}(n)\mathbf{a}_m(n) + [\mathbf{r}_m^{fH}(n)\mathbf{b}_m(n-1) + r_{m+1}^{f*}(n)]k_m^f(n) \\ \text{or } P_{m+1}^f(n) &= P_m^f(n) + \beta_m^*(n)k_m^f(n) = P_m^f(n) - \frac{|\beta_m(n)|^2}{P_m^b(n-1)} \end{aligned} \quad (7.3.36)$$

If we work in a similar manner, we obtain

$$P_{m+1}^b(n) = P_m^b(n-1) + \beta_m(n)k_m^b(n) = P_m^b(n-1) - \frac{|\beta_m(n)|^2}{P_m^f(n)} \quad (7.3.37)$$

and

$$P_{m+1}^c(n) = P_m^c(n) - \beta_m^{c*}(n)k_m^c(n) = P_m^c(n) - \frac{|\beta_m^c(n)|^2}{P_m^b(n)} \quad (7.3.38)$$

If the subtrahends in the previous recursions are nonzero, increasing the order of the filter always improves the estimates, that is, $P_{m+1}^c(n) \leq P_m^c(n)$. Also, the conditions $P_m^f(n) \neq 0$ and $P_m^b(n) \neq 0$ are critical for the invertibility of $\mathbf{R}_m(n)$ and the computation of the optimum filters. The above relations are special cases of (7.1.45) and can be derived from the LDL^H decomposition (see Problem 7.7). The presence of vectors with mixed optimum nesting (upper-lower and lower-upper) in the definitions of $\beta_m(n)$ and $\beta_m^c(n)$ does not lead to similar order recursions for these quantities. However, for stationary processes we can break the dot products in (7.3.17) and (7.3.25) into scalar recursions, using an algorithm first introduced by Schür (see Section 7.6).

7.3.2 Lattice-Ladder Structure

We saw that the shift invariance of the input data vector made it possible to develop the Levinson recursions for the BLP and the FLP. We next show that these recursions can be used to simplify the triangular order-recursive estimation structure of Figure 7.1 by reducing it to a more efficient (linear instead of triangular), lattice-ladder filter structure that simultaneously provides the FLP, BLP, and FIR filtering estimates.

The computation of the estimation errors using direct-form structures is based on the following equations:

$$\begin{aligned} e_m^f(n) &= x(n) + \mathbf{a}_m^H(n)\mathbf{x}_m(n-1) \\ e_m^b(n) &= x(n-m) + \mathbf{b}_m^H(n)\mathbf{x}_m(n) \\ e_m(n) &= y(n) - \mathbf{c}_m^H(n)\mathbf{x}_m(n) \end{aligned} \quad (7.3.39)$$

Using (7.3.2), (7.3.27), and (7.3.39), we obtain

$$\begin{aligned} e_{m+1}^f(n) &= x(n) + \left\{ \begin{bmatrix} \mathbf{a}_m(n) \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m(n-1) \\ 1 \end{bmatrix} k_m^f(n) \right\}^H \begin{bmatrix} \mathbf{x}_m(n-1) \\ x(n-1-m) \end{bmatrix} \\ &= x(n) + \mathbf{a}_m^H(n)\mathbf{x}_m(n-1) + [\mathbf{b}_m^H(n-1)\mathbf{x}_m(n-1) + x(n-1-m)]k_m^{f*}(n) \end{aligned}$$

or

$$e_{m+1}^f(n) = e_m^f(n) + k_m^{f*}(n)e_m^b(n-1) \quad (7.3.40)$$

In a similar manner, we obtain

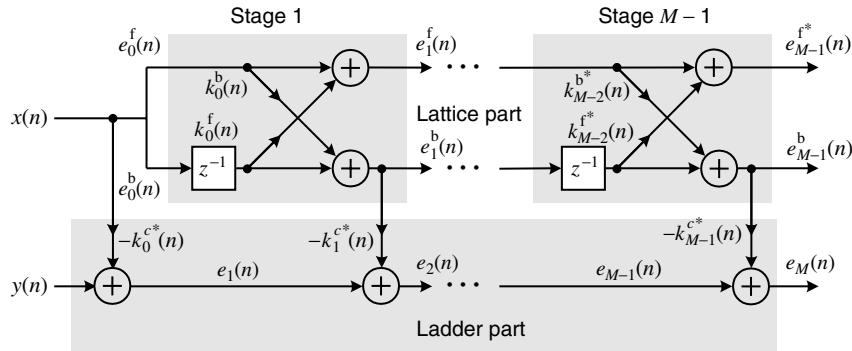
$$e_{m+1}^b(n) = e_m^b(n-1) + k_m^{b*}(n)e_m^f(n) \quad (7.3.41)$$

using (7.3.2), (7.3.23), and (7.3.39). Relations (7.3.40) and (7.3.41) are executed for $m = 0, 1, \dots, M-2$, with $e_0^f(n) = e_0^b(n) = x(n)$, and constitute a lattice filter that implements the FLP and the BLP.

Using (7.3.2), (7.3.15), and (7.3.39), we can show that the optimum filtering error can be computed by

$$e_{m+1}(n) = e_m(n) - k_m^{c*}(n)e_m^b(n) \quad (7.3.42)$$

which is executed for $m = 0, 1, \dots, M-1$, with $e_0(n) = y(n)$. The last equation provides the ladder part, which is coupled with the lattice predictor to implement the optimum filter. The result is the time-varying lattice-ladder structure shown in Figure 7.3. Notice that a new set of lattice-ladder coefficients has to be computed for every n , using $\mathbf{R}_m(n)$ and $\mathbf{d}_m(n)$. The parameters of the lattice-ladder structure can be obtained by LDL^H decomposition using (7.3.31). Suppose now that we know $P_0^f(n) = P_0^b(n) = P_x(n)$, $P_0^b(n-1)$, $P_0^c(n) = P_y(n)$, $\{\beta_m(n)\}_{0}^{M-1}$, and $\{\beta_m^c(n)\}_{0}^M$. Then we can determine $P_m^f(n)$, $P_m^b(n)$, and $P_m^c(n)$ for all m , using (7.3.36) through (7.3.38), and all filter coefficients, using (7.3.16), (7.3.24), and

**FIGURE 7.3**

Lattice-ladder structure for FIR optimum filtering and prediction.

(7.3.28). However, to obtain a completely time-recursive updating algorithm, we need time updatings for $\beta_m(n)$ and $\beta_m^c(n)$. As we will see later, this is possible if $\mathbf{R}(n)$ and $\mathbf{d}(n)$ are fixed or are defined by known time-updating formulas.

We recall that the BLP error vector $e_{m+1}^b(n)$ is the innovations vector of the data $\mathbf{x}_{m+1}(n)$. Notice that as a result of the shift invariance of the input data vector, the triangular decorrelator of the general linear estimator (see Figure 7.1) is replaced by a simpler, “linear” lattice structure. For stationary processes, the lattice-ladder filter is time-invariant, and we need to compute only one set of coefficients that can be used for all signals with the same \mathbf{R} and \mathbf{d} (see Section 7.5).

7.3.3 Simplifications for Stationary Stochastic Processes

When $x(n)$ and $y(n)$ are jointly wide-sense stationary (WSS), the optimum estimators are time-invariant and we have the following simplifications:

- All quantities are independent of n ; thus we do not need time recursions for the BLP parameters.
- $\mathbf{b}_m = \mathbf{J}\mathbf{a}_m^*$ (see Section 6.5.4), and thus we do not need the Levinson recursion for the BLP \mathbf{b}_m .

Both simplifications are a consequence of the Toeplitz structure of the correlation matrix \mathbf{R}_m . Indeed, comparing the partitionings

$$\mathbf{R}_{m+1}(n) = \begin{bmatrix} \mathbf{R}_m & \mathbf{J}\mathbf{r}_m \\ \mathbf{r}_m^H \mathbf{J} & r(0) \end{bmatrix} = \begin{bmatrix} r(0) & \mathbf{r}_m^T \\ \mathbf{r}_m^* & \mathbf{R}_m \end{bmatrix} \quad (7.3.43)$$

where

$$\mathbf{r}_m \triangleq [r(1) \ r(2) \ \cdots \ r(m)]^T \quad (7.3.44)$$

with (7.3.3) and (7.3.4), we have

$$\begin{aligned} \mathbf{R}_m(n) &= \mathbf{R}_m(n-1) = \mathbf{R}_m \\ \mathbf{r}_m^f(n) &= \mathbf{r}_m^* \\ \mathbf{r}_m^b(n) &= \mathbf{J}\mathbf{r}_m \end{aligned} \quad (7.3.45)$$

which can be used to simplify the order recursions derived for nonstationary processes. Indeed, we can easily show that

$$\mathbf{a}_{m+1} = \begin{bmatrix} \mathbf{a}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m \\ 1 \end{bmatrix} k_m \quad (7.3.46)$$

where

$$\mathbf{b}_m = \mathbf{J}\mathbf{a}_m^* \quad (7.3.47)$$

$$k_m \triangleq k_m^f = k_m^{b*} = -\frac{\beta_m}{P_m} \quad (7.3.48)$$

$$\beta_m \triangleq \beta_m^f = \beta_m^{b*} = \mathbf{b}_m^H \mathbf{r}_m^* + r^*(m+1) \quad (7.3.49)$$

$$P_m \triangleq P_m^b = P_m^f = P_{m-1} + \beta_{m-1}^* k_{m-1} = P_{m-1} + \beta_{m-1} k_{m-1}^* \quad (7.3.50)$$

This recursion provides a complete order-recursive algorithm for the computation of the FLP \mathbf{a}_m for $1 \leq m \leq M$ from the autocorrelation sequence $r(l)$ for $0 \leq l \leq M$.

The optimum filters \mathbf{c}_m for $1 \leq m \leq M$ can be obtained from the quantities \mathbf{a}_m and P_m for $1 \leq m \leq M-1$ and \mathbf{d}_M , using the following Levinson recursion

$$\mathbf{c}_{m+1} = \begin{bmatrix} \mathbf{c}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{J}\mathbf{a}_m \\ 1 \end{bmatrix} k_m^c \quad (7.3.51)$$

where

$$k_m^c \triangleq \frac{\beta_m^c}{P_m} \quad (7.3.52)$$

and

$$\beta_m^c = \mathbf{b}_m^H \mathbf{d}_m + d_{m+1} \quad (7.3.53)$$

The MMSE P_m^c is then given by

$$P_m^c = P_{m-1}^c - \beta_m^c k_m^c \quad (7.3.54)$$

and although it is not required by the algorithm, P_m^c is useful for selecting the order of the optimum filter. Both algorithms are discussed in greater detail in Section 7.4.

7.3.4 Algorithms Based on the \mathbf{UDU}^H Decomposition

Hermitian positive definite matrices can also be factorized as

$$\mathbf{R} = \mathbf{U}\bar{\mathbf{D}}\mathbf{U}^H \quad (7.3.55)$$

where \mathbf{U} is a unit upper triangular matrix and $\bar{\mathbf{D}}$ is a diagonal matrix with positive elements $\bar{\xi}_i$, using the function $[\mathbf{U}, \mathbf{D}] = \text{udut}(\mathbf{R})$ (see Problem 7.8). Using the decomposition (7.3.55), we can obtain the solution of the normal equations by solving the triangular systems, first for $\bar{\mathbf{k}}$

$$(\mathbf{U}\bar{\mathbf{D}})\bar{\mathbf{k}} \triangleq \mathbf{d} \quad (7.3.56)$$

and then for \mathbf{c}

$$\mathbf{U}^H \mathbf{c} = \bar{\mathbf{k}} \quad (7.3.57)$$

by backward and forward substitution, respectively. The MMSE estimate can be computed by

$$\hat{y} = \mathbf{c}^H \mathbf{x} = \bar{\mathbf{k}}^H \bar{\mathbf{w}} \quad (7.3.58)$$

where

$$\bar{\mathbf{w}} \triangleq \mathbf{U}^{-1} \mathbf{x} \quad (7.3.59)$$

is an *innovations* vector for the data vector \mathbf{x} . It can be shown that the rows of $\mathbf{A} \triangleq \mathbf{U}^{-1}$ are the linear MMSE estimator of x_m based on $[x_{m+1} \ x_{m+2} \ \dots \ x_M]^T$. Furthermore, the \mathbf{UDU}^H factorization (7.3.55) can be obtained by the Gram-Schmidt algorithm, starting with x_M and proceeding “backward” to x_1 (see Problem 7.9). The various triangular decompositions of the correlation matrix \mathbf{R} are summarized in Table 7.1.

If we define the reversed vectors $\tilde{\mathbf{x}} \triangleq \mathbf{J}\mathbf{x}$ and $\tilde{\mathbf{w}} \triangleq \mathbf{J}\mathbf{w}$, we obtain

$$\tilde{\mathbf{x}} = \mathbf{J}\mathbf{x} = \mathbf{J}\mathbf{L}\mathbf{J}\mathbf{J}\mathbf{w} = \mathbf{J}\mathbf{L}\mathbf{J}\tilde{\mathbf{w}} \triangleq \tilde{\mathbf{U}}\tilde{\mathbf{w}} \quad (7.3.60)$$

TABLE 7.1
Summary of the triangular decompositions of the correlation matrix.

Matrix \ Decomposition				
\mathbf{R}	\mathbf{LDL}^H		$\mathbf{U}\bar{\mathbf{D}}\mathbf{U}^H$	$\mathbf{A} = \mathbf{U}^{-1}$
\mathbf{R}^{-1}	$\mathbf{A}^H\bar{\mathbf{D}}\mathbf{A}$		$\mathbf{B}^H\mathbf{D}^{-1}\mathbf{B}$	$\mathbf{B} = \mathbf{L}^{-1}$

because $\mathbf{J}^2 = \mathbf{I}$ and $\tilde{\mathbf{U}} = \mathbf{JLJ}$ is upper triangular. The correlation matrix of $\tilde{\mathbf{x}}$ is

$$\tilde{\mathbf{R}} = E\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^H\} = \tilde{\mathbf{U}}\tilde{\mathbf{D}}\tilde{\mathbf{U}}^H \quad (7.3.61)$$

where $\tilde{\mathbf{D}} \triangleq E\{\tilde{\mathbf{w}}\tilde{\mathbf{w}}^H\}$ is the diagonal correlation matrix of $\tilde{\mathbf{w}}$. Equation (7.3.61) provides the \mathbf{UDU}^H decomposition of $\tilde{\mathbf{R}}$.

A natural question arising at this point is whether we can develop order-recursive algorithms and structures for optimum filtering using the \mathbf{UDU}^H instead of the \mathbf{LDL}^H decomposition of the correlation matrix. The \mathbf{UDU}^H decomposition is coupled to a partitioning of $\mathbf{R}_{m+1}(n)$ starting at the lower right corner and moving to the upper left corner that provides the following sequence of submatrices

$$\mathbf{R}_1(n-m) \rightarrow \mathbf{R}_2(n-m+1) \rightarrow \dots \rightarrow \mathbf{R}_m(n) \quad (7.3.62)$$

which, in turn, are related to the FLPs

$$\mathbf{a}_1(n-m) \rightarrow \mathbf{a}_2(n-m+1) \rightarrow \dots \rightarrow \mathbf{a}_m(n) \quad (7.3.63)$$

and the FLP errors

$$e_1^f(n-m+1) \rightarrow e_2^f(n-m+2) \rightarrow \dots \rightarrow e_m^f(n) \quad (7.3.64)$$

If we define the FLP error vector

$$\mathbf{e}_{m+1}^f(n) = [e_m^f(n) \ e_{m-1}^f(n-1) \ \dots \ e_0^f(n-m)]^T \quad (7.3.65)$$

we see that

$$\mathbf{e}_{m+1}^f(n) = \mathbf{A}_{m+1}(n)\mathbf{x}_{m+1}(n) \quad (7.3.66)$$

where

$$\mathbf{A}_{m+1}(n) \triangleq \begin{bmatrix} 1 & a_1^{(m)}(n) & a_2^{(m)}(n) & \dots & a_m^{(m)}(n) \\ 0 & 1 & a_1^{(m-1)}(n-1) & \dots & a_{m-1}^{(m-1)}(n-1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & a_1^{(1)}(n-m+1) \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.3.67)$$

The elements of the vector $\mathbf{e}_{m+1}^f(n)$ are uncorrelated, and the \mathbf{LDL}^H decomposition of the inverse correlation matrix (see Problem 7.10) is given by

$$\mathbf{R}_{m+1}^{-1}(n) = \mathbf{A}_{m+1}^H(n)\bar{\mathbf{D}}_{m+1}^{-1}(n)\mathbf{A}_{m+1}(n) \quad (7.3.68)$$

where $\bar{\mathbf{D}}_{m+1}(n)$ is the correlation matrix of $\mathbf{e}_{m+1}^f(n)$. Using $\mathbf{e}_{m+1}^f(n)$ as an orthogonal basis instead of $\mathbf{e}_{m+1}^b(n)$ results in a complicated lattice structure because of the additional delay elements required for the forward prediction errors. Thus, the \mathbf{LDL}^H decomposition is the method of choice in practical applications for linear MMSE estimation.

Since the correlation matrix of a stationary, stochastic process is Toeplitz, we can explore its special structure to develop efficient, order-recursive algorithms for the linear system solution, matrix triangularization, and matrix inversion. Although we develop such algorithms in the context of optimum FIR filtering and prediction, the results apply to other applications involving Toeplitz matrices (Golub and van Loan 1996).

Suppose that we know the optimum filter \mathbf{c}_m is given by

$$\mathbf{c}_m = \mathbf{R}_m^{-1} \mathbf{d}_m \quad (7.4.1)$$

and we wish to use it to compute the optimum filter \mathbf{c}_{m+1}

$$\mathbf{c}_{m+1} = \mathbf{R}_{m+1}^{-1} \mathbf{d}_{m+1} \quad (7.4.2)$$

We first notice that the matrix \mathbf{R}_{m+1} and the vector \mathbf{d}_{m+1} can be partitioned as follows

$$\mathbf{R}_{m+1} = \begin{bmatrix} r(0) & \cdots & r(m-1) & | & r(m) \\ \vdots & \ddots & \vdots & & \vdots \\ r^*(m-1) & \cdots & r(0) & | & r(1) \\ \hline r^*(m) & \cdots & r^*(1) & | & r(0) \end{bmatrix} = \begin{bmatrix} \mathbf{R}_m & \mathbf{Jr}_m \\ \mathbf{r}_m^H \mathbf{J} & r(0) \end{bmatrix} \quad (7.4.3)$$

$$\mathbf{d}_{m+1} = \begin{bmatrix} \mathbf{d}_m \\ d_{m+1} \end{bmatrix} \quad (7.4.4)$$

which shows that both quantities have the optimum nesting property, that is, $\mathbf{R}_{m+1}^{[m]} = \mathbf{R}_m$ and $\mathbf{d}_{m+1}^{[m]} = \mathbf{d}_m$.

Using the matrix inversion by partitioning lemma (7.1.24), we obtain

$$\mathbf{R}_{m+1}^{-1} = \begin{bmatrix} \mathbf{R}_m^{-1} & \mathbf{0} \\ \mathbf{0}^H & 0 \end{bmatrix} + \frac{1}{P_m^b} \begin{bmatrix} \mathbf{b}_m \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{b}_m^H & 1 \end{bmatrix} \quad (7.4.5)$$

where

$$\mathbf{b}_m = -\mathbf{R}_m^{-1} \mathbf{Jr}_m \quad (7.4.6)$$

and

$$P_m^b = r(0) + \mathbf{r}_m^H \mathbf{J} \mathbf{b}_m \quad (7.4.7)$$

Substitution of (7.4.4) and (7.4.5) into (7.4.2) gives

$$\mathbf{c}_{m+1} = \begin{bmatrix} \mathbf{c}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m \\ 1 \end{bmatrix} k_m^c \quad (7.4.8)$$

where

$$k_m^c \triangleq \frac{\beta_m^c}{P_m^b} \quad (7.4.9)$$

and

$$\beta_m^c \triangleq \mathbf{b}_m^H \mathbf{d}_m + d_{m+1} = -\mathbf{c}_m^H \mathbf{Jr}_m + d_{m+1} \quad (7.4.10)$$

Equations (7.4.8) through (7.4.10) constitute a Levinson recursion for the optimum filter and have been obtained without making use of the Toeplitz structure of \mathbf{R}_{m+1} .

The development of a complete order-recursive algorithm is made possible by exploiting the Toeplitz structure. Indeed, when the correlation matrix \mathbf{R}_m is Toeplitz, we have

$$\mathbf{b}_m = \mathbf{J}\mathbf{a}_m^* \quad (7.4.11)$$

and

$$P_m \triangleq P_m^b = P_m^f \quad (7.4.12)$$

as we recall from Section 6.5. Since we can determine \mathbf{b}_m from \mathbf{a}_m , we need to perform only *one* Levinson recursion, either for \mathbf{b}_m or for \mathbf{a}_m .

To avoid the use of the lower right corner partitioning, we develop an order recursion for the FLP \mathbf{a}_m . Indeed, to compute \mathbf{a}_{m+1} from \mathbf{a}_m , recall that

$$\mathbf{a}_{m+1} = -\mathbf{R}_{m+1}^{-1} \mathbf{r}_{m+1}^* \quad (7.4.13)$$

which, when combined with (7.4.5) and

$$\mathbf{r}_{m+1} = \begin{bmatrix} \mathbf{r}_m \\ r(m+1) \end{bmatrix} \quad (7.4.14)$$

leads to the Levinson recursion

$$\mathbf{a}_{m+1} = \begin{bmatrix} \mathbf{a}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m \\ 1 \end{bmatrix} k_m \quad (7.4.15)$$

where

$$k_m \triangleq -\frac{\beta_m}{P_m} \quad (7.4.16)$$

$$\beta_m \triangleq \mathbf{b}_m^H \mathbf{r}_m^* + r^*(m+1) = \mathbf{a}_m^T \mathbf{J} \mathbf{r}_m^* + r^*(m+1) \quad (7.4.17)$$

and

$$P_m = r(0) + \mathbf{r}_m^H \mathbf{a}_m^* = r(0) + \mathbf{a}_m^T \mathbf{r}_m \quad (7.4.18)$$

Also, using (7.1.46) and (7.2.6), we can show that

$$P_m = \frac{\det \mathbf{R}_{m+1}}{\det \mathbf{R}_m} \quad \text{and} \quad \det \mathbf{R}_m = \prod_{i=0}^{m-1} P_i \quad \text{with } P_0 = r(0) \quad (7.4.19)$$

which emphasizes the importance of P_m for the invertibility of the autocorrelation matrix. The MMSE P_m for either the forward or the backward predictor of order m can be computed recursively as follows:

$$\begin{aligned} P_{m+1} &= r(0) + [\mathbf{r}_m^H r^*(m+1)] \left\{ \begin{bmatrix} \mathbf{a}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m \\ 1 \end{bmatrix} k_m \right\}^* \\ &= r(0) + \mathbf{r}_m^H \mathbf{a}_m^* + [\mathbf{r}_m^H \mathbf{b}_m^* + r^*(m+1)] k_m^* \end{aligned} \quad (7.4.20)$$

or

$$P_{m+1} = P_m + \beta_m k_m^* = P_m + \beta_m^* k_m = P_m(1 - |k_m|^2) \quad (7.4.21)$$

The following recursive formula for the computation of the MMSE

$$P_{m+1}^c = P_m^c - \beta_m^c k_m^c = P_m^c - \beta_m^c k_m^c \quad (7.4.22)$$

can be found by using (7.4.8).

Therefore, the algorithm of Levinson consists of two parts: a set of recursions that compute the optimum FLP or BLP and a set of recursions that use this information to compute the optimum filter. The part that computes the linear predictors is known as the Levinson-Durbin algorithm and was pointed out by Durbin (1960). From a linear system solution point of view, the algorithm of Levinson solves a Hermitian Toeplitz system with *arbitrary* right-hand side vector \mathbf{d} ; the Levinson-Durbin algorithm deals with the special case $\mathbf{d} = \mathbf{r}^*$ or $\mathbf{J}\mathbf{r}$. Additional interpretations are discussed in Section 7.7.

Algorithm of Levinson-Durbin

The algorithm of Levinson-Durbin, which takes as input the autocorrelation sequence $r(0), r(1), \dots, r(M)$ and computes the quantities \mathbf{a}_m , P_m , and k_{m-1} for $m = 1, 2, \dots, M$, is illustrated in the following examples.

EXAMPLE 7.4.1. Determine the FLP $\mathbf{a}_2 = [a_1^{(2)} \ a_2^{(2)}]^T$ and the MMSE P_2 from the autocorrelation values $r(0), r(1)$, and $r(2)$.

Solution. To initialize the algorithm, we determine the first-order predictor by solving the normal equations $r(0)a_1^{(1)} = -r^*(1)$. Indeed, we have

$$a_1^{(1)} = -\frac{r^*(1)}{r(0)} = k_0 = -\frac{\beta_0}{P_0}$$

which implies that $\beta_0 = r^*(1)$ $P_0 = r(0)$

To update to order 2, we need k_1 and hence β_1 and P_1 , which can be obtained by

$$\beta_1 = a_1^{(1)}r^*(1) + r^*(2) = \frac{r(0)r^*(2) - [r^*(1)]^2}{r(0)}$$

$$P_1 = P_0 + \beta_0 k_0^* = \frac{r^2(0) - |r(1)|^2}{r(0)}$$

as

$$k_1 = \frac{[r^*(1)]^2 - r(0)r^*(2)}{r^2(0) - |r(1)|^2}$$

Therefore, using Levinson's recursion, we obtain

$$a_1^{(2)} = a_1^{(1)} + a_1^{(1)*}k_1 = \frac{r(1)r^*(2) - r(0)r^*(1)}{r^2(0) - |r(1)|^2}$$

and

$$a_2^{(2)} = k_1$$

which agree with the results obtained in Example 6.5.1. The resulting MMSE can be found by using $P_2 = P_1 + \beta_1 k_1^*$.

EXAMPLE 7.4.2. Use the Levinson-Durbin algorithm to compute the third-order forward predictor for a signal $x(n)$ with autocorrelation sequence $r(0) = 3, r(1) = 2, r(2) = 1$, and $r(3) = \frac{1}{2}$.

Solution. To initialize the algorithm, we notice that the first-order predictor is given by $r(0)a_1^{(1)} = -r(1)$ and that for $m = 0$, (7.4.15) gives $a_1^{(1)} = k_0$. Hence, we have

$$a_1^{(1)} = -\frac{r(1)}{r(0)} = -\frac{2}{3} = k_0 = \frac{\beta_0}{P_0}$$

which implies

$$P_0 = r(0) = 3 \quad \beta_0 = r(1) = 2$$

To compute \mathbf{a}_2 by (7.4.15), we need $a_1^{(1)}, b_1^{(1)} = a_1^{(1)}$, and $k_1 = -\beta_1/P_1$. From (7.4.21), we have

$$P_1 = P_0 + \beta_0 k_0 = 3 + 2(-\frac{2}{3}) = \frac{5}{3}$$

and from (7.4.17)

$$\beta_1 = \mathbf{r}_1^T \mathbf{J} \mathbf{a}_1 + r(2) = 2(-\frac{2}{3}) + 1 = -\frac{1}{3}$$

Hence,

$$k_1 = -\frac{\beta_1}{P_1} = -\frac{-\frac{1}{3}}{\frac{5}{3}} = \frac{1}{5}$$

and

$$\mathbf{a}_2 = \begin{bmatrix} -\frac{2}{3} \\ 0 \end{bmatrix} + \frac{1}{5} \begin{bmatrix} -\frac{2}{3} \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{4}{5} \\ \frac{1}{5} \end{bmatrix}$$

Continuing in the same manner, we obtain

$$P_2 = P_1 + \beta_1 k_1 = \frac{5}{3} + (-\frac{1}{3})(\frac{1}{5}) = \frac{8}{5}$$

$$\beta_2 = \mathbf{r}_2^T \mathbf{J} \mathbf{a}_2 + r(3) = [2 \quad 1] \begin{bmatrix} \frac{1}{5} \\ -\frac{4}{5} \end{bmatrix} + \frac{1}{2} = \frac{1}{10}$$

$$k_2 = -\frac{\beta_2}{P_2} = -\frac{\frac{1}{10}}{\frac{8}{5}} = -\frac{1}{16}$$

$$\mathbf{a}_3 = \begin{bmatrix} \mathbf{a}_2 \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{J} \mathbf{a}_2 \\ 1 \end{bmatrix} k_3 = \begin{bmatrix} -\frac{4}{5} \\ \frac{1}{5} \\ 0 \end{bmatrix} - \begin{bmatrix} \frac{1}{5} \\ -\frac{4}{5} \\ 1 \end{bmatrix} \frac{1}{16} = \begin{bmatrix} -\frac{13}{16} \\ \frac{1}{4} \\ -\frac{1}{16} \end{bmatrix}$$

$$P_3 = P_2 + \beta_2 k_2 = \frac{8}{5} + \frac{1}{10} \left(-\frac{1}{16} \right) = \frac{51}{32}$$

The algorithm of Levinson-Durbin, summarized in Table 7.2, requires M^2 operations and is implemented by the function `[a, k, P0] = durbin(r, M)`.

TABLE 7.2
Summary of the Levinson-
Durbin algorithm.

-
1. **Input:** $r(0), r(1), r(2), \dots, r(M)$
 2. **Initialization**
 - (a) $P_0 = r(0), \beta_0 = r^*(1)$
 - (b) $k_0 = -r^*(1)/r(0), a_1^{(1)} = k_0$
 3. **For** $m = 1, 2, \dots, M-1$
 - (a) $P_m = P_{m-1} + \beta_{m-1} k_{m-1}^*$
 - (b) $\mathbf{r}_m = [r(1) \ r(2) \ \dots \ r(m)]^T$
 - (c) $\beta_m = \mathbf{a}_m^T \mathbf{J} \mathbf{r}_m^* + r^*(m+1)$
 - (d) $k_m = -\frac{\beta_m}{P_m}$
 - (e) $\mathbf{a}_{m+1} = \begin{bmatrix} \mathbf{a}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{J} \mathbf{a}_m^* \\ 1 \end{bmatrix} k_m$
 4. $P_M = P_{M-1} + \beta_M k_M^*$
 5. **Output:** $\mathbf{a}_M, \{k_m\}_0^{M-1}, \{P_m\}_1^M$
-

Algorithm of Levinson

The next example illustrates the algorithm of Levinson that can be used to solve a system of linear equations with a Hermitian Toeplitz matrix and arbitrary right-hand side vector.

EXAMPLE 7.4.3. Consider an optimum filter with input $x(n)$ and desired response $y(n)$. The autocorrelation of the input signal is $r(0) = 3, r(1) = 2$, and $r(2) = 1$. The cross-correlation between the desired response and input is $d_1 = 1, d_2 = 2$, and $d_3 = \frac{5}{2}$; and the power of $y(n)$ is $P_y = 3$. Design a third-order optimum FIR filter, using the algorithm of Levinson.

Solution. We start initializing the algorithm by noticing that for $m = 0$ we have $r(0)a_1^{(1)} = -r(1)$, which gives

$$a_1^{(1)} = k_0 = -\frac{r(1)}{r(0)} = -\frac{2}{3}$$

$$P_0 = r(0) = 3 \quad \beta_0 = r(1) = 2$$

and

$$P_1 = P_0 + \beta_0 k_0 = 3 + 2(-\frac{2}{3}) = \frac{5}{3}$$

Next, we compute the Levinson recursion for the first-order optimum filter

$$P_0^c = 5 \quad \beta_0^c = d_1 = 1$$

$$k_0^c = c_1^{(1)} = \frac{d_1}{r(0)} = \frac{1}{3}$$

$$P_1^c = P_0^c - \beta_0^c k_0^c = 3 - 1(\frac{1}{3}) = \frac{8}{3}$$

Then we carry the Levinson recursion for $m = 1$ to obtain

$$\beta_1 = \mathbf{r}_1^T \mathbf{J} \mathbf{a}_1 + r(2) = 2(-\frac{2}{3}) + 1 = -\frac{1}{3}$$

$$k_1 = -\frac{\beta_1}{P_1} = -\frac{-\frac{1}{3}}{\frac{8}{3}} = \frac{1}{5}$$

$$\mathbf{a}_2 = \begin{bmatrix} -\frac{2}{3} \\ 0 \end{bmatrix} + \frac{1}{5} \begin{bmatrix} -\frac{2}{3} \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{4}{5} \\ \frac{1}{5} \end{bmatrix}$$

$$P_2^c = P_1^c + \beta_1 k_1 = \frac{8}{3} + (-\frac{1}{3})(\frac{1}{5}) = \frac{8}{5}$$

for the optimum predictor, and

$$\beta_1^c = \mathbf{a}_1^T \mathbf{J} \mathbf{d}_1 + d_2 = -\frac{2}{3}(1) + 2 = \frac{4}{3}$$

$$k_1^c = \frac{\beta_1^c}{P_1} = \frac{\frac{4}{3}}{\frac{8}{3}} = \frac{4}{5}$$

$$\mathbf{c}_2 = \begin{bmatrix} \frac{1}{3} \\ 0 \end{bmatrix} + \frac{4}{5} \begin{bmatrix} -\frac{2}{3} \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{5} \\ \frac{4}{5} \end{bmatrix}$$

$$P_2^c = P_1^c - \beta_1^c k_1^c = \frac{8}{3} - \frac{4}{3}(\frac{4}{5}) = \frac{8}{5}$$

for the optimum filter. The last recursion ($m = 2$) is carried out only for the optimum filter and gives

$$\beta_2^c = \mathbf{a}_2^T \mathbf{J} \mathbf{d}_2 + d_3 = \begin{bmatrix} \frac{1}{5} & -\frac{4}{5} \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \frac{5}{2} = \frac{11}{10}$$

$$k_2^c = \frac{\beta_2^c}{P_2} = \frac{\frac{11}{10}}{\frac{8}{5}} = \frac{11}{16}$$

$$\mathbf{c}_3 = \begin{bmatrix} \mathbf{c}_2 \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{J} \mathbf{a}_2 \\ 1 \end{bmatrix} k_2^c = \begin{bmatrix} -\frac{1}{5} \\ \frac{4}{5} \\ 0 \end{bmatrix} + \frac{11}{16} \begin{bmatrix} \frac{1}{5} \\ -\frac{4}{5} \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{16} \\ \frac{1}{4} \\ \frac{11}{16} \end{bmatrix}$$

$$P_3^c = P_2^c - \beta_2^c k_2^c = \frac{8}{5} - \frac{11}{10}(\frac{11}{16}) = \frac{27}{32}$$

The algorithm of Levinson, summarized in Table 7.3, is implemented by the MATLAB function `[c, k, kc, Pc] = levins(R, d, Py, M)` and requires $2M^2$ operations because it involves two dot products and two scalar-vector multiplications. A parallel processing implementation of the algorithm is not possible because the dot products involve additions that cannot be executed simultaneously. Notice that adding $M = 2^q$ numbers using $M/2$ adders requires $q = \log_2 M$ steps. This bottleneck can be avoided by using the algorithm of Schür (see Section 7.6).

Minimum phase and autocorrelation extension

Using (7.4.16), we can also express the recursion (7.4.21) as

$$P_{m+1} = P_m(1 - |k_m|^2) = P_m - \frac{|\beta_m|^2}{P_m} \quad (7.4.23)$$

TABLE 7.3
Summary of the algorithm of Levinson.

-
1. **Input:** $\{r(l)\}_0^M, \{d_m\}_1^M, P_y$
 2. **Initialization**
 - (a) $P_0 = r(0), \beta_0 = r^*(1), P_0^c = P_y$
 - (b) $k_0 = -\beta_0/P_0, a_1^{(1)} = k_0$
 - (c) $\beta_0^c = d_1$
 - (d) $k_0^c = -\beta_0^c/P_0, c_1^{(1)} = k_0^c$
 - (e) $P_1^c = P_0^c + \beta_0^c k_0^{c*}$
 3. **For** $m = 1, 2, \dots, M-1$
 - (a) $\mathbf{r}_m = [r(1) \ r(2) \ \dots \ r(m)]^T$
 - (b) $\beta_m = \mathbf{a}_m^T \mathbf{J} \mathbf{r}_m^* + r^*(m+1)$
 - (c) $P_m = P_{m-1} + \beta_{m-1} k_{m-1}^*$
 - (d) $k_m = -\frac{\beta_m}{P_m}$
 - (e) $\mathbf{a}_{m+1} = \begin{bmatrix} \mathbf{a}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{J} \mathbf{a}_m^* \\ 1 \end{bmatrix} k_m$
 - (f) $\beta_m^c = -\mathbf{c}_m^H \mathbf{J} \mathbf{r}_m + d_{m+1}$
 - (g) $k_m^c = \frac{\beta_m^c}{P_m}$
 - (h) $\mathbf{c}_{m+1} = \begin{bmatrix} \mathbf{c}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{J} \mathbf{a}_m^* \\ 1 \end{bmatrix} k_m^c$
 - (i) $P_{m+1}^c = P_m^c + \beta_m^c k_m^{c*}$
 4. **Output:** $\mathbf{a}_M, \mathbf{c}_M, \{k_m, k_m^c\}_0^{M-1}, \{P_m, P_m^c\}_0^M$
-

which, since $P_m \geq 0$, implies that

$$P_{m+1} \leq P_m \quad (7.4.24)$$

and since the matrix \mathbf{R}_m is positive definite, then $P_m > 0$ and (7.4.23) implies that

$$|k_m| \leq 1 \quad (7.4.25)$$

for all $1 \leq m < M$. If

$$P_0 > \dots > P_{M-1} > P_M = 0 \quad (7.4.26)$$

then the process $x(n)$ is predictable and (7.4.23) implies that

$$k_M = \pm 1 \quad \text{and} \quad |k_m| < 1 \quad 1 \leq m < M \quad (7.4.27)$$

(see Section 6.6.4). Also if

$$P_{M-1} > P_M = \dots = P_\infty = P > 0 \quad (7.4.28)$$

from (7.4.23) we have

$$k_m = 0 \quad \text{for } m > M \quad (7.4.29)$$

which implies that the process $x(n)$ is AR(M) and $e_M^f(n) \sim WN(0, P_M)$ (see Section 4.2.3). Finally, we note that since the sequence P_0, P_1, P_2, \dots is nonincreasing, its limit as $m \rightarrow \infty$ exists and is nonnegative. A regular process must satisfy $|k_m| < 1$ for all m , because $|k_m| = 1$ implies that $P_m = 0$, which contradicts the regularity assumption.

For $m = 0$, (7.4.19) gives $P_0 = r(0)$. Carrying out (7.4.23) from $m = 0$ to $m = M$, we obtain

$$P_M = r(0) \prod_{m=1}^M (1 - |k_{m-1}|^2) \quad (7.4.30)$$

which converges, as $M \rightarrow \infty$, if $|k_m| < 1$.

To compute the forward prediction error of an FLP of order m , we use the formula

$$e_m^f(n) = x(n) + \mathbf{a}_m^H \mathbf{x}_m(n-1) = x(n) + \sum_{k=1}^m a_k^{(m)*} x(n-k) \quad (7.5.1)$$

Similarly, for the BLP we have

$$e_m^b(n) = x(n-m) + \mathbf{b}_m^H \mathbf{x}_m(n) = x(n-m) + \sum_{k=0}^{m-1} b_k^{(m)*} x(n-k) \quad (7.5.2)$$

Both filters can be implemented using the direct-form filter structure shown in Figure 7.4. Since \mathbf{a}_m and \mathbf{b}_m do not have the optimum nesting property, we cannot obtain order-recursive direct-form structures for the computation of the prediction errors. However, next we show that we can derive an order-recursive lattice-ladder structure for the implementation of optimum predictors and filters using the algorithm of Levinson.

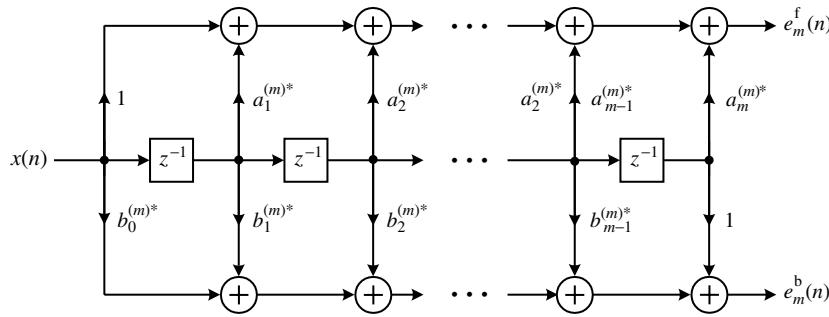


FIGURE 7.4
Direct-form structure for the computation of the m th-order forward and backward prediction errors.

7.5.1 Lattice-Ladder Structures

We note that the data vector for the $(m+1)$ st-order predictor can be partitioned in the following ways:

$$\mathbf{x}_{m+1}(n) = [x(n) \ x(n-1) \ \cdots \ x(n-m+1) \ x(n-m)]^T \\ = [\mathbf{x}_m^T(n) \ x(n-m)]^T \quad (7.5.3)$$

$$= [x(n) \ \mathbf{x}_m^T(n-1)]^T \quad (7.5.4)$$

Using (7.5.1), (7.5.3), (7.4.15), and (7.5.2), we obtain

$$\begin{aligned} e_{m+1}^f(n) &= x(n) + \left\{ \begin{bmatrix} \mathbf{a}_m \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m \\ 1 \end{bmatrix} k_m \right\}^H \begin{bmatrix} \mathbf{x}_m(n-1) \\ x(n-m-1) \end{bmatrix} \\ &= x(n) + \mathbf{a}_m^H \mathbf{x}_m(n-1) + k_m^* [\mathbf{b}_m^H \mathbf{x}_m(n-1) + x(n-m-1)] \\ \text{or} \quad e_{m+1}^f(n) &= e_m^f(n) + k_m^* e_m^b(n-1) \end{aligned} \quad (7.5.5)$$

Using (7.4.11) and (7.4.15), we obtain the following Levinson-type recursion for the backward predictor:

$$\mathbf{b}_{m+1} = \begin{bmatrix} 0 \\ \mathbf{b}_m \end{bmatrix} + \begin{bmatrix} 1 \\ \mathbf{a}_m \end{bmatrix} k_m^*$$

The backward prediction error is

$$\begin{aligned} e_{m+1}^b(n) &= x(n-m-1) + \left\{ \begin{bmatrix} 0 \\ \mathbf{b}_m \end{bmatrix} + \begin{bmatrix} 1 \\ \mathbf{a}_m \end{bmatrix} k_m^* \right\}^H \begin{bmatrix} x(n) \\ \mathbf{x}_m(n-1) \end{bmatrix} \\ &= x(n-m-1) + \mathbf{b}_m^H \mathbf{x}_m(n-1) + k_m [x(n) + \mathbf{a}_m^H \mathbf{x}_m(n-1)] \\ \text{or} \quad e_{m+1}^b(n) &= e_m^b(n-1) + k_m e_m^f(n) \end{aligned} \quad (7.5.6)$$

Recursions (7.5.5) and (7.5.6) can be computed for $m = 0, 1, \dots, M-1$. The initial conditions $e_0^f(n)$ and $e_0^b(n)$ are easily obtained from (7.5.1) and (7.5.2). The recursions also lead to the following all-zero lattice algorithm

$$\begin{aligned} e_0^f(n) &= e_0^b(n) = x(n) \\ e_m^f(n) &= e_{m-1}^f(n) + k_{m-1}^* e_{m-1}^b(n-1) \quad m = 1, 2, \dots, M \\ e_m^b(n) &= k_{m-1} e_{m-1}^f(n) + e_{m-1}^b(n-1) \quad m = 1, 2, \dots, M \\ e(n) &= e_M^f(n) \end{aligned} \quad (7.5.7)$$

that is implemented using the structure shown in Figure 7.5. The *lattice parameters* k_m are known as *reflection coefficients* in the speech processing and geophysics areas.

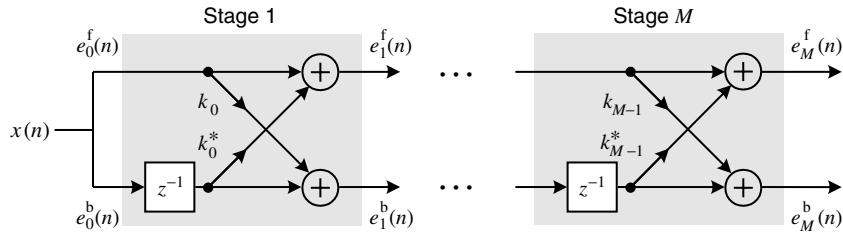


FIGURE 7.5

All-zero lattice structure for the implementation of the forward and backward prediction error filters.

The Levinson recursion for the optimum filter, (7.4.8) through (7.4.10), adds a *ladder* part to the lattice structure for the forward and backward predictors. Using (7.4.8), (7.5.7), and the partitioning in (7.5.3), we can express the filtering error of order $m+1$ in terms of $e_m(n)$ and $e_m^b(n)$ as follows

$$e_{m+1}(n) = y(n) - \mathbf{c}_{m+1}^H \mathbf{x}_{m+1}(n) = e_m(n) - k_m^{c*} e_m^b(n) \quad (7.5.8)$$

for $m = 0, 1, \dots, M-1$. The resulting lattice-ladder structure is similar to the one shown in Figure 7.3. However, owing to stationarity all coefficients are constant, and $k_m^f(n) = k_m^b(n) = k_m$. We note that the efficient solution of the M th-order optimum filtering problem is derived from the solution of the $(M-1)$ st-order forward and backward prediction problems of the input process. In fact, the lattice part serves to decorrelate the samples $x(n), x(n-1), \dots, x(n-M)$, producing the uncorrelated samples $e_0^b(n), e_1^b(n), \dots, e_M^b(n)$ (innovations), which are then linearly combined (“recorrelated”) to obtain the optimum estimate of the desired response.

System functions. We next express the various lattice relations in terms of z -transforms. Taking the z -transform of (7.5.1) and (7.5.2), we obtain

$$E_m^f(z) = \left(1 + \sum_{k=1}^M a_k^{(m)*} z^{-k} \right) X(z) \triangleq A_m(z)X(z) \quad (7.5.9)$$

$$E_m^b(z) = \left(z^{-m} + \sum_{k=1}^M b_k^{(m)*} z^{-k+1} \right) X(z) \triangleq B_m(z)X(z) \quad (7.5.10)$$

where $A_m(z)$ and $B_m(z)$ are the system functions of the paths from the input to the outputs of the m th stage of the lattice. Using the symmetry relation $\mathbf{a}_m = \mathbf{J}\mathbf{b}_m^*$, $1 \leq m \leq M$, we obtain

$$B_m(z) = z^{-m} A_m^* \left(\frac{1}{z^*} \right) \quad (7.5.11)$$

Note that if z_0 is a zero of $A_m(z)$, then z_0^{-1} is a zero of $B_m(z)$. Therefore, if $A_m(z)$ is minimum-phase, then $B_m(z)$ is maximum-phase.

Taking the z -transform of the lattice equations (7.5.7), we have for the m th stage

$$E_m^f(z) = E_{m-1}^f(z) + k_{m-1}^* z^{-1} E_{m-1}^b(z) \quad (7.5.12)$$

$$E_m^b(z) = k_{m-1} E_{m-1}^f(z) + z^{-1} E_{m-1}^b(z) \quad (7.5.13)$$

Dividing both equations by $X(z)$ and using (7.5.9) and (7.5.10), we have

$$A_m(z) = A_{m-1}(z) + k_{m-1}^* z^{-1} B_{m-1}(z) \quad (7.5.14)$$

$$B_m(z) = k_{m-1} A_{m-1}(z) + z^{-1} B_{m-1}(z) \quad (7.5.15)$$

which, when initialized with

$$A_0(z) = B_0(z) = 1 \quad (7.5.16)$$

describe the lattice filter in the z domain.

The z -transform of the ladder-part (7.5.8) is given by

$$E_{m+1}(z) = E_m(z) - k_m^* E_m^b(z) \quad (7.5.17)$$

where $E_m(z)$ is the z -transform of the error sequence $e_m(n)$.

All-pole or “inverse” lattice structure. If we wish to recover the input $x(n)$ from the prediction error $e(n) = e_M^f(n)$, we can use the following all-pole lattice filter algorithm

$$\begin{aligned} e_M^f(n) &= e(n) \\ e_{m-1}^f(n) &= e_m^f(n) - k_{m-1}^* e_{m-1}^b(n-1) \quad m = M, M-1, \dots, 1 \\ e_m^b(n) &= e_{m-1}^b(n-1) + k_{m-1} e_{m-1}^f(n) \quad m = M, M-1, \dots, 1 \\ x(n) &= e_0^f(n) = e_0^b(n) \end{aligned} \quad (7.5.18)$$

which is derived as explained in Section 2.5 and is implemented by using the structure in Figure 7.6. Although the system functions of the all-zero lattice in (7.5.7) and the all-pole lattice in (7.5.18) are $H_{AZ}(z) = A(z)$ and $H_{AP}(z) = 1/A(z)$, the two lattice structures are described by the same set of lattice coefficients. The difference is the signal flow (see feedback loops in the all-pole structure). This structure is used in speech processing applications (Rabiner and Schafer 1978).

7.5.2 Some Properties and Interpretations

Lattice filters have some important properties and interesting interpretations that make them a useful tool in optimum filtering and signal modeling.

Optimal nesting. The all-zero lattice filter has an *optimal nesting* property when it is used for the implementation of an FLP. Indeed, if we use the lattice parameters obtained via the algorithm of Levinson-Durbin, the all-zero lattice filter driven by the signal $x(n)$

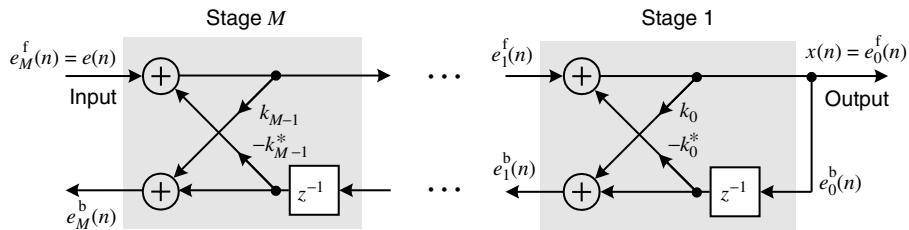


FIGURE 7.6

All-pole lattice structure for recovering the input signal from the forward prediction error.

produces prediction errors $e_m^f(n)$ and $e_m^b(n)$ at the output of the m th stage for all $1 \leq m \leq M$. This implies that we can increase the order of the filter by attaching additional stages without destroying the optimality of the previous stages. In contrast, the direct-form filter structure implementation requires the computation of the entire predictor for each stage. However, the nesting property does not hold for the all-pole lattice filter because of the feedback path.

Orthogonality. The backward prediction errors $e_m^b(n)$ for $0 \leq m \leq M$ are uncorrelated (see Section 7.2), that is,

$$E\{e_m^b(n)e_k^{b*}(n)\} = \begin{cases} P_m & k = m \\ 0 & k \neq m \end{cases} \quad (7.5.19)$$

and constitute the innovations representation of the input samples $x(n), x(n-1), \dots, x(n-m)$. We see that at a given time instant n , the backward prediction errors for orders $m = 0, 1, 2, \dots, M$ are uncorrelated and are part of a nonstationary sequence because the variance $E|e_m^b(n)|^2 = P_m$ depends on n . This should be expected because, for a given n , each $e_m^b(n)$ is computed using a different set of predictor coefficients. In contrast, for a given m , the sequence $e_m^b(n)$ is stationary for $-\infty < n < \infty$.

Reflection coefficients. The all-pole lattice structure is very useful in the modeling of layered media, where each stage of the lattice models one layer or section of the medium. Traveling waves in geophysical layers, in acoustic tubes of varying cross-sections, and in multisectional transmission lines have been modeled in this fashion. The modeling is performed such that the wave travel time through each section is the same, but the sections may have different impedances. The m th section is modeled with the signals $e_m^f(n)$ and $e_m^b(n)$ representing the forward and backward traveling waves, respectively.

If Z_m and Z_{m-1} are the characteristic impedances at sections m and $m-1$, respectively, then k_m represents the reflection coefficients between the two sections, given by

$$k_m = \frac{Z_m - Z_{m-1}}{Z_m + Z_{m-1}} \quad (7.5.20)$$

For this reason, the lattice parameters k_m are often known as *reflection coefficients*. As reflection coefficients, it makes good sense that their magnitudes not exceed unity. The termination of the lattice assumes a perfect reflection, and so the reflected wave $e_0^b(n)$ is equal to the transmitted wave $e_0^f(n)$. The result of this specific termination is an overall all-pole model (Rabiner and Schafer 1978).

Partial correlation coefficients. The *partial correlation coefficient (PCC)* between $x(n)$ and $x(n-m-1)$ (see also Section 7.2.2) is defined as the correlation coefficient

between $e_m^f(n)$ and $e_m^b(n-1)$, that is,

$$\text{PCC}\{x(n-m-1); x(n)\} \triangleq \frac{\text{PARCOR}\{x(n-m-1); x(n)\}}{\sqrt{E\{|e_m^b(n-1)|^2\}E\{|e_m^f(n)|^2\}}} \quad (7.5.21)$$

and, therefore, it takes values in the range $[-1, 1]$ (Kendall and Stuart 1979).

Working as in Section 7.2, we can show that

$$E\{e_m^b(n-1)e_m^{f*}(n)\} = \mathbf{b}_m^H \mathbf{r}_m + r(m+1) = \beta_m \quad (7.5.22)$$

which in conjunction with

$$E\{|e_m^b(n-1)|^2\} = E\{|e_m^f(n)|^2\} = P_m \quad (7.5.23)$$

and (7.4.16), results in

$$k_m = -\frac{\beta_m}{P_m} = -\text{PCC}\{x(n-m-1); x(n)\} \quad (7.5.24)$$

That is, for stationary processes the lattice parameters are the negative of the partial autocorrelation sequence and satisfy the relation

$$|k_m| \leq 1 \quad \text{for all } 0 \leq m \leq M-1 \quad (7.5.25)$$

derived also for (7.4.25) using an alternate approach.

Minimum phase. According to Theorem 2.3 (Section 2.5), the roots of the polynomial $A(z)$ are inside the unit circle if and only if

$$|k_m| < 1 \quad \text{for all } 0 \leq m \leq M-1 \quad (7.5.26)$$

which implies that the filters with system functions $A(z)$ and $1/A(z)$ are minimum-phase. The strict inequalities (7.5.26) are satisfied if the stationary process $x(n)$ is nonpredictable, which is the case when the Toeplitz autocorrelation matrix \mathbf{R} is positive definite.

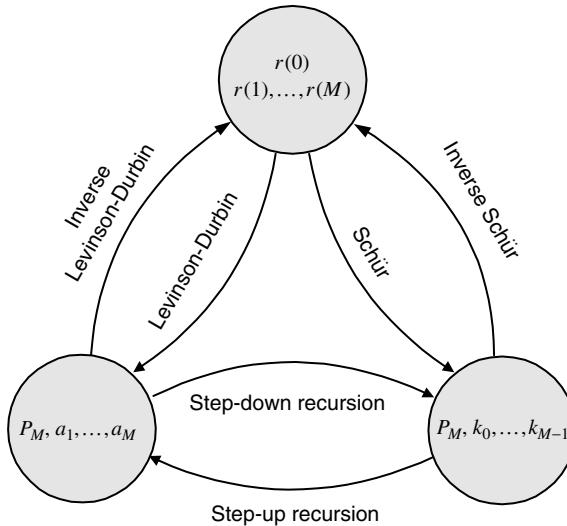
Lattice-ladder optimization. As we saw in Section 2.5, the output of an FIR lattice filter is a nonlinear function of the lattice parameters. Hence, if we try to design an optimum lattice filter by minimizing the MSE with respect to the lattice parameters, we end up with a nonlinear optimization problem (see Problem 7.11). In contrast, the Levinson algorithm leads to a lattice-ladder realization of the optimum filter through the order-recursive solution of a linear optimization problem. This subject is of interest to signal modeling and adaptive filtering (see Chapters 9 and 10).

7.5.3 Parameter Conversions

We have shown that the M th-order forward linear predictor of a stationary process $x(n)$ is uniquely specified by a set of linear equations in terms of the autocorrelation sequence and the prediction error filter is minimum-phase. Furthermore, it can be implemented using either a direct-form structure with coefficients $a_1^{(M)}, a_2^{(M)}, \dots, a_M^{(M)}$ or a lattice structure with parameters k_1, k_2, \dots, k_M . Next we show how to convert between the following equivalent representations of a linear predictor:

1. Direct-form filter structure: $\{P_M, a_1, a_2, \dots, a_M\}$.
2. Lattice filter structure: $\{P_M, k_0, k_1, \dots, k_{M-1}\}$.
3. Autocorrelation sequence: $\{r(0), r(1), \dots, r(M)\}$.

The transformation between the above representations is performed using the algorithms shown in Figure 7.7.

**FIGURE 7.7**

Equivalent representations for minimum-phase linear prediction error filters.

Lattice-to-direct (step-up) recursion. Given the lattice parameters k_1, k_2, \dots, k_M and the MMSE error P_M , we can compute the forward predictor \mathbf{a}_M by using the following recursions

$$\mathbf{a}_m = \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{J} \mathbf{a}_{m-1}^* \\ 1 \end{bmatrix} k_{m-1} \quad (7.5.27)$$

$$P_m = P_{m-1}(1 - |k_{m-1}|^2) \quad (7.5.28)$$

for $m = 1, 2, \dots, M$. This conversion is implemented by the function `[a, PM]=stepup(k)`.

Direct-to-lattice (step-down) recursion. Using the partitioning

$$\begin{aligned} \bar{\mathbf{a}}_m &= [a_1^{(m)} \ a_2^{(m)} \ \cdots \ a_{m-1}^{(m)}]^T \\ k_{m-1} &= a_m^{(m)} \end{aligned} \quad (7.5.29)$$

we can write recursion (7.5.27) as

$$\bar{\mathbf{a}}_m = \mathbf{a}_{m-1} + \mathbf{J} \bar{\mathbf{a}}_{m-1}^* k_{m-1}$$

or by taking the complex conjugate and multiplying both sides by \mathbf{J}

$$\mathbf{J} \bar{\mathbf{a}}_m^* = \mathbf{J} \mathbf{a}_{m-1}^* + \mathbf{a}_{m-1} k_{m-1}^*$$

Eliminating $\mathbf{J} \mathbf{a}_{m-1}^*$ from the last two equations and solving for \mathbf{a}_{m-1} , we obtain

$$\mathbf{a}_{m-1} = \frac{\bar{\mathbf{a}}_m - \mathbf{J} \bar{\mathbf{a}}_m^* k_{m-1}}{1 - |k_{m-1}|^2} \quad (7.5.30)$$

From (7.5.28), we have

$$P_{m-1} = \frac{P_m}{1 - |k_{m-1}|^2} \quad (7.5.31)$$

Given \mathbf{a}_M and P_M , we can obtain k_m and P_m for $0 \leq m \leq M-1$ by computing the last two recursions for $m = M, M-1, \dots, 2$. We should stress that both recursions break down if $|k_m| = \pm 1$. The step-down algorithm is implemented by the function `[k]=stepdown(a)`.

EXAMPLE 7.5.1. Given the third-order FLP coefficients $a_1^{(3)}, a_2^{(3)}, a_3^{(3)}$, compute the lattice parameters k_0, k_1, k_2 .

Solution. With the help of (7.5.29) the vector relation (7.5.30) can be written in scalar form as

$$k_{m-1} = a_m^{(m)} \quad (7.5.32)$$

and

$$a_i^{(m-1)} = \frac{a_i^{(m)} - a_{m-i}^{(m)*} k_{m-1}}{1 - |k_{m-1}|^2} \quad (7.5.33)$$

which can be used to implement the step-down algorithm for $m = M, M-1, \dots, 2$ and $i = 1, 2, \dots, m-1$. Starting with $m = 3$ and $i = 1, 2$, we have

$$k_2 = a_3^{(3)} \quad a_1^{(2)} = \frac{a_1^{(3)} - a_2^{(3)*} k_2}{1 - |k_2|^2} \quad a_2^{(2)} = \frac{a_2^{(3)} - a_1^{(3)*} k_2}{1 - |k_2|^2}$$

Similarly, for $m = 2$ and $i = 1$, we obtain

$$k_1 = a_2^{(2)} \quad a_1^{(1)} = \frac{a_1^{(2)} - a_1^{(2)*} k_1}{1 - |k_1|^2} = k_0$$

which completes the solution.

The step-up and step-down recursions also can be expressed in polynomial form as

$$A_m(z) = A_{m-1}(z) + k_{m-1}^* z^{-m} A_{m-1}^*(\frac{1}{z^*}) \quad (7.5.34)$$

and

$$A_{m-1}(z) = \frac{A_m(z) - k_{m-1}^* z^{-m} A_m^*(1/z^*)}{1 - |k_{m-1}|^2} \quad (7.5.35)$$

respectively.

Lattice parameters to autocorrelation. If we know the lattice parameters k_1, k_2, \dots, k_M and P_M , we can compute the values $r(0), r(1), \dots, r(M)$ of the autocorrelation sequence using the formula

$$r(m+1) = -k_m^* P_m - \mathbf{a}_m^H \mathbf{J} \mathbf{r}_m \quad (7.5.36)$$

which follows from (7.4.16) and (7.4.17), in conjunction with (7.5.27) and (7.4.21) for $m = 1, 2, \dots, M$. Equation (7.5.36) is obtained by eliminating β_m from (7.4.9) and (7.4.10). This algorithm is used by the function `r=k2r(k, PM)`. Another algorithm that computes the autocorrelation sequence from the lattice coefficients and does not require the intermediate computation of \mathbf{a}_m is provided in Section 7.6.

EXAMPLE 7.5.2. Given P_0, k_0, k_1 , and k_2 , compute the autocorrelation values $r(0), r(1), r(2)$, and $r(3)$.

Solution. Using $r(0) = P_0$ and

$$r(m+1) = -k_m^* P_m - \mathbf{a}_m^H \mathbf{J} \mathbf{r}_m$$

for $m = 0$, we have

$$r(1) = -k_0^* P_0$$

For $m = 1$

$$r(2) = -k_1^* P_1 - a_1^{(1)*} r(1)$$

where

$$P_1 = P_0(1 - |k_0|^2)$$

Finally, for $m = 2$ we obtain

$$r(3) = -k_2^* P_2 - [a_1^{(2)*} r(2) + k_1^* r(1)]$$

where

$$P_2 = P_1(1 - |k_1|^2)$$

and

$$a_1^{(2)} = a_1^{(1)} + a_1^{(1)*} k_1 = k_0 + k_0^* k_1$$

from the Levinson recursion.

Direct parameters to autocorrelation. Given \mathbf{a}_M and P_M , we can compute the autocorrelation sequence $r(0), r(1), \dots, r(M)$ by using (7.5.29) through (7.5.36). This method is known as the *inverse Levinson algorithm* and is implemented by the function `r=a2r(a, PM)`.

7.6 ALGORITHM OF SCHÜR

The algorithm of Schür is an order-recursive procedure for the computation of the lattice parameters k_1, k_2, \dots, k_M of the optimum forward predictor from the autocorrelation sequence $r(0), r(1), \dots, r(M)$ without computing the direct-form coefficients $\mathbf{a}_m, m = 1, 2, \dots, M$. The reverse process is known as the *inverse Schür algorithm*. The algorithm also can be extended to compute the ladder parameters of the optimum filter and the LDL^H decomposition of a Toeplitz matrix. The algorithm has its roots in the original work of Schür (Schür 1917), who developed a procedure to test whether a polynomial is analytic and bounded in the unit disk.

7.6.1 Direct Schür Algorithm

We start by defining the cross-correlation sequences between $e_m^f(n)$, $e_m^b(n)$, and $x(n)$

$$\xi_m^f(l) \triangleq E\{x(n-l)e_m^{f*}(n)\} \quad \text{with } \xi_m^f(l) = 0, \text{ for } 1 \leq l \leq m \quad (7.6.1)$$

$$\xi_m^b(l) \triangleq E\{x(n-l)e_m^{b*}(n)\} \quad \text{with } \xi_m^b(l) = 0, \text{ for } 0 \leq l < m \quad (7.6.2)$$

which are also known as *gapped functions* because of the regions of zeros created by the orthogonality principle (Robinson and Treitel 1980).

Multiplying the direct-form equations (7.5.1) and (7.5.2) by $x^*(n-l)$ and taking the mathematical expectation of both sides, we obtain

$$\xi_m^f(l) = r(l) + \mathbf{a}_m^H \tilde{\mathbf{r}}_m(l-1) \quad (7.6.3)$$

and

$$\xi_m^b(l) = r(l-m) + \mathbf{b}_m^H \tilde{\mathbf{r}}_m(l) \quad (7.6.4)$$

where

$$\tilde{\mathbf{r}}_m(l) \triangleq [r(l) \ r(l-1) \ \dots \ r(l-m+1)]^T \quad (7.6.5)$$

We notice that $\xi_m^f(l)$ and $\xi_m^b(l)$ can be interpreted as forward and backward autocorrelation prediction errors, because they occur when we feed the sequence $r(0), r(1), \dots, r(m+1)$ through the optimum predictors \mathbf{a}_m and \mathbf{b}_m of the process $x(n)$. Using the property $\mathbf{b}_m = \mathbf{J}\mathbf{a}_m^*$, we can show that (see Problem 7.29)

$$\xi_m^b(l) = \xi_m^{f*}(m-l) \quad (7.6.6)$$

If we set $l = m+1$ in (7.6.3) and $l = m$ in (7.6.4), and notice that $\tilde{\mathbf{r}}_m(m) = \mathbf{J}\mathbf{r}_m^*$, then we have

$$\xi_m^f(m+1) = r(m+1) + \mathbf{a}_m^H \mathbf{J}\mathbf{r}_m^* = \beta_m^* \quad (7.6.7)$$

and

$$\xi_m^b(m) = r(0) + \mathbf{r}_m^H \mathbf{J}\mathbf{b}_m = P_m \quad (7.6.8)$$

respectively. Therefore, we have

$$k_m = -\frac{\beta_m}{P_m} = -\frac{\xi_m^f(m+1)}{\xi_m^b(m)} \quad (7.6.9)$$

that is, we can compute k_{m+1} in terms of $\xi_m^f(l)$ and $\xi_m^b(l)$.

Multiplying the lattice recursions (7.5.7) by $x^*(n-l)$ and taking the mathematical expectation of both sides, we obtain

$$\begin{aligned} \xi_0^f(l) &= \xi_0^b(l) = r(l) \\ \xi_m^f(l) &= \xi_{m-1}^f(l) + k_{m-1}^* \xi_{m-1}^b(l-1) \quad m = 1, 2, \dots, M \\ \xi_m^b(l) &= k_{m-1} \xi_{m-1}^f(l) + \xi_{m-1}^b(l-1) \quad m = 1, 2, \dots, M \end{aligned} \quad (7.6.10)$$

which provides a lattice structure for the computation of the cross-correlations $\xi_m^f(l)$ and $\xi_m^b(l)$. In contrast, (7.6.7) and (7.6.8) provide a computation using a direct-form structure.

In the next example we illustrate how to use the lattice structure (7.6.10) to compute the lattice parameters k_1, k_2, \dots, k_M from the autocorrelation sequence $r(0), r(1), \dots, r(M)$ without the intermediate explicit computation of the predictor coefficients \mathbf{a}_m .

EXAMPLE 7.6.1. Use the algorithm of Schür to compute the lattice parameters $\{k_0, k_1, k_2\}$ and the MMSE P_3 from the autocorrelation sequence coefficients

$$r(0) = 3 \quad r(1) = 2 \quad r(2) = 1 \quad r(3) = \frac{1}{2}$$

Solution. Starting with (7.6.9) for $m = 0$, we have

$$k_0 = -\frac{\xi_0^f(1)}{\xi_0^b(0)} = -\frac{r(1)}{r(0)} = -\frac{2}{3}$$

because $\xi_0^f(l) = \xi_0^b(l) = r(l)$. To compute k_1 , we need $\xi_1^f(2)$ and $\xi_1^b(1)$, which are obtained from (7.6.10) by setting $l = 2$. Indeed, we have

$$\xi_1^f(2) = \xi_0^f(2) + k_0 \xi_0^b(1) = 1 + (-\frac{2}{3})2 = -\frac{1}{3}$$

$$\xi_1^b(1) = \xi_0^b(0) + k_0 \xi_0^f(1) = 3 + (-\frac{2}{3})2 = \frac{5}{3} = P_1$$

and

$$k_1 = -\frac{\xi_1^f(2)}{\xi_1^b(1)} = -\frac{-\frac{1}{3}}{\frac{5}{3}} = \frac{1}{5}$$

The computation of k_2 requires $\xi_2^f(3)$ and $\xi_2^b(2)$, which in turn need $\xi_1^f(3)$ and $\xi_1^b(2)$. These quantities are computed by

$$\xi_1^f(3) = \xi_0^f(3) + k_0 \xi_0^b(2) = \frac{1}{2} + (-\frac{2}{3})1 = -\frac{1}{6}$$

$$\xi_1^b(2) = \xi_0^b(1) + k_0 \xi_0^f(2) = 2 + (-\frac{2}{3})1 = \frac{4}{3}$$

$$\xi_2^f(3) = \xi_1^f(3) + k_1 \xi_1^b(2) = -\frac{1}{6} + \frac{1}{5} \cdot \frac{4}{3} = \frac{1}{10}$$

$$\xi_2^b(2) = \xi_1^b(1) + k_1 \xi_1^f(2) = \frac{4}{3} + \frac{1}{5}(-\frac{1}{6}) = \frac{8}{5} = P_2$$

and the lattice coefficient is

$$k_2 = -\frac{\xi_2^f(3)}{\xi_2^b(2)} = -\frac{\frac{1}{10}}{\frac{8}{5}} = -\frac{1}{16}$$

The final MMSE is computed by

$$P_3 = P_2(1 - |k_2|^2) = \frac{8}{5}(1 - \frac{1}{256}) = \frac{51}{32}$$

although we could use the formula $\xi_m^b(m) = P_m$ as well. Therefore the lattice coefficients and the MMSE are found to be

$$k_0 = -\frac{2}{3} \quad k_1 = \frac{1}{5} \quad k_2 = -\frac{1}{16} \quad P_3 = \frac{51}{32}$$

It is worthwhile to notice that the k_m parameters can be obtained by “feeding” the sequence $r(0), r(1), \dots, r(M)$ through the lattice filter as a signal and switching on the stages one by one after computing the required lattice coefficient. The value of k_m is computed at time $n = m$ from the inputs to stage m (see Problem 7.30).

The procedure outlined in the above example is known as the algorithm of Schür and has good numerical properties because the quantities used in the lattice structure (7.6.10) are bounded. Indeed, from (7.6.1) and (7.6.2) we have

$$|\xi_m^f(l)|^2 \leq |E\{|e_m^f(n)|^2\}| |E\{|x(n-l)|^2\}| \leq P_m r(0) \leq r^2(0) \quad (7.6.11)$$

$$|\xi_m^b(l)|^2 \leq |E\{|e_m^b(n)|^2\}| |E\{|x(n-l)|^2\}| \leq P_m r(0) \leq r^2(0) \quad (7.6.12)$$

because $P_m \leq P_0 = r(0)$. As a result of this fixed dynamic range, the algorithm of Schür can be easily implemented with fixed-point arithmetic. The numeric stability of the Schür algorithm provided the motivation for its use in speech processing applications (LeRoux and Gueguen 1977).

7.6.2 Implementation Considerations

Figure 7.8 clarifies the computational steps in Example 7.4.2, using three decomposition trees that indicate the quantities needed to compute k_0, k_1 , and k_2 when we use the lattice recursions (7.6.10) for real-valued signals. We can easily see that the computations for k_0 are part of those for k_1 , which in turn are part of the computations for k_2 . Thus, the tree for k_2 includes also the quantities needed to compute k_0 and k_1 . The computations required to compute k_0, k_1, k_2 , and k_3 are

$$\begin{array}{ll}
 1. \quad k_0 = -\frac{\xi_0^f(1)}{\xi_0^b(0)} & 9. \quad \xi_2^f(4) = \xi_1^f(4) + k_1 \xi_1^b(3) \\
 2. \quad \xi_1^f(4) = \xi_0^f(4) + k_0 \xi_0^b(3) & 10. \quad \xi_2^b(3) = \xi_1^b(2) + k_1 \xi_1^f(3) \\
 3. \quad \xi_1^b(3) = \xi_0^b(2) + k_0 \xi_0^f(3) & 11. \quad \xi_2^f(3) = \xi_1^f(3) + k_1 \xi_1^b(2) \\
 4. \quad \xi_1^f(3) = \xi_0^f(3) + k_0 \xi_0^b(2) & 12. \quad \xi_2^b(2) = \xi_1^b(1) + k_1 \xi_1^f(2) \\
 5. \quad \xi_1^b(2) = \xi_0^b(1) + k_0 \xi_0^f(2) & 13. \quad k_2 = -\frac{\xi_2^f(3)}{\xi_2^b(2)} \\
 6. \quad \xi_1^f(2) = \xi_0^f(2) + k_0 \xi_0^b(1) & 14. \quad \xi_3^f(4) = \xi_2^f(4) + k_2 \xi_2^b(3) \\
 7. \quad \xi_1^b(1) = \xi_0^b(0) + k_0 \xi_0^f(1) & 15. \quad \xi_3^b(3) = \xi_2^b(2) + k_2 \xi_2^f(3) \\
 8. \quad k_1 = -\frac{\xi_1^f(2)}{\xi_1^b(1)} & 16. \quad k_3 = -\frac{\xi_3^f(4)}{\xi_3^b(3)}
 \end{array}$$

With the help of the corresponding tree decomposition diagram, this can be arranged as shown in Figure 7.9. The obtained computational structure was named the *superlattice* because it consists of a triangular array of latticelike stages (Carayannis et al. 1985). Note that the superlattice has no redundancy and is characterized by local interconnections; that is, the quantities needed at any given node are available from the immediate neighbors.

The two-dimensional layout of the superlattice suggests various algorithms to perform the computations.

1. *Parallel algorithm.* We first note that all equations involving the coefficient k_m constitute one stage of the superlattice and can be computed in parallel after the computation of k_m because all inputs to the current stage are available from the previous one. This algorithm can be implemented by $2(M - 1)$ processors in $M - 1$ “parallel” steps (Kung and Hu 1983). Since each step involves one division to compute k_m and then $2(M - m)$ multiplications and additions for the parallel computations, the number of utilized processors decreases from $2(M - 1)$ to 1. The algorithm is not order-recursive because the order M must be known before the superlattice structure is set up.
2. *Sequential algorithm.* A sequential implementation of the parallel algorithm is essentially equivalent to the version introduced for speech processing applications (LeRoux and Gueguen 1977). This algorithm, which is implemented by the function `k=schur1g(r,M)` and summarized in Table 7.4, starts with Equation (1) and computes sequentially Equations (2), (3), etc.

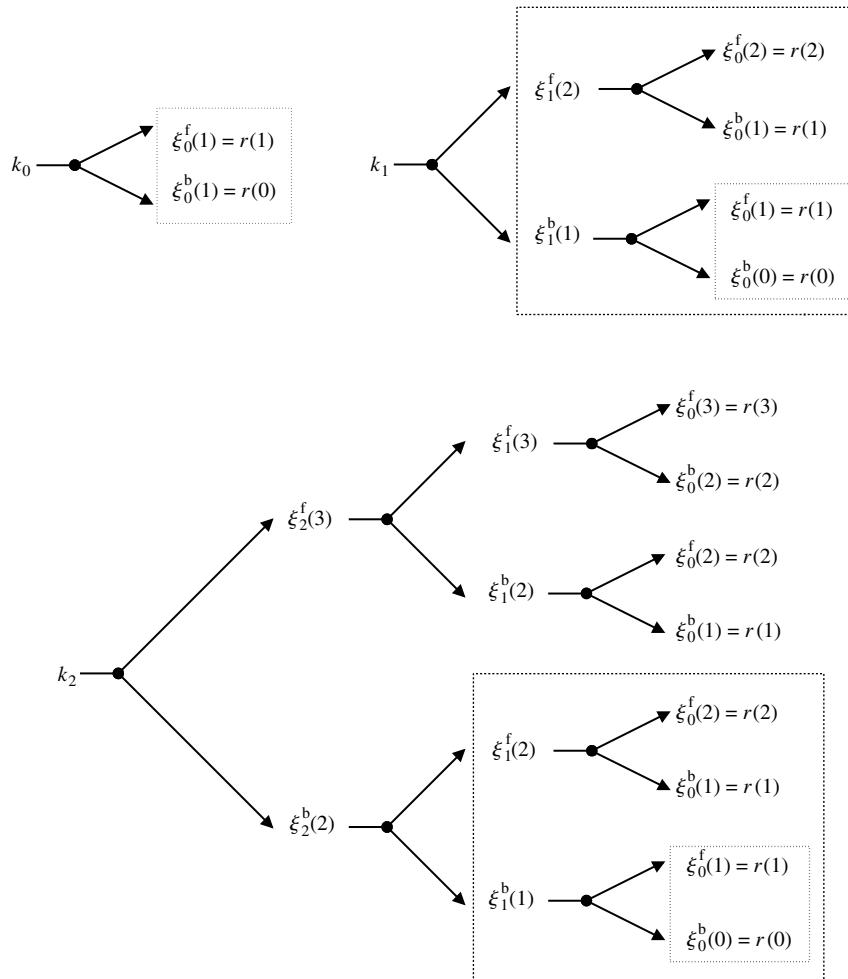
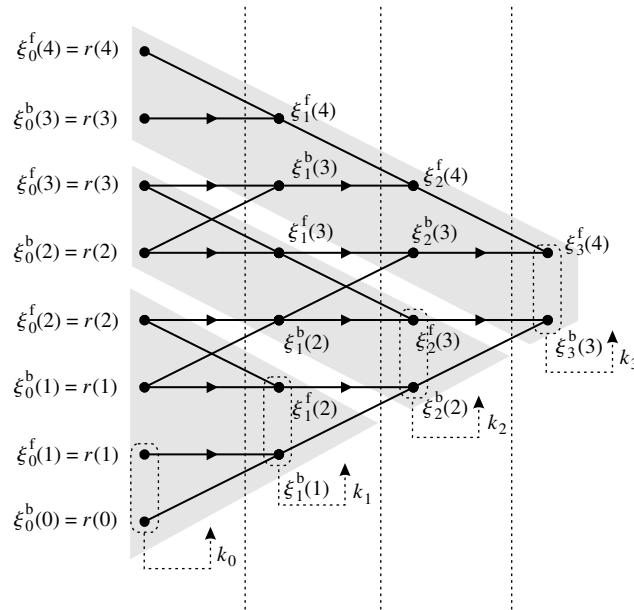


FIGURE 7.8
Tree decomposition for the computations required by the algorithm of Schür.

3. *Sequential order-recursive algorithm.* The parallel algorithm starts at the left of the superlattice and performs the computations within the vertical strips in parallel. Clearly, the order M should be fixed before we start, and the algorithm is not order-recursive. Careful inspection of the superlattice reveals that we can obtain an order-recursive algorithm by organizing the computations in terms of the *slanted* shadowed strips shown in Figure 7.9. Indeed, we start with k_0 and then perform the computations in the first slanted strip to determine the quantities $\xi_1^f(2)$ and $\xi_1^b(1)$ needed to compute k_1 . We proceed with the next slanted strip, compute k_2 , and conclude with the computation of the last strip and k_3 . The computations within each slanted strip are performed sequentially.
4. *Partitioned-parallel algorithm.* Suppose that we have P processors with $P < M$. This algorithm partitions the superlattice into groups of P consecutive slanted strips (partitions) and performs the computations of each partition, in parallel, using the P processors. It turns out that by storing some intermediate quantities, we have everything needed by the superlattice to compute all the partitions, one at a time (Koukoutsis et al. 1991). This algorithm provides a very convenient scheme for the implementation of the superlattice using multiprocessing (see Problem 7.31).

**FIGURE 7.9**

Superlattice structure organization of the algorithm of Schür.
The input is the autocorrelation sequence and the output the
lattice parameters.

TABLE 7.4
Summary of the algorithm of Schür.

-
1. **Input:** $\{r(l)\}_0^M$
 2. **Initialization**
 - (a) For $l = 0, 1, \dots, M$
 - (b) $\xi_0^f(l) = \xi_0^b(l) = r(l)$
 - (c) $k_0 = -\frac{\xi_0^f(1)}{\xi_0^b(0)}$
 - (d) $P_1 = r(0)(1 - |k_1|^2)$
 3. **For** $m = 1, 2, \dots, M - 1$
 - (a) **For** $l = m, m + 1, \dots, M$
 - (b) $\xi_m^f(l) = \xi_{m-1}^f(l) + k_{m-1}^* \xi_{m-1}^b(l-1)$
 - (c) $\xi_m^b(l) = k_{m-1} \xi_{m-1}^f(l) + \xi_{m-1}^b(l-1)$
 - (d) $k_m = -\frac{\xi_m^f(m+1)}{\xi_m^b(m)}$
 - (e) $P_{m+1} = P_m(1 - |k_m|^2)$
 4. **Output:** $\{k_m\}_0^{M-1}, \{P_m\}_1^M$
-

Extended Schür algorithm. To extend the Schür algorithm for the computation of the ladder parameters k_m^c , we define the cross-correlation sequence

$$\xi_m^c(l) \triangleq E\{x(n-l)e_m^*(l)\} \quad \text{with } \xi_m^c(l) = 0, \text{ for } 0 \leq l < m \quad (7.6.13)$$

due to the orthogonality principle. Multiplying (7.5.8) by $x^*(n-l)$ and taking the mathematical expectation, we obtain a direct form

$$\xi_m^c(l) = d_{l+1} - \mathbf{c}_m^T \tilde{\mathbf{r}}_m(l) \quad (7.6.14)$$

For $l = m$, we have

$$\xi_m^c(l) = \xi_m^c(l) - k_m^{c*} \xi_m^b(l) \quad (7.6.15)$$

For $l = m$, we have

$$\xi_m^c(m) = d_{l+1} - \mathbf{c}_m^H \mathbf{J} \mathbf{r}_m = \beta_m^c \quad (7.6.16)$$

and

$$k_m^c = \frac{\beta_m^c}{P_m} = \frac{\xi_m^c(m)}{\xi_m^b(m)} \quad (7.6.17)$$

that is, we can compute the sequence k_m^c using a lattice-ladder structure.

The computations can be arranged in the form of a *superladder* structure, shown in Figure 7.10 (Koukoutsis et al. 1991). See also Problem 7.32. In turn, (7.6.17) can be used in conjunction with the superlattice to determine the lattice-ladder parameters of the optimum FIR filter. The superladder structure is illustrated in the following example.

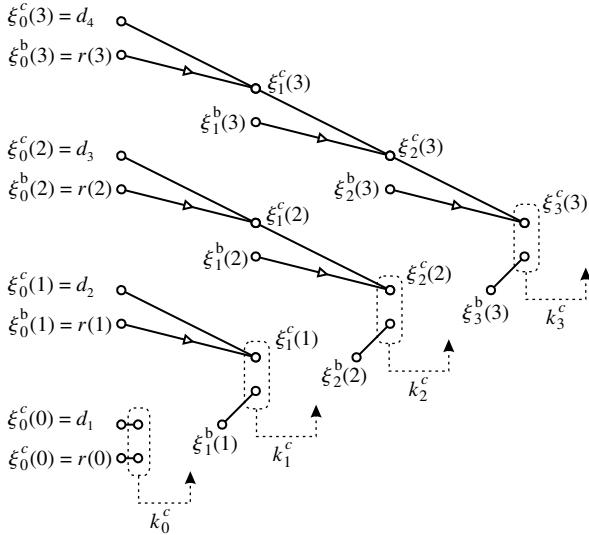


FIGURE 7.10
Graphical illustration of the superladder structure.

EXAMPLE 7.6.2. Determine the lattice-ladder parameters of an optimum FIR filter with input autocorrelation sequence given in Example 7.6.1 and cross-correlation sequence $d_1 = 1$, $d_2 = 2$, and $d_3 = \frac{5}{2}$, using the extended Schür algorithm.

Solution. Since the lattice parameters were obtained in Example 7.6.1, we only need to find the ladder parameters. Hence, using (7.6.15), (7.6.17), and the values of $\xi_m^b(l)$ computed in Example 7.6.1, we have

$$\begin{aligned} k_0^c &= -\frac{\xi_0^c(0)}{\xi_0^b(0)} = -\frac{d_1}{r(0)} = -\frac{1}{3} \\ \xi_1^c(1) &= \xi_0^c(1) + k_0^c \xi_0^b(1) = 2 - \frac{1}{3}(2) = \frac{4}{3} \\ \xi_1^c(2) &= \xi_0^c(2) + k_0^c \xi_0^b(2) = \frac{5}{2} - \frac{1}{3}(1) = \frac{13}{6} \\ k_1^c &= -\frac{\xi_1^c(1)}{\xi_1^b(1)} = -\frac{\frac{4}{3}}{\frac{5}{3}} = -\frac{4}{5} \\ \xi_2^c(2) &= \xi_1^c(2) + k_1^c \xi_1^b(2) = \frac{13}{6} - \frac{4}{5}(\frac{4}{3}) = \frac{11}{10} \\ k_2^c &= -\frac{\xi_2^c(2)}{\xi_2^b(2)} = -\frac{\frac{11}{10}}{\frac{8}{5}} = -\frac{11}{16} \end{aligned}$$

which provide the values of the ladder parameters. These values are identical to those obtained in Example 7.4.3.

7.6.3 Inverse Schür Algorithm

The inverse Schür algorithm computes the autocorrelation sequence coefficients $r(0), r(1), \dots, r(m)$ from the lattice parameters k_0, k_1, \dots, k_M and the MMSE P_M of the linear predictor. The organization of computations is best illustrated by the following example.

EXAMPLE 7.6.3. Given the lattice filter coefficients

$$k_0 = -\frac{2}{3} \quad k_1 = \frac{1}{5} \quad k_2 = -\frac{1}{16}$$

and the MMSE $P_3 = 51/32$, compute the autocorrelation samples $r(0), r(1), r(2)$, and $r(3)$, using the inverse Schür algorithm.

Solution. We base our approach on the part of the superlattice structure shown in Figure 7.9 that is enclosed by the nodes $\xi_0^b(0), \xi_0^f(3), \xi_2^f(3)$, and $\xi_2^b(2)$. To start at the lower left corner, we compute $r(0)$, using (7.4.30):

$$r(0) = \frac{P_3}{\prod_{m=0}^2 (1 - k_m^2)} = \frac{\frac{51}{32}}{(1 - \frac{4}{9})(1 - \frac{1}{25})(1 - \frac{1}{256})} = 3$$

This also follows from (7.5.31). Then, continuing the computations from the line defined by $r(0)$ and $\xi_2^b(2)$ to the node defined by $\xi_0^f(3) = r(3)$, we have

$$\begin{aligned} r(1) &= -k_0 r(0) = -(-\frac{2}{3})3 = 2 \\ \xi_1^b(1) &= \xi_0^b(0) + k_0 \xi_0^f(1) = 3 + (-\frac{2}{3})2 = \frac{5}{3} \\ \xi_1^f(2) &= -k_1 \xi_1^b(1) = -\frac{1}{5}(\frac{5}{3}) = -\frac{1}{3} \\ r(2) &= \xi_0^f(2) = \xi_1^f(2) - k_0 \xi_0^b(1) = -\frac{1}{3} - (-\frac{2}{3})2 = 1 \\ \xi_1^b(2) &= \xi_0^b(1) + k_0 \xi_0^f(2) = 2 + (-\frac{2}{3})1 = \frac{4}{3} \\ \xi_2^b(2) &= \xi_1^b(1) + k_1 \xi_1^f(2) = \frac{5}{3} + \frac{1}{5}(-\frac{1}{3}) = \frac{8}{5} \\ \xi_2^f(3) &= -k_2 \xi_2^b(2) = -(-\frac{1}{16})(\frac{8}{5}) = \frac{1}{10} \\ \xi_1^f(3) &= \xi_2^f(3) - k_1 \xi_1^b(2) = \frac{1}{10} - \frac{1}{5}(\frac{4}{3}) = -\frac{1}{6} \\ r(3) &= \xi_0^f(3) = \xi_1^f(3) - k_0 \xi_0^b(2) = -\frac{1}{6} - (-\frac{2}{3})1 = \frac{1}{2} \end{aligned}$$

as can be easily verified by the reader. Thus, the autocorrelation sequence is

$$r(0) = 3 \quad r(1) = 2 \quad r(2) = 1 \quad r(3) = \frac{1}{2}$$

which agree with the autocorrelation sequence coefficients used in Example 7.6.1 with the direct Schür algorithm.

The inverse Schür algorithm is implemented by the function `r=invschur(k, PM)`, which follows the same procedure as the previous example.

7.7 TRIANGULARIZATION AND INVERSION OF TOEPLITZ MATRICES

In this section, we develop LDL^H decompositions for both Toeplitz matrices and the inverse of Toeplitz matrices, followed by a recursion for the computation of the inverse of a Toeplitz matrix.

7.7.1 LDL^H Decomposition of Inverse of a Toeplitz Matrix

375

SECTION 7.7

Triangularization and
Inversion of Toeplitz
Matrices

Since \mathbf{R}_m is a Hermitian Toeplitz matrix that also happens to be persymmetric, that is, $\mathbf{JR}_m\mathbf{J} = \mathbf{R}_m^*$, taking its inverse, we obtain

$$\mathbf{JR}_m^{-1}\mathbf{J} = (\mathbf{R}_m^*)^{-1} \quad (7.7.1)$$

The last equation shows that the inverse of a Toeplitz matrix, although not Toeplitz, is persymmetric. From (7.1.58), we recall that the BLP coefficients and the MMSE P_m^b provide the quantities for the UDU^T decomposition of \mathbf{R}_{m+1}^{-1} , that is,

$$\mathbf{R}_{m+1}^{-1} = \mathbf{B}_{m+1}^H \mathbf{D}_{m+1}^{-1} \mathbf{B}_{m+1} \quad (7.7.2)$$

where
$$\mathbf{B}_{m+1} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ b_0^{(1)} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_0^{(m-1)} & b_1^{(m-1)} & \cdots & 1 & 0 \\ b_0^{(m)} & b_1^{(m)} & \cdots & b_{m-1}^{(m)} & 1 \end{bmatrix} \quad (7.7.3)$$

and

$$\mathbf{D}_{m+1} = \text{diag} \{P_0^b, P_1^b, \dots, P_m^b\} \quad (7.7.4)$$

For a Toeplitz matrix \mathbf{R}_{m+1} , we can obtain the LDL^H decomposition of its inverse by using (7.7.2) and the property $\mathbf{J} = \mathbf{J}^{-1}$ of the exchange matrix. Starting with (7.7.1), we obtain

$$(\mathbf{R}_{m+1}^*)^{-1} = \mathbf{JR}_{m+1}^{-1}\mathbf{J} = (\mathbf{JB}_{m+1}^H\mathbf{J})(\mathbf{JD}_{m+1}^{-1}\mathbf{J})(\mathbf{JB}_{m+1}\mathbf{J}) \quad (7.7.5)$$

If we define

$$\mathbf{A}_{m+1} \triangleq \mathbf{JB}_{m+1}^*\mathbf{J} \quad (7.7.6)$$

and

$$\bar{\mathbf{D}}_{m+1} \triangleq \mathbf{JD}_{m+1}^{-1}\mathbf{J} = \text{diag} \{P_m, P_{m-1}, \dots, P_0\} \quad (7.7.7)$$

then (7.7.2) gives

$$\mathbf{R}_{m+1}^{-1} = \mathbf{A}_{m+1}^H \bar{\mathbf{D}}_{m+1}^{-1} \mathbf{A}_{m+1} \quad (7.7.8)$$

which provides the unique LDL^H decomposition of the matrix \mathbf{R}_{m+1}^{-1} . Indeed, using the property $\mathbf{a}_j = \mathbf{Jb}_j^*$ for $1 \leq j \leq m$, or equivalently $a_i^{(j)*} = b_{j-i}^{(j)*}$, we can write matrix $\mathbf{A}_{m+1} = \mathbf{JB}_{m+1}^*\mathbf{J}$ as

$$\mathbf{A}_{m+1} = \begin{bmatrix} 1 & a_1^{(m)*} & a_2^{(m)*} & \cdots & a_m^{(m)*} \\ 0 & 1 & a_1^{(m-1)*} & \cdots & a_{m-1}^{(m-1)*} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \ddots & a_1^{(1)*} \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \quad (7.7.9)$$

which is an upper unit triangular matrix. We stress that the property $\mathbf{JB}_{m+1}^*\mathbf{J} = \mathbf{A}_{m+1}$ and the above derivation of (7.7.8) hold for Toeplitz matrices only. However, the decomposition in (7.7.2) holds for any Hermitian, positive definite matrix (see Section 7.1.4).

As we saw in Section 6.3, the solution of the normal equations $\mathbf{R}\mathbf{c} = \mathbf{d}$ can be obtained in three steps as

$$\mathbf{R} = \mathbf{LDL}^H \Rightarrow \mathbf{L}\mathbf{D}\mathbf{k}^c = \mathbf{d} \Rightarrow \mathbf{L}^H\mathbf{c} = \mathbf{k}^c \quad (7.7.10)$$

where the LDL^H decomposition requires about $M^3/6$ flops and the solution of each triangular system $M^2/2$ flops. Since $\mathbf{R}^{-1} = \mathbf{B}^H \mathbf{D}^{-1} \mathbf{B}$, the Levinson-Durbin algorithm performs the UDU^H decomposition of \mathbf{R}^{-1} when \mathbf{R} is Toeplitz, at a cost of M^2 flops; that is, it

reduces the computational complexity by an order of magnitude. The Levinson recursion for the optimum filter is equivalent to the solution of the two triangular systems and requires M^2 operations.

EXAMPLE 7.7.1. Compute the lattice-ladder parameters of an MMSE finite impulse response filter specified by the normal equations

$$\begin{bmatrix} 3 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} h(0) \\ h(1) \\ h(2) \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ \frac{5}{2} \end{bmatrix}$$

using two different approaches: the LDL^H decomposition and the algorithm of Levinson.

Solution. The LDL^H decomposition of \mathbf{R} is

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{2}{3} & 1 & 0 \\ \frac{1}{3} & \frac{4}{5} & 1 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & \frac{5}{3} & 0 \\ 0 & 0 & \frac{8}{5} \end{bmatrix} \quad \mathbf{L}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{2}{3} & 1 & 0 \\ \frac{1}{5} & -\frac{4}{5} & 1 \end{bmatrix}$$

and using (7.3.31), we have

$$\mathbf{k}_3^c = \mathbf{D}^{-1} \mathbf{L}^{-1} \mathbf{d} = \left[\frac{1}{3} \frac{4}{5} \frac{11}{16} \right]^T$$

which gives the three ladder parameters. The two lattice parameters are obtained by solving the system

$$\mathbf{L}^{[2]} \mathbf{D}^{[2]} \mathbf{k}_2 = \mathbf{r}_2^b \quad \text{with } \mathbf{r}_2^b = [1 \ 2]^T$$

which gives $k_0 = \frac{1}{3}$ and $k_1 = \frac{4}{5}$. The results agree with those obtained in Example 7.4.3 using the algorithm of Levinson. We also note that the rows of \mathbf{L}^{-1} provide the first- and second-order forward and backward linear predictors. This is the case because the matrix is Toeplitz. For symmetric matrices the LDL^H decomposition provides the backward predictors only.

7.7.2 LDL^H Decomposition of a Toeplitz Matrix

The computation of the LDL^H decomposition of a symmetric, positive definite matrix requires on the order of M^3 computations. In Section 7.1, we saw that the cross-correlation between $x(n)$ and $e_m^b(n)$ is related to the LDL^H decomposition of the correlation matrix \mathbf{R}_m . We next show that we can extend the Schür algorithm to compute the LDL^H decomposition of a Toeplitz matrix with $O(M^2)$ computations using the cross-correlations $\xi_m^b(l)$.

To illustrate the basic process, we note that evaluating the product on the left with the help of (7.6.4), we obtain

$$\begin{bmatrix} r(0) & r(1) & r(2) & r(3) \\ r(1) & r(0) & r(1) & r(2) \\ r(2) & r(1) & r(0) & r(1) \\ r(3) & r(2) & r(1) & r(0) \end{bmatrix} \begin{bmatrix} 1 & b_0^{(1)*} & b_0^{(2)*} & b_0^{(3)*} \\ 0 & 1 & b_1^{(2)*} & b_1^{(3)*} \\ 0 & 0 & 1 & b_2^{(3)*} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \xi_0^{b(0)} & 0 & 0 & 0 \\ \xi_0^{b(1)} & \xi_1^{b(1)} & 0 & 0 \\ \xi_0^{b(2)} & \xi_1^{b(2)} & \xi_2^{b(2)} & 0 \\ \xi_0^{b(3)} & \xi_1^{b(3)} & \xi_2^{b(3)} & \xi_3^{b(3)} \end{bmatrix}$$

that is, a lower triangular matrix $\tilde{\mathbf{L}}$, which can be written as

$$\tilde{\mathbf{L}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{\xi_0^{b(1)}}{P_0} & 1 & 0 & 0 \\ \frac{\xi_0^{b(2)}}{P_0} & \frac{\xi_1^{b(2)}}{P_1} & 1 & 0 \\ \frac{\xi_0^{b(3)}}{P_0} & \frac{\xi_1^{b(3)}}{P_1} & \frac{\xi_2^{b(3)}}{P_2} & 1 \end{bmatrix} \begin{bmatrix} P_0 & 0 & 0 & 0 \\ 0 & P_1 & 0 & 0 \\ 0 & 0 & P_2 & 0 \\ 0 & 0 & 0 & P_3 \end{bmatrix} = \mathbf{L} \mathbf{D}$$

because $P_m = \xi_m^b(m) \geq 0$. Therefore, $\mathbf{R}\mathbf{B}^H = \mathbf{L}\mathbf{D}$ and since \mathbf{R} is Hermitian, we have $\mathbf{R} = \mathbf{L}\mathbf{D}\mathbf{B}^{-H} = \mathbf{B}^{-1}\mathbf{D}\mathbf{L}^H$, which implies that $\mathbf{B}^{-1} = \mathbf{L}$. This results in the following \mathbf{LDL}^H factorization of the $(M+1) \times (M+1)$ symmetric Toeplitz matrix \mathbf{R}

$$\mathbf{R} = \mathbf{LDL}^H \quad (7.7.11)$$

where $\mathbf{L} = \mathbf{B}^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ \bar{\xi}_0^b(1) & 1 & \cdots & 0 & 0 \\ \bar{\xi}_0^b(2) & \bar{\xi}_1^b(2) & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \bar{\xi}_0^b(M) & \bar{\xi}_1^b(M) & \cdots & \bar{\xi}_{M-1}^b(M) & 1 \end{bmatrix} \quad (7.7.12)$

$$\bar{\xi}_m^b(l) = \frac{\xi_m^b(l)}{\xi_m^b(m)} = \frac{\xi_m^b(l)}{P_m} \quad (7.7.13)$$

and

$$\mathbf{D} = \text{diag } \{P_0, P_1, \dots, P_M\} \quad (7.7.14)$$

The basic recursion (7.6.10) in the algorithm of Schür can be extended to compute the elements of $\tilde{\mathbf{L}}$ and hence the \mathbf{LDL}^H factorization of the Toeplitz matrix \mathbf{R} (see Problem 7.33).

Since a Toeplitz matrix is persymmetric, that is, $\mathbf{JRJ} = \mathbf{R}^*$, we have

$$\mathbf{R} = \mathbf{JR}^*\mathbf{J} = (\mathbf{JL}^*\mathbf{J})(\mathbf{JD}\mathbf{J})(\mathbf{JL}^H\mathbf{J}) \triangleq \mathbf{UDU}^H \quad (7.7.15)$$

which provides the \mathbf{UDU}^H decomposition of \mathbf{R} . Notice that the relation $\mathbf{U} = \mathbf{JL}^*\mathbf{J}$ also can be obtained from $\mathbf{A} = \mathbf{JB}^*\mathbf{J}$ [see (7.4.11)], which in turn is a consequence of the symmetry between forward and backward prediction for stationary processes.

The validity of (7.6.10) also can be shown by computing the product

$$\mathbf{RA}^H = \begin{bmatrix} r(0) & r(1) & r(2) & r(3) \\ r(1) & r(0) & r(1) & r(2) \\ r(2) & r(1) & r(0) & r(1) \\ r(3) & r(2) & r(1) & r(0) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ a_1^{(3)*} & 1 & 0 & 0 \\ a_2^{(3)*} & a_1^{(2)*} & 1 & 0 \\ a_3^{(3)*} & a_2^{(2)*} & a_1^{(1)*} & 1 \end{bmatrix} \quad (7.7.16)$$

$$= \begin{bmatrix} \xi_3^f(0) & \xi_2^f(-1) & \xi_1^f(-2) & \xi_0^f(-3) \\ 0 & \xi_2^f(0) & \xi_1^f(-1) & \xi_0^f(-2) \\ 0 & 0 & \xi_1^f(0) & \xi_0^f(-1) \\ 0 & 0 & 0 & \xi_0^f(0) \end{bmatrix} \quad (7.7.17)$$

with the help of (7.6.3) and $r(-l) = r^*(l)$. The formula $\mathbf{U} = \mathbf{JL}^*\mathbf{J}$ relates $\xi_m^f(l)$ and $\xi_m^b(l)$, as expected by (7.6.10).

7.7.3 Inversion of Real Toeplitz Matrices

From the discussion in Section 7.1, it follows from (7.1.12) that the inverse \mathbf{Q}_M of a symmetric, positive definite matrix \mathbf{R}_M is given by

$$\mathbf{Q}_M \triangleq \begin{bmatrix} \mathbf{Q} & \mathbf{q} \\ \mathbf{q}^T & q \end{bmatrix} \quad (7.7.18)$$

with

$$\mathbf{q} = \frac{\mathbf{b}}{P} \quad (7.7.19)$$

$$q = \frac{1}{P} \quad (7.7.20)$$

and

$$\mathbf{Q} = \mathbf{R}^{-1} + \frac{1}{P} \mathbf{b} \mathbf{b}^T \quad (7.7.21)$$

as given by (7.1.18), (7.1.19), and (7.1.21). The matrix \mathbf{Q} is an $(M - 1) \times (M - 1)$ matrix, and \mathbf{b} is the $(M - 1)$ st-order BLP. Next we show that for Toeplitz matrices we can compute \mathbf{Q}_M with $O(M^2)$ computations.

First, we note that the last column and the last row of \mathbf{Q}_M can be obtained by solving the Toeplitz system $\mathbf{R}\mathbf{b} = -\mathbf{J}\mathbf{r}$ using the Levinson-Durbin algorithm. Then we show that we can compute the elements of \mathbf{Q} by exploiting the persymmetry property of Toeplitz matrices, moving from the known edges to the interior. Indeed, since \mathbf{R} is persymmetric, that is, $\mathbf{R} = \mathbf{J}\mathbf{R}\mathbf{J}$, we have $\mathbf{R}^{-1} = \mathbf{J}\mathbf{R}^{-1}\mathbf{J}$, that is, \mathbf{R}^{-1} is also persymmetric. From (7.7.21), we have

$$\langle \mathbf{Q} \rangle_{ij} = \langle \mathbf{R}^{-1} \rangle_{ij} + P q_i q_j = \langle \mathbf{R}^{-1} \rangle_{M-j,M-i} + P q_i q_j \quad (7.7.22)$$

because \mathbf{R}^{-1} is persymmetric, and

$$\langle \mathbf{R}^{-1} \rangle_{M-j,M-i} = \langle \mathbf{Q} \rangle_{M-j,M-i} - P q_{M-j} q_{M-i} \quad (7.7.23)$$

Combining (7.7.22) and (7.7.23), we obtain

$$\langle \mathbf{Q} \rangle_{ij} = \langle \mathbf{Q} \rangle_{M-j,M-i} - P(q_i q_j - q_{M-j} q_{M-i}) \quad (7.7.24)$$

which in conjunction with persymmetry makes possible the computation of the elements of \mathbf{Q} from \mathbf{q} and P . The process is illustrated for $M = 6$ in the following diagram

$$\mathbf{Q}_6 = \begin{bmatrix} p_1 & p_1 & p_1 & p_1 & p_1 & k \\ p_1 & p_2 & p_2 & p_2 & u_1 & k \\ p_1 & p_2 & p_3 & u_2 & u_1 & k \\ p_1 & p_2 & u_2 & u_2 & u_1 & k \\ p_1 & u_1 & u_1 & u_1 & u_1 & k \\ k & k & k & k & k & k \end{bmatrix}$$

where we start with the known elements k and then compute the u elements by using the updating property (7.7.22) and the elements p by using the persymmetry property (7.7.24) in the following order: $k \rightarrow p_1 \rightarrow u_1 \rightarrow p_2 \rightarrow u_2 \rightarrow p_3$. Clearly, because the matrix $\mathbf{Q}_M = \mathbf{R}_M^{-1}$ is both symmetric and persymmetric, we need to compute only the elements in the following wedge:

$$\begin{array}{cccccc} p_1 & p_1 & p_1 & p_1 & p_1 & k \\ & p_2 & p_2 & p_2 & u_1 & \\ & & p_3 & u_2 & & \end{array}$$

which can be easily extended to the general case. This algorithm, which was introduced by Trench (1964), requires $O(M^2)$ operations and is implemented by the function

`Q=invtoepl(r,M)`

The algorithm is generalized for complex Toeplitz matrices in Problem 7.40.

7.8 KALMAN FILTER ALGORITHM

The various optimum linear filter algorithms and structures that we discussed so far in this chapter provide us with the determination of filter coefficients or optimal estimates using some form of recursive update. Some algorithms and structures are order-recursive while others are time-recursive. In effect, they tell us how the past values should be updated to

determine the present values. Unfortunately, these techniques do not lend themselves very well to the more complicated nonstationary problems. Readers will note carefully that the only case in which we obtained efficient order-recursive algorithms and structures was in the stationary environment, using the approaches of Levinson and Schür.

In 1960, R. E. Kalman provided an alternative approach to formulating the MMSE linear filtering problem using dynamic models. This “Kalman filter” technique was quickly hailed as a practical solution to a number of problems that were intractable using the more established Wiener methods. As we see in this section, the Kalman filter algorithm is actually a special case of the optimal linear filter algorithms that we have studied. However, it is used in a number of fields such as aerospace and navigation, where a signal trajectory can be well defined. Its use in statistical signal processing is somewhat limited (adaptive filters discussed in Chapter 10 are more appropriate). The two main features of the Kalman filter formulation and solution are the dynamic (or state-space) modeling of the random processes under consideration and the time-recursive processing of the input data.

In this section, we discuss only the discrete-time Kalman filter. The continuous-time version is covered in several texts including Gelb (1977) and Brown and Hwang (1997). As a motivation to this approach, we begin with the following estimation problem.

7.8.1 Preliminary Development

Suppose that we want to obtain a linear MMSE estimate of a random variable y using the related random variables (observations) $\{x_1, x_2, \dots, x_m\}$, that is,

$$\hat{y}_m \triangleq E\{y|x_1, x_2, \dots, x_m\} \quad (7.8.1)$$

as described in Section 7.1.5. Furthermore, we want to obtain this estimate in an order-recursive fashion, that is, determine \hat{y}_m in terms of \hat{y}_{m-1} . We considered and solved this problem in Section 7.1. Our approach, which is somewhat different from that in Section 7.1, is as follows: Assume that we have computed the corresponding estimate \hat{y}_{m-1} , we have the observations $\{x_1, x_2, \dots, x_m\}$, and we wish to determine the estimate \hat{y}_m . Then we carry out the following steps:

1. We first determine the optimal *one-step prediction* of x_m , that is,

$$\begin{aligned} \hat{x}_{m|m-1} &\triangleq \{x_m|x_1, x_2, \dots, x_{m-1}\} \\ &= [\mathbf{R}_{m-1}^{-1} \mathbf{r}_{m-1}^b]^H \mathbf{x}_{m-1} = -\mathbf{b}_{m-1}^H \mathbf{x}_{m-1} \\ &= -\sum_{k=1}^{m-1} [b_k^{(m-1)}]^* x_k \end{aligned} \quad (7.8.2)$$

where the vector and matrix quantities are as defined in Section 7.1.

2. When the new data value x_m is received, we determine the optimal *prediction error*

$$e_m^b \triangleq x_m - \hat{x}_{m|m-1} = w_m \quad (7.8.3)$$

which is the new information or innovations contained in the new data.

3. Determine a linear MMSE estimate of y , given the new information w_m :

$$E\{y|w_m\} = E\{y_m w_m^*\} (E\{w_m w_m^*\})^{-1} w_m \quad (7.8.4)$$

4. Finally, form a linear estimate \hat{y}_m of the form

$$\hat{y}_m = \hat{y}_{m-1} + E\{y|w_m\} = \hat{y}_{m-1} + E\{y_m w_m^*\} (E\{w_m w_m^*\})^{-1} w_m \quad (7.8.5)$$

The algorithm is initialized with $\hat{y}_0 = 0$. Note that the quantity $E\{y_m w_m^*\} (E\{w_m w_m^*\})^{-1}$ is equal to the coefficient k_m^* and that we have rederived (7.1.51). For the implementation of (7.8.5), see Figure 7.1.

EXAMPLE 7.8.1. Let the observed random data be obtained from a stationary random process; that is, the data are of the form

$$\{x(1), x(2), \dots, x(n), \dots\} \quad r(n, l) = r(n - l)$$

Also instead of estimating a single random variable, we want to estimate the sample $y(n)$ of a random process $\{y(n)\}$ that is jointly stationary with $x(n)$. Then, following the analysis leading to (7.8.5), we obtain

$$\hat{y}(n) = \hat{y}(n - 1) + k_n^* w(n) = \hat{y}(n - 1) + k_n^* [x(n) + \sum_{k=0}^{n-1} [b_k^{(n-1)}]^* x(k)] \quad (7.8.6)$$

It is interesting to note that, because of stationarity, we have a time-recursive algorithm in (7.8.6). The coefficients $\{k_n^*\}$ can be obtained recursively by using the algorithms of Levinson or Schür. However, the data prediction term does require a growing memory. Indeed, if we define the vector

$$\mathbf{x}(n) = [x(1) \ x(2) \ \dots \ x(n)]^T$$

whose order is equal to time index n , we have

$$\hat{y}(n) = \sum_{k=1}^n [c_k^{(n)}]^* x(k) \triangleq \mathbf{c}_n^H \mathbf{x}(n)$$

The optimum estimator is given by

$$\mathbf{R}_n \mathbf{c}_n = \mathbf{d}_n$$

$$\text{where } \mathbf{R}_n \triangleq E\{\mathbf{x}(n)\mathbf{x}^H(n)\} \quad \mathbf{d}_n \triangleq E\{\mathbf{x}(n)y^*(n)\}$$

Since, owing to stationarity, the matrix \mathbf{R}_n is Toeplitz, we can derive a lattice-ladder structure $\{k_n, k_n^c\}$ that solves this problem recursively (see Section 7.4). When each new observation $\{y(n+1)\}$ is received, we use the moments $r(n+1)$ and $d(n+1)$ to compute new lattice-ladder parameters $\{k_{n+1}, k_{n+1}^c\}$ and we add a new stage to the “growing-order” (and, therefore, growing-memory) filter.

The above example underscores two problems with our estimation technique if we were to obtain a true time-recursive algorithm with finite memory. The first problem concerns the time-recursive update for the k_n^* term or, in particular, for $E\{y_m w_m^*\}$ and $(E\{w_m w_m^*\})^{-1}$. We alluded to this problem in Section 7.1. In the example, we solved this problem by assuming a stationary signal environment. The second problem deals with the infinite memory in (7.8.2). This problem can be solved if we are able to compute the data prediction term also in a time-recursive fashion. In the stationary case, this problem can be solved by using the Levinson-Durbin or Schür algorithm. For nonstationary situations, the above two problems are solved by the Kalman filter by assuming appropriate dynamic models for the process to be estimated and for the observation data.

Consider the optimal one-step prediction term in (7.8.2), defined as

$$\hat{x}(n|n-1) \triangleq E\{x(n)|x(0), \dots, x(n-1)\} \quad (7.8.7)$$

which requires growing memory. If we assume the following linear data relation model

$$x(n) = H(n)y(n) + v(n) \quad (7.8.8)$$

$$\text{with } E\{v(n)y^*(l)\} = 0 \quad \text{for all } n, l \quad (7.8.9)$$

$$E\{v(n)v^*(l)\} = r_v(n)\delta_{n,l} \quad \text{for all } n, l \quad (7.8.10)$$

then (7.8.7) becomes

$$\begin{aligned} \hat{x}(n|n-1) &= E\{[H(n)y(n) + v(n)]|x(0), \dots, x(n-1)\} \\ &= H(n)\hat{y}(n|n-1) \end{aligned} \quad (7.8.11)$$

where we have used the notation

$$\hat{y}(n|n-1) \triangleq E\{y(n)|x(0), \dots, x(n-1)\} \quad (7.8.12)$$

Thus, we will be successful in obtaining a finite-memory computation for $\hat{x}(n|n-1)$ if we can obtain a recursion for $\hat{y}(n|n-1)$ in terms of $\hat{y}(n-1|n-1)$. This is possible if we assume the following linear signal model

$$y(n) = a(n-1)y(n-1) + \eta(n) \quad (7.8.13)$$

with appropriate statistical assumptions on the random process $\eta(n)$. Thus it is now possible to complete the development of the Kalman filter. The *signal model* (7.8.13) provides the dynamics of the time evolution of the signal to be estimated while (7.8.8) is known as the *observation model*, since it relates the signal $y(n)$ with the observation $x(n)$. These models are formally defined in the next section.

7.8.2 Development of Kalman Filter

Since the Kalman filter is also well suited for vector processes, we begin by assuming that the random process to be estimated can be modeled in the form

$$\mathbf{y}(n) = \mathbf{A}(n-1)\mathbf{y}(n-1) + \mathbf{B}(n)\boldsymbol{\eta}(n) \quad (7.8.14)$$

which is known as the *signal* (or *state vector*) *model* where

$\mathbf{y}(n) = k \times 1$ signal state vector at time n

$\mathbf{A}(n-1) = k \times k$ matrix that relates $\mathbf{y}(n-1)$ to $\mathbf{y}(n)$ in absence of a forcing function

$\boldsymbol{\eta}(n) = k \times 1$ zero-mean white noise sequence with covariance matrix $\mathbf{R}_\eta(n)$

$\mathbf{B}(n) = k \times k$ input matrix

(7.8.15)

The matrix $\mathbf{A}(n-1)$ is known as the *state-transition matrix* while $\boldsymbol{\eta}(n)$ is also known as the *modeling error vector*.

The *observation* (or *measurement*) *model* is described using the linear relationship

$$\mathbf{x}(n) = \mathbf{H}(n)\mathbf{y}(n) + \mathbf{v}(n) \quad (7.8.16)$$

where

$\mathbf{x}(n) = m \times 1$ signal state vector at time n

$\mathbf{H}(n) = m \times k$ matrix that gives ideal linear relationship between $\mathbf{y}(n)$ and $\mathbf{x}(n)$

$\mathbf{v}(n) = k \times 1$ zero-mean white noise sequence with covariance matrix $\mathbf{R}_v(n)$

(7.8.17)

The matrix $\mathbf{H}(n)$ is known as the *output matrix*, and the sequence $\mathbf{v}(n)$ is known as the *observation error*.

We further assume the following statistical properties:

$$E\{\mathbf{y}(n)\mathbf{v}^H(l)\} = \mathbf{0} \quad \text{for all } n, l \quad (7.8.18)$$

$$E\{\boldsymbol{\eta}(n)\mathbf{v}^H(l)\} = \mathbf{0} \quad \text{for all } n, l \quad (7.8.19)$$

$$E\{\boldsymbol{\eta}(n)\mathbf{y}^H(-1)\} = \mathbf{0} \quad \text{for all } n \quad (7.8.20)$$

$$E\{\mathbf{y}(-1)\} = \mathbf{0} \quad (7.8.21)$$

$$E\{\mathbf{y}(-1)\mathbf{y}^H(-1)\} = \mathbf{R}_y(-1) \quad (7.8.22)$$

The first three relations, (7.8.18) to (7.8.20), imply orthogonality between respective random variables while the last two, (7.8.21) and (7.8.22), establish the mean and covariance of the initial-condition vector $\mathbf{y}(-1)$.

From (7.8.14) and (7.8.21) the mean of $\mathbf{y}(n) = \mathbf{0}$ for all n , and the evolution of its correlation matrix is given by

$$\mathbf{R}_y(n) = \mathbf{A}(n-1)\mathbf{R}_y(n-1)\mathbf{A}^H(n-1) + \mathbf{B}(n)\mathbf{R}_\eta(n)\mathbf{B}^H(n) \quad (7.8.23)$$

From (7.8.16), the mean of $\mathbf{x}(n) = \mathbf{0}$ for all n , and from (7.8.23) the evolution of its correlation matrix is given by

$$\begin{aligned}\mathbf{R}_x(n) &= \mathbf{H}(n)[\mathbf{A}(n-1)\mathbf{R}_y(n-1)\mathbf{A}^H(n-1) \\ &\quad + \mathbf{B}(n)\mathbf{R}_v(n)\mathbf{B}^H(n)]\mathbf{H}^H(n) + \mathbf{R}_v(n)\end{aligned}\quad (7.8.24)$$

Evolution of optimal estimates

We now assume that we have available the MMSE estimate $\hat{\mathbf{y}}(n-1|n-1)$ of $\mathbf{y}(n-1)$ based on the observations up to and including time $n-1$. Using (7.8.14) and (7.8.20), the one-step prediction of $\mathbf{y}(n)$ is given by

$$\hat{\mathbf{y}}(n|n-1) = \mathbf{A}(n-1)\hat{\mathbf{y}}(n-1|n-1) \quad (7.8.25)$$

with initial condition $\hat{\mathbf{y}}(-1|-1) = \mathbf{y}(-1)$. From (7.8.16), the one-step prediction of $\mathbf{x}(n)$ is given by

$$\hat{\mathbf{x}}(n|n-1) = \mathbf{H}(n)\hat{\mathbf{y}}(n|n-1) = \mathbf{H}(n)\mathbf{A}(n-1)\hat{\mathbf{y}}(n-1|n-1) \quad (7.8.26)$$

Thus we have a recursive formula to compute the predicted observation. The prediction error (7.8.3) from (7.8.16) is now given by

$$\begin{aligned}\mathbf{w}(n) &= \mathbf{x}(n) - \hat{\mathbf{x}}(n|n-1) \\ &= \mathbf{H}(n)\mathbf{y}(n) + \mathbf{v}(n) - \mathbf{H}(n)\hat{\mathbf{y}}(n|n-1) \\ &= \mathbf{H}(n)\tilde{\mathbf{y}}(n|n-1) + \mathbf{v}(n)\end{aligned}\quad (7.8.27)$$

where we have defined the signal prediction error

$$\tilde{\mathbf{y}}(n|n-1) \triangleq \mathbf{y}(n) - \hat{\mathbf{y}}(n|n-1) \quad (7.8.28)$$

Now the quantity corresponding to $E\{w_m w_m^*\}$ in (7.8.5) is given by

$$\mathbf{R}_w(n) = E\{\mathbf{w}(n)\mathbf{w}^H(n)\} = \mathbf{H}(n)\mathbf{R}_{\tilde{\mathbf{y}}}(n|n-1)\mathbf{H}^H(n) + \mathbf{R}_v(n) \quad (7.8.29)$$

where $\mathbf{R}_{\tilde{\mathbf{y}}}(n|n-1) \triangleq E\{\tilde{\mathbf{y}}(n|n-1)\tilde{\mathbf{y}}^H(n|n-1)\}$ (7.8.30)

is called the prediction (*a priori*) error covariance matrix. Similarly, from (7.8.27) the quantity corresponding to $E\{y_m w_m^*\}$ in (7.8.5) is given by

$$\begin{aligned}E\{\mathbf{y}(n)\mathbf{w}^H(n)\} &= E\{\mathbf{y}(n)[\tilde{\mathbf{y}}^H(n|n-1)\mathbf{H}^H(n) + \mathbf{v}^H(n)]\} \\ &= E\{[\tilde{\mathbf{y}}(n|n-1) + \hat{\mathbf{y}}(n|n-1)] \\ &\quad \times [\tilde{\mathbf{y}}^H(n|n-1)\mathbf{H}^H(n) + \mathbf{v}^H(n)]\} \\ &= E\{\tilde{\mathbf{y}}(n|n-1)\tilde{\mathbf{y}}^H(n|n-1)\}\mathbf{H}^H(n) \\ &= \mathbf{R}_{\tilde{\mathbf{y}}}(n|n-1)\mathbf{H}^H(n)\end{aligned}\quad (7.8.31)$$

since the optimal prediction error $\tilde{\mathbf{y}}(n|n-1)$ is orthogonal to the optimal prediction $\hat{\mathbf{y}}(n|n-1)$. Now the updated MMSE estimate (which is also known as the *filtered estimate*) corresponding to (7.8.5) is

$$\begin{aligned}\hat{\mathbf{y}}(n|n) &= \hat{\mathbf{y}}(n|n-1) + \mathbf{R}_{\tilde{\mathbf{y}}}(n|n-1)\mathbf{H}^H(n)\mathbf{R}_w^{-1}(n)\{\mathbf{x}(n) - \hat{\mathbf{x}}(n|n-1)\} \\ &= \hat{\mathbf{y}}(n|n-1) + \mathbf{K}(n)\{\mathbf{x}(n) - \mathbf{H}(n)\hat{\mathbf{y}}(n|n-1)\}\end{aligned}\quad (7.8.32)$$

where we have defined a new quantity

$$\mathbf{K}(n) \triangleq \mathbf{R}_{\tilde{\mathbf{y}}}(n|n-1)\mathbf{H}^H(n)\mathbf{R}_w^{-1}(n) \quad (7.8.33)$$

which is known as the *Kalman gain matrix* and where $\hat{\mathbf{y}}(n|n-1)$ is given in terms of $\hat{\mathbf{y}}(n-1|n-1)$ using (7.8.25). Thus we have

$$\text{Prediction: } \hat{\mathbf{y}}(n|n-1) = \mathbf{A}(n-1)\hat{\mathbf{y}}(n-1|n-1) \quad (7.8.34)$$

$$\text{Filter: } \hat{\mathbf{y}}(n|n) = \hat{\mathbf{y}}(n|n-1) + \mathbf{K}(n)\{\mathbf{x}(n) - \mathbf{H}(n)\hat{\mathbf{y}}(n|n-1)\}$$

and we have succeeded in obtaining a time-updating algorithm for recursively computing the MMSE estimates. All that remains is a time evolution of the gain matrix $\mathbf{K}(n)$. Since $\mathbf{R}_w(n)$ from (7.8.29) also depends on $\mathbf{R}_{\tilde{y}}(n|n-1)$, what we need is an update equation for the error covariance matrix.

Evolution of error covariance matrices

First we define the filtered error as

$$\begin{aligned}\tilde{\mathbf{y}}(n|n) &\triangleq \mathbf{y}(n) - \hat{\mathbf{y}}(n|n) \\ &= \mathbf{y}(n) - \hat{\mathbf{y}}(n|n-1) - \mathbf{K}(n)\{\mathbf{x}(n) - \mathbf{H}(n)\hat{\mathbf{y}}(n|n-1)\} \\ &= \tilde{\mathbf{y}}(n|n-1) - \mathbf{K}(n)\mathbf{w}(n)\end{aligned}\quad (7.8.35)$$

where we have used (7.8.27) and (7.8.34). Then the filtered error covariance is given by

$$\begin{aligned}\mathbf{R}_{\tilde{y}}(n|n) &\triangleq E\{\tilde{\mathbf{y}}(n|n)\tilde{\mathbf{y}}^H(n|n)\} \\ &= \mathbf{R}_{\tilde{y}}(n|n-1) - \mathbf{K}(n)\mathbf{R}_w(n)\mathbf{K}^H(n) \\ &= \mathbf{R}_{\tilde{y}}(n|n-1) - \mathbf{K}(n)\mathbf{R}_w(n)\mathbf{R}_w^{-1}(n)\mathbf{H}(n)\mathbf{R}_{\tilde{y}}(n|n-1) \\ &= [\mathbf{I} - \mathbf{K}(n)\mathbf{H}(n)]\mathbf{R}_{\tilde{y}}(n|n-1)\end{aligned}\quad (7.8.36)$$

where in the second-to-last step we substituted (7.8.33) for $\mathbf{K}^H(n)$. The error covariance $\mathbf{R}_{\tilde{y}}(n|n)$ is also known as the *a posteriori* error covariance. Finally, we need to determine the a priori prediction error covariance at time n from $\mathbf{R}_{\tilde{y}}(n-1|n-1)$ to complete the recursive calculations. From the prediction equation in (7.8.34), we obtain the prediction error at time n as

$$\begin{aligned}\mathbf{y}(n) - \hat{\mathbf{y}}(n|n-1) &= \mathbf{A}(n-1)\mathbf{y}(n-1) + \mathbf{B}(n)\eta(n) - \mathbf{A}(n-1)\hat{\mathbf{y}}(n-1|n-1) \\ \tilde{\mathbf{y}}(n|n-1) &= \mathbf{A}(n-1)\tilde{\mathbf{y}}(n-1|n-1) + \mathbf{B}(n)\eta(n)\end{aligned}\quad (7.8.37)$$

$$\text{or } \mathbf{R}_{\tilde{y}}(n|n-1) = \mathbf{A}(n-1)\mathbf{R}_{\tilde{y}}(n-1|n-1)\mathbf{A}^H(n-1) + \mathbf{B}(n)\mathbf{R}_{\eta}(n)\mathbf{B}^H(n) \quad (7.8.38)$$

with initial condition $\mathbf{R}_{\tilde{y}}(-1|-1) = \mathbf{R}_y(-1)$. Thus we have

$$\begin{aligned}\text{A priori error covariance: } \mathbf{R}_{\tilde{y}}(n|n-1) &= \mathbf{A}(n-1)\mathbf{R}_{\tilde{y}}(n-1|n-1)\mathbf{A}^H(n-1) \\ &\quad + \mathbf{B}(n)\mathbf{R}_{\eta}(n)\mathbf{B}^H(n) \\ \text{Kalman gain: } \mathbf{K}(n) &= \mathbf{R}_{\tilde{y}}(n|n-1)\mathbf{H}^H(n)\mathbf{R}_w^{-1}(n) \\ \text{A posteriori error covariance: } \mathbf{R}_{\tilde{y}}(n|n) &= [\mathbf{I} - \mathbf{K}(n)\mathbf{H}(n)]\mathbf{R}_{\tilde{y}}(n|n-1)\end{aligned}\quad (7.8.39)$$

The complete Kalman filter algorithm is given in Table 7.5, and the block diagram description is provided in Figure 7.11.

EXAMPLE 7.8.2. Let $y(n)$ be an AR(2) process described by

$$y(n) = 1.8y(n-1) - 0.81y(n-2) + 0.1\eta(n) \quad n \geq 0 \quad (7.8.40)$$

where $\eta(n) \sim \text{WGN}(0, 1)$ and $y(-1) = y(-2) = 0$. We want to determine the linear MMSE estimate of $y(n)$, $n \geq 0$, by observing

$$x(n) = y(n) + \sqrt{10}v(n) \quad n \geq 0 \quad (7.8.41)$$

where $v(n) \sim \text{WGN}(0, 10)$ and orthogonal to $\eta(n)$.

Solution. From (7.8.40) and (7.8.41), we first formulate the state vector and observation equations:

$$\mathbf{y}(n) \triangleq \begin{bmatrix} y(n) \\ y(n-1) \end{bmatrix} = \begin{bmatrix} 1.8 & -0.81 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y(n-1) \\ y(n-2) \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0 \end{bmatrix} \eta(n) \quad (7.8.42)$$

$$\text{and } \mathbf{x}(n) = [1 \quad 0] \begin{bmatrix} y(n) \\ y(n-1) \end{bmatrix} + \sqrt{10}v(n) \quad (7.8.43)$$

TABLE 7.5
Summary of the Kalman filter algorithm.

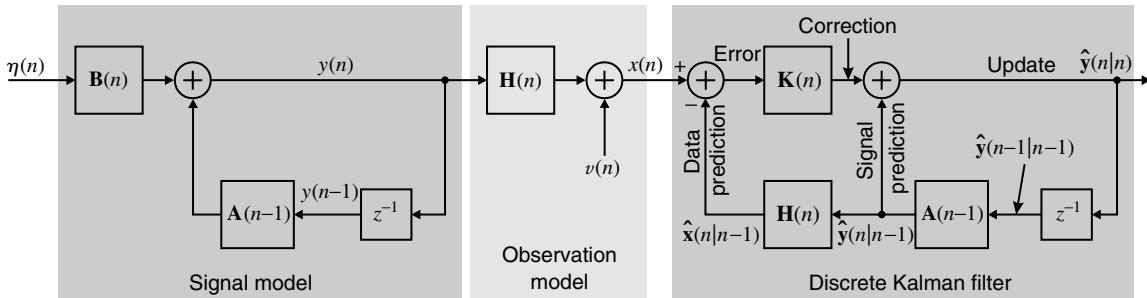
1. **Input:**
 - (a) Signal model parameters: $\mathbf{A}(n-1)$, $\mathbf{B}(n)$, $\mathbf{R}_\eta(n)$; $n = 0, 1, 2, \dots$
 - (b) Observation model parameters: $\mathbf{H}(n)$, $\mathbf{R}_v(n)$; $n = 0, 1, 2, \dots$
 - (c) Observation data: $\mathbf{y}(n)$; $n = 0, 1, 2, \dots$
2. **Initialization:** $\hat{\mathbf{y}}(0|0) = \mathbf{y}(-1) = \mathbf{0}$; $\mathbf{R}_{\hat{\mathbf{y}}}(-1|0) = \mathbf{R}_y(-1)$
3. **Time recursion:** For $n = 0, 1, 2, \dots$
 - (a) Signal prediction: $\hat{\mathbf{y}}(n|n-1) = \mathbf{A}(n-1)\hat{\mathbf{y}}(n-1|n-1)$
 - (b) Data prediction: $\hat{\mathbf{x}}(n|n-1) = \mathbf{H}(n)\hat{\mathbf{y}}(n|n-1)$
 - (c) A priori error covariance:

$$\mathbf{R}_{\hat{\mathbf{y}}}(n|n-1) = \mathbf{A}(n-1)\mathbf{R}_{\hat{\mathbf{y}}}(n-1|n-1)\mathbf{A}^H(n-1) + \mathbf{B}(n)\mathbf{R}_\eta(n)\mathbf{B}^H(n)$$
 - (d) Kalman gain:

$$\mathbf{K}(n) = \mathbf{R}_{\hat{\mathbf{y}}}(n|n-1)\mathbf{H}^H(n)\mathbf{R}_w^{-1}(n)$$

$$\mathbf{R}_w(n) = \mathbf{H}(n)\mathbf{R}_{\hat{\mathbf{y}}}(n|n-1)\mathbf{H}^H(n) + \mathbf{R}_v(n)$$
 - (e) Signal update: $\hat{\mathbf{y}}(n|n) = \hat{\mathbf{y}}(n|n-1) + \mathbf{K}(n)[\mathbf{y}(n) - \hat{\mathbf{x}}(n|n-1)]$
 - (f) A posteriori error covariance:

$$\mathbf{R}_{\hat{\mathbf{y}}}(n|n) = [\mathbf{I} - \mathbf{K}(n)\mathbf{H}(n)]\mathbf{R}_{\hat{\mathbf{y}}}(n|n-1)$$
4. **Output:** Filtered estimate $\hat{\mathbf{y}}(n|n)$, $n = 0, 1, 2, \dots$

**FIGURE 7.11**

The block diagram of the Kalman filter model and algorithm.

Hence the relevant matrix quantities are

$$\mathbf{A}(n) = \begin{bmatrix} 1.8 & -0.81 \\ 1 & 0 \end{bmatrix} \quad \mathbf{B}(n) = \begin{bmatrix} 0.1 \\ 0 \end{bmatrix} \quad \mathbf{R}_\eta(n) = 1$$

and

$$\mathbf{H}(n) = [1 \ 0] \quad \mathbf{R}_v(n) = 10 \quad (7.8.44)$$

Now the Kalman filter equation from Table 7.5 can be implemented with zero initial conditions. Note that since the system matrices are constant, the processes $x(n)$ and $y(n)$ are asymptotically stationary.

Using (7.8.40) and (7.8.41), we generated 100 samples of $y(n)$ and $x(n)$. The observation $x(n)$ was processed using the Kalman filter equations to obtain $\hat{y}_f(n) = \hat{y}(n|n)$, and the results are shown in Figure 7.12. Owing to a large observation noise variance, the $x(n)$ values are very noisy around the signal $y(n)$ values. However, the Kalman filter was able to track $x(n)$ closely and reduce the noise $v(n)$ degradation. In Figure 7.13 we show the evolution of Kalman filter gain values $K_1(n)$ and $K_2(n)$ along with the estimation error variance. The filter reaches its steady state in about 20 samples and becomes a stationary filter as expected. In such situations, the gain and error covariance equations can be implemented off-line (since these equations are data-independent) to obtain a constant-gain matrix. The data then can be filtered using this constant gain to reduce on-line computational complexity.

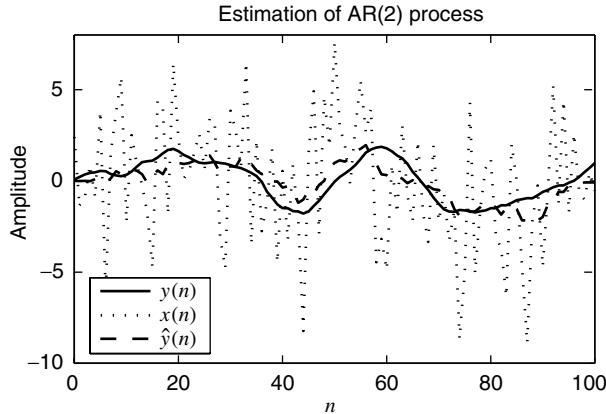


FIGURE 7.12
Estimation of AR(2) process
using Kalman filter in
Example 7.8.2.

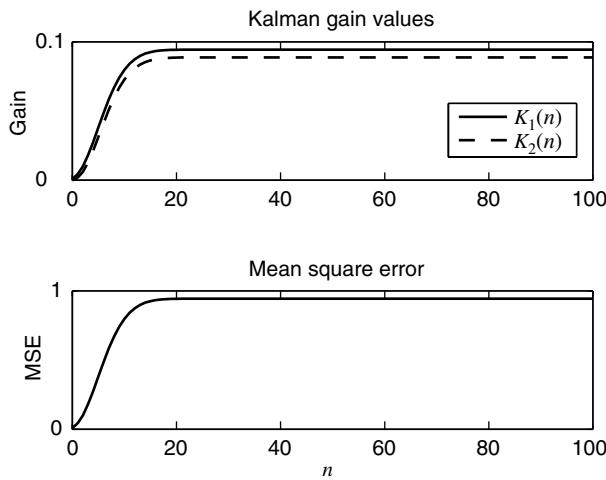


FIGURE 7.13
Kalman filter gains and
estimation error covariance in
Example 7.8.2.

In the next example, we consider the case of the estimation of position of an object in a linear motion subjected to random acceleration.

EXAMPLE 7.8.3. Consider an object traveling in a straight-line motion that is perturbed by random acceleration. Let $y_p(n) = y_c(nT)$ be the true position of the object at the n th sampling instant, where T is the sampling interval in seconds and $y_c(t)$ is the instantaneous position. This position is measured by a sensor that records noisy observations. Let $x(n)$ be the measured position at the n th sampling instant. Then we can model the observation as

$$x(n) = y_p(n) + v(n) \quad n \geq 0 \quad (7.8.45)$$

where $v(n) \sim \text{WGN}(0, \sigma_v^2)$. To derive the state dynamic equation, we assume that the object is in a steady-state motion (except for the random acceleration). Let $y_v(n) = \dot{y}_c(nT)$ be the true velocity at the n th sampling instant, where $\dot{y}_c(t)$ is the instantaneous velocity. Then we have the following equations of motion

$$y_v(n) = y_v(n-1) + y_a(n-1)T \quad (7.8.46)$$

$$y_p(n) = y_p(n-1) + y_v(n-1)T + \frac{1}{2}y_a(n-1)T^2 \quad (7.8.47)$$

where we have assumed that the acceleration $\ddot{y}_c(t)$ is constant over the sampling interval and that $y_a(n-1)$ is the acceleration over $(n-1)T \leq t < nT$. We now define the state vector as

$$\mathbf{y}(n) \triangleq \begin{bmatrix} y_p(n) \\ y_v(n) \end{bmatrix} \quad (7.8.48)$$

and the modeling error as $\eta(n) \triangleq y_a(n - 1)$, which is assumed to be random with $\eta(n) \sim \text{WGN}(0, \sigma_\eta^2)$ and orthogonal to $v(n)$. Thus (7.8.46) and (7.8.47) can be arranged in vector form as

$$\mathbf{y}(n) = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \mathbf{y}(n-1) + \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix} \eta(n) \quad n \geq 0 \quad (7.8.49)$$

Thus we have

$$\mathbf{A} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix}$$

Similarly, the observation (7.8.45) is given by

$$x(n) = [1 \ 0] \mathbf{y}(n) + v(n) \quad n \geq 0 \quad (7.8.50)$$

and hence $\mathbf{H} = [1 \ 0]$. Let the initial conditions be $y_p(-1)$ and $y_v(-1)$. Now given the noisy observations $\{x(n)\}$ and all the necessary information $[T, \sigma_v^2, \sigma_\eta^2, y_p(-1)$, and $y_v(-1)]$, we can recursively estimate the position and velocity of the object at each sampling instance. An approach similar to this is used in aircraft navigation systems.

Using the following values

$$T = 0.1 \quad \sigma_v^2 = \sigma_\eta^2 = 0.25 \quad y_p(-1) = 0 \quad y_v(-1) = 1$$

we simulated the trajectory of the object over $[0, 10]$ second interval. From Table 7.5 Kalman filter equations were obtained, and the true positions as well as velocities were estimated using the noisy positions. Figure 7.14 shows the estimation results. The top graph shows the true, noisy, and estimated positions. The bottom graph shows the true and estimated velocities. Due to random acceleration values (which are moderate), the true velocity has small deviations from the constant value of 1 while the true position trajectory is approximately linear. The estimates of the position follow the true values very closely. However, the velocity estimates have more errors around the true velocities. This is because no direct measurements of velocities are available; therefore, the velocity of the object can be inferred only from position measurements.

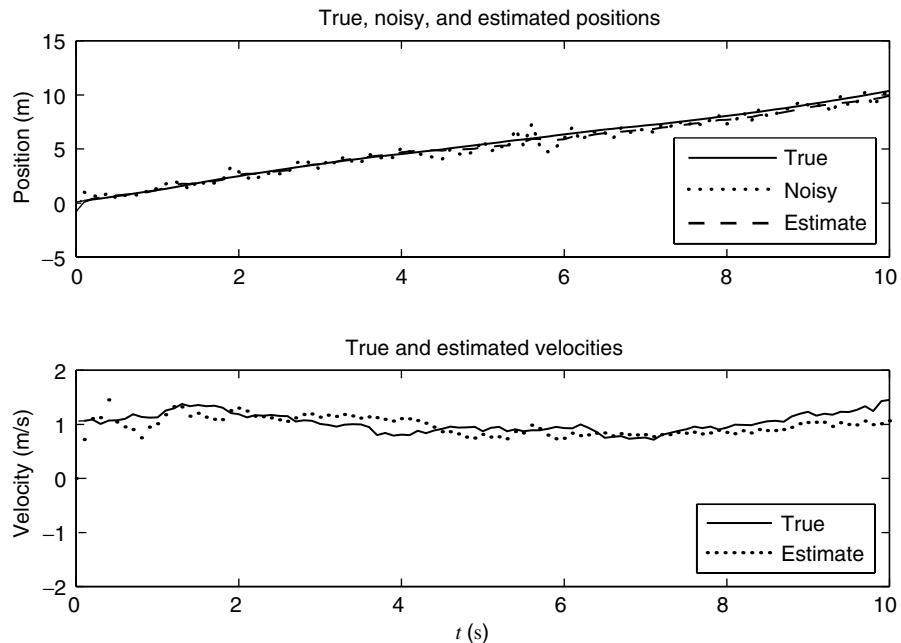


FIGURE 7.14

Estimation of positions and velocities using Kalman filter in Example 7.8.3.

In Figure 7.15, we show the trajectories of Kalman gain values and trace of the error covariance matrices. The top graph contains the gain values corresponding to position (K_p) and velocity (K_v). The steady state of the filter is reached in about 3 s. The bottom left graph contains the a priori and a posteriori error covariances, which also reach the steady-state values in 3 s and which appear to be very close to each other. Therefore, in the bottom right graph we show an exploded view of the steady-state region over a 1-s interval. It is interesting to note that the steady-state error covariances before and after processing an observation are not the same. As a result of making an observation, the a posteriori errors are reduced from the a priori ones. However, owing to random acceleration, the errors increase during the intervals between observations. This is shown as dotted lines in Figure 7.15. The steady state is reached when the decrease in errors achieved by each observation is canceled by the increase between observations.

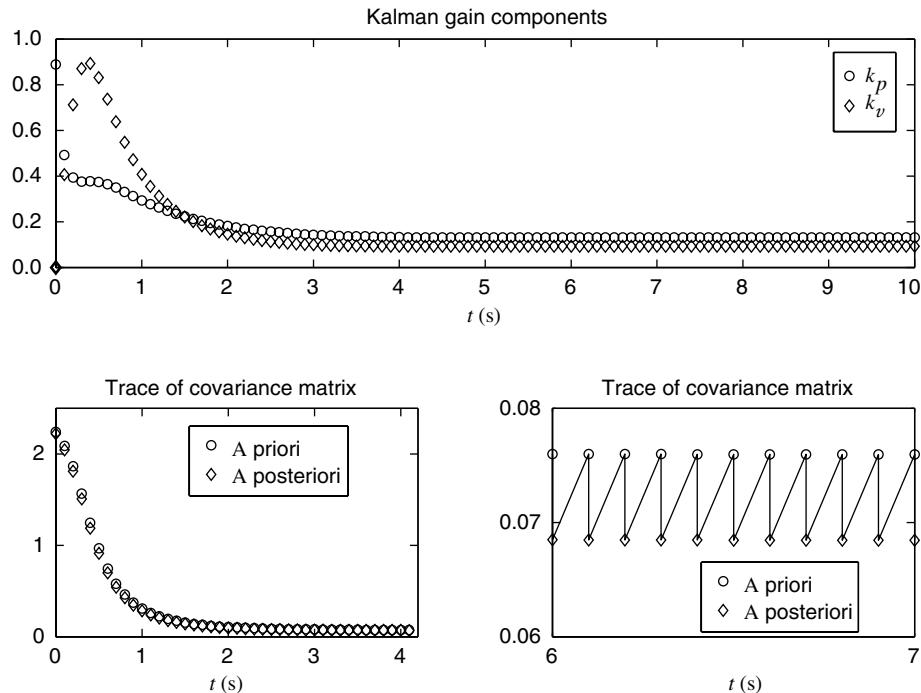


FIGURE 7.15
Kalman filter gains and estimation error variances in Example 7.8.3.

It should be clear from the above two examples that the Kalman filter can recursively estimate signal values because of the assumption of dynamic models (7.8.14) and (7.8.16). Therefore, in this sense, the Kalman filter approach is a special case of the more general Wiener filter problem that we considered earlier. In many signal processing applications (e.g., data communication systems), assumption of such models is difficult to justify, which limits the use of Kalman filters.

7.9 SUMMARY

The application of optimum FIR filters and linear combiners involves the following two steps.

- *Design.* In this step, we determine the optimum values of the estimator parameters by solving the normal equations formed by using the *known* second-order moments. For stationary processes the design step is done only once. For nonstationary processes, we repeat the design when the statistics change.

- *Implementation.* In this step, we use the optimum parameters and the input data to compute the optimum estimate.

The type and complexity of the algorithms and structures available for the design and implementation of linear MMSE estimators depend on two factors:

- The shift invariance of the input data vector.
- The stationarity of the signals that determine the second-order moments in the normal equations.

As we introduce more structure (shift invariance or stationarity), the algorithms and structures become simpler. From a mathematical point of view, this is reflected in the structure of the correlation matrix, which starting from general Hermitian at one end becomes Toeplitz at the other.

Linear combiners

The input vector is not shift-invariant because the optimum estimate is computed by using samples from M different signals. The correlation matrix \mathbf{R} is Hermitian and usually positive definite. The normal equations are solved by using the LDL^H decomposition, and the optimum estimate is computed by using the obtained parameters. However, in many applications where we need the optimum estimate and not the coefficients of the optimum combiner, we can implement the MMSE linear combiner, using the *orthogonal order-recursive structure* shown in Figure 7.1. This structure consists of two parts: (1) a triangular decorrelator (orthogonalizer) that decorrelates the input data vector and produces its innovations vector and (2) a linear combiner that combines the uncorrelated innovations to compute the optimum estimates for *all orders* $1 \leq m \leq M$.

FIR filters and predictors

In this case the input data vector is shift-invariant, which leads to simplifications, whose extent depends on the stationarity of the involved signals.

Nonstationary case. In general, the correlation matrix is Hermitian and positive definite with no additional structure, and the LDL^H decomposition is the recommended method to solve the normal equations. However, the input shift invariance leads to a remarkable coupling between FLP, BLP, and FIR filtering, resulting in a simplified orthogonal order-recursive structure, which now takes the form of a lattice ladder filter (see Figure 7.3). The backward prediction errors of all orders $1 \leq m \leq M$ provide the innovations of the input data vector. The parameters of lattice structure (decorrelator) are specified by the components of the LDL^H decomposition of the input correlation matrix. The coefficients of the ladder part (correlator) depend on both the input correlation matrix and the cross-correlation between the desired response and the input data vector.

Stationary case. In this case, the addition of stationarity to the shift invariance makes the correlation matrix Toeplitz. The presence of the Toeplitz structure has the following consequences:

1. The development of efficient order-recursive algorithms, with computational complexity proportional to M^2 , for the solution of the normal equations and the triangularization of the correlation matrix.
 - a. Levinson algorithm solves $\mathbf{R}\mathbf{c} = \mathbf{d}$ for arbitrary right-hand side vector \mathbf{d} ($2M^2$ operations).
 - b. Levinson-Durbin algorithm solves $\mathbf{R}\mathbf{a} = -\mathbf{r}^*$ when the right-hand side has special structure (M^2 operations).
 - c. Schür algorithm computes directly the lattice-ladder parameters from the autocorrelation and cross-correlation sequences.

2. The MMSE FLP, BLP, and FIR filters are time-invariant; that is, their coefficients (direct-form or lattice-ladder structures) are constant and should be computed only once.

389

PROBLEMS

The algorithms for MMSE filtering and prediction of stationary processes are the simplest ones. However, we can also develop efficient algorithms for nonstationary processes that have special structure. There are two cases of interest:

- The Kalman filtering algorithm that can be used for processes generated by a state-space model with *known* parameters.
- Algorithms for α -stationary processes, that is, processes whose correlation matrix is near to Toeplitz, as measured by a special distance known as the *displacement rank* (Morf et al. 1977).

PROBLEMS

- 7.1** By first computing the matrix product

$$\begin{bmatrix} \mathbf{R}_m & \mathbf{r}_m^b \\ \mathbf{r}_m^{bH} & \rho_m^b \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & -\mathbf{R}_m^{-1}\mathbf{r}_m \\ \mathbf{0}_m & 1 \end{bmatrix}$$

and then the determinants of both sides, prove Equation (7.1.25). Another proof, obtained using the LDL^H decomposition, is given by Equation (7.2.4).

- 7.2** Prove the matrix inversion lemma for lower right corner partitioned matrices, which is described by Equations (7.1.26) and (7.1.28).

- 7.3** This problem generalizes the matrix inversion lemmas to nonsymmetric matrices.

- (a) Show that if \mathbf{R}^{-1} exists, the inverse of an upper left corner partitioned matrix is given by

$$\begin{bmatrix} \mathbf{R} & \mathbf{r} \\ \tilde{\mathbf{r}}^T & \sigma \end{bmatrix}^{-1} = \frac{1}{\alpha} \begin{bmatrix} \alpha \mathbf{R}^{-1} + \mathbf{wv}^T & \mathbf{w} \\ \mathbf{v}^T & 1 \end{bmatrix}$$

where

$$\mathbf{Rw} \triangleq -\mathbf{r}$$

$$\mathbf{R}^T \mathbf{v} \triangleq -\tilde{\mathbf{r}}$$

$$\alpha \triangleq \sigma - \tilde{\mathbf{r}}^T \mathbf{R}^{-1} \mathbf{r} = \sigma + \mathbf{v}^T \mathbf{r} = \sigma + \tilde{\mathbf{r}}^T \mathbf{w}$$

- (b) Show that if \mathbf{R}^{-1} exists, the inverse of a lower right corner partitioned matrix is given by

$$\begin{bmatrix} \sigma & \tilde{\mathbf{r}}^T \\ \mathbf{r} & \mathbf{R} \end{bmatrix}^{-1} = \frac{1}{\alpha} \begin{bmatrix} 1 & \mathbf{v}^T \\ \mathbf{w} & \alpha \mathbf{R}^{-1} + \mathbf{wv}^T \end{bmatrix}$$

where

$$\mathbf{Rw} \triangleq -\mathbf{r}$$

$$\mathbf{R}^T \mathbf{v} \triangleq -\tilde{\mathbf{r}}$$

$$\alpha \triangleq \sigma - \tilde{\mathbf{r}}^T \mathbf{R}^{-1} \mathbf{r} = \sigma + \mathbf{v}^T \mathbf{r} = \sigma + \tilde{\mathbf{r}}^T \mathbf{w}$$

- (c) Check the validity of the lemmas in parts (a) and (b), using MATLAB.

- 7.4** Develop an order-recursive algorithm to solve the linear system in Example 7.1.2, using the lower right corner partitioning lemma (7.1.26).

- 7.5** In this problem we consider two different approaches for inversion of symmetric and positive definite matrices by constructing an arbitrary fourth-order positive definite correlation matrix \mathbf{R} and comparing their computational complexities.

- (a) Given that the inverse of a lower (upper) triangular matrix is itself lower (upper) triangular, develop an algorithm for triangular matrix inversion.
(b) Compute the inverse of \mathbf{R} , using the algorithm in part (a) and Equation (7.1.58).

- (c) Build up the inverse of \mathbf{R} , using the recursion (7.1.24).
 (d) Estimate the number of operations for each method as a function of order M , and check their validity for $M = 4$, using MATLAB.
- 7.6** Using the appropriate orthogonality principles and definitions, prove Equation (7.3.32).
- 7.7** Prove Equations (7.3.36) to (7.3.38), using Equation (7.1.45).
- 7.8** Working as in Example 6.3.1, develop an algorithm for the upper-lower decomposition of a symmetric positive definite matrix. Then use it to factorize the matrix in Example 6.3.1, and verify your results, using the function $[U, D] = \text{udut}(R)$.
- 7.9** In this problem we explore the meaning of the various quantities in the decomposition $\mathbf{R} = \mathbf{U}\bar{\mathbf{D}}\mathbf{U}^H$ of the correlation matrix.
 (a) Show that the rows of $\mathbf{A} = \mathbf{U}^{-1}$ are the MMSE estimator of x_m from $x_{m+1}, x_{m+2}, \dots, x_M$.
 (b) Show that the decomposition $\mathbf{R} = \mathbf{U}\bar{\mathbf{D}}\mathbf{U}^H$ can be obtained by the Gram-Schmidt orthogonalization process, starting with the random variable x_M and ending with x_1 , that is, proceeding backward.
- 7.10** In this problem we clarify the various quantities and the form of the partitionings involved in the \mathbf{UDU}^H decomposition, using an $m = 4$ correlation matrix.
 (a) Prove that the components of the forward prediction error vector (7.3.65) are uncorrelated.
 (b) Writing explicitly the matrix \mathbf{R} , identify and express the quantities in Equations (7.3.62) through (7.3.67).
 (c) Using the matrix \mathbf{R} in Example 6.3.2, compute the predictors in (7.3.67) by using the corresponding normal equations, verify your results, comparing them with the rows of matrix \mathbf{A} computed directly from the \mathbf{LDL}^H decomposition of \mathbf{R}^{-1} or the \mathbf{UDU}^H decomposition of \mathbf{R} (see Table 7.1).
- 7.11** Given an all-zero lattice filter with coefficients k_0 and k_1 , determine the MSE $P(k_0, k_1)$ as a function of the required second-order moments, assumed jointly stationary, and plot the error performance surface. Use the statistics in Example 6.2.1.
- 7.12** Given the autocorrelation $r(0) = 1, r(1) = r(2) = \frac{1}{2}$, and $r(3) = \frac{1}{4}$, determine all possible representations for the third-order prediction error filter (see Figure 7.7).
- 7.13** Repeat Problem 7.12 for $k_0 = k_1 = k_2 = \frac{1}{3}$ and $P_3 = (\frac{2}{3})^3$.
- 7.14** Use Levinson's algorithm to solve the normal equations $\mathbf{R}\mathbf{c} = \mathbf{d}$ where $\mathbf{R} = \text{Toeplitz}\{3, 2, 1\}$ and $\mathbf{d} = [6 \ 6 \ 2]^T$.
- 7.15** Consider a random sequence with autocorrelation $\{r(l)\}_0^3 = \{1, 0.8, 0.6, 0.4\}$. (a) Determine the FLP \mathbf{a}_m and the corresponding error P_m^f for $m = 1, 2, 3$. (b) Determine and draw the flow diagram of the third-order lattice prediction error filter.
- 7.16** Using the Levinson-Durbin algorithm, determine the third-order linear predictor \mathbf{a}_3 and the MMSE P_3 for a signal with autocorrelation $r(0) = 1, r(1) = r(2) = \frac{1}{2}$, and $r(3) = \frac{1}{4}$.
- 7.17** Given the autocorrelation sequence $r(0) = 1, r(1) = r(2) = \frac{1}{2}$, and $r(3) = \frac{1}{4}$, compute the lattice and direct-form coefficients of the prediction error filter, using the algorithm of Schür.
- 7.18** Determine ρ_1 and ρ_2 so that the matrix $\mathbf{R} = \text{Toeplitz}\{1, \rho_1, \rho_2\}$ is positive definite.
- 7.19** Suppose that we want to fit an AR(2) model to a sinusoidal signal with random phase in additive noise. The autocorrelation sequence is given by

$$r(l) = P_0 \cos \omega_0 l + \sigma_v^2 \delta(l)$$

(a) Determine the model parameters $a_1^{(2)}$, $a_2^{(2)}$, and σ_w^2 in terms of P_0 , ω_0 , and σ_v^2 . (b) Determine the lattice parameters of the model. (c) What are the limiting values of the direct and lattice parameters of the model when $\sigma_v^2 \rightarrow 0$?

7.20 Given the parameters $r(0) = 1$, $k_0 = k_1 = \frac{1}{2}$, and $k_2 = \frac{1}{4}$, determine all other equivalent representations of the prediction error filter (see Figure 7.7).

7.21 Let $\{r(l)\}_0^P$ be samples of the autocorrelation sequence of a stationary random signal $x(n)$.
(a) Is it possible to extend $r(l)$ for $|l| > P$ so that the PSD

$$R(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r(l)e^{-j\omega l}$$

is valid, that is, $R(e^{j\omega}) \geq 0$? (b) Using the algorithm of Levinson-Durbin, develop a procedure to check if a given autocorrelation extension is valid. (c) Use the algorithm in part (b) to find the necessary and sufficient conditions so that $r(0) = 1$, $r(1) = \rho_1$, and $r(2) = \rho_2$ are a valid autocorrelation sequence. Is the resulting extension unique?

7.22 Justify the following statements. (a) The whitening filter for a stationary process $x(n)$ is time-varying. (b) The filter in part (a) can be implemented by using a lattice structure and switching its stages on one by one with the arrival of each new sample. (c) If $x(n)$ is AR(P), the whitening filter becomes time-invariant $P + 1$ sampling intervals after the first sample is applied. Note: We assume that the input is applied to the filter at $n = 0$. If the input is applied at $n = -\infty$, the whitening filter of a stationary process is always time-invariant.

7.23 Given the parameters $r(0) = 1$, $k_0 = \frac{1}{2}$, $k_1 = \frac{1}{3}$, and $k_2 = \frac{1}{4}$, compute the determinant of the matrix $\mathbf{R}_4 = \text{Toeplitz}\{r(0), r(1), r(2), r(3)\}$.

7.24 (a) Determine the lattice second-order prediction error filter (PEF) for a sequence $x(n)$ with autocorrelation $r(l) = (\frac{1}{2})^{|l|}$. (b) Repeat part (a) for the sequence $y(n) = x(n) + v(n)$, where $v(n) \sim \text{WN}(0, 0.2)$ is uncorrelated to $x(n)$. (c) Explain the change in the lattice parameters using frequency domain reasoning (think of the PEF as a whitening filter).

7.25 Consider a prediction error filter specified by $P_3 = (\frac{15}{16})^2$, $k_0 = \frac{1}{4}$, $k_1 = \frac{1}{2}$, and $k_2 = \frac{1}{4}$.
(a) Determine the direct-form filter coefficients. (b) Determine the autocorrelation values $r(1)$, $r(2)$, and $r(3)$. (c) Determine the value $r(4)$ so that the MMSE P_4 for the corresponding fourth-order filter is the minimum possible.

7.26 Consider a prediction error filter $A_M(z) = 1 + a_1^{(M)}z^{-1} + \cdots + a_M^{(M)}z^{-M}$ with lattice parameters k_1, k_2, \dots, k_M . (a) Show that if we set $\hat{k}_m = (-1)^m k_m$, then $\hat{a}_m^{(M)} = (-1)^m a_m^{(M)}$. (b) What are the new filter coefficients if we set $\hat{k}_m = \rho^m k_m$, where ρ is a complex number with $|\rho| = 1$? What happens if $|\rho| < 1$?

7.27 Suppose that we are given the values $\{r(l)\}_{-m+1}^{m-1}$ of an autocorrelation sequence such that the Toeplitz matrix \mathbf{R}_m is positive definite. (a) Show that the values of $r(m)$ such that \mathbf{R}_{m+1} is positive definite determine a disk in the complex plane. Find the center α_m and the radius ζ_m of this disk. (b) By induction show that there are infinitely many extensions of $\{r(l)\}_{-m+1}^{m-1}$ that make $\{r(l)\}_{-\infty}^{\infty}$ a valid autocorrelation sequence.

7.28 Consider the MA(1) sequence $x(n) = w(n) + d_1 w(n-1)$, $w(n) \sim \text{WN}(0, \sigma_w^2)$. (a) Show that

$$\det \mathbf{R}_m = r(0) \det \mathbf{R}_{m-1} - |r(1)|^2 \mathbf{R}_{m-2} \quad m \geq 2$$

(b) Show that $k_m = -r^m(1) / \det \mathbf{R}_m$ and that

$$\frac{1}{k_m} = -\frac{r(0)}{r(1)} \frac{1}{k_{m-1}} - \frac{r^*(1)}{r(1)} \frac{1}{k_{m-2}}$$

(c) Determine the initial conditions and solve the recursion in (b) to show that

$$k_m = \frac{(1 - |d_1|^2)(-d_1)^m}{1 - |d_1|^{2m+2}}$$

which tends to zero as $m \rightarrow \infty$.

7.29 Prove Equation (7.6.6) by exploiting the symmetry property $\mathbf{b}_m = \mathbf{J}\mathbf{a}_m^*$.

7.30 In this problem we show that the lattice parameters can be obtained by “feeding” the autocorrelation sequence through the lattice filter as a signal and switching on the stages one by one after the required lattice coefficient is computed. The value of k_m is computed at time $n = m$ from the inputs to stage m . (a) Using (7.6.10), draw the flow diagram of a third-order lattice filter that implements this algorithm. (b) Using the autocorrelation sequence in Example 7.6.1, “feed” the sequence $\{r(n)\}_0^3 = \{3, 2, 1, \frac{1}{2}\}$ through the filter one sample at a time, and compute the lattice parameters. Hint: Use Example 7.6.1 for guidance.

7.31 Draw the supperlattice structure for $M = 8$, and show how it can be partitioned to distribute the computations to three processors for parallel execution.

7.32 Derive the superladder structure shown in Figure 7.10.

7.33 Extend the algorithm of Schür to compute the LDL^H decomposition of a Hermitian Toeplitz matrix, and write a MATLAB function for its implementation.

7.34 Given the matrix $\mathbf{R}_3 = \text{Toeplitz}\{1, \frac{1}{2}, \frac{1}{2}\}$, use the appropriate order-recursive algorithms to compute the following: (a) The LDL^H and UDU^H decompositions of \mathbf{R} , (b) the LDL^H and UDU^H decompositions of \mathbf{R}^{-1} , and (c) the inverse matrix \mathbf{R}^{-1} .

7.35 Consider the AR(1) process $x(n) = \rho x(n-1) + w(n)$, where $w(n) \sim WN(0, \sigma_w^2)$ and $-1 < \rho < 1$. (a) Determine the correlation matrix \mathbf{R}_{M+1} of the process. (b) Determine the M th-order FLP, using the algorithm of Levinson-Durbin. (c) Determine the inverse matrix \mathbf{R}_{M+1}^{-1} , using the triangular decomposition discussed in Section 7.7.

7.36 If $r(l) = \cos \omega_0 l$, determine the second-order prediction error filter and check whether it is minimum-phase.

7.37 Show that the MMSE linear predictor of $x(n+D)$ in terms of $x(n), x(n-1), \dots, x(n-M+1)$ for $D \geq 1$ is given by

$$\mathbf{R}\mathbf{a}^{(D)} = -\mathbf{r}^{(D)}$$

where $\mathbf{r}^{(D)} = [r(D) \ r(D+1) \ \dots \ r(D+M-1)]^T$. Develop a recursion that computes $\mathbf{a}^{(D+1)}$ from $\mathbf{a}^{(D)}$ by exploring the shift invariance of the vector $\mathbf{r}^{(D)}$. See Manolakis et al. (1983).

7.38 The normal equations for the optimum symmetric signal smoother (see Section 6.5.1) can be written as

$$\mathbf{R}_{2m+1}\mathbf{c}_{2m+1} = \begin{bmatrix} \mathbf{0} \\ P_{2m+1} \\ \mathbf{0} \end{bmatrix}$$

where P_{2m+1} is the MMSE, $\mathbf{c}_{2m+1} = \mathbf{J}\mathbf{c}_{2m+1}^*$, and $c_m^{(2m+1)} = 1$. (a) Using a “central” partitioning of \mathbf{R}_{2m+3} and the persymmetry property of Toeplitz matrices, develop a recursion to determine \mathbf{c}_{2m+3} from \mathbf{c}_{2m+1} . (b) Develop a complete order-recursive algorithm for the computation of $\{\mathbf{c}_{2m+1}, P_{2m+1}\}_0^M$ (see Kok et al. 1993).

7.39 Using the triangular decomposition of a Toeplitz correlation matrix, show that (a) the forward prediction errors of various orders and at the same time instant, that is,

$$\mathbf{e}^f(n) = [e_0^f(n) \ e_1^f(n) \ \dots \ e_m^f(n)]^T$$

$$\bar{\mathbf{e}}^f(n) = [e_M^f(n) \ e_{M-1}^f(n-1) \ \cdots \ e_0^f(n-M)]^T$$

are uncorrelated.

7.40 Generalize the inversion algorithm described in Section 7.7.3 to handle Hermitian Toeplitz matrices.

7.41 Consider the estimation of a constant α from its noisy observations. The signal and observation models are given by

$$\begin{aligned} y(n+1) &= y(n) & n > 0 & \quad y(0) = \alpha \\ x(n) &= y(n) + v(n) & v(n) & \sim \text{WGN}(0, \sigma_v^2) \end{aligned}$$

(a) Develop scalar Kalman filter equations, assuming the initial condition on the a posteriori error variance $R_{\tilde{y}}(0|0)$ equal to r_0 .

(b) Show that the a posteriori error variance $R_{\tilde{y}}(n|n)$ is given by

$$R_{\tilde{y}}(n|n) = \frac{r_0}{1 + (r_0/\sigma_v^2)n} \quad (\text{P.1})$$

(c) Show that the optimal filter for the estimation of the constant α is given by

$$\hat{y}(n) = \hat{y}(n-1) + \frac{r_0/\sigma_v^2}{1 + (r_0/\sigma_v^2)n} [x(n) - \hat{y}(n-1)]$$

7.42 Consider a random process with PSD given by

$$R_s(e^{j\omega}) = \frac{4}{2.4661 - 1.629 \cos \omega + 0.81 \cos 2\omega}$$

(a) Using MATLAB, plot the PSD $R_s(e^{j\omega})$ and determine the resonant frequency ω_0 .

(b) Using spectral factorization, develop a signal model for the process of the form

$$\begin{aligned} \mathbf{y}(n) &= \mathbf{A}\mathbf{y}(n-1) + \mathbf{B}\eta(n) \\ s(n) &= [1 \ 0]\mathbf{y}(n) \end{aligned}$$

where $\mathbf{y}(n)$ is a 2×1 vector, $\eta(n) \sim \text{WGN}(0, 1)$, and \mathbf{A} and \mathbf{B} are matrices with appropriate dimensions.

(c) Let $x(n)$ be the observed values of $s(n)$ given by

$$x(n) = s(n) + v(n) \quad v(n) \sim \text{WGN}(0, 1)$$

Assuming reasonable initial conditions, develop Kalman filter equations and implement them, using MATLAB. Study the performance of the filter by simulating a few sample functions of the signal process $s(n)$ and its observation $x(n)$.

7.43 Alternative form of the Kalman filter. A number of different identities and expressions can be obtained for the quantities defining the Kalman filter.

(a) By manipulating the last two equations in (7.8.39) show that

$$\begin{aligned} \mathbf{R}_{\tilde{y}}(n|n) &= \mathbf{R}_{\tilde{y}}(n|n-1) - \mathbf{R}_{\tilde{y}}(n|n-1)\mathbf{H}^H(n) \\ &\quad \times [\mathbf{H}(n)\mathbf{R}_{\tilde{y}}(n|n-1)\mathbf{H}^H(n) + \mathbf{R}_v(n)]^{-1}\mathbf{H}\mathbf{R}_{\tilde{y}}(n|n-1) \end{aligned} \quad (\text{P.2})$$

(b) If the inverses of $\mathbf{R}_{\tilde{y}}(n|n)$, $\mathbf{R}_{\tilde{y}}(n|n-1)$, and \mathbf{R}_v exist, then show that

$$\mathbf{R}_{\tilde{y}}^{-1}(n|n) = \mathbf{R}_{\tilde{y}}^{-1}(n|n-1) + \mathbf{H}^H(n)\mathbf{R}_v^{-1}(n)\mathbf{H}(n) \quad (\text{P.3})$$

This shows that the update of the error covariance matrix does not require the Kalman gain matrix (but does require matrix inverses).

(c) Finally show that the gain matrix is given by

$$\mathbf{K}(n) = \mathbf{R}_{\tilde{y}}(n|n)\mathbf{H}^H(n)\mathbf{R}_v^{-1}(n) \quad (\text{P.4})$$

which is computed by using the a posteriori error covariance matrix.

- 7.44** In Example 7.8.3 we assumed that only the position measurements were available for estimation. In this problem we will assume that we also have a noisy sensor to measure velocity measurements. Hence the observation model is

$$\mathbf{x}(n) \triangleq \begin{bmatrix} x_p(n) \\ x_v(n) \end{bmatrix} = \begin{bmatrix} y_p(n) + v_1(n) \\ y_v(n) + v_2(n) \end{bmatrix} \quad (\text{P.5})$$

where $v_1(n)$ and $v_2(n)$ are two independent zero-mean white Gaussian noise sources with variances $\sigma_{v_1}^2$ and $\sigma_{v_2}^2$, respectively.

- (a) Using the state vector model given in Example 7.8.3 and the observation model in (P.5), develop Kalman filter equations to estimate position and velocity of the object at each n .
- (b) Using the parameter values

$$T = 0.1 \quad \sigma_{v_1}^2 = \sigma_{v_2}^2 = \sigma_\eta^2 = 0.25 \quad y_p(-1) = 0 \quad y_v(-1) = 1$$

simulate the true and observed positions and velocities of the object. Using your Kalman filter equations, generate plots similar to the ones given in Figures 7.14 and 7.15.

- (c) Discuss the effects of velocity measurements on the estimates.

- 7.45** In this problem, we will assume that the acceleration $y_a(n)$ is an AR(1) process rather than a white noise process. Let $y_a(n)$ be given by

$$y_a(n) = \alpha y_a(n - 1) + \eta(n) \quad \eta(n) \sim \text{WGN}(0, \sigma_\eta^2) \quad y_a(-1) = 0 \quad (\text{P.6})$$

- (a) Augment the state vector $\mathbf{y}(n)$ in (7.8.48), using variable $y_a(n)$, and develop the state vector as well as the observation model, assuming that only the position is measured.
- (b) Using the above model and the parameter values

$$T = 0.1 \quad \alpha = 0.9 \quad \sigma_v^2 = \sigma_\eta^2 = 0.25 \\ y_p(-1) = 0 \quad y_v(-1) = 1 \quad y_a(-1) = 0$$

simulate the linear motion of the object. Using Kalman filter equations, estimate the position, velocity, and acceleration values of the object at each n . Generate performance plots similar to the ones given in Figures 7.14 and 7.15.

- (c) Now assume that noisy measurements of $y_v(n)$ and $y_a(n)$ are also available, that is, the observation model is

$$\mathbf{x}(n) \triangleq \begin{bmatrix} x_p(n) \\ x_v(n) \\ x_a(n) \end{bmatrix} = \begin{bmatrix} y_p(n) + v_1(n) \\ y_v(n) + v_2(n) \\ y_a(n) + v_3(n) \end{bmatrix} \quad (\text{P.7})$$

where $v_1(n)$, $v_2(n)$, and $v_3(n)$ are IID zero-mean white Gaussian noise sources with variance σ_v^2 . Repeat parts (a) and (b) above.

Least-Squares Filtering and Prediction

In this chapter, we deal with the design and properties of linear combiners, finite impulse response (FIR) filters, and linear predictors that are optimum in the least-squares error (LSE) sense. The principle of least squares is widely used in practice because second-order moments are rarely known. In the first part of this chapter (Sections 8.1 through 8.4), we concentrate on the design, properties, and applications of least-squares (LS[†]) estimators. Section 8.1 discusses the principle of LS estimation. The unique aspects of the different implementation structures, starting with the general linear combiner followed by the FIR filter and predictor, are treated in Sections 8.2 to 8.4. In the second part (Sections 8.5 to 8.7), we discuss various numerical algorithms for the solution of the LSE normal equations and the computation of LSE estimates including QR decomposition techniques (Householder reflections, Givens rotations, and modified Gram-Schmidt orthogonalization) and the singular value decomposition (SVD).

8.1 THE PRINCIPLE OF LEAST SQUARES

The *principle of least squares* was introduced by the German mathematician Carl Friedrich Gauss, who used it to determine the orbit of the asteroid Ceres in 1821 by formulating the estimation problem as an optimization problem.

The design of optimum filters in the minimum mean square error (MMSE) sense, discussed in Chapter 6, requires the a priori knowledge of second-order moments. However, such statistical information is simply *not* available in most practical applications, for which we can only obtain measurements of the input and desired response signals. To avoid this problem, we can (1) estimate the required second-order moments from the available data (see Chapter 5), if possible, to obtain an estimate of the optimum MMSE filter, or (2) design an optimum filter by minimizing a criterion of performance that is a function of the available data.

In this chapter, we use the minimization of the sum of the squares of the estimation error as the criterion of performance for the design of optimum filters. This method, known as *least-squares error (LSE) estimation*, requires the measurement of *both* the input signal and the desired response signal. A natural question arising at this point is, What is the purpose of estimating the values of a *known*, desired response signal? There are several answers:

[†]A note about abbreviations used throughout the chapter: The two acronyms *LSE* and *LS* will be used almost interchangeably. Although *LSE* is probably the more accurate term, *LS* has become a standard reference to LSE estimators.

1. In system modeling applications, the goal is to obtain a mathematical model describing the input-output behavior of an actual system. A quality estimator provides a good model for the system. The desired result is the estimator or system model, not the actual estimate.
2. In linear predictive coding, the useful result is the prediction error or the respective predictor coefficients.
3. In many applications, the desired response is not available (e.g., digital communications). Therefore, we do not always have a complete set of data from which to design the LSE estimator. However, if the data do not change significantly over a number of sets, then one special complete set, the training set, is used to design the estimator. The resulting estimator is then applied to the processing of the remaining incomplete sets.

The use of measured signal values to determine the coefficients of the estimator leads to some fundamental differences between MMSE and LSE estimation that are discussed where appropriate.

To summarize, depending on the available information, there are two ways to design an optimum estimator: (1) If we know the second-order moments, we use the MMSE criterion and design a filter that is optimum for all possible sets of data with the same statistics. (2) If we only have a block of data, we use the LSE criterion to design an estimator that is optimum for the given block of data. Optimum MMSE estimators are obtained by using ensemble averages, whereas LSE estimators are obtained by using finite-length time averages. For example, an MMSE estimator, designed using ensemble averages, is optimum for all realizations. In contrast, an LSE estimator, designed using a block of data from a particular realization, depends on the numerical values of samples used in the design. If the processes are ergodic, the LSE estimator approaches the MMSE estimator as the block length of the data increases toward infinity.

8.2 LINEAR LEAST-SQUARES ERROR ESTIMATION

We start with the derivation of general linear LS filters that are implemented using the linear combiner structure described in Section 6.2. A set of measurements of the desired response $y(n)$ and the input signals $x_k(n)$ for $1 \leq k \leq M$ has been taken for $0 \leq n \leq N - 1$. As in optimum MMSE estimation, the problem is to estimate the desired response $y(n)$ using the linear combination

$$\hat{y}(n) = \sum_{k=1}^M c_k^*(n) x_k(n) = \mathbf{c}^H(n) \mathbf{x}(n) \quad (8.2.1)$$

We define the estimation error as

$$e(n) = y(n) - \hat{y}(n) = y(n) - \mathbf{c}^H(n) \mathbf{x}(n) \quad (8.2.2)$$

and the coefficients of the combiner are determined by minimizing the sum of the squared errors

$$E \triangleq \sum_{n=0}^{N-1} |e(n)|^2 \quad (8.2.3)$$

that is, the *energy* of the error signal. *For this minimization to be possible, the coefficient vector $\mathbf{c}(n)$ should be held constant over the measurement time interval $0 \leq n \leq N - 1$.* The constant vector \mathbf{c}_{ls} resulting from this optimization depends on the measurement set and is known as the *linear LSE estimator*. In the statistical literature, LSE estimation is known as linear regression, where (8.2.2) is called a *regression function*, $e(n)$ are known as *residuals* (leftovers), and $\mathbf{c}(n)$ is the *regression vector* (Montgomery and Peck 1982).

The system of equations in (8.2.2), or equivalently $e^*(n) = y^*(n) - \mathbf{x}^H(n) \mathbf{c}$, can be written in matrix form as

$$\begin{bmatrix} e^*(0) \\ e^*(1) \\ \vdots \\ e^*(N-1) \end{bmatrix} = \begin{bmatrix} y^*(0) \\ y^*(1) \\ \vdots \\ y^*(N-1) \end{bmatrix} - \begin{bmatrix} x_1^*(0) & x_2^*(0) & \cdots & x_M^*(0) \\ x_1^*(1) & x_2^*(1) & \cdots & x_M^*(1) \\ \vdots & \vdots & \ddots & \vdots \\ x_1^*(N-1) & x_2^*(N-1) & \cdots & x_M^*(N-1) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix} \quad (8.2.4)$$

or more compactly as

$$\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{c} \quad (8.2.5)$$

where

$$\begin{aligned} \mathbf{e} &\triangleq [e(0) \ e(1) \ \cdots \ e(N-1)]^H && \text{error data vector } (N \times 1) \\ \mathbf{y} &\triangleq [y(0) \ y(1) \ \cdots \ y(N-1)]^H && \text{desired response vector } (N \times 1) \\ \mathbf{X} &\triangleq [\mathbf{x}(0) \ \mathbf{x}(1) \ \cdots \ \mathbf{x}(N-1)]^H && \text{input data matrix } (N \times M) \\ \mathbf{c} &\triangleq [c_1 \ c_2 \ \cdots \ c_M]^T && \text{combiner parameter vector } (M \times 1) \end{aligned} \quad (8.2.6)$$

are defined by comparing (8.2.4) to (8.2.5). The input data matrix \mathbf{X} can be partitioned either columnwise or rowwise as follows:

$$\mathbf{X} \triangleq [\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \dots, \tilde{\mathbf{x}}_M] = \begin{bmatrix} \mathbf{x}^H(0) \\ \mathbf{x}^H(1) \\ \vdots \\ \mathbf{x}^H(N-1) \end{bmatrix} \quad (8.2.7)$$

where the columns $\tilde{\mathbf{x}}_k$ of \mathbf{X}

$$\tilde{\mathbf{x}}_k \triangleq [x_k(0) \ x_k(1) \ \cdots \ x_k(N-1)]^H$$

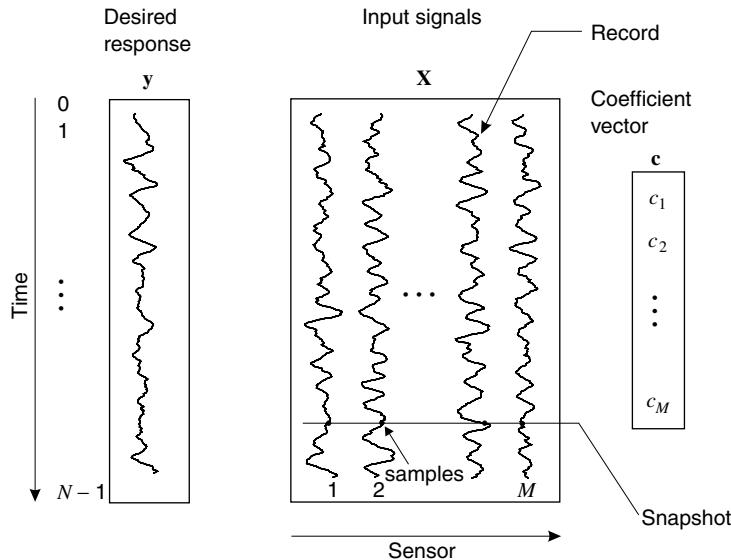
will be called *data records* and the rows

$$\mathbf{x}(n) \triangleq [x_1(n) \ x_2(n) \ \cdots \ x_M(n)]^T$$

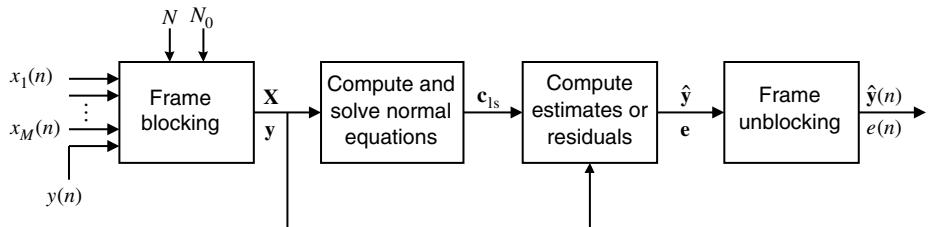
will be called *snapshots*. Both of these partitionings of the data matrix, which are illustrated in Figure 8.1, are useful in the derivation, interpretation, and computation of LSE estimators.

The LSE estimator operates in a block processing mode; that is, it processes a frame of N snapshots using the steps shown in Figure 8.2. The input signals are blocked into frames of N snapshots with successive frames overlapping by N_0 samples. The values of N and N_0 depend on the application. The required estimate or residual signals are unblocked at the final stage of the processor.

If we set $\mathbf{e} = \mathbf{0}$, we have a set of N equations with M unknowns. If $N = M$, then (8.2.4) usually has a unique solution. For $N > M$, we have an overdetermined system of linear equations that typically has no solution. Conversely, if $N < M$, we have an underdetermined system that has an infinite number of solutions. However, even if $M > N$ or $N > M$, the system (8.2.4) has a natural, unique, least-squares solution. We next focus our attention on overdetermined systems since they play a very important role in practical applications. The underdetermined least-squares problem is examined in Section 8.7.2.

**FIGURE 8.1**

The columns of the data matrix are the records of data collected at each input (sensor), whereas each row contains the samples from all inputs at the same instant.

**FIGURE 8.2**

Block processing implementation of a general linear LSE estimator.

8.2.1 Derivation of the Normal Equations

We provide an algebraic and a geometric solution to the LSE estimation problem; a calculus-based derivation is given in Problem 8.1.

Algebraic derivation. The energy of the error can be written as

$$\begin{aligned} E &= \mathbf{e}^H \mathbf{e} = (\mathbf{y}^H - \mathbf{c}^H \mathbf{X}^H)(\mathbf{y} - \mathbf{X}\mathbf{c}) \\ &= \mathbf{y}^H \mathbf{y} - \mathbf{c}^H \mathbf{X}^H \mathbf{y} - \mathbf{y}^H \mathbf{X} \mathbf{c} + \mathbf{c}^H \mathbf{X}^H \mathbf{X} \mathbf{c} \\ &= E_y - \mathbf{c}^H \hat{\mathbf{d}} - \hat{\mathbf{d}}^H \mathbf{c} + \mathbf{c}^H \hat{\mathbf{R}} \mathbf{c} \end{aligned} \quad (8.2.8)$$

where

$$E_y \triangleq \mathbf{y}^H \mathbf{y} = \sum_{n=0}^{N-1} |y(n)|^2 \quad (8.2.9)$$

$$\hat{\mathbf{R}} \triangleq \mathbf{X}^H \mathbf{X} = \sum_{n=0}^{N-1} \mathbf{x}(n) \mathbf{x}^H(n) \quad (8.2.10)$$

$$\hat{\mathbf{d}} \triangleq \mathbf{X}^H \mathbf{y} = \sum_{n=0}^{N-1} \mathbf{x}(n) y^*(n) \quad (8.2.11)$$

Note that these quantities can be viewed as time-average estimates of the desired response power, correlation matrix of the input data vector, and the cross-correlation vector between the desired response and the data vector, when these quantities are divided by the number of data samples N .

We emphasize that all formulas derived for the MMSE criterion hold for the LSE criterion if we replace the expectation $E\{\cdot\}$ with the time-average operator $(1/N) \sum_{n=0}^{N-1} (\cdot)$. This results from the fact that both criteria are quadratic cost functions. Therefore, working as in Section 6.2.2, we conclude that if the time-average correlation matrix $\hat{\mathbf{R}}$ is positive definite, the LSE estimator \mathbf{c}_{ls} is provided by the solution of the normal equations

$$\hat{\mathbf{R}}\mathbf{c}_{ls} = \hat{\mathbf{d}} \quad (8.2.12)$$

and the minimum sum of squared errors is given by

$$E_{ls} = E_y - \hat{\mathbf{d}}^H \hat{\mathbf{R}}^{-1} \hat{\mathbf{d}} = E_y - \hat{\mathbf{d}}^H \mathbf{c}_{ls} \quad (8.2.13)$$

Since $\hat{\mathbf{R}}$ is Hermitian, we only need to compute the elements

$$\hat{r}_{ij} = \tilde{\mathbf{x}}_i^H \tilde{\mathbf{x}}_j \quad (8.2.14)$$

in the upper triangular part, which requires $M(M + 1)/2$ dot products. The right-hand side requires M dot products

$$\hat{d}_i = \tilde{\mathbf{x}}_i^H \mathbf{y} \quad (8.2.15)$$

Note that each dot product involves N arithmetic operations, each consisting of one multiplication and one addition. Thus, to form the normal equations requires a total of

$$\frac{1}{2}M(M + 1)N + MN = \frac{1}{2}M^2N + \frac{3}{2}MN \quad (8.2.16)$$

arithmetic operations. When $\hat{\mathbf{R}}$ is nonsingular, which is the case when $\hat{\mathbf{R}}$ is positive definite, we can solve the normal equations using either the LDL^H or the Cholesky decomposition (see Section 6.3). However, it should be stressed at this point that most of the computational work lies in forming the normal equations rather than their solution. The formulation of the overdetermined LS equations and the normal equations is illustrated graphically in Figure 8.3. The solution of LS problems has been extensively studied in various application areas and in numerical analysis. The basic methods for the solution of the LS problem, which are discussed in this book, are shown in Figure 8.4. We just stress here that for overdetermined LS problems, well-behaved data, and sufficient numerical precision, all these methods provide comparable results.

Geometric derivation. We may think of the desired response record \mathbf{y} and the data records $\tilde{\mathbf{x}}_k$, $1 \leq k \leq M$, as vectors in an N -dimensional vector space, with the dot product and length defined by

$$\langle \tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j \rangle \triangleq \tilde{\mathbf{x}}_i^H \tilde{\mathbf{x}}_j = \sum_{n=0}^{N-1} x_i(n) x_j^*(n) \quad (8.2.17)$$

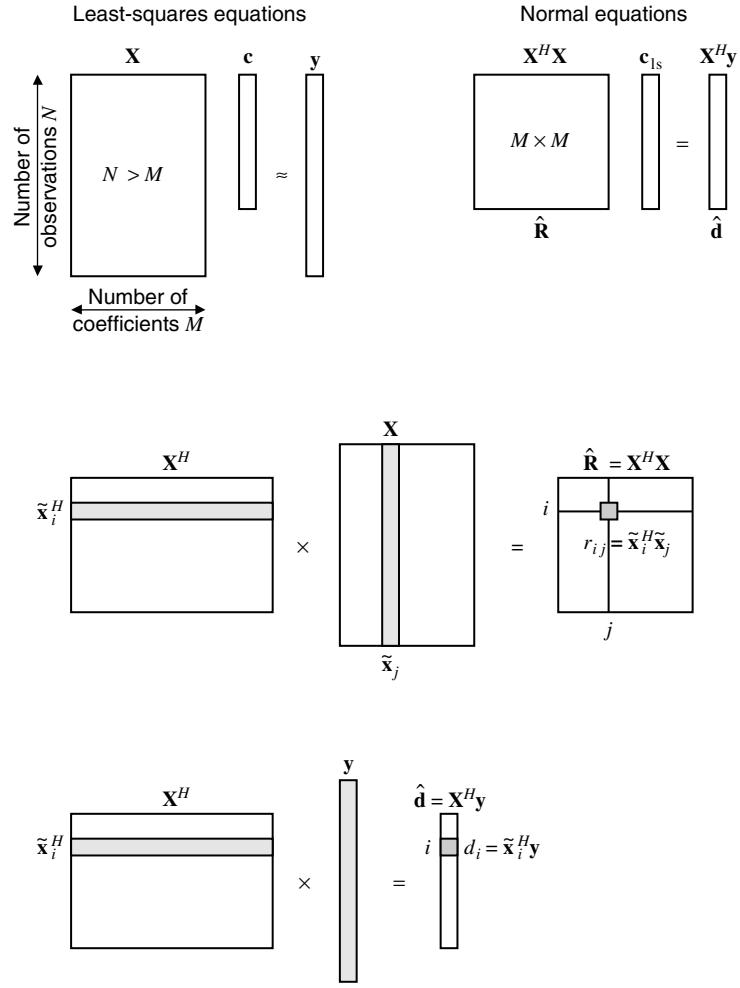
$$\text{and} \quad \|\tilde{\mathbf{x}}\|^2 \triangleq \langle \tilde{\mathbf{x}}, \tilde{\mathbf{x}} \rangle = \sum_{n=0}^{N-1} |x(n)|^2 = E_x \quad (8.2.18)$$

respectively. The estimate of the desired response record can be expressed as

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{c} = \sum_{k=1}^M c_k \tilde{\mathbf{x}}_k \quad (8.2.19)$$

that is, as a linear combination of the data records.

The M vectors $\tilde{\mathbf{x}}_k$ form an M -dimensional subspace, called the *estimation space*, which is the column space of data matrix \mathbf{X} . Clearly, any estimate $\hat{\mathbf{y}}$ must lie in the estimation space. The desired response record \mathbf{y} , in general, lies outside the estimation space. The estimation

**FIGURE 8.3**

The LS problem and computation of the normal equations.

space for $M = 2$ and $N = 3$ is illustrated in Figure 8.5. The error vector \mathbf{e} points from the tip of $\hat{\mathbf{y}}$ to the tip of \mathbf{y} . The squared length of \mathbf{e} is minimum when \mathbf{e} is perpendicular to the estimation space, that is, $\mathbf{e} \perp \tilde{\mathbf{x}}_k$ for $1 \leq k \leq M$.

Therefore, we have the orthogonality principle

$$\langle \tilde{\mathbf{x}}_k, \mathbf{e} \rangle = \tilde{\mathbf{x}}_k^H \mathbf{e} = 0 \quad 1 \leq k \leq M \quad (8.2.20)$$

or more compactly

$$\mathbf{X}^H \mathbf{e} = \mathbf{X}^H (\mathbf{y} - \mathbf{X} \mathbf{c}_{ls}) = \mathbf{0}$$

or

$$(\mathbf{X}^H \mathbf{X}) \mathbf{c}_{ls} = \mathbf{X}^H \mathbf{y} \quad (8.2.21)$$

which we recognize as the LSE normal equations from (8.2.12).

The LS solution splits the desired response \mathbf{y} into two orthogonal components, namely, $\hat{\mathbf{y}}_{ls}$ and \mathbf{e}_{ls} . Therefore,

$$\|\mathbf{y}\|^2 = \|\hat{\mathbf{y}}_{ls}\|^2 + \|\mathbf{e}_{ls}\|^2 \quad (8.2.22)$$

and, using (8.2.18) and (8.2.19), we have

$$E_{ls} = E_y - \mathbf{c}_{ls}^H \mathbf{X}^H \mathbf{X} \mathbf{c}_{ls} = E_y - \mathbf{c}_{ls}^H \mathbf{X}^H \mathbf{y} \quad (8.2.23)$$

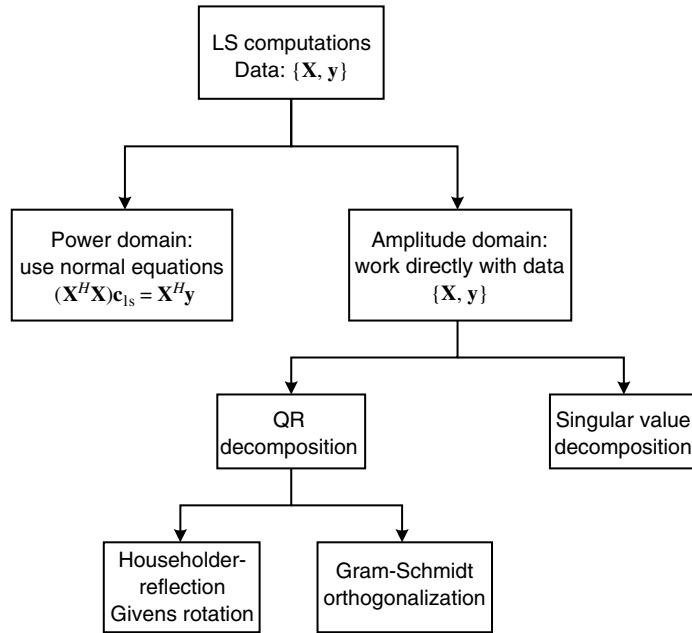


FIGURE 8.4
 Classification of different computational algorithms for the solution of the LS problem.

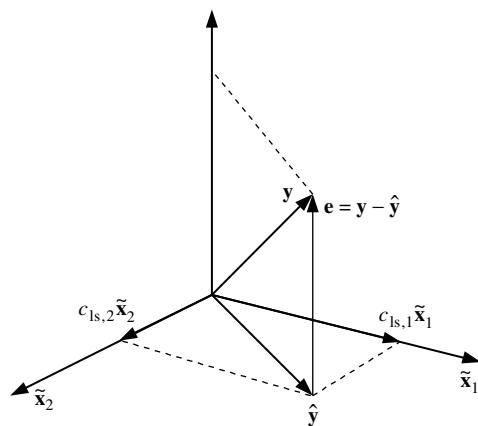


FIGURE 8.5
 Vector space interpretation of LSE estimation for $N = 3$ (dimension of data space) and $M = 2$ (dimension of estimation subspace).

which is identical to (8.2.13). The normalized total squared error is

$$\mathcal{E} \triangleq \frac{E_{ls}}{E_y} = 1 - \frac{E_{\hat{y}}}{E_y} \quad (8.2.24)$$

which is in the range $0 \leq \mathcal{E} \leq 1$, with limits of 0 and 1, which correspond to the worst and best cases, respectively.

Uniqueness. The solution of the LSE normal equations exists and is unique if the time-average correlation matrix $\hat{\mathbf{R}} = \mathbf{X}^H \mathbf{X}$ is invertible. We shall prove the following:

THEOREM 8.1. The time-average correlation matrix $\hat{\mathbf{R}} = \mathbf{X}^H \mathbf{X}$ is invertible if and only if the columns $\tilde{\mathbf{x}}_k$ of \mathbf{X} are linearly independent, or equivalently if and only if $\hat{\mathbf{R}}$ is positive definite.

Proof. If the columns of \mathbf{X} are linearly independent, then for every $\mathbf{z} \neq \mathbf{0}$ we have $\mathbf{Xz} \neq \mathbf{0}$. This implies that for every $\mathbf{z} \neq \mathbf{0}$

$$\mathbf{z}^H(\mathbf{X}^H\mathbf{X})\mathbf{z} = (\mathbf{Xz})^H\mathbf{Xz} = \|\mathbf{Xz}\|^2 > 0 \quad (8.2.25)$$

that is, $\hat{\mathbf{R}}$ is positive definite and hence nonsingular.

If the columns of \mathbf{X} are linearly dependent, then there is a vector $\mathbf{z}_0 \neq \mathbf{0}$ such that $\mathbf{Xz}_0 = \mathbf{0}$. Therefore, $\mathbf{X}^H\mathbf{Xz}_0 = \mathbf{0}$, which implies that $\hat{\mathbf{R}} = \mathbf{X}^H\mathbf{X}$ is singular.

For a matrix to have linearly independent columns, the number of rows should be equal to or larger than the number of columns; that is, we must have more equations than unknowns. To summarize, *the overdetermined ($N > M$) LS problem has a unique solution provided by the normal equations in (8.2.12) if the time-average correlation matrix $\hat{\mathbf{R}}$ is positive definite, or equivalently if the data matrix \mathbf{X} has linearly independent columns.*

In this case, the LS solution can be expressed as

$$\mathbf{c}_{ls} = \mathbf{X}^+\mathbf{y} \quad (8.2.26)$$

where

$$\mathbf{X}^+ \triangleq (\mathbf{X}^H\mathbf{X})^{-1}\mathbf{X}^H \quad (8.2.27)$$

is an $M \times N$ matrix known as the *pseudo-inverse* or the *Moore-Penrose generalized inverse* of matrix \mathbf{X} (Golub and Van Loan 1996; Strang 1980).

The LS estimate $\hat{\mathbf{y}}_{ls}$ of \mathbf{y} can be expressed as

$$\hat{\mathbf{y}}_{ls} = \mathbf{Py} \quad (8.2.28)$$

where

$$\mathbf{P} \triangleq \mathbf{X}(\mathbf{X}^H\mathbf{X})^{-1}\mathbf{X}^H \quad (8.2.29)$$

is known as the *projection matrix* because it projects the data vector \mathbf{y} onto the column space of \mathbf{X} to provide the LS estimate $\hat{\mathbf{y}}_{ls}$ of \mathbf{y} . Similarly, the LS error vector \mathbf{e}_{ls} can be expressed as

$$\mathbf{e}_{ls} = (\mathbf{I} - \mathbf{P})\mathbf{y} \quad (8.2.30)$$

where \mathbf{I} is the $N \times N$ identity matrix. The projection matrix \mathbf{P} is Hermitian and *idempotent*, that is,

$$\mathbf{P} = \mathbf{P}^H \quad (8.2.31)$$

and

$$\mathbf{P}^2 = \mathbf{P}^H\mathbf{P} = \mathbf{P} \quad (8.2.32)$$

respectively.

When the columns of \mathbf{X} are linearly dependent, the LS problem has many solutions. Since all these solutions satisfy the normal equations and the orthogonal projection of \mathbf{y} onto the column space of \mathbf{X} is unique, all these solutions produce an error vector \mathbf{e} of equal length, that is, the same LSE. This subject is discussed in Section 8.6.2 (minimum-norm solution).

EXAMPLE 8.2.1. Suppose that we wish to estimate the sequence $\mathbf{y} = [1 \ 2 \ 3 \ 2]^T$ from the observation vectors $\tilde{\mathbf{x}}_1 = [1 \ 2 \ 1 \ 1]^T$ and $\tilde{\mathbf{x}}_2 = [2 \ 1 \ 2 \ 3]^T$. Determine the optimum filter, the error vector \mathbf{e}_{ls} , and the LSE E_{ls} .

Solution. We first compute the quantities

$$\hat{\mathbf{R}} = \mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}^T \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 7 & 9 \\ 9 & 18 \end{bmatrix} \quad \hat{\mathbf{d}} = \mathbf{X}^T\mathbf{y} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}^T \begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 10 \\ 16 \end{bmatrix}$$

and we then solve the normal equations $\hat{\mathbf{R}}\mathbf{c}_{ls} = \hat{\mathbf{d}}$ to obtain the LS estimator

$$\mathbf{c}_{ls} = \hat{\mathbf{R}}^{-1}\hat{\mathbf{d}} = \begin{bmatrix} \frac{2}{5} & -\frac{1}{5} \\ -\frac{1}{5} & \frac{7}{45} \end{bmatrix} \begin{bmatrix} 10 \\ 16 \end{bmatrix} = \begin{bmatrix} \frac{4}{5} \\ \frac{22}{45} \end{bmatrix}$$

$$E_{ls} = E_y - \hat{\mathbf{d}}^T \mathbf{c}_{ls} = 18 - \begin{bmatrix} 10 \\ 16 \end{bmatrix}^T \begin{bmatrix} \frac{4}{5} \\ \frac{22}{45} \end{bmatrix} = \frac{98}{45}$$

The projection matrix is

$$\mathbf{P} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T = \begin{bmatrix} \frac{2}{9} & \frac{1}{9} & \frac{2}{9} & \frac{1}{3} \\ \frac{1}{9} & \frac{43}{45} & \frac{1}{9} & -\frac{2}{15} \\ \frac{2}{9} & \frac{1}{9} & \frac{2}{9} & \frac{1}{3} \\ \frac{1}{3} & -\frac{2}{15} & \frac{1}{3} & \frac{3}{5} \end{bmatrix}$$

which can be used to determine the error vector

$$\mathbf{e}_{ls} = \mathbf{y} - \mathbf{Py} = \left[-\frac{7}{9} - \frac{4}{45} \frac{11}{9} - \frac{4}{15} \right]^T$$

whose squared norm is equal to $\|\mathbf{e}_{ls}\|^2 = \frac{98}{45} = E_{ls}$, as expected. We can also easily verify the orthogonality principle $\mathbf{e}_{ls}^T \tilde{\mathbf{x}}_1 = \mathbf{e}_{ls}^T \tilde{\mathbf{x}}_2 = 0$.

Weighted least-squares estimation. The previous results were derived by using an LS criterion that treats every error $e(n)$ equally. However, based on a priori information, we may wish to place greater importance on different errors, using the weighted LS criterion

$$E_w = \sum_{n=0}^{N-1} w(n)|e(n)|^2 = \mathbf{e}^H \mathbf{W} \mathbf{e} \quad (8.2.33)$$

where

$$\mathbf{W} \triangleq \text{diag}\{w(0), w(1), \dots, w(N-1)\} \quad (8.2.34)$$

is a diagonal weighting matrix with positive elements. Usually, we choose small weights where the errors are expected to be large, and vice versa. Minimization of E_w with respect to \mathbf{c} yields the *weighted LS (WLS) estimator*

$$\mathbf{c}_{wls} = (\mathbf{X}^H \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^H \mathbf{W} \mathbf{y} \quad (8.2.35)$$

assuming that the inverse of the matrix $\mathbf{X}^H \mathbf{W} \mathbf{X}$ exists. We can easily see that when $\mathbf{W} = \mathbf{I}$, then $\mathbf{c}_{wls} = \mathbf{c}_{ls}$. The criterion in (8.2.33) can be generalized by choosing \mathbf{W} to be any Hermitian, positive definite matrix (see Problem 8.2).

8.2.2 Statistical Properties of Least-Squares Estimators

A useful approach for evaluating the quality of an LS estimator is to study its statistical properties. Toward this end, we assume that the obtained measurements \mathbf{y} actually have been generated by

$$\mathbf{y} = \mathbf{X} \mathbf{c}_o + \mathbf{e}_o \quad (8.2.36)$$

where \mathbf{e}_o is the random measurement error vector. We may think of \mathbf{c}_o as the “true” parameter vector. Using (8.2.36), we see that (8.2.21) gives

$$\mathbf{c}_{ls} = \mathbf{c}_o + (\mathbf{X}^H \mathbf{X})^{-1} \mathbf{X}^H \mathbf{e}_o \quad (8.2.37)$$

We make the following assumptions about the random measurement error vector \mathbf{e}_o :

1. The error vector \mathbf{e}_o has zero mean

$$E\{\mathbf{e}_o\} = \mathbf{0} \quad (8.2.38)$$

2. The error vector \mathbf{e}_o has uncorrelated components with constant variance $\sigma_{e_o}^2$; that is, the correlation matrix is given by

$$\mathbf{R}_{e_o} = E\{\mathbf{e}_o \mathbf{e}_o^H\} = \sigma_{e_o}^2 \mathbf{I} \quad (8.2.39)$$

3. There is no information about \mathbf{e}_o contained in data matrix \mathbf{X} ; that is,

$$E\{\mathbf{e}_o | \mathbf{X}\} = E\{\mathbf{e}_o\} = \mathbf{0} \quad (8.2.40)$$

4. If \mathbf{X} is a deterministic $N \times M$ matrix, then it has rank M . This means that \mathbf{X} is a full-column rank and that $\mathbf{X}^H \mathbf{X}$ is invertible. If \mathbf{X} is a stochastic $N \times M$ matrix, then $E\{(\mathbf{X}^H \mathbf{X})^{-1}\}$ exists.

In the following analysis, we consider two possibilities: \mathbf{X} is deterministic and stochastic. Under these conditions, the LS estimator \mathbf{c}_{ls} has several desirable properties.

Deterministic data matrix

In this case, we assume that the LS estimators are obtained from the deterministic data values; that is, the matrix \mathbf{X} is treated as a matrix of constants. Then the properties of the LS estimators can be derived from the statistical properties of the random measurement error vector \mathbf{e}_o .

PROPERTY 8.2.1. The LS estimator \mathbf{c}_{ls} is an unbiased estimator of \mathbf{c}_o , that is,

$$E\{\mathbf{c}_{ls}\} = \mathbf{c}_o \quad (8.2.41)$$

Proof. Taking the expectation of both sides of (8.2.37), we have

$$E\{\mathbf{c}_{ls}\} = E\{\mathbf{c}_o\} + (\mathbf{X}^H \mathbf{X})^{-1} \mathbf{X}^H E\{\mathbf{e}_o\} = \mathbf{c}_o$$

because \mathbf{X} is deterministic and $E\{\mathbf{e}_o\} = \mathbf{0}$.

PROPERTY 8.2.2. The covariance matrix of \mathbf{c}_{ls} corresponding to the error $\mathbf{c}_{ls} - \mathbf{c}_o$ is

$$\mathbf{\Gamma}_{ls} \triangleq E\{(\mathbf{c}_{ls} - \mathbf{c}_o)(\mathbf{c}_{ls} - \mathbf{c}_o)^H\} = \sigma_{e_o}^2 (\mathbf{X}^H \mathbf{X})^{-1} = \sigma_{e_o}^2 \hat{\mathbf{R}}^{-1} \quad (8.2.42)$$

Proof. Using (8.2.37), (8.2.39), and the definition (8.2.42), we easily obtain

$$\mathbf{\Gamma}_{ls} = (\mathbf{X}^H \mathbf{X})^{-1} \mathbf{X}^H E\{\mathbf{e}_o \mathbf{e}_o^H\} \mathbf{X} (\mathbf{X}^H \mathbf{X})^{-1} = \sigma_{e_o}^2 (\mathbf{X}^H \mathbf{X})^{-1}$$

Note that the diagonal elements of matrix $\sigma_e^2 \hat{\mathbf{R}}^{-1}$ are also equal to the variance of the LS combiner vector \mathbf{c}_{ls} .

PROPERTY 8.2.3. An unbiased estimate of the error variance $\sigma_{e_o}^2$ is given by

$$\hat{\sigma}_{e_o}^2 = \frac{E_{ls}}{N - M} \quad (8.2.43)$$

where N is the number of observations, M is the number of parameters, and E_{ls} is the LS error.

Proof. Using (8.2.30) and (8.2.36), we obtain

$$\mathbf{e}_{ls} = (\mathbf{I} - \mathbf{P})\mathbf{y} = (\mathbf{I} - \mathbf{P})\mathbf{e}_o$$

which results in

$$E_{ls} = \mathbf{e}_{ls}^H \mathbf{e}_{ls} = \mathbf{e}_o^H (\mathbf{I} - \mathbf{P})^H (\mathbf{I} - \mathbf{P}) \mathbf{e}_o = \mathbf{e}_o^H (\mathbf{I} - \mathbf{P}) \mathbf{e}_o$$

because of (8.2.32). Since E_{ls} depends on \mathbf{e}_o , it is a random variable whose expected value is

$$\begin{aligned} E\{E_{ls}\} &= E\{\mathbf{e}_o^H (\mathbf{I} - \mathbf{P}) \mathbf{e}_o\} = E\{\text{tr}[(\mathbf{I} - \mathbf{P}) \mathbf{e}_o \mathbf{e}_o^H]\} \\ &= \text{tr}[(\mathbf{I} - \mathbf{P}) E\{\mathbf{e}_o \mathbf{e}_o^H\}] = \sigma_e^2 \text{tr}(\mathbf{I} - \mathbf{P}) \end{aligned}$$

since $\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA})$, where tr is the trace function. However,

$$\begin{aligned} \text{tr}(\mathbf{I} - \mathbf{P}) &= \text{tr}[\mathbf{I} - \mathbf{X}(\mathbf{X}^H \mathbf{X})^{-1} \mathbf{X}^H] \\ &= \text{tr}[\mathbf{I}_{N \times N} - (\mathbf{X}^H \mathbf{X})^{-1} \mathbf{X}^H \mathbf{X}] \\ &= \text{tr}(\mathbf{I}_{N \times N}) - \text{tr}[(\mathbf{X}^H \mathbf{X})^{-1} \mathbf{X}^H \mathbf{X}] \\ &= \text{tr}(\mathbf{I}_{N \times N}) - \text{tr}(\mathbf{I}_{M \times M}) = N - M \end{aligned}$$

therefore

$$\sigma_{e_o}^2 = \frac{E\{E_{ls}\}}{N - M} \quad (8.2.44)$$

which proves that $\hat{\sigma}_e^2$ is an unbiased estimate of $\sigma_{e_o}^2$.

Similar to (8.2.41), the mean value of \mathbf{c}_{wls} is

$$E\{\mathbf{c}_{wls}\} = E\{\mathbf{c}_o\} + (\mathbf{X}^H \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^H \mathbf{W} E\{\mathbf{e}_o\} = E\{\mathbf{c}_o\} \quad (8.2.45)$$

that is, the WLS estimator is an unbiased estimate of \mathbf{c}_o . The covariance matrix of \mathbf{c}_{wls} is

$$\boldsymbol{\Gamma}_{wls} = (\mathbf{X}^H \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^H \mathbf{W} \mathbf{R}_{e_o} \mathbf{W} \mathbf{X} (\mathbf{X}^H \mathbf{W} \mathbf{X})^{-1} \quad (8.2.46)$$

where \mathbf{R}_{e_o} is the correlation matrix of \mathbf{e}_o . It is easy to see that when $\mathbf{R}_{e_o} = \sigma_{e_o}^2 \mathbf{I}$ and $\mathbf{W} = \mathbf{I}$, we obtain (8.2.42).

PROPERTY 8.2.4. The trace of $\boldsymbol{\Gamma}_{wls}$ attains its minimum when $\mathbf{W} = \mathbf{R}_{e_o}^{-1}$. The resulting estimator

$$\mathbf{c}_{mv} = (\mathbf{X}^H \mathbf{R}_{e_o}^{-1} \mathbf{X})^{-1} \mathbf{X}^H \mathbf{R}_{e_o}^{-1} \mathbf{y} \quad (8.2.47)$$

is known as the *minimum variance or Markov estimator* and is the *best linear unbiased estimator (BLUE)*.

Proof. The proof is somewhat involved. Interested readers can see Goodwin and Payne (1977) and Scharf (1991).

PROPERTY 8.2.5. If $\mathbf{R}_{e_o} = \sigma_{e_o}^2 \mathbf{I}$, the LS estimator \mathbf{c}_{ls} is also the best linear unbiased estimator.

Proof. It follows from (8.2.47) with the substitution $\mathbf{R}_{e_o} = \sigma_{e_o}^2 \mathbf{I}$.

PROPERTY 8.2.6. When the random observation vector \mathbf{e}_o has a normal distribution with mean zero and correlation matrix $\mathbf{R}_{e_o} = \sigma_{e_o}^2 \mathbf{I}$, that is, when its components are uncorrelated, the LS estimator \mathbf{c}_{ls} is also the maximum likelihood estimator.

Proof. Since the components of vector \mathbf{e}_o are uncorrelated and normally distributed with zero mean and variance σ_e^2 , the likelihood function for real-valued \mathbf{e}_o is given by

$$L(\mathbf{c}) = \prod_{n=0}^{N-1} \frac{1}{\sqrt{2\pi\sigma_{e_o}^2}} \exp\left[-\frac{|\mathbf{e}_o(n)|^2}{2\sigma_{e_o}^2}\right] \quad (8.2.48)$$

and its logarithm by

$$\ln L(\mathbf{c}) = -\frac{1}{2\sigma_{e_o}^2} \mathbf{e}_o^H \mathbf{e}_o - \frac{N}{2} \ln(2\pi\sigma_{e_o}^2) = -\frac{1}{2\sigma_{e_o}^2} (\mathbf{y} - \mathbf{X}\mathbf{c})^H (\mathbf{y} - \mathbf{X}\mathbf{c}) + \text{const} \quad (8.2.49)$$

For complex-valued \mathbf{e}_o , the terms $\sqrt{2\pi}\sigma_{e_o}$ and $2\sigma_{e_o}^2$ in (8.2.48) are replaced by $\pi\sigma_{e_o}^2$ and $\sigma_{e_o}^2$, respectively. Since the logarithm is a monotonic function, maximization of $L(\mathbf{c})$ is equivalent to minimization of $\ln L(\mathbf{c})$. It is easy to see, by comparison with (8.2.8), that the LS solution maximizes this likelihood function.

Stochastic data matrix

We now extend the statistical properties of \mathbf{c}_{ls} from the preceding section to the situation in which the data values in \mathbf{X} are obtained from a random source with a known probability distribution. This situation is best handled by first obtaining the desired results conditioned on \mathbf{X} , which is equivalent to the deterministic case. We then determine the unconditional results by (statistical) averaging over the conditional distributions using the following properties of the conditional averages.

The *conditional mean* and the *conditional covariance* of a random vector $\mathbf{x}(\zeta)$, given another random vector $\mathbf{y}(\zeta)$, are defined by

$$\boldsymbol{\mu}_{x|y} \triangleq E\{\mathbf{x}(\zeta)|\mathbf{y}(\zeta)\}$$

and

$$\boldsymbol{\Gamma}_{x|y} \triangleq E\{[\mathbf{x}(\zeta) - \boldsymbol{\mu}_{x|y}][\mathbf{x}(\zeta) - \boldsymbol{\mu}_{x|y}]^H \mid \mathbf{y}(\zeta)\}$$

respectively. Since both quantities are random objects, it can be shown that

$$\boldsymbol{\mu}_x = E\{\mathbf{x}(\zeta)\} = E_y\{E\{\mathbf{x}(\zeta)|\mathbf{y}(\zeta)\}\}$$

which is known as the *law of iterated expectations* and that

$$\boldsymbol{\Gamma}_x = \boldsymbol{\Gamma}_{\mu_{x|y}}^{(y)} + \boldsymbol{\mu}_{\Gamma_{x|y}}^{(y)}$$

which is called the *decomposition of the covariance rule*. This rule states that the covariance of a random vector $\mathbf{x}(\zeta)$ decomposes into the covariance of the conditional mean plus the mean of the conditional covariance. The covariance of the conditional mean, $\boldsymbol{\mu}_{x|y}$, is given by

$$\boldsymbol{\Gamma}_{\mu_{x|y}}^{(y)} \triangleq E_y\{[\boldsymbol{\mu}_{x|y} - \boldsymbol{\mu}_x][\boldsymbol{\mu}_{x|y} - \boldsymbol{\mu}_x]^H\}$$

where the notation $\boldsymbol{\Gamma}_{[\cdot]}^{(y)}$ indicates the covariance over the distribution of $\mathbf{y}(\zeta)$. More details can be found in Greene (1993).

PROPERTY 8.2.7. The LS estimator \mathbf{c}_{ls} is an unbiased estimator of \mathbf{c}_o .

Proof. Taking the conditional expectation with respect to \mathbf{X} of both sides of (8.2.37), we obtain

$$E\{\mathbf{c}_{ls}|\mathbf{X}\} = E\{\mathbf{c}_o|\mathbf{X}\} + (\mathbf{X}^H \mathbf{X})^{-1} \mathbf{X}^H E\{\mathbf{e}_o|\mathbf{X}\} \quad (8.2.50)$$

Now using the law of iterated expectations, we get

$$E\{\mathbf{c}_{ls}\} = E_{\mathbf{X}}\{E\{\mathbf{c}_{ls}|\mathbf{X}\}\} = \mathbf{c}_o + E\{(\mathbf{X}^H \mathbf{X})^{-1} \mathbf{X}^H E\{\mathbf{e}_o|\mathbf{X}\}\}$$

Since $E\{\mathbf{e}_o|\mathbf{X}\} = \mathbf{0}$, from assumption 3, we have $E\{\mathbf{c}_{ls}\} = \mathbf{c}_o$. Thus \mathbf{c}_{ls} is also unconditionally unbiased.

PROPERTY 8.2.8. The covariance matrix of \mathbf{c}_{ls} corresponding to the error $\mathbf{c}_{ls} - \mathbf{c}_o$ is

$$\boldsymbol{\Gamma}_{ls} \triangleq E\{(\mathbf{c}_{ls} - \mathbf{c}_o)(\mathbf{c}_{ls} - \mathbf{c}_o)^H\} = \sigma_{e_o}^2 E\{(\mathbf{X}^H \mathbf{X})^{-1}\} \quad (8.2.51)$$

Proof. From (8.2.42), the conditional covariance matrix of \mathbf{c}_{ls} , conditional on \mathbf{X} , is

$$E\{(\mathbf{c}_{ls} - \mathbf{c}_o)(\mathbf{c}_{ls} - \mathbf{c}_o)^H | \mathbf{X}\} = \sigma_{e_o}^2 (\mathbf{X}^H \mathbf{X})^{-1} \quad (8.2.52)$$

For the unconditional covariance, we use the decomposition of covariance rule to obtain

$$\begin{aligned} E\{(\mathbf{c}_{ls} - \mathbf{c}_o)(\mathbf{c}_{ls} - \mathbf{c}_o)^H\} &= E_{\mathbf{X}}\{E\{(\mathbf{c}_{ls} - \mathbf{c}_o)(\mathbf{c}_{ls} - \mathbf{c}_o)^H | \mathbf{X}\}\} \\ &\quad + E_{\mathbf{X}}\{(E\{\mathbf{c}_{ls}|\mathbf{X}\} - \mathbf{c}_o)(E\{\mathbf{c}_{ls}|\mathbf{X}\} - \mathbf{c}_o)^H\} \end{aligned}$$

The second term on the right-hand side above is equal to zero since $E\{\mathbf{c}_{ls}|\mathbf{X}\} = \mathbf{c}_o$ and hence

$$\begin{aligned} E\{(\mathbf{c}_{ls} - \mathbf{c}_o)(\mathbf{c}_{ls} - \mathbf{c}_o)^H\} &= E_{\mathbf{X}}\{E\{(\mathbf{c}_{ls} - \mathbf{c}_o)(\mathbf{c}_{ls} - \mathbf{c}_o)^H | \mathbf{X}\}\} \\ &= E_{\mathbf{X}}\{\sigma_{e_o}^2 (\mathbf{X}^H \mathbf{X})^{-1}\} = \sigma_{e_o}^2 E\{(\mathbf{X}^H \mathbf{X})^{-1}\} \end{aligned}$$

Thus the earlier result in (8.2.42) is modified by the expected value (or averaging) of $(\mathbf{X}^H \mathbf{X})^{-1}$.

One important conclusion about the statistical properties of the LS estimator is that the results obtained for the deterministic data matrix \mathbf{X} are also valid for the stochastic case. This conclusion also applies for the Markov estimators and maximum likelihood estimators (Greene 1993).

8.3 LEAST-SQUARES FIR FILTERS

We will now apply the theory of linear LS error estimation to the design of FIR filters. The treatment closely follows the notation and approach in Section 6.4. Recall that the filtering error is

$$e(n) = y(n) - \sum_{k=0}^{M-1} h(k) x(n-k) \triangleq y(n) - \mathbf{c}^H \mathbf{x}(n) \quad (8.3.1)$$

where $y(n)$ is the desired response,

$$\mathbf{x}(n) = [x(n) \ x(n-1) \ \cdots \ x(n-M+1)]^T \quad (8.3.2)$$

is the input data vector, and

$$\mathbf{c} = [c_0 \ c_1 \ \cdots \ c_{M-1}]^T \quad (8.3.3)$$

is the filter coefficient vector related to impulse response by $c_k = h^*(k)$. Suppose that we take measurements of the desired response $y(n)$ and the input signal $x(n)$ over the time interval $0 \leq n \leq N-1$. We hold the coefficients $\{c_k\}_0^{M-1}$ of the filter constant within this period and set any other required data samples equal to zero. For example, at time $n = 0$, that is, when we take the first measurement $x(0)$, the filter needs the samples $x(0), x(-1), \dots, x(-M+1)$ to compute the output sample $\hat{y}(0)$. Since the samples $x(-1), \dots, x(-M+1)$ are not available, to operate the filter, we should replace them with *arbitrary* values or start the filtering operation at time $n = M-1$. Indeed, for $M-1 \leq n \leq N-1$, all the input samples of $x(n)$ required by the filter to compute the output $\{\hat{y}(n)\}_{M-1}^{N-1}$ are available. If we want to compute the output while the last sample $x(N-1)$ is still in the filter memory, we must continue the filtering operation until $n = N+M-2$. Again, we need to assign arbitrary values to the unavailable samples $x(N), \dots, x(N+M-2)$. Most often, we set the unavailable samples equal to zero, which can be thought of as windowing the sequences $x(n)$ and $y(n)$ with a rectangular window. To simplify the illustration, suppose that $N = 7$ and $M = 3$. Writing (8.3.1) for $n = 0, 1, \dots, N+M-1$ and arranging in matrix form, we obtain

$$\begin{array}{ll} 0 \rightarrow & \left[\begin{array}{c} e^*(0) \\ e^*(1) \\ \hline e^*(2) \\ e^*(3) \\ e^*(4) \\ e^*(5) \\ \hline e^*(6) \\ e^*(7) \\ \hline e^*(8) \end{array} \right] = \left[\begin{array}{c} y^*(0) \\ y^*(1) \\ \hline y^*(2) \\ y^*(3) \\ y^*(4) \\ y^*(5) \\ \hline y^*(6) \\ 0 \\ \hline 0 \end{array} \right] - \left[\begin{array}{ccc} x^*(0) & 0 & 0 \\ x^*(1) & x^*(0) & 0 \\ \hline x^*(2) & x^*(1) & x^*(0) \\ x^*(3) & x^*(2) & x^*(1) \\ x^*(4) & x^*(3) & x^*(2) \\ x^*(5) & x^*(4) & x^*(3) \\ \hline x^*(6) & x^*(5) & x^*(4) \\ 0 & x^*(6) & x^*(5) \\ \hline 0 & 0 & x^*(6) \end{array} \right] \left[\begin{array}{c} c_0 \\ c_1 \\ c_2 \end{array} \right] \end{array} \quad (8.3.4)$$

or, in general,

$$\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{c} \quad (8.3.5)$$

where the exact form of \mathbf{e} , \mathbf{y} , and \mathbf{X} depends on the range $N_i \leq n \leq N_f$ of measurements to be used, which in turn determines the range of summation

$$E = \sum_{n=N_i}^{N_f} |e(n)|^2 = \mathbf{e}^H \mathbf{e} \quad (8.3.6)$$

in the LS criterion. The LS FIR filter is found by solving the LS normal equations

$$(\mathbf{X}^H \mathbf{X}) \mathbf{c}_{ls} = \mathbf{X}^H \mathbf{y} \quad (8.3.7)$$

or

$$\hat{\mathbf{R}} \mathbf{c}_{ls} = \hat{\mathbf{d}} \quad (8.3.8)$$

with an LS error of

$$E_{ls} = E_y - \hat{\mathbf{d}}^H \mathbf{c}_{ls} \quad (8.3.9)$$

where E_y is the energy of the desired response signal. The elements of the time-average correlation matrix $\hat{\mathbf{R}}$ are given by

$$\hat{r}_{ij} = \tilde{\mathbf{x}}_i^H \tilde{\mathbf{x}}_j = \sum_{n=N_i}^{N_f} x(n+1-i)x^*(n+1-j) \quad 1 \leq i, j \leq M \quad (8.3.10)$$

where $\tilde{\mathbf{x}}_i$ are the columns of data matrix \mathbf{X} . A simple manipulation of (8.3.10) leads to

$$\hat{r}_{i+1,j+1} = \hat{r}_{ij} + x(N_i - i)x^*(N_i - j) - x(N_f + 1 - i)x^*(N_f + 1 - j) \quad 1 \leq i, j < M \quad (8.3.11)$$

which relates the elements of matrix $\hat{\mathbf{R}}$ that are located on the same diagonal. This property holds because the columns of \mathbf{X} are obtained by shifting the first column. The recursion in (8.3.11) suggests the following way of efficiently computing $\hat{\mathbf{R}}$:

1. Compute the first row of $\hat{\mathbf{R}}$ by using (8.3.10). This requires M dot products and a total of about $M(N_f - N_i)$ operations.
2. Compute the remaining elements in the upper triangular part of $\hat{\mathbf{R}}$, using (8.3.11). This required number of operations is proportional to M^2 .
3. Compute the lower triangular part of $\hat{\mathbf{R}}$, using the Hermitian symmetry relation $\hat{r}_{ji} = \hat{r}_{ij}^*$.

Notice that direct computation of the upper triangular part of $\hat{\mathbf{R}}$ using (8.3.10), that is, without the recursion, requires approximately $M^2N/2$ operations, which increases significantly for moderate or large values of M .

There are four ways to select the summation range $N_i \leq n \leq N_f$ that are used in LS filtering and prediction:

No windowing. If we set $N_i = M - 1$ and $N_f = N - 1$, we only use the available data and there are no distortions caused by forcing the data at the borders to artificial values.

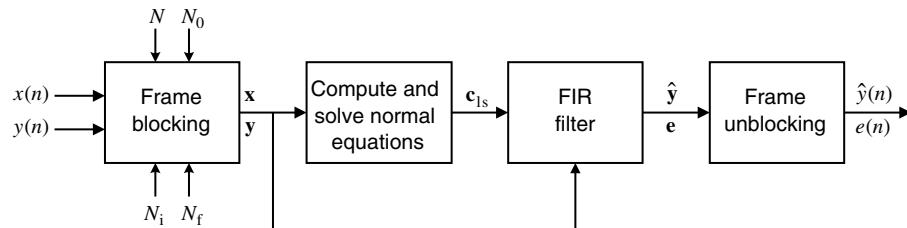
Prewindowing. This corresponds to $N_i = 0$ and $N_f = N - 1$ and is equivalent to setting the samples $x(0), x(-1), \dots, x(-M + 1)$ equal to zero. As a result, the term $x(M - i)x(M - j)$ does not appear in (8.3.11). This method is widely used in LS adaptive filtering.

Postwindowing. This corresponds to $N_i = M - 1$ and $N_f = N + M - 2$ and is equivalent to setting the samples $x(N), \dots, x(N + M - 2)$ equal to zero. As a result, the term $x(M - i)x(M - j)$ does not appear in (8.3.11). This method is not used very often for practical applications without prewindowing.

Full windowing. In this method, we impose both prewindowing and postwindowing (full windowing) to the input data and postwindowing to the desired response. The range of summation is from $N_i = 0$ to $N_f = N + M - 2$, and as a result of full windowing, Eq. (8.3.11) becomes $\hat{r}_{i+1,j+1} = \hat{r}_{ij}$. Therefore, the elements \hat{r}_{ij} , depend on $i - j$, and matrix $\hat{\mathbf{R}}$ is Toeplitz. In this case, the normal equations (8.2.12) can be obtained from the Wiener-Hopf equations (6.4.11) by replacing the theoretical autocorrelations with their estimated values (see Section 5.2).

Clearly, as $N \gg M$ the performance difference between the various methods becomes insignificant. The no-windowing and full-windowing methods are known in the signal processing literature as the *autocorrelation* and *covariance methods*, respectively (Makhoul 1975b). We avoid these terms because they can lead to misleading statistical interpretations. We notice that in the LS filtering problem, the data matrix \mathbf{X} is Toeplitz and the normal equations matrix $\hat{\mathbf{R}} = \mathbf{X}^H \mathbf{X}$ is the product of two Toeplitz matrices. However, $\hat{\mathbf{R}}$ is Toeplitz only in the full-windowing case when \mathbf{X} is banded Toeplitz. In all other cases $\hat{\mathbf{R}}$ is *near to Toeplitz* or $\hat{\mathbf{R}}$ is *close to Toeplitz* in a sense made precise in Morf, et al. (1977).

The matrix $\hat{\mathbf{R}}$ and vector $\hat{\mathbf{d}}$, for the various windowing methods, are computed by using the MATLAB function `[R, d] = lsmatvec(x, M, method, y)`, which is based on (8.3.10) and (8.3.11). Then the LS filter is computed by `c1s=R\d`. Figure 8.6 shows an FIR LSE filter operating in block processing mode.

**FIGURE 8.6**

Block processing implementation of an FIR LSE filter.

EXAMPLE 8.3.1. To illustrate the design of least-squares FIR filters, suppose that we have a set of measurements of $x(n)$ and $y(n)$ for $0 \leq n \leq N - 1$ with $N = 100$ that have been generated by the difference equation

$$y(n) = 0.5x(n) + 0.5x(n - 1) + v(n)$$

The input $x(n)$ and the additive noise $v(n)$ are uncorrelated processes from a normal (Gaussian) distribution with mean $E\{x(n)\} = E\{v(n)\} = 0$ and variance $\sigma_x^2 = \sigma_v^2 = 1$. Fitting the model

$$\hat{y}(n) = h(0)x(n) + h(1)x(n - 1)$$

to the measurements with the no-windowing LS criterion, we obtain

$$\mathbf{c}_{ls} = \begin{bmatrix} 0.5361 \\ 0.5570 \end{bmatrix} \quad \hat{\sigma}_e^2 = 1.0419 \quad \hat{\sigma}_e^2 \hat{\mathbf{R}}^{-1} = \begin{bmatrix} 0.0073 & -0.0005 \\ -0.0005 & 0.0071 \end{bmatrix}$$

using (8.3.7), (8.3.9), (8.2.44), and (8.2.42). If the mean of the additive noise is nonzero, for example, if $E\{v(n)\} = 1$, we get

$$\mathbf{c}_{ls} = \begin{bmatrix} 0.4889 \\ 0.5258 \end{bmatrix} \quad \hat{\sigma}_e^2 = 1.8655 \quad \hat{\sigma}_e^2 \hat{\mathbf{R}}^{-1} = \begin{bmatrix} 0.0131 & -0.0009 \\ -0.0009 & 0.0127 \end{bmatrix}$$

which shows that the variance of the estimates, that is, the diagonal elements of $\hat{\sigma}_e^2 \hat{\mathbf{R}}^{-1}$, increases significantly. Suppose now that the recording device introduces an outlier in the input data at $x(30) = 20$. The estimated LS model and its associated statistics are given by

$$\mathbf{c}_{ls} = \begin{bmatrix} 0.1796 \\ 0.1814 \end{bmatrix} \quad \hat{\sigma}_e^2 = 1.6270 \quad \hat{\sigma}_e^2 \hat{\mathbf{R}}^{-1} = \begin{bmatrix} 0.0030 & 0.0000 \\ 0.0000 & 0.0030 \end{bmatrix}$$

Similarly, when an outlier is present in the output data, for example, at $y(30) = 20$, then the LS model and its statistics are

$$\mathbf{c}_{ls} = \begin{bmatrix} 0.6303 \\ 0.4653 \end{bmatrix} \quad \hat{\sigma}_e^2 = 5.0979 \quad \hat{\sigma}_e^2 \hat{\mathbf{R}}^{-1} = \begin{bmatrix} 0.0357 & -0.0025 \\ -0.0025 & 0.0347 \end{bmatrix}$$

In general, LS estimates are very sensitive to colored additive noise and outliers (Ljung 1987). Note that all the LS solutions in this example were produced with one sample realization $x(n)$ and that the results will vary for any other realizations.

LS inverse filters. Given a causal filter with impulse response $g(n)$, its inverse filter $h(n)$ is specified by $g(n) * h(n) = \delta(n - n_0)$, $n_0 \geq 0$. We focus on causal inverse filters, which are often infinite impulse response (IIR), and we wish to approximate them by some FIR filter $c_{ls}(n) = h^*(n)$ that is optimum according to the LS criterion. In this case, the actual impulse response $g(n) * c_{ls}^*(n)$ of the combined system deviates from the desired response $\delta(n - n_0)$, resulting in an error $e(n)$. The convolution equation

$$e(n) = \delta(n - n_0) - \sum_{k=0}^M c_{ls}^*(k) g(n - k) \quad (8.3.12)$$

can be formulated in matrix form as follows for $M = 2$ and $N = 6$

$$\begin{bmatrix} e^*(0) \\ e^*(1) \\ \hline e^*(2) \\ e^*(3) \\ e^*(4) \\ e^*(5) \\ e^*(6) \\ \hline e^*(7) \\ e^*(8) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \end{bmatrix} - \begin{bmatrix} g^*(0) & 0 & 0 \\ g^*(1) & g^*(0) & 0 \\ \hline g^*(2) & g^*(1) & g^*(0) \\ g^*(3) & g^*(2) & g^*(1) \\ g^*(4) & g^*(3) & g^*(2) \\ g^*(5) & g^*(4) & g^*(3) \\ g^*(6) & g^*(5) & g^*(4) \\ \hline 0 & g^*(6) & g^*(5) \\ 0 & 0 & g^*(6) \end{bmatrix} \begin{bmatrix} c_{ls}(0) \\ c_{ls}(1) \\ c_{ls}(2) \end{bmatrix}$$

assuming that $n_0 = 0$. In general,

$$\mathbf{e} = \delta_i - \mathbf{G}\mathbf{c}_{ls}^{(i)} \quad (8.3.13)$$

where δ_i is a vector whose i th element is 1 and whose remaining elements are all zero. The LS inverse filter and the corresponding error are given by

$$(\mathbf{G}^H \mathbf{G})\mathbf{c}_{ls}^{(i)} = \mathbf{G}^H \delta_i \quad (8.3.14)$$

$$\text{and } E_{ls}^{(i)} = 1 - \delta_i^H \mathbf{G}\mathbf{c}_{ls}^{(i)} = 1 - g^*(i)c_{ls}^{(i)}(i) \quad 0 \leq i \leq M + N \quad (8.3.15)$$

respectively.

Using the projection operators (8.2.29) and (8.2.30), we can express the LS error as

$$E_{ls}^{(i)} = \delta_i^H (\mathbf{P} - \mathbf{I})^H (\mathbf{P} - \mathbf{I}) \delta_i \quad (8.3.16)$$

where

$$\mathbf{P} = \mathbf{G}(\mathbf{G}^H \mathbf{G})^{-1} \mathbf{G}^H \quad (8.3.17)$$

The total error for all possible delays $0 \leq i \leq N + M$ can be written as

$$E_{total} = \sum_{i=0}^{N+M} E_{ls}^{(i)} = \text{tr}[\mathbf{D}^H (\mathbf{P} - \mathbf{I})^H (\mathbf{P} - \mathbf{I}) \mathbf{D}] \quad (8.3.18)$$

where

$$\mathbf{D} \triangleq [\delta_0 \ \delta_1 \ \delta_2 \ \cdots \ \delta_{N+M}] = \mathbf{I}$$

is the $(N + M + 1) \times (N + M + 1)$ identity matrix. Since $\mathbf{D} = \mathbf{I}$, $\mathbf{P} = \mathbf{P}^H$, and $\mathbf{P}^2 = \mathbf{P}$, we obtain

$$E_{total} = \text{tr}[\mathbf{D}^H (\mathbf{P} - \mathbf{I})^H (\mathbf{P} - \mathbf{I}) \mathbf{D}] = \text{tr}(\mathbf{I} - \mathbf{P}) = \text{tr}(\mathbf{I}) - \text{tr}(\mathbf{P})$$

or

$$E_{total} = N \quad (8.3.19)$$

because $\text{tr}(\mathbf{I}) = N + M + 1$ and

$$\text{tr}(\mathbf{P}) = \text{tr}[\mathbf{G}(\mathbf{G}^H \mathbf{G})^{-1} \mathbf{G}^H] = \text{tr}[\mathbf{G}^H \mathbf{G}(\mathbf{G}^H \mathbf{G})^{-1}] = M + 1 \quad (8.3.20)$$

Hence, E_{total} depends on the length $N + 1$ of the filter $g(n)$ and is independent of the length $M + 1$ of the inverse filter $c_{ls}(n)$. If the minimum $E_{ls}^{(i)}$, for a given N , occurs at delay $i = i_0$, we have

$$E_{ls}^{(i_0)} \leq \frac{N}{N + M + 1} \quad (8.3.21)$$

which shows that $E_{ls}^{(i_0)} \rightarrow 0$ as $M \rightarrow \infty$ (Claerbout and Robinson 1963).

EXAMPLE 8.3.2. Suppose that $g(n) = \delta(n) - \alpha\delta(n - 1)$, where α is a real constant. The exact inverse filter is

$$H(z) = \frac{1}{1 - \alpha z^{-1}} \Rightarrow h(n) = \alpha^n u(n)$$

and is minimum-phase only if $-1 < \alpha < 1$. The inverse LS filter for $M = 1$ and $N \geq 2$ is obtained by applying (8.3.14) with

$$\mathbf{G} = \begin{bmatrix} 1 & 0 \\ -\alpha & 1 \\ 0 & -\alpha \end{bmatrix} \quad \text{and} \quad \boldsymbol{\delta} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

The normal equations are

$$\begin{bmatrix} 1 + \alpha^2 & -\alpha \\ -\alpha & 1 + \alpha^2 \end{bmatrix} \begin{bmatrix} c_{ls}(0) \\ c_{ls}(1) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (8.3.22)$$

leading to the LS inverse filter

$$c_{ls}(0) = \frac{1 + \alpha^2}{1 + \alpha^2 + \alpha^4} \quad c_{ls}(1) = \frac{\alpha}{1 + \alpha^2 + \alpha^4}$$

with LS error

$$E_{ls} = 1 - c_{ls}(0) = \frac{\alpha^4}{1 + \alpha^2 + \alpha^4}$$

The system function of the LS inverse filter is

$$H_{ls}(z) = \frac{1 + \alpha^2}{1 + \alpha^2 + \alpha^4} \left(1 + \frac{\alpha}{1 + \alpha^2} z^{-1} \right)$$

and has a zero at $z_1 = -\alpha/(1 + \alpha^2) = -1/(\alpha + \alpha^{-1})$. Since $|z_1| < 1$ for any value of α , the LS inverse filter is minimum-phase even if $g(n)$ is not. This stems from the fact that the normal equations (8.3.22) specify a one-step forward linear predictor with a correlation matrix that is Toeplitz and positive definite for any value of α (see Section 7.4).

8.4 LINEAR LEAST-SQUARES SIGNAL ESTIMATION

We now discuss the application of the LS method to general signal estimation, FLP, BLP, and combined forward and backward linear prediction. The reader is advised to review Section 6.5, which provides a detailed discussion of the same problems for the MMSE criterion. The presentation in this section closely follows the viewpoint and notation in Section 6.5.

8.4.1 Signal Estimation and Linear Prediction

Suppose that we wish to compute the linear LS signal estimator $c_k^{(i)}$ defined by

$$e^{(i)}(n) = \sum_{k=0}^M c_k^{(i)*} x(n-k) = \mathbf{c}^{(i)H} \bar{\mathbf{x}}(n) \quad \text{with } c_i^{(i)} \triangleq 1 \quad (8.4.1)$$

from the data $x(n)$, $0 \leq n \leq N-1$. Using (8.4.1) and following the process that led to (8.3.4), we obtain

$$\mathbf{e}^{(i)} = \bar{\mathbf{X}} \mathbf{c}^{(i)} \quad (8.4.2)$$

$$\text{where } \bar{\mathbf{X}} = \begin{bmatrix} x^*(0) & 0 & \cdots & 0 \\ x^*(1) & x^*(0) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ x^*(M) & x^*(M-1) & \cdots & x^*(0) \\ \vdots & \vdots & & \vdots \\ x^*(N-1) & x^*(N-2) & \cdots & x^*(N-M-1) \\ 0 & x^*(N-1) & \cdots & x^*(N-M) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x^*(N-1) \end{bmatrix} \quad (8.4.3)$$

is the combined data and desired response matrix with all the unavailable samples set equal to zero (full windowing). Matrix $\bar{\mathbf{X}}$ can be partitioned columnwise as

$$\bar{\mathbf{X}} = [\mathbf{X}_1 \ \mathbf{y} \ \mathbf{X}_2] \quad (8.4.4)$$

where \mathbf{y} , the desired response, is the i th column of $\bar{\mathbf{X}}$. Using (8.4.4), we can easily show that the LS signal estimator $\mathbf{c}_{ls}^{(i)}$ and the associated LS error $E_{ls}^{(i)}$ are determined by

$$(\bar{\mathbf{X}}^H \bar{\mathbf{X}}) \mathbf{c}_{ls}^{(i)} = \begin{bmatrix} \mathbf{0} \\ E_{ls}^{(i)} \\ \mathbf{0} \end{bmatrix} \quad (8.4.5)$$

where $E_{ls}^{(i)}$ is the i th element of the right-hand side vector (see Problem 8.3). If we define the time-average correlation matrix

$$\bar{\mathbf{R}} \triangleq \bar{\mathbf{X}}^H \bar{\mathbf{X}} \quad (8.4.6)$$

and use the augmented normal equations in (8.4.5), we obtain a set of equations that have the same form as (6.5.12), the equations for the MMSE signal estimator. Therefore, after we have computed $\bar{\mathbf{R}}$, using the command `Rbar=lsmatvec(x,M+1,method)`, we can use the steps in Table 6.3 to compute the LS forward linear predictor (FLP), the backward linear predictor (BLP), the symmetric smoother, or any other signal estimator with delay i . Again, we use the standard notation $E_{ls}^{(0)} = E^f$ and $\mathbf{c}_{ls}^{(0)} = \mathbf{a}$ for the FLP and $E_{ls}^{(M)} = E^b$ and $\mathbf{c}_{ls}^{(M)} = \mathbf{b}$ for the BLP.

All formulas given in Section 6.5 hold for LS signal estimators if the matrix $\mathbf{R}(n)$ is replaced by $\bar{\mathbf{R}}$. However, we stress that although the optimum MMSE signal estimator $\mathbf{c}_o^{(i)}(n)$ is a deterministic vector, the LS signal estimator $\mathbf{c}_{ls}^{(i)}$ is a random vector that is a function of the random measurements $x(n)$, $0 \leq n \leq N - 1$. In the full-windowing case, matrix $\bar{\mathbf{R}}$ is Toeplitz; if it is also positive definite, then the FLP is minimum-phase. Although the use of full windowing leads to these nice properties, it also creates some “edge effects” and bias in the estimates because we try to estimate some signal values using values that are not part of the signal by forcing the samples leading and lagging the available data measurements to zero.

EXAMPLE 8.4.1. Suppose that we are given the signal segment $x(n) = \alpha^n$, $0 \leq n \leq N$, where α is an arbitrary complex-valued constant. Determine the first-order one-step forward linear predictor, using the full-windowing and no-windowing methods.

Solution. We start by forming the combined desired response and data matrix

$$\bar{\mathbf{X}}^H = \begin{bmatrix} x(0) & x(1) & \cdots & x(N) & 0 \\ 0 & x(0) & \cdots & x(N-1) & x(N) \end{bmatrix}$$

For the full-windowing method, the matrix

$$\bar{\mathbf{R}} = \bar{\mathbf{X}}^H \bar{\mathbf{X}} = \begin{bmatrix} \hat{r}_x(0) & \hat{r}_x(1) \\ \hat{r}_x^*(1) & \hat{r}_x(0) \end{bmatrix}$$

is Toeplitz with elements

$$\hat{r}_x(0) = \sum_{n=0}^N |x(n)|^2 = \sum_{n=0}^N |\alpha|^{2n} = \frac{1 - |\alpha|^{2(N+1)}}{1 - |\alpha|^2}$$

$$\text{and } \hat{r}_x(1) = \sum_{n=1}^N x(n)x^*(n-1) = \sum_{n=1}^N \alpha^n (\alpha^*)^{n-1} = \alpha^* \frac{1 - |\alpha|^{2N}}{1 - |\alpha|^2}$$

Therefore, we have

$$\begin{bmatrix} \hat{r}_x(0) & \hat{r}_x(1) \\ \hat{r}_x^*(1) & \hat{r}_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1^{(1)} \end{bmatrix} = \begin{bmatrix} E^f \\ 0 \end{bmatrix}$$

whose solution gives

$$a_1^{(1)} = -\frac{\hat{r}_x^*(1)}{\hat{r}_x(0)} = -\alpha \frac{1 - |\alpha|^{2N}}{1 - |\alpha|^{2(N+1)}}$$

and

$$E_1^f = \hat{r}_x(0) + \hat{r}_x(1)a_1^{(1)} = \frac{1 - |\alpha|^{2(2N+1)}}{1 - |\alpha|^{2(N+1)}}$$

Since for every sequence $|\hat{r}_x(l)| \leq |\hat{r}_x(0)|$, we have $|a_1^{(1)}| \leq 1$; that is, the obtained prediction error filter always is minimum-phase. Furthermore, if $|\alpha| < 1$, then $\lim_{N \rightarrow \infty} a_1^{(1)} = -\alpha$ and $\lim_{N \rightarrow \infty} E_1^f = 1 = x(0)$. In the no-windowing case, the matrix

$$\bar{\mathbf{R}} = \bar{\mathbf{X}}^H \bar{\mathbf{X}} = \begin{bmatrix} \hat{r}_{11} & \hat{r}_{12} \\ \hat{r}_{12}^* & \hat{r}_{22} \end{bmatrix}$$

is Hermitian but not Toeplitz with elements

$$\begin{aligned} \hat{r}_{11} &= \sum_{n=1}^N |x(n)|^2 = |\alpha|^2 \frac{1 - |\alpha|^{2N}}{1 - |\alpha|^2} & \hat{r}_{22} &= \sum_{n=0}^{N-1} |x(n)|^2 = \frac{1 - |\alpha|^{2N}}{1 - |\alpha|^2} \\ \hat{r}_{12} &= \sum_{n=1}^N x(n)x^*(n-1) = \alpha^* \frac{1 - |\alpha|^{2N}}{1 - |\alpha|^2} \end{aligned}$$

Solving the linear system

$$\begin{bmatrix} \hat{r}_{11} & \hat{r}_{12} \\ \hat{r}_{12}^* & \hat{r}_{22} \end{bmatrix} \begin{bmatrix} 1 \\ a_1^{(1)} \end{bmatrix} = \begin{bmatrix} \bar{E}_1^f \\ 0 \end{bmatrix}$$

we obtain

$$\bar{a}_1^{(1)} = -\frac{\hat{r}_{12}^*}{\hat{r}_{22}} = -\alpha$$

and

$$\bar{E}_1^f = \hat{r}_{11} + \hat{r}_{12}\bar{a}_1^{(1)} = 0$$

We see that the no-windowing method provides a perfect linear predictor because there is no distortion due to windowing. However, the obtained prediction error filter is minimum-phase only when $|\alpha| < 1$.

EXAMPLE 8.4.2. To illustrate the statistical properties of least-squares FLP, we generate $K = 500$ realizations of the MA(1) process $x(n) = w(n) + \frac{1}{2}w(n-1)$, where $w(n) \sim \text{WN}(0, 1)$ (see Example 6.5.2). Each realization $x(\zeta_i, n)$ has duration $N = 100$ samples. We use these data to design an $M = 2$ order FLP, using the no-windowing LS method. The estimated mean and variance of the obtained K FLP vectors are

$$\text{Mean}\{\mathbf{a}(\zeta_i)\} = \begin{bmatrix} -0.4695 \\ 0.1889 \end{bmatrix} \quad \text{and} \quad \text{var}\{\mathbf{a}(\zeta_i)\} = \begin{bmatrix} 0.0086 \\ 0.0092 \end{bmatrix}$$

whereas the average of the variances $\hat{\sigma}_e^2$ is 0.9848. We notice that both means are close to the theoretical values obtained in Example 6.5.2. The covariance matrix of a given LS estimate \mathbf{a}_{LS} was found to be

$$\hat{\sigma}_e^2 \hat{\mathbf{R}}^{-1} = \begin{bmatrix} 0.0099 & -0.0043 \\ -0.0043 & 0.0099 \end{bmatrix}$$

whose diagonal elements are close to the components of $\text{var}\{\mathbf{a}\}$, as expected. The bias in the estimate \mathbf{a}_{LS} results from the fact that the residuals in the LS equations are correlated with each other (see Problem 8.4).

8.4.2 Combined Forward and Backward Linear Prediction (FBLP)

For stationary stochastic processes, the optimum MMSE forward and backward linear predictors have even conjugate symmetry, that is,

$$\mathbf{a}_o = \mathbf{J}\mathbf{b}_o^* \tag{8.4.7}$$

because both directions of time have the same second-order statistics. Formally, this property stems from the Toeplitz structure of the autocorrelation matrix (see Section 6.5). However, we could possibly improve performance by minimizing the total forward and backward squared error

$$E^{fb} = \sum_{n=N_i}^{N_f} \{|e^f(n)|^2 + |e^b(n)|^2\} = (\mathbf{e}^f)^H \mathbf{e}^f + (\mathbf{e}^b)^H \mathbf{e}^b \quad (8.4.8)$$

under the constraint

$$\mathbf{a}^{fb} \triangleq \mathbf{a} = \mathbf{J}\mathbf{b}^* \quad (8.4.9)$$

The FLP and BLP overdetermined sets of equations are

$$\mathbf{e}^f = \bar{\mathbf{X}} \begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix} \quad \text{and} \quad \mathbf{e}^b = \bar{\mathbf{X}} \begin{bmatrix} \mathbf{b} \\ 1 \end{bmatrix} \quad (8.4.10)$$

$$\text{or} \quad \mathbf{e}^f = \bar{\mathbf{X}} \begin{bmatrix} 1 \\ \mathbf{a}^{fb} \end{bmatrix} \quad \text{and} \quad \mathbf{e}^{b*} = \bar{\mathbf{X}}^* \begin{bmatrix} \mathbf{b}^* \\ 1 \end{bmatrix} = \bar{\mathbf{X}}^* \mathbf{J} \begin{bmatrix} 1 \\ \mathbf{a}^{fb} \end{bmatrix} \quad (8.4.11)$$

where we have used (8.4.9) and the property $\mathbf{JJ}^* = \mathbf{I}$ of the exchange matrix. If we combine the above two equations as

$$\begin{bmatrix} \mathbf{e}^f \\ \mathbf{e}^{b*} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{X}} \\ \bar{\mathbf{X}}^* \mathbf{J} \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{a}^{fb} \end{bmatrix} \quad (8.4.12)$$

then the forward-backward linear predictor that minimizes E^{fb} is given by (see Problem 8.5)

$$\begin{aligned} & \begin{bmatrix} \bar{\mathbf{X}} \\ \bar{\mathbf{X}}^* \mathbf{J} \end{bmatrix}^H \begin{bmatrix} \bar{\mathbf{X}} \\ \bar{\mathbf{X}}^* \mathbf{J} \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{a}_{ls}^{fb} \end{bmatrix} = \begin{bmatrix} E_{ls}^{fb} \\ \mathbf{0} \end{bmatrix} \\ & \text{or} \quad (\bar{\mathbf{X}}^H \bar{\mathbf{X}} + \mathbf{J} \bar{\mathbf{X}}^T \bar{\mathbf{X}}^* \mathbf{J}) \begin{bmatrix} 1 \\ \mathbf{a}_{ls}^{fb} \end{bmatrix} = \begin{bmatrix} E_{ls}^{fb} \\ \mathbf{0} \end{bmatrix} \end{aligned} \quad (8.4.13)$$

which can be solved by using the steps described in Table 6.3. The time-average forward-backward correlation matrix

$$\hat{\mathbf{R}}_{fb} \triangleq \bar{\mathbf{X}}^H \bar{\mathbf{X}} + \mathbf{J} \bar{\mathbf{X}}^T \bar{\mathbf{X}}^* \mathbf{J} \quad (8.4.14)$$

with elements

$$\hat{r}_{ij}^{fb} = \hat{r}_{ij} + \hat{r}_{M-i,M-j}^* \quad 0 \leq i, j \leq M \quad (8.4.15)$$

is persymmetric; that is, $\mathbf{J} \hat{\mathbf{R}}_{fb} \mathbf{J} = \hat{\mathbf{R}}_{fb}^*$ and its elements are conjugate symmetric about both main diagonals. In MATLAB we compute $\hat{\mathbf{R}}_{fb}$ by these commands:

```
Rbar=lsmatvec(x,M+1,method)
Rfb=Rbar+fliipud(fliplr(conj(Rbar)))
```

The FBLP method is used with *no windowing* and was originally introduced independently by Ulrych and Clayton (1976) and Nuttall (1976) as a spectral estimation technique under the name *modified covariance method* (see Section 9.2). If we use full windowing, then $\mathbf{a}^{fb} = (\mathbf{a} + \mathbf{J}\mathbf{b}^*)/2$ (see Problem 8.6).

8.4.3 Narrowband Interference Cancelation

Several practical applications require the removal of *narrowband interference (NBI)* from a wideband desired signal corrupted by additive white noise. For example, ground and

foliage-penetrating radars operate from 0.01 to 1 GHz and use either an impulse or a chirp waveform. To achieve high resolution, these waveforms are extremely wideband, occupying at least 100 MHz within the range of 0.01 to 1 GHz. However, these frequency ranges are extensively used by TV and FM stations, cellular phones, and other relatively narrowband (less than 1 MHz) radio-frequency (RF) sources. Clearly, these sources spoil the radar returns with narrowband RF interference (Miller et al. 1997). Since the additive noise is often due to the sensor circuitry, it will be referred to as *sensor thermal noise*. Next we provide a practical solution to this problem, using an LS linear predictor. Suppose that the corrupted signal $x(n)$ is given by

$$x(n) = s(n) + y(n) + v(n) \quad (8.4.16)$$

where

$$s(n) = \text{signal of interest} \quad (8.4.17)$$

$$y(n) = \text{narrowband interference}$$

$$v(n) = \text{thermal (white) noise}$$

are the individual components, assumed to be stationary stochastic processes.

We wish to design an NBI canceler that estimates and rejects the interference signal $y(n)$ from the signal $x(n)$, while preserving the signal of interest $s(n)$. Since signals $y(n)$ and $x(n)$ are correlated, we can form an estimate of the NBI using the optimum linear estimator

$$\hat{y}(n) = \mathbf{c}_o^H \mathbf{x}(n - D) \quad (8.4.18)$$

where

$$\mathbf{Rc}_o = \mathbf{d} \quad (8.4.19)$$

$$\mathbf{R} = E\{\mathbf{x}(n - D)\mathbf{x}^H(n - D)\} \quad (8.4.20)$$

$$\mathbf{d} = E\{\mathbf{x}(n - D)\mathbf{y}^*(n)\} \quad (8.4.21)$$

and D is an integer delay whose use will be justified shortly. Note that if $D = 1$, then (8.4.18) is the LS forward linear predictor. If $\hat{y}(n) = y(n)$, the output of the canceler is $x(n) - \hat{y}(n) = s(n) + v(n)$; that is, the NBI is completely excised, and the desired signal is corrupted by white noise only and is said to be thermal noise-limited.

Since, in practice, the required second-order moments are not available, we need to use an LS estimator instead. However, the quantity $\mathbf{X}^H \mathbf{y}$ in (8.2.21) requires the NBI signal $y(n)$, which is also not available. To overcome this obstacle, consider the optimum MMSE D -step forward linear predictor

$$e^f(n) = x(n) + \mathbf{a}^H \mathbf{x}(n - D) \quad (8.4.22)$$

$$\mathbf{Ra} = -\mathbf{r}^f \quad (8.4.23)$$

where \mathbf{R} is given by (8.4.20) and

$$\mathbf{r}^f = E\{\mathbf{x}(n - D)\mathbf{x}^*(n)\} \quad (8.4.24)$$

In many NBI cancelation applications, the components of the observed signal have the following properties:

1. The desired signal $s(n)$, the NBI $y(n)$, and the thermal noise $v(n)$ are mutually uncorrelated.
2. The thermal noise $v(n)$ is white; that is, $r_v(l) = \sigma_v^2 \delta(l)$.
3. The desired signal $s(n)$ is wideband and therefore has a short correlation length; that is, $r_s(l) = 0$ for $|l| \geq D$.
4. The NBI has a long correlation length; that is, its autocorrelation takes significant values over the range $0 \leq |l| \leq M$ for $M > D$.

In practice, the second and third properties mean that the desired signal and the thermal noise are approximately uncorrelated after a certain small lag. These are precisely the properties exploited by the canceler to separate the NBI from the desired signal and the background noise.

As a result of the first assumption, we have

$$E\{x(n-k)y^*(n)\} = E\{y(n-k)y^*(n)\} = r_y(k) \quad \text{for all } k \quad (8.4.25)$$

and

$$r_x(l) = r_s(l) + r_y(l) + r_v(l) \quad (8.4.26)$$

Making use of the second and third assumptions, we have

$$r_x(l) = r_y(l) \quad \text{for } l \neq 0, 1, \dots, D-1 \quad (8.4.27)$$

The exclusion of the lags for $l \neq 0, 1, \dots, D-1$ in \mathbf{r} and \mathbf{d} is critical, and we have arranged for that by forcing the filter and the predictor to form their estimates using the *delayed* data vector $\mathbf{x}(n-D)$. From (8.4.21), (8.4.24), and (8.4.27), we conclude that $\mathbf{d} = \mathbf{r}^f$ and therefore $\mathbf{c}_o = \mathbf{a}_o$. Thus, the optimum NBI estimator \mathbf{c}_o is equal to the D -step linear predictor \mathbf{a}_o , which can be determined exclusively from the input signal $x(n)$. The cleaned signal is

$$x(n) - \hat{y}(n) = x(n) + \mathbf{a}_o^H \mathbf{x}(n-D) = e^f(n) \quad (8.4.28)$$

which is identical to the D -step forward prediction error. This leads to the linear prediction NBI canceler shown in Figure 8.7.

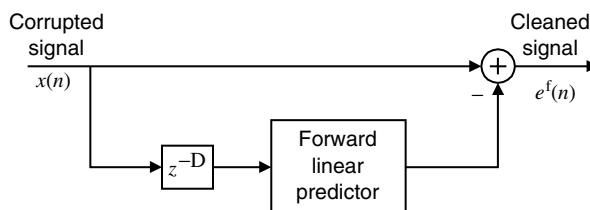


FIGURE 8.7
Block diagram of linear prediction NBI canceler.

To illustrate the performance of the linear prediction NBI canceler, we consider an impulse radar operating in a location with commercial radio and TV stations. The desired signal is a short-duration impulse corrupted by additive thermal noise and NBI (see Figure 8.8). The spectrum of the NBI is shown in Figure 8.9. We use a block of data ($N = 4096$) to design an FBLP with $D = 1$ and $M = 100$ coefficients, using the LS criterion with no windowing. Then we compute the cleaned signal, using (8.4.28). The cleaned signal, its spectrum, and the magnitude response of the NBI canceler are shown in Figures 8.8 and 8.9. We see that the canceler acts as a notch filter that optimally puts notches at the peaks of the NBI. A detailed description of the design of optimum least-squares NBI cancelers is given in Problem 8.27.

8.5 LS COMPUTATIONS USING THE NORMAL EQUATIONS

The solution of the normal equations for both MMSE and LSE estimation problems is computed by using the same algorithms. The key difference is that in MMSE estimation \mathbf{R} and \mathbf{d} are known, whereas in LSE estimation they need to be computed from the observed input and desired response signal samples. Therefore, it is natural to want to take advantage of the same algorithms developed for MMSE estimation in Chapter 7, whenever possible. However, keep in mind that despite algorithmic similarities, there are fundamental differences between the two classes of estimators that are dictated by the different nature of the the criteria of performance (see Section 8.1). In this section, we show how the computational algorithms and structures developed for linear MMSE estimation can be applied to linear LSE estimation, relying heavily on the material presented in Chapter 7.

The computation of a general linear LSE estimator requires the solution of a linear system

$$\hat{\mathbf{R}}\mathbf{c}_{ls} = \hat{\mathbf{d}} \quad (8.5.1)$$

where the time-average correlation matrix $\hat{\mathbf{R}}$ is Hermitian and positive definite [see (8.2.25)]. We can solve (8.5.1) by using the LDL^H or the Cholesky decomposition introduced in Section 6.3. The computation of linear LSE estimators involves the steps summarized in Table 8.1. We again stress that the major computational effort is involved in the computation of $\hat{\mathbf{R}}$ and $\hat{\mathbf{d}}$.

Steps 2 and 3 in (6.3.16) can be facilitated by a single extended LDL^H decomposition. To this end, we form the augmented data matrix

$$\bar{\mathbf{X}} = [\mathbf{X} \mathbf{y}] \quad (8.5.2)$$

and compute its time-average correlation matrix

$$\bar{\mathbf{R}} = \bar{\mathbf{X}}^H \bar{\mathbf{X}} = \begin{bmatrix} \mathbf{X}^H \mathbf{X} & \mathbf{X}^H \mathbf{y} \\ \mathbf{y}^H \mathbf{X} & \mathbf{y}^H \mathbf{y} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{R}} & \hat{\mathbf{d}} \\ \hat{\mathbf{d}}^H & E_y \end{bmatrix} \quad (8.5.3)$$

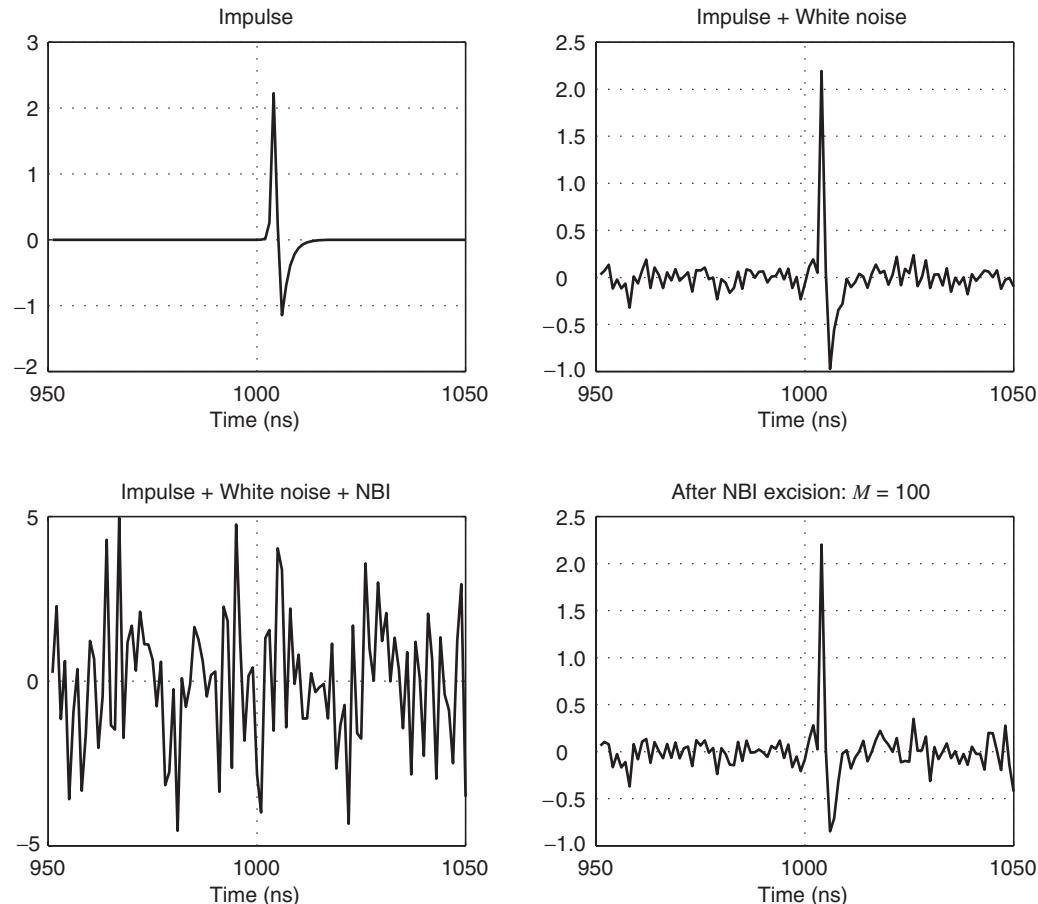


FIGURE 8.8

NBI cancellation: time-domain results.

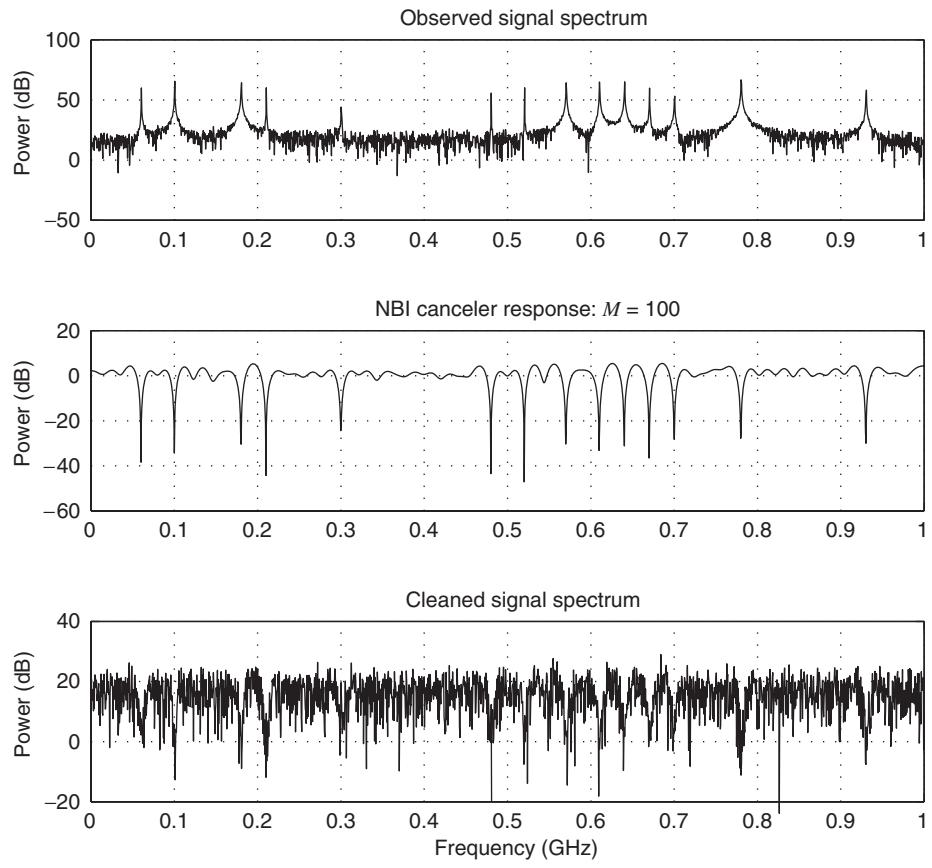


FIGURE 8.9

NBI canceler: frequency-domain results.

TABLE 8.1

Comparison between the LDL^H and Cholesky decomposition methods for the solution of normal equations.

Step	LDL^H decomposition	Cholesky decomposition	Description
1	$\hat{\mathbf{R}} = \mathbf{X}^H \mathbf{X}$, $\hat{\mathbf{d}} = \mathbf{X}^H \mathbf{y}$		Normal equations $\hat{\mathbf{R}} \mathbf{c}_{ls} = \hat{\mathbf{d}}$
2	$\hat{\mathbf{R}} = LDL^H$	$\hat{\mathbf{R}} = \mathcal{L} \mathcal{L}^H$	Triangular decomposition
3	$LD\mathbf{k} = \hat{\mathbf{d}}$	$\mathcal{L}\tilde{\mathbf{k}} = \hat{\mathbf{d}}$	Forward substitution $\rightarrow \mathbf{k}$ or $\tilde{\mathbf{k}}$
4	$\mathbf{L}^H \mathbf{c}_{ls} = \mathbf{k}$	$\mathcal{L}^H \mathbf{c}_{ls} = \tilde{\mathbf{k}}$	Backward substitution $\rightarrow \mathbf{c}_{ls}$
5	$E_{ls} = \mathbf{E}_y - \mathbf{k}^H \mathbf{D} \mathbf{k}$	$E_{ls} = \mathbf{E}_y - \tilde{\mathbf{k}}^H \tilde{\mathbf{k}}$	LSE computation
6	$\mathbf{e}_{ls} = \mathbf{y} - \mathbf{X} \mathbf{c}_{ls}$	$\mathbf{e}_{ls} = \mathbf{y} - \mathbf{X} \mathbf{c}_{ls}$	Computation of residuals

We then can show (see Problem 8.9) that the LDL^H decomposition of $\bar{\mathbf{R}}$ is given by

$$\bar{\mathbf{R}} = \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{k}^H & 1 \end{bmatrix} \begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0}^H & E_{ls} \end{bmatrix} \begin{bmatrix} \mathbf{L}^H & \mathbf{k}^H \\ \mathbf{0}^H & 1 \end{bmatrix} \quad (8.5.4)$$

and thus provides the vector \mathbf{k} and the LSE E_{ls} . Therefore, we can solve the normal equations (8.5.1), using the LDL^H decomposition of $\bar{\mathbf{R}}$ to compute \mathbf{L} and \mathbf{k} and then solving $\mathbf{L}^H \mathbf{c}_{ls} = \mathbf{k}$ to compute \mathbf{c}_{ls} .

A careful inspection of the design equations for the general, m th-order, MMSE and LSE estimators, derived in Chapter 6 and summarized in Table 8.2, shows that the LSE

TABLE 8.2

Comparison between the MMSE and LSE normal equations for general linear estimation.

	MMSE	LSE
Available information	$\mathbf{R}_m(n), \mathbf{d}_m(n)$	$\{\mathbf{x}_m(n), y(n), n_i \leq n \leq n_f\}$
Normal equations	$\mathbf{R}_m(n)\mathbf{c}_m(n) = \mathbf{d}_m(n)$	$\hat{\mathbf{R}}_m\mathbf{c}_m = \hat{\mathbf{d}}_m$
Minimum error	$P_m(n) = P_y(n) - \mathbf{d}_m^H(n)\mathbf{c}_m(n)$	$E_m = E_y - \hat{\mathbf{d}}_m^H\mathbf{c}_m$
Correlation matrix	$\mathbf{R}_m(n) \triangleq E\{\mathbf{x}_m(n)\mathbf{x}_m^H(n)\}$	$\hat{\mathbf{R}}_m = \mathbf{X}_m^H\mathbf{X}_m = \sum_{n=0}^{N-1} \mathbf{x}_m(n)\mathbf{x}_m^H(n)$
Cross-correlation vector	$\mathbf{d}_m(n) \triangleq E\{\mathbf{x}_m(n)y^*(n)\}$	$\hat{\mathbf{d}}_m = \mathbf{X}_m^H\mathbf{y} = \sum_{n=0}^{N-1} \mathbf{x}_m(n)y^*(n)$
Power	$P_y(n) = E\{ y(n) ^2\}$	$E_y = \mathbf{y}^H\mathbf{y} = \sum_{n=0}^{N-1} y(n) ^2$

equations can be obtained from the MMSE equations by replacing the linear operator $E\{\cdot\}$ by the linear operator $\sum_n(\cdot)$. As a result, all algorithms developed in Sections 7.1 and 7.2 can be used for linear LSE estimation problems.

For example, we can easily see that $\hat{\mathbf{R}}_M$, $\hat{\mathbf{d}}_M$, \mathbf{L}_M , \mathbf{D}_M , and \mathbf{k}_M have the optimum nesting property described in Section 7.1.1, that is, $\hat{\mathbf{R}}_m = \hat{\mathbf{R}}_M^{[m]}$ and so on. As a result, the factors of the LDL^H decomposition have the optimum nesting property, and we can obtain an order-recursive structure for the computation of the LSE estimate $\hat{y}_m(n)$. Indeed, if we define

$$\mathbf{w}_m(n) = \mathbf{L}_m^{-1}\mathbf{x}_m(n) \quad 0 \leq n \leq N-1 \quad (8.5.5)$$

$$\text{then } \hat{\mathbf{R}}_m = \sum_{n=0}^{N-1} \mathbf{x}_m(n)\mathbf{x}_m^H(n) = \mathbf{L}_m \left[\sum_{n=0}^{N-1} \mathbf{w}_m(n)\mathbf{w}_m^H(n) \right] \mathbf{L}_m \triangleq \mathbf{L}_m \mathbf{D}_m \mathbf{L}_m^H \quad (8.5.6)$$

where the matrix \mathbf{D}_m is diagonal because the LDL^H decomposition is unique. If we define the record vectors

$$\tilde{\mathbf{w}}_j \triangleq [w_j(0) \ w_j(1) \ \cdots \ w_j(N-1)]^H \quad (8.5.7)$$

and the data matrix

$$\mathbf{W}_m \triangleq [\tilde{\mathbf{w}}_1 \ \tilde{\mathbf{w}}_2 \ \cdots \ \tilde{\mathbf{w}}_m] \quad (8.5.8)$$

$$\text{then } \mathbf{D}_m = \mathbf{W}_m^H \mathbf{W}_m = \text{diag}\{\xi_1, \xi_2, \dots, \xi_m\} \quad (8.5.9)$$

$$\text{where } \xi_i = \sum_{n=0}^{N-1} |w_i(n)|^2 = \tilde{\mathbf{w}}_i^H \tilde{\mathbf{w}}_i \quad (8.5.10)$$

From (8.5.9), we have

$$\tilde{\mathbf{w}}_i^H \tilde{\mathbf{w}}_j = 0 \quad \text{for } i \neq j \quad (8.5.11)$$

that is, the columns of \mathbf{W}_m are orthogonal and, in this sense, are the innovation vectors of the columns of data matrix \mathbf{X}_m , according to the LS interpretation of orthogonality introduced in Section 8.2.

Following the approach in Section 7.1.5, we can show that the following order-recursive algorithm

$$w_m(n) = x_m(n) - \sum_{i=1}^{m-1} l_{i-1}^{(m-1)*} w_i(n) \quad (8.5.12)$$

$$\hat{y}_m(n) = \hat{y}_{m-1}(n) + k_m^* w_m(n)$$

$$\text{or} \quad e_m(n) = e_{m-1}(n) - k_m^* w_m(n)$$

computed for $n = 0, 1, \dots, N - 1$ and $m = 1, 2, \dots, M$, provides the LSE estimates for orders $1 \leq m \leq M$.

The statistical interpretations of innovation and partial correlation for $w_m(n)$ and k_{m+1} hold now in a deterministic LSE sense. For example, the *partial correlation* between $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{x}}_{m+1}$ is defined by using the residual records $\tilde{\mathbf{e}}_m = \tilde{\mathbf{y}} - \mathbf{X}_m \mathbf{c}_m$ and $\tilde{\mathbf{e}}_m^b = \tilde{\mathbf{x}}_{m+1} + \mathbf{X}_m \mathbf{b}_m$, where \mathbf{b}_m is the least-squares error BLP. Indeed, if $\beta_{m+1} \triangleq \tilde{\mathbf{e}}_m^H \tilde{\mathbf{e}}_m^b$, we can show that $k_{m+1} = \beta_{m+1}/\xi_{m+1}$ (see Problem 8.11).

EXAMPLE 8.5.1. Solve the LS problem with the following data matrix and desired response signal:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \\ 3 & 1 & 3 \\ 1 & 0 & 1 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 3 \end{bmatrix}$$

Solution. We start by computing the time-average correlation matrix and cross-correlation vector

$$\hat{\mathbf{R}} = \begin{bmatrix} 15 & 8 & 13 \\ 8 & 6 & 6 \\ 13 & 6 & 12 \end{bmatrix} \quad \hat{\mathbf{d}} = \begin{bmatrix} 20 \\ 9 \\ 18 \end{bmatrix}$$

followed by the LDL^H decomposition of $\hat{\mathbf{R}}$ using the MATLAB function `[L, D] = ldlt(X)`. This gives

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 0.5333 & 1 & 0 \\ 0.8667 & -0.5385 & 1 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 15 & 0 & 0 \\ 0 & 1.7333 & 0 \\ 0 & 0 & 0.2308 \end{bmatrix}$$

and working through the steps in Table 8.1, we find the LS solution and LSE to be

$$\mathbf{c}_{ls} = [3.0 \ -1.5 \ -1.0]^T \quad E_{ls} = 1.5$$

using the following sequence of MATLAB commands

```
k=L\ dhat;
cls=L'\ k;
Els=sum((y-X'*cls).^2);
```

These results can be verified by using the command `cls=Rhat\ dhat`.

8.5.2 LSE FIR Filtering and Prediction

As we stressed in Section 7.3, the fundamental difference between general linear estimation and FIR filtering and prediction, which is the key to the development of efficient order-recursive algorithms, is the shift invariance of the input data vector

$$\mathbf{x}_{m+1}(n) = [x(n) \ x(n-1) \ \cdots \ x(n-m+1) \ x(n-m)]^T \quad (8.5.13)$$

The input data vector can be partitioned as

$$\mathbf{x}_{m+1}(n) = \begin{bmatrix} \mathbf{x}_m(n) \\ x(n-m) \end{bmatrix} = \begin{bmatrix} x(n) \\ \mathbf{x}_m(n-1) \end{bmatrix} \quad (8.5.14)$$

which shows that samples from different times are incorporated as the order is increased. This creates a coupling between order and time updatings that has significant implications in the development of efficient algorithms. Indeed, we can easily see that the matrix

$$\hat{\mathbf{R}}_{m+1} = \sum_{n=N_i}^{N_f} \mathbf{x}_{m+1}(n) \mathbf{x}_{m+1}^H(n) \quad (8.5.15)$$

can be partitioned as

$$\hat{\mathbf{R}}_{m+1} = \begin{bmatrix} \hat{\mathbf{R}}_m & \hat{\mathbf{r}}_m^b \\ \hat{\mathbf{r}}_m^{bH} & E_m^b \end{bmatrix} = \begin{bmatrix} E_m^f & \hat{\mathbf{r}}_m^{fH} \\ \hat{\mathbf{r}}_m^f & \hat{\mathbf{R}}_m^f \end{bmatrix} \quad (8.5.16)$$

where $\hat{\mathbf{R}}_m^f = \hat{\mathbf{R}}_m + \mathbf{x}_m(N_i - 1)\mathbf{x}_m^H(N_i - 1) - \mathbf{x}_m(N_f)\mathbf{x}_m^H(N_f)$ (8.5.17)

is the matrix equivalent of (8.2.28). We notice that the relationship between $\hat{\mathbf{R}}_m^f$ and $\hat{\mathbf{R}}_m$, which allows for the development of a complete set of order-recursive algorithms for FIR filtering and prediction, depends on the choice of N_i and N_f , that is, the windowing method selected.

As we discussed in Section 8.3, there are four cases of interest. In the *full-windowing* case ($N_i = 0, N_f = N + M - 2$), we have $\hat{\mathbf{R}}_m^f = \hat{\mathbf{R}}_m$ and $\hat{\mathbf{R}}_m$ is Toeplitz. Therefore, all the algorithms and structures developed in Chapter 7 for Toeplitz matrices can be utilized.

In the *prewindowing* case ($N_i = 0, N_f = N - 1$), Equation (8.5.17) becomes

$$\hat{\mathbf{R}}_m^f = \hat{\mathbf{R}}_m - \mathbf{x}_m(N - 1)\mathbf{x}_m^H(N - 1) \quad (8.5.18)$$

Since $\mathbf{x}_m(n) = \mathbf{0}$ for $n \leq 0$ (prewindowing), $\hat{\mathbf{R}}_m$ is a function of N . If we use the definition

$$\hat{\mathbf{R}}_m(N) \triangleq \sum_{n=0}^{N-1} \mathbf{x}_m(n)\mathbf{x}_m^H(n) \quad (8.5.19)$$

then the time-updating (8.5.18) can be written as

$$\hat{\mathbf{R}}_m^f = \hat{\mathbf{R}}_m(N - 1) = \hat{\mathbf{R}}_m(N) - \mathbf{x}_m(N - 1)\mathbf{x}_m^H(N - 1) \quad (8.5.20)$$

and the order-updating (8.5.16) as

$$\hat{\mathbf{R}}_{m+1}(N) = \begin{bmatrix} \hat{\mathbf{R}}_m(N) & \hat{\mathbf{r}}_m^b(N) \\ \hat{\mathbf{r}}_m^{bH}(N) & E_m^b(N) \end{bmatrix} = \begin{bmatrix} E_m^f(N) & \hat{\mathbf{r}}_m^{fH}(N) \\ \hat{\mathbf{r}}_m^f(N) & \hat{\mathbf{R}}_m(N - 1) \end{bmatrix} \quad (8.5.21)$$

which has the same form as (7.3.3). Therefore, all order recursions developed in Section 7.3 can be applied in the prewindowing case. However, to get a complete algorithm, we need recursions for the time updatings of the BLP $\mathbf{b}_m(N - 1) \rightarrow \mathbf{b}_m(N)$ and $E_m^b(N - 1) \rightarrow E_m^b(N)$, which can be developed by using the time-recursive algorithms developed in Chapter 10 for LS adaptive filters. The *postwindowing* case can be developed in a similar fashion, but it is of no particular practical interest.

In the *no-windowing* case ($N_i = M - 1, N_f = N - 1$), matrices $\hat{\mathbf{R}}_m$ and $\hat{\mathbf{R}}_m^f$ depend on both M and N . Thus, although the development of order recursions can be done as in the prewindowing case, the time updatings are more complicated due to (8.5.17) (Morf et al. 1977). Setting the lower limit to $N_i = M - 1$ means that all filters $\mathbf{c}_m, 1 \leq m \leq M$, are optimized over the interval $M - 1 \leq n \leq N - 1$, which makes the optimum nesting property possible. If we set $N_i = m - 1$, each filter \mathbf{c}_m is optimized over the interval $m - 1 \leq n \leq N - 1$; that is, it utilizes all the available data. However, in this case, the optimum nesting property $\hat{\mathbf{R}}_m = \hat{\mathbf{R}}_M^{[m]}$ does not hold, and the resulting order-recursive algorithms are slightly more complicated (Kalouptsidis et al. 1984).

The development of order-recursive algorithms for FBLP least-squares filters and predictors with linear phase constraints, for example, $\mathbf{c}_m = \pm \mathbf{J}\mathbf{c}_m^*$, is more complicated, in general. A review of existing algorithms and more references can be found in Theodoridis and Kalouptsidis (1993).

In conclusion, we notice that order-recursive algorithms are more efficient than the LDL^H decomposition-based solutions only if N is much larger than M . Furthermore, their numerical properties are inferior to those of the LDL^H decomposition methods; therefore, a bit of extra caution needs to be exercised when order-recursive algorithms are employed.

8.6 LS COMPUTATIONS USING ORTHOGONALIZATION TECHNIQUES

When we use the LDL^H or Cholesky decomposition for the computation of LSE filters, we first must compute the time-average correlation matrix $\hat{\mathbf{R}} = \mathbf{X}^H \mathbf{X}$ and the time-average cross-correlation vector $\hat{\mathbf{d}} = \mathbf{X}^H \mathbf{y}$ from the data \mathbf{X} and \mathbf{y} . Although this approach is widely used in practice, there are certain applications that require methods with better numerical properties. When numerical considerations are a major concern, the orthogonalization techniques, discussed in this section, and the singular value decomposition, discussed in Section 8.7, are the methods of choice for the solution of LS problems.

Orthogonal transformations are linear changes of variables that preserve length. In matrix notation

$$\mathbf{y} = \mathbf{Q}^H \mathbf{x} \quad (8.6.1)$$

where \mathbf{Q} is an orthogonal matrix, that is,

$$\mathbf{Q}^{-1} = \mathbf{Q}^H \Rightarrow \mathbf{Q}\mathbf{Q}^H = \mathbf{I} \quad (8.6.2)$$

From this property, we can easily see that

$$\|\mathbf{y}\|^2 = \mathbf{y}^H \mathbf{y} = \mathbf{x}^H \mathbf{Q} \mathbf{Q}^H \mathbf{x} = \mathbf{x}^H \mathbf{x} = \|\mathbf{x}\|^2 \quad (8.6.3)$$

that is, multiplying a vector by an orthogonal matrix does not change the length of the vector.[†] As a result, algorithms that use orthogonal transformations do not amplify roundoff errors, resulting in more accurate numerical algorithms. There are two ways to look at the solution of LS problems using orthogonalization techniques:

- Use orthogonal matrices to transform the data matrix \mathbf{X} to a form that simplifies the solution of the normal equations without affecting the time-average correlation matrix $\hat{\mathbf{R}} = \mathbf{X}^H \mathbf{X}$. For any orthogonal matrix \mathbf{Q} , we have

$$\hat{\mathbf{R}} = \mathbf{X}^H \mathbf{X} = \mathbf{X}^H \mathbf{Q} \mathbf{Q}^H \mathbf{X} = (\mathbf{Q}^H \mathbf{X})^H \mathbf{Q}^H \mathbf{X} \quad (8.6.4)$$

Clearly, we can repeat this process as many times as we wish until the matrix $\mathbf{X}^H \mathbf{Q}_1 \mathbf{Q}_2 \dots$ is in a form that simplifies the solution of the LS problem.

- Since orthogonal transformations preserve the length of a vector, multiplying the residual $\mathbf{e} = \mathbf{y} - \mathbf{Xc}$ by an orthogonal matrix does not change the total squared error. Hence, multiplying the residuals by \mathbf{Q}^H gives

$$\min_{\mathbf{c}} \|\mathbf{e}\| = \min_{\mathbf{c}} \|\mathbf{y} - \mathbf{Xc}\| = \min_{\mathbf{c}} \|\mathbf{Q}^H(\mathbf{y} - \mathbf{Xc})\| \quad (8.6.5)$$

Thus, the goal is to find a matrix \mathbf{Q} that simplifies the solution of the LS problem.

Suppose that we have already found an $N \times N$ orthogonal matrix \mathbf{Q} such that

$$\mathbf{X} = \mathbf{Q} \begin{bmatrix} \mathcal{R} \\ \mathbf{0} \end{bmatrix} \quad (8.6.6)$$

where, in practice, \mathbf{Q} is constructed to make the $M \times M$ matrix \mathcal{R} upper triangular.[‡] Using (8.6.5), we have

$$\|\mathbf{e}\| = \|\mathbf{Q}^H \mathbf{e}\| = \|\mathbf{Q}^H \mathbf{y} - \mathbf{Q}^H \mathbf{Xc}\| \quad (8.6.7)$$

Using the partitioning

$$\mathbf{Q} \triangleq [\mathbf{Q}_1 \ \mathbf{Q}_2] \quad (8.6.8)$$

where \mathbf{Q}_1 has M columns, we obtain

$$\mathbf{X} = \mathbf{Q}_1 \mathcal{R} \quad (8.6.9)$$

[†]Matrix \mathbf{Q} is an arbitrary unitary matrix and should not be confused with the eigenvector matrix of \mathbf{R} .

[‡]The symbol \mathcal{U} would be more appropriate for the upper triangular matrix \mathcal{R} which can also be mistaken for the correlation matrix \mathbf{R} . However, we chose \mathcal{R} because, otherwise, it would be difficult to use the well-established term *QR factorization*.

which is known as the “*thin*” QR decomposition. Similarly,

$$\mathbf{z} \triangleq \mathbf{Q}^H \mathbf{y} = \begin{bmatrix} \mathbf{Q}_1^H \mathbf{y} \\ \mathbf{Q}_2^H \mathbf{y} \end{bmatrix} \triangleq \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} \quad (8.6.10)$$

where \mathbf{z}_1 has M components and \mathbf{z}_2 has $N - M$ components. Substitution of (8.6.9) and (8.6.10) into (8.6.7) gives

$$\|\mathbf{e}\| = \left\| \begin{bmatrix} \mathcal{R}\mathbf{c} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{Q}_1^H \mathbf{y} \\ \mathbf{Q}_2^H \mathbf{y} \end{bmatrix} \right\| = \left\| \begin{bmatrix} \mathcal{R}\mathbf{c} - \mathbf{z}_1 \\ -\mathbf{z}_2 \end{bmatrix} \right\| \quad (8.6.11)$$

Since the term $\mathbf{z}_2 = \mathbf{Q}_2^H \mathbf{y}$ does not depend on the parameter vector \mathbf{c} , the length of $\|\mathbf{e}\|$ becomes minimum if we set $\mathbf{c} = \mathbf{c}_{ls}$, that is,

$$\mathcal{R}\mathbf{c}_{ls} = \mathbf{z}_1 \quad (8.6.12)$$

and

$$E_{ls} = \|\mathbf{Q}_2^H \mathbf{y}\|^2 = \|\mathbf{z}_2\| \quad (8.6.13)$$

where the upper triangular system in (8.6.12) can be solved for \mathbf{c}_{ls} by back substitution.

The steps for the solution of the LS problem using the QR decomposition are summarized in Table 8.3.

TABLE 8.3
Solution of the LS problem using the QR decomposition method.

Step	Computations	Description
1	$\mathbf{X} = \mathbf{Q} \begin{bmatrix} \mathcal{R} \\ \mathbf{0} \end{bmatrix}$	QR decomposition
2	$\mathbf{z} = \mathbf{Q}^H \mathbf{y} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix}$	Transformation and partitioning of \mathbf{y}
3	$\mathcal{R}\mathbf{c}_{ls} = \mathbf{z}_1$	Backward substitution $\rightarrow \mathbf{c}_{ls}$
4	$E_{ls} = \ \mathbf{z}_2\ ^2$	Computation of LS error
5	$\mathbf{e}_{ls} = \mathbf{Q} \begin{bmatrix} \mathbf{0} \\ \mathbf{z}_2 \end{bmatrix}$	Back transformation of residuals

Using the QR decomposition (8.6.6), we have

$$\hat{\mathbf{R}} = \mathbf{X}^H \mathbf{X} = \mathcal{R}^H \mathcal{R} \quad (8.6.14)$$

which, in conjunction with the unique Cholesky decomposition $\hat{\mathbf{R}} = \mathcal{L} \mathcal{L}^H$, gives

$$\mathcal{R} = \mathcal{L}^H \quad (8.6.15)$$

that is, the QR factorization computes the Cholesky factor \mathcal{R} directly from data matrix \mathbf{X} . Also, since $\mathcal{L}^H \mathbf{c}_{ls} = \tilde{\mathbf{k}}$, we have

$$\tilde{\mathbf{k}} = \mathbf{z}_1 \quad (8.6.16)$$

which, owing to the Cholesky decomposition, leads to

$$E_{ls} = E_y - \tilde{\mathbf{k}}^H \tilde{\mathbf{k}} = \|\mathbf{z}_2\|^2 \quad (8.6.17)$$

because $\|\mathbf{y}\|^2 = \|\mathbf{Q}^H \mathbf{y}\|^2 = \|\mathbf{z}_1\|^2 + \|\mathbf{z}_2\|^2$.

If we form the augmented matrix

$$\bar{\mathbf{X}} = [\mathbf{X} \ \mathbf{y}] \quad (8.6.18)$$

the QR decomposition of $\bar{\mathbf{X}}$ provides the triangular factor

$$\begin{bmatrix} \mathcal{R} & \tilde{\mathbf{k}} \\ \mathbf{0} & \tilde{\xi} \end{bmatrix} \quad (8.6.19)$$

which is identical to the one obtained from the Cholesky decomposition of $\bar{\mathbf{R}} = \bar{\mathbf{X}}^H \bar{\mathbf{X}}$ with $\mathcal{R} = \mathcal{L}^H$ and $\tilde{\xi}^2 = E_{ls}$ (see Problem 8.14).

EXAMPLE 8.6.1. Solve the LS problem in Example 8.5.1

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \\ 3 & 1 & 3 \\ 1 & 0 & 1 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 3 \end{bmatrix}$$

using the QR decomposition approach.

Solution. Using the MATLAB function $[Q, R] = qr(\mathbf{X})$, we obtain

$$\mathbf{Q} = \begin{bmatrix} -0.2582 & -0.3545 & 0.8006 & 0.4082 \\ -0.5164 & -0.7089 & -0.4804 & 0.0000 \\ -0.7746 & 0.4557 & 0.1601 & -0.4082 \\ -0.2582 & 0.4051 & -0.3203 & 0.8165 \end{bmatrix}$$

$$\mathcal{R} = \begin{bmatrix} -3.8730 & -2.0656 & -3.5666 \\ 0 & -1.3166 & 0.7089 \\ 0 & 0 & 0.4804 \\ 0 & 0 & 0 \end{bmatrix}$$

and following the steps in Table 8.3, we find the LS solution and the LSE to be

$$\mathbf{c}_{ls} = [3.0 \ -1.5 \ -1.0]^T \quad E_{ls} = 1.5$$

using the sequence of MATLAB commands

```
z=Q'*y;
cls=R(1:3,1:3)'\z(1:3);
Els=sum(z(4).^2);
```

In applications that require only the error (or residual) vector \mathbf{e}_{ls} , we do not need to solve the triangular system $\mathcal{R}\mathbf{c}_{ls} = \mathbf{z}_1$. Instead, we can compute directly the error by $\mathbf{e}_{ls} = \mathbf{Q}[0 \ z_2]$ or the MATLAB command $e=Q*[zeros(1,M) z2']'$. This approach is known as *direct error* (or *residual*) *extraction* and plays an important role in LS adaptive filtering algorithms and architectures (see Chapter 10).

It is generally agreed in numerical analysis that orthogonal decomposition methods applied directly to data matrix \mathbf{X} are *preferable* to the computation and solution of the normal equations whenever numerical stability is important (Hager 1988; Golub and Van Loan 1996). The sensitivity of the solution \mathbf{c}_{ls} to perturbations in the data \mathbf{X} and \mathbf{y} depends on the ratio of the largest to the smallest eigenvalues of $\hat{\mathbf{R}}$ and does not depend on the algorithm used to compute the solution. Furthermore, the numerical accuracy required to compute \mathcal{L} directly from \mathbf{X} is one-half of that required to compute \mathcal{L} from $\hat{\mathbf{R}}$. The “squaring” $\hat{\mathbf{R}} = \mathbf{X}^H \mathbf{X}$ of the data to form the time-average correlation matrix results in a loss of information and should be avoided if the numerical precision is not deemed sufficient. Algorithms that compute \mathcal{L} directly from \mathbf{X} are known as *square root methods*. However, by paraphrasing Rader (1996), we use the terms *amplitude-domain techniques* for methods that compute \mathcal{L} directly from \mathbf{X} and *power-domain techniques* for methods that compute \mathcal{L} indirectly from $\hat{\mathbf{R}} = \mathbf{X}^H \mathbf{X}$. These ideas are illustrated in the following example.

EXAMPLE 8.6.2. Let

$$\mathbf{X} = \begin{bmatrix} 1 & 1 \\ \epsilon & 0 \\ 0 & \epsilon \end{bmatrix} \quad \hat{\mathbf{R}} = \mathbf{X}^T \mathbf{X} = \begin{bmatrix} 1 + \epsilon^2 & 1 \\ 1 & 1 + \epsilon^2 \end{bmatrix}$$

where $\mathbf{X}^T \mathbf{X}$ is clearly positive definite and nonsingular. Let the desired signal be $\mathbf{y} = [2 \ \epsilon \ \epsilon]^T$ so that $\hat{\mathbf{d}} = [2 + \epsilon^2 \ 2 + \epsilon^2]^T$. If ϵ is such that $1 + \epsilon^2 = 1$, due to limited numerical precision,

the matrix $\mathbf{X}^T \mathbf{X}$ becomes singular. If we set $\epsilon = 10^{-8}$, solving the LS equations for \mathbf{c}_{ls} using the MATLAB command $\mathbf{cls} = \mathbf{Rhat} \backslash \mathbf{dhat}$ is not possible since $\hat{\mathbf{R}}$ is singular to the working precision of MATLAB. However, if the problem is solved using the QR decomposition as shown in Example 8.6.1, we find $\mathbf{c}_{ls} = [1 \ 1]^T$. Note that even for slightly larger values of ϵ the MATLAB command $\mathbf{cls} = \mathbf{Rhat} \backslash \mathbf{dhat}$ is able to find a solution that differs from the true LS solution since $\hat{\mathbf{R}}$ is ill conditioned.

There are two classes of orthogonal decomposition algorithms:

1. Methods that compute the orthogonal matrix \mathbf{Q} : Householder reflections and Givens rotations
2. Methods that compute \mathbf{Q}_1 : classical and modified Gram-Schmidt orthogonalizations

These decompositions are illustrated in Figure 8.10. The cost of the QR decomposition using the Givens rotations is twice the cost of using Householder reflections or the Gram-Schmidt orthogonalization. The standard method for the computation of the QR decomposition and the solution of LS problems employs the Householder transformation. The Givens rotations are preferred for the implementation of adaptive LS filters (see Chapter 10).

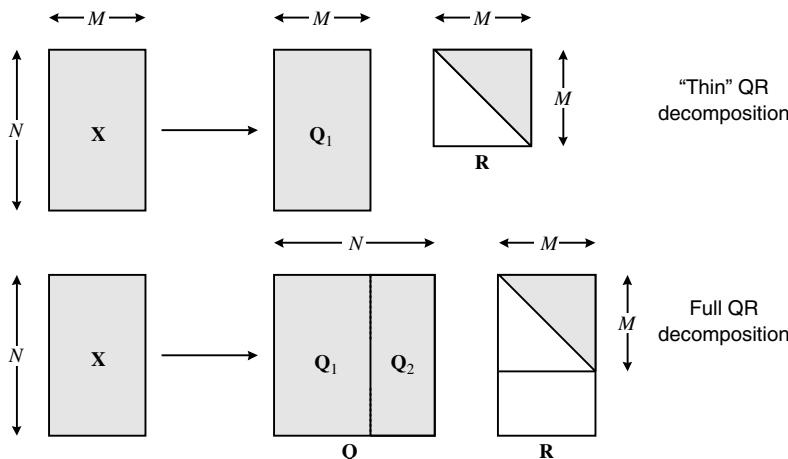


FIGURE 8.10

Pictorial illustration of the differences between thin and full QR decompositions.

8.6.1 Householder Reflections

Consider a vector \mathbf{x} and a fixed line l in the plane (see Figure 8.11). If we reflect \mathbf{x} about the line l , we obtain a vector \mathbf{y} that is the mirror image of \mathbf{x} . Clearly, the vector \mathbf{x} and its *reflection* \mathbf{y} have the same length. We define a unit vector \mathbf{w} in the direction of $\mathbf{x} - \mathbf{y}$ as

$$\mathbf{w} \triangleq \frac{1}{\|\mathbf{x} - \mathbf{y}\|}(\mathbf{x} - \mathbf{y}) \quad (8.6.20)$$

assuming that \mathbf{x} and \mathbf{y} are nonzero vectors.

Since the projection of \mathbf{x} on \mathbf{w} is $(\mathbf{w}^H \mathbf{x})\mathbf{w}$, simple inspection of Figure 8.11 gives

$$\mathbf{y} = \mathbf{x} - 2(\mathbf{w}^H \mathbf{x})\mathbf{w} = \mathbf{x} - 2(\mathbf{w}\mathbf{w}^H)\mathbf{x} = (\mathbf{I} - 2\mathbf{w}\mathbf{w}^H)\mathbf{x} \triangleq \mathbf{Hx}$$

where

$$\mathbf{H} \triangleq \mathbf{I} - 2\mathbf{w}\mathbf{w}^H \quad (8.6.21)$$

In general, any matrix \mathbf{H} of the form (8.6.21) with $\|\mathbf{w}\| = 1$ is known as a *Householder reflection* or *Householder transformation* (Householder 1958) and has the following properties

$$\mathbf{H}^H = \mathbf{H} \quad \mathbf{H}^H \mathbf{H} = \mathbf{I} \quad \mathbf{H}^{-1} = \mathbf{H}^H \quad (8.6.22)$$

that is, the matrix \mathbf{H} is unitary.

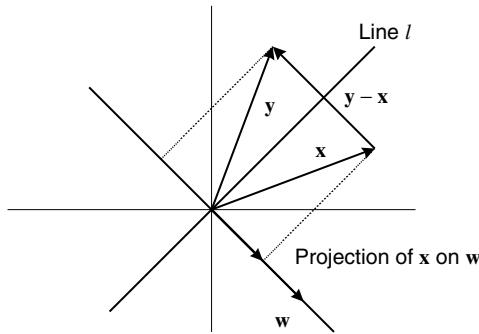


FIGURE 8.11
The Householder reflection vector.

We can build a Householder matrix \mathbf{H}_k that leaves intact the first $k - 1$ components of a given vector \mathbf{x} , changes the k th component, and annihilates (zeros out) the remaining components, that is,

$$y_i = (\mathbf{H}\mathbf{x})_i = \begin{cases} x_i & i = 1, 2, \dots, k - 1 \\ y_k & i = k \\ 0 & i = k + 1, \dots, N \end{cases} \quad (8.6.23)$$

where y_k is to be determined. If we set

$$y_k = \pm \left(\sum_{i=k}^N |x_i|^2 \right)^{1/2} e^{j\theta_k} \quad (8.6.24)$$

where θ_k is the angle part of x_k (if complex-valued), then both \mathbf{x} and \mathbf{y} have the same length. There are two choices for the sign of y_k . Since the computation of \mathbf{w} by (8.6.20) involves subtraction (which can lead to severe numerical problems when two numbers are nearly equal), we choose the negative sign so that y_k and x_k have opposite signs. Hence, $y_k - x_k$ is never the difference between nearly equal numbers. Therefore, using (8.6.20), we find that \mathbf{w} is given by

$$\mathbf{w} = \frac{1}{\sqrt{2s_k(s_k + |x_k|)}} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ (|x_k| + s_k)e^{j\theta_k} \\ x_{k+1} \\ \vdots \\ x_N \end{bmatrix} \quad (8.6.25)$$

$$\text{where } s_k \triangleq \left(\sum_{i=k}^N |x_i|^2 \right)^{1/2} \quad (8.6.26)$$

In general, an $N \times M$ matrix \mathbf{X} with $N > M$ can be diagonalized with a sequence of M Householder transformations

$$\mathbf{H}_M \cdots \mathbf{H}_2 \mathbf{H}_1 \mathbf{X} = \mathcal{R} \quad (8.6.27)$$

$$\text{or } \mathbf{X} = \mathbf{Q} \mathcal{R} \quad (8.6.28)$$

$$\text{where } \mathbf{Q} \triangleq \mathbf{H}_1 \mathbf{H}_2 \cdots \mathbf{H}_M \quad (8.6.29)$$

Note that for $M = N$ we need only $M - 1$ reflections.

We next illustrate by an example how to compute the QR decomposition of a rectangular matrix by using a sequence of Householder transformations.

EXAMPLE 8.6.3. Find the QR decomposition of the data matrix

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 6 & 7 \end{bmatrix}$$

using Householder reflections.

Solution. Using (8.6.25), we compute the vector $\mathbf{w}_1 = [0.7603 \ 0.2054 \ 0.6162]^T$ and the Householder reflection matrix \mathbf{H}_1 for the first column of \mathbf{X} . The modified data matrix is

$$\mathbf{H}_1 \mathbf{X} = \begin{bmatrix} -6.4031 & -7.8087 \\ 0 & 0.3501 \\ 0 & -0.9496 \end{bmatrix}$$

Similarly, we compute the vector $\mathbf{w}_2 = [0 \ 0.8203 \ -0.5719]^T$ and matrix \mathbf{H}_2 for the second column of $\mathbf{H}_1 \mathbf{X}$, which results in the desired QR decomposition

$$\mathbf{H}_2 \mathbf{H}_1 \mathbf{X} = \mathcal{R} = \begin{bmatrix} -6.4031 & -7.8087 \\ 0 & -1.0121 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{Q} = \mathbf{H}_1 \mathbf{H}_2 = \begin{bmatrix} -0.1562 & -0.7711 & -0.6172 \\ -0.3123 & -0.5543 & -0.7715 \\ -0.9370 & 0.3133 & -0.1543 \end{bmatrix}$$

This result can be verified by using the MATLAB function `[Q, R] = qr(X)`, which implements the Householder transformation.

8.6.2 The Givens Rotations

The second elementary transformation that does not change the length of a vector is a rotation about an axis (see Figure 8.12). To describe the method of Givens, we assume for simplicity that the vectors are real-valued. The components of the rotated vector \mathbf{y} in terms of the components of the original vector \mathbf{x} are

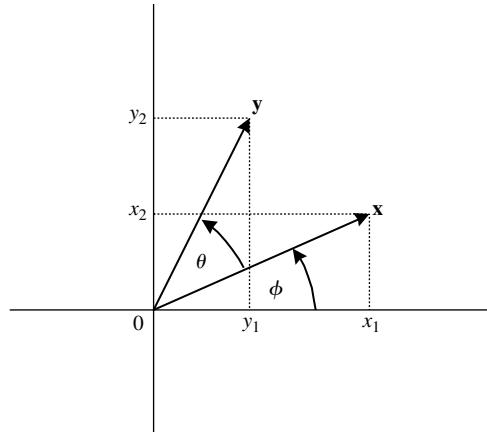
$$y_1 = r \cos(\phi + \theta) = x_1 \cos \theta - x_2 \sin \theta$$

$$y_2 = r \sin(\phi + \theta) = x_1 \sin \theta + x_2 \cos \theta$$

or in matrix form

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \triangleq \mathbf{G}(\theta) \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (8.6.30)$$

FIGURE 8.12
The Givens rotation.



where θ is the angle of rotation. We can easily show that the rotation matrix $\mathbf{G}(\theta)$ in (8.6.30) is orthogonal and has a determinant $\det \mathbf{G}(\theta) = 1$.

Any matrix of the form

$$\mathbf{G}_{ij}(\theta) \triangleq \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{array}{l} \leftarrow i \\ \uparrow \\ i \end{array} \quad \begin{array}{l} \leftarrow j \\ \uparrow \\ j \end{array} \quad (8.6.31)$$

with

$$c^2 + s^2 = 1 \quad (8.6.32)$$

is known as a *Givens rotation*. When this matrix is applied to a vector \mathbf{x} , it rotates the components x_i and x_j through an angle $\theta = \arctan(s/c)$ while leaving all other components intact (Givens 1958). Because of (8.6.30), we can write $c = \cos \theta$ and $s = \sin \theta$ for some angle θ . It can easily be shown that the matrix $\mathbf{G}_{ij}(\theta)$ is orthogonal.

The Givens rotations have two attractive features. First, performing the rotation $\mathbf{y} = \mathbf{G}_{ij}(\theta)\mathbf{x}$ as

$$\begin{aligned} y_i &= cx_i - sx_j \\ y_j &= sx_i + cx_j \\ y_k &= x_k \quad k \neq i, j \end{aligned} \quad (8.6.33)$$

requires only four multiplications and two additions. Second, we can choose c and s to annihilate the j th component of a vector. Indeed, if we set

$$c = \frac{x_i}{\sqrt{x_i^2 + x_j^2}} \quad s = -\frac{x_j}{\sqrt{x_i^2 + x_j^2}} \quad (8.6.34)$$

in (8.6.31), then

$$y_i = \sqrt{x_i^2 + x_j^2} \quad \text{and} \quad y_j = 0 \quad (8.6.35)$$

Using a sequence of Givens rotations, we can annihilate (zero out) all elements of a matrix \mathbf{X} below the main diagonal to obtain the upper triangular matrix of the QR decomposition. The product of all the Givens rotation matrices provides matrix \mathbf{Q} . We stress that the order of rotations cannot be arbitrary because later rotations can destroy zeros introduced earlier. A version of the Givens algorithm without square roots, which is known as the *fast Givens QR*, is discussed in Golub and Van Loan (1996).

We illustrate this procedure with the next example.

EXAMPLE 8.6.4. The QR decomposition can be found in order to find the LS solution using the Givens rotations. Given the same data matrix \mathbf{X} as in Example 8.6.3

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 6 & 7 \end{bmatrix}$$

we first zero the last element of the first column, that is, element (3, 1), using the Givens rotation matrix \mathbf{G}_{31} with $c = -0.1664$ and $s = 0.9864$. Indeed, using (8.6.34), we have

$$\mathbf{G}_{31}\mathbf{X} = \begin{bmatrix} -6.0828 & -7.2336 \\ 2 & 3 \\ 0 & 0.8220 \end{bmatrix}$$

Then the element (2, 1) is eliminated by using the Givens rotation matrix \mathbf{G}_{21} with $c = 0.9550$ and $s = 0.3123$, resulting in

$$\mathbf{G}_{21}\mathbf{G}_{31}\mathbf{X} = \begin{bmatrix} -6.4031 & -7.8087 \\ 0 & 0.5905 \\ 0 & 0.8220 \end{bmatrix}$$

Finally, the QR factorization is found after applying the Givens rotation matrix \mathbf{G}_{32} with $c = -0.5834$ and $s = 0.8122$:

$$\mathcal{R} = \mathbf{G}_{32}\mathbf{G}_{21}\mathbf{G}_{31}\mathbf{X} = \begin{bmatrix} -6.4031 & -7.8087 \\ 0 & -1.0121 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{Q} = \mathbf{G}_{31}^T \mathbf{G}_{21}^T \mathbf{G}_{32}^T = \begin{bmatrix} -0.1562 & -0.7711 & -0.6172 \\ -0.3123 & -0.5543 & -0.7715 \\ -0.9370 & 0.3133 & -0.1543 \end{bmatrix}$$

which, as expected, agrees with the QR decomposition found in Example 8.6.3.

In the case of complex-valued vectors, the components of rotated vector \mathbf{y} in (8.6.30) are given by

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos \theta & -e^{-j\psi} \sin \theta \\ e^{j\psi} \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (8.6.36)$$

where $c \triangleq \cos \theta$ and $s \triangleq e^{j\psi} \sin \theta$. The element $-s$ of the rotation matrix $\mathbf{G}_{ij}(\theta)$ is replaced by $-s^*$, where $c^2 + |s|^2 = 1$ instead of (8.6.32).

8.6.3 Gram-Schmidt Orthogonalization

If we are given a set of M linearly independent vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M$, we can create an orthonormal basis $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_M$ that spans the same space by using a systematic procedure known as the *classical Gram-Schmidt (GS) orthogonalization method* (see also Section 7.2.4). The GS method starts by choosing

$$\mathbf{q}_1 = \frac{\mathbf{x}_1}{\|\mathbf{x}_1\|} \quad (8.6.37)$$

as the first basis vector. To obtain \mathbf{q}_2 , we express \mathbf{x}_2 as the sum of two components: its projection $(\mathbf{q}_1^H \mathbf{x}_2)\mathbf{q}_1$ onto \mathbf{q}_1 and a vector \mathbf{p}_2 that is perpendicular to \mathbf{q}_1 . Hence,

$$\mathbf{p}_2 = \mathbf{x}_2 - (\mathbf{q}_1^H \mathbf{x}_2)\mathbf{q}_1 \quad (8.6.38)$$

and \mathbf{q}_2 is obtained by normalizing \mathbf{p}_2 , that is,

$$\mathbf{q}_2 = \frac{\mathbf{p}_2}{\|\mathbf{p}_2\|} \quad (8.6.39)$$

The vectors \mathbf{q}_1 and \mathbf{q}_2 have unit length, are orthonormal, and span the same space as \mathbf{x}_1 and \mathbf{x}_2 . In general, the orthogonal basis vector \mathbf{q}_j is obtained by removing from \mathbf{x}_j its projections onto the already computed vectors \mathbf{q}_1 to \mathbf{q}_{j-1} . Therefore, we have

$$\mathbf{p}_j = \mathbf{x}_j - \sum_{i=1}^{j-1} (\mathbf{q}_i^H \mathbf{x}_j)\mathbf{q}_i \quad \text{and} \quad \mathbf{q}_j = \frac{\mathbf{p}_j}{\|\mathbf{p}_j\|} \quad (8.6.40)$$

for all $1 \leq j \leq M$.

The GS algorithm can be used to obtain the “thin” $\mathbf{Q}_1 \mathcal{R}$ factorization. Indeed, if we define

$$r_{ij} \triangleq \mathbf{q}_i^H \mathbf{x}_j \quad r_{jj} \triangleq \|\mathbf{p}_j\| \quad (8.6.41)$$

we have

$$\mathbf{p}_j = r_{jj} \mathbf{q}_j = \mathbf{x}_j - \sum_{i=1}^{j-1} r_{ij} \mathbf{q}_i \quad (8.6.42)$$

or by solving for \mathbf{x}_j

$$\mathbf{x}_j = \sum_{i=1}^j \mathbf{q}_i r_{ij} \quad j = 1, 2, \dots, M \quad (8.6.43)$$

Using matrix notation, we can express this relation as $\mathbf{X} = \mathbf{Q}_1 \mathcal{R}$, which is exactly the thin $\mathbf{Q}_1 \mathcal{R}$ factorization in (8.6.9).

Major drawbacks of the GS procedure are that it does not produce accurate results and that the resulting basis may not be orthogonal when implemented using finite-precision arithmetic. However, we can achieve better numerical behavior if we reorganize the computations in a form known as the *modified Gram-Schmidt (MGS) algorithm* (Björck 1967). We start the first step by defining \mathbf{q}_1 as before

$$\mathbf{q}_1 = \frac{\mathbf{x}_1}{\|\mathbf{x}_1\|} \quad (8.6.44)$$

However, all the remaining vectors $\mathbf{x}_2, \dots, \mathbf{x}_M$ are modified to be orthogonal to \mathbf{q}_1 by subtracting from each vector its projection onto \mathbf{q}_1 , that is,

$$\mathbf{x}_i^{(1)} = \mathbf{x}_i - (\mathbf{q}_1^H \mathbf{x}_i) \mathbf{q}_1 \quad i = 2, \dots, M \quad (8.6.45)$$

At the second step, we define the vector

$$\mathbf{q}_2 = \frac{\mathbf{x}_2^{(1)}}{\|\mathbf{x}_2^{(1)}\|} \quad (8.6.46)$$

which is already orthogonal to \mathbf{q}_1 . Then we modify the remaining vectors to make them orthogonal to \mathbf{q}_2

$$\mathbf{x}_i^{(2)} = \mathbf{x}_i^{(1)} - (\mathbf{q}_2^H \mathbf{x}_i^{(1)}) \mathbf{q}_2 \quad i = 3, \dots, M \quad (8.6.47)$$

Continuing in a similar manner, we compute \mathbf{q}_m and the updated vectors $\mathbf{x}_i^{(m)}$ by

$$\mathbf{q}_m = \frac{\mathbf{x}_m^{(m-1)}}{\|\mathbf{x}_m^{(m-1)}\|} \quad (8.6.48)$$

$$\text{and } \mathbf{x}_i^{(m)} = \mathbf{x}_i^{(m-1)} - (\mathbf{q}_m^H \mathbf{x}_i^{(m-1)}) \mathbf{q}_m \quad i = m + 1, \dots, M \quad (8.6.49)$$

The MGS algorithm involves the following steps, outlined in Table 8.4 and is implemented by the function `Q=mgs(X)`. The superior numerical properties of the modified algorithm stem

TABLE 8.4
Orthogonalization of a set of vectors using the modified Gram-Schmidt algorithm.

Modified GS Algorithm
For $m = 1$ to M
$r_{mm} = \ \mathbf{x}_m\ ^2$
$\mathbf{q}_m = \mathbf{x}_m / r_{mm}$
For $i = m + 1$ to M
$r_{mi} = \mathbf{q}_m^H \mathbf{x}_i$
$\mathbf{x}_i \leftarrow \mathbf{x}_i - r_{mi} \mathbf{q}_m$
next i
next m

from the fact that successive $\mathbf{x}_i^{(m)}$ generated by (8.6.49) decrease in size and that the dot product $\mathbf{q}_m^H \mathbf{x}_i^{(m-1)}$ can be computed more accurately than the dot product $\mathbf{q}_m^H \mathbf{x}_i$.

EXAMPLE 8.6.5. Consider an LS problem (Dahlquist and Björck 1974) with

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 \\ \epsilon & 0 & 0 \\ 0 & \epsilon & 0 \\ 0 & 0 & \epsilon \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where $\epsilon^2 \ll 1$, that is, ϵ^2 can be neglected compared to 1. We first compute $\mathbf{X}^T \mathbf{X}$ and $\mathbf{X}^T \mathbf{y}$ to determine the normal equations

$$\begin{bmatrix} 1 + \epsilon^2 & 1 & 1 \\ 1 & 1 + \epsilon^2 & 1 \\ 1 & 1 & 1 + \epsilon^2 \end{bmatrix} \mathbf{c}_{\text{ls}} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

which provide the exact solution $\mathbf{c}_{\text{ls}} = [1 \ 1 \ 1]^T / (3 + \epsilon^2)$. Numerically, the matrix $\mathbf{X}^T \mathbf{X}$ is singular on any computer with accuracy such that $1 + \epsilon^2$ is rounded to 1. Applying the MGS algorithm to the column vectors of the augmented matrix $[\mathbf{X} \ \mathbf{y}]$, and taking into consideration that $1 + \epsilon^2$ is rounded to 1, we obtain

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ \epsilon & -\epsilon & -\frac{\epsilon}{2} \\ 0 & \epsilon & -\frac{\epsilon}{2} \\ 0 & 0 & \epsilon \end{bmatrix} \quad \mathbf{e} = -\frac{\epsilon}{3} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\mathcal{R} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{z} = \begin{bmatrix} 1 \\ \frac{1}{2} \\ \frac{1}{3} \end{bmatrix}$$

which corresponds to the thin QR decomposition. Solving $\mathcal{R} \mathbf{c}_{\text{ls}} = \mathbf{z}$, we obtain $\mathbf{c}_{\text{ls}} = [1 \ 1 \ 1]^T / 3$, which agrees with the exact solution under the assumption that $1 + \epsilon^2$ is rounded to 1.

8.7 LS COMPUTATIONS USING THE SINGULAR VALUE DECOMPOSITION

The *singular value decomposition (SVD)* plays a prominent role in the theoretical analysis and practical solution of LS problems because (1) it provides a unified framework for the solution of overdetermined and underdetermined LS problems with full rank or that are rank-deficient and (2) it is the best numerical method to solve LS problems in practice. In this section, we discuss the existence and fundamental properties of the SVD, show how to use it for solving the LS problem, and apply it to determine the numerical rank of a matrix. More details are given in Golub and Van Loan (1996), Leon (1990), Stewart (1973), Watkins (1991), and Klema and Laub (1980).

8.7.1 Singular Value Decomposition

The eigenvalue decomposition reduces a *Hermitian* matrix to a diagonal matrix by premultiplying and postmultiplying it by a single unitary matrix. The singular value decomposition, introduced in the next theorem, reduces a *general* matrix to a diagonal one by premultiplying and postmultiplying it by two different unitary matrices.

THEOREM 8.2. Any real $N \times M$ matrix \mathbf{X} with rank r (recall that r is defined as the number of linearly independent columns of a matrix) can be written as

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^H \quad (8.7.1)$$

where \mathbf{U} is an $N \times N$ unitary matrix, \mathbf{V} is an $M \times M$ unitary matrix, and Σ is an $N \times M$ matrix with $\langle \Sigma \rangle_{ij} = 0, i \neq j$, and $\langle \Sigma \rangle_{ii} = \sigma_i > 0, i = 1, 2, \dots, r$. The numbers σ_i are known as the *singular values* of \mathbf{X} and are usually arranged in decreasing order as $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$.

Proof. We follow the derivation given in Stewart (1973). Since the matrix $\mathbf{X}^H\mathbf{X}$ is positive semidefinite, it has nonnegative eigenvalues $\sigma_1^2, \sigma_2^2, \dots, \sigma_M^2$ such that $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0 = \sigma_{r+1} = \dots = \sigma_M$ for $0 \leq r \leq M$. Let $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_M$ be the eigenvectors corresponding to the eigenvalues $\sigma_1^2, \sigma_2^2, \dots, \sigma_M^2$. Consider the partitioning $\mathbf{V} = [\mathbf{V}_1 \ \mathbf{V}_2]$, where \mathbf{V}_1 consists of the first r columns of \mathbf{V} . If $\Sigma_r = \text{diag}\{\sigma_1, \sigma_2, \dots, \sigma_r\}$, then we obtain $\mathbf{V}_1^H\mathbf{X}^H\mathbf{X}\mathbf{V}_1 = \Sigma_r^2$ and

$$\Sigma_r^{-1}\mathbf{V}_1^H\mathbf{X}^H\mathbf{X}\mathbf{V}_1\Sigma_r^{-1} = \mathbf{I} \quad (8.7.2)$$

Since $\mathbf{V}_2^H\mathbf{X}^H\mathbf{X}\mathbf{V}_2 = \mathbf{0}$, we have

$$\mathbf{X}\mathbf{V}_2 = \mathbf{0} \quad (8.7.3)$$

If we define

$$\mathbf{U}_1 \triangleq \mathbf{X}\mathbf{V}_1\Sigma_r^{-1} \quad (8.7.4)$$

then (8.7.2) gives $\mathbf{U}_1^H\mathbf{U}_1 = \mathbf{I}$; that is, the columns of \mathbf{U}_1 are unitary. A unitary matrix $\mathbf{U} \triangleq [\mathbf{U}_1 \ \mathbf{U}_2]$ is found by properly choosing the components of \mathbf{U}_2 , that is, $\mathbf{U}_2^H\mathbf{U}_1 = \mathbf{0}$ and $\mathbf{U}_2^H\mathbf{U}_2 = \mathbf{I}$. Then

$$\mathbf{U}^H\mathbf{X}\mathbf{V} = \begin{bmatrix} \mathbf{U}_1^H \\ \mathbf{U}_2^H \end{bmatrix} \mathbf{X} [\mathbf{V}_1 \ \mathbf{V}_2] = \begin{bmatrix} \mathbf{U}_1^H\mathbf{X}\mathbf{V}_1 & \mathbf{U}_1^H(\mathbf{X}\mathbf{V}_2) \\ \mathbf{U}_2^H\mathbf{X}\mathbf{V}_1 & \mathbf{U}_2^H(\mathbf{X}\mathbf{V}_2) \end{bmatrix} = \begin{bmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (8.7.5)$$

because of (8.7.2), (8.7.3), and $\mathbf{U}_2^H\mathbf{X}\mathbf{V}_1 = (\mathbf{U}_2^H\mathbf{U}_1)\Sigma_r = \mathbf{0}$.

The SVD of a matrix, which is illustrated in Figure 8.13, provides a wealth of information about the structure of the matrix. Figure 8.14 provides a geometric interpretation of the SVD of a 2×2 matrix \mathbf{X} (see Problem 8.23 for details).

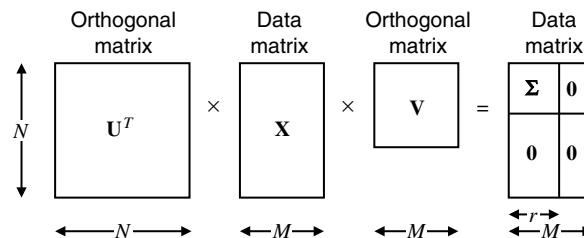


FIGURE 8.13
Pictorial representation of the singular value decomposition of a matrix.

Properties and interpretations. We next provide a summary of interpretations and properties whose proofs are given in the references and the problems.

- Postmultiplying (8.7.1) by \mathbf{V} and equating columns, we obtain

$$\mathbf{X}\mathbf{v}_i = \begin{cases} \sigma_i \mathbf{u}_i & i = 1, 2, \dots, r \\ 0 & i = r + 1, \dots, M \end{cases} \quad (8.7.6)$$

that is, \mathbf{v}_i (columns of \mathbf{V}) are the *right singular vectors* of \mathbf{X} .

- Premultiplying (8.7.1) by \mathbf{U}^H and equating rows, we obtain

$$\mathbf{u}_i^H\mathbf{X} = \begin{cases} \sigma_i \mathbf{v}_i^H & i = 1, 2, \dots, r \\ 0 & i = r + 1, \dots, N \end{cases} \quad (8.7.7)$$

that is, \mathbf{u}_i (columns of \mathbf{U}) are the *left singular vectors* of \mathbf{X} .

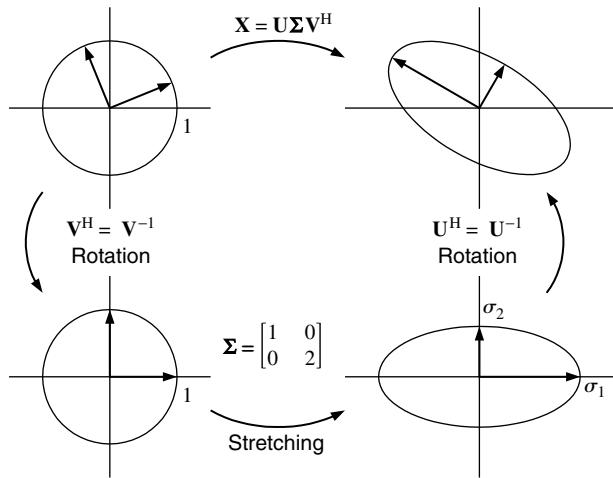


FIGURE 8.14

The SVD of a 2×2 matrix maps the unit circle into an ellipse whose semimajor and semiminor axes are equal to the singular values of the matrix.

3. Let $\lambda_i(\cdot)$ and $\sigma_i^2(\cdot)$ denote the i th largest eigenvalue and singular value of a given matrix, respectively. The vectors $\mathbf{v}_1, \dots, \mathbf{v}_M$ are eigenvectors of $\mathbf{X}^H \mathbf{X}$; $\mathbf{u}_1, \dots, \mathbf{u}_N$ are eigenvectors of $\mathbf{X} \mathbf{X}^H$, for which the squares of the singular values $\sigma_1^2, \dots, \sigma_r^2$ of \mathbf{X} are the first r nonzero eigenvalues of $\mathbf{X}^H \mathbf{X}$ and $\mathbf{X} \mathbf{X}^H$, that is,

$$\lambda_i(\mathbf{X}^H \mathbf{X}) = \lambda_i(\mathbf{X} \mathbf{X}^H) = \sigma_i^2(\mathbf{X}) \quad (8.7.8)$$

4. In the product $\mathbf{X} = \mathbf{U} \Sigma \mathbf{V}^H$, the last $N - r$ columns of \mathbf{U} and $M - r$ columns of \mathbf{V} are superfluous because they interact only with blocks of zeros in Σ . This leads to the following *thin SVD* representation of \mathbf{X}

$$\mathbf{X} = \mathbf{U}_r \Sigma_r \mathbf{V}_r^H \quad (8.7.9)$$

where \mathbf{U}_r and \mathbf{V}_r consist of the first r columns of \mathbf{U} and \mathbf{V} , respectively, and $\Sigma_r = \text{diag}\{\sigma_1, \sigma_2, \dots, \sigma_r\}$.

5. The SVD can be expressed as

$$\mathbf{X} = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^H \quad (8.7.10)$$

that is, as a sum of cross products weighted by the singular values.

6. If the matrix \mathbf{X} has rank r , then:

- The first r columns of \mathbf{U} form an orthonormal basis for the space spanned by the columns of \mathbf{X} (*range space* or *column space* of \mathbf{X}).
- The first r columns of \mathbf{V} form an orthonormal basis for the space spanned by the rows of \mathbf{X} (*range space* of \mathbf{X}^H or *row space* of \mathbf{X}).
- The last $M - r$ columns of \mathbf{V} form an orthonormal basis for the space of vectors orthogonal to the rows of \mathbf{X} (*null space* of \mathbf{X}).
- The last $N - r$ columns of \mathbf{U} form an orthonormal basis for the *null space* of \mathbf{X}^H .

7. The Euclidean norm of \mathbf{X} is

$$\|\mathbf{X}\| = \sigma_1 \quad (8.7.11)$$

8. The Frobenius norm of \mathbf{X} , that is, the square root of the sum of the squares of its elements, is

$$\|\mathbf{X}\|_F \triangleq \sqrt{\sum_{i=1}^N \sum_{j=1}^M |x_{ij}|^2} = \sqrt{\sigma_1^2 + \sigma_2^2 + \dots + \sigma_r^2} \quad (8.7.12)$$

9. The difference between the transformations implied by eigenvalue and SVD transformations can be summarized as follows:

Eigenvalue decomposition			SVD		
$\mathbf{R} = \mathbf{Q}\Lambda\mathbf{Q}^H$			$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^H$		
			\mathbf{X}	\mathbf{X}^H	
\mathbf{q}_1	$\xrightarrow{\lambda_1}$	\mathbf{q}_1	\mathbf{v}_1	$\xrightarrow{\sigma_1}$	\mathbf{u}_1
\mathbf{q}_2	$\xrightarrow{\lambda_2}$	\mathbf{q}_2	\mathbf{v}_2	$\xrightarrow{\sigma_2}$	\mathbf{u}_2
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
\mathbf{q}_M	$\xrightarrow{\lambda_M}$	\mathbf{q}_M	\mathbf{v}_r	$\xrightarrow{\sigma_r}$	\mathbf{u}_r
			\mathbf{v}_{r+1}	$\left. \begin{array}{c} \vdots \\ \mathbf{v}_M \end{array} \right\} \rightarrow 0$	\mathbf{u}_{r+1}
					\vdots
					$\left. \begin{array}{c} \mathbf{u}_{r+1} \\ \vdots \\ \mathbf{u}_M \end{array} \right\} \rightarrow 0$

This illustrates the need for left and right singular values and vectors.

We can compute the SVD of a matrix \mathbf{X} by forming the matrices $\mathbf{X}^H\mathbf{X}$ and $\mathbf{X}\mathbf{X}^H$ and computing their eigenvalues and eigenvectors (see Problem 8.21). However, we should avoid this approach because the “squaring” of \mathbf{X} to form these correlation matrices results in a *loss of information* (see Example 8.6.2).

In practice, the SVD is computed by using the algorithm of Golub and Reinsch (1970) or the R-SVD algorithm described in Chan (1982), which for $N \gg M$ is twice as fast. The state of the art in SVD research is provided in Golub and Van Loan (1996), whereas reliable numerical algorithms and code are given in LA-PACK, LINPACK, and Numerical Recipes in C (Press et al. 1992).

8.7.2 Solution of the LS Problem

So far, we have discussed the solution of the overdetermined ($N > M$) LS problem with full-rank ($r = M$) data matrices using the normal equations and the QR decomposition techniques. We next show how the SVD can be used to solve the LS problem *without* making any assumptions about the dimensions N and M or the rank r of data matrix \mathbf{X} .

Suppose that we know the exact SVD of data matrix $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^H$. Since \mathbf{U} is orthogonal,

$$\|\mathbf{y} - \mathbf{X}\mathbf{c}\| = \|\mathbf{y} - \mathbf{U}\Sigma\mathbf{V}^H\mathbf{c}\| = \|\mathbf{U}^H\mathbf{y} - \Sigma\mathbf{V}^H\mathbf{c}\| \quad (8.7.13)$$

If we define

$$\mathbf{y}' \triangleq \mathbf{U}^H\mathbf{y} \quad \mathbf{c}' \triangleq \mathbf{V}^H\mathbf{c}$$

we obtain the LSE

$$\|\mathbf{y} - \mathbf{X}\mathbf{c}\|^2 = \|\mathbf{y}' - \Sigma\mathbf{c}'\|^2 = \sum_{i=1}^r |y'_i - \sigma_i c'_i|^2 + \sum_{i=r+1}^N |y'_i|^2 \quad (8.7.14)$$

which is minimized if and only if $c'_i = y'_i/\sigma_i$ for $i = 1, 2, \dots, r$. We notice that when $r < M$, the terms c'_{r+1}, \dots, c'_M do not appear in (8.7.14). Therefore, they have no effect on the residual and can be chosen arbitrarily. To illustrate this point, consider the geometric interpretation in Figure 8.5. There is only one linear combination of the linearly independent vectors $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$ that determines the optimum LS estimate. If the data matrix has one more column $\tilde{\mathbf{x}}_3$ that lies in the same plane, then there are an infinite number of linear combinations $c_1\tilde{\mathbf{x}}_1 + c_2\tilde{\mathbf{x}}_2 + c_3\tilde{\mathbf{x}}_3$ that satisfy the LSE criterion. To obtain a *unique* LS solution from all solutions \mathbf{c} that minimize $\|\mathbf{y} - \mathbf{X}\mathbf{c}\|$, we choose the one with the minimum length $\|\mathbf{c}\|$. Since

the matrix \mathbf{V} is orthogonal, we have $\|\mathbf{c}'\| = \|\mathbf{V}^H \mathbf{c}\| = \|\mathbf{c}\|$, and the norm $\|\mathbf{c}\|$ is minimized when the norm $\|\mathbf{c}'\|$ is minimized. Hence, choosing $c'_{r+1} = \dots = c'_M = 0$ provides the *minimum-norm solution* to the LS problem. In summary, the unique, minimum-norm solution to the LS problem is

$$\mathbf{c}_{\text{ls}} = \sum_{i=1}^r \frac{\mathbf{u}_i^H \mathbf{y}}{\sigma_i} \mathbf{v}_i \quad (8.7.15)$$

where $c'_i = \begin{cases} \frac{y'_i}{\sigma_i} = \frac{\mathbf{u}_i^H \mathbf{y}}{\sigma_i} & i = 1, \dots, r \\ 0 & i = r + 1, \dots, M \end{cases} \quad (8.7.16)$

and $E_{\text{ls}} = \|\mathbf{y} - \mathbf{X}\mathbf{c}_{\text{ls}}\|^2 = \sum_{i=r+1}^N |y'_i|^2 = \sum_{i=r+1}^N |\mathbf{u}_i^H \mathbf{y}|^2 \quad (8.7.17)$

is the corresponding LS error.

We next express the unique minimum-norm solution to the LS problem in terms of the pseudoinverse of data matrix \mathbf{X} using the SVD. To this end, we note that (8.7.16) can be written in matrix form

$$\mathbf{c}' = \Sigma^+ \mathbf{y}' \quad (8.7.18)$$

where $\Sigma^+ \triangleq \begin{bmatrix} \Sigma_r^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (8.7.19)$

is an $N \times N$ matrix with $\Sigma_r^{-1} = \text{diag}\{1/\sigma_1, \dots, 1/\sigma_r\}$. Therefore, using (8.7.15) and (8.7.19), we obtain

$$\mathbf{c}_{\text{ls}} = \mathbf{V} \Sigma^+ \mathbf{U}^H \mathbf{y} = \mathbf{X}^+ \mathbf{y} \quad (8.7.20)$$

where $\mathbf{X}^+ \triangleq \mathbf{V} \Sigma^+ \mathbf{U}^H = \sum_{i=1}^r \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^H \quad (8.7.21)$

is the pseudoinverse of matrix \mathbf{X} . For full-rank matrices, the pseudoinverse is defined as $\mathbf{X}^+ = (\mathbf{X}^H \mathbf{X})^{-1} \mathbf{X}^H$ (Golub and Van Loan 1996), so that using (8.7.21) leads to the LS solution in (8.2.21). If $N = M = \text{rank}(\mathbf{X})$, then $\mathbf{X}^+ = \mathbf{X}^{-1}$. Therefore, (8.7.21) holds for any rectangular or square matrix that is either full rank or rank-deficient. Formally, \mathbf{X}^+ can be defined independently of the LS problem as the unique $M \times N$ matrix \mathbf{A} that satisfies the four Moore-Penrose conditions

$$\begin{aligned} \mathbf{X} \mathbf{A} \mathbf{X}^+ &= \mathbf{X} & (\mathbf{X} \mathbf{A})^H &= \mathbf{X} \mathbf{A} \\ \mathbf{A} \mathbf{X} \mathbf{A}^+ &= \mathbf{A} & (\mathbf{A} \mathbf{X})^H &= \mathbf{A} \mathbf{X} \end{aligned} \quad (8.7.22)$$

which implies that $\mathbf{X} \mathbf{X}^+$ and $\mathbf{X}^+ \mathbf{X}$ are orthogonal projections onto the range space of \mathbf{X} and \mathbf{X}^H (see Problem 8.25). However, we stress that the pseudoinverse is, for the most part, a theoretical tool, and there is seldom any reason for its use in practice.

In summary, the computation of the LS estimator using the SVD involves the steps shown in Table 8.5. The vector \mathbf{c}_{ls} is unique and satisfies two requirements: (1) It minimizes the sum of the errors, and (2) it has the smallest Euclidean norm.

The following example illustrates the use of the SVD for the computation of the LS estimator.

EXAMPLE 8.7.1. Solve the LS problem with the following data matrix and desired response signal:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \\ 3 & 1 & 3 \\ 1 & 0 & 1 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 3 \end{bmatrix}$$

TABLE 8.5
**Solution of the LS problem using
the SVD method.**

Step	Description
1	Compute the SVD $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^H$
2	Determine the rank r of \mathbf{X}
3	Compute $y'_i = \mathbf{u}_i^H \mathbf{y}, i = 1, \dots, N$
4	Compute $\mathbf{c}_{ls} = \sum_{i=1}^r \frac{y'_i}{\sigma_i} \mathbf{v}_i$
5	Compute $E_{ls} = \sum_{i=r+1}^N y'_i ^2$

Solution. We start by computing the SVD of $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$ by using the MATLAB function $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = \text{svd}(\mathbf{X})$. This gives

$$\mathbf{U} = \begin{bmatrix} 0.3041 & 0.2170 & 0.8329 & 0.4082 \\ 0.4983 & 0.7771 & -0.3844 & 0.0000 \\ 0.7768 & -0.4778 & 0.0409 & -0.4082 \\ 0.2363 & -0.3474 & -0.3960 & 0.8165 \end{bmatrix}$$

$$\mathbf{\Sigma} = \begin{bmatrix} 5.5338 & 0 & 0 \\ 0 & 1.5139 & 0 \\ 0 & 0 & 0.2924 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{V} = \begin{bmatrix} 0.6989 & 0.3754 & -0.60882 \\ -0.0063 & 0.8544 & -0.5196 \\ -0.7152 & 0.3593 & 0.5994 \end{bmatrix}^T$$

which implies that the data matrix has rank $r = 3$. Next we compute

$$\mathbf{y}' = \mathbf{U}^T \mathbf{y} = \begin{bmatrix} 5.1167 \\ -1.1821 \\ -0.9602 \\ 1.2247 \end{bmatrix} \quad \mathbf{c}_{ls} = \begin{bmatrix} 3.0 \\ -1.5 \\ -1.0 \end{bmatrix} \quad E_{ls} = 1.5$$

by the MATLAB commands

```
yp=U' *y;
cls=V*(yp(1:r) ./diag(S));
Els=sum(yp(r+1:N).^2);
```

which implement steps 3, 4, and 5 in Table 8.5. The LS solution also can be obtained from $\mathbf{c}_{ls} = \mathbf{X} \setminus \mathbf{y}$. If we set $(\mathbf{X})_{23} = 2$, the first and last columns of \mathbf{X} become linearly dependent, the SVD has only two nonzero singular values, and the `svd` function warns that \mathbf{X} is rank-deficient.

Table 8.6 shows the numerical operations required by the various LS solution methods (Golub and Van Loan 1996). For full-rank (nonsingular) data matrices, all other methods are simpler than the SVD. However, these methods are inaccurate when \mathbf{X} is rank-deficient (nearly singular). In such cases, the SVD reveals the near singularity of the data matrix and is the method of choice because it provides a reliable computation of the numerical rank (see the next section).

Normal equations versus QR decomposition. The squaring of \mathbf{X} to form the time-average correlation matrix $\hat{\mathbf{R}} = \mathbf{X}^H \mathbf{X}$ results in a loss of information and should be avoided. Since $\|\mathbf{X}^{-1}\| = 1/\sigma_{\min}$, the condition number of \mathbf{X} is

$$\kappa(\mathbf{X}) = \|\mathbf{X}\| \|\mathbf{X}^{-1}\| = \frac{\sigma_{\max}}{\sigma_{\min}} \quad (8.7.23)$$

TABLE 8.6
Computational complexity of LS computation algorithms.

LS Algorithm	FLOPS (floating point operations)
Normal equations	$NM^2 + M^3/3$
Householder orthogonalization	$2NM^2 - 2M^3/3$
Givens orthogonalization	$3NM^2 - M^3$
Modified Gram-Schmidt	$2NM^2$
Golub-Reinsch SVD	$4NM^2 + 8M^3$
R-SVD	$2NM^2 + 11M^3$

which is analogous to the eigenvalue ratio for square Hermitian matrices. Hence,

$$\kappa(\mathbf{X}^H \mathbf{X}) = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{\sigma_{\max}^2}{\sigma_{\min}^2} = \kappa^2(\mathbf{X}) \quad (8.7.24)$$

which shows that squaring a matrix can only worsen its condition.

The study of the sensitivity of the LS problem is complicated. However, the following conclusions (Golub and Van Loan 1996; Van Loan 1997) can be drawn:

1. The sensitivity of the LS solution is roughly proportional to the quantity $\kappa(\mathbf{X}) + \sqrt{E_{ls}}\kappa^2(\mathbf{X})$. Hence, any method produces inaccurate results when applied to ill-conditioned problems with large E_{ls} .
2. The method of normal equations produces a solution \mathbf{c}_{ls} whose relative error is approximately $\text{eps} \cdot \kappa^2(\mathbf{X})$, where eps is the machine precision.
3. The QR method (Householder, Givens, MGS) produces a solution \mathbf{c}_{ls} whose relative error is approximately $\text{eps} \cdot [\kappa(\mathbf{X}) + \sqrt{E_{ls}}\kappa^2(\mathbf{X})]$.

In general, QR methods are more accurate than and can be used for a wider class of data matrices than the normal equations approach, even if the latter is about twice as fast.

In many practical applications, we need to update the Cholesky or QR decomposition after the original data matrix has been modified by the addition or deletion of a row or column (rank 1 modifications). Techniques for the efficient computation of these decompositions by updating the existing ones can be found in Golub and Van Loan (1996) and Gill et al. (1974).

8.7.3 Rank-Deficient LS Problems

In theory, it is relatively easy to determine the rank of a matrix or that a matrix is rank-deficient. However, both tasks become complicated in practice when the elements of the matrix are specified with inadequate accuracy or the matrix is near singular. The SVD provides the means of determining how close a matrix is to being rank-deficient, which in turn leads to the concept of numerical rank. To this end, suppose that the elements of matrix \mathbf{X} are known with an accuracy of order ϵ , and its computed singular values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_M$ are such that

$$\sigma_{r+1}^2 + \sigma_{r+2}^2 + \dots + \sigma_M^2 < \epsilon^2 \quad (8.7.25)$$

Then if we set $\Sigma_r \triangleq \text{diag}\{\sigma_1, \dots, \sigma_r, 0, \dots, 0\}$ and

$$\mathbf{X}_r \triangleq \mathbf{U} \begin{bmatrix} \Sigma_r \\ \mathbf{0} \end{bmatrix} \mathbf{V}^H \quad (8.7.26)$$

we have

$$\|\mathbf{X} - \mathbf{X}_r\|_F = \sqrt{\sigma_{r+1}^2 + \sigma_{r+2}^2 + \dots + \sigma_M^2} < \epsilon \quad (8.7.27)$$

and matrix \mathbf{X} is said to be near a matrix of rank r or \mathbf{X} has *numerical rank* r . It can be shown that \mathbf{X}_r is the matrix of rank r that is nearest to \mathbf{X} in the Frobenius norm sense (Leon

1990; Stewart 1973). This result has important applications in signal modeling and data compression.

Computing the LS solution for rank-deficient data matrices requires extra care. When a singular value is equal to a very small number, its reciprocal, which is a singular value of the pseudoinverse \mathbf{X}^+ , is a very large number. As a result, the LS solution deviates substantially from the “true” solution.

One way to handle this problem is to replace each singular value below a certain cutoff value (*thresholding*) with zero. A typical threshold is a fraction of σ_1 determined by either the machine precision available or the accuracy of the elements in the data matrix (measurement accuracy). For example, if the data matrix is accurate to six decimal places, we set the threshold at $10^{-6}\sigma_1$ (Golub and Van Loan 1996).

Another way is to replace the LS criterion (8.7.14) by

$$E\{\mathbf{c}, \psi\} = \|\mathbf{y} - \mathbf{X}\mathbf{c}\|^2 + \psi \|\mathbf{c}\|^2 \quad (8.7.28)$$

where the constant $\psi > 0$ reflects the importance of the norm of the solution vector. The term $\|\mathbf{c}\|$ acts a stabilizer, that is, prevents the solution \mathbf{c}_ψ from becoming too large (*regularization*). Indeed, using the method of Lagrange multipliers, we can show that

$$\mathbf{c}_\psi = \sum_{i=1}^r \frac{\sigma_i}{\sigma_i^2 + \psi} (\mathbf{u}_i^H \mathbf{y}) \mathbf{v}_i \quad (8.7.29)$$

which is known as the *regularized solution*. We note that $\mathbf{c}_\psi = \mathbf{c}_{ls}$ when $\psi = 0$. However, when $\psi > 0$, as $\sigma_i \rightarrow 0$ the term $\sigma_i/(\sigma_i^2 + \psi)$ in (8.7.29) tends to zero while the term $1/\sigma_i \rightarrow \infty$ in (8.7.15) tends to infinity. Furthermore, it can be shown that $\|\mathbf{c}_{ls}\| \leq \|\mathbf{y}\|/\sigma_r$ and $\|\mathbf{c}_\psi\| \leq \|\mathbf{y}\|/\sqrt{\psi}$ (Hager 1988).

Since the minimum-norm LS solution requires only the first r columns of \mathbf{U} , where r is the numerical rank of \mathbf{X} , we can use the thin SVD. If $N \gg M$, the computation of either \mathbf{U}_r or \mathbf{U} is expensive. However, in practical SVD algorithms, \mathbf{U} is computed as the product of many reflections and rotations. Hence, we can compute $\mathbf{y}' = \mathbf{U}^H \mathbf{y}$ by updating \mathbf{y} at each step i with each orthogonal transformation, that is, $\mathbf{U}_i^H \mathbf{y} \rightarrow \mathbf{y}$.

8.8 SUMMARY

In this chapter we discussed the theory, implementation, and application of linear estimators (combiners, filters, and predictors) that are optimum according to the LSE criterion of performance. The fundamental differences between linear MMSE and LSE estimators are as follows:

- MMSE estimators are designed using ensemble average second-order moments \mathbf{R} and \mathbf{d} ; they can be designed prior to operation, and during their normal operation they need only the input signals.
- LSE estimators are designed using time-average estimates $\hat{\mathbf{R}}$ and $\hat{\mathbf{d}}$ of the second-order moments or data matrix \mathbf{X} and the desired response vector \mathbf{y} . For this reason LSE estimators are sometimes said to be *data-adaptive*. The design and operation of LSE estimators are *coupled* and are usually accomplished by using either of the following approaches:
 - Collect a block of training data \mathbf{X}_{tr} and \mathbf{y}_{tr} and use them to design an LSE estimator; use it to process subsequent blocks. Clearly, this approach is meaningful if all blocks have statistically similar characteristics.
 - For each collected block of data \mathbf{X} and \mathbf{y} , compute the LSE filter \mathbf{c}_{ls} or the LSE estimate $\hat{\mathbf{y}}$ (whatever is needed).

There are various numerical algorithms designed to compute LSE estimators and estimates. For well-behaved data and sufficient numerical precision, all these methods produce

the same results and therefore provide the same LSE performance, that is, the same total squared error.

However, when ill-conditioned data, finite precision, or computational complexity is a concern, the choice of the LS computational algorithm is very important.

We saw that there are two major families of numerical algorithms for dealing with LS problems:

Power-domain techniques solve LS estimation problems using the time-average moments $\hat{\mathbf{R}} = \mathbf{X}^H \mathbf{X}$ and $\hat{\mathbf{d}} = \mathbf{X}^H \mathbf{y}$. The most widely used methods are the LDL^H and Cholesky decompositions.

Amplitude-domain techniques operate directly on data matrix \mathbf{X} and the desired response vector. In general, they require more computations and have better numerical properties than power-domain methods. This group includes the QR orthogonalization methods (Householder, Givens, and modified Gram-Schmidt) and the SVD method.

The QR decomposition methods apply a unitary transformation to the data matrix to reduce it to an upper triangular one, whereas the GS methods apply an upper triangular matrix transformation to orthogonalize the columns of the data matrix.

In conclusion, we emphasize that there are various ways to compute the coefficients of an optimum estimator and the value of the optimum estimate. We stress that the performance of any optimum estimator, as measured by the MMSE or LSE, does *not* depend on the particular implementation as long as we have sufficient numerical precision. Therefore, if we want to investigate how well an optimum estimator performs in a certain application, we can use any implementation, as long as computational complexity is not a consideration.

PROBLEMS

- 8.1 By differentiating (8.2.8) with respect to the vector \mathbf{c} , show that the LSE estimator \mathbf{c}_{ls} is given by the solution of the normal equations (8.2.12).
- 8.2 Let the weighted LSE be given by $E_w = \mathbf{e}^H \mathbf{W} \mathbf{e}$, where \mathbf{W} is a Hermitian positive definite matrix.
 - (a) By minimizing E_w with respect to the vector \mathbf{c} , show that the weighted LSE estimator is given by (8.2.35).
 - (b) Using the LDL^H decomposition $\mathbf{W} = \mathbf{L} \mathbf{D} \mathbf{L}^H$, show that the weighted LS criterion corresponds to prefiltering the error or the data.
- 8.3 Using direct substitution of (8.4.4) into (8.4.5), show that the LS estimator $\mathbf{c}_{ls}^{(i)}$ and the associated LS error $E_{ls}^{(i)}$ are determined by (8.4.5).
- 8.4 Consider a linear system described by the difference equation $y(n) = 0.9y(n-1) + 0.1x(n-1) + v(n)$, where $x(n)$ is the input signal, $y(n)$ is the output signal, and $v(n)$ is an output disturbance. Suppose that we have collected $N = 1000$ samples of input-output data and that we wish to estimate the system coefficients, using the LS criterion with no windowing. Determine the coefficients of the model $y(n) = ay(n-1) + dx(n-1)$ and their estimated covariance matrix $\hat{\sigma}_e^2 \hat{\mathbf{R}}^{-1}$ when
 - (a) $x(n) \sim \text{WGN}(0, 1)$ and $v(n) \sim \text{WGN}(0, 1)$ and
 - (b) $x(n) \sim \text{WGN}(0, 1)$ and $v(n) = 0.8v(n-1) + w(n)$ is an AR(1) process with $w(n) \sim \text{WGN}(0, 1)$. Comment upon the quality of the obtained estimates by comparing the matrices $\hat{\sigma}_e^2 \hat{\mathbf{R}}^{-1}$ obtained in each case.
- 8.5 Use Lagrange multipliers to show that Equation (8.4.13) provides the minimum of (8.4.8) under the constraint (8.4.9).

- 8.6** If full windowing is used in LS, then the autocorrelation matrix is Toeplitz. Using this fact, show that in the combined FBLP the predictor is given by

$$\mathbf{a}^{\text{fb}} = \frac{1}{2}(\mathbf{a} + \mathbf{J}\mathbf{b}^*)$$

- 8.7** Consider the noncausal “middle” sample linear signal estimator specified by (8.4.1) with $M = 2L$ and $i = L$.

- (a) Show that if we apply full windowing to the data matrix, the resulting signal estimator is conjugate symmetric, that is, $\mathbf{c}^{(L)} = \mathbf{J}\mathbf{c}^{(L)*}$. This property does not hold for any other windowing method.
- (b) Derive the normal equations for the signal estimator that minimizes the total squared error $E^{(L)} = \|\mathbf{e}^{(L)}\|^2$ under the constraint $\mathbf{c}^{(L)} = \mathbf{J}\mathbf{c}^{(L)*}$.
- (c) Show that if we enforce the normal equation matrix to be centro-Hermitian, that is, we use the normal equations

$$(\bar{\mathbf{X}}^H \bar{\mathbf{X}} + \mathbf{J} \bar{\mathbf{X}}^T \bar{\mathbf{X}}^* \mathbf{J}) \mathbf{c}^{(L)} = \begin{bmatrix} \mathbf{0} \\ E^{(L)} \\ \mathbf{0} \end{bmatrix}$$

then the resulting signal smoother is conjugate symmetric.

- (d) Illustrate parts (a) to (c), using the data matrix

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \\ 3 & 1 & 3 \\ 1 & 0 & 1 \\ 1 & 2 & 1 \end{bmatrix}$$

and check which smoother provides the smallest total squared error. Try to justify the obtained answer.

- 8.8** A useful impulse response for some geophysical signal processing applications is the Mexican hat wavelet

$$g(t) = \frac{2}{\sqrt{3}}\pi^{-1/4}(1-t^2)e^{-t^2/2}$$

which is the second derivative of a Gaussian pulse.

- (a) Plot the wavelet $g(t)$ and the magnitude and phase of its Fourier transform.
- (b) By examining the spectrum of the wavelet, determine a reasonable sampling frequency F_s .
- (c) Design an optimum LS inverse FIR filter for the discrete-time wavelet $g(nT)$, where $T = 1/F_s$. Determine a reasonable value for M by plotting the LSE E_M as a function of order M . Investigate whether we can improve the inverse filter by introducing some delay n_0 . Determine the best value of n_0 and plot the impulse response of the resulting filter and the combined impulse response $g(n) * h(n - n_0)$, which should resemble an impulse.
- (d) Repeat part (c) by increasing the sampling frequency by a factor of 2 and comparing with the results obtained in part (c).

- 8.9** (a) Prove Equation (8.5.4) regarding the LDL^H decomposition of the augmented matrix $\bar{\mathbf{R}}$.
 (b) Solve the LS estimation problem in Example 8.5.1, using the LDL^H decomposition of $\bar{\mathbf{R}}$ and the partitionings in (8.5.4).

- 8.10** Prove the order-recursive algorithm described by the relations given in (8.5.12). Demonstrate the validity of this approach, using the data in Example 8.5.1.

- 8.11** In this problem, we wish to show that the statistical interpretations of innovation and partial correlation for $w_m(n)$ and k_{m+1} in (8.5.12) hold in a deterministic LSE sense. To this end, suppose that the “partial correlation” between $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{x}}_{m+1}$ is defined using the residual records $\tilde{\mathbf{e}}_m = \tilde{\mathbf{y}} - \mathbf{X}_m \mathbf{c}_m$ and $\tilde{\mathbf{e}}_m^b = \tilde{\mathbf{x}}_{m+1} + \mathbf{X}_m \mathbf{b}_m$, where \mathbf{b}_m is the LSE BLP. Show that $k_{m+1} = \beta_{m+1}/\xi_{m+1}$, where $\beta_{m+1} \triangleq \tilde{\mathbf{e}}_m^H \tilde{\mathbf{e}}_m^b$ and $\xi_{m+1} = \tilde{\mathbf{e}}_m^b H \tilde{\mathbf{e}}_m^b$. Demonstrate the validity of these formulas using the data in Example 8.5.1.

- 8.12** Show that the Cholesky decomposition of a Hermitian positive definite matrix \mathbf{R} can be computed by using the following algorithm

```

for j = 1 to M
     $l_{ij} = (r_{ij} - \sum_{k=1}^{j-1} |l_{jk}|^2)^{1/2}$ 
for i = j + 1 to M
     $l_{ij} = (r_{ij} - \sum_{k=1}^{j-1} l_{ik}^* l_{jk}) / l_{jj}$ 
end i
end j

```

and write a MATLAB function for its implementation. Test your code using the built-in MATLAB function `chol`.

- 8.13** Compute the LDL^T and Cholesky decompositions of the following matrices:

$$\mathbf{X}_1 = \begin{bmatrix} 9 & 3 & -6 \\ 3 & 4 & 1 \\ -6 & 1 & 9 \end{bmatrix} \quad \text{and} \quad \mathbf{X}_2 = \begin{bmatrix} 6 & 4 & -2 \\ 4 & 5 & 3 \\ -2 & 3 & 6 \end{bmatrix}$$

- 8.14** Solve the LS problem in Example 8.6.1,

- (a) using the QR decomposition of the augmented data matrix $\tilde{\mathbf{X}} = [\mathbf{X} \ \mathbf{y}]$ and
- (b) using the Cholesky decomposition of the matrix $\bar{\mathbf{R}} = \tilde{\mathbf{X}}^H \tilde{\mathbf{X}}$.

Note: Use MATLAB built-in functions for the QR and Cholesky decompositions.

- 8.15** (a) Show that a unit vector \mathbf{w} is an eigenvector of the matrix $\mathbf{H} = \mathbf{I} - 2\mathbf{w}\mathbf{w}^H$. What is the corresponding eigenvalue?
(b) If a vector \mathbf{z} is orthogonal to \mathbf{w} , show that \mathbf{z} is an eigenvector of \mathbf{H} . What is the corresponding eigenvalue?

- 8.16** Solve the LS problem

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 2 \\ 1 & -1 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} -3 \\ 10 \\ 3 \\ 6 \end{bmatrix}$$

using the Householder transformation.

- 8.17** Solve Problem 8.16 by using the Givens transformation.

- 8.18** Compute the QR decomposition of the data matrix

$$\mathbf{X} = \begin{bmatrix} 4 & 2 & 1 \\ 2 & 0 & 1 \\ 2 & 0 & -1 \\ 1 & 2 & 1 \end{bmatrix}$$

using the GS and MGS methods, and compare the obtained results.

- 8.19** Solve the following LS problem

$$\mathbf{X} = \begin{bmatrix} 1 & -2 & -1 \\ 2 & 0 & 1 \\ 2 & -4 & 2 \\ 4 & 0 & 0 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -2 \end{bmatrix}$$

by computing the QR decomposition using the GS algorithm.

- 8.20** Show that the computational organization of the MGS algorithm shown in Table 8.4 can be used to compute the GS algorithm if we replace the step $r_{im} = \mathbf{q}_i^H \mathbf{x}_m$ by $r_{im} = \mathbf{q}_i^H \mathbf{q}_m$.

- 8.21** Compute the SVD of $\mathbf{X} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}$ by computing the eigenvalues and eigenvectors of $\mathbf{X}^H \mathbf{X}$ and $\mathbf{X} \mathbf{X}^H$. Check with the results obtained using the `svd` function.

- 8.22** Repeat Problem 8.21 for

$$(a) \mathbf{X} = \begin{bmatrix} 6 & 2 \\ -7 & 6 \end{bmatrix} \text{ and}$$

$$(b) \mathbf{X} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

- 8.23** Write a MATLAB program to produce the plots in Figure 8.14, using the matrix $\mathbf{X} = \begin{bmatrix} 6 & 2 \\ -7 & 6 \end{bmatrix}$.
Hint: Use a parametric description of the circle in polar coordinates.

- 8.24** For the matrix $\mathbf{X} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}^T$ determine \mathbf{X}^+ and verify that \mathbf{X} and \mathbf{X}^+ satisfy the four Moore-Penrose conditions (8.7.22).

- 8.25** Prove the four Moore-Penrose conditions in (8.7.22) and explain why $\mathbf{X}\mathbf{X}^+$ and $\mathbf{X}^+\mathbf{X}$ are orthogonal projections onto the range space of \mathbf{X} and \mathbf{X}^H .

- 8.26** In this problem we examine in greater detail the radio-frequency interference cancellation experiment discussed in Section 8.4.3. We first explain the generation of the various signals and then proceed with the design and evaluation of the LS interference canceler.

- (a) The useful signal is a pointlike target defined by

$$s(t) = \frac{d}{dt} \left(\frac{1}{e^{-\alpha t/t_r} + e^{\alpha t/t_f}} \right) \triangleq \frac{dg(t)}{dt}$$

where $\alpha = 2.3$, $t_r = 0.4$, and $t_f = 2$. Given that $F_s = 2$ GHz, determine $s(n)$ by computing the samples $g(n) = g(nT)$ in the interval $-2 \leq nT \leq 6$ ns and then computing the first difference $s(n) = g(n) - g(n-1)$. Plot the signal $s(n)$ and its Fourier transform (magnitude and phase), and check whether the pointlike and wideband assumptions are justified.

- (b) Generate $N = 4096$ samples of the narrowband interference using the formula

$$z(n) = \sum_{i=1}^L A_i \sin(\omega_i n + \phi_i)$$

and the following information:

```
Fs=2; % All frequencies are measured in GHz.
F=0.1*[0.6 1 1.8 2.1 3 4.8 5.2 5.7 6.1 6.4 6.7 7 7.8 9.3]';
L=length(F);
om=2*pi*F/Fs;
A=[0.5 1 0.5 0.1 0.3 0.5 1 1 1 0.5 0.3 1.5 0.5]';
rand('seed',1954);
phi=2*pi*rand(L,1);
```

- (c) Compute and plot the periodogram of $z(n)$ to check the correctness of your code.
 (d) Generate N samples of white Gaussian noise $v(n) \sim \text{WGN}(0, 0.1)$ and create the observed signal $x(n) = 5s(n - n_0) + z(n) + v(n)$, where $n_0 = 1000$. Compute and plot the periodogram of $x(n)$.
 (e) Design a one-step ahead ($D = 1$) linear predictor with $M = 100$ coefficients using the FBLP method with no windowing. Then use the obtained FBLP to clean the corrupted signal $x(n)$ as shown in Figure 8.7. To evaluate the performance of the canceler, generate the plots shown in Figures 8.8 and 8.9.

8.27 Careful inspection of Figure 8.9 indicates that the the D -step prediction error filter, that is, the system with input $x(n)$ and output $e^f(n)$, acts as a whitening filter. In this problem, we try to solve Problem 8.26 by designing a practical whitening filter using a power spectral density (PSD) estimate of the corrupted signal $x(n)$.

- (a) Estimate the PSD $\hat{R}_x^{(\text{PA})}(e^{j\omega_k})$, $\omega_k = 2\pi k/N_{\text{FFT}}$, of the signal $x(n)$, using the method of averaged periodograms. Use a segment length of $L = 256$ samples, 50 percent overlap, and $N_{\text{FFT}} = 512$.
- (b) Since the PSD does not provide any phase information, we shall design a whitening FIR filter with linear phase by

$$\tilde{H}(k) = \frac{1}{\sqrt{\hat{R}_x^{(\text{PA})}(e^{j\omega_k})}} e^{-j \frac{2\pi}{N_{\text{FFT}}} \frac{N_{\text{FFT}}-1}{2} k}$$

where $\tilde{H}(k)$ is the DFT of the impulse response of the filter, that is,

$$\tilde{H}(k) = \sum_{n=0}^{N_{\text{FFT}}-1} h(n) e^{-j \frac{2\pi}{N_{\text{FFT}}} nk}$$

with $0 \leq k \leq N_{\text{FFT}} - 1$.

- (c) Use the obtained whitening filter to clean the corrupted signal $x(n)$, and compare its performance with the FBLP canceler by generating plots similar to those shown in Figures 8.8 and 8.9.
- (d) Repeat part (c) with $L = 128$, $N_{\text{FFT}} = 512$ and $L = 512$, $N_{\text{FFT}} = 1024$ and check whether spectral resolution has any effect upon the performance. *Note:* Information about the design and implementation of FIR filters using the DFT can be found in Proakis and Manolakis (1996).

8.28 Repeat Problem 8.27, using the multitaper method of PSD estimation.

8.29 In this problem we develop an RFI canceler using a symmetric linear smoother with guard samples defined by

$$e(n) = x(n) - \hat{x}(n) \triangleq x(n) + \sum_{k=D}^M c_k [x(n-k) + x(n+k)]$$

where $1 \leq D < M$ prevents the use of the D adjacent samples to the estimation of $x(n)$.

- (a) Following the approach used in Section 8.4.3, demonstrate whether such a canceler can be used to mitigate RFI and under what conditions.
- (b) If there is theoretical justification for such a canceler, estimate its coefficients, using the method of LS with no windowing for $M = 50$ and $D = 1$ for the situation described in Problem 8.26.
- (c) Use the obtained filter to clean the corrupted signal $x(n)$, and compare its performance with the FBLP canceler by generating plots similar to those shown in Figures 8.8 and 8.9.
- (d) Repeat part (c) for $D = 2$.

8.30 In Example 6.7.1 we studied the design and performance of an optimum FIR inverse system. In this problem, we design and analyze the performance of a similar FIR LS inverse filter, using training input-output data.

- (a) First, we generate $N = 100$ observations of the input signal $y(n)$ and the noisy output signal $x(n)$. We assume that $y(n) \sim \text{WGN}(0, 1)$ and $v(n) \sim \text{WGN}(0, 0.1)$. To avoid transient effects, we generate 200 samples and retain the last 100 samples to generate the required data records.
- (b) Design an LS inverse filter with $M = 10$ for $0 \leq D < 10$, using no windowing, and choose the best value of delay D .
- (c) Repeat part (b) using full windowing.
- (d) Compare the LS filters obtained in parts (b) and (c) with the optimum filter designed in Example 6.7.1. What are your conclusions?

8.31 In this problem we estimate the equalizer discussed in Example 6.8.1, using input-output training data, and we evaluate its performance using Monte Carlo simulation.

- (a) Generate $N = 1000$ samples of input-desired response data $\{x(n), a(n)\}_0^{N-1}$ and use them to estimate the correlation matrix $\hat{\mathbf{R}}_x$ and the cross-correlation vector $\hat{\mathbf{d}}$ between $\mathbf{x}(n)$ and $y(n - D)$. Use $D = 7$, $M = 11$, and $W = 2.9$. Solve the normal equations to determine the LS FIR equalizer and the corresponding LSE.
- (b) Repeat part (a) 500 times; by changing the seed of the random number generators, compute the average (over the realizations) coefficient vector and average LSE, and compare with the optimum MSE equalizer obtained in Example 6.8.1. What are your conclusions?
- (c) Repeat parts (a) and (b) by setting $W = 3.1$.

Signal Modeling and Parametric Spectral Estimation

This chapter is a transition from theory to practice. It focuses on the selection of an appropriate model for a given set of data, the estimation of the model parameters, and how well the model actually “fits the data.” Although the development of parameter estimation techniques requires a strong theoretical background, the selection of a good model and its subsequent evaluation require the user to have sufficient practical experience and a familiarity with the intended application. We provide complete, detailed algorithms for fitting pole-zero models to data using least-squares techniques. The estimation of all-pole model parameters involves the solution of a linear system of equations, whereas pole-zero modeling requires nonlinear least-squares optimization. The chapter is roughly organized into two separate but related parts.

In the first part, we begin in Section 9.1 by explaining the steps that are required in the model-building process. Then, in Section 9.2, we introduce various least-squares algorithms for the estimation of parameters of direct and lattice all-pole models, provide different interpretations, and discuss some order selection criteria. For pole-zero models we provide, in Section 9.3, a nonlinear optimization algorithm that estimates the parameters of the model by minimizing the least-squares criterion. We conclude this part with Section 9.4 in which we discuss the applications of pole-zero models to spectral estimation and speech processing.

In the second part, we begin with the method of minimum-variance spectral estimation (Capon’s method). Then we describe frequency estimation methods based on the harmonic model: the Pisarenko harmonic decomposition and the MUSIC, minimum-norm, and ESPRIT algorithms. These methods are suitable for applications in which the signals of interest can be represented by *complex exponential* or *harmonic models*. Signals consisting of complex exponentials are found in a variety of applications including as formant frequencies in speech processing, moving targets in radar, and spatially propagating signals in array processing.

9.1 THE MODELING PROCESS: THEORY AND PRACTICE

In this section, we discuss the modeling of *real-world* signals using parametric pole-zero (PZ) signal models, whose theoretical properties were discussed in Chapter 4. We focus on PZ (P , Q) models with white input sequences, which are also known as ARMA (P , Q) random signal models. These models are defined by the linear constant-coefficient difference

equation

$$x(n) = - \sum_{k=1}^P a_k x(n-k) + w(n) + \sum_{k=1}^Q d_k w(n-k) \quad (9.1.1)$$

where $w(n) \sim \text{WN}(0, \sigma_w^2)$ with $\sigma_w^2 < \infty$. The power spectral density (PSD) of the output signal is

$$R(e^{j\omega}) = \sigma_w^2 \left| \frac{1 + \sum_{k=1}^Q d_k e^{-j\omega k}}{1 + \sum_{k=1}^P a_k e^{-j\omega k}} \right|^2 = \sigma_w^2 \frac{|D(e^{-j\omega})|^2}{|A(e^{-j\omega})|^2} \quad (9.1.2)$$

which is a rational function completely specified by the parameters, $\{a_1, a_2, \dots, a_P\}$, $\{d_1, \dots, d_Q\}$, and σ_w^2 . We stress that since these models are linear, time-invariant (LTI), the resulting process $x(n)$ is stationary, which is ensured if the corresponding systems are BIBO stable.

The essence of signal modeling and of the resulting parametric spectrum estimation is the following: Given finite-length data $\{x(n)\}_{n=0}^{N-1}$, which can be regarded as a sample sequence of the signal under consideration, we want to estimate signal model parameters $\{\hat{a}_k\}_1^P$, $\{\hat{b}_k\}_1^Q$, and $\hat{\sigma}_w^2$, to satisfy a prescribed criterion. Furthermore, if the parameter estimates are sufficiently accurate, then the following formula

$$\hat{R}(e^{j\omega}) = \hat{\sigma}_w^2 \left| \frac{1 + \sum_{k=1}^Q \hat{d}_k e^{-j\omega k}}{1 + \sum_{k=1}^P \hat{a}_k e^{-j\omega k}} \right|^2 = \hat{\sigma}_w^2 \frac{|\hat{D}(e^{-j\omega})|^2}{|\hat{A}(e^{-j\omega})|^2} \quad (9.1.3)$$

should provide a reasonable estimate of the signal PSD. A similar argument applies to harmonic signal models and harmonic spectrum estimation in which the model parameters are the amplitudes and frequencies of complex exponentials (see Section 3.3.6).

The development of such models involves the steps shown in Figure 9.1. In this chapter, we assume that we have removed trends, seasonal variations, and other nonstationarities from the data. We further assume that unit poles have been removed from the data by using the differencing approach discussed in Box et al. (1994).

Model selection

In this step, we basically select the structure of the model (direct or lattice), and we make a preliminary decision on the orders P and Q of the model. The most important aid to model selection is the insight and understanding of the signal and the physical mechanism that generates it. Hence, in some applications (e.g., speech processing) physical considerations point to the type and order of the model; when we lack a priori information or we have insufficient knowledge of the mechanism generating the signal, we resort to data analysis methods.

In general, to select a candidate model, we estimate the autocorrelation, partial autocorrelation, and power spectrum from the available data, and we compare them to the corresponding quantities obtained from the theoretical models (see Table 4.1). This preliminary data analysis provides sufficient information to choose a PZ model and some initial estimate for P and Q to start a model building process. Several order selection criteria have been developed that penalize both model misfit and a large number of parameters. Although theoretically interesting and appealing, these criteria are of limited value when we deal with actual signals.

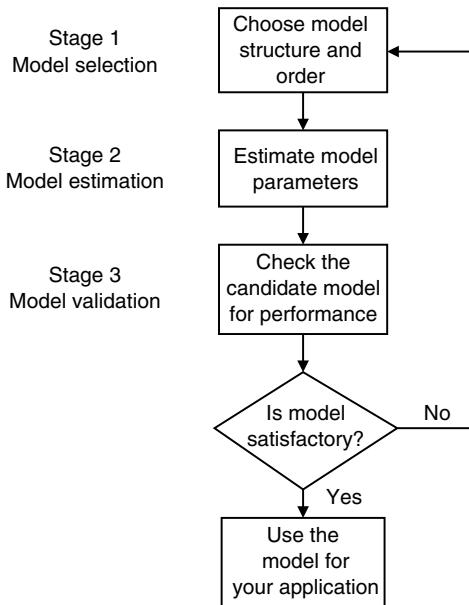


FIGURE 9.1
 Steps in the signal model building process.

The model structure influences (1) the complexity of the algorithm that estimates the model parameters and (2) the shape of the criterion function (quadratic or nonquadratic). Therefore, the structure (direct or lattice) is not critical to the performance of the model, and its choice is not as crucial as the choice of the order of the model.

Model estimation

In this step, also known as *model fitting*, we use the available data $\{x(n)\}_0^{N-1}$ to estimate the parameters of the selected model, using optimization of some criterion. Although there are several criteria (e.g., maximum likelihood, spectral matching) that can be used to measure the performance or quality of a PZ model, we concentrate on the least-squares (LS) error criterion. As we shall see, the estimation of all-pole (AP) models leads to linear optimization problems whereas the estimation of all-zero (AZ) and PZ models requires the solution of nonlinear optimization problems. Parameter estimation for PZ models using other criteria can be found in Kay (1988), Box et al. (1994), Porat (1994), and Ljung (1987).

Model validation

Here we investigate how well the obtained model captures the key features of the data. We then take corrective actions, if necessary, by modifying the order of the model, and repeat the process until we get an acceptable model. The goal of the model validation process is to find out whether the model

- Agrees sufficiently with the observed data
- Describes the “true” signal generation system
- Solves the problem that initiated the design process

Of course, the ultimate test is whether the model satisfies the requirements of the intended application, that is, the objective and subjective criteria that specify the performance of the model, computational complexity, cost, etc. In this discussion, we concentrate on how well the model fits the observed data in an LS error statistical sense.

The existence of any structure in the residual or prediction error signal indicates a misfit between the model and the data. Hence, a key validation technique is to check whether the residual process, which is generated by the inverse of the fitted model, is a realization of white noise. This can be checked by using, among others, the following statistical techniques (Brockwell and Davis 1991; Bendat and Piersol 1986):

Autocorrelation test. It can be shown (Kendall and Stuart 1983) that when N is sufficiently large, the distribution of the estimated autocorrelation coefficients $\hat{\rho}(l) = \hat{r}(l)/\hat{r}(0)$ is approximately Gaussian with zero mean and variance of $1/N$. The approximate 95 percent confidence limits are $\pm 1.96/\sqrt{N}$. Any estimated values of $\hat{\rho}(l)$ that fall outside these limits are “significantly” different from zero with 95 percent confidence. Values well beyond these limits indicate nonwhiteness of the residual signal.

Power spectrum density test. Given a set of data $\{x(n)\}_{n=0}^{N-1}$, the *standardized cumulative periodogram* is defined by

$$\tilde{I}(k) \triangleq \begin{cases} 0 & k < 1 \\ \frac{\sum_{i=1}^k \hat{R}(e^{j2\pi i/N})}{\sum_{i=1}^K \hat{R}(e^{j2\pi i/N})} & 1 \leq k \leq K \\ 1 & k > K \end{cases} \quad (9.1.4)$$

where K is the integer part of $N/2$. If the process $x(n)$ is white Gaussian noise (WGN), then the random variables $\tilde{I}(k)$, $k = 1, 2, \dots, K$, are independently and uniformly distributed in the interval $(0, 1)$, and the plot of $\tilde{I}(k)$ should be approximately linear with respect to k (Jenkins and Watts 1968). The hypothesis is rejected at level 0.05 if $\tilde{I}(k)$ exits the boundaries specified by

$$\tilde{I}^{(b)}(k) = \frac{k-1}{K-1} \pm 1.36(K-1)^{-1/2} \quad 1 \leq k \leq K \quad (9.1.5)$$

Partial autocorrelation test. This test is similar to the autocorrelation test. Given the residual process $x(n)$, it can be shown (Kendall and Stuart 1983) that when N is sufficiently large, the partial autocorrelation sequence (PACS) values $\{k_l\}$ for lag l [defined in (4.2.44)] are approximately independent with distribution WN $(0, 1/N)$. This means that roughly 95 percent of the PACS values fall within the bounds $\pm 1.96/\sqrt{N}$. If we observe values consistently well beyond this range for N sufficiently large, it may indicate nonwhiteness of the signal.

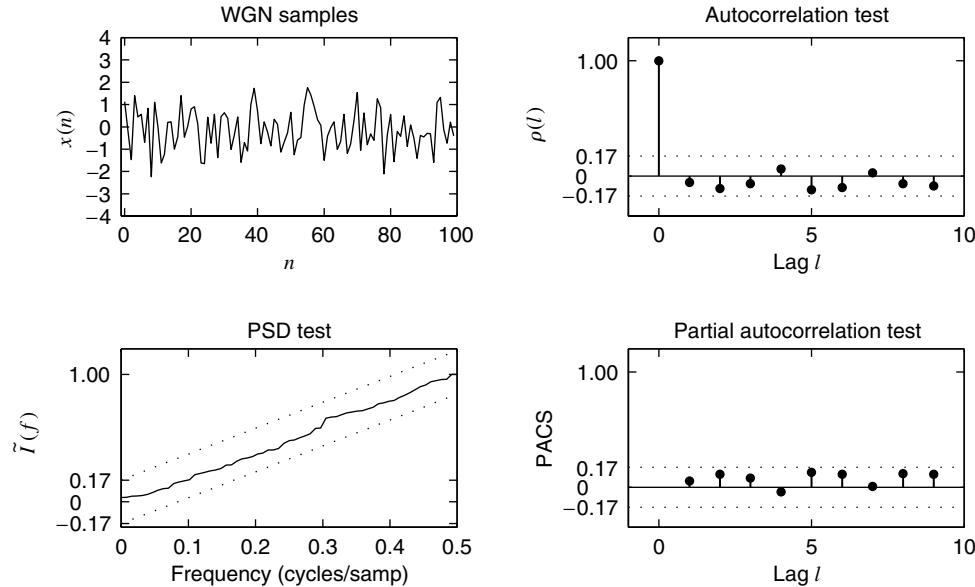
EXAMPLE 9.1.1. To apply the above tests and interpret their results, we consider a WGN sequence $x(n)$. By using the `randn` function, 100 samples of $x(n)$ with zero mean and unit variance were generated. These samples are shown in Figure 9.2. From these samples, the autocorrelation estimates up to lag 40, denoted by $\{\hat{r}(l)\}_{l=0}^{40}$, were computed using the `autoc` function, from which the correlation coefficients $\hat{\rho}(l)$ were obtained. The first 10 coefficients are shown in Figure 9.2 along with the appropriate confidence limits. As expected, the first coefficient at lag 0 is unity while the remaining coefficients are within the limits.

Next, using the `psd` function, a periodogram based on 100 samples was computed, from which the cumulative periodogram $\tilde{I}(k)$ was obtained and plotted as a function of the normalized frequency, as shown in Figure 9.2. The confidence limits are also shown. The computed cumulative periodogram is a monotonic increasing function lying within the limits.

Finally, using the `durbin` function, PACS sequence $\{k_l\}_{l=1}^{40}$ was computed from the estimated correlations and plotted in Figure 9.2. Again all the values for lags $l \geq 1$ are within the confidence limits. Thus all three tests suggest that the 100-point data are almost surely from a white noise sequence.

Although the whiteness of the residuals is a good test for model fitting, it does not provide a definite answer to the problem. Some additional procedures include checking whether

- The criterion of performance decreases (fast enough) as we increase the order of the model.

**FIGURE 9.2**

Validation tests on white Gaussian noise in Example 9.1.1.

- The estimate of the variance of the residual decreases as the number \$N\$ of observations increases.
- Some estimated parameters that have physical meaning (e.g., reflection coefficients) assume values that make sense.
- The estimated parameters have sufficient accuracy for the intended application.

Finally, to demonstrate that the model is sufficiently accurate for the purpose for which it was designed, we can use a method known as *cross-validation*. Basically, in cross-validation we use one set of data to fit the model and another, statistically independent set of data to test it. Cross-validation is of paramount importance when we build models for control, forecasting, and pattern recognition (Ljung 1987). However, in signal processing applications, such as spectral estimation and signal compression, where the goal is to provide a good fit of the model to the analyzed data, cross-validation is not as useful.

9.2 ESTIMATION OF ALL-POLE MODELS

We next use the principle of least squares to estimate parameters of all-pole signal models assuming both white and periodic excitations. We also discuss criteria for model order selection, techniques for estimation of all-pole lattice parameters, and the relationship between all-pole estimation methods using the methods of least squares and maximum entropy. The relationship between all-pole model estimation and minimum-variance spectral estimation is explored in Section 9.5.

9.2.1 Direct Structures

Consider the AR(\$P_0\$) model where we use \$a_k^*\$ instead of \$a_k\$ to comply with Chapter 8 notation.

$$x(n) = - \sum_{k=1}^{P_0} a_k^* x(n-k) + w(n) \quad (9.2.1)$$

where $w(n) \sim \text{WN}(0, \sigma_w^2)$. The P th-order linear predictor of $x(n)$ is given by

$$\hat{x}(n) = -\sum_{k=1}^P \hat{a}_k^* x(n-k) \quad (9.2.2)$$

and the corresponding prediction error sequence is

$$e(n) = x(n) - \hat{x}(n) = x(n) + \sum_{k=1}^P \hat{a}_k^* x(n-k) \quad (9.2.3)$$

$$= \hat{\mathbf{a}}^H \mathbf{x}(n) \quad (9.2.4)$$

where $\hat{a}_0 = 1$ and

$$\hat{\mathbf{a}} = [1 \ \hat{a}_1 \ \cdots \ \hat{a}_P]^T \quad (9.2.5)$$

$$\mathbf{x}(n) = [x(n) \ x(n-1) \ \cdots \ x(n-P)]^T \quad (9.2.6)$$

Thus the error over the range $N_i \leq n \leq N_f$ can be expressed as a vector

$$\mathbf{e} = \bar{\mathbf{X}} \hat{\mathbf{a}} \quad (9.2.7)$$

where $\bar{\mathbf{X}}$ is the data matrix defined in (8.4.3). For the full-windowing case, the data matrix $\bar{\mathbf{X}}$ is given by

$$\bar{\mathbf{X}}^H = \begin{bmatrix} x(0) & x(1) & \cdots & x(P) & \cdots & 0 & \cdots & 0 \\ 0 & x(0) & \cdots & x(P-1) & \cdots & x(N-1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x(0) & \cdots & x(N-P) & \cdots & x(N-1) \end{bmatrix} \quad (9.2.8)$$

while for the no-windowing case the data matrix $\bar{\mathbf{X}}$ is

$$\bar{\mathbf{X}}^H = \begin{bmatrix} x(P) & x(P+1) & \cdots & x(N-2) & x(N-1) \\ x(P-1) & x(P) & \cdots & x(N-3) & x(N-2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x(0) & x(1) & \cdots & x(N-P-2) & x(N-P-1) \end{bmatrix} \quad (9.2.9)$$

Notice that if $P = P_0$ and $\hat{a}_k = a_k$, the prediction error $e(n)$ is identical to the white noise excitation $w(n)$. Furthermore, if $\text{AR}(P_0)$ is minimum-phase, then $w(n)$ is the innovation process of $x(n)$ and $\hat{x}(n)$ is the MMSE prediction of $x(n)$. Thus, we can obtain a good estimate of the model parameters by minimizing some function of the prediction error.

In theory, we minimize the MSE $E\{|e(n)|^2\}$. In practice, since this is not possible, we estimate $\{a_k\}_1^P$ for a given P by minimizing the total squared error

$$\mathcal{E}_P = \sum_{n=N_i}^{N_f} |e(n)|^2 = \sum_{n=N_i}^{N_f} \left| x(n) + \sum_{k=1}^P \hat{a}_k^* x(n-k) \right|^2 \quad (9.2.10)$$

$$= \sum_{n=N_i}^{N_f} |\hat{\mathbf{a}}^H \mathbf{x}(n)|^2 = \hat{\mathbf{a}}^H \bar{\mathbf{X}}^H \bar{\mathbf{X}} \hat{\mathbf{a}} \quad (9.2.11)$$

over the range $N_i \leq n \leq N_f$. Hence, we can use the methods discussed in Section 8.4 for the computation of LS linear predictors. In particular, the forward linear predictor coefficient $\{\hat{a}_k\}_{k=1}^P$ and the associated LS error $\hat{\mathcal{E}}_P$ are obtained by solving the normal equations

$$(\bar{\mathbf{X}}^H \bar{\mathbf{X}}) \hat{\mathbf{a}} = \begin{bmatrix} \hat{\mathcal{E}}_P \\ \mathbf{0} \end{bmatrix} \quad (9.2.12)$$

The solution of (9.2.12) is discussed extensively in Chapter 8.

The least-squares AP(P) parameter estimates have properties similar to those of linear prediction. For example, if the process $w(n)$ is Gaussian, the least-squares no-windowing estimates are also maximum-likelihood estimates (Jenkins and Watts 1968). The variance of the excitation process can be obtained from the LS error $\hat{\mathcal{E}}_P$ by

$$\hat{\sigma}_w^2 = \frac{1}{N+P} \hat{\mathcal{E}}_P = \frac{1}{N+P} \sum_{n=0}^{N+P-1} |e(n)|^2 \quad \text{full windowing} \quad (9.2.13)$$

$$\text{or} \quad \hat{\sigma}_w^2 = \frac{1}{N-P} \hat{\mathcal{E}}_P = \frac{1}{N-P} \sum_{n=P}^{N-1} |e(n)|^2 \quad \text{no windowing} \quad (9.2.14)$$

for the full-windowing or no-windowing methods, respectively. Furthermore, in the full-windowing case, if the Toeplitz correlation matrix is positive definite, the obtained model is guaranteed to be minimum-phase (see Section 7.4). MATLAB functions

`[ahat,e,V] = arwin(x,P)` and `[ahat,e,V] = arls(x,P)`

are provided that compute the model parameters, the error sequence, and the modeling error using the full-windowing and no-windowing methods, respectively.

We present three examples below to illustrate the all-pole model determination and its use in PSD estimation. The first example uses real data consisting of water-level measurements of Lake Huron from 1875 to 1972. The second example also uses real data containing sunspot numbers for 1770 through 1869. These sunspot numbers have an approximate cycle of period around 10 to 12 years. The Lake Huron and sunspot data are shown in Figure 9.3. The third example generates simulated AR(4) data to estimate model parameters and through them the PSD values. In each case, the mean was computed and removed from the data prior to processing.

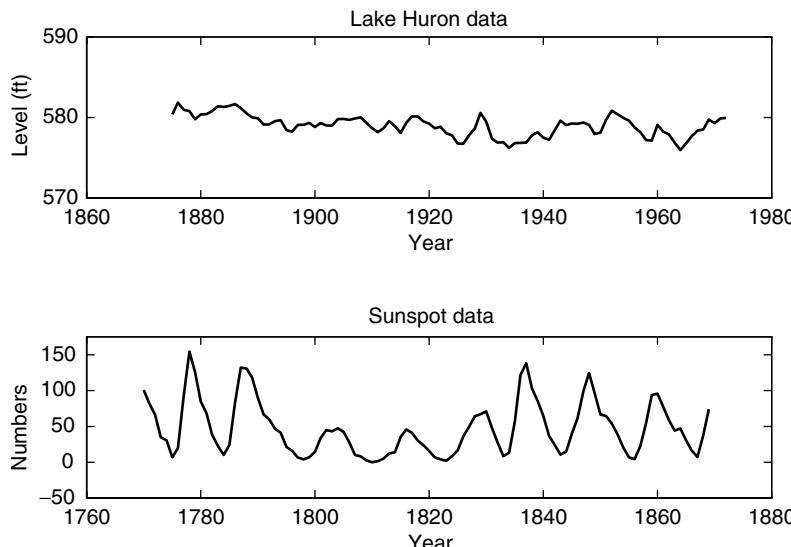


FIGURE 9.3

The Lake Huron and sunspot data used in Examples 9.2.1 and 9.2.2.

EXAMPLE 9.2.1. A careful examination of Lake Huron water-level measurement data indicates that a low-order all-pole model might be a suitable representation of the data. To test this hypothesis, first- and second-order models were considered. Using the full-windowing method, model

parameters were computed:

$$\begin{array}{lll} \text{First-order} & \hat{a}_1 = -0.791, & \hat{\sigma}_w^2 = 0.5024 \\ \text{Second-order} & \hat{a}_1 = -1.002, & \hat{a}_2 = 0.2832, \quad \hat{\sigma}_w^2 = 0.4460 \end{array}$$

Using these model parameters, the data were filtered and the residuals were computed. Three tests for checking the whiteness of the residuals as described in Section 9.1 were performed to ascertain the validity of models. In Figure 9.4, we show the residuals, the autocorrelation test, the PSD test, and the partial correlation test for the first-order model. The partial correlation test indicates that the PACS coefficient at lag 1 is outside the confidence limits and thus the first-order model is a poor fit. In Figure 9.5 we show the same plots for the second-order model. Clearly, these tests show that the residuals are approximately white. Therefore, the AR(2) model appears to be a good match to the data.

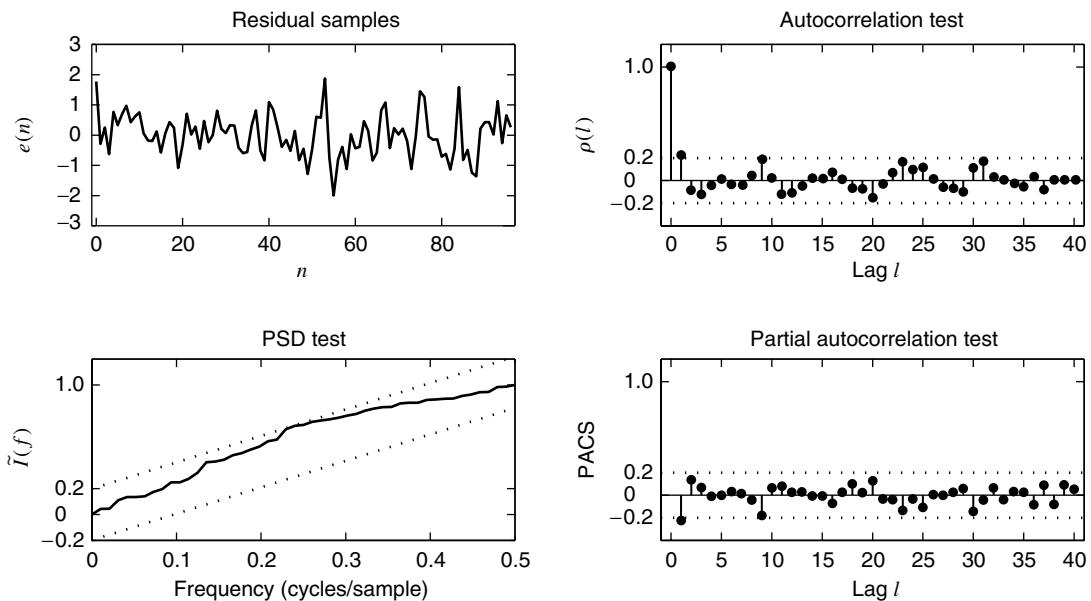


FIGURE 9.4

Validation tests on the first-order model fit to the Lake Huron water-level measurement data in Example 9.2.1.

EXAMPLE 9.2.2. Figure 9.6 shows the PACS coefficients of the sunspot numbers along with the 95 percent confidence limits. Since all PACS values beyond lag 2 fall well inside the limits, a second-order model is a possible candidate for the data. Therefore, the second-order model parameters were estimated from the data to obtain the model

$$x(n) = 1.318x(n-1) - 0.634x(n-2) + w(n) \quad \hat{\sigma}_w^2 = 289.2$$

In Figure 9.7 we show the residuals obtained by filtering the data along with three tests for its whiteness. The plots show that the estimated model is a reasonable fit to the data. Finally, in Figure 9.8 we show the PSD estimated from the AR(2) model as well as from the periodogram. The periodogram is very noisy and is devoid of any structure. The AR(2) spectrum is smoother and distinctly shows a peak at 0.1 cycle per sampling interval. Since the sampling rate is 1 sampling interval per year, the peak corresponds to 10 years per cycle, which agrees with the observations. Thus the parametric approach to PSD estimation was appropriate.

EXAMPLE 9.2.3. We illustrate the least-squares algorithms described above, using the AR(4) process $x(n)$ introduced in Example 5.3.2. The system function of the model is given by

$$H(z) = \frac{1}{1 - 2.7607z^{-1} + 3.8106z^{-2} - 2.6535z^{-3} + 0.9238z^{-4}}$$

and the excitation is a zero-mean Gaussian white noise with unit variance. Suppose that we are given the $N = 250$ samples of $x(n)$ shown in Figure 9.9 and we wish to model the underlying process by using an all-pole model. To identify a candidate model, we compute the autocorrelation, partial autocorrelation, and periodogram, using the available data. Careful inspection of Figure 9.9 and the signal model characteristics given in Table 4.1 suggests an AR model. Since the PACS plot cuts off around $P = 5$, we choose $P = 4$ and fit an AR(4) model to the data, using both the full-windowing and no-windowing methods. Figure 9.10 shows the actual spectrum of the process, the spectra of the estimated models, and the periodogram. Clearly, the no-windowing estimate provides a better fit because it does not impose any windowing on the data. Figure 9.11 shows the residual, autocorrelation, partial autocorrelation, and periodogram for the no-windowing-based model. We see that the residuals can be assumed uncorrelated with reasonable confidence, which implies that the model captures the second-order statistics of the data.

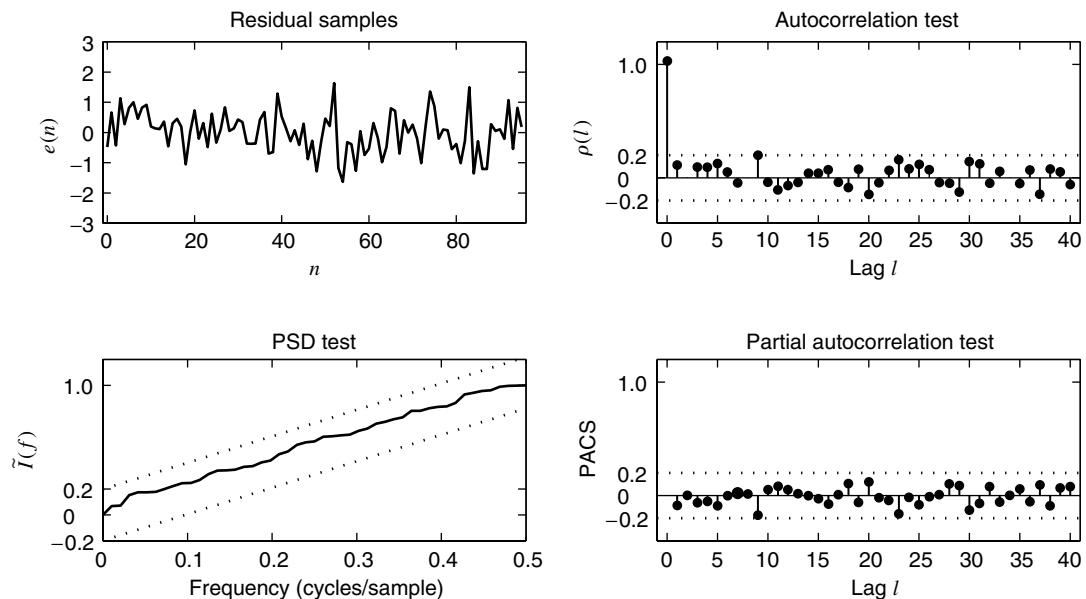


FIGURE 9.5

Validation tests on the second-order model fit to the Lake Huron water-level measurement data in Example 9.2.1.

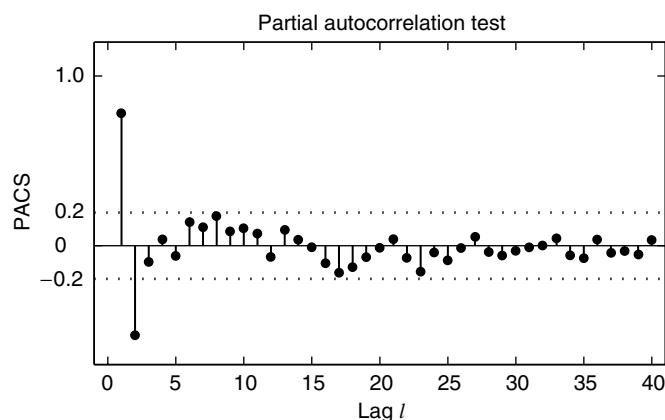


FIGURE 9.6

The PACS values of the sunspot numbers in Example 9.2.2.

Modified covariance method. The LS method described above to estimate model parameters uses the forward linear predictor and prediction error. There is also another approach that is based on the backward linear predictor. Recall that the backward linear predictor derived from the known correlations is the complex conjugate of the forward predictor (and likewise, the corresponding errors are identical). However, the LS estimators and errors based on the actual data are different because the data read in each direction are *different* from a statistical viewpoint. Hence, it is much more reasonable to consider both forward and backward predictors and to minimize the combined error

$$\begin{aligned}\mathcal{E}_P^{\text{fb}} &\triangleq \sum_{n=N_i}^{N_f} [|e^f(n)|^2 + |e^b(n)|^2] \\ &= \sum_{n=N_i}^{N_f} [|\hat{\mathbf{a}}^H \mathbf{x}(n)|^2 + |\hat{\mathbf{a}}^T \mathbf{x}^*(n)|^2] \\ &= \hat{\mathbf{a}}^H \bar{\mathbf{X}}^H \bar{\mathbf{X}} \hat{\mathbf{a}} + \hat{\mathbf{a}}^T \bar{\mathbf{X}}^T \bar{\mathbf{X}}^* \hat{\mathbf{a}}\end{aligned}\quad (9.2.15)$$

subject to the constraint that the first component of $\hat{\mathbf{a}}$ is 1. The minimization of $\mathcal{E}_P^{\text{fb}}$ leads to the set of normal equations

$$(\bar{\mathbf{X}}^H \bar{\mathbf{X}} + \bar{\mathbf{X}}^T \bar{\mathbf{X}}^*) \hat{\mathbf{a}} = \begin{bmatrix} \hat{\mathcal{E}}_P^{\text{fb}} \\ \mathbf{0} \end{bmatrix} \quad (9.2.16)$$

which can be solved efficiently to obtain the model parameters (see Section 8.4.2). This method of using the forward-backward predictors is called the *modified covariance method*. Not only does it have the advantage of minimizing the combined global error, but also since it uses more data in (9.2.16), it gives better estimates and lower error. A similar minimization approach, but implemented at each local stage, is used in Burg's method, which is discussed in Section 9.2.2.

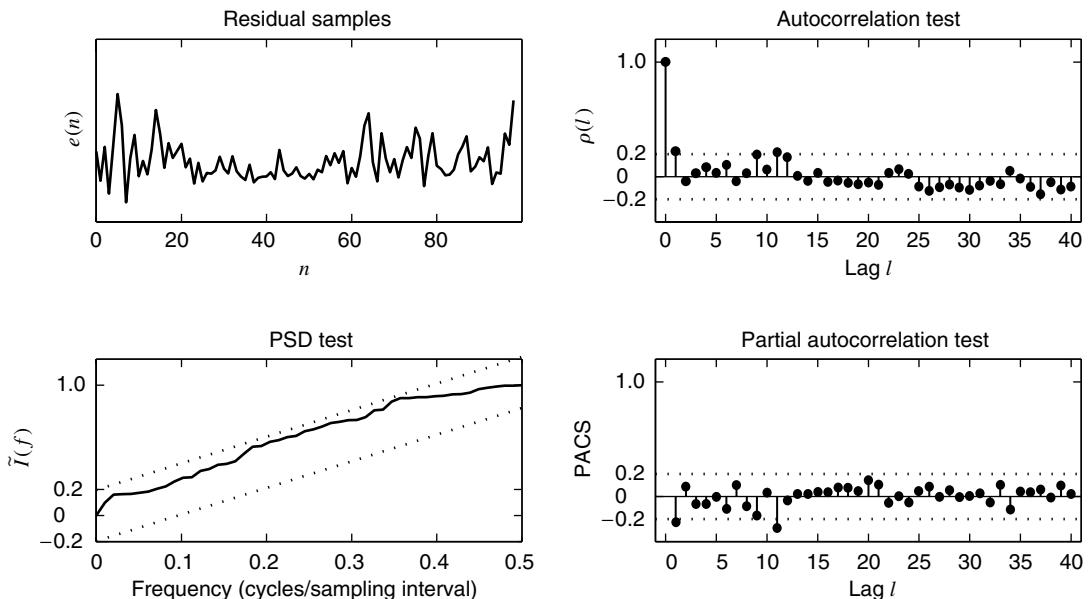
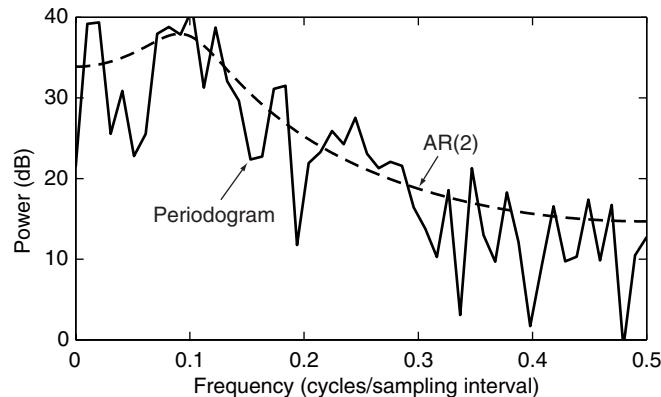
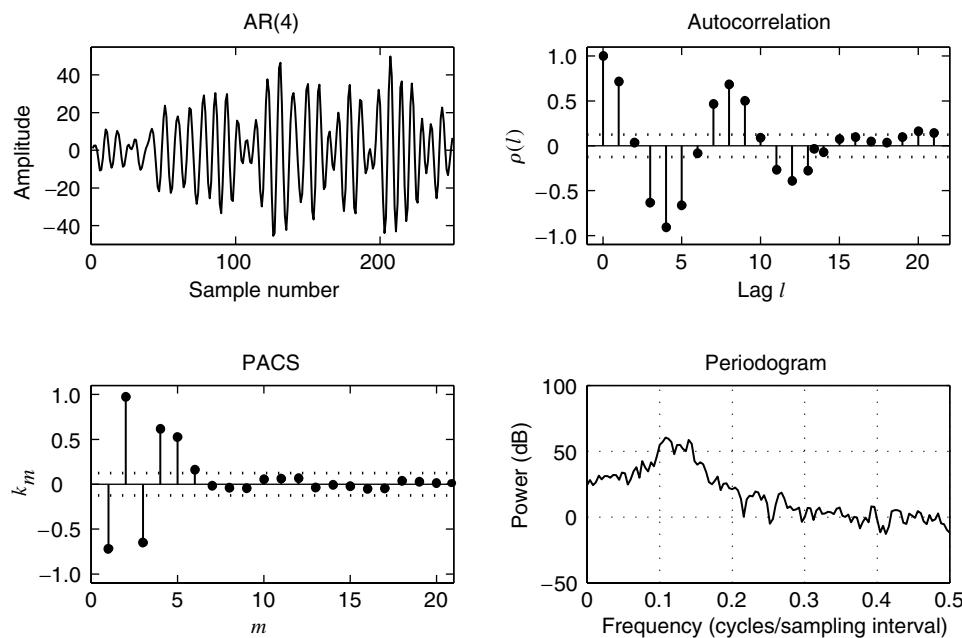


FIGURE 9.7

Validation tests on the second-order model fit to the sunspot numbers in Example 9.2.2.

**FIGURE 9.8**

Comparison of the periodogram and the AR(2) spectrum in Example 9.2.2.

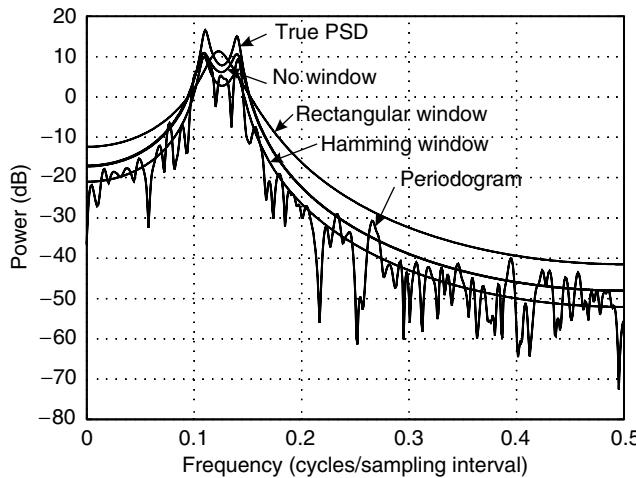
**FIGURE 9.9**

Data segment from an AR(4) process, and the corresponding autocorrelation, partial autocorrelation, and periodogram.

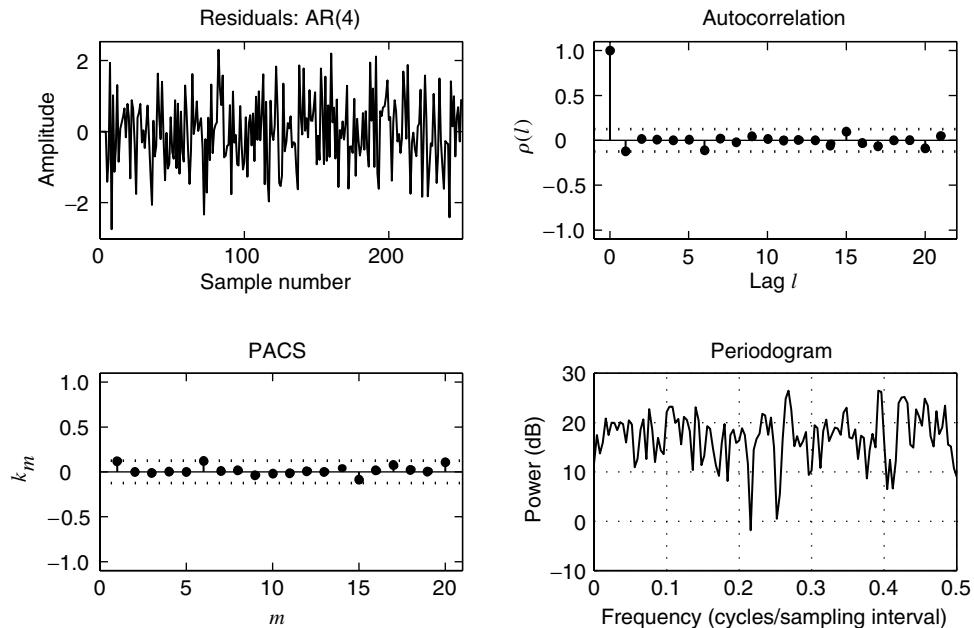
Frequency-domain interpretation. In the case of full windowing, by using Parseval's theorem, the error energy can be written as

$$\mathcal{E} = \sum_{n=-\infty}^{\infty} |e(n)|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{|X(e^{j\omega})|^2}{|\hat{H}(e^{j\omega})|^2} d\omega \quad (9.2.17)$$

where $|X(e^{j\omega})|^2$ is the spectrum of the modeled windowed signal segment and $\hat{H}(e^{j\omega})$ is the frequency response of the estimated all-pole model [or estimated spectrum of $x(n)$]. This expression is a good approximation for the other windowing methods if $N \gg P$. Since the integrand in (9.2.17) is positive, minimizing the error \mathcal{E} is equivalent to minimizing the

**FIGURE 9.10**

Periodogram, theoretical AR(4) spectrum, and AR(4) model spectra using full windowing, Hamming windowing, and no windowing.

**FIGURE 9.11**

Residual sequence for the AR(4) data, and the corresponding autocorrelation, partial autocorrelation, and periodogram.

integrated ratio of the energy spectrum of the modeled signal segment to its all-pole-based spectrum.

The presence of this ratio in (9.2.17) has three additional consequences. (1) The quality of the spectral matching is *uniform* over the whole frequency range, irrespective of the shape of the spectrum. (2) Since regions where $|X(e^{j\omega})| > |\hat{H}(e^{j\omega})|$ contribute more to the total error than regions where $|X(e^{j\omega})| < |\hat{H}(e^{j\omega})|$ do, the match is better near spectral peaks than near spectral valleys. (3) The all-pole model provides a good estimate of the *envelope* of the signal spectrum $|X(e^{j\omega})|^2$. These properties are apparent in Figure 9.12,

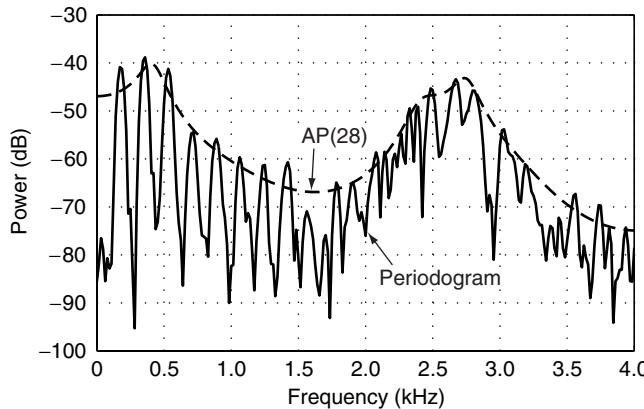
**FIGURE 9.12**

Illustration of the spectral envelope matching property of all-pole models.

which shows a comparison between $20 \log |X(e^{j\omega})|$ (obtained using the periodogram) and $20 \log |\hat{H}(e^{j\omega})|$ [obtained by an AP(28) model fitted using full windowing] for a 20-ms, Hamming windowed, speech signal sampled at 20 kHz. Note that the slope of $|\hat{H}(e^{j\omega})|$ is always zero at frequencies $\omega = 0$ and $\omega = \pi$, as expected. More details on these issues can be found in Makhoul (1975b).

The error energy (9.2.17) is also related to the Itakura-Saito (IS) distortion measure, which is given by

$$d_{\text{IS}}(R_1, R_2) \triangleq \frac{1}{2\pi} \int_{-\pi}^{\pi} [\exp V(e^{j\omega}) - V(e^{j\omega}) - 1] d\omega \quad (9.2.18)$$

where $R_1(e^{j\omega})$ and $R_2(e^{j\omega})$ are two spectra, and

$$V(e^{j\omega}) \triangleq \log R_1(e^{j\omega}) - \log R_2(e^{j\omega}) \quad (9.2.19)$$

Indeed, we can show that

$$d_{\text{IS}}(R_1, R_2) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{R_1(e^{j\omega})}{R_2(e^{j\omega})} d\omega - \log \frac{\sigma_1^2}{\sigma_2^2} - 1 \quad (9.2.20)$$

where σ_1^2 and σ_2^2 are the variances of the innovation sequences corresponding to the factorization of spectra $R_1(e^{j\omega})$ and $R_2(e^{j\omega})$, respectively. More details can be found in Rabiner and Juang (1993).

Order selection criteria. The order of an all-pole signal model plays an important role in the modeling problem. It determines the number of parameters to be estimated and hence the computational complexity of the algorithm. But more importantly, it affects the quality of the spectrum estimates. If a much lower order is selected, then the resulting spectrum will be smooth and will display poor resolution. If a much larger order is used, then the spectrum may contain spurious peaks at best and a phenomenon called *spectrum splitting* at worst, in which a single peak is split into two separate and distinct peaks (Hayes 1996).

Several criteria have been proposed over the years for model order selection; however, in practice nothing surpasses the graphical approach outlined in Examples 9.2.1 and 9.2.2 combined with the experience of the user. Therefore, we only provide a brief summary of some well-known criteria and refer the interested reader to Kay (1988), Porat (1994), and Ljung (1987) for more details. The simplest approach would be to monitor the modeling error and then select the order at which this error enters a steady state. However, for all-pole models, the modeling error is monotonically decreasing, which makes this approach all but

impossible. The general idea behind the suggested criterion is to introduce a penalty function in the modeling error that increases with the model order P . We present the following four criteria that are based on the above general idea.

FPE criterion. The *final prediction error (FPE)* criterion, proposed by Akaike (1970), is based on the function

$$\text{FPE}(P) = \frac{N + P}{N - P} \hat{\sigma}_P^2 \quad (9.2.21)$$

where $\hat{\sigma}_P^2$ is the modeling error [or variance of the residual of the estimated AP(P) model]. We note that the term $\hat{\sigma}_P^2$ decreases or remains the same with increasing P , whereas the term $(N + P)/(N - P)$ accounts for the increase in $\hat{\sigma}_P^2$ due to inaccuracies in the estimated parameters and increases with P . Clearly, FPE(P) is an inflated version of $\hat{\sigma}_P^2$. The FPE order selection criterion is to choose P that will minimize the function in (9.2.21).

AIC. The *Akaike information criterion (AIC)*, also introduced by Akaike (1974), is based on the function

$$\text{AIC}(P) = N \log \hat{\sigma}_P^2 + 2P \quad (9.2.22)$$

It is a very general criterion that provides an estimate of the Kullback-Leibler distance (Kullback 1959) between an assumed and the true probability density function of the data. The performances of the FPE criterion and the AIC are quite similar.

MDL criterion. The *minimum description length (MDL)* criterion was proposed by Risannen (1978) and uses the function

$$\text{MDL}(P) = N \log \hat{\sigma}_P^2 + P \log N \quad (9.2.23)$$

The first term in (9.2.23) decreases with P , but the second penalty term increases. It has been shown (Risannen 1978) that this criterion provides a consistent order estimate in that as the probability that the estimated order is equal to the true order approaches 1, the data length N tends to infinity.

CAT. This criterion is based on Parzen's *criterion autoregressive transfer (CAT)* function (Parzen 1977), which is given by

$$\text{CAT}(P) = \frac{1}{N} \sum_{k=1}^P \frac{N - k}{N \hat{\sigma}_k^2} - \frac{N - P}{N \hat{\sigma}_P^2} \quad (9.2.24)$$

This criterion is asymptotically equivalent to the AIC and the MDL criteria.

Basically, all order selection criteria add to the variance of the residuals a term that grows with the order of the model and estimate the order of the model by minimizing the resulting criterion. However, when $P \ll N$, which is the case in many practical applications, the criterion does not exhibit a clear minimum that makes the order selection process difficult (see Problem 9.1).

9.2.2 Lattice Structures

We noted in Section 7.5 that a prediction error filter, and hence the AP model, can also be implemented by using a lattice structure. The P th-order forward prediction error $e(n) = e_P^f(n)$ and the total squared error

$$\mathcal{E}_P = \sum_{n=N_i}^{N_f} |e(n)|^2 \quad (9.2.25)$$

are nonlinear functions of the lattice parameters k_m , $0 \leq m \leq P - 1$. For example, if $P = 2$, we have

$$e_2^f(n) = x(n) + (k_0^* + k_0 k_1^*)x(n-1) + k_1^*x(n-2)$$

which shows that $e_2^f(n)$ depends on the product $k_0 k_1^*$. Thus, fitting an all-pole lattice model by minimizing \mathcal{E}_P with respect to k_m , $0 \leq m \leq P - 1$, leads to a difficult nonlinear optimization problem.

We can avoid this problem by replacing the above “global” optimization with P “local” optimizations from $m = 1$ to P , one for each stage of the lattice. From the lattice equations

$$e_m^f(n) = e_{m-1}^f(n) + k_{m-1}^* e_{m-1}^b(n-1) \quad (9.2.26)$$

$$e_m^b(n) = e_{m-1}^b(n-1) + k_{m-1} e_{m-1}^f(n) \quad (9.2.27)$$

we see that the m th-order prediction errors depend on the coefficient k_{m-1} only. Furthermore, the values of $e_{m-1}^f(n)$ and $e_{m-1}^b(n)$ have been computed by using k_{m-2} , which has been determined from the optimization step at the previous stage.

Hence, to minimize the forward prediction error

$$\mathcal{E}_m^f = \sum_{n=N_i}^{N_f} |e_m^f(n)|^2 \quad (9.2.28)$$

we substitute (9.2.26) into (9.2.28) and differentiate[†] with respect to k_{m-1}^* . This leads to the following optimum value of k_{m-1}

$$k_{m-1}^{\text{FP}} = -\frac{\beta_{m-1}^{\text{fb}}}{\mathcal{E}_{m-1}^b} \quad (9.2.29)$$

where

$$\beta_m^{\text{fb}} = \sum_{n=N_i}^{N_f} [e_m^f(n)]^* e_m^b(n-1) \quad (9.2.30)$$

and

$$\mathcal{E}_m^b = \sum_{n=N_i}^{N_f} |e_m^b(n-1)|^2 \quad (9.2.31)$$

Similarly, minimization of the backward prediction error (9.2.31) gives

$$k_{m-1}^{\text{BP}} = -\frac{\beta_{m-1}^{\text{fb}}}{\mathcal{E}_{m-1}^f} \quad (9.2.32)$$

Burg (1967) suggested the estimation of k_{m-1} by minimizing

$$\mathcal{E}_m^{\text{fb}} = \sum_{n=N_i}^{N_f} \{|e_m^f(n)|^2 + |e_m^b(n)|^2\} \quad (9.2.33)$$

at each stage of the lattice.[‡] Indeed, substituting (9.2.26) and (9.2.27) in the last equation, we obtain the relationship

$$\mathcal{E}_m^{\text{fb}} = (1 + |k_{m-1}|^2)\mathcal{E}_{m-1}^f + 4 \operatorname{Re}(k_{m-1}^* \beta_{m-1}^{\text{fb}}) + (1 + |k_{m-1}|^2)\mathcal{E}_{m-1}^b \quad (9.2.34)$$

[†]See Appendix B for a discussion of how to find an optimum of a real-valued function of a complex variable and its conjugate.

[‡]This approach should not be confused with the maximum entropy method introduced also by Burg and discussed later.

If we set $\partial \mathcal{E}_m^{\text{fb}} / \partial k_{m-1}^* = 0$, we obtain the following estimate of k_{m-1} :

$$k_{m-1}^B = -\frac{\beta_{m-1}^{\text{fb}}}{\frac{1}{2}(\mathcal{E}_{m-1}^f + \mathcal{E}_{m-1}^b)} = \frac{2k_{m-1}^{\text{FP}}k_{m-1}^{\text{BP}}}{k_{m-1}^{\text{FP}} + k_{m-1}^{\text{BP}}} \quad (9.2.35)$$

We note that k_{m-1}^B is the harmonic mean of k_{m-1}^{FP} and k_{m-1}^{BP} . We also stress that the obtained model is *different* from the one resulting from the forward-backward least-squares (FBLS) method through global optimization [see (9.2.16)].

Itakura and Saito (1971) proposed an estimate of k_{m-1} based on replacing the theoretical ensemble averages in (7.5.24) by time averages. Their estimate is given by

$$k_{m-1}^{\text{IS}} = -\frac{\beta_{m-1}^{\text{fb}}}{\sqrt{\mathcal{E}_{m-1}^f \mathcal{E}_{m-1}^b}} = \text{sign}(k_{m-1}^{\text{FP}} \text{ or } k_{m-1}^{\text{BP}}) \sqrt{k_{m-1}^{\text{FP}} k_{m-1}^{\text{BP}}} \quad (9.2.36)$$

and is also known as the geometric mean method. Since it can be shown that

$$|k_{m-1}^B| \leq |k_{m-1}^{\text{IS}}| \leq 1 \quad (9.2.37)$$

both estimates result in minimum-phase models (see Problem 9.2). From (9.2.36) and (9.2.37) we conclude that if $|k_{m-1}^{\text{FP}}| < 1$, then $|k_{m-1}^{\text{BP}}| > 1$ and vice versa; that is, if the FLP is minimum-phase, then the BLP is maximum-phase and vice versa. Several other estimates are discussed in Makhoul (1977) and Viswanathan and Makhoul (1975).

In all previous methods, we use *no windowing*; that is, we set $N_i = m$ and $N_f = N-1$. If we use data windowing, all the above estimates are identical to the data windowing estimates obtained using the algorithm of Levinson-Durbin (see Problem 9.3).

The variance of the residuals can be estimated by

$$\hat{\sigma}_m^2 = \frac{1}{2} \frac{\mathcal{E}_m^{\text{fb}}}{N-m} \quad (9.2.38)$$

which for large values of N (see Problem 9.12) can be approximated by

$$\hat{\sigma}_m^2 = \hat{\sigma}_{m-1}^2 (1 - |k_{m-1}|^2) \quad (9.2.39)$$

where

$$\hat{\sigma}_0^2 = \frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2 \quad (9.2.40)$$

The computations for the lattice estimation methods are summarized in Table 9.1, and the algorithms are implemented by the function `[k, var] = aplatest(x, P)`.

9.2.3 Maximum Entropy Method

We next show how LS all-pole modeling is related to Burg's method of maximum entropy. To this end, suppose that $x(n)$ is a normal, stationary process with zero mean. The M -dimensional complex-valued vector $\mathbf{x} \triangleq \mathbf{x}_M(n)$ obeys a normal distribution

$$p(\mathbf{x}) = \frac{1}{\pi^M \det \mathbf{R}} \exp(-\mathbf{x}^H \mathbf{R}^{-1} \mathbf{x}) \quad (9.2.41)$$

where \mathbf{R} is a Toeplitz correlation matrix. By definition, its entropy is given by

$$\mathcal{H}(\mathbf{x}) \triangleq -E\{\log p(\mathbf{x})\} = M \log \pi + \log(\det \mathbf{R}) + M \quad (9.2.42)$$

because $E\{\mathbf{x}^H \mathbf{R}^{-1} \mathbf{x}\} = M$. If the process $x(n)$ is regular, that is, $|k_m| < 1$ for all m , we have

$$\det \mathbf{R} = \prod_{m=0}^{M-1} P_m \quad \text{and} \quad P_m = r(0) \prod_{j=1}^m (1 - |k_j|^2) \quad (9.2.43)$$

Algorithm for estimation of AP lattice parameters.

1. **Input:** $x(n)$ for $N_i \leq n \leq N_f$
2. **Initialization**
 - a. $e_0^f(n) = e_0^b(n) = x(n).$
 - b. Compute β_0^{fb} , \mathcal{E}_0^f , and \mathcal{E}_0^b from $x(n).$
 - c. Compute k_0^{FP} and $k_0^{BP}.$
 - d. Compute either k_0^{IS} or k_0^B from k_0^{FP} and $k_0^{BP}.$
 - e. Apply the first stage of the lattice to $x(n)$ using either k_0^{IS} or k_0^B to obtain $e_1^f(n)$ and $e_1^b(n).$
3. **For** $m = 2, 3, \dots, P$
 - a. Compute β_{m-1}^{fb} , \mathcal{E}_{m-1}^f , and \mathcal{E}_{m-1}^b from $e_{m-1}^f(n)$ and $e_{m-1}^b(n).$
 - b. Compute k_{m-1}^{FP} and $k_{m-1}^{BP}.$
 - c. Compute either k_{m-1}^{IS} or k_{m-1}^B from k_{m-1}^{FP} and $k_{m-1}^{BP}.$
 - d. Apply the m th stage of the lattice to $e_{m-1}^f(n)$ and $e_{m-1}^b(n)$ using either k_{m-1}^{IS} or k_{m-1}^B to obtain $e_m^f(n)$ and $e_m^b(n).$
4. **Output:** Either k_m^{IS} or k_m^B for $m = 1, 2, \dots, P$ and $e_m^f(n)$ and $e_m^b(n).$

where $P_m = P_m^f = P_m^b$ (see Section 7.4). If we substitute (9.2.43) into (9.2.42), we obtain

$$\mathcal{H}(\mathbf{x}) = M \log \pi + M + M \log r(0) + \sum_{m=1}^{M-1} (M-m) \log(1 - |k_m|^2) \quad (9.2.44)$$

which expresses the entropy in terms of $r(0)$ and the PACS k_m , $1 \leq m \leq M \leq \infty$ [recall that any parametric model can be specified by $r(0)$ and the PACS]. Suppose now that we are given the first $P+1$ values $r(0), r(1), \dots, r(P)$ of the autocorrelation sequence and we wish to find a model, by choosing the remaining values $r(l)$, $l > P$, so that the entropy is maximized. From (9.2.44), we see that the entropy is maximized if we choose $k_m = 0$ for $m > P$, that is, by modeling the process $x(n)$ by an AR(P) model. In conclusion, among all regular Gaussian processes with the same first $P+1$ autocorrelation values, the AR(P) process has the maximum entropy. Any other choices for k_m , $m > P$, that satisfy the condition $|k_m| < 1$ lead to a valid extension of the autocorrelation sequence. The “extended” values $r(l)$, $l > P$, can be obtained by using the inverse Levinson-Durbin or the inverse Schür algorithm (see Chapter 7). The relation between autoregressive modeling and the principle of maximum entropy, known as the *maximum entropy method*, was introduced by Burg (1967, 1975). We note that the above proof, given in Porat (1994), is different from the original proof provided by Burg (Burg 1975; Therrien 1992). An interesting discussion of various arguments in favor of and against the maximum entropy method can be found in Makhoul (1986).

9.2.4 Excitations with Line Spectra

When the excitation of a parametric model has a spectrum with lines at L frequencies ω_m , the spectrum of the output signal provides information about the frequency response of the model at these frequencies only. For simplicity, assume equidistant samples at frequencies $\omega_m = 2\pi m/L$, $0 \leq m \leq L-1$. Given a set of values $R_x(e^{j\omega_m}) = |X(e^{j\omega_m})|^2$, we wish to find an AP(P) model whose spectrum $\hat{R}_h(e^{j\omega})$ matches $R_x(\omega_m)$ at the given frequencies, by minimizing the criterion

$$\tilde{\mathcal{E}} = \frac{d_0}{L} \sum_{m=1}^L \frac{R_x(e^{j\omega_m})}{\hat{R}_h(e^{j\omega_m})} \quad (9.2.45)$$

which is the discrete version of (9.2.17) and d_0 is the gain of the model (see Section 4.2). The minimization of (9.2.45) with respect to the model parameters $\{a_k\}$ results in the Yule-Walker equations

$$\sum_{k=0}^P a_k^* \tilde{r}(i-k) = \begin{cases} \tilde{\epsilon} & i = 0 \\ 0 & 1 \leq i \leq P \end{cases} \quad (9.2.46)$$

where $\tilde{r}(l) = \frac{1}{L} \sum_{m=1}^L R_x(e^{j\omega_m}) e^{j\omega_m l}$ (9.2.47)

For continuous spectra, linear prediction uses the autocorrelation

$$r(l) = \frac{1}{2\pi} \int_{-\pi}^{\pi} R_x(e^{j\omega}) e^{j\omega l} d\omega \quad (9.2.48)$$

which is related to $\tilde{r}(l)$ by

$$\tilde{r}(l) = \sum_{m=-\infty}^{\infty} r(l - Lm) \quad (9.2.49)$$

that is, $\tilde{r}(l)$ is an aliased version of $r(l)$. We have seen that linear prediction equates the autocorrelation of the AP(P) model to the autocorrelation of the modeled signal for the first $P + 1$ lags. Hence, when we use linear prediction for a signal with line spectra, the autocorrelation of the all-pole model will be matched to $\tilde{r}(l) \neq r(l)$ and will always result in a model different from the original. Clearly, the correlation matching condition cannot compensate for the autocorrelation aliasing, which becomes more pronounced as L decreases. This phenomenon, which is severe for voiced sounds with high pitch, is illustrated in Problem 9.13. A method that provides better estimates, by minimizing a discrete version of the Itakura-Saito error measure, has been developed for both AP and PZ models by El-Jaroudi and Makhoul (1991, 1989).

9.3 ESTIMATION OF POLE-ZERO MODELS

The estimation of PZ(P, Q) model parameters for $Q \neq 0$ leads to a nonlinear LS optimization problem. As a result, a vast number of suboptimum methods, with reduced computational complexity, have been developed to avoid this problem. For example, some techniques estimate the AP(P) and AZ(Q) parameters separately. However, today the availability of high-speed computers has made exact least-squares the method of choice. Since the nonlinear LS optimization with respect to complex vectors and its conjugate is inherently difficult, and since this optimization does not provide any additional insight into the solution technique, we assume, in this section, that the quantities are real-valued. Furthermore, most of the real-world applications of pole-zero models almost always involve real-valued signals and systems. The extension to the complex-valued case is straightforward.

Consider the PZ(P, Q) model

$$x(n) = - \sum_{k=1}^P a_k x(n-k) + w(n) + \sum_{k=1}^Q d_k w(n-k) \quad (9.3.1)$$

where $w(n) \sim \text{WN}(0, \sigma_w^2)$. Using vector notation, we can express (9.3.1) as

$$x(n) = \mathbf{z}^T(n-1) \mathbf{c}_{\text{pz}} + w(n) \quad (9.3.2)$$

where $\mathbf{z}(n) \triangleq [-x(n) \cdots -x(n-P+1) \ w(n) \cdots w(n-Q+1)]^T$ (9.3.3)

and $\mathbf{c}_{\text{pz}} = [\mathbf{a}^T \ \mathbf{d}^T] = [a_1 \cdots a_P \ d_1 \cdots d_Q]^T$ (9.3.4)

Assume for a moment that the excitation $w(n)$ is known. Then we can predict $x(n)$ from past values, using the following linear predictor

$$\hat{x}(n) = \mathbf{z}^T(n-1)\mathbf{c} \quad (9.3.5)$$

where

$$\mathbf{c} = [\hat{a}_1 \cdots \hat{a}_P \hat{d}_1 \cdots \hat{d}_Q]^T \quad (9.3.6)$$

are the predictor parameters. The prediction error

$$e(n) = x(n) - \hat{x}(n) = x(n) - \mathbf{z}^T(n-1)\mathbf{c} \quad (9.3.7)$$

equals $w(n)$ if $\mathbf{c} = \mathbf{c}_{\text{pz}}$. Minimization of the total squared error

$$\mathcal{E}(\mathbf{c}) \triangleq \sum_{n=N_i}^{N_f} e^2(n) \quad (9.3.8)$$

leads to the following linear system of equations

$$\hat{\mathbf{R}}_z \mathbf{c} = \hat{\mathbf{r}}_z \quad (9.3.9)$$

where

$$\hat{\mathbf{R}}_z = \sum_{n=N_i}^{N_f} \mathbf{z}(n-1) \mathbf{z}^T(n-1) \quad (9.3.10)$$

and

$$\hat{\mathbf{r}}_z = \sum_{n=N_i}^{N_f} \mathbf{z}(n-1) x(n) \quad (9.3.11)$$

Usually, we use residual windowing, which implies that $N_i = \max(P, Q)$ and $N_f = N - 1$. Since the matrix $\hat{\mathbf{R}}_z$ is symmetric and positive semidefinite, we can solve (9.3.9) using LDL^H decomposition. Thus, if we know the excitation $w(n)$, the least-squares estimation of the PZ(P, Q) model parameters reduces to the solution of a linear system of equations. An estimate of the input variance is given by

$$\hat{\sigma}_w^2 = \frac{1}{N - \max(P, Q)} \sum_{n=\max(P, Q)}^{N-1} e^2(n) \quad (9.3.12)$$

This method, which is implemented by the function `pzls.m`, is known as the *equation-error method* and can be used to identify a system from input-output data (Ljung 1987) (see Problem 9.14).

9.3.2 Unknown Excitation

In most applications, the excitation $w(n)$ is never known. However, we can obtain a good estimate of $x(n)$ by replacing $w(n)$ by $e(n)$ in (9.3.3). This makes a natural choice if the model used to obtain $e(n)$ is reasonably accurate. The prediction error is then given by

$$e(n) = x(n) - \hat{x}(n) = x(n) - \hat{\mathbf{z}}^T(n-1)\mathbf{c} \quad (9.3.13)$$

where $\hat{\mathbf{z}}(n) \triangleq [-x(n) \cdots -x(n-P+1) e(n) \cdots e(n-Q+1)]^T \quad (9.3.14)$

If we write (9.3.13) explicitly

$$e(n) = - \sum_{k=1}^Q \hat{a}_k e(n-k) + x(n) + \sum_{k=1}^P \hat{d}_k x(n-k) \quad (9.3.15)$$

we see that the prediction error is obtained by exciting the inverse model with the signal $x(n)$. Hence, the inverse model has to be stable. To satisfy this condition, we require the estimated model to be minimum-phase.

The recursive computation of $e(n)$ by (9.3.15) makes the prediction error a nonlinear function of the model parameters. To illustrate this, consider the prediction error for a first-order model, that is, for $P = Q = 1$

$$e(n) = x(n) + \hat{a}_1 x(n-1) - \hat{d}_1 e(n-1)$$

Assuming $e(0) = 0$, we have for $n = 1, 2, 3$

$$\begin{aligned} e(1) &= x(1) + \hat{a}_1 x(0) \\ e(2) &= x(2) + \hat{a}_1 x(1) - \hat{d}_1 e(1) \\ &= x(2) + (\hat{a}_1 - \hat{d}_1)x(1) - \hat{a}_1 \hat{d}_1 x(0) \\ e(3) &= x(3) + \hat{a}_1 x(2) - \hat{d}_1 e(2) \\ &= x(3) + (\hat{a}_1 - \hat{d}_1)x(2) - (\hat{a}_1 - \hat{d}_1)\hat{d}_1 x(1) + \hat{a}_1 \hat{d}_1^2 x(0) \end{aligned}$$

which shows that $e(n)$ is a nonlinear function of the model parameters if $Q \neq 0$. Thus, the total squared error

$$\mathcal{E}(\mathbf{c}) = \sum_{n=N_i}^{N_f} e^2(n) \quad (9.3.16)$$

expressed in terms of the signal values $x(0), x(1), \dots, x(N-1)$, is a nonquadratic function of the model parameters. Sometimes, $\mathcal{E}(\mathbf{c})$ has several local minima. The model parameters can be obtained by minimizing the total square error using nonlinear optimization techniques.

9.3.3 Nonlinear Least-Squares Optimization

We next outline such a technique that is based on the method of Gauss–Newton. More details can be found in Scales (1985); Luenberger (1984); and Gill, Murray, and Wright (1981). To this end, we expand the function $\mathcal{E}(\mathbf{c})$ as a Taylor series

$$\mathcal{E}(\mathbf{c}_0 + \Delta\mathbf{c}) = \mathcal{E}(\mathbf{c}_0) + (\Delta\mathbf{c})^T \nabla \mathcal{E}(\mathbf{c}_0) + \frac{1}{2}(\Delta\mathbf{c})^T [\nabla^2 \mathcal{E}(\mathbf{c}_0)](\Delta\mathbf{c}) + \dots \quad (9.3.17)$$

where $\nabla \mathcal{E}(\mathbf{c}) \triangleq \left[\frac{\partial \mathcal{E}}{\partial c_1} \frac{\partial \mathcal{E}}{\partial c_2} \dots \frac{\partial \mathcal{E}}{\partial c_{p+q}} \right]^T$ (9.3.18)

is the vector of the first partial derivatives or *gradient vector* and $\nabla^2 \mathcal{E}(\mathbf{c})$, whose (i, j) th element is $\partial^2 \mathcal{E} / (\partial c_i \partial c_j)$, is the (symmetric) matrix of second partial derivatives (*Hessian matrix*).

The Taylor expansion of a quadratic function has only the first three terms. Indeed, for the known excitation case we have

$$\nabla \mathcal{E}(\mathbf{c}) = 2 \sum_{n=N_i}^{N_f} \mathbf{z}(n-1) e(n) = 2(\mathbf{r}_z - \hat{\mathbf{R}}_z \mathbf{c}) \quad (9.3.19)$$

and $\nabla^2 \mathcal{E}(\mathbf{c}) = 2 \sum_{n=N_i}^{N_f} \mathbf{z}(n-1) \mathbf{z}^T(n-1) = 2\hat{\mathbf{R}}_z$ (9.3.20)

Higher-order terms are zero, and if \mathbf{c}_0 is the minimum, then $\nabla \mathcal{E}(\mathbf{c}_0) = \mathbf{0}$. In this case, (9.3.17) becomes

$$\mathcal{E}(\mathbf{c}_0 + \Delta\mathbf{c}) = \mathcal{E}(\mathbf{c}_0) + (\Delta\mathbf{c})^T \hat{\mathbf{R}}_z (\Delta\mathbf{c})$$

which shows that if $\hat{\mathbf{R}}_z$ is positive definite, that is, $(\Delta \mathbf{c})^T \hat{\mathbf{R}}_z (\Delta \mathbf{c}) \geq 0$, then any deviation from the minimum results in an increase in the total squared error.

This relationship holds approximately for nonquadratic functions, as long as \mathbf{c}_0 is close to a minimum. Thus, if we are at a point \mathbf{c}_i with total squared error $\mathcal{E}(\mathbf{c}_i)$, we can move to a point \mathbf{c}_{i+1} with total squared error $\mathcal{E}(\mathbf{c}_{i+1}) \leq \mathcal{E}(\mathbf{c}_i)$ by moving in the direction of $-\nabla \mathcal{E}(\mathbf{c}_i)$. This suggests the following iterative procedure

$$\mathbf{c}_{i+1} = \mathbf{c}_i - \mu_i \mathbf{G}_i \nabla \mathcal{E}(\mathbf{c}_i) \quad (9.3.21)$$

where the positive scalar μ_i controls the length of the descent and matrix \mathbf{G}_i modifies the direction of the descent, as is specified by the gradient vector. Various choices for these quantities lead to various optimization algorithms. For quadratic functions, choosing $\mathbf{c}_0 = \mathbf{0}$, $\mu_0 = 1$, and $\mathbf{G}_0 = (2\hat{\mathbf{R}}_z)^{-1}$ (inverse of the Hessian matrix) gives $\mathbf{c}_1 = \hat{\mathbf{R}}_z^{-1} \hat{\mathbf{r}}_z$; that is, we find the unique minimum in one step. This provides the motivation for modifying the direction of the gradient using the inverse of the Hessian matrix, even for nonquadratic functions. This choice is justified as long as we are close to a minimum.

Using (9.3.13), we compute the Hessian as follows

$$\nabla^2 \mathcal{E}(\mathbf{c}) = \nabla[\nabla \mathcal{E}(\mathbf{c})]^T = 2 \sum_{n=N_i}^{N_f} \boldsymbol{\psi}(n) \boldsymbol{\psi}^T(n) + 2 \sum_{n=N_i}^{N_f} [\nabla \boldsymbol{\psi}^T(n)] e(n) \quad (9.3.22)$$

$$\text{where } \boldsymbol{\psi}(n) \triangleq \nabla e(n) = \left[\frac{\partial e(n)}{\partial \hat{a}_1} \dots \frac{\partial e(n)}{\partial \hat{a}_P} \frac{\partial e(n)}{\partial \hat{d}_1} \dots \frac{\partial e(n)}{\partial \hat{d}_Q} \right]^T \quad (9.3.23)$$

We usually approximate the Hessian with the first summation in (9.3.22), that is,

$$\mathbf{H} = 2 \sum_{n=N_i}^{N_f} \boldsymbol{\psi}(n) \boldsymbol{\psi}^T(n) \quad (9.3.24)$$

Similarly, the gradient is given by

$$\nabla \mathcal{E}(\mathbf{c}) \triangleq \mathbf{v} = 2 \sum_{n=N_i}^{N_f} \boldsymbol{\psi}(n) e(n) \quad (9.3.25)$$

If we set $\mathbf{G} = \mathbf{H}^{-1}$, the direction vector $\mathbf{g} = \mathbf{G}\mathbf{v} = \mathbf{H}^{-1}\mathbf{v}$ can be obtained by solving the following linear system of equations:

$$\mathbf{H}\mathbf{g} = \mathbf{v} \quad (9.3.26)$$

Clearly, the factor 2 in the definitions of \mathbf{H} and \mathbf{v} does not affect the solution \mathbf{g} , and can be dropped. Although the matrix \mathbf{H} is guaranteed by (9.3.24) to be positive semidefinite, in practice it may be singular or close to singular. To avoid such problems in solving (9.3.26), we *regularize* the matrix by adding a small positive constant δ to its diagonal; that is, we approximate the Hessian by $\mathbf{H} + \delta \mathbf{I}$, where \mathbf{I} is the identity matrix. This approach is known as the *Levenberg-Marquardt regularization* (Dennis and Schnabel 1983; Ljung 1987).

We next compute the gradient $\boldsymbol{\psi}(n) = \nabla e(n)$, using (9.3.23) and (9.3.15). Indeed, we have

$$\frac{\partial e(n)}{\partial \hat{a}_j} = x(n-j) - \sum_{k=1}^Q \hat{d}_k \frac{\partial e(n-k)}{\partial \hat{a}_j} \quad j = 1, 2, \dots, P \quad (9.3.27)$$

$$\text{and } \frac{\partial e(n)}{\partial \hat{d}_j} = -e(n-j) - \sum_{k=1}^Q \hat{d}_k \frac{\partial e(n-k)}{\partial \hat{d}_j} \quad j = 1, 2, \dots, Q \quad (9.3.28)$$

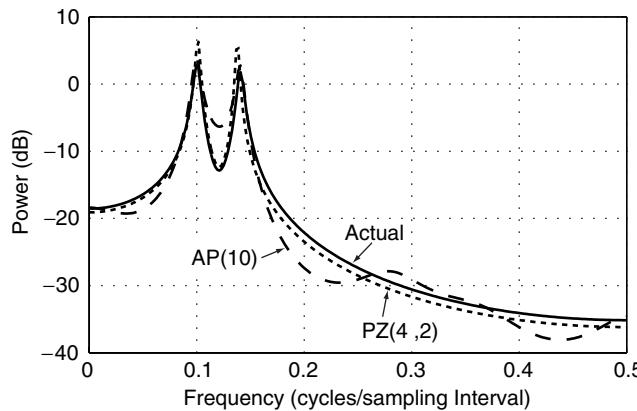
**FIGURE 9.13**

Illustration of the capability of a PZ(4, 2) and AP(10) model to estimate the PSD of an ARMA(4, 2) process from a 300-sample segment.

Thus, the components of the gradient vector are obtained by driving the all-pole filter

$$H_\psi(z) = \frac{1}{D(z)} = \frac{1}{1 + \sum_{k=1}^Q \hat{d}_k z^{-k}} \quad (9.3.29)$$

with the signals $x(n)$ and $-e(n)$, respectively. This filter is stable if the estimated model is minimum-phase.

The above development leads to the following iterative algorithm, implemented in the MATLAB function `armals.m`, which computes the parameters of a PZ(P, Q) model from the data $x(0), x(1), \dots, x(N-1)$ by minimizing the LS error. The LS pole-zero modeling algorithm consists of the following steps:

1. Fit an AP($P + Q$) model to the data, using the no-windowing LS method, and compute the prediction error $e(n)$ (see Section 9.2).
2. Fit a PZ(P, Q) model to the data $\{x(n), e(n), 0 \leq n \leq N-1\}$, using the known excitation method. Convert the model to minimum-phase, if necessary. Use Equations (9.3.9) to (9.3.11).
3. Start the iterative minimization procedure, which involves the following steps:
 - a. Compute the gradient $\psi(n)$, using (9.3.27) and (9.3.28).
 - b. Compute the Hessian \mathbf{H} and the gradient \mathbf{v} , using (9.3.24) and (9.3.25).
 - c. Solve (9.3.26) to compute the search vector \mathbf{g} . If necessary, use the Levenberg-Marquardt regularization technique.
 - d. For $\mu = 1, \frac{1}{2}, \dots, \frac{1}{10}$, compute $\mathbf{c} \leftarrow \mathbf{c} + \mu \mathbf{g}$, convert the model to minimum-phase, if necessary, and compute the corresponding value of $\mathcal{E}(\mathbf{c})$. Choose the value of \mathbf{c} that gives the smaller total squared error.[†]
 - e. Stop if $\mathcal{E}(\mathbf{c})$ does not change significantly or if a certain number of iterations have been exceeded.
4. Compute the estimate of the input variance, using (9.3.15) and (9.3.12).

The application of the LS PZ(P, Q) model estimation algorithm is illustrated in Figure 9.13, which shows the actual PSD of a PZ(4, 2) model and the estimated PSDs, using an LS PZ(4,

[†]This approach was suggested in Ljung (1987), problem 10S.1.

2) and an AP(10) model fitted to a 300-sample segment of the output process. We notice that, in contrast to the PZ model, the AP model does not provide a good match at the spectral zero. More details are provided in Problem 9.15.

9.4 APPLICATIONS

Pole-zero modeling has many applications in such fields as spectral estimation, speech processing, geophysics, biomedical signal processing, and general time series analysis and forecasting (Marple 1987; Kay 1988; Robinson and Treitel 1980; Box, Jenkins, and Reinsel 1994). In this section, we discuss the application of pole-zero models to spectral estimation and speech processing.

9.4.1 Spectral Estimation

After we have estimated the parameters of a PZ model, we can compute the PSD of the analyzed process by

$$\hat{R}(e^{j\omega}) = \hat{\sigma}_w^2 \frac{\left| 1 + \sum_{k=1}^Q \hat{d}_k e^{j\omega k} \right|^2}{\left| 1 + \sum_{k=1}^P \hat{a}_k e^{j\omega k} \right|^2} \quad (9.4.1)$$

In practice, we mainly use AP models because (1) the all-zero PSD estimator is essentially identical to the Blackman-Tukey one (see Problem 9.16) and (2) the application of pole-zero PSD estimators is limited by computational and other practical difficulties. Also, any continuous PSD can be approximated arbitrarily well by the PSD of an AP(P) model if P is chosen large enough (Anderson 1971). However, in practice, the value of P is limited by the amount of available data (usually $P < N/3$). The statistical properties of all-pole PSD estimators are difficult to obtain; however, it has been shown that the estimator is consistent only if the analyzed process is AR(P_0) with $P_0 \leq P$. Furthermore, the quality of the estimator degrades if the process is contaminated by noise. More details about pole-zero PSD estimation can be found in Kay (1988), Porat (1994), and Percival and Walden (1993).

The performance of all-pole PSD estimators depends on the method used to estimate the model parameters, the order of the model, and the presence of noise. The effect of model mismatch is shown in Figure 9.13 and is further investigated in Problem 9.17. Order selection in all-pole PSD estimation is absolutely critical: If P is too large, the obtained PSD exhibits spurious peaks; if P is too small, the structure of the PSD is smoothed over. The increased resolution of the parametric techniques, compared to the nonparametric PSD estimation methods, is basically the result of imposing structure on the data (i.e., a model). The model makes possible the extrapolation of the ACS, which in turns leads to better resolution. However, if the adopted model is inaccurate, that is, if it does not match the data, then the “gained” resolution reflects the model and not the data! As a result, despite their popularity and their “success” with simulated signals, the application of parametric PSD estimation techniques to actual experimental data is rather limited.

Figure 9.14 shows the results of a Monte Carlo simulation of various all-pole PSD estimation techniques. We see that, except for the windowing approach that results in a significant loss of resolution, all other techniques have similar performance. However, we should mention that the forward/backward LS all-pole modeling method is considered to provide the best results (Marple 1987).

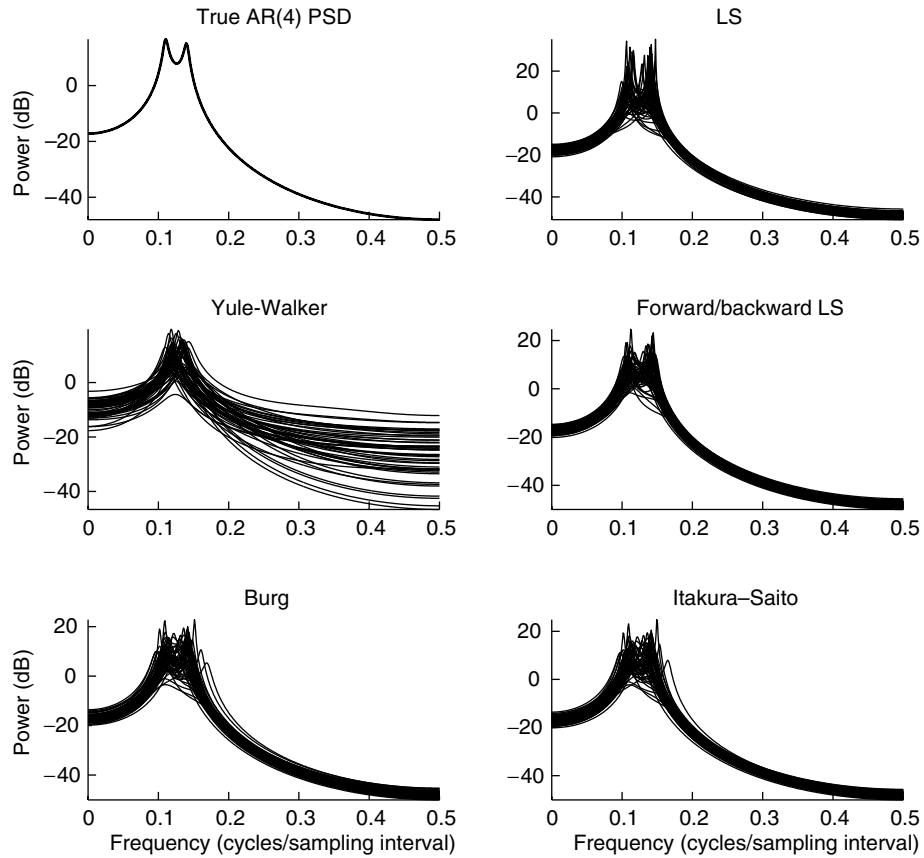


FIGURE 9.14

Monte Carlo simulation for the comparison of all-pole PSD estimation techniques, using 50 realizations of a 50-sample segment from an AR(4) process using fourth-order AP models.

In practice, it is our experience that the best way to estimate the PSD of an actual signal is to combine parametric *prewhitening* with nonparametric PSD estimation methods. The process is illustrated in Figure 9.15 and involves the following steps:

1. Fit an AP(P) model to the data using the forward LS, forward/backward LS, or Burg's method with no windowing.
2. Compute the residual (prediction error)

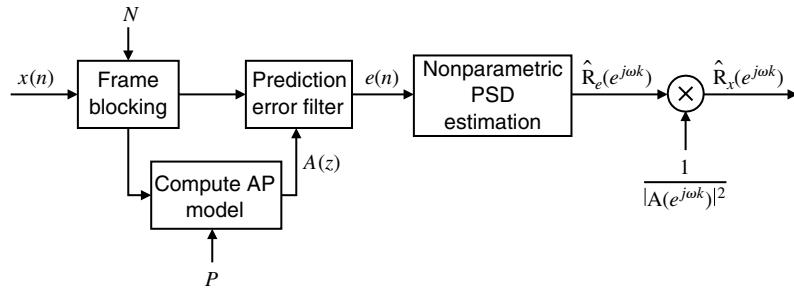
$$e(n) = x(n) + \sum_{k=1}^P a_k^* x(n-k) \quad P \leq n \leq N-1 \quad (9.4.2)$$

and then compute and plot its ACS, PACS, and cumulative periodogram (see Figure 9.2) to see if it is reasonably white. The goal is not to completely whiten the residual but to reduce its spectral dynamic range, that is, to increase its spectral flatness to *avoid* spectral leakage.

3. Compute the PSD $\hat{R}_e(e^{j\omega_k})$, using one of the nonparametric techniques discussed in Chapter 5.
4. Compute the PSD of $x(n)$ by

$$\hat{R}_x(e^{j\omega_k}) = \frac{\hat{R}_e(e^{j\omega_k})}{|A(e^{j\omega_k})|^2} \quad (9.4.3)$$

that is, by applying postcoloring to “undo” the prewhitening.

**FIGURE 9.15**

Block diagram of nonparametric PSD estimation using linear prediction prewhitening.

The main goal of AP modeling here is to reduce the spectral dynamic range to avoid leakage. In other words, we need a good linear predictor regardless of whether the process is true AR(P). Therefore, very accurate order selection and model fit are not critical, because all spectral structure not captured by the model is still in the residuals. Needless to say, if the periodogram of $x(n)$ has a small dynamic range, we do not need prewhitening. Another interesting application of prewhitening is for the detection of outliers in practical data (Martin and Thomson 1982).

EXAMPLE 9.4.1. To illustrate the effectiveness of the above prewhitening and postcoloring method, consider the AR(4) process $x(n)$ used in Example 9.2.3. This process has a large dynamic range, and hence the nonparametric methods such as Welch's periodogram averaging method will suffer from leakage problems. Using the system function of the model

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 - 2.7607z^{-1} + 3.8106z^{-2} - 2.6535z^{-3} + 0.9238z^{-4}}$$

and WGN (0, 1) input sequence, we generated 256 samples of $x(n)$. These samples were then used to obtain the all-pole LS predictor coefficients using the `arwin` function. The spectrum $|A(e^{j\omega})|^{-2}$ corresponding to this estimated model is shown in Figure 9.16 as a dashed curve. The signal samples were prewhitened using the model to obtain the residuals $e(n)$. The nonparametric PSD estimate $\hat{R}_e(e^{j\omega})$ of $e(n)$ was computed by using Welch's method with $L = 64$ and 50 percent overlap. Finally, $\hat{R}_e(e^{j\omega})$ was postcolored using the spectrum $|A(e^{j\omega})|^{-2}$ to obtain $\hat{R}_x(e^{j\omega})$, which is shown in Figure 9.16 as a solid line. For comparison purposes, the Welch

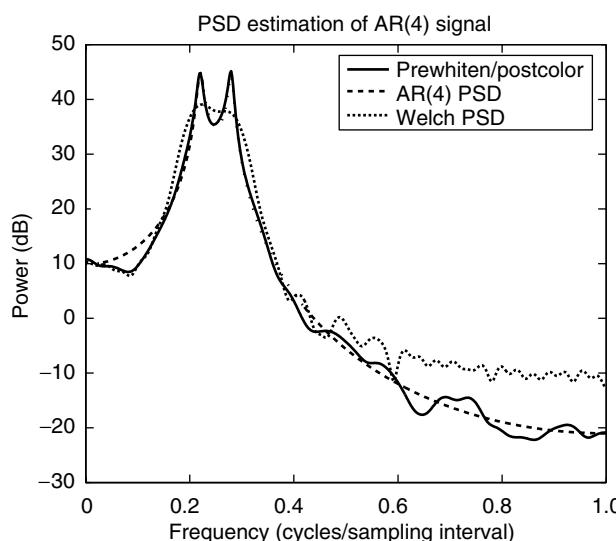


FIGURE 9.16
Spectral estimation of AR(4) process using prewhitening and postcoloring method in Example 9.4.1.

PSD estimate of $x(n)$ is also shown as a dotted line. As expected, the nonparametric estimate does not resolve the two peaks in the true spectrum and suffers from leakage at high frequencies. However, the combined nonparametric and parametric estimate resolves two peaks with ease and also follows the true spectrum quite well. Therefore, the use of the parametric method as a preprocessor is highly recommended especially in large-dynamic-range situations.

9.4.2 Speech Modeling

All-pole modeling using LS linear prediction is widely employed in speech processing applications because (1) it provides a good approximation to the vocal tract for voiced sounds and adequate approximation for unvoiced and transient sounds, (2) it results in a good separation between source (fine spectral structure) and vocal tract (spectral envelop), and (3) it is analytically tractable and leads to efficient software and hardware implementations.

Figure 9.17 shows a typical AP modeling system, also known as the *linear predictive coding (LPC)* processor, that is used in speech synthesis, coding, and recognition applications. The processor operates in a block processing mode; that is, it processes a frame of N samples and computes a vector of model parameters using the following basic steps:

1. *Preemphasis*. The digitized speech signal is filtered by the high-pass filter

$$H_1(z) = 1 - \alpha z^{-1} \quad 0.9 \leq \alpha \leq 1 \quad (9.4.4)$$

to reduce the dynamic range of the spectrum, that is, to flatten the spectral envelope, and make subsequent processing less sensitive to numerical problems (Makhoul 1975a). Usually $\alpha = 0.95$, which results in about a 32 dB boost in the spectrum at $\omega = \pi$ over that at $\omega = 0$. The preemphasizer can be made adaptive by setting $\alpha = \rho(1)$, where $\rho(l)$ is the normalized autocorrelation of the frame, which corresponds to a first-order optimum prediction error filter.

2. *Frame blocking*. Here the preemphasized signal is blocked into frames of N samples with successive frames overlapping by $N_0 \simeq N/3$ samples. In speech recognition $N = 300$ with a sampling rate $F_s = 6.67$ Hz, which corresponds to 45-ms frames overlapping by 15 ms.
3. *Windowing*. Each frame is multiplied by an N -sample window (usually Hamming) to smooth the discontinuities at the beginning and the end of the frame.
4. *Autocorrelation computation*. Here the LPC processor computes the first $P + 1$ values of the autocorrelation sequence. Usually, $P = 8$ in speech recognition and $P = 12$ in speech coding applications. The value of $r(0)$ provides the energy of the frame, which is useful for speech detection.
5. *LPC analysis*. In this step the processor uses the $P + 1$ autocorrelations to compute an LPC parameter set for each speech frame. Depending on the required parameters, we

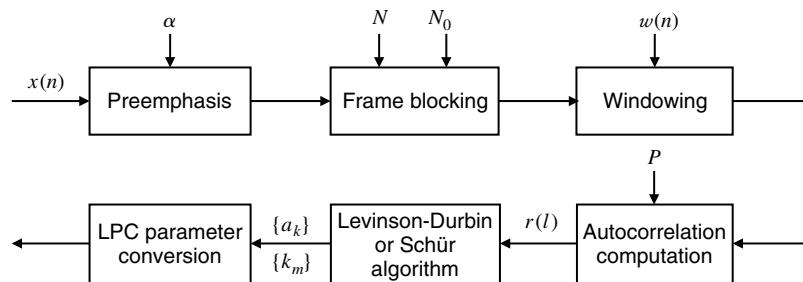


FIGURE 9.17

Block diagram of an AP modeling processor for speech coding and recognition.

can use the algorithm of Levinson-Durbin or the algorithm of Schür. The most widely used parameters are

$$\begin{aligned}
 a_m &= a_m^{(P)} && \text{LPC coefficients} \\
 k_m & && \text{PACS} \\
 g_m &= \frac{1}{2} \log \frac{1 - k_m}{1 + k_m} = \tanh^{-1} k_m && \text{log area ratio coefficients} \\
 c(m) & && \text{cepstral coefficients} \\
 \omega_m & && \text{line spectrum pairs}
 \end{aligned}$$

where $1 \leq m \leq P$, except for the cepstrum, which is computed up to about $3P/2$. The line spectrum pair parameters, which are pole angles of the singular filters, were discussed in Section 2.5.8, and their application to speech processing is considered in Furui (1989).

The log area ratio and the line spectrum pair coefficients have good quantization properties and are used for speech coding (Rabiner and Schafer 1978; Furui 1989); the cepstral coefficients provide an excellent discriminant for speech and speaker recognition applications (Rabiner and Juang 1993; Mammone et al. 1996). AP models are extensively used for the modeling of speech sounds. However, the AP model does not provide an accurate description of the speech spectral envelope when the speech production process resembles a PZ system (Atal and Schroeder 1978). This can happen when (1) the nasal tract is coupled to the main vocal tract through the velar opening, for example, during the generation of nasals and nasalized sounds, (2) the source of excitation is not at the glottis but is in the interior of the vocal tract (Flanagan 1972), and (3) the transmission or recording channel has zeros in its response. Although a zero can be approximated with arbitrary precision by a number of poles, this approximation is usually inefficient and leads to spectral distortion and other problems. These problems can be avoided by using pole-zero modeling, as illustrated in the following example. More details about pole-zero speech modeling can be found in Atal and Schroeder (1978).

Figure 9.18(a) shows a Hamming window segment from an artificial nasal speech signal sampled at $F_s = 10$ kHz. According to acoustic theory, such sounds require both poles and zeros in the vocal tract system function. Before the fitting of the model, the data are passed through a preemphasis filter with $\alpha = 0.95$. Figure 9.18(b) shows the periodogram of the speech segment, the spectrum of an AP(16) model using data windowing, and the spectrum of a PZ(12, 6) model using the least-squares algorithm described in Section 9.3.3 (see Problem 9.18 for details). We see that the pole-zero model matches zeros (“valleys”) in the periodogram of the data better than other models do.

9.5 MINIMUM-VARIANCE SPECTRUM ESTIMATION

Spectral estimation methods were discussed in Chapter 5 that are based on the discrete Fourier transform (DFT) and are data-independent; that is, the processing does not depend on the actual values of the samples to be analyzed. Window functions can be employed to cut down on sidelobe leakage, at the expense of resolution. These methods have, as a rule of thumb, an approximate resolution of $\Delta f \approx 1/N$ cycles per sampling interval. Thus, for all these methods, resolution performance is limited by the number of available data samples N . This problem is only accentuated when the data must be subdivided into segments to reduce the variance of the spectrum estimate by averaging periodograms. The effective resolution is then on the order of $1/M$, where M is the window length of the segments. For many applications the amount of data available for spectrum estimation may be limited

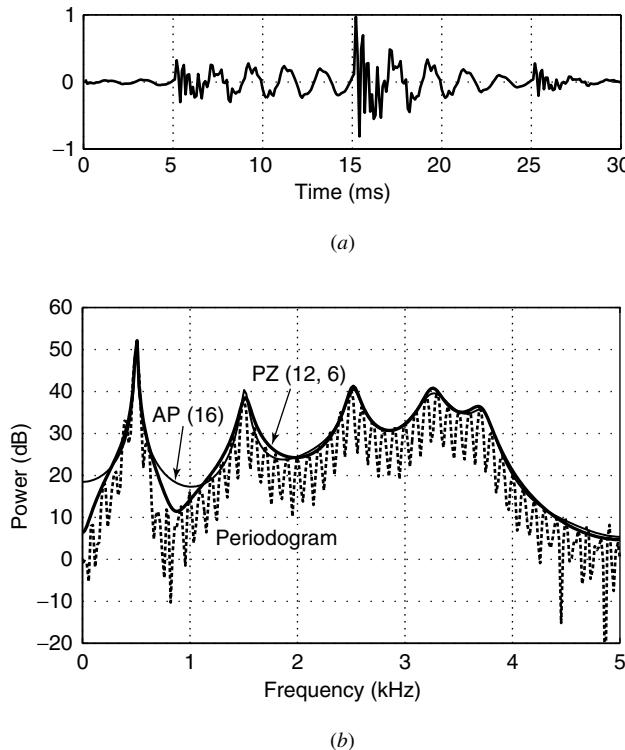


FIGURE 9.18
(a) Speech segment and
(b) periodogram, spectrum of a data windowing-based AP(16)
model, and spectrum of a
residual windowing-based
PZ(12, 6) model.

either because the signal may only be considered stationary over limited intervals of time or may only be collected over a short finite interval.

Many times, it may be necessary to resolve spectral peaks that are spaced closer than the $1/M$ limit imposed by the amount of data available. All the DFT-based methods use a predetermined, fixed processing that is independent of the values of the data. However, there are methods, termed *data-adaptive spectrum estimation* (Lacoss 1971), that can exploit actual characteristics of the data to offer significant improvements over the data-independent, DFT-based methods, particularly in the case of limited data samples. *Minimum-variance spectral estimation* is one such technique (Capon 1969). Like the methods from Chapter 5, the minimum-variance spectral estimator is nonparametric; that is, it does not assume an underlying model for the data. However, the spectral estimator adapts itself to the characteristics of the data in order to reject as much out-of-band energy, that is, leakage, as possible. In addition, minimum-variance spectral estimation provides improved resolution—better than the $\Delta f \approx 1/N$ associated with the DFT-based methods. As a result, the minimum-variance method is commonly referred to as a *high-resolution spectral estimator*. Note that model-based data-adaptive methods, such as the LS all-pole method, also have high resolving capabilities when the model adequately represents the data.

Theory

We derive the minimum-variance spectral estimator by using a filter bank structure in which each of the filters adapts its response to the data. Recall that the goal of a power spectrum estimator is to determine the power content of a signal at a certain frequency. To this end, we would like to measure $R(e^{j2\pi f})$ at the frequency of interest only and not have our estimate influenced by energy present at other frequencies. Thus, we might interpret spectral estimation as a methodology in determining the ideal, frequency-selective filter for each frequency. Recall the filter bank interpretation of a power spectral estimator from Chapter 5. This ideal filter for f_k should pass energy within its bandwidth Δf but reject all

$$|H_k(e^{j2\pi f})|^2 = \begin{cases} \frac{1}{\Delta f} & |f - f_k| \leq \frac{\Delta f}{2} \\ 0 & \text{otherwise} \end{cases} \quad (9.5.1)$$

where the factor $\Delta f \sim 1/M$ accounts for the filter bandwidth.[†] Therefore, the filter does not impart a gain across the bandwidth of the filter, and the output of the filter is a measure of power in the frequency band around f_k . However, since such an ideal filter does not exist in practice, we need to design one that passes energy at the center frequency while rejecting as much out-of-band energy as possible.

A filter bank-based spectral estimator should have filters at all frequencies of interest. The filters have equal spacing in frequency, spanning the fundamental frequency range $-\frac{1}{2} \leq f < \frac{1}{2}$. Let us denote the total number of frequencies by K and the center frequency of the k th filter as

$$f_k = \frac{k-1}{K} - \frac{1}{2} \quad (9.5.2)$$

for $k = 1, 2, \dots, K$. The output of the k th filter is the convolution of the signal $x(n)$ with the impulse response of the filter $h_k(n)$, which can also be expressed in vector form as

$$y_k(n) = h_k(n) * x(n) = \sum_{m=0}^{M-1} h_k(m)x(n-m) = \mathbf{c}_k^H \mathbf{x}(n) \quad (9.5.3)$$

where

$$\mathbf{c}_k = [h_k^*(0) \ h_k^*(1) \ \cdots \ h_k^*(M-1)]^T \quad (9.5.4)$$

is the impulse response of the k th filter, and

$$\mathbf{x}(n) = [x(n) \ x(n-1) \ \cdots \ x(n-M+1)]^T \quad (9.5.5)$$

is the input data vector. In addition, we define the *frequency vector* $\mathbf{v}(f)$ as a vector of complex exponentials at frequency f within the time-window vector from (9.5.5)

$$\mathbf{v}(f) = [1 \ e^{-j2\pi f} \ \cdots \ e^{-j2\pi f(M-1)}]^T \quad (9.5.6)$$

When the frequency vector $\mathbf{v}(f)$ is chosen as the filter weight vector in (9.5.4), then the filter will pass signals at frequency f . Note that if we have $\mathbf{c}_k = \mathbf{v}(f_k)$, then the resulting filter bank performs a DFT since $\mathbf{v}(f)$ is a column vector in the DFT matrix. Thus, all the DFT-based methods, when interpreted using a filter bank structure, use a form of $\mathbf{v}(f)$, possibly with a window, as filter weights. See Chapter 5 for the filter bank interpretation of the DFT.

The output $y_k(n)$ of the k th filter should ideally give an estimate of the power spectrum at f_k . The output power of the k th filter is

$$E\{|y_k(n)|^2\} = \mathbf{c}_k^H \mathbf{R}_x \mathbf{c}_k \quad (9.5.7)$$

where $\mathbf{R}_x = E\{\mathbf{x}(n)\mathbf{x}^H(n)\}$ is the correlation matrix of the input data vector from (9.5.5). Since the ideal filter response from (9.5.1) cannot be realized, we instead constrain our filter \mathbf{c}_k to have a response at the center frequency f_k of

$$H_k(f_k) = |\mathbf{c}_k^H \mathbf{v}(f_k)|^2 = M \quad (9.5.8)$$

This constraint ensures that the center frequency of our bandpass filter is at the frequency f_k . To eliminate as much out-of-band energy as possible, the filter is formulated as the filter that minimizes its output power subject to the center frequency constraint in (9.5.8), that is,

$$\min \mathbf{c}_k^H \mathbf{R}_x \mathbf{c}_k \quad \text{subject to} \quad \mathbf{c}_k^H \mathbf{v}(f_k) = \sqrt{M} \quad (9.5.9)$$

[†]A similar normalization was performed for all the DFT-based methods. Note that the same is not true of a sinusoidal signal that has zero bandwidth. See Example 9.5.2.

This constraint requires the filter to have a response of \sqrt{M} to a frequency vector from (9.5.6) at the frequency of interest while rejecting (minimizing) energy from all other frequencies. The solution to this constrained optimization problem can be found via Lagrange multipliers (see Appendix B and Problem 9.22) to be

$$\mathbf{c}_k = \frac{\sqrt{M} \mathbf{R}_x^{-1} \mathbf{v}(f_k)}{\mathbf{v}^H(f_k) \mathbf{R}_x^{-1} \mathbf{v}(f_k)} \quad (9.5.10)$$

By substituting (9.5.10) into (9.5.3), we obtain the output of the k th filter. The power of this signal, from (9.5.7), is the minimum-variance spectral estimate

$$\hat{R}_M^{(\text{mv})}(e^{j2\pi f_k}) = E\{|y_k(n)|^2\} = \frac{M}{\mathbf{v}^H(f_k) \mathbf{R}_x^{-1} \mathbf{v}(f_k)} \quad (9.5.11)$$

where the subscript M denotes the length of the data vector used to compute the spectral estimate. Note that in order to compute the minimum-variance spectral estimate, we need to find the inverse of the correlation matrix, which is a Toeplitz matrix since $x(n)$ is stationary. Efficient techniques for computing the inverse of a Toeplitz matrix were discussed in Chapter 7.

Implementation

A spectral estimator attempts to determine the power of a random process as a function of frequency based on a finite set of observations. Since the minimum-variance estimate of the spectrum involves the correlation matrix of the input data vector, which is unknown in practice, the correlation matrix must be estimated from the data. An estimate of the $M \times M$ correlation matrix, known as the *sample correlation matrix*, is given by[†]

$$\hat{\mathbf{R}}_x = \frac{1}{N - M + 1} \mathbf{X}^H \mathbf{X} \quad (9.5.12)$$

where $\mathbf{X}^H = [\mathbf{x}(M) \quad \mathbf{x}(M+1) \quad \cdots \quad \mathbf{x}(N)]$

$$= \begin{bmatrix} x(M) & x(M+1) & \cdots & x(N) \\ x(M-1) & x(M) & \cdots & x(N-1) \\ \vdots & \vdots & \ddots & \vdots \\ x(1) & x(2) & \cdots & x(N-M+1) \end{bmatrix} \quad (9.5.13)$$

is the data matrix formed from $x(n)$ for $0 \leq n \leq N-1$. Any of the other methods of forming a data matrix discussed in Chapter 8 can also be employed. Note that the data matrix in (9.5.13) does not produce a Toeplitz matrix $\hat{\mathbf{R}}_x$ in (9.5.12), though other methods from Chapter 8 will produce a Toeplitz sample correlation matrix.

An estimate of the spectrum based on the sample correlation matrix is found by substituting $\hat{\mathbf{R}}_x$ for the true correlation matrix \mathbf{R}_x in (9.5.11). Note that, in practice, the sample correlation matrix is not actually computed. The form of the sample correlation matrix resembles the product of the data matrices in the least-squares (LS) problem that is addressed in Chapter 8. Therefore, we might compute the upper triangular factor of the data matrix \mathbf{X} by using one of the techniques discussed in Chapter 8, such as a QR factorization. Indeed, if we compute the QR factorization made up of the orthonormal matrix \mathbf{Q}_x and the upper triangular factor \mathbf{R}_x

$$\mathbf{X} = \mathbf{Q}_x \mathbf{R}_x \quad (9.5.14)$$

then the minimum-variance spectrum estimator based on the sample correlation matrix is

$$\hat{R}_M^{(\text{scmv})}(e^{j2\pi f_k}) = \frac{1}{N - M + 1} \frac{M}{|\mathbf{v}^H(f_k) \mathbf{R}_x^{-H}|^2} \quad (9.5.15)$$

[†]We have normalized by $N - M + 1$, the number of realizations of the time-window vector $\mathbf{x}(n)$ in the data matrix \mathbf{X} . This normalization is necessary so that the output of the filter bank corresponds to an estimate of power.

Note that the conjugation of the upper triangular matrix comes about through the formulation of the data matrix in (9.5.13).

We have not addressed the issue of choosing the filter length M . Ideally, M is chosen to be as large as possible in order to maximize the rejection of out-of-band energy. However, from a practical point of view, we must place a limit on the filter length. As the filter length increases, the size of the data matrix grows, which increases the amount of computation necessary. In addition, since we are inherently estimating the correlation matrix, reducing the variance of this estimator requires averaging over a set of realizations of the input data vector $\mathbf{x}(n)$. Thus, for a fixed data record size of N , we must balance the length of the time window M against the number of realizations of the input data vector $N - M + 1$.

As we will demonstrate in the following example, the minimum-variance spectrum estimator provides a means of achieving high resolution, certainly better than the $\Delta f \sim 1/M$ limit of the DFT-based methods. High resolving capability essentially means that the minimum-variance spectrum estimator can better distinguish complex exponential signals closely spaced in frequency. This topic is explored further in Section 9.6. However, high resolution does not come without a cost. In practice, the spectrum cannot be estimated over a continuous frequency interval and must be computed at a finite set of discrete frequency points. Since the minimum-variance estimator is based on \mathbf{R}_x^{-1} , it is very sensitive to the exact frequency points at which the spectrum is estimated. Therefore, the minimum-variance spectrum needs to be computed at a very fine frequency spacing in order to accurately measure the power of such a complex exponential. In some applications where computational cost is a concern, the DFT-based methods are probably preferred, as long as they provide the necessary resolution and sidelobe leakage is properly controlled.

EXAMPLE 9.5.1. In this example, we explore the resolving capability of the minimum-variance spectrum estimator and compare its performance to that of a DFT-based method (Bartlett) and the all-pole method. Two closely spaced complex exponentials, both with an amplitude of $\sqrt{10}$, at discrete-time frequencies of $f = 0.1$ and $f = 0.12$ are contained in noise with unit power $\sigma_w^2 = 1$. We apply the spectrum estimators with time-window lengths (or order) $M = 16, 32, 64$, and 128 to signals consisting of 500 time samples. The estimated spectra were then averaged over 100 realizations. The resulting average spectrum estimates are shown in Figure 9.19. Note that the frequency spacing of the two complex exponentials is $\Delta f = 0.02$, suggesting a time-window length of at least $M = 50$ to resolve them with a DFT-based method. The minimum-variance spectrum estimator, however, is able to resolve them at the $M = 32$ window length, for which they are clearly not distinguishable using the DFT-based method. On the other hand, the all-pole spectrum estimate is able to resolve the two complex exponentials even for as low an order as $M = 16$, for which the minimum-variance spectrum was not successful. In general, the superior resolving capability of the all-pole model over the minimum-variance spectrum estimator is due to an averaging effect that comes about through the nonparametric nature of the minimum-variance method. This subject is explored following the next example. Note that the estimated noise level is most accurately measured by the minimum-variance method in all cases. Recall that the signal amplitude was $\sqrt{10}$, yet the estimated power at the frequencies of the complex exponentials increases as the window length M increases. In the filter bank interpretation of the minimum-variance spectrum estimator, the normalization assumed a constant signal power level across the bandwidth of the frequency-selective filter. However, the complex exponential is actually an impulse in frequency and has zero bandwidth. Therefore, the estimated power will grow with the length of the time window used for the spectrum estimator as a result of this bandwidth normalization. The gain imparted on a complex exponential signal is explored in Example 9.5.2.

EXAMPLE 9.5.2. Consider the complex exponential signal with frequency f_1 contained in noise

$$x(n) = \alpha_1 e^{j2\pi f_1 n} + w(n)$$

where $\alpha_1 = |\alpha_1|e^{j\psi_1}$ is a complex number with constant amplitude $|\alpha_1|$ and random phase ψ_1 with uniform distribution over $[0, 2\pi]$. The correlation matrix of $x(n)$ is

$$\mathbf{R}_x = |\alpha_1|^2 \mathbf{v}(f_1) \mathbf{v}^H(f_1) + \sigma_w^2 \mathbf{I}$$

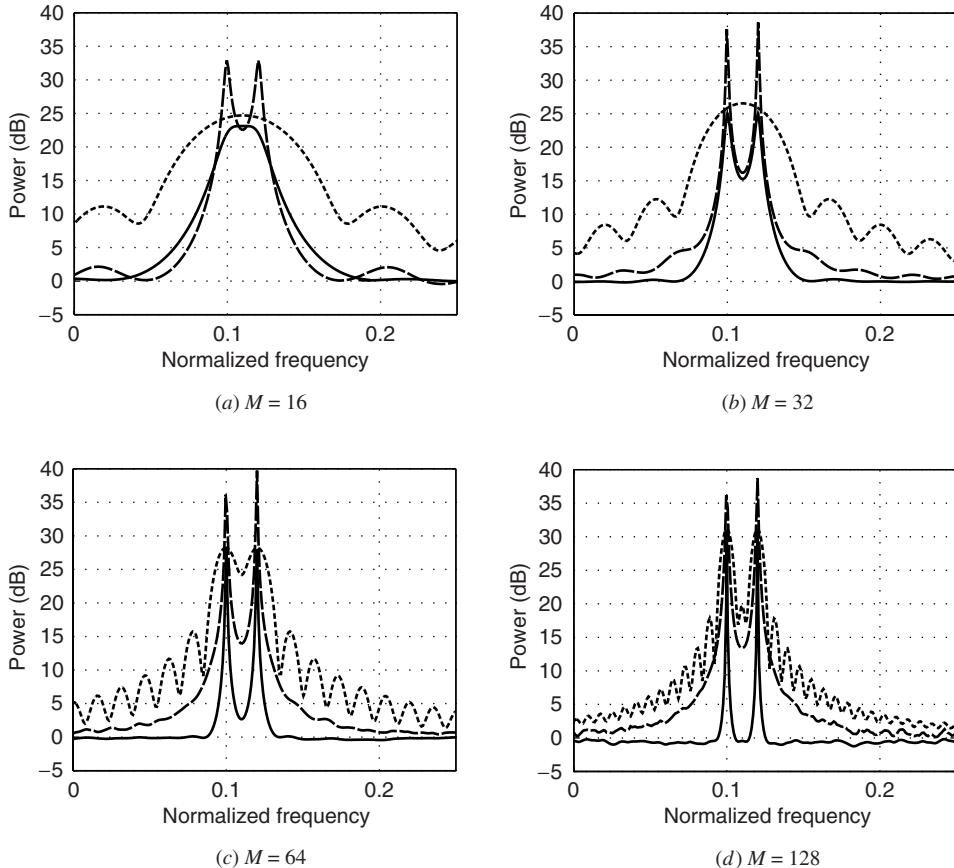


FIGURE 9.19

Comparison of the minimum-variance (solid line), all-pole (large dashed line), and Fourier-based (small dashed line) spectrum estimators for different time window lengths M .

Using the matrix inversion lemma from Appendix B, we can write the inverse of the correlation matrix as

$$\mathbf{R}_x^{-1} = \frac{1}{\sigma_w^2} \mathbf{I} - \frac{|\alpha_1|^2 \mathbf{v}(f_1) \mathbf{v}^H(f_1)}{\sigma_w^2 [\sigma_w^2 + |\alpha_1|^2 \mathbf{v}(f_1) \mathbf{v}^H(f_1)]} = \frac{1}{\sigma_w^2} \left[\mathbf{I} - \frac{|\alpha_1|^2}{\sigma_w^2 + M |\alpha_1|^2} \mathbf{v}(f_1) \mathbf{v}^H(f_1) \right]$$

Substituting this expression for the inverse of the correlation matrix into (9.5.11) for the minimum-variance spectrum estimate, we have

$$\hat{R}_M^{(\text{mv})}(e^{j2\pi f_1}) = \frac{M}{\mathbf{v}^H(f_1) \mathbf{R}_x^{-1} \mathbf{v}(f_1)} = \frac{\sigma_w^2}{1 - \frac{|\alpha_1|^2/M}{\sigma_w^2 + M |\alpha_1|^2} |\mathbf{v}^H(f_1) \mathbf{v}(f_1)|^2}$$

Recall that the norm of the frequency vector $\mathbf{v}(f)$ from (9.5.6) is $\mathbf{v}^H(f_1) \mathbf{v}(f_1) = M$. Therefore, the minimum-variance power spectrum estimate at $f = f_1$ is

$$\hat{R}_M^{(\text{mv})}(e^{j2\pi f_1}) = \sigma_w^2 + M |\alpha_1|^2$$

that is, the sum of the noise power and the signal power times the time-window length. This gain of M on the signal power comes about through the normalization we imposed on our filter in (9.5.8). This normalization assumed the signal had equal amplitude across the passband of the filter. A complex exponential, on the other hand, has no bandwidth and thus this normalization imparts a gain of M on the signal. Therefore, if an estimate of the amplitude of a complex exponential is desired, this gain must be accounted for. Last, let us examine the behavior of the minimum variance spectrum estimator at the other frequencies that contain only noise. In the

case of $M \gg 1$, then $\mathbf{v}^H(f) \mathbf{v}(f_1) \approx 0$ and

$$\hat{R}_M^{(\text{mv})}(e^{j2\pi f}) \approx \sigma_w^2$$

Relationship between the minimum-variance and all-pole spectrum estimation methods

The minimum-variance spectrum estimator has an interesting relation to the all-pole spectrum estimator discussed in Section 9.4. Recall from (9.5.11) that the minimum-variance spectrum estimate is a function of \mathbf{R}_x^{-1} . The inverse of a Toeplitz correlation matrix was studied in Chapter 7 and from (7.7.8) can be written as an LDL^H decomposition

$$\mathbf{R}_x^{-1} = \mathbf{A}^H \bar{\mathbf{D}}^{-1} \mathbf{A} \quad (9.5.16)$$

where the upper triangular matrix \mathbf{A} from (7.7.9) is given by

$$\mathbf{A} = \begin{bmatrix} 1 & a_1^{(M)*} & a_2^{(M)*} & \cdots & a_{M-1}^{(M)*} \\ 0 & 1 & a_1^{(M-1)*} & \cdots & a_{M-2}^{(M-1)*} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_1^{(2)*} \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \quad (9.5.17)$$

and the diagonal matrix $\bar{\mathbf{D}}$ is

$$\bar{\mathbf{D}} = \text{diag}\{P_M, P_{M-1}, \dots, P_1\} \quad (9.5.18)$$

Recall from Chapter 7 that the columns of the lower triangular factor $\mathbf{L} = \mathbf{A}^H$ are the coefficients of the forward linear predictors of orders $m = 1, 2, \dots, M-1$ for the signal $x(n)$ with correlation matrix \mathbf{R}_x . P_m is the residual output power resulting from the application of this m th-order forward linear predictor to the signal $x(n)$. In turn, the forward linear predictor coefficients form the m th-order all-pole model. The model orders are found in descending order as the column index increases. Let us denote the column vector of coefficients for the m th-order all-pole model as

$$\mathbf{a}_m = [1 \ a_1^{(m)} \ a_2^{(m)} \ \cdots \ a_m^{(m)}]^T \quad (9.5.19)$$

We can write the estimate of the spectrum derived from an m th-order all-pole model in vector notation as

$$\hat{R}_m^{(\text{ap})}(e^{j2\pi f}) = \frac{P_m}{|\mathbf{v}_m^H(f)\mathbf{a}_m|^2} \quad (9.5.20)$$

where $\mathbf{v}_m(f)$ is the frequency vector from (9.5.6) of order $M = m$. Then we can substitute (9.5.16) into the minimum-variance spectrum estimator from (9.5.11) to obtain

$$\hat{R}_M^{(\text{mv})}(e^{j2\pi f}) = \frac{M}{\mathbf{v}_M^H(f)\mathbf{R}_x^{-1}\mathbf{v}_M(f)} = \frac{M}{\mathbf{v}_M^H(f)\mathbf{A}^H \bar{\mathbf{D}}^{-1} \mathbf{A} \mathbf{v}_M(f)} \quad (9.5.21)$$

Therefore, we can write the following relationship between the reciprocals of the minimum-variance and all-pole model spectrum estimators

$$\frac{1}{\hat{R}_M^{(\text{mv})}(e^{j2\pi f})} = \sum_{m=1}^M \frac{|\mathbf{v}_m^H(f)\mathbf{a}_m|^2}{MP_m} = \frac{1}{M} \sum_{m=1}^M \frac{1}{\hat{R}_m^{(\text{ap})}(e^{j2\pi f})} \quad (9.5.22)$$

where the subscripts denote the order of the respective spectrum estimators. Thus, the minimum-variance spectrum estimator for a filter of length M is formed by averaging spectrum estimates from all-pole models of orders 1 through M . Note that the resolving capabilities of the all-pole model improve with increasing model order. As a result, the resolution of the minimum-variance spectrum estimator must be worse than that of the

M th-order all-pole model as we observed in Example 9.5.1. However, on the other hand, this averaging of all-pole model spectra indicates a lower variance for the minimum-variance spectrum estimator.

9.6 HARMONIC MODELS AND FREQUENCY ESTIMATION TECHNIQUES

The pole-zero models we have discussed so far assume a linear time-invariant system that is excited by white noise. However, in many applications, the signals of interest are complex exponentials contained in white noise for which a *sinusoidal* or *harmonic model* is more appropriate. Signals consisting of complex exponentials are found as formant frequencies in speech processing, moving targets in radar, and spatially propagating signals in array processing.[†] For real signals, complex exponentials make up a complex conjugate pair (sinusoids), whereas for complex signals, they may occur at a single frequency.

For complex exponentials found in noise, the parameters of interest are the frequencies of the signals. Therefore, our goal is to estimate these frequencies from the data. One might consider estimating the power spectrum by using the nonparametric methods discussed in Chapter 5 or the minimum-variance spectral estimate from Section 9.5. The frequency estimates of the complex exponentials are then the frequencies at which peaks occur in the spectrum. Certainly, the use of these nonparametric methods seems appropriate for complex exponential signals since they make no assumptions about the underlying process. We might also consider making use of an all-pole model for the purposes of spectrum estimation as discussed in Section 9.4.1, also known as the maximum entropy method (MEM) spectral estimation technique. Even though some of these methods can achieve very fine resolution, none of these methods accounts for the underlying model of complex exponentials in noise. As in all modeling problems, the use of the appropriate model is desirable from an intuitive point of view and advantageous in terms of performance. We begin by describing the harmonic signal model, deriving the model in a vector notation, and looking at the eigendecomposition of the correlation matrix of complex exponentials in noise. Then we describe frequency estimation methods based on the harmonic model: the Pisarenko harmonic decomposition, and the MUSIC, minimum-norm, and ESPRIT algorithms.

These methods have the ability to resolve complex exponentials closely spaced in frequency and has led to the name *superresolution* commonly being associated with them. However, a word of caution on the use of these harmonic models. The high level of performance in terms of resolution is achieved by assuming an underlying model of the data. As with all other parametric methods, the performance of these techniques depends upon how closely this mathematical model matches the actual physical process that produced the signals. Deviations from this assumption result in model mismatch and will produce frequency estimates for a signal that may not have been produced by complex exponentials. In this case, the frequency estimates have little meaning.

9.6.1 Harmonic Model

Consider the signal model that consists of P complex exponentials in noise

$$x(n) = \sum_{p=1}^P \alpha_p e^{j2\pi n f_p} + w(n) \quad (9.6.1)$$

[†]In array processing, a spatially propagating wave produces a complex exponential signal as measured across uniformly spaced sensors in an array. The frequency of the complex exponential is determined by the angle of arrival of the impinging, spatially propagating signal. Thus, in array processing the frequency estimation problem is known as *angle-of-arrival (AOA)* or *direction-of-arrival (DOA)* estimation. This topic is discussed in Section 11.7.

The normalized, discrete-time frequency of the p th component is

$$f_p = \frac{\omega_p}{2\pi} = \frac{F_p}{F_s} \quad (9.6.2)$$

where ω_p is the discrete-time frequency in radians, F_p is the actual frequency of the p th complex exponential, and F_s is the sampling frequency. The complex exponentials may occur either individually or in complex conjugate pairs, as in the case of real signals. In general, we want to estimate the frequencies and possibly also the amplitudes of these signals. Note that the phase of each complex exponential is contained in the amplitude, that is,

$$\alpha_p = |\alpha_p| e^{j\psi_p} \quad (9.6.3)$$

where the phases ψ_p are uncorrelated random variables uniformly distributed over $[0, 2\pi]$. The magnitude $|\alpha_p|$ and the frequency f_p are deterministic quantities. If we consider the spectrum of a harmonic process, we note that it consists of a set of impulses with a constant background level at the power of the white noise $\sigma_w^2 = E\{|w(n)|^2\}$. As a result, the power spectrum of complex exponentials is commonly referred to as a *line spectrum*, as illustrated in Figure 9.20.

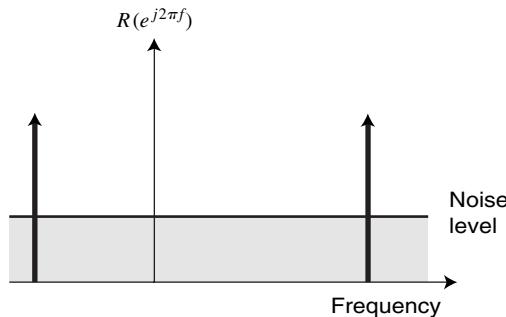


FIGURE 9.20

The spectrum of complex exponentials in noise.

Since we will make use of matrix methods based on a certain time window of length M , it is useful to characterize the signal model in the form of a vector over this time window consisting of the sample delays of the signal. Consider the signal $x(n)$ from (9.6.1) at its current and future $M - 1$ values. This time window can be written as

$$\mathbf{x}(n) = [x(n) \ x(n+1) \ \cdots \ x(n+M-1)]^T \quad (9.6.4)$$

We can then write the signal model consisting of complex exponentials in noise from (9.6.1) for a length- M time-window vector as

$$\mathbf{x}(n) = \sum_{p=1}^P \alpha_p \mathbf{v}(f_p) e^{j2\pi n f_p} + \mathbf{w}(n) = \mathbf{s}(n) + \mathbf{w}(n) \quad (9.6.5)$$

where $\mathbf{w}(n) = [w(n) \ w(n+1) \ \cdots \ w(n+M-1)]^T$ is the time-window vector of white noise and

$$\mathbf{v}(f) = [1 \ e^{j2\pi f} \ \cdots \ e^{j2\pi(M-1)f}]^T \quad (9.6.6)$$

is the time-window frequency vector. Note that $\mathbf{v}(f)$ is simply a length- M DFT vector at frequency f . We differentiate here between the signal $\mathbf{s}(n)$, consisting of the sum of complex exponentials, and the noise component $\mathbf{w}(n)$, respectively.

Consider the time-window vector model consisting of a sum of complex exponentials in noise from (9.6.5). The autocorrelation matrix of this model can be written as the sum of signal and noise autocorrelation matrices

$$\begin{aligned} \mathbf{R}_x &= E\{\mathbf{x}(n)\mathbf{x}^H(n)\} = \mathbf{R}_s + \mathbf{R}_w \\ &= \sum_{p=1}^P |\alpha_p|^2 \mathbf{v}(f_p) \mathbf{v}^H(f_p) + \sigma_w^2 \mathbf{I} = \mathbf{V} \mathbf{A} \mathbf{V}^H + \sigma_w^2 \mathbf{I} \end{aligned} \quad (9.6.7)$$

where

$$\mathbf{V} = [\mathbf{v}(f_1) \mathbf{v}(f_2) \cdots \mathbf{v}(f_P)] \quad (9.6.8)$$

is an $M \times P$ matrix whose columns are the time-window frequency vectors from (9.6.6) at frequencies f_p of the complex exponentials and

$$\mathbf{A} = \begin{bmatrix} |\alpha_1|^2 & 0 & \cdots & 0 \\ 0 & |\alpha_2|^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & |\alpha_P|^2 \end{bmatrix} \quad (9.6.9)$$

is a diagonal matrix of the powers of each of the respective complex exponentials. The autocorrelation matrix of the white noise is

$$\mathbf{R}_w = \sigma_w^2 \mathbf{I} \quad (9.6.10)$$

which is full rank, as opposed to \mathbf{R}_s which is rank-deficient for $P < M$. In general, we will always choose the length of our time window M to be greater than the number of complex exponentials P .

The autocorrelation matrix can also be written in terms of its eigendecomposition

$$\mathbf{R}_x = \sum_{m=1}^M \lambda_m \mathbf{q}_m \mathbf{q}_m^H = \mathbf{Q} \Lambda \mathbf{Q}^H \quad (9.6.11)$$

where λ_m are the eigenvalues in descending order, that is, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$, and \mathbf{q}_m are their corresponding eigenvectors. Here Λ is a diagonal matrix made up of the eigenvalues found in descending order on the diagonal, while the columns of \mathbf{Q} are the corresponding eigenvectors. The eigenvalues due to the signals can be written as the sum of the signal power in the time window and the noise:

$$\lambda_m = M|\alpha_m|^2 + \sigma_w^2 \quad \text{for } m \leq P \quad (9.6.12)$$

The remaining eigenvalues are due to the noise only, that is,

$$\lambda_m = \sigma_w^2 \quad \text{for } m > P \quad (9.6.13)$$

Therefore, the P largest eigenvalues correspond to the signal made up of complex exponentials and the remaining eigenvalues have equal value and correspond to the noise. Thus, we can partition the correlation matrix into portions due to the signal and noise eigenvectors

$$\begin{aligned} \mathbf{R}_x &= \sum_{m=1}^P (M|\alpha_m|^2 + \sigma_w^2) \mathbf{q}_m \mathbf{q}_m^H + \sum_{m=P+1}^M \sigma_w^2 \mathbf{q}_m \mathbf{q}_m^H \\ &= \mathbf{Q}_s \Lambda_s \mathbf{Q}_s^H + \sigma_w^2 \mathbf{Q}_w \mathbf{Q}_w^H \end{aligned} \quad (9.6.14)$$

where

$$\mathbf{Q}_s = [\mathbf{q}_1 \mathbf{q}_2 \cdots \mathbf{q}_P] \quad \mathbf{Q}_w = [\mathbf{q}_{P+1} \cdots \mathbf{q}_M] \quad (9.6.15)$$

are matrices whose columns consist of the signal and noise eigenvectors, respectively. The matrix Λ_s is a $P \times P$ diagonal matrix containing the signal eigenvalues from (9.6.12). Thus, the M -dimensional subspace that contains the observations of the time-window signal vector from (9.6.5) can be split into two subspaces spanned by the signal and noise eigenvectors, respectively. These two subspaces, known as the *signal subspace* and the *noise subspace*, are orthogonal to each other since the correlation matrix is Hermitian symmetric.[†] All the subspace methods discussed later in this section rely on the partitioning of the vector space into signal and noise subspaces. Recall from Chapter 8 in (8.2.29) that the projection matrix

[†]The eigenvectors of a Hermitian symmetric matrix are orthogonal.

from an M -dimensional space onto an L -dimensional subspace ($L < M$) spanned by a set of vectors $\mathbf{Z} = [\mathbf{z}_1 \mathbf{z}_2 \cdots \mathbf{z}_L]$ is

$$\mathbf{P} = \mathbf{Z}(\mathbf{Z}^H \mathbf{Z})^{-1} \mathbf{Z}^H \quad (9.6.16)$$

Therefore, we can write the matrices that project an arbitrary vector onto the signal and noise subspaces as

$$\mathbf{P}_s = \mathbf{Q}_s \mathbf{Q}_s^H \quad \mathbf{P}_w = \mathbf{Q}_w \mathbf{Q}_w^H \quad (9.6.17)$$

since the eigenvectors of the correlation matrix are orthonormal ($\mathbf{Q}_s^H \mathbf{Q}_s = \mathbf{I}$ and $\mathbf{Q}_w^H \mathbf{Q}_w = \mathbf{I}$). Since the two subspaces are orthogonal

$$\mathbf{P}_w \mathbf{Q}_s = \mathbf{0} \quad \mathbf{P}_s \mathbf{Q}_w = \mathbf{0} \quad (9.6.18)$$

then all the time-window frequency vectors from (9.6.5) must lie completely in the signal subspace, that is,

$$\mathbf{P}_s \mathbf{v}(f_p) = \mathbf{v}(f_p) \quad \mathbf{P}_w \mathbf{v}(f_p) = \mathbf{0} \quad (9.6.19)$$

These concepts are central to the subspace-based frequency estimation methods discussed in Sections 9.6.2 through 9.6.5.

Note that in our analysis, we are considering the theoretical or true correlation matrix \mathbf{R}_x . In practice, the correlation matrix is not known and must be estimated from the measured data samples. If we have a time-window signal vector from (9.6.4), then we can form the data matrix by stacking the rows with measurements of the time-window data vector at a time n

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^T(0) \\ \mathbf{x}^T(1) \\ \vdots \\ \mathbf{x}^T(n) \\ \vdots \\ \mathbf{x}^T(N-2) \\ \mathbf{x}^T(N-1) \end{bmatrix} = \begin{bmatrix} x(0) & x(1) & \cdots & x(M-1) \\ x(1) & x(2) & \cdots & x(M) \\ \vdots & \vdots & \vdots & \vdots \\ x(n) & x(n+1) & \cdots & x(n+M-1) \\ \vdots & \vdots & \vdots & \vdots \\ x(N-2) & x(N-1) & \cdots & x(N+M-3) \\ x(N-1) & x(N) & \cdots & x(N+M-2) \end{bmatrix} \quad (9.6.20)$$

which has dimensions of $N \times M$, where N is the number of data records or snapshots and M is the time-window length. From this matrix, we can form an estimate of the correlation matrix, referred to as the sample correlation matrix

$$\hat{\mathbf{R}}_x = \frac{1}{N} \mathbf{X}^H \mathbf{X} \quad (9.6.21)$$

In the case of an estimated sample correlation matrix, the noise eigenvalues are no longer equal because of the finite number of samples used to compute $\hat{\mathbf{R}}$. Therefore, the nice, clean threshold between signal and noise eigenvalues, as described in (9.6.12) and (9.6.13), no longer exists. The model order estimation techniques discussed in Section 9.2 can be employed to attempt to determine the number of complex exponentials P present. In practice, these methods are best used as rough estimates, as their performance is not very accurate, especially for short data records.

For several of the frequency estimation techniques described in this section, the analysis considers the use of eigenvalues and eigenvectors of the correlation matrix for the purposes of defining signal and noise subspaces.[†] In practice, we estimate the signal and noise subspaces by using the eigenvectors and eigenvalues of the sample correlation matrix. Note that for notational expedience we will not differentiate between eigenvectors and eigenvalues of

[†]The ESPRIT method uses a singular value decomposition of data matrix \mathbf{X} .

the true and sample correlation matrices. However, the reader should always keep in mind that the sample correlation matrix eigendecomposition is what must be used for implementation. We note that use of an estimate rather than the true correlation matrix will result in a degradation in performance, the analysis of which is beyond the scope of this book.

9.6.2 Pisarenko Harmonic Decomposition

The *Pisarenko harmonic decomposition* (PHD) was the first frequency estimation method proposed that was based on the eigendecomposition of the correlation matrix and its partitioning into signal and noise subspaces (Pisarenko 1973). This method uses the eigenvector associated with the smallest eigenvalue to estimate the frequencies of the complex exponentials. Although this method has limited practical use owing to its sensitivity to noise, it is of great theoretical interest because it was the first method based on signal and noise subspace principles and it helped to fuel the development of many well-known subspace methods, such as MUSIC and ESPRIT.

Consider the model of complex exponentials contained in noise in (9.6.5) and the eigendecomposition of its correlation matrix in (9.6.14). The eigenvector corresponding to the minimum eigenvalue must be orthogonal to all the eigenvectors in the signal subspace. Thus, we choose the time window to be of length

$$M = P + 1 \quad (9.6.22)$$

that is, 1 greater than the number of complex exponentials. Therefore, the noise subspace consists of a single eigenvector

$$\mathbf{Q}_w = \mathbf{q}_M \quad (9.6.23)$$

corresponding to the minimum eigenvalue λ_M . By virtue of the orthogonality between the signal and noise subspaces, each of the P complex exponentials in the time-window signal vector model in (9.6.5) is orthogonal to this eigenvector

$$\mathbf{v}^H(f_p)\mathbf{q}_M = \sum_{k=1}^M q_M(k)e^{-j2\pi f_p(k-1)} = 0 \quad \text{for } m \leq P \quad (9.6.24)$$

Making use of this property, we can compute

$$\bar{R}_{\text{phd}}(e^{j2\pi f}) = \frac{1}{|\mathbf{v}^H(f)\mathbf{q}_M|^2} = \frac{1}{|Q_M(e^{j2\pi f})|^2} \quad (9.6.25)$$

which is commonly referred to as a *pseudospectrum*. The frequencies are then estimated by observing the P peaks in $\bar{R}_{\text{phd}}(e^{j2\pi f})$. Note that since (9.6.25) requires a search of all frequencies $-0.5 \leq f \leq 0.5$, in practice a dense sampling of the frequencies is generally necessary. The quantity

$$Q_M(e^{j2\pi f}) = \mathbf{v}^H(f)\mathbf{q}_M = \sum_{k=1}^M q_M(k)e^{-j2\pi f(k-1)} \quad (9.6.26)$$

is simply the Fourier transform of the M th eigenvector corresponding to the minimum eigenvalue. Thus, the pseudospectrum for the Pisarenko harmonic decomposition $\bar{R}_{\text{phd}}(e^{j2\pi f})$ can be efficiently implemented by computing the FFT of \mathbf{q}_M with sufficient zero padding to provide the necessary frequency resolution. Then $\bar{R}_{\text{phd}}(e^{j2\pi f})$ is simply the reciprocal of the spectrum of the noise eigenvector, that is, the squared magnitude of its Fourier transform. Note that $\bar{R}_{\text{phd}}(e^{j2\pi f})$ is not an estimate of the true power spectrum since it contains no information about the powers of the complex exponentials $|\alpha_p|^2$ or the background noise level σ_w^2 . However, these amplitudes can be found by using the estimated frequencies and the corresponding time-window frequency vectors along with the relationship of eigenvalues and eigenvectors. See Problem 9.24 for details.

Alternately, the frequencies of the complex exponentials can be found by computing the zeros of the Fourier transform of the M th eigenvector in (9.6.23). The z -transform of this eigenvector is

$$Q_M(z) = \sum_{k=1}^M q_M(k)z^{-k} = \prod_{k=1}^{M-1} (1 - e^{j2\pi f_k} z^{-1}) \quad (9.6.27)$$

where the phases of the $P = M - 1$ roots of this polynomial are the frequencies f_k of the $P = M - 1$ complex exponentials.

As we stated up front, the significance of the Pisarenko harmonic decomposition is seen mostly from a theoretical perspective. The limitations of its practical use stem from the fact that it uses a single noise eigenvector and, as a result, lacks the necessary robustness needed for most applications. Since the correlation matrix is not known and must be estimated from data, the resulting noise eigenvector of the estimated correlation matrix is only an estimate of the actual noise eigenvector. Because we only use one noise eigenvector, this method is very sensitive to any errors in the estimation of the noise eigenvector.

EXAMPLE 9.6.1. We demonstrate the use of the Pisarenko harmonic decomposition with a sinusoid in noise. The amplitude and frequency of the sinusoid are $\alpha = 1$ and $f = 0.2$, respectively. The additive noise has unit power ($\sigma_w^2 = 1$). Using MATLAB, this signal is generated:

```
x = sin(2*pi*f*[0:(N-1)]') + (randn(N,1)+j*randn(N,1))/sqrt(2);
```

Since the number of complex exponentials is equal to $P = 2$ (a complex conjugate pair for a sinusoid), the time-window length is chosen to be $M = 3$. After forming the $N \times M$ data matrix \mathbf{X} and computing the sample correlation matrix $\hat{\mathbf{R}}_x$, we can compute the pseudospectrum as follows:

```
[Q0,D] = eig(R); % eigendecomposition
[lambda,index] = sort(abs(diag(D))); % order by eigenvalue magnitude
lambda = lambda(M:-1:1); Q=Q0(:,index(M:-1:1));
Rbar = 1./abs(fftshift(fft(Q(:,M),Nfft))).^2;
```

Figure 9.21 shows the pseudospectrum of the Pisarenko harmonic decomposition for a single realization with an FFT size of 1024. Note the two peaks near $f = \pm 0.2$. Recall that

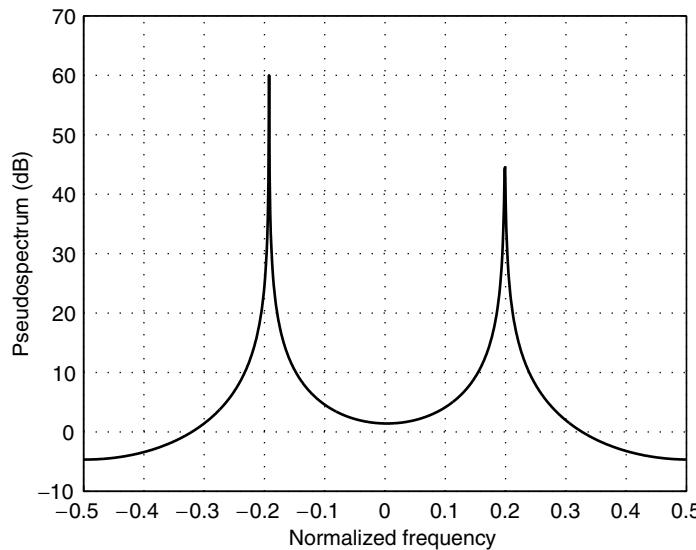


FIGURE 9.21

Pseudospectrum for the Pisarenko harmonic decomposition of a sinusoid in noise with frequency $f = 0.2$.

this is a pseudospectrum, so that the actual values do not correspond to an estimate of power. A MATLAB routine for estimating frequencies using the Pisarenko harmonic decomposition is provided in phd.m.

9.6.3 MUSIC Algorithm

The *multiple signal classification (MUSIC)* frequency estimation method was proposed as an improvement on the Pisarenko harmonic decomposition (Biennvenu and Kopp 1983; Schmidt 1986). Like the Pisarenko harmonic decomposition, the M -dimensional space is split into signal and noise components using the eigenvectors of the correlation matrix from (9.6.15). However, rather than limit the length of the time window to $M = P + 1$, that is, 1 greater than the number of complex exponentials, allow the size of the time window to be $M > P + 1$. Therefore, the noise subspace has a dimension greater than 1. Using this larger dimension allows for averaging over the noise subspace, providing an improved, more robust frequency estimation method than Pisarenko harmonic decomposition.

Because of the orthogonality between the noise and signal subspaces, all the time-window frequency vectors of the complex exponentials are orthogonal to the noise subspace from (9.6.19). Thus, for each eigenvector ($P < m \leq M$)

$$\mathbf{v}^H(f_p)\mathbf{q}_m = \sum_{k=1}^M q_m(k)e^{-j2\pi f_p(k-1)} = 0 \quad (9.6.28)$$

for all the P frequencies f_p of the complex exponentials. Therefore, if we compute a pseudospectrum for each noise eigenvector as

$$\bar{R}_m(e^{j2\pi f}) = \frac{1}{|\mathbf{v}^H(f)\mathbf{q}_m|^2} = \frac{1}{|Q_m(e^{j2\pi f})|^2} \quad (9.6.29)$$

the polynomial $Q_m(e^{j2\pi f})$ has $M - 1$ roots, P of which correspond to the frequencies of the complex exponentials. These roots produce P peaks in the pseudospectrum from (9.6.29). Note that the pseudospectra of all $M - P$ noise eigenvectors share these roots that are due to the signal subspace. The remaining roots of the noise eigenvectors, however, occur at different frequencies. There are no constraints on the location of these roots, so that some may be close to the unit circle and produce extra peaks in the pseudospectrum. A means of reducing the levels of these spurious peaks in the pseudospectrum is to average the $M - P$ pseudospectra of the individual noise eigenvectors

$$\bar{R}_{\text{music}}(e^{j2\pi f}) = \frac{1}{\sum_{m=P+1}^M |\mathbf{v}^H(f)\mathbf{q}_m|^2} = \frac{1}{\sum_{m=P+1}^M |Q_m(e^{j2\pi f})|^2} \quad (9.6.30)$$

which is known as the MUSIC pseudospectrum. The frequency estimates of the P complex exponentials are then taken as the P peaks in this pseudospectrum. Again, the term *pseudospectrum* is used because the quantity in (9.6.30) does not contain information about the powers of the complex exponentials or the background noise level. Note that for $M = P + 1$, the MUSIC method is equivalent to Pisarenko harmonic decomposition.

The implicit assumption in the MUSIC pseudospectrum is that the noise eigenvalues all have equal power $\lambda_m = \sigma_w^2$, that is, the noise is white. However, in practice, when an estimate is used in place of the actual correlation matrix, the noise eigenvalues will not be equal. The differences become more pronounced when the correlation matrix is estimated from a small number of data samples. Thus, a slight variation on the MUSIC algorithm, known as the *eigenvector (ev) method*, was proposed to account for the potentially different

noise eigenvalues (Johnson and DeGraaf 1982). For this method, the pseudospectrum is

$$\bar{R}_{\text{ev}}(e^{j\omega}) = \frac{1}{\sum_{m=P+1}^M \frac{1}{\lambda_m} |\mathbf{v}^H(f) \mathbf{q}_m|^2} = \frac{1}{\sum_{k=P+1}^M \frac{1}{\lambda_m} |Q_m(e^{j2\pi f})|^2} \quad (9.6.31)$$

where λ_m is the eigenvalue corresponding to the eigenvector \mathbf{q}_m . The pseudospectrum of each eigenvector is normalized by its corresponding eigenvalue. In the case of equal noise eigenvalues ($\lambda_m = \sigma_w^2$) for $P + 1 \leq m \leq M$, the eigenvector and MUSIC methods are identical.

The peaks in the MUSIC pseudospectrum correspond to the frequencies at which the denominator in (9.6.30) $\sum_{m=P+1}^M |Q_m(e^{j2\pi f})|^2$ approaches zero. Therefore, we might want to consider the z -transform of this denominator

$$\bar{P}_{\text{music}}(z) = \sum_{m=P+1}^M Q_m(z) Q_m^* \left(\frac{1}{z^*} \right) \quad (9.6.32)$$

which is the sum of the z -transforms of the pseudospectrum due to each noise eigenvector. This $(2M - 1)$ th-order polynomial has $M - 1$ pairs of roots with one inside and one outside the unit circle. Since we assume that the complex exponentials are not damped, their corresponding roots must lie on the unit circle. Thus, if we have found the $M - 1$ roots of (9.6.32), the P closest roots to the unit circle will correspond to the complex exponentials. The phases of these roots are then the frequency estimates. This method of rooting the polynomial corresponding to the MUSIC pseudospectrum is known as *root-MUSIC* (Barabell 1983). Note that in many cases, a rooting method is more efficient than computing a pseudospectrum at a very fine frequency resolution that may require a very large FFT. Statistical performance analyses of the MUSIC algorithm can be found in Kaveh and Barabell (1986) and Stoica and Nehorai (1989). For the performance of the root-MUSIC method see Rao and Hari (1989). A routine for the MUSIC algorithm is provided in `music.m` and a routine for the root-MUSIC algorithm is provided in `rootmusic.m`.

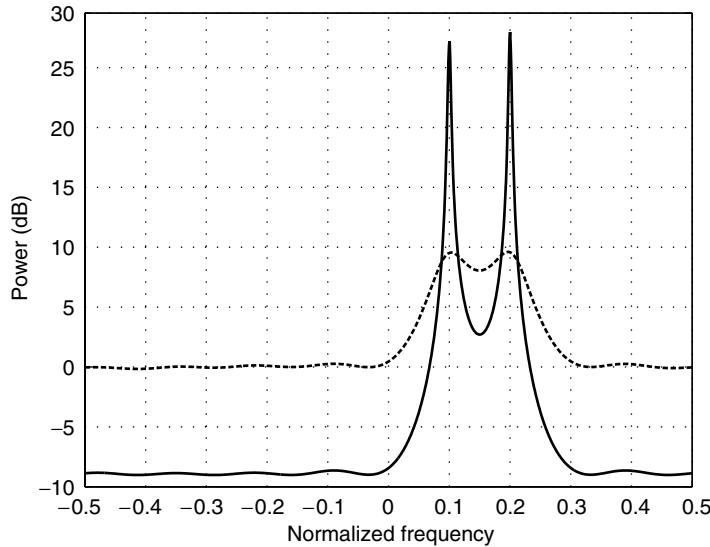
EXAMPLE 9.6.2. In this example, we demonstrate the use of the MUSIC algorithm and examine its performance in terms of resolution with respect to that of the minimum-variance spectral estimator. Consider the following scenario: Two complex exponentials in unit power noise ($\sigma_w^2 = 1$) with normalized frequencies $f = 0.1, 0.2$ both with amplitudes of $\alpha = 1$. We generate $N = 128$ samples of the signal and use a frequency vector of length $M = 8$. Proceeding as we did in Example 9.6.1, we compute the eigendecomposition and partition it into signal and noise subspaces. The MUSIC pseudospectrum is computed as

```
Qbar = zeros(Nfft,1);
for n = 1:(M-P)
    Qbar = Qbar + abs(fftshift(fft(Q(:,M-(n-1)),Nfft))).^2;
end
Rbar = 1./Qbar;
```

The minimum-variance spectral estimate and the MUSIC pseudospectrum are computed and averaged over 1000 realizations using an FFT size of 1024. The result is shown in Figure 9.22. The two exponentials have been clearly resolved using the MUSIC algorithm, whereas they are not very clear using the minimum-variance spectral estimate. Since the minimum-variance spectral estimator is nonparametric and makes no assumptions about the underlying model, it cannot achieve the resolution of the MUSIC algorithm.

9.6.4 Minimum-Norm Method

The minimum-norm method (Kumaresan and Tufts 1983), like the MUSIC algorithm, uses a time-window vector of length $M > P + 1$ for the purposes of frequency estimation. For MUSIC, a larger time window is used than for Pisarenko harmonic decomposition, resulting

**FIGURE 9.22**

Comparison of the minimum-variance spectral estimate (dashed line) and the MUSIC pseudospectrum (solid line) for two complex exponentials in noise.

in a larger noise subspace. The use of a larger subspace provides the necessary robustness for frequency estimation when an estimated correlation matrix is used. The same principle is applied in the minimum-norm frequency estimation method. However, rather than average the pseudospectra of all the noise subspace eigenvectors to reduce spurious peaks, as in the case of the MUSIC algorithm, a different approach is taken.

Consider a single vector \mathbf{u} contained in the noise subspace. The pseudospectrum of this vector is given by

$$\bar{R}(e^{j2\pi f}) = \frac{1}{|\mathbf{v}^H(f)\mathbf{u}|^2} \quad (9.6.33)$$

Since the vector \mathbf{u} lies in the noise subspace, its pseudospectrum in (9.6.33) has P peaks corresponding to the complex exponentials in the signal subspace. However, \mathbf{u} is length M so that its pseudospectrum may exhibit an additional $M - P - 1$ peaks that do not correspond to the frequencies of the complex exponentials. These spurious peaks lead to frequency estimation errors. In the case of Pisarenko harmonic decomposition, spurious peaks were not a concern since $M = P + 1$ and therefore its pseudospectrum in (9.6.25) only had P peaks. On the other hand, the MUSIC algorithm diluted the strength of these spurious peaks since its pseudospectrum in (9.6.30) is produced by averaging the pseudospectra of the $M - P$ noise eigenvectors.

Recall the projection onto the noise subspace from (9.6.17) is

$$\mathbf{P}_w = \mathbf{Q}_w \mathbf{Q}_w^H \quad (9.6.34)$$

where \mathbf{Q}_w is the matrix of noise eigenvectors. Therefore, for any vector \mathbf{u} that lies in the noise subspace

$$\mathbf{P}_w \mathbf{u} = \mathbf{u} \quad \mathbf{P}_s \mathbf{u} = \mathbf{0} \quad (9.6.35)$$

where \mathbf{P}_s is the signal subspace projection matrix and $\mathbf{0}$ is the length- P zero vector. Now let us consider the z -transform of the coefficients of $\mathbf{u} = [u(1) \ u(2) \ \cdots \ u(M)]^T$

$$U(z) = \sum_{k=0}^{M-1} u(k+1)z^{-k} = \prod_{k=1}^P (1 - e^{j2\pi f_k} z^{-1}) \prod_{k=P+1}^{M-1} (1 - z_k z^{-1}) \quad (9.6.36)$$

This polynomial is the product of the P roots corresponding to complex exponentials that lie on the unit circle and the $M - P - 1$ roots that in general do not lie directly on the unit circle but can potentially produce spurious peaks in the pseudospectrum of \mathbf{u} . Therefore, we want to choose \mathbf{u} so that it minimizes the spurious peaks due to these other roots of its associated polynomial $U(z)$.

The minimum-norm method, as its name implies, seeks to minimize the norm of \mathbf{u} in order to avoid spurious peaks in its pseudospectrum. Using (9.6.35), the norm of a vector \mathbf{u} contained in the noise subspace is

$$\|\mathbf{u}\|^2 = \mathbf{u}^H \mathbf{u} = \mathbf{u}^H \mathbf{P}_w \mathbf{u} \quad (9.6.37)$$

However, an unconstrained minimization of this norm will produce the zero vector. Therefore, we place the constraint that the first element of \mathbf{u} must equal 1.[†] This constraint can be expressed as

$$\delta_1^H \mathbf{u} = 1 \quad (9.6.38)$$

where $\delta_1 = [1 \ 0 \ \dots \ 0]^T$. Then the determination of the minimum-norm vector comes down to solving the following constrained minimization problem:

$$\min \|\mathbf{u}\|^2 = \mathbf{u}^H \mathbf{P}_w \mathbf{u} \quad \text{subject to} \quad \delta_1^H \mathbf{u} = 1 \quad (9.6.39)$$

The solution can be found by using Lagrange multipliers (see Appendix B) and is given by

$$\mathbf{u}_{mn} = \frac{\mathbf{P}_w \delta_1}{\delta_1^H \mathbf{P}_w \delta_1} \quad (9.6.40)$$

The frequency estimates are then obtained from the peaks in the pseudospectrum of the minimum-norm (mn) vector, \mathbf{u}_{mn}

$$\bar{R}_{mn}(e^{j2\pi f}) = \frac{1}{|\mathbf{v}^H(f)\mathbf{u}_{mn}|^2} \quad (9.6.41)$$

The performance of the minimum-norm frequency estimation method is similar to that of MUSIC. For a performance comparison see Kaveh and Barabell (1986). Note that it is also possible to implement the minimum-norm method by rooting a polynomial rather than computing a pseudospectrum (see Problem 9.25).

EXAMPLE 9.6.3. In this example, we illustrate the use of the minimum-norm method and compare its performance to that of the other three frequency estimation methods discussed in this chapter: Pisarenko harmonic decomposition, the MUSIC algorithm, and the eigenvector method. The pseudospectrum of the minimum-norm method is found by first computing the minimum-norm vector \mathbf{u}_{mn} and then finding its pseudospectrum, that is,

```
delta1 = zeros(M,1); delta1(1) = 1;
Pn=Q(:,(P+1):M)*Q(:,(P+1):M)'; % noise subspace projection matrix
u = (Pn*e1)/(e1'*Pn*e1); % minimum-norm vector
Rbar = 1./abs(fftshift(fft(u,Nfft))).^2; % pseudospectrum
```

Consider the case of $P = 4$ complex exponentials in noise with frequencies $f = 0.1, 0.25, 0.4$, and -0.1 , all with an amplitude of $\alpha = 1$. The power of the noise is set to $\alpha_w^2 = 1$ with 100 realizations. The time-window length used was $M = 8$ for all the methods except Pisarenko harmonic decomposition, which is constrained to use $M = P + 1 = 5$. The pseudospectra are shown in Figure 9.23 with an FFT size of 1024, where we have not averaged in order to demonstrate the variance of the various methods. Here we see the large variance in the frequency estimates that is produced by Pisarenko harmonic decomposition compared to the other methods, which is a direct result of using a one-dimensional noise subspace. The other methods all perform comparably in terms of estimating the frequencies of the complex exponentials. Note the fluctuations in the pseudospectrum of the eigenvector method that result from the normalization

[†]The choice of a value of 1 is somewhat arbitrary, since any nonzero constant will result in a similar solution.

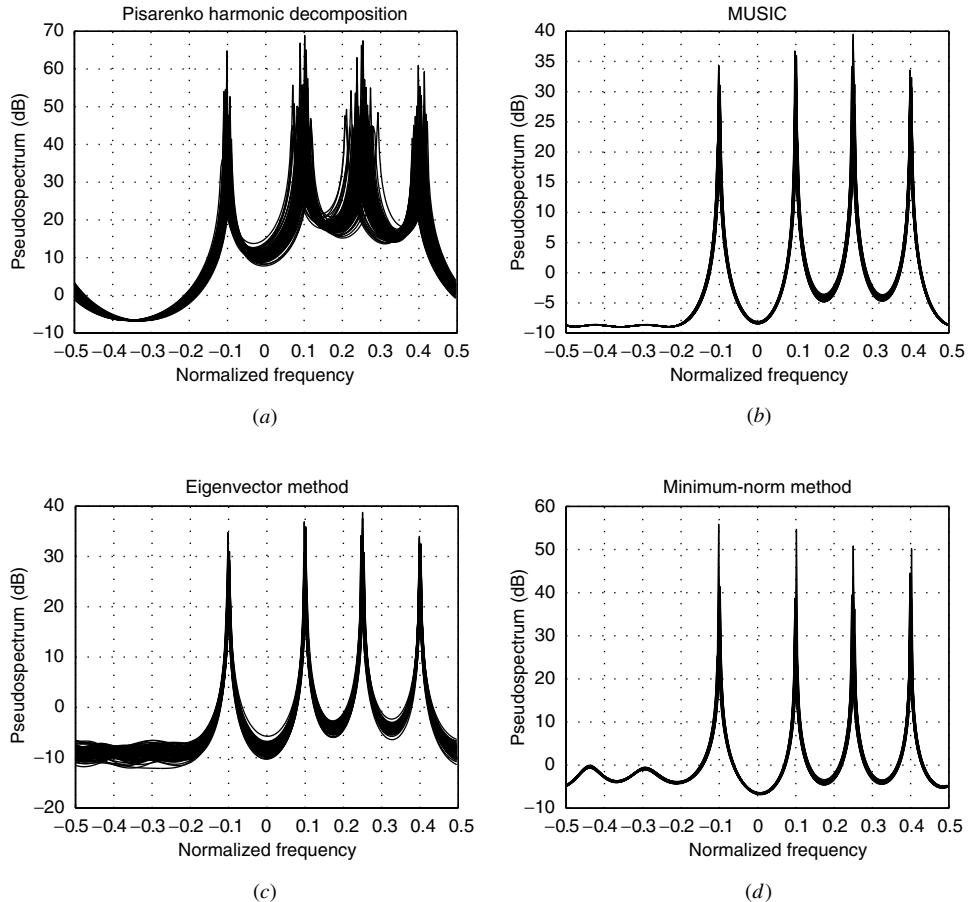


FIGURE 9.23

Comparison of the eigendecomposition-based frequency estimation methods: (a) Pisarenko harmonic decomposition, (b) MUSIC, (c) eigenvector method, and (d) minimum-norm method.

by the eigenvalues. Since these eigenvalues vary over realizations, the pseudospectra will also reflect a similar variation. Routines for the eigenvector method and the minimum-norm method are provided in `ev_method.m` and `minnorm.m`, respectively.

9.6.5 ESPRIT Algorithm

A frequency estimation technique that is built upon the same principles as other subspace methods but further exploits a deterministic relationship between subspaces is the *estimation of signal parameters via rotational invariance techniques (ESPRIT)* algorithm. This method differs from the other subspace methods discussed so far in this chapter in that the signal subspace is estimated from the data matrix \mathbf{X} rather than the estimated correlation matrix $\hat{\mathbf{R}}_x$. The essence of ESPRIT lies in the rotational property between staggered subspaces that is invoked to produce the frequency estimates. In the case of a discrete-time signal or time series, this property relies on observations of the signal over two identical intervals staggered in time. This condition arises naturally for discrete-time signals, provided that the sampling is performed uniformly in time.[†] Extensions of the ESPRIT method to a spatial

[†]This condition is violated in the case of a nonuniformly sampled time series.

array of sensors, the application for which it was originally proposed, will be discussed in Chapter 11 in Section 11.7. We first describe the original, least-squares version of the algorithm (Roy et al. 1986) and then extend the derivation to total least-squares ESPRIT (Roy and Kailath 1989), which is the preferred method for use. Since the derivation of the algorithm requires an extensive amount of formulation and matrix manipulations, we have included a block diagram in Figure 9.24 to be used as a guide through this process.

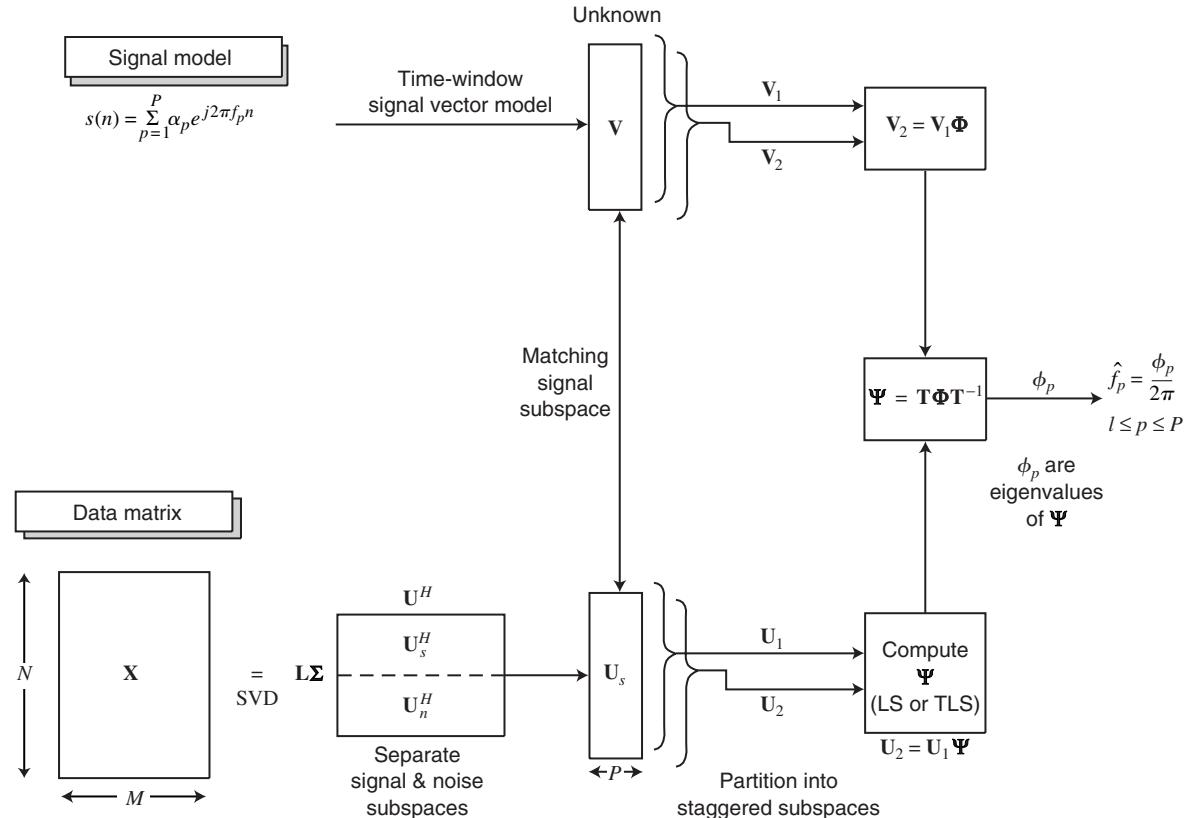


FIGURE 9.24

Block diagram demonstrating the flow of the ESPRIT algorithm starting from the data matrix through the frequency estimates.

Consider a single complex exponential $s_0(n) = e^{j2\pi f n}$ with complex amplitude α and frequency f . This signal has the following property

$$s_0(n+1) = \alpha e^{j2\pi f(n+1)} = s_0(n)e^{j2\pi f} \quad (9.6.42)$$

that is, the next sample value is a phase-shifted version of the current value. This phase shift can be represented as a rotation on the unit circle $e^{j2\pi f}$. Recall the time-window vector model from (9.6.4) consisting of a signal $\mathbf{s}(n)$, made up of complex exponentials, and the noise component $\mathbf{w}(n)$

$$\mathbf{x}(n) = \sum_{p=1}^P \alpha_p \mathbf{v}(f_p) e^{j2\pi n f_p} + \mathbf{w}(n) = \mathbf{V}\Phi^n \boldsymbol{\alpha} + \mathbf{w}(n) = \mathbf{s}(n) + \mathbf{w}(n) \quad (9.6.43)$$

where the P columns of matrix \mathbf{V} are length- M time-window frequency vectors of the

complex exponentials

$$\mathbf{V} = [\mathbf{v}(f_1) \mathbf{v}(f_2) \cdots \mathbf{v}(f_P)] \quad (9.6.44)$$

The vector $\boldsymbol{\alpha}$ consists of the amplitudes of the complex exponentials α_p . On the other hand, matrix Φ is the diagonal matrix of phase shifts between neighboring time samples of the individual, complex exponential components of $\mathbf{s}(n)$

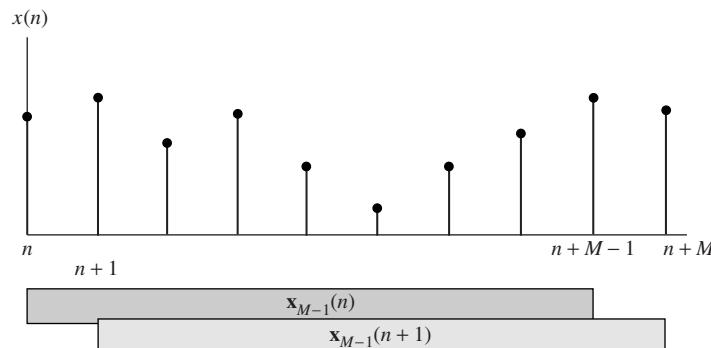
$$\Phi = \text{diag } \{\phi_1, \phi_2, \dots, \phi_P\} = \begin{bmatrix} e^{j2\pi f_1} & 0 & \cdots & 0 \\ 0 & e^{j2\pi f_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & e^{j2\pi f_P} \end{bmatrix} \quad (9.6.45)$$

where $\phi_p = e^{j2\pi f_p}$ for $p = 1, 2, \dots, P$. Since the frequencies of the complex exponentials f_p completely describe this rotation matrix, frequency estimates can be obtained by finding Φ . Let us consider two overlapping subwindows of length $M - 1$ within the length M time-window vector. This subwindowing operation is illustrated in Figure 9.25. Consider the signal consisting of the sum of complex exponentials

$$\mathbf{s}(n) = \begin{bmatrix} \mathbf{s}_{M-1}(n) \\ s(n+M-1) \\ \vdots \\ \mathbf{s}_{M-1}(n+1) \end{bmatrix} = \begin{bmatrix} s(n) \\ s(n+M-1) \\ \vdots \\ s(n+1) \end{bmatrix} \quad (9.6.46)$$

where $\mathbf{s}_{M-1}(n)$ is the length- $(M - 1)$ subwindow of $\mathbf{s}(n)$, that is,

$$\mathbf{s}_{M-1}(n) = \mathbf{V}_{M-1} \Phi^n \boldsymbol{\alpha} \quad (9.6.47)$$



Clearly, by examining (9.6.49), these two matrices of time-window frequency vectors are related as

$$\mathbf{V}_2 = \mathbf{V}_1 \Phi \quad (9.6.51)$$

Note that each of these two matrices spans a different, though related, $(M - 1)$ -dimensional subspace.

Now suppose that we have a data matrix \mathbf{X} from (9.6.20) with N data records of the length- M time-window vector signal $\mathbf{x}(n)$. Using the singular value decomposition (SVD) discussed in Chapter 8, we can write the data matrix as[†]

$$\mathbf{X} = \mathbf{L} \boldsymbol{\Sigma} \mathbf{U}^H \quad (9.6.52)$$

where \mathbf{L} is an $N \times N$ matrix of left singular vectors and \mathbf{U} is an $M \times M$ matrix of right singular vectors. Both of these matrices are unitary; that is, $\mathbf{L}^H \mathbf{L} = \mathbf{I}$ and $\mathbf{U}^H \mathbf{U} = \mathbf{I}$. The matrix $\boldsymbol{\Sigma}$ has dimensions $N \times M$ consisting of singular values on the main diagonal ordered in descending magnitude. The squared magnitudes of the singular values are equal to the eigenvalues of $\hat{\mathbf{R}}$ scaled by a factor of N from (9.6.21), and the columns of \mathbf{U} are their corresponding eigenvectors. Thus, \mathbf{U} forms an orthonormal basis for the underlying M -dimensional vector space. This subspace can be partitioned into signal and noise subspaces as

$$\mathbf{U} = [\mathbf{U}_s | \mathbf{U}_n] \quad (9.6.53)$$

where \mathbf{U}_s is the matrix of right-hand singular vectors corresponding to the singular values with the P largest magnitudes. Note that since the signal portion consists of the sum of complex exponentials modeled as time-window frequency vectors $\mathbf{v}(f)$, all these frequency vectors, for $f = f_1, f_2, \dots, f_P$, must also lie in the signal subspace. As a result, the matrices \mathbf{V} and \mathbf{U}_s span the same subspace. Therefore, there exists an invertible transformation \mathbf{T} that maps \mathbf{U}_s into \mathbf{V} , that is,

$$\mathbf{V} = \mathbf{U}_s \mathbf{T} \quad (9.6.54)$$

The transformation \mathbf{T} is never solved for in this derivation, but instead is only formulated as a mapping between these two matrices within the signal subspace.

Proceeding as we did with the matrix \mathbf{V} in (9.6.50), we can partition the signal subspace into two smaller $(M - 1)$ -dimensional subspaces as

$$\mathbf{U}_s = \begin{bmatrix} \mathbf{U}_1 \\ * * \dots * \end{bmatrix} = \begin{bmatrix} * * \dots * \\ \mathbf{U}_2 \end{bmatrix} \quad (9.6.55)$$

where \mathbf{U}_1 and \mathbf{U}_2 correspond to the unstaggered and staggered subspaces, respectively. Since \mathbf{V}_1 and \mathbf{V}_2 correspond to the same subspaces, the relation from (9.6.54) must also hold for these subspaces

$$\mathbf{V}_1 = \mathbf{U}_1 \mathbf{T} \quad \mathbf{V}_2 = \mathbf{U}_2 \mathbf{T} \quad (9.6.56)$$

The staggered and unstaggered components of the matrix \mathbf{V} in (9.6.50) are related through the subspace rotation Φ in (9.6.51). Since the matrices \mathbf{U}_1 and \mathbf{U}_2 also span these respective, related subspaces, a similar, though different, rotation must exist that relates (rotates) \mathbf{U}_1 to \mathbf{U}_2

$$\mathbf{U}_2 = \mathbf{U}_1 \Psi \quad (9.6.57)$$

where Ψ is this rotation matrix.

Recall that frequency estimation comes down to solving for the subspace rotation matrix Φ . We can estimate Φ by making use of the relations in (9.6.56) together with the

[†] Our notation differs slightly from that introduced in Chapter 8 in order to avoid confusion with the matrix of time-window frequency vectors \mathbf{V} .

rotations between the staggered signal subspaces in (9.6.51) and (9.6.57). In this process, the matrices \mathbf{U}_1 and \mathbf{U}_2 are known from the SVD on data matrix \mathbf{X} . First, we solve for Ψ from the relation in (9.6.57), using the method of least-squares (LS) from Chapter 8

$$\Psi = (\mathbf{U}_1^H \mathbf{U}_1)^{-1} \mathbf{U}_1^H \mathbf{U}_2 \quad (9.6.58)$$

Substituting (9.6.57) into (9.6.56), we have

$$\mathbf{V}_2 = \mathbf{U}_2 \mathbf{T} = \mathbf{U}_1 \Psi \mathbf{T} \quad (9.6.59)$$

Similarly, we can also solve for \mathbf{V}_2 , using the relation in (9.6.51) and substituting (9.6.56) for \mathbf{V}_1

$$\mathbf{V}_2 = \mathbf{V}_1 \Phi = \mathbf{U}_1 \mathbf{T} \Phi \quad (9.6.60)$$

Thus, equating the two right-hand sides of (9.6.59) and (9.6.60), we have the following relation between the two subspace rotations

$$\Psi \mathbf{T} = \mathbf{T} \Phi \quad (9.6.61)$$

or equivalently

$$\Psi = \mathbf{T} \Phi \mathbf{T}^{-1} \quad (9.6.62)$$

Equations (9.6.61) and (9.6.62) should be recognized as the relationship between eigenvectors and eigenvalues of the matrix Ψ (Golub and Van Loan 1996). Therefore, the diagonal elements of Φ , ϕ_p for $p = 1, 2, \dots, P$, are simply the eigenvalues of Ψ . As a result, the estimates of the frequencies are

$$\hat{f}_p = \frac{\angle \phi_p}{2\pi} \quad (9.6.63)$$

where $\angle \phi_p$ is the phase of ϕ_p . Although the principle behind the ESPRIT algorithm, namely, the use of subspace rotations, is quite simple, one can easily get lost in the details of the derivation of the algorithm. Note that we have only used simple matrix relationships. An illustrative example of the implementation of ESPRIT in MATLAB is given in Example 9.6.4 to help clarify the details of the algorithm. However, first we give a total least-squares version of the algorithm, which is the preferred method for use.

Note that the subspaces \mathbf{U}_1 and \mathbf{U}_2 are *both* only estimates of the true subspaces that correspond to \mathbf{V}_1 and \mathbf{V}_2 , respectively, obtained from the data matrix \mathbf{X} . The estimate of the subspace rotation was obtained by solving (9.6.57) using the LS criterion

$$\Psi_{\text{ls}} = (\mathbf{U}_1^H \mathbf{U}_1)^{-1} \mathbf{U}_1^H \mathbf{U}_2 \quad (9.6.64)$$

This LS solution is obtained by minimizing the errors in an LS sense from the following formulation

$$\mathbf{U}_2 + \mathbf{E}_2 = \mathbf{U}_1 \Psi \quad (9.6.65)$$

where \mathbf{E}_2 is a matrix consisting of errors between \mathbf{U}_2 and the true subspace corresponding to \mathbf{V}_2 . Note that this LS formulation assumes errors only on the estimation of \mathbf{U}_2 and no errors between \mathbf{U}_1 and the true subspace that it is attempting to estimate corresponding to \mathbf{V}_1 . Therefore, since \mathbf{U}_1 is also an estimated subspace, a more appropriate formulation is

$$\mathbf{U}_2 + \mathbf{E}_2 = (\mathbf{U}_1 + \mathbf{E}_1) \Psi \quad (9.6.66)$$

where \mathbf{E}_1 is the matrix representing the errors between \mathbf{U}_1 and the true subspace corresponding to \mathbf{V}_1 . A solution to this problem, known as *total least squares (TLS)*, is obtained by minimizing the Frobenius norm of the two error matrices

$$\|\mathbf{E}_1 \quad \mathbf{E}_2\|_F \quad (9.6.67)$$

Since the principles of TLS are beyond the scope of this book, we simply give the procedure to obtain the TLS solution of Ψ and refer the interested reader to Golub and Van Loan (1996).

First, form a matrix made up of the staggered signal subspace matrices \mathbf{U}_1 and \mathbf{U}_2 placed side by side,[†] and perform an SVD

$$[\mathbf{U}_1 \ \mathbf{U}_2] = \tilde{\mathbf{L}} \tilde{\boldsymbol{\Sigma}} \tilde{\mathbf{U}}^H \quad (9.6.68)$$

We then operate on the $2P \times 2P$ matrix $\tilde{\mathbf{U}}$ of right singular vectors. This matrix is partitioned into $P \times P$ quadrants

$$\tilde{\mathbf{U}} = \begin{bmatrix} \tilde{\mathbf{U}}_{11} & \tilde{\mathbf{U}}_{12} \\ \tilde{\mathbf{U}}_{21} & \tilde{\mathbf{U}}_{22} \end{bmatrix} \quad (9.6.69)$$

The TLS solution for the subspace rotation matrix Ψ is then

$$\Psi_{\text{tls}} = -\tilde{\mathbf{U}}_{12}\tilde{\mathbf{U}}_{22}^{-1} \quad (9.6.70)$$

The frequency estimates are then obtained from (9.6.62) and (9.6.63) by using Ψ_{tls} from (9.6.70). Although the TLS version of ESPRIT involves slightly more computations, it is generally preferred over the LS version based on formulation in (9.6.66). A statistical analysis of the performance of the ESPRIT algorithms is given in Ottersten et al. (1991).

EXAMPLE 9.6.4. In this illustrative example, we demonstrate the use of both the LS and TLS versions of the ESPRIT algorithm on a set of complex exponentials in white noise using MATLAB. First, generate a signal $s(n)$ of length $N = 128$ consisting of complex exponential signals at normalized frequencies $f = 0.1, 0.15, 0.4$, and -0.15 , all with amplitude $\alpha = 1$. Each of the complex exponentials is generated by $\exp(j*2*pi*f*[0:(N-1)]')$. The overall signal in white noise with unit power ($\sigma_w^2 = 1$) is then

$$x = s + (\text{randn}(N, 1) + j * \text{randn}(N, 1)) / \sqrt(2);$$

We form the data matrix corresponding to (9.6.20) for a time window of length $M = 8$. The least-squares ESPRIT algorithm is then performed as follows:

```
[L,S,U] = svd(X);
Us = U(:,1:P); % signal subspace
U1 = Us(1:(M-1),:); U2 = Us(2:M,:); % signal subspaces
Psi = U1\U2; % LS solution for Psi
```

If we are using the TLS version of ESPRIT, then solve for

```
[LL,SS,UU] = svd([U1 U2]);
UU12 = UU(1:P,(P+1):(2*P));
UU22 = UU((P+1):(2*P),(P+1):(2*P));
Psi = -UU12*inv(UU22); % TLS solution for Psi
```

The frequencies are found by computing the phases of the eigenvalues of Ψ , that is,

```
phi = eig(Psi); % eigenvalues of Psi
fhat = angle(diag(phi))/(2*pi); % frequency estimates
```

In both cases, we average over 1000 realizations and obtain average estimated frequencies very close to the true values $f = 0.1, 0.15, 0.4$, and -0.15 used to generate the signals. Routines for both the LS and TLS versions of ESPRIT are provided in `esprit_ls.m` and `esprit_tls.m`.

9.7 SUMMARY

In this chapter, we have examined the modeling process for both pole-zero and harmonic signal models. As for all signal modeling problems, the procedure begins with the selection of the appropriate model for the signal under consideration. Then the signal model is applied by estimating the model parameters from a collection of data samples. However, as we

[†]Note that this matrix $[\mathbf{U}_1 \ \mathbf{U}_2] \neq \mathbf{U}_s = [\mathbf{U}_1^T \ \mathbf{U}_2^T]^T$ from (9.6.55).

have stressed throughout this chapter, nothing is more valuable in the modeling process than specific knowledge of the signal and its underlying process in order to assess the validity of the model for a particular signal. For this reason, we began the chapter with a discussion of a model building procedure, starting with the choice of the appropriate model and the estimation of its parameters, and concluding with the validation of the model. Clearly, if the model is not well-suited for the signal, the application of the model becomes meaningless.

In the first part of the chapter, we considered the application of the parametric signal models that were discussed in Chapter 4. The estimation of all-pole models was presented for both direct and lattice structures. Within this context, we used various model order selection criteria to determine the order of the all-pole model. However, these criteria are not necessarily limited to all-pole models. In addition, the relationship was given between the all-pole model and Burg's method of maximum entropy. Next, we considered the pole-zero modeling. Using a nonlinear least-squares technique, a method was presented for estimating the parameters of the pole-zero model. The use of pole-zero models for the purposes of spectral estimation along with their application to speech modeling was also considered.

The latter part of the chapter focused on harmonic signal models, that is, modeling signals using the sum of complex exponentials. The harmonic modeling problem becomes one of estimating the frequency of the complex exponentials. As a bridge between these pole-zero and harmonic models, we discussed the topic of minimum-variance spectral estimation. As will be explored in the problems that follow, there are several interesting relations between the minimum-variance spectrum and the harmonic models. In addition, a relationship between the minimum-variance spectral estimator and the all-pole model was established. Then, we discuss some of the more popular harmonic modeling methods. Starting with the Pisarenko harmonic decomposition, the first such model, we discuss the MUSIC, eigenvector, root-MUSIC, and minimum-norm methods for frequency estimation. All of these methods are based on computing a pseudospectrum or a rooting polynomial from an estimated correlation matrix. Finally, we give a brief derivation of the ESPRIT algorithm, both in its original LS form and the more commonly used TLS form.

PROBLEMS

- 9.1** Consider the random process $x(n)$ described in Example 9.2.3 that is simulated by exciting the system function

$$H(z) = \frac{1}{1 - 2.7607z^{-1} + 3.8108z^{-2} - 2.6535z^{-3} + 0.9238z^{-4}}$$

using a WGN(0, 1) process. Generate $N = 250$ samples of the process $x(n)$.

- (a) Write a MATLAB function that implements the modified covariance method to obtain $\text{AR}(P)$ model coefficients and the modeling error variance $\hat{\sigma}_P^2$ as a function of P , using N samples of $x(n)$.
- (b) Compute and plot the variance $\hat{\sigma}_P^2$, FPE(P), AIC(P), MDL(P), and CAT(P) for $P = 1, 2, \dots, 15$.
- (c) Comment on your results and the usefulness of model selection criteria for the process $x(n)$.

- 9.2** Consider the Burg approach of minimizing forward-backward LS error $\mathcal{E}_m^{\text{fb}}$ in (9.2.33).

- (a) Show that by using (9.2.26) and (9.2.27), $\mathcal{E}_m^{\text{fb}}$ can be put in the form of (9.2.34).
- (b) By minimizing $\mathcal{E}_m^{\text{fb}}$ with respect to k_{m-1} , show that the expression for the optimum k_{m-1}^B is given by (9.2.35).
- (c) Show that $|k_{m-1}^B| < 1$.
- (d) Show that $|k_{m-1}^B| < |k_{m-1}^{\text{IS}}| \leq 1$ where k_{m-1}^{IS} is defined in (9.2.36).

$$H(z) = \frac{1}{1 - 0.9z^{-1} + 0.81z^{-2}}$$

excited by a WGN(0, 1) process. Illustrate numerically that if we use the full-windowing method, that is, the matrix $\tilde{\mathbf{X}}$ in (9.2.8), then the PACS estimates $\{k_m^{\text{FP}}\}_{m=0}^1$, $\{k_m^{\text{BP}}\}_{m=0}^1$, and $\{k_m^{\text{B}}\}_{m=0}^1$ of Section 9.2 are identical and hence can be obtained by using the Levinson-Durbin algorithm.

9.4 Generate sample sequences of an AR(2) process

$$x(n) = w(n) - 1.5857x(n-1) - 0.9604x(n-2)$$

where $w(n) \sim \text{WGN}(0, 1)$. Choose $N = 256$ samples for each realization.

- (a) Design a first-order optimum linear predictor, and compute the prediction error $e_1(n)$. Test the whiteness of the error sequence $e_1(n)$ using the autocorrelation, PSD, and partial correlation methods, discussed in Section 9.1. Show your results as an overlay plot using 20 realizations.
- (b) Repeat the above part, using second- and third-order linear predictors.
- (c) Comment on your plots.

9.5 Generate sample functions of the process

$$x(n) = 0.5w(n) + 0.5w(n-1)$$

where $w(n) \sim \text{WGN}(0, 1)$. Choose $N = 256$ samples for each realization.

- (a) Test the whiteness of $x(n)$ and show your results, using overlay plots based on 10 realizations.
- (b) Process $x(n)$ through the AR(1) filter

$$H(z) = \frac{1}{1 + 0.95z^{-1}}$$

to obtain $y(n)$. Test the whiteness of $y(n)$ and show your results, using overlay plots based on 10 realizations.

9.6 The process $x(n)$ contains a complex exponential in white noise, that is,

$$x(n) = Ae^{j(\omega_0 n + \theta)} + w(n)$$

where A is a real positive constant, θ is a random variable uniformly distributed over $[0, 2\pi]$, ω_0 is a constant between 0 and π , and $w(n) \sim \text{WGN}(0, \sigma_w^2)$. The purpose of this problem is to analytically obtain a maximum entropy method (MEM) estimate by fitting an AR(P) model and then evaluating $\{a_k\}_0^P$ model coefficients.

- (a) Show that the $(P+1) \times (P+1)$ autocorrelation matrix of $x(n)$ is given by

$$\mathbf{R}_x = A^2 \mathbf{e} \mathbf{e}^H + \sigma_w^2 \mathbf{I}$$

where $\mathbf{e} = [1 \ e^{-j\omega_0} \ \dots \ e^{-jP\omega_0}]^T$.

- (b) By solving autocorrelation normal equations, show that

$$\begin{aligned} \mathbf{a}_P &\triangleq [1 \ a_1 \ \dots \ a_P]^T \\ &= \left(1 + \frac{A^2}{\sigma_w^2 + A^2 P} \right) \left[\mathbf{e} - \frac{A^2}{\sigma_w^2 + (P+1)A^2} [1 \ 0 \ \dots \ 0]^T \right] \end{aligned}$$

- (c) Show that the MEM estimate based on the above coefficients is given by

$$\hat{R}_x(e^{j\omega}) = \frac{\sigma_w^2 \left[1 - \frac{A^2}{\sigma_w^2 + (P+1)A^2} \right]}{\left| 1 - \frac{A^2}{\sigma_w^2 + (P+1)A^2} W_R(e^{j(\omega-\omega_0)}) \right|^2}$$

where $W_R(e^{j\omega})$ is the DTFT of the $(P+1)$ length rectangular window.

9.7 An AR(2) process $y(n)$ is observed in noise $v(n)$ to obtain $x(n)$, that is,

$$x(n) = y(n) + v(n) \quad v(n) \sim \text{WGN}(0, \sigma_v^2)$$

where $v(n)$ is uncorrelated with $y(n)$ and

$$y(n) = 1.27y(n-1) - 0.81y(n-2) + w(n) \quad w(n) \sim \text{WGN}(0, 1)$$

- (a) Determine and plot the true power spectrum $R_x(e^{j\omega})$.
- (b) Generate 10 realizations of $x(n)$, each with $N = 256$ samples. Using the LS approach with forward-backward linear predictor, estimate the power spectrum for $P = 2$ and $\sigma_v^2 = 1$. Obtain an overlay plot of this estimate, and compare it with the true spectrum.
- (c) Repeat part (b), using $\sigma_v^2 = 10$. Comment on the effect of increasing noise variance on spectrum estimates.
- (d) Since the noise variance σ_v^2 affects only $r_x(0)$, investigate the effect of subtracting a small amount from $r_x(0)$ on the spectrum estimates in part (c).

9.8 Let $x(n)$ be a random process whose correlation is estimated. The values for the first five lags are $r_x(0) = 1$, $r_x(1) = 0.7$, $r_x(2) = 0.5$, $r_x(3) = 0.3$, and $r_x(4) = 0$.

- (a) Determine and plot the Blackman-Tukey power spectrum estimate.
- (b) Assume that $x(n)$ is modeled by an AP(2) model. Determine and plot its spectrum estimate.
- (c) Now repeat (b) assuming that AP(4) is an appropriate model for $x(n)$. Determine and plot the spectrum estimate.

9.9 The narrowband process $x(n)$ is generated using the AP(4) model

$$H(z) = \frac{1}{1 + 0.98z^{-1} + 1.92z^{-2} + 0.94z^{-3} + 0.92z^{-4}}$$

driven by $\text{WGN}(0, 0.001)$. Generate 10 realizations, each with $N = 256$ samples, of this process.

- (a) Determine and plot the true power spectrum $R_x(e^{j\omega})$.
- (b) Using the LS approach with forward linear predictor, estimate the power spectrum for $P = 4$. Obtain an overlay plot of this estimate, and compare it with the true spectrum.
- (c) Repeat part (b) with $P = 8$ and 12. Provide a qualitative description of your results with respect to model order size.
- (d) Using the LS approach with forward-backward linear predictor, estimate the power spectrum for $P = 4$. Obtain an overlay plot of this estimate. Compare it with the plot in part (b).

9.10 Consider the following PZ(4, 2) model

$$H(z) = \frac{1 - z^{-2}}{1 + 0.41z^{-4}}$$

driven by $\text{WGN}(0, 1)$ to obtain a broadband ARMA process $x(n)$. Generate 10 realizations, each with $N = 256$ samples, of this process.

- (a) Determine and plot the true power spectrum $R_x(e^{j\omega})$.
- (b) Using the LS approach with forward-backward linear predictor, estimate the power spectrum for $P = 12$. Obtain an overlay plot of this estimate, and compare it with the true spectrum.
- (c) Using the nonlinear LS pole-zero modeling algorithm of Section 9.3.3, estimate the power spectrum for $P = 4$ and $Q = 2$. Obtain an overlay plot of this estimate, and compare it with the plot in part (b).

9.11 A random process $x(n)$ is given by

$$x(n) = \cos\left(\frac{\pi n}{3} + \theta_1\right) + w(n) - w(n-2) + \cos\left(\frac{2\pi n}{3} + \theta_2\right)$$

where $w(n) \sim \text{WGN}(0, 1)$ and θ_1 and θ_2 are IID random variables uniformly distributed between 0 and 2π . Generate a sample sequence with $N = 256$ samples.

- (a) Determine and plot the true spectrum $R_x(e^{j\omega})$.
- (b) Using the LS approach with forward-backward linear predictor, estimate the power spectrum for $P = 10, 20$, and 40 from the generated sample sequence. Compare it with the true spectrum.

- (c) Using the nonlinear LS pole-zero modeling algorithm of Section 9.3.3, estimate the power spectrum for $P = 4$ and $Q = 2$. Compare it with the true spectrum and with the plot in part (b).

9.12 Show that, for large values of N , the modeling error variance estimate given by Equation (9.2.38) can be approximated by the estimate given by Equation (9.2.39).

9.13 This problem investigates the effect of correlation aliasing observed in LS estimation of model parameters when the AP model is excited by discrete spectra. Consider an AP(1) model with pole at $z = \alpha$ excited by a periodic sequence of period N . Let $x(n)$ be the output sequence.

- (a) Show that the correlation at lag 1 satisfies

$$r_x(1) = \frac{\alpha^{N-1} + \alpha}{1 + \alpha^N} r_x(0) \quad (\text{P.1})$$

- (b) Using the LS approach, determine the estimate $\hat{\alpha}$ as a function of α and N . Compute $\hat{\alpha}$ for $\alpha = 0.9$ and $N = 10$.
- (c) Generate $x(n)$, using $\alpha = 0.95$ and the periodic impulse train with $N = 10$. Compute and plot the correlation sequence $r_x(l)$, $0 \leq l \leq N - 1$, of $x(n)$. Compare your plot with the AP(1) model correlation for $\alpha = 0.95$. Comment on your observations and discuss why they explain the discrepancy between α and $\hat{\alpha}$.
- (d) Repeat part (c) for $N = 100$ and 1000 . Show analytically and numerically that $\hat{\alpha} \rightarrow \alpha$ as $N \rightarrow \infty$.

9.14 In this problem, we investigate the equation error method of Section 9.3.1. Consider the PZ(2, 2) model

$$x(n) = 0.3x(n - 1) + 0.4x(n - 2) + w(n) + 0.25w(n - 2)$$

Generate $N = 200$ samples of $x(n)$, using $w(n) \sim \text{WGN}(0, \sqrt{10})$. Record values of both $x(n)$ and $w(n)$.

- (a) Using the residual windowing method, that is, $N_i = \max(P, Q)$ and $N_f = N - 1$, compute the estimates of the above model parameters.
- (b) Compute the input variance estimate $\hat{\sigma}_w^2$ from your estimated values in part (a). Compare it with the actual value σ_w^2 and with (9.3.12).

9.15 Consider the following PZ(4, 2) model

$$\begin{aligned} x(n) = & 1.8766x(n - 1) - 2.6192x(n - 2) + 1.6936x(n - 3) - 0.8145x(n - 4) \\ & + w(n) + 0.05w(n - 1) - 0.855w(n - 2) \end{aligned}$$

excited by $w(n) \sim \text{WGN}(0, \sqrt{10})$. Generate 300 samples of $x(n)$.

- (a) Using the nonlinear LS pole-zero modeling algorithm of Section 9.3.3, estimate the parameters of the above model from the $x(n)$ data segment.
- (b) Assuming the AP(10) model for the data segment, estimate its parameters by using the LS approach described in Section 9.2.
- (c) Generate a plot similar to Figure 9.13 by computing spectra corresponding to the true PZ(4, 2), estimated PZ(4, 2), and estimated AP(10) models. Compare and comment on your results.

9.16 Using matrix notation, show that AZ power spectrum estimation is equivalent to the Blackman-Tukey method discussed in Chapter 5.

9.17 Consider the PZ(4, 2) model given in Problem 9.15. Generate 300 samples of $x(n)$.

- (a) Fit an AP(5) model to the data and plot the resulting spectrum.
- (b) Fit an AP(10) model to the data and plot the resulting spectrum.
- (c) Fit an AP(50) model to the data and plot the resulting spectrum.
- (d) Compare your plots with the true spectrum, and discuss the effect of model mismatch on the quality of the spectrum.

- 9.18** Use the supplied (about 50-ms) segment of a speech signal sampled at 8192 samples per second.
- Compute a periodogram of the speech signal (see Chapter 5).
 - Using data windowing, fit an AP(16) model to the speech data and compute the spectrum.
 - Using the residual windowing, fit a PZ(12, 6) model to the speech data and compute the spectrum.
 - Plot the above three spectra on one graph, and comment on the performance of each method.
- 9.19** One practical approach to spectrum estimation discussed in Section 9.4 is the prewhitening and postcoloring method.
- Develop a MATLAB function to implement this method. Use the forward/backward LS method to determine AP(P) parameters and the Welch method for nonparametric spectrum estimation.
 - Verify your function on the short segment of the speech segment from Problem 9.18.
 - Compare your results with those obtained in Problem 9.18.
- 9.20** Consider a white noise process with variance σ_w^2 . Find its minimum-variance power spectral estimate.
- 9.21** Find the minimum-variance spectrum of a first-order all pole model, that is,
- $$x(n) = -a_1 x(n-1) + w(n)$$
- 9.22** The filter coefficient vector for the minimum-variance spectrum estimator is given in (9.5.10). Using Lagrange multipliers, discussed in Appendix B, solve this constrained optimization to find this weight vector.
- 9.23** Using the relationship between the minimum-variance and the all-pole model spectrum estimators in (9.5.22), generate a recursive relationship for the minimum-variance spectrum estimators of increasing window length. In other words, write $\hat{R}_{M+1}^{(\text{mv})}(e^{j2\pi f})$ in terms of $\hat{R}_M^{(\text{mv})}(e^{j2\pi f})$ and the all-pole model spectrum estimator $\hat{R}_M^{(\text{ap})}(e^{j2\pi f})$ in (9.5.20).
- 9.24** In Pisarenko harmonic decomposition, discussed in Section 9.6.2, we determine the frequencies of the complex exponentials in white noise through the use of the pseudospectrum. The word *pseudospectrum* was used because its value does not correspond to an estimated power. Find a set of linear equations that can be solved to find the powers of the complex exponentials. *Hint:* Use the relationship of eigenvalues and eigenvectors $\mathbf{R}_x \mathbf{q}_m = \lambda_m \mathbf{q}_m$ for $m = 1, 2, \dots, M$.
- 9.25** For the MUSIC algorithm, we showed a means of using the MUSIC pseudospectrum to derive a polynomial that could be rooted to obtain frequency estimates, which is known as root-MUSIC. Find a similar rooting method for the minimum-norm frequency estimation procedure.
- 9.26** The Pisarenko harmonic decomposition, MUSIC, and minimum-norm algorithms yield frequency estimates by computing a pseudospectrum using the Fourier transforms of the eigenvectors. However, these pseudospectra do not actually estimate a power. Derive the minimum-variance spectral estimator in terms of the Fourier transforms of the eigenvectors and the associated eigenvalues. Relate this result to the MUSIC and eigenvector method pseudospectra.
- 9.27** Show that the pseudospectrum for the MUSIC algorithm is equivalent to the minimum-variance spectrum in the case of an infinite signal-to-noise ratio.
- 9.28** Find a relationship between the minimum-norm pseudospectrum and the all-pole model spectrum in the case of an infinite signal-to-noise ratio.
- 9.29** In (9.5.22), we derived a relationship between the minimum-variance spectral estimator and spectrum estimators derived from all-pole models of orders 1 to M . Find a similar relationship between the pseudospectra of the MUSIC and minimum-norm algorithms that shows that the MUSIC pseudospectrum is a weighted average of minimum-norm pseudospectra.

Adaptive Filters

In Chapter 1, we discussed different practical applications that demonstrated the need for adaptive filters, pointed out the key aspects of the underlying *signal operating environment* (SOE), and illustrated the key features and types of adaptive filters. The defining characteristic of an adaptive filter is its ability to operate satisfactorily, according to a criterion of performance acceptable to the user, in an unknown and possibly time-varying environment without the intervention of the designer. In Chapter 6, we developed the theory of optimum filters under the assumption that the filter designer has complete knowledge of the statistical properties (usually second-order moments) of the SOE. However, in real-world applications such information is seldom available, and the most practical solution is to use an adaptive filter. Adaptive filters can improve their performance, during normal operation, by learning the statistical characteristics through processing current signal observations.

In this chapter, we develop a mathematical framework for the design and performance evaluation of adaptive filters, both theoretically and by simulation. The goal of an adaptive filter is to “find and track” the optimum filter corresponding to the same signal operating environment with complete knowledge of the required statistics. In this context, optimum filters provide both guidance for the development of adaptive algorithms and a yardstick for evaluating the theoretical performance of adaptive filters. We start in Section 10.1 with discussion of a few typical application problems that can be effectively solved by using an adaptive filter. The performance of adaptive filters is evaluated using the concepts of stability, speed of adaptation, quality of adaptation, and tracking capabilities. These issues and the key features of an adaptive filter are discussed in Section 10.2. Since most adaptive algorithms originate from deterministic optimization methods, in Section 10.3 we introduce the family of steepest-descent algorithms and study their properties. Sections 10.4 and 10.5 provide a detailed discussion of the derivation, properties, and applications of the two most important adaptive filtering algorithms: the least mean square (LMS) and the recursive least-squares (RLS) algorithms. The conventional RLS algorithm, introduced in Section 10.5, can be used for either array processing (multiple-sensor or general input data vector) applications or FIR filtering (single-sensor or shift-invariant input data vector) applications. Section 10.6 deals with different implementations of the RLS algorithm for array processing applications, whereas Section 10.7 provides fast implementations of the RLS algorithm for the FIR filtering case. The development of the later algorithms is a result of the shift invariance of the data stored in the memory of the FIR filter. Finally, in Section 10.8 we provide a concise introduction to the tracking properties of the LMS and the RLS algorithms.

10.1 TYPICAL APPLICATIONS OF ADAPTIVE FILTERS

As we have already seen in Chapter 1, many practical applications cannot be successfully solved by using fixed digital filters because either we do not have sufficient information to design a digital filter with fixed coefficients or the design criteria change during the normal operation of the filter. Most of these applications can be successfully solved by using a special type of “smart” filters known collectively as *adaptive filters*. The distinguishing feature of adaptive filters is that they can modify their response to improve their performance during operation without any intervention from the user.

The best way to introduce adaptive filters is with some applications for which they are well suited. These and other applications are discussed in greater detail in the sequel as we develop the necessary background and tools.

10.1.1 Echo Cancellation in Communications

An echo is the delayed and distorted version of an original signal that returns to its source. In some applications (radar, sonar, or ultrasound), the echo is the wanted signal; however, in communication applications, the echo is an unwanted signal that must be eliminated. There are two types of echoes in communication systems: (1) *electrical or line echoes*, which are generated electrically due to impedance mismatches at points along the transmission medium, and (2) *acoustic echoes*, which result from the reflection of sound waves and acoustic coupling between a microphone and a loudspeaker.

Here we focus on electrical echoes in voice communications; electrical echoes in data communications are discussed in Section 10.4.4, and acoustic echoes in teleconferencing and hands-free telephony were discussed in Section 1.4.1.

Electrical echoes are observed on long-distance telephone circuits. A simplified form of such a circuit, which is sufficient for the present discussion, is shown in Figure 10.1. The local links from the customer to the telephone office consist of bidirectional two-wire connections, whereas the connection between the telephone offices is a four-wire carrier facility that may include a satellite link. The conversion between two-wire and four-wire links is done by special devices known as *hybrids*. An ideal hybrid should pass (1) the incoming signal to the two-wire output without any leakage into its output port and (2) the signal from the two-wire circuit to its output port without reflecting any energy back to the two-wire line (Sondhi and Berkley 1980). In practice, due to impedance mismatches, the hybrids do not operate perfectly. As a result, some energy on the incoming branch of the four-wire circuit leaks into the outgoing branch and returns to the source as an echo (see Figure 10.1). This echo, which is usually 11 dB down from the original signal, makes it difficult to carry on a conversation if the round-trip delay is larger than 40 ms. Satellite links, as a consequence of high altitude, involve round-trip delays of 500 to 600 ms.

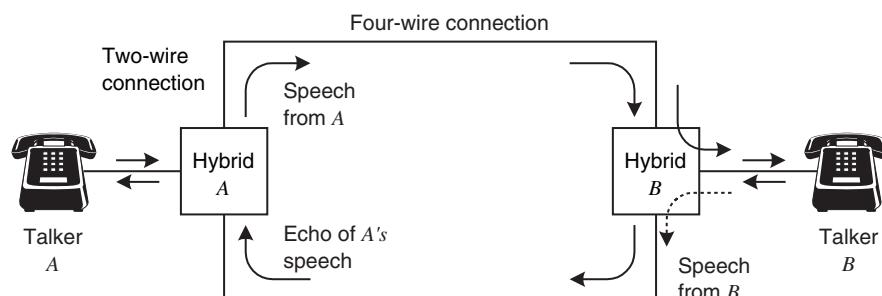


FIGURE 10.1

Echo generation in a long-distance telephone network.

The first devices used by telephone companies to control voice echoes were echo suppressors. Basically, an *echo suppressor* is a voice-activated switch that attempts to impose an open circuit on the return path from listener to talker when the listener is silent (see Figure 10.2). The main problems with these devices are speech clipping during double-talking and the inability to effectively deal with round-trip delays longer than 100 ms (Weinstein 1977).

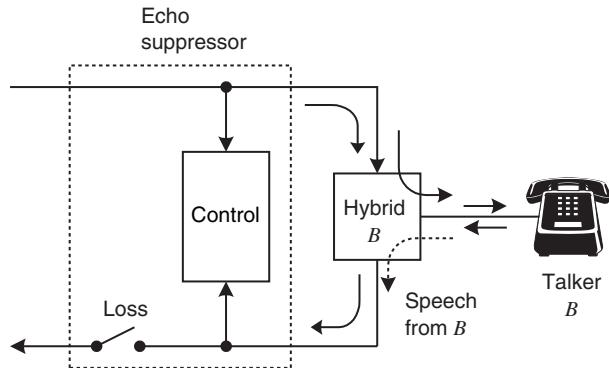


FIGURE 10.2

Principle of echo suppression.

The problems associated with echo suppressors could be largely avoided if we could estimate the *transmission path* from point C to point D (see Figure 10.3), which is known as the *echo path*. If we knew the echo path, we could design a filter that produced a copy or replica of the echo signal when driven by the signal at point C. Subtraction of the echo replica from the signal at point D will eliminate the echo without distorting the speech of the second talker that may be present at point D. The resulting device, shown in Figure 10.3, is known as an *echo canceler*.

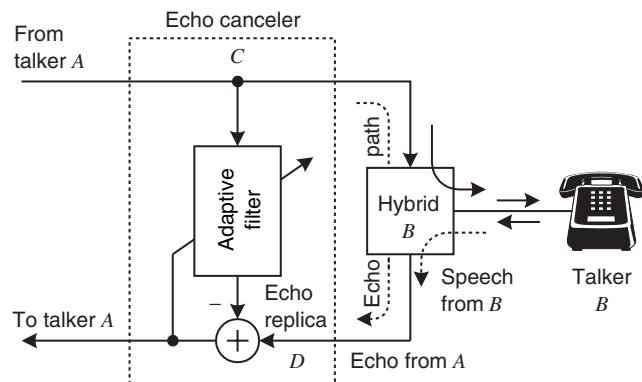


FIGURE 10.3

Principle of echo cancellation.

In practice, the channel characteristics are generally *not* known. For dial-up telephone lines, the channel differs from call to call, and the characteristics of radio and microwave channels (phase perturbations, fading, etc.) change significantly with time. Therefore, we *cannot* design and use a *fixed* echo canceler with satisfactory performance for all possible connections. There are two possible ways around this problem:

1. Design a compromise *fixed echo canceler* based on some “average” echo path, assuming that we have sufficient information about the connections to be seen by the canceler.
2. Design an *adaptive echo canceler* that can “learn” the echo path when it is first turned on and afterward “tracks” its variations without any intervention from the designer. Since an adaptive canceler matches the echo path for any given connection, it performs better than a compromise fixed canceler.

We stress that the main task of the canceler is to estimate the echo signal with sufficient accuracy; the estimation of the echo path is simply the means of achieving this goal. The performance of the canceler is measured by the attenuation, in decibels, of the echo, which is known as *echo return loss enhancement*. The adaptive echo canceler achieves this goal by modifying its response, using the residual echo signal in an as yet unspecified way.

Adaptive echo cancelers are widely used in voice telecommunications, and the international standards organization CCITT has issued a set of recommendations (CCITT G.165) that outlines the basic requirements for echo cancelers. More details can be found in Weinstein (1977) and Murano et al. (1990).

10.1.2 Equalization of Data Communication Channels

Channel equalization, which is probably the most widely employed technique in practical data transmission systems, was first introduced in Section 1.4.1. In Section 6.8 we discussed the design of symbol rate zero-forcing and optimum MSE equalizers. As we recall, every pulse propagating through the channel suffers a certain amount of *time dispersion* because the frequency response of the channel deviates from the ideal one of constant magnitude and linear phase. Some typical sources of dispersion for practical communication channels are summarized in Table 10.1. As a result, the tails of adjacent pulses interfere with the measurement of the current pulse (intersymbol interference) and can lead to an incorrect decision.

TABLE 10.1
Summary of causes of dispersion in various communications systems.

Transmission system	Causes of dispersion
Cable TV	Transmitter filtering; coaxial-cable dispersion; cable amplifiers; reflections from impedance mismatches; bandpass filters
Microwave radio	Transmitter filtering; reflections from impedance mismatches; multipath propagation; scattering; input bandpass filtering
Voiceband modems	Digital-to-analog image suppression; channel filtering; twisted-pair transmission line; multiplexing and demultiplexing filters; hybrids; antialias lowpass filters
Troposcatter radio	Transmitter filtering; atmospheric dispersion; scattering at interface between troposphere and stratosphere; receiver bandpass filtering; input amplifiers

Source: From Treichler et al. 1996.

Since the channel can be modeled as a linear system, assuming that the receiver and the transmitter do not include any nonlinear operations, we can compensate for its distortion by using a linear equalizer. The goal of the equalizer is to restore the received pulse, as closely as possible, to its original shape. The equalizer transforms the channel to a near-ideal one if its response resembles the *inverse* of the channel. Since the channel is unknown and possibly time-varying, there are two ways to approach the problem: (1) Design a compromise fixed equalizer to obtain satisfactory performance over a broad range of channels, or (2) design an equalizer that can learn the inverse of the particular channel and then track its variation in real time.

The characteristics of the equalizer are adjusted by some algorithm that attempts to attain the best possible performance. The most appropriate criterion of performance for data transmission systems is the probability of error. However, it cannot be used for two reasons: (1) the “correct” symbol is unknown to the receiver (otherwise there would be no reason to communicate), and (2) the number of decisions needed to estimate the low probabilities of error is extremely large. Thus, practical equalizers assess their performance by using some function of the difference between the correct symbol and their output. The operation of practical equalizers involves three modes of operation, dependent on how we substitute for the unavailable correct symbol sequence.

Training mode: A known *training sequence* is transmitted, and the equalizer attempts to improve its performance by comparing its output to a synchronized replica of the training sequence stored at the receiver. Usually this mode is used when the equalizer starts a transmission session.

Decision-directed mode: At the end of the training session, when the equalizer starts making reliable decisions, we can replace the training sequence with the equalizer’s own decisions.

“Blind” or self-recovering mode: There are several applications in which the use of a training sequence is not desired or feasible. This may occur in multipoint networks for computer communications or in wideband digital systems over coaxial facilities during rerouting (Godard 1980; Sato 1975). Also when the decision-directed mode of a microwave channel equalizer fails, after deep fades, we do not have a reverse channel to call for retraining (Foschini 1985). In such cases, where the equalizer should be able to learn or recover the characteristics of the channel without the benefit of a training sequence, we say that the equalizer operates in blind or self-recovering mode.

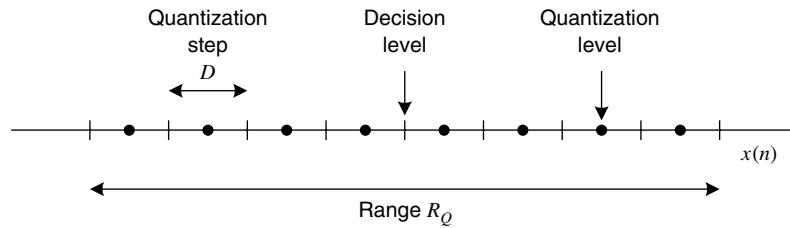
Adaptive equalization is a mature technology that has had the greatest impact on digital communications systems, including voiceband, microwave and troposcatter radio, and cable TV modems (Qureshi 1985; Lee and Messerschmitt 1994; Gitlin et al. 1992; Bingham 1988; Treichler et al. 1996, 1998).

10.1.3 Linear Predictive Coding

The efficient storage and transmission of analog signals using digital systems requires the minimization of the number of bits necessary to represent the signal while maintaining the quality to an acceptable level according to a certain criterion of performance. The conversion of an analog (continuous-time, continuous-amplitude) signal to a digital (discrete-time, discrete-amplitude) signal involves two processes: sampling and quantization. Sampling converts a continuous-time signal to a discrete-time signal by measuring its amplitude at equidistant intervals of time. Quantization involves the representation of the measured continuous amplitude by using a finite number of symbols. Therefore, a small range of amplitudes will use the same symbol (see Figure 10.4). A code word is assigned to each symbol by the coder. When the digital representation is used for digital signal processing, the quantization levels and the corresponding code words are uniformly distributed. However, for coding applications, levels may be nonuniformly distributed to match the distribution of the signal amplitudes.

For all practical purposes, the range of a quantizer is equal to $R_Q = \Delta \cdot 2^B$, where Δ is the quantization step size and B is the number of bits, and should cover the dynamic range of the signal. The difference between the unquantized sample $x(n)$ and the quantized sample $\hat{x}(n)$, that is,

$$e(n) \triangleq \hat{x}(n) - x(n) \quad (10.1.1)$$

**FIGURE 10.4**

Partitioning of the range of a 3-bit (eight-level) uniform quantizer.

is known as the *quantization error* and is always in the range $-\Delta/2 \leq e(n) \leq \Delta/2$. If we define the signal-to-noise ratio by

$$\text{SNR} \triangleq \frac{E\{x^2(n)\}}{E\{e^2(n)\}} \quad (10.1.2)$$

it can be shown (Rabiner and Schafer 1978; Jayant and Noll 1984) that

$$\text{SNR(dB)} \simeq 6B \quad (10.1.3)$$

which states that each added binary digit increases the SNR by 6 dB.

For a fixed number of bits, decreasing the dynamic range of the signal (and therefore the range of the quantizer) decreases the required quantization step and therefore the average quantization error power. Therefore, we can increase the SNR by reducing the dynamic range, or equivalently the variance of the signal. If the signal samples are significantly correlated, the variance of the difference between adjacent samples is smaller than the variance of the original signal. Thus, we can improve the SNR by quantizing this difference instead of the original signal.

The differential quantization concept is exploited by the *linear predictive coding (LPC)* system illustrated in Figure 10.5. The quantized signal is the difference

$$d(n) = x(n) - \tilde{x}(n) \quad (10.1.4)$$

where $\tilde{x}(n)$ is an estimate or prediction of the signal $x(n)$ obtained by the predictor using a quantized version

$$\hat{x}(n) = \tilde{x}(n) + \hat{d}(n) \quad (10.1.5)$$

of the original signal (see Figure 10.5). If the quantization error of the difference signal is

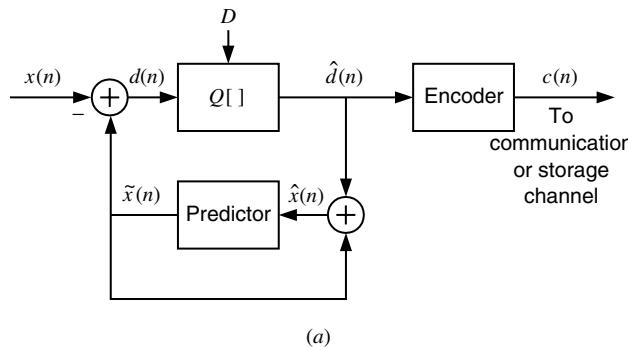
$$e_d(n) = \hat{d}(n) - d(n) \quad (10.1.6)$$

we obtain

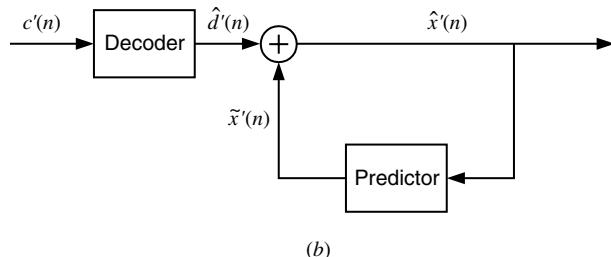
$$\hat{x}(n) = x(n) + e_d(n) \quad (10.1.7)$$

using (10.1.4) and (10.1.5). The significance of (10.1.7) is that the quantization error of the original signal is equal to the quantization error of the difference signal, independently of the properties of the predictor. Note that if $c'(n) = c(n)$, that is, there are no transmission or storage errors, then the signal reconstructed by the decoder is $\hat{x}'(n) = \hat{x}(n)$. If the prediction is good, the dynamic range of $d(n)$ should be smaller than the dynamic range of $x(n)$, resulting in a smaller quantization noise for the same number of bits or the same quantization noise with a smaller number of bits. The performance of the LPC system depends on the accuracy of the predictor. In most practical applications, we use a linear predictor that forms an estimate (prediction) $\tilde{x}(n)$ of the present sample $x(n)$ as a linear combination of the M past samples, that is,

$$\tilde{x}(n) = \sum_{k=1}^M a_k \hat{x}(n-k) \quad (10.1.8)$$



(a)



(b)

FIGURE 10.5
 Block diagram of a linear predictive coding system: (a) coder and (b) decoder.

The coefficients $\{a_k\}_1^M$ of the linear predictor are determined by exploiting the correlation between adjacent samples of the input signal with the objective to make the prediction error as small as possible. Since the statistical properties of the signal $x(n)$ are unknown and change with time, we *cannot* design an optimum fixed predictor. The established practical solution uses an *adaptive* linear predictor that automatically adjusts its coefficients to compute a “good” prediction at each time instant. A detailed discussion of adaptive linear prediction and its application to audio, speech, and video signal coding is provided in Jayant and Noll (1984).

10.1.4 Noise Cancelation

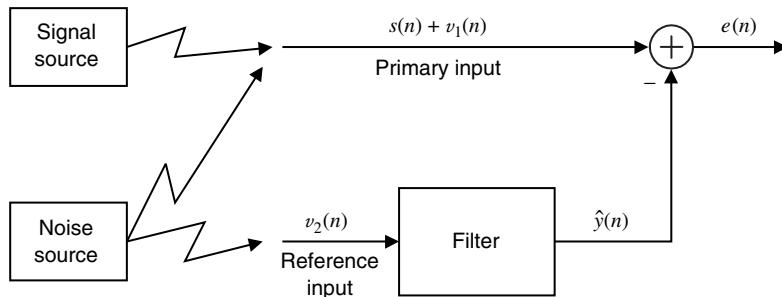
In Section 1.4.1 we discussed the concept of active noise control using adaptive filters. We now provide a theoretical explanation for the general problem of noise canceling using multiple sensors. The principle of general noise cancelation is illustrated in Figure 10.6. The signal of interest $s(n)$ is corrupted by uncorrelated additive noise $v_1(n)$, and the combined signal $s(n) + v_1(n)$ provides what is known as *primary input*. A second sensor, located at a different point, acquires a noise $v_2(n)$ (*reference input*) that is uncorrelated with the signal $s(n)$ but correlated with the noise $v_1(n)$. If we can design a filter that provides a good estimate $\hat{y}(n)$ of the noise $v_1(n)$, by exploiting the correlation between $v_1(n)$ and $v_2(n)$, then we could recover the desired signal by subtracting $\hat{y}(n) \approx v_1(n)$ from the primary input.

Let us assume that the signals $s(n)$, $v_1(n)$, and $v_2(n)$ are jointly wide-sense stationary with zero mean values. The “clean” signal is given by the error

$$e(n) = s(n) + [v_1(n) - \hat{y}(n)]$$

where $\hat{y}(n)$ depends on the filter structure and parameters. The MSE is given by

$$E\{|e(n)|^2\} = E\{|s(n)|^2\} + E\{|v_1(n) - \hat{y}(n)|^2\}$$

**FIGURE 10.6**

Principle of adaptive noise cancelation using a reference input.

because the signals $s(n)$ and $v_1(n) - \hat{y}(n)$ are uncorrelated. Since the signal power is not influenced by the filter, if we design a filter that minimizes the total output power $E\{|e(n)|^2\}$, then that filter will minimize the output noise power $E\{|v_1(n) - \hat{y}(n)|^2\}$. Therefore, $\hat{y}(n)$ will be the MMSE estimate of the noise $v_1(n)$, and the canceler maximizes the output signal-to-noise ratio. If we know the second-order moments of the primary and reference inputs, we can design an optimum linear canceler using the techniques discussed in Chapter 6. However, in practice, the design of an optimum canceler is not feasible because the required statistical moments are either unknown or time-varying. Once again, a successful solution can be obtained by using an adaptive filter that automatically adjusts its parameters to obtain the best possible estimate of the interfering noise (Widrow et al. 1975).

10.2 PRINCIPLES OF ADAPTIVE FILTERS

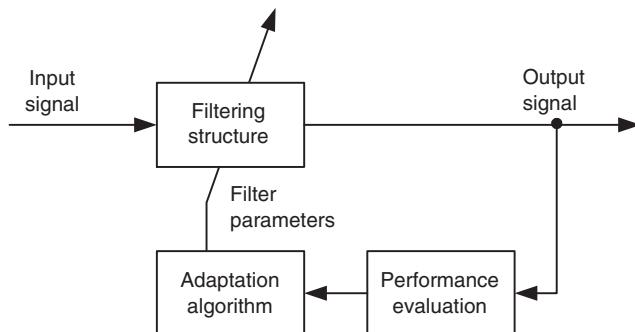
In this section, we discuss a mathematical framework for the analysis and performance evaluation of adaptive algorithms. The goal is to develop design guidelines for the application of adaptive algorithms to practical problems. The need for adaptive filters and representative applications that can benefit from their use have been discussed in Sections 1.4.1 and 10.1.

10.2.1 Features of Adaptive Filters

The applications we have discussed are only a sample from a multitude of practical problems that can be successfully solved by using adaptive filters, that is, filters that automatically change their characteristics to attain the right response at the right time. Every adaptive filtering application involves one or more input signals and a desired response signal that may or may not be accessible to the adaptive filter. We collectively refer to these signals as the *signal operating environment (SOE)* of the adaptive filter. Every adaptive filter consists of three modules (see Figure 10.7):

Filtering structure. This module forms the output of the filter using measurements of the input signal or signals. The filtering structure is *linear* if the output is obtained as a linear combination of the input measurements; otherwise it is said to be *nonlinear*. For example, the filtering module can be an adjustable finite impulse response (FIR) digital filter implemented with a direct or lattice structure or a recursive filter implemented using a cascade structure. The structure is fixed by the designer, and its parameters are adjusted by the adaptive algorithm.

Criterion of performance (COP). The output of the adaptive filter and the desired response (when available) are processed by the COP module to assess its quality with respect to the requirements of the particular application. The choice of the

**FIGURE 10.7**

Basic elements of a general adaptive filter.

criterion is a balanced compromise between what is acceptable to the user of the application and what is mathematically tractable; that is, it can be manipulated to derive an adaptive algorithm. Most adaptive filters use some average form of the square error because it is mathematically tractable and leads to the design of useful practical systems.

Adaptation algorithm. The adaptive algorithm uses the value of the criterion of performance, or some function of it, and the measurements of the input and desired response (when available) to decide how to modify the parameters of the filter to improve its performance. The complexity and the characteristics of the adaptive algorithm are functions of the filtering structure and the criterion of performance.

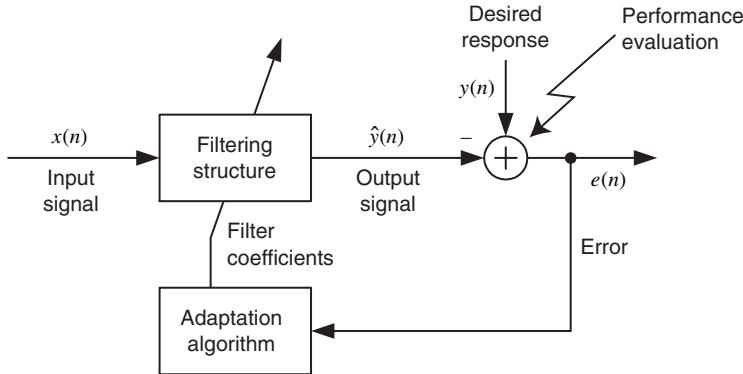
The design of any adaptive filter requires some generic *a priori* information about the SOE and a deep understanding of the particular application. This information is needed by the designer to choose the criterion of performance and the filtering structure. Clearly, unreliable *a priori* information and/or incorrect assumptions about the SOE can lead to serious performance degradations or even unsuccessful adaptive filter applications. The conversion of the performance assessment to a successful parameter adjustment strategy, that is, the design of an adaptive algorithm, is the most difficult step in the design and application of adaptive filters.

If the characteristics of the SOE are constant, the goal of the adaptive filter is to find the parameters that give the best performance and then stop the adjustment. The initial period, from the time the filter starts its operation until the time it gets reasonably close to its best performance, is known as the *acquisition* or *convergence mode*. However, when the characteristics of the SOE change with time, the adaptive filter should first find and then continuously readjust its parameters to track these changes. In this case, the filter starts with an acquisition phase that is followed by a *tracking mode*.

A very influential factor in the design of adaptive algorithms is the availability of a desired response signal. We have seen that for certain applications, the desired response may not be available for use by the adaptive filter. Therefore, the adaptation must be performed in one of two ways:

Supervised adaptation. At each time instant, the adaptive filter knows in advance the desired response, computes the error (i.e., the difference between the desired and actual response), evaluates the criterion of performance, and uses it to adjust its coefficients. In this case, the structure in Figure 10.7 is simplified to that of Figure 10.8.

Unsupervised adaptation. When the desired response is unavailable, the adaptive filter cannot explicitly form and use the error to improve its behavior. In some applications, the input signal has some measurable property (i.e., constant envelope) that is lost by the time it reaches the adaptive filter. The adaptive filter adjusts its parameters in such a way as to restore the lost property of the input signal. The *property restoral*

**FIGURE 10.8**

Basic elements of a supervised adaptive filter.

approach to adaptive filtering was introduced in Treichler et al. (1987). In some other applications (e.g., digital communications) the basic task of the adaptive filter is to classify each received pulse to one of a finite set of symbols. In this case we basically have a problem of unsupervised classification (Fukunaga 1990).

In this chapter we focus our discussion on supervised adaptive filters, that is, filters that have access to a desired response signal; unsupervised adaptive filters, which operate without the benefit of a desired response, are discussed in Section 12.3, in the context of blind equalization.

10.2.2 Optimum versus Adaptive Filters

We have mentioned several times that the theory of stochastic processes provides the mathematical framework for the design and analysis of optimum filters. In Chapter 6, we introduced filters that are optimum according to the MSE criterion of performance; and in Chapter 7, we developed algorithms and structures for their efficient design and implementation. However, optimum filters are a theoretical tool and cannot be used in practical applications because we do not know the statistical quantities (e.g., second-order moments) that are required for their design. Adaptive filters can be thought as the practical counterpart of optimum filters: They try to reach the performance of optimum filters by processing measurements of the SOE in real time, which makes up for the lack of a priori statistics.

For this analysis, we consider the general case of a linear combiner that includes filtering and prediction as special cases. However, for convenience we use the terms *filters* and *filtering*. We remind the reader that, from a mathematical point of view, the key difference between a linear combiner and an FIR filter or predictor is the shift invariance (temporal ordering) of the input data vector. This difference, which is illustrated in Figure 10.9, also has important implications in the implementation of adaptive filters. To this end, suppose that the SOE is comprised of M input signals $x_k(n, \zeta)$ and a desired response signal $y(n, \zeta)$, which are sample realizations of random sequences.[†]

Then the estimate of $y(n, \zeta)$ is computed by using the linear combiner

$$\hat{y}(n, \zeta) = \sum_{k=1}^M c_k^*(n) x_k(n, \zeta) \triangleq \mathbf{c}^H(n) \mathbf{x}(n, \zeta) \quad (10.2.1)$$

where

$$\mathbf{c}(n) = [c_1(n) \ c_2(n) \ \cdots \ c_M(n)]^T \quad (10.2.2)$$

[†]For clarity, in this section only, we include the dependence on ζ to denote random variables.

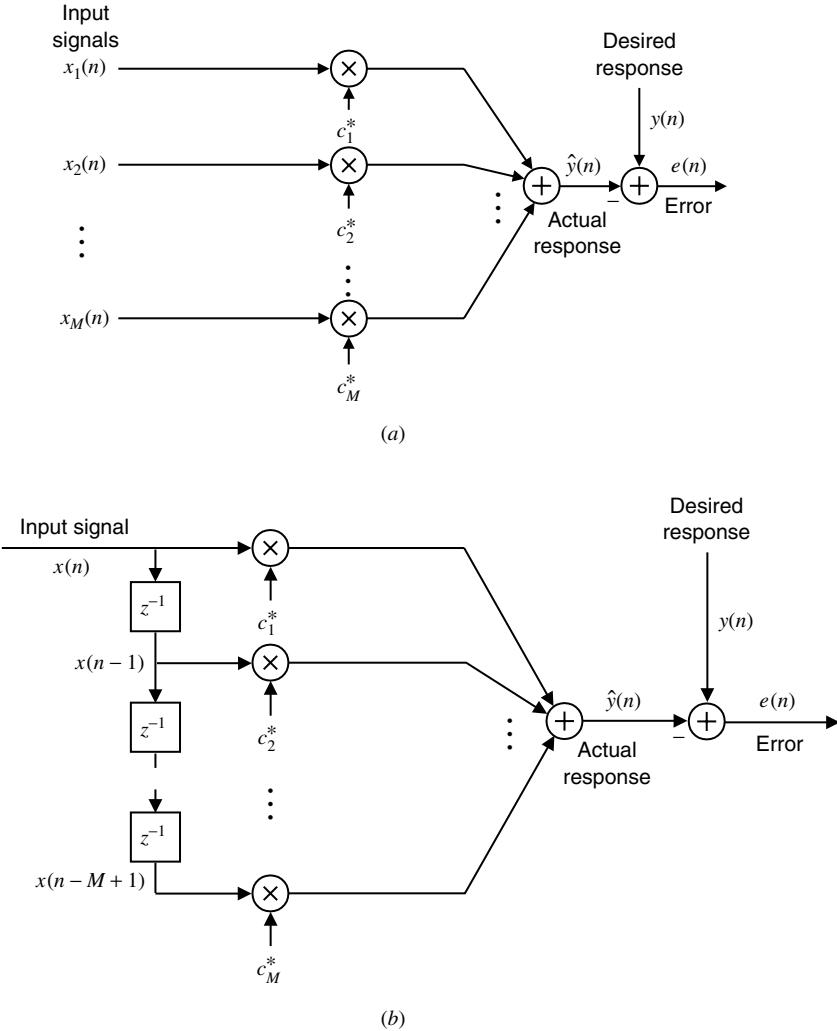
**FIGURE 10.9**

Illustration of the difference of the input signal between (a) a multiple-input linear combiner and (b) a single-input FIR filter.

is the coefficient vector and

$$\mathbf{x}(n, \zeta) = [x_1(n, \zeta) \ x_2(n, \zeta) \ \cdots \ x_M(n, \zeta)]^T \quad (10.2.3)$$

is the input data vector. For single-sensor applications, the input data vector is shift-invariant

$$\mathbf{x}(n) = [x(n, \zeta) \ x(n-1, \zeta) \ \cdots \ x(n-M+1, \zeta)]^T \quad (10.2.4)$$

and the linear combiner takes the form of the FIR filter

$$\hat{y}(n, \zeta) = \sum_{k=0}^{M-1} h(n, k)x(n-k, \zeta) \triangleq \mathbf{c}^H(n)\mathbf{x}(n, \zeta) \quad (10.2.5)$$

where $c_k(n) = h^*(n, k)$ are the samples of the impulse response at time n .

Optimum filters. If we know the second-order moments of the SOE, we can design an optimum filter $\mathbf{c}_o(n)$ by solving the normal equations

$$\mathbf{R}(n)\mathbf{c}_o(n) = \mathbf{d}(n) \quad (10.2.6)$$

where

$$\mathbf{R}(n) = E\{\mathbf{x}(n, \zeta)\mathbf{x}^H(n, \zeta)\} \quad (10.2.7)$$

and

$$\mathbf{d}(n) = E\{\mathbf{x}(n, \zeta)y^*(n, \zeta)\} \quad (10.2.8)$$

are the correlation matrix of the input data vector and the cross-correlation between the input data vector and the desired response, respectively. During its normal operation, the optimum filter works with specific realizations of the SOE, that is,

$$\hat{y}_o(n, \zeta) = \mathbf{c}_o^H(n)\mathbf{x}(n, \zeta) \quad (10.2.9)$$

$$\varepsilon_o(n, \zeta) = y(n, \zeta) - \hat{y}_o(n, \zeta) \quad (10.2.10)$$

where $\hat{y}_o(n, \zeta)$ is the optimum estimate and $\varepsilon_o(n, \zeta)$ is the optimum instantaneous error [see Figure 10.10(a)]. However, the filter is optimized with respect to its average performance across all possible realizations of the SOE, and the MMSE

$$P_o(n) = E\{|\varepsilon_o(n, \zeta)|^2\} = P_y(n) - \mathbf{d}^H(n)\mathbf{c}_o(n) \quad (10.2.11)$$

shows how well the filter performs on average. Also, we emphasize that the optimum coefficient vector is a nonrandom quantity and that the desired response is not essential for the operation of the optimum filter [see Equation (10.2.9)].

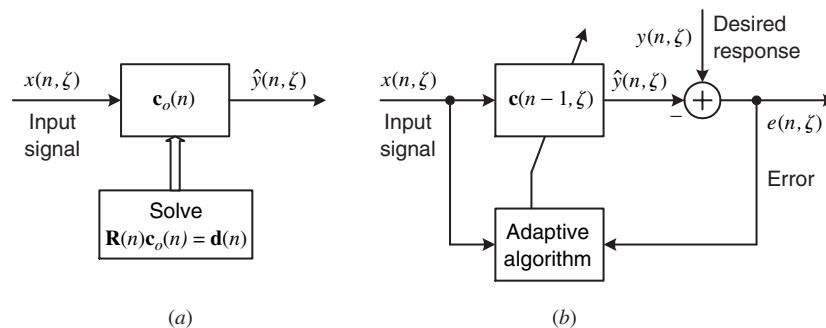


FIGURE 10.10

Illustration of the difference in operation between (a) optimum filters and (b) adaptive filters.

If the SOE is stationary, the optimum filter is computed once and is used with *all* realizations $\{\mathbf{x}(n, \zeta), y(n, \zeta)\}$. For nonstationary environments, the optimum filter design is repeated at every time instant n because the optimum filter is time-varying.

Adaptive filters. In most practical applications, where the second-order moments $\mathbf{R}(n)$ and $\mathbf{d}(n)$ are *unknown*, the use of an adaptive filter is the best solution. If the SOE is ergodic, we have

$$\mathbf{R} = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N \mathbf{x}(n, \zeta)\mathbf{x}^H(n, \zeta) \quad (10.2.12)$$

$$\mathbf{d} = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N \mathbf{x}(n, \zeta)y^*(n, \zeta) \quad (10.2.13)$$

because ensemble averages are equal to time averages (see Section 3.3). If we collect a sufficient amount of data $\{\mathbf{x}(n, \zeta), y(n, \zeta)\}_{n=0}^{N-1}$, we can obtain an acceptable estimate of the optimum filter by computing the estimates

$$\hat{\mathbf{R}}_N(\zeta) = \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{x}(n, \zeta)\mathbf{x}^H(n, \zeta) \quad (10.2.14)$$

$$\hat{\mathbf{d}}_N(\zeta) = \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{x}(n, \zeta) y^*(n, \zeta) \quad (10.2.15)$$

by time-averaging and then solving the linear system

$$\hat{\mathbf{R}}_N(\zeta) \mathbf{c}_N(\zeta) = \hat{\mathbf{d}}_N(\zeta) \quad (10.2.16)$$

The obtained coefficients can be used to filter the data in the interval $0 \leq n \leq N - 1$ or to start filtering the data for $n \geq N$, on a sample-by-sample basis, in real time. This procedure, which we called *block adaptive filtering* in Chapter 8, should be repeated each time the properties of the SOE change significantly. Clearly, block adaptive filters cannot track statistical variations within the operating block and cannot be used in all applications.

Indeed, there are applications, for example, adaptive equalization, in which each input sample should be processed immediately after its observation and before the arrival of the next sample. In such cases, we should use a sample-by-sample adaptive filter that starts filtering immediately after the observation of the pair $\{\mathbf{x}(0), y(0)\}$ using a “guess” $\mathbf{c}(-1)$ for the adaptive filter coefficients. Usually, the initial guess $\mathbf{c}(-1)$ is a very poor estimate of the optimum filter \mathbf{c}_o . However, this estimate is improved with time as the filter processes additional pairs of observations.

As we discussed in Section 10.2.1, an adaptive filter consists of three key modules: an adjustable filtering structure that uses input samples to compute the output, the criterion of performance that monitors the performance of the filter, and the adaptive algorithm that updates the filter coefficients. The key component of any adaptive filter is the *adaptive algorithm*, which is a rule to determine the filter coefficients from the available data $\mathbf{x}(n, \zeta)$ and $y(n, \zeta)$ [see Figure 10.10(b)]. The dependence of $\mathbf{c}(n, \zeta)$ on the input signal makes the adaptive filter a nonlinear and time-varying stochastic system.

The data available to the adaptive filter at time n are the input data vector $\mathbf{x}(n, \zeta)$, the desired response $y(n, \zeta)$, and the most recent update $\mathbf{c}(n - 1, \zeta)$ of the coefficient vector. The adaptive filter, at each time n , performs the following computations:

1. *Filtering*:

$$\hat{y}(n, \zeta) = \mathbf{c}^H(n - 1, \zeta) \mathbf{x}(n, \zeta) \quad (10.2.17)$$

2. *Error formation*:

$$e(n, \zeta) = y(n, \zeta) - \hat{y}(n, \zeta) \quad (10.2.18)$$

3. *Adaptive algorithm*:

$$\mathbf{c}(n, \zeta) = \mathbf{c}(n - 1, \zeta) + \Delta \mathbf{c}\{\mathbf{x}(n, \zeta), e(n, \zeta)\} \quad (10.2.19)$$

where the increment or correction term $\Delta \mathbf{c}(n, \zeta)$ is chosen to bring $\mathbf{c}(n, \zeta)$ close to \mathbf{c}_o , with the passage of time. If we can successively determine the corrections $\Delta \mathbf{c}(n, \zeta)$ so that $\mathbf{c}(n, \zeta) \simeq \mathbf{c}_o$, that is, $\|\mathbf{c}(n, \zeta) - \mathbf{c}_o\| < \delta$, for some $n > N_\delta$, we obtain a good approximation for \mathbf{c}_o by avoiding the explicit averagings (10.2.14), (10.2.15), and the solution of the normal equations (10.2.16). A key requirement is that $\Delta \mathbf{c}(n, \zeta)$ must vanish if the error $e(n, \zeta)$ vanishes. Hence, $e(n, \zeta)$ plays a major role in determining the increment $\Delta \mathbf{c}(n, \zeta)$.

We notice that the estimate $\hat{y}(n, \zeta)$ of the desired response $y(n, \zeta)$ is evaluated using the *current* input vector $\mathbf{x}(n, \zeta)$ and the *past* filter coefficients $\mathbf{c}(n - 1, \zeta)$. The estimate $\hat{y}(n, \zeta)$ and the corresponding error $e(n, \zeta)$ can be considered as *predicted* estimates compared to the *actual* estimates that would be evaluated using the *current* coefficient vector $\mathbf{c}(n, \zeta)$. Coefficient updating methods that use the predicted error $e(n, \zeta)$ are known as *a priori type adaptive algorithms*.

If we use the *actual* estimates, obtained using the current estimate $\mathbf{c}(n, \zeta)$ of the adaptive filter coefficients, we have

1. *Filtering*:

$$\hat{y}_a(n, \zeta) = \mathbf{c}^H(n, \zeta) \mathbf{x}(n, \zeta) \quad (10.2.20)$$

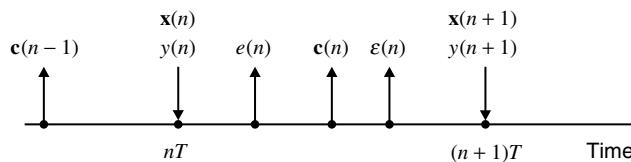
2. Error formation:

$$\varepsilon(n, \zeta) = y(n, \zeta) - \hat{y}_a(n, \zeta) \quad (10.2.21)$$

3. Adaptive algorithm:

$$\mathbf{c}(n, \zeta) = \mathbf{c}(n-1, \zeta) + \Delta \mathbf{c}\{\mathbf{x}(n, \zeta), \varepsilon(n, \zeta)\} \quad (10.2.22)$$

which are known as *a posteriori* type adaptive algorithms. The terms *apriori* and *a posteriori* were introduced in Carayannis et al. (1983) to emphasize the use of estimates evaluated before or after the updating of the filter coefficients. The difference between a priori and a posteriori errors and their meanings will be further clarified when we discuss adaptive least-squares filters in Section 10.5. The timing diagram for the above two algorithms is shown in Figure 10.11.

**FIGURE 10.11**

Timing diagrams for a priori and a posteriori adaptive algorithms.

In conclusion, the objective of an adaptive filter is to use the available data at time n , namely, $\{\mathbf{x}(n, \zeta), y(n, \zeta), \mathbf{c}(n-1, \zeta)\}$, to update the “old” coefficient vector $\mathbf{c}(n-1, \zeta)$ to a “new” estimate $\mathbf{c}(n, \zeta)$ so that $\mathbf{c}(n, \zeta)$ is closer to the optimum filter vector $\mathbf{c}_o(n)$ and the output $\hat{y}(n)$ is a better estimate of the desired response $y(n)$. Most adaptive algorithms have the following form:

$$\begin{bmatrix} \text{New} \\ \text{coefficient} \\ \text{vector} \end{bmatrix} = \begin{bmatrix} \text{old} \\ \text{coefficient} \\ \text{vector} \end{bmatrix} + \begin{bmatrix} \text{adaptation} \\ \text{gain} \\ \text{vector} \end{bmatrix} \cdot \begin{pmatrix} \text{error} \\ \text{signal} \end{pmatrix} \quad (10.2.23)$$

where the error signal is the difference between the desired response and the predicted or actual outputs of the adaptive filter. One of the fundamental differences among the various algorithms is the optimality of the used adaptation gain vector and the amount of computation required for its evaluation.

10.2.3 Stability and Steady-State Performance of Adaptive Filters

We now address the issues of stability and performance of adaptive filters. Since the goal of an adaptive filter $\mathbf{c}(n, \zeta)$ is *first to find and then track* the optimum filter $\mathbf{c}_o(n)$ as *quickly and accurately* as possible, we can evaluate its performance by measuring some function of its deviation

$$\tilde{\mathbf{c}}(n, \zeta) \triangleq \mathbf{c}(n, \zeta) - \mathbf{c}_o(n) \quad (10.2.24)$$

from the corresponding optimum filter. Clearly, an acceptable adaptive filter should be stable in the bounded-input bounded-output (BIBO) sense, and its performance should be close to that of the associated optimum filter. The analysis of BIBO stability is extremely difficult because adaptive filters are nonlinear, time-varying systems working in a random SOE. The performance of adaptive filters is primarily measured by investigating the value of the MSE as a function of time. To discuss these problems, first we consider an adaptive filter working in a stationary SOE, and then we extend our discussion to a nonstationary SOE.

Stability

The adaptive filter starts its operation at time, say, $n = 0$, and by processing the observations $\{x(n, \zeta), y(n, \zeta)\}_0^\infty$ generates a sequence of vectors $\{\mathbf{c}(n, \zeta)\}_0^\infty$ using the adaptive algorithm. Since the FIR filtering structure is always stable, the output or the error of the adaptive filter will be bounded if its coefficients are always kept close to the coefficients of the associated optimum filter. However, the presence of the feedback loop in every adaptive filter (see Figure 10.10) raises the issue of stability. In a stationary SOE, where the optimum filter \mathbf{c}_o is constant, convergence of $\mathbf{c}(n, \zeta)$ to \mathbf{c}_o as $n \rightarrow \infty$ will guarantee the BIBO stability of the adaptive filter. For a specific realization ζ , the k th component $c_k(n, \zeta)$ or the norm $\|\mathbf{c}(n, \zeta)\|$ of the vector $\mathbf{c}(n, \zeta)$ is a sequence of numbers that might or might not converge.[†] Since the coefficients $c_k(n, \zeta)$ are random, we must use the concept of *stochastic convergence* (Papoulis 1991).

We say that a random sequence converges *everywhere* if the sequence $c_k(n, \zeta)$ converges for every ζ , that is,

$$\lim_{n \rightarrow \infty} c_k(n, \zeta) = c_{o,k}(\zeta) \quad (10.2.25)$$

where the limit $c_{o,k}(\zeta)$ depends, in general, on ζ . Requiring the adaptive filter to converge to \mathbf{c}_o for every possible realization of the SOE is both hard to guarantee and not necessary, because some realizations may have very small or zero probability of occurrence.

If we wish to ensure that the adaptive filter converges for the realizations of the SOE that may actually occur, we can use the concept of convergence almost everywhere. We say that the random sequence $c_k(n, \zeta)$ converges *almost everywhere* or *with probability 1* if

$$P\left\{\lim_{n \rightarrow \infty} |c_k(n, \zeta) - c_{o,k}(\zeta)| = 0\right\} = 1 \quad (10.2.26)$$

which implies that there can be some sample sequences that do not converge, which must occur with probability zero.

Another type of stochastic convergence that is used in adaptive filtering is defined by

$$\lim_{n \rightarrow \infty} E\{|c_k(n, \zeta) - c_{o,k}(\zeta)|^2\} = \lim_{n \rightarrow \infty} E\{|\tilde{c}_k(n, \zeta)|^2\} = 0 \quad (10.2.27)$$

and is known as *convergence in the MS sense*. The primary reason for the use of mean square (MS) convergence is that unlike the almost-everywhere convergence, it uses only one sequence of numbers that takes into account the averaging effect of all sample sequences. Furthermore, it uses second-order moments for verification and has an interpretation in terms of power. Convergence in MS does not imply—nor is implied by—convergence with probability 1. Since

$$\frac{E\{|\tilde{c}_k(n, \zeta)|^2\}}{\delta} = \frac{|E\{\tilde{c}_k(n, \zeta)\}|^2}{\delta} + \frac{\text{var}\{\tilde{c}_k(n, \zeta)\}}{\delta^2} \quad (10.2.28)$$

if we can show that $E\{\tilde{c}_k(n)\} \rightarrow 0$ as $n \rightarrow \infty$ and $\text{var}\{\tilde{c}_k(n, \zeta)\}$ is bounded for all n , we can ensure convergence in MS. In this case, we can say that an adaptive filter that operates in a stationary SOE is an asymptotically stable filter.

Performance measures

In theoretical investigations, any quantity that measures the deviation of an adaptive filter from the corresponding optimum filter can be used to evaluate its performance.

The *mean square deviation (MSD)*

$$\mathcal{D}(n) \triangleq E\{\|\mathbf{c}(n, \zeta) - \mathbf{c}_o(n)\|^2\} = E\{\|\tilde{\mathbf{c}}(n, \zeta)\|^2\} \quad (10.2.29)$$

[†]We recall that a sequence of real nonrandom numbers a_0, a_1, a_2, \dots converges to a number a if and only if for every positive number δ there exists a positive integer N_δ such that for all $n > N_\delta$, we have $|a_n - a| < \delta$. This is abbreviated by $\lim_{n \rightarrow \infty} a_n = a$.

measures the average distance between the coefficient vectors of the adaptive and optimum filters. Although the MSD is not measurable in practice, it is useful in analytical studies. Adaptive algorithms that minimize $\mathcal{D}(n)$ for each value of n are known as algorithms with *optimum learning*.

In Section 6.2.2 we showed that if the input correlation matrix is positive definite, any deviation, say, $\tilde{\mathbf{c}}(n)$, of the optimum filter coefficients from their optimum setting increases the mean square error (MSE) by an amount equal to $\tilde{\mathbf{c}}^H(n)\mathbf{R}\tilde{\mathbf{c}}(n)$, known as *excess MSE (EMSE)*. In adaptive filters, the random deviation $\tilde{\mathbf{c}}(n, \zeta)$ from the optimum results in an EMSE, which is measured by the ensemble average of $\tilde{\mathbf{c}}^H(n, \zeta)\mathbf{R}\tilde{\mathbf{c}}(n, \zeta)$. For a posteriori adaptive filters, the MSE can be decomposed as

$$P'(n) \triangleq E\{|\varepsilon(n, \zeta)|^2\} \triangleq P'_o(n) + P'_{\text{ex}}(n) \quad (10.2.30)$$

where $P'_{\text{ex}}(n)$ is the EMSE and $P'_o(n)$ is the MMSE given by

$$P'_o(n) \triangleq E\{|\varepsilon_o(n, \zeta)|^2\} \quad (10.2.31)$$

with

$$\varepsilon_o(n, \zeta) \triangleq y(n, \zeta) - \mathbf{c}_o^H(n)\mathbf{x}(n, \zeta) \quad (10.2.32)$$

as the a posteriori optimum filtering error. Clearly, the a posteriori EMSE $P'_{\text{ex}}(n)$ is given by

$$P'_{\text{ex}}(n) \triangleq P'(n) - P'_o(n) \quad (10.2.33)$$

For a priori adaptive algorithms, where we use the “old” coefficient vector $\mathbf{c}(n-1, \zeta)$, it is more appropriate to use the a priori EMSE given by

$$P_{\text{ex}}(n) \triangleq P(n) - P_o(n) \quad (10.2.34)$$

where

$$P(n) \triangleq E\{|e(n, \zeta)|^2\} \quad (10.2.35)$$

and

$$P_o(n) \triangleq E\{|e_o(n, \zeta)|^2\} \quad (10.2.36)$$

with

$$e_o(n, \zeta) \triangleq y(n, \zeta) - \mathbf{c}_o^H(n-1)\mathbf{x}(n, \zeta) \quad (10.2.37)$$

as the a priori optimum filtering error. If the SOE is stationary, we have $\varepsilon_o(n, \zeta) = e_o(n, \zeta)$; that is, the optimum a priori and a posteriori errors are identical.

The dimensionless ratio

$$\mathcal{M}(n) \triangleq \frac{P_{\text{ex}}(n)}{P_o(n)} \quad \text{or} \quad \mathcal{M}'(n) \triangleq \frac{P'_{\text{ex}}(n)}{P'(n)} \quad (10.2.38)$$

known as *misadjustment*, is a useful measure of the quality of adaptation. Since the EMSE is always positive, *there is no adaptive filter that can perform (on the average) better than the corresponding optimum filter*. In this sense, we can say that the excess MSE or the misadjustment measures the cost of adaptation.

Acquisition and tracking

Plots of the MSD, MSE, or $\mathcal{M}(n)$ as a function of n , which are known as *learning curves*, characterize the performance of an adaptive filter and are widely used in theoretical and experimental studies. When the adaptive filter starts its operation, its coefficients provide a poor estimate of the optimum filter and the MSD or the MSE is very large. As the number of observations processed by the adaptive filter increases with time, we expect the quality of the estimate $\mathbf{c}(n, \zeta)$ to improve, and therefore the MSD and the MSE to decrease. The property of an adaptive filter to bring the coefficient vector $\mathbf{c}(n, \zeta)$ close to the optimum filter \mathbf{c}_o , independently of the initial condition $\mathbf{c}(-1)$ and the statistical properties of the SOE, is called *acquisition*. During the acquisition phase, we say that the adaptive filter is in a transient mode of operation.

A natural requirement for any adaptive algorithm is that adaptation stops after the algorithm has found the optimum filter \mathbf{c}_o . However, owing to the randomness of the SOE

and the finite amount of data used by the adaptive filter, its coefficients continuously fluctuate about their optimum settings, that is, about the coefficients of the optimum filter, in a random manner. As a result, the adaptive filter reaches a steady-state mode of operation, after a certain time, and its performance stops improving.

The transient and steady-state modes of operation in a stationary SOE are illustrated in Figure 10.12(a). The duration of the acquisition phase characterizes the *speed of adaptation* or *rate of convergence* of the adaptive filter, whereas the steady-state EMSE or misadjustment characterizes the *quality of adaptation*. These properties depend on the SOE, the filtering structure, and the adaptive algorithm.

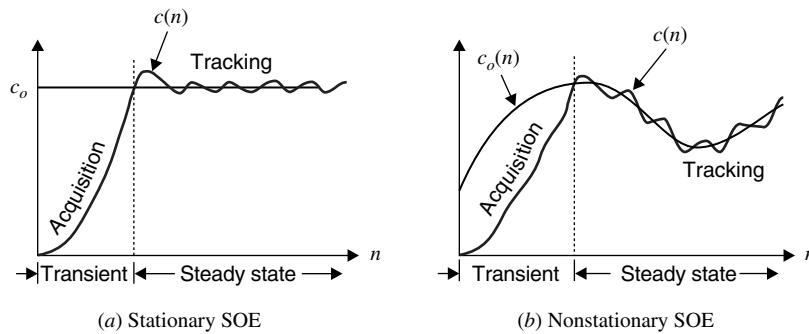


FIGURE 10.12
Modes of operation in a stationary and nonstationary SOE.

At each time n , any adaptive filter computes an estimate of the optimum filter using a finite amount of data. The error resulting from the finite amount of data is known as *estimation error*. An additional error, known as the *lag error*, results when the adaptive filter attempts to track a time-varying optimum filter $\mathbf{c}_o(n)$ in a nonstationary SOE. The modes of operation of an adaptive filter in a nonstationary SOE are illustrated in Figure 10.12(b). The SOE of the adaptive filter becomes nonstationary if $\mathbf{x}(n, \zeta)$ or $\mathbf{y}(n, \zeta)$ or both are nonstationary. The nonstationarity of the input is more severe than that of the desired response because it may affect the invertibility of $\mathbf{R}(n)$. Since the adaptive filter has to first acquire and then track the optimum filter, tracking is a steady-state property. Therefore, in general, the speed of adaptation (a transient-phase property) and the tracking capability (a steady-state property) are two different characteristics of the adaptive filter. Clearly, tracking is feasible only if the statistics of the SOE change “slowly” compared to the speed of tracking of the adaptive filter. These concepts will become more precise in Section 10.8, where we discuss the tracking properties of adaptive filters.

10.2.4 Some Practical Considerations

The complexity of the hardware or software implementation of an adaptive filter is basically determined by the following factors: (1) the number of instructions per time update or computing time required to complete one time updating; (2) the number of memory locations required to store the data and the program instructions; (3) the structure of information flow in the algorithm, which is very important for implementations using parallel processing, systolic arrays, or VLSI chips; and (4) the investment in hardware design tools and software development. We focus on implementations for general-purpose computers or special-purpose digital signal processors that basically involve programming in a high level or assembly language. More details about DSP software development can be found in Embree and Kimble (1991) and in Lapsley et al. (1997).

The digital implementation of adaptive filters implies the use of finite-word-length arithmetic. As a result, the performance of the practical (finite-precision) adaptive filters deviates from the performance of ideal (infinite-precision) adaptive filters. Finite-precision implementation affects the performance of adaptive filters in several complicated ways. The major factors are (1) the quantization of the input signal(s) and the desired response, (2) the quantization of filter coefficients, and (3) the roundoff error in the arithmetic operations used to implement the adaptive filter. The nonlinear nature of adaptive filters coupled with the nonlinearities introduced by the finite-word-length arithmetic makes the performance evaluation of practical adaptive filters extremely difficult. Although theoretical analysis provides insight and helps to clarify the behavior of adaptive filters, the most effective way is to simulate the filter and measure its performance.

Finite precision affects two important properties of adaptive filters, which, although related, are not equivalent. Let us denote by $\mathbf{c}_{ip}(n)$ and $\mathbf{c}_f(n)$ the coefficient vectors of the filter implemented using infinite- and finite-precision arithmetic, respectively. An adaptive filter is said to be *numerically stable* if the difference vector $\mathbf{c}_{ip}(n) - \mathbf{c}_f(n)$ remains always bounded, that is, the roundoff error propagation system is stable. Numerical stability is an inherent property of the adaptive algorithm and cannot be altered by increasing the numerical precision. Indeed, increasing the word length or reorganizing the computations will simply delay the divergence of an adaptive filter; only actual change of the algorithm can stabilize an adaptive filter by improving the properties of the roundoff error propagation system (Ljung and Ljung 1985; Cioffi 1987).

The *numerical accuracy* of an adaptive filter measures the deviation, at steady state, of any obtained estimates from theoretically expected values, due to roundoff errors. Numerical accuracy results in an increase of the output error without catastrophic problems and can be reduced by increasing the word length. In contrast, lack of numerical stability leads to catastrophic overflow (divergence or blowup of the algorithm) as a result of roundoff error accumulation. Numerically unstable algorithms converging before “explosion” may provide good numerical accuracy. Therefore, although the two properties are related, one does not imply the other.

Two other important issues are the sensitivity of an algorithm to bad or abnormal input data (e.g., poorly exciting input) and its sensitivity to initialization. All these issues are very important for the application of adaptive algorithms to real-world problems and are further discussed in the context of specific algorithms.

10.3 METHOD OF STEEPEST DESCENT

Most adaptive filtering algorithms are obtained by simple modifications of iterative methods for solving deterministic optimization problems. Studying these techniques helps one to understand several aspects of the operation of adaptive filters. In this section we discuss gradient-based optimization methods because they provide the ground for the development of the most widely used adaptive filtering algorithms.

As we discussed in Section 6.2.1, the error performance surface of an optimum filter, in a stationary SOE, is given by

$$P(\mathbf{c}) = P_y - \mathbf{c}^H \mathbf{d} - \mathbf{d}^H \mathbf{c} + \mathbf{c}^H \mathbf{R} \mathbf{c} \quad (10.3.1)$$

where $P_y = E\{|y(n)|^2\}$. Equation (10.3.1) is a quadratic function of the coefficients and represents a bowl-shaped surface (when \mathbf{R} is positive definite) and has a unique minimum at \mathbf{c}_o (optimum filter). There are two distinct ways to find the minimum of (10.3.1):

1. Solve the normal equations $\mathbf{R}\mathbf{c} = \mathbf{d}$, using a direct linear system solution method.
2. Find the minimum of $P(\mathbf{c})$, using an iterative minimization algorithm.

Although direct methods provide the solution in a finite number of steps, sometimes we prefer iterative methods because they require less numerical precision, are computationally less expensive, work when \mathbf{R} is not invertible, and are the only choice for nonquadratic performance functions.

In all iterative methods, we start with an approximate solution (a guess), which we keep changing until we reach the minimum. Thus, to find the optimum \mathbf{c}_o , we start at some arbitrary point \mathbf{c}_0 , usually the null vector $\mathbf{c}_0 = \mathbf{0}$, and then start a search for the “bottom of the bowl.” The key is to choose the steps in a systematic way so that each step takes us to a lower point until finally we reach the bottom. What differentiates various optimization algorithms is how we choose the direction and the size of each step.

Steepest-descent algorithm (SDA)

If the function $P(\mathbf{c})$ has continuous derivatives, it is possible to approximate its value at an arbitrary neighboring point $\mathbf{c} + \Delta\mathbf{c}$ by using the Taylor expansion

$$P(\mathbf{c} + \Delta\mathbf{c}) = P(\mathbf{c}) + \sum_{i=1}^M \frac{\partial P(\mathbf{c})}{\partial c_i} \Delta c_i + \frac{1}{2} \sum_{i=1}^M \sum_{j=1}^M \Delta c_i \frac{\partial^2 P(\mathbf{c})}{\partial c_i \partial c_j} \Delta c_j + \dots \quad (10.3.2)$$

or more compactly

$$P(\mathbf{c} + \Delta\mathbf{c}) = P(\mathbf{c}) + (\Delta\mathbf{c})^T \nabla P(\mathbf{c}) + \frac{1}{2} (\Delta\mathbf{c})^T [\nabla^2 P(\mathbf{c})] (\Delta\mathbf{c}) + \dots \quad (10.3.3)$$

where $\nabla P(\mathbf{c})$ is the gradient vector, with elements $\partial P(\mathbf{c})/\partial c_i$, and $\nabla^2 P(\mathbf{c})$ is the Hessian matrix, with elements $\partial^2 P(\mathbf{c})/(\partial c_i \partial c_j)$. For simplicity we consider filters with real coefficients, but the conclusions apply when the coefficients are complex. For the quadratic function (10.3.1), we have

$$\nabla P(\mathbf{c}) = 2(\mathbf{R}\mathbf{c} - \mathbf{d}) \quad (10.3.4)$$

$$\nabla^2 P(\mathbf{c}) = 2\mathbf{R} \quad (10.3.5)$$

and the higher-order terms are zero. For nonquadratic functions, higher-order terms are nonzero, but if $\|\Delta\mathbf{c}\|$ is small, we can use a quadratic approximation. We note that if $\nabla P(\mathbf{c}_o) = \mathbf{0}$ and \mathbf{R} is positive definite, then \mathbf{c}_o is the minimum because $(\Delta\mathbf{c})^T [\nabla^2 P(\mathbf{c}_o)] (\Delta\mathbf{c}) > 0$ for any nonzero $\Delta\mathbf{c}$. Hence, if we choose the step $\Delta\mathbf{c}$ so that $(\Delta\mathbf{c})^T \nabla P(\mathbf{c}) < 0$, we will have $P(\mathbf{c} + \Delta\mathbf{c}) < P(\mathbf{c})$, that is, we make a step to a point closer to the minimum. Since $(\Delta\mathbf{c})^T \nabla P(\mathbf{c}) = \|\Delta\mathbf{c}\| \|\nabla P(\mathbf{c})\| \cos \theta$, the reduction in MSE is maximum when $\Delta\mathbf{c} = -\nabla P(\mathbf{c})$. For this reason, the direction of the negative gradient is known as the direction of *steepest descent*. This leads to the following iterative minimization algorithm

$$\mathbf{c}_k = \mathbf{c}_{k-1} + \mu [-\nabla P(\mathbf{c}_{k-1})] \quad k \geq 0 \quad (10.3.6)$$

which is known as the *method of steepest descent* (Scales 1985). The positive constant μ , known as the *step-size parameter*, controls the size of the descent in the direction of the negative gradient. The algorithm is usually initialized with $\mathbf{c}_0 = \mathbf{0}$. The steepest-descent algorithm (SDA) is illustrated in Figure 10.13 for a single-parameter case.

For the cost function in (10.3.1), the SDA becomes

$$\mathbf{c}_k = \mathbf{c}_{k-1} + 2\mu(\mathbf{d} - \mathbf{R}\mathbf{c}_{k-1}) = (\mathbf{I} - 2\mu\mathbf{R})\mathbf{c}_{k-1} + 2\mu\mathbf{d} \quad (10.3.7)$$

which is a recursive difference equation. Note that k denotes an iteration in the SDA and has nothing to do with time. However, this iterative optimization can be combined with filtering to obtain a type of “asymptotically” optimum filter defined by

$$e(n, \zeta) = y(n, \zeta) - \mathbf{c}_{n-1}^H \mathbf{x}(n, \zeta) \quad (10.3.8)$$

$$\mathbf{c}_n = \mathbf{c}_{n-1} + 2\mu(\mathbf{d} - \mathbf{R}\mathbf{c}_{n-1}) \quad (10.3.9)$$

and is further discussed in Problem 10.2.

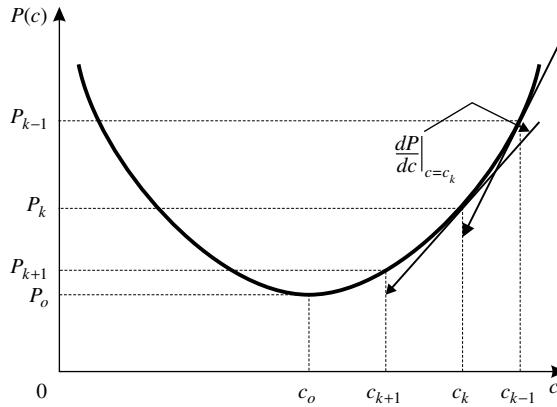


FIGURE 10.13
Illustration of gradient search of the MSE surface for the minimum error point.

There are two key performance factors in the design of iterative optimization algorithms: stability and rate of convergence.

Stability

An algorithm is said to be *stable* if it converges to the minimum regardless of the starting point. To investigate the stability of SDA, we rewrite (10.3.7) in terms of the coefficient error vector

$$\tilde{\mathbf{c}}_k \triangleq \mathbf{c}_k - \mathbf{c}_o \quad k \geq 0 \quad (10.3.10)$$

$$\text{as} \quad \tilde{\mathbf{c}}_k = (\mathbf{I} - 2\mu\mathbf{R})\tilde{\mathbf{c}}_{k-1} \quad k \geq 0 \quad (10.3.11)$$

which is a homogeneous difference equation. Using the principal-components transformation $\mathbf{R} = \mathbf{Q}\Lambda\mathbf{Q}^H$ (see Section 3.5), we can write (10.3.11) as

$$\tilde{\mathbf{c}}'_k = (\mathbf{I} - 2\mu\Lambda)\tilde{\mathbf{c}}'_{k-1} \quad k \geq 0 \quad (10.3.12)$$

$$\text{where} \quad \tilde{\mathbf{c}}'_k = \mathbf{Q}^H\tilde{\mathbf{c}}_k \quad k \geq 0 \quad (10.3.13)$$

is the transformed coefficient error vector. Since Λ is diagonal, (10.3.12) consists of a set of M decoupled first-order difference equations

$$\tilde{c}'_{k,i} = (1 - 2\mu\lambda_i)\tilde{c}'_{k-1,i} \quad i = 1, 2, \dots, M, k \geq 0 \quad (10.3.14)$$

with each describing a *natural mode* of the SDA. The solutions of (10.3.12) are given by

$$\tilde{c}'_{k,i} = (1 - 2\mu\lambda_i)^k \tilde{c}'_{0,i} \quad k \geq 0 \quad (10.3.15)$$

If for all $1 \leq i \leq M$

$$-1 < 1 - 2\mu\lambda_i < 1 \quad (10.3.16)$$

$$\text{or equivalently} \quad 0 < \mu < \frac{1}{\lambda_i} \quad (10.3.17)$$

then $\tilde{c}'_{k,i}$, $1 \leq i \leq M$, tends to zero as $k \rightarrow \infty$. This implies that \mathbf{c}_k converges *exponentially* to \mathbf{c}_o as $k \rightarrow \infty$ because $\|\tilde{\mathbf{c}}'_k\| = \|\mathbf{Q}^T\tilde{\mathbf{c}}_k\| = \|\tilde{\mathbf{c}}_k\|$. If \mathbf{R} is positive definite, its eigenvalues are positive and

$$0 < \mu < \frac{1}{\lambda_{\max}} \quad (10.3.18)$$

provides a necessary and sufficient condition for the convergence of SDA.

To investigate the transient behavior of the SDA as a function of k , we note that using (10.3.10), (10.3.11), and (10.3.14), we have

$$c_{k,i} = c_{o,i} + \sum_{i=1}^M q_{ik} \tilde{c}'_{0,i} (1 - 2\mu\lambda_i)^k \quad (10.3.19)$$

where $c_{o,i}$ are the optimum coefficients and q_{ik} the elements of the eigenvector matrix \mathbf{Q} . The MSE at step k is

$$P_k = P_o + \sum_{i=1}^M \lambda_i (1 - 2\mu\lambda_i)^{2k} |\tilde{c}'_{0,i}|^2 \quad (10.3.20)$$

and can be obtained by substituting (10.3.19) in (10.3.1). If μ satisfies (10.3.18), we have $\lim_{k \rightarrow \infty} P_k = P_o$ and the MSE converges exponentially to the optimum value. The curve obtained by plotting the MSE P_k as a function of the number of iterations k is known as the *learning curve*.

Rate of convergence

The rate (or speed) of convergence depends upon the algorithm and the nature of the performance surface. The most influential effect is inflicted by the condition number of the Hessian matrix that determines the shape of the contours of $P(\mathbf{c})$. When $P(\mathbf{c})$ is quadratic, it can be shown (Luenberger 1984) that

$$P(\mathbf{c}_k) \leq \left[\frac{\mathcal{X}(\mathbf{R}) - 1}{\mathcal{X}(\mathbf{R}) + 1} \right]^2 P(\mathbf{c}_{k-1}) \quad (10.3.21)$$

where $\mathcal{X}(\mathbf{R}) = \lambda_{\max}/\lambda_{\min}$ is the condition number of \mathbf{R} . If we recall that the eigenvectors corresponding to λ_{\min} and λ_{\max} point to the directions of minimum and maximum curvature, respectively, we see that the convergence slows down as the contours become more eccentric (flattened). For circular contours, that is, when $\mathcal{X}(\mathbf{R}) = 1$, the algorithm converges in one step. We stress that even if the $M - 1$ eigenvalues of \mathbf{R} are equal and the remaining one is far away, still the convergence of the SDA is very slow.

The rate of convergence can be characterized by using the *time constant* τ_i defined by

$$1 - 2\mu\lambda_i = \exp\left(-\frac{1}{\tau_i}\right) \simeq 1 - \frac{1}{\tau_i} \quad (10.3.22)$$

which provides the time (or number of iterations) it takes for the i th mode $c_{k,i}$ of (10.3.19) to decay to $1/e$ of its initial value $c_{0,i}$. When $\mu \ll 1$, we obtain

$$\tau_i \simeq \frac{1}{2\mu\lambda_i} \quad (10.3.23)$$

In a similar fashion, the time constant $\tau_{i,\text{mse}}$ for the MSE P_k can be shown to be

$$\tau_{i,\text{mse}} \simeq \frac{1}{4\mu\lambda_i} \quad (10.3.24)$$

by using (10.3.20) and (10.3.22).

Thus, for all practical purposes, the time constant (for coefficient \mathbf{c}_k or for MSE P_k) of the SDA is $\tau \simeq 1/(\mu\lambda_{\min})$, which in conjunction with $\mu < 1/\lambda_{\max}$ results in $\tau > \lambda_{\max}/\lambda_{\min}$. Hence, *the larger the eigenvalue spread of the input correlation matrix \mathbf{R} , the longer it takes for the SDA to converge*.

In the following example, we illustrate above-discussed properties of the SDA by using it to compute the parameters of a second-order forward linear predictor.

EXAMPLE 10.3.1. Consider a signal generated by the second-order autoregressive AR(2) process

$$x(n) + a_1 x(n-1) + a_2 x(n-2) = w(n) \quad (10.3.25)$$

where $w(n) \sim \text{WGN}(0, \sigma_w^2)$. Parameters a_1 and a_2 are chosen so that the system (10.3.25) is minimum-phase. We want to design an adaptive filter that uses the samples $x(n-1)$ and $x(n-2)$ to predict the value $x(n)$ (desired response).

If we multiply (10.3.25) by $x(n-k)$, for $k = 0, 1, 2$, and take the mathematical expectation of both sides, we obtain a set of linear equations

$$r(0) + a_1 r(1) + a_2 r(2) = \sigma_w^2 \quad (10.3.26)$$

$$r(1) + a_1 r(0) + a_2 r(1) = 0 \quad (10.3.27)$$

$$r(2) + a_1 r(1) + a_2 r(0) = 0 \quad (10.3.28)$$

which can be used to express the autocorrelation of $x(n)$ in terms of model parameters a_1 , a_2 , and σ_w^2 . Indeed, solving (10.3.26) through (10.3.28), we obtain

$$\begin{aligned} r(0) &= \sigma_x^2 = \frac{1+a_2}{1-a_2} \frac{\sigma_w^2}{(1+a_2)^2 - a_1^2} \\ r(1) &= \frac{-a_1}{1+a_2} r(0) \\ r(2) &= \left(-a_2 + \frac{a_1^2}{1+a_2} \right) r(0) \end{aligned} \quad (10.3.29)$$

We choose $\sigma_x^2 = 1$, so that

$$\sigma_w^2 = \frac{(1-a_2)[(1+a_2)^2 - a_1^2]}{1+a_2} \sigma_x^2 \quad (10.3.30)$$

The coefficients of the optimum predictor

$$\hat{y}(n) = \hat{x}(n) = c_{o,1}x(n-1) + c_{o,2}x(n-2) \quad (10.3.31)$$

are given by (see Section 6.5)

$$r(0)c_{o,1} + r(1)c_{o,2} = r(1) \quad (10.3.32)$$

$$r(1)c_{o,1} + r(0)c_{o,2} = r(2) \quad (10.3.33)$$

with

$$P_o^f = r(0) + r(1)c_{o,1} + r(0)c_{o,2} \quad (10.3.34)$$

whose comparison with (10.3.26) through (10.3.28) shows that $c_{o,1} = -a_1$, $c_{o,2} = -a_2$, and $P_o^f = \sigma_w^2$, as expected.

The eigenvalues of the input correlation matrix

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} \quad (10.3.35)$$

are

$$\lambda_{1,2} = \left(1 \mp \frac{a_1}{1+a_2} \right) \sigma_x^2 \quad (10.3.36)$$

from which the eigenvalue spread is

$$\mathcal{X}(\mathbf{R}) = \frac{\lambda_1}{\lambda_2} = \frac{1-a_1+a_2}{1+a_1+a_2} \quad (10.3.37)$$

which, if $a_2 > 0$ and $a_1 < 0$, is larger than 1.

Now we perform MATLAB experiments with varying eigenvalue spread $\mathcal{X}(\mathbf{R})$ and step-size parameter μ . In these experiments, we choose σ_w^2 so that $\sigma_x^2 = 1$. The SDA is given by

$$\mathbf{c}_k \triangleq [c_{k,1} \ c_{k,2}]^T = \mathbf{c}_{k-1} + 2\mu(\mathbf{d} - \mathbf{R}\mathbf{c}_{k-1})$$

where

$$\mathbf{d} = [r(1) \ r(2)]^T \quad \text{and} \quad \mathbf{c}_0 = [0 \ 0]^T$$

We choose two different sets of values for a_1 and a_2 , one for a small and the other for a large eigenvalue spread. These values are shown in Table 10.2 along with the corresponding eigenvalue spread $\mathcal{X}(\mathbf{R})$ and the MMSE σ_w^2 .

TABLE 10.2
Parameter values used in the SDA for the second-order forward prediction problem.

Eigenvalue spread	a_1	a_2	λ_1	λ_2	$\mathcal{X}(\mathbf{R})$	σ_w^2
Small	-0.1950	0.95	1.1	0.9	1.22	0.0965
Large	-1.5955	0.95	1.818	0.182	9.99	0.0322

Using each set of parameter values, the SDA is implemented starting with the null coefficient vector \mathbf{c}_0 with two values of step-size parameters. To describe the transient behavior of the algorithm, it is informative to plot the trajectory of $c_{k,1}$ versus $c_{k,2}$ as a function of the iteration index k along with the contours of the error surface $P(\mathbf{c}_k)$. The trajectory of \mathbf{c}_k begins at the origin $\mathbf{c}_0 = \mathbf{0}$ and ends at the optimum value $\mathbf{c}_o = -[a_1 \ a_2]^T$. This illustration of the transient behavior can also be obtained in the domain of the transformed error coefficients $\tilde{\mathbf{c}}'_k$. Using (10.3.15), we see these coefficients are given by

$$\tilde{\mathbf{c}}'_k = \begin{bmatrix} \tilde{c}'_{k,1} \\ \tilde{c}'_{k,2} \end{bmatrix} = \begin{bmatrix} (1 - 2\mu\lambda_1)^k \tilde{c}'_{0,1} \\ (1 - 2\mu\lambda_2)^k \tilde{c}'_{0,2} \end{bmatrix} \quad (10.3.38)$$

where $\tilde{\mathbf{c}}'_0$ from (10.3.10) and (10.3.13) is given by

$$\tilde{\mathbf{c}}'_0 = \begin{bmatrix} \tilde{c}'_{0,1} \\ \tilde{c}'_{0,2} \end{bmatrix} = \mathbf{Q}^T \tilde{\mathbf{c}}_0 = \mathbf{Q}^T (\mathbf{c}_0 - \mathbf{c}_o) = -\mathbf{Q}^T \mathbf{c}_o = \mathbf{Q}^T \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \quad (10.3.39)$$

Thus the trajectory of $\tilde{\mathbf{c}}'_k$ begins at $\tilde{\mathbf{c}}'_0$ and ends at the origin $\tilde{\mathbf{c}}'_k = \mathbf{0}$. The contours of the MSE function in the transformed domain are given by $P_k - P_o$. From (10.3.20), these contours are given by

$$P_k - P_o^f = \sum_{i=1}^2 \lambda_i (\tilde{c}'_{k,i})^2 = \lambda_1 (\tilde{c}'_{k,1})^2 + \lambda_2 (\tilde{c}'_{k,2})^2 \quad (10.3.40)$$

Small eigenvalue spread and overdamped response. For this experiment, the parameter values were selected to obtain the eigenvalue spread approximately equal to 1 [$\mathcal{X}(\mathbf{R}) = 1.22$]. The step size selected was $\mu = 0.15$, which is less than $1/\lambda_{\max} = 1/1.1 = 0.9$ for convergence. For this value of μ , the transient response is overdamped. Figure 10.14 shows four graphs indicating

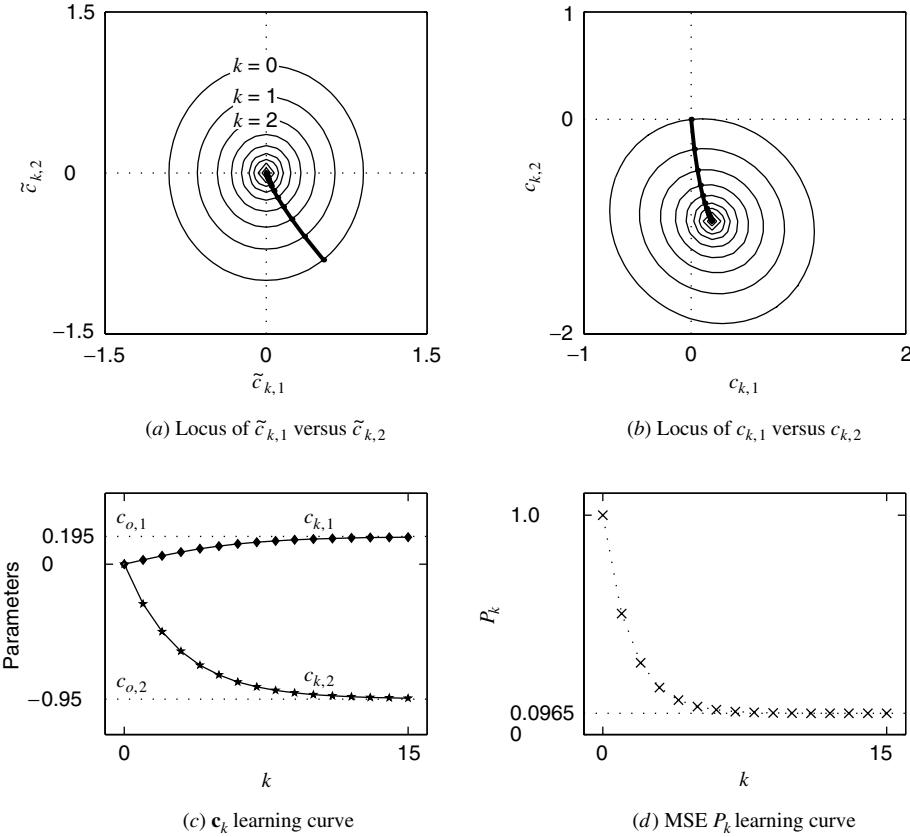


FIGURE 10.14

Performance curves for the steepest-descent algorithm used in the linear prediction problem with step-size parameter $\mu = 0.15$ and eigenvalue spread $\mathcal{X}(\mathbf{R}) = 1.22$.

the behavior of the algorithm. In the graph (a), the trajectory of $\tilde{\mathbf{c}}'_k$ is shown for $0 \leq k \leq 15$ along with the corresponding loci $\tilde{\mathbf{c}}'_k$ for a fixed value of $P_k - P_o$. The first two loci for $k = 0$ and 1 are numbered to show the direction of the trajectory. Graph (b) shows the corresponding trajectory and the contours for \mathbf{c}_k . Graph (c) shows plots of $c_{k,1}$ and $c_{k,2}$ as a function of iteration step k , while graph (d) shows a similar learning curve for the MSE P_k . Several observations can be made about these plots. The contours of constant $\tilde{\mathbf{c}}'_k$ are almost circular since the spread is approximately 1, while those of \mathbf{c}_k are somewhat elliptical, which is to be expected. The trajectories of $\tilde{\mathbf{c}}'_k$ and \mathbf{c}_k as a function of k are normal to the contours. The coefficients converge to their optimum values in a monotonic fashion, which confirms the overdamped nature of the response. Also this convergence is rapid, in about 15 steps, which is to be expected for a small eigenvalue spread.

Large eigenvalue spread and overdamped response. For this experiment, the parameter values were selected so that the eigenvalue spread was approximately equal to 10 [$\mathcal{X}(\mathbf{R}) = 9.99$]. The step size was again selected as $\mu = 0.15$. Figure 10.15 shows the performance plots for this experiment, which are similar to those of Figure 10.14. The observations are also similar except for those due to the larger spread. First, the contours, even in the transformed domain, are elliptical; second, the convergence is slow, requiring about 60 steps in the algorithm. The transient response is once again overdamped.

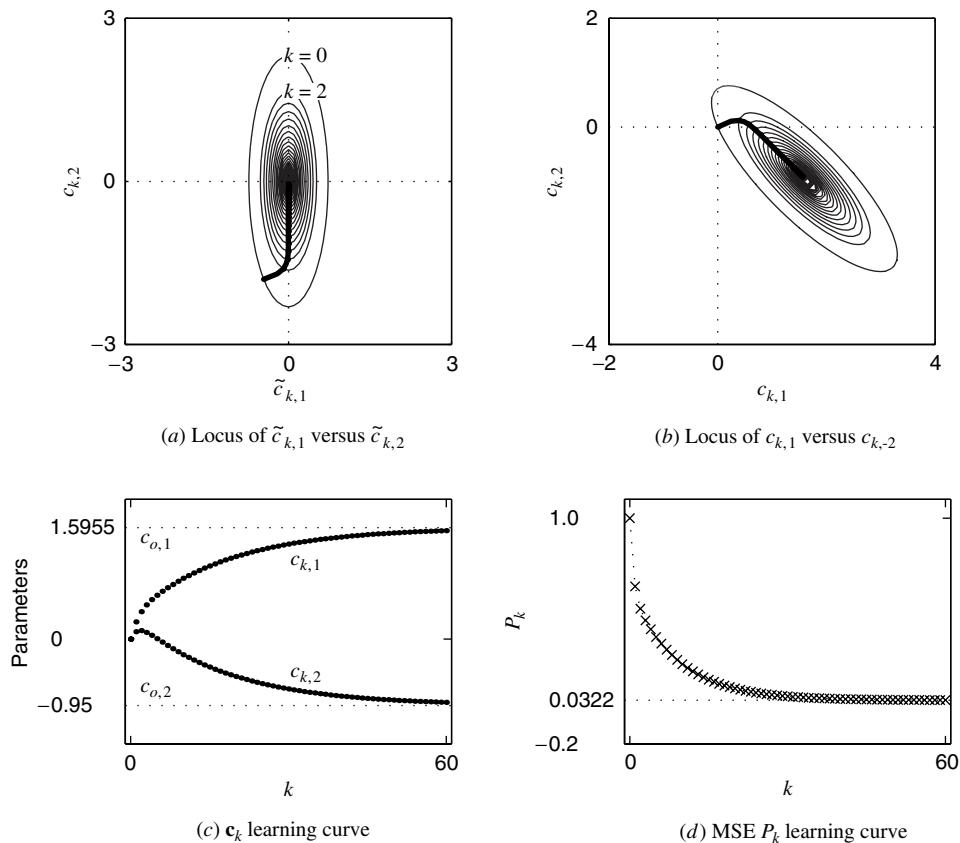


FIGURE 10.15

Performance curves for the steepest-descent algorithm used in the linear prediction problem with step-size parameter $\mu = 0.15$ and eigenvalue spread $\mathcal{X}(\mathbf{R}) = 10$.

Large eigenvalue spread and underdamped response. Finally, in the third experiment, we consider the model parameters of the above case and increase the step size to $\mu = 0.5 (< 1/\lambda_{\max} = 0.55)$ so that the transient response is underdamped. Figure 10.16 shows the corresponding plots.

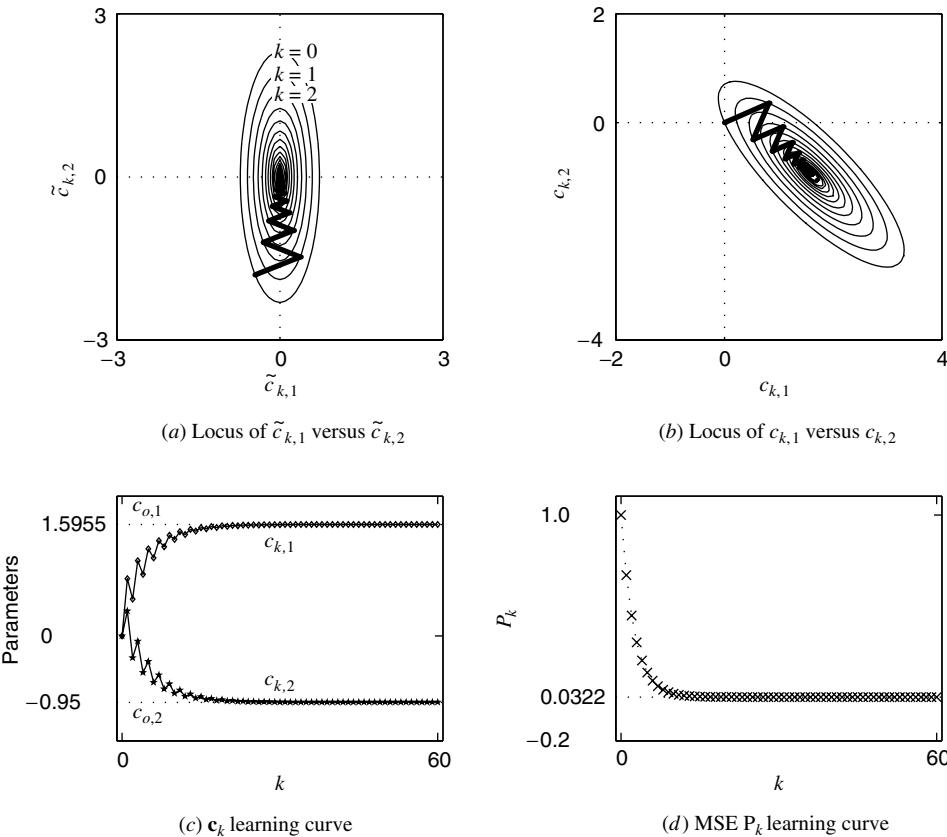


FIGURE 10.16

Performance curves for the steepest-descent algorithm used in the linear prediction problem with eigenvalue spread $\lambda(\mathbf{R}) = 10$ and varying step-size parameters $\mu = 0.15$ and $\mu = 0.5$.

Note that the coefficients converge in an oscillatory fashion; however, the convergence is fairly rapid compared to that of the overdamped case. Thus the selection of the step size is an important design issue.

Newton's type of algorithms

Another family of algorithms with a faster rate of convergence includes Newton's method and its modifications. The basic idea of Newton's method is to achieve convergence in one step when $P(\mathbf{c})$ is quadratic. Thus, if \mathbf{c}_k is to be the minimum of $P(\mathbf{c})$, the gradient $\nabla P(\mathbf{c}_k)$ of $P(\mathbf{c})$ evaluated at \mathbf{c}_k (10.2.19) should be zero. From (10.2.19), we can write

$$\nabla P(\mathbf{c}_k) = \nabla P(\mathbf{c}_{k-1}) + \nabla^2 P(\mathbf{c}_{k-1})\Delta\mathbf{c}_k = \mathbf{0} \quad (10.3.41)$$

Thus $\nabla P(\mathbf{c}_k) = \mathbf{0}$ leads to the step increment

$$\Delta\mathbf{c}_k = -[\nabla^2 P(\mathbf{c}_{k-1})]^{-1}\nabla P(\mathbf{c}_{k-1}) \quad (10.3.42)$$

and hence the adaptive algorithm is given by

$$\mathbf{c}_k = \mathbf{c}_{k-1} - \mu[\nabla^2 P(\mathbf{c}_{k-1})]^{-1}\nabla P(\mathbf{c}_{k-1}) \quad (10.3.43)$$

where $\mu > 0$ is the step size. For quadratic error surfaces, from (10.3.4) and (10.3.5), we obtain with $\mu = 1$

$$\mathbf{c}_k = \mathbf{c}_{k-1} - [\nabla^2 P(\mathbf{c}_{k-1})]^{-1}\nabla P(\mathbf{c}_{k-1}) = \mathbf{c}_{k-1} - (\mathbf{c}_{k-1} - \mathbf{R}^{-1}\mathbf{d}) = \mathbf{c}_o \quad (10.3.44)$$

which shows that indeed the algorithm converges in one step.

For the quadratic case, since $\nabla^2 P(\mathbf{c}_{k-1}) = 2\mathbf{R}$ from (10.3.1), we can express Newton's algorithm as

$$\mathbf{c}_k = \mathbf{c}_{k-1} - \mu \mathbf{R}^{-1} \nabla P(\mathbf{c}_{k-1}) \quad (10.3.45)$$

where μ is the step size that regulates the convergence rate. Other modified Newton methods replace the Hessian matrix $\nabla^2 P(\mathbf{c}_{k-1})$ with another matrix, which is guaranteed to be positive definite and, in some way, close to the Hessian. These Newton-type algorithms generally provide faster convergence. However, in practice, the inversion of \mathbf{R} is numerically intensive and can lead to a numerically unstable solution if special care is not taken. Therefore, the SDA is more popular in adaptive filtering applications.

When the function $P(\mathbf{c})$ is nonquadratic, it is approximated locally by a quadratic function that is minimized exactly. However, the step obtained in (10.3.42) does not lead to the minimum of $P(\mathbf{c})$, and the iteration should be repeated several times. A more detailed treatment of linear and nonlinear optimization techniques can be found in Scales (1985) and in Luenberger (1984).

10.4 LEAST-MEAN-SQUARE ADAPTIVE FILTERS

In this section, we derive, analyze the performance, and present some practical applications of the *least-mean-square* (LMS) adaptive algorithm. The LMS algorithm, introduced by Widrow and Hoff (1960), is widely used in practice due to its simplicity, computational efficiency, and good performance under a variety of operating conditions.

10.4.1 Derivation

We first present two approaches to the derivation of the LMS algorithm that will help the reader to understand its operation. The first approach uses approximation to the gradient function while the second approach uses geometric arguments.

Optimization approach. The SDA uses the second-order moments \mathbf{R} and \mathbf{d} to iteratively compute the optimum filter $\mathbf{c}_o = \mathbf{R}^{-1}\mathbf{d}$, starting with an initial guess, usually $\mathbf{c}_0 = \mathbf{0}$, and then obtaining better approximations by taking steps in the direction of the negative gradient, that is,

$$\mathbf{c}_k = \mathbf{c}_{k-1} + \mu [-\nabla P(\mathbf{c}_{k-1})] \quad (10.4.1)$$

where

$$\nabla P(\mathbf{c}_{k-1}) = 2(\mathbf{R}\mathbf{c}_{k-1} - \mathbf{d}) \quad (10.4.2)$$

is the gradient of the performance function (10.3.1). In practice, where only the input $\{\mathbf{x}(j)\}_0^n$ and the desired response $\{y(j)\}_0^n$ are known, we can only compute an estimate of the “true” or exact gradient (10.4.2) using the available data. To develop an adaptive algorithm from (10.4.1), we take the following steps: (1) replace the iteration subscript k by the time index n ; and (2) replace \mathbf{R} and \mathbf{d} by their instantaneous estimates $\mathbf{x}(n)\mathbf{x}^H(n)$ and $\mathbf{x}(n)y^*(n)$, respectively. The instantaneous estimate of the gradient (10.4.2) becomes

$$\nabla P(\mathbf{c}_{k-1}) = 2\mathbf{R}\mathbf{c}_{k-1} - 2\mathbf{d} \simeq 2\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{c}(n-1) - 2\mathbf{x}(n)y^*(n) = -2\mathbf{x}(n)e^*(n) \quad (10.4.3)$$

where

$$e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n) \quad (10.4.4)$$

is the a priori filtering error. The estimate (10.4.3) also can be obtained by starting with the approximation $P(\mathbf{c}) \simeq |e(n)|^2$ and taking its gradient. The coefficient adaptation algorithm is

$$\mathbf{c}(n) = \mathbf{c}(n-1) + 2\mu\mathbf{x}(n)e^*(n) \quad (10.4.5)$$

which is obtained by substituting (10.4.3) and (10.4.4) in (10.4.1). The step-size parameter 2μ is also known as the *adaptation gain*.

The LMS algorithm, specified by (10.4.5) and (10.4.4), has both important similarities to and important differences from the SDA (10.3.7). The SDA contains deterministic quantities while the LMS operates on random quantities. The SDA is *not* an adaptive algorithm because it only depends on the second-order moments \mathbf{R} and \mathbf{d} and not on the SOE $\{\mathbf{x}(n, \zeta), y(n, \zeta)\}$. Also, the iteration index k has nothing to do with time. Simply stated, the SDA provides an iterative solution to the linear system $\mathbf{R}\mathbf{c} = \mathbf{d}$.

Geometric approach. Suppose that an adaptive filter operates in a stationary signal environment seeking the optimum filter \mathbf{c}_o . At time n the filter has access to input vector $\mathbf{x}(n)$, the desired response $y(n)$, and the previous or old coefficient estimate $\mathbf{c}(n - 1)$. Its goal is to use this information to determine a new estimate $\mathbf{c}(n)$ that is closer to the optimum vector \mathbf{c}_o or equivalently to choose $\mathbf{c}(n)$ so that $\|\tilde{\mathbf{c}}(n)\| < \|\tilde{\mathbf{c}}(n-1)\|$, where $\tilde{\mathbf{c}}(n) = \mathbf{c}(n) - \mathbf{c}_o$ is the coefficient error vector given by (10.2.24). Eventually, we want $\|\tilde{\mathbf{c}}(n)\|$ to become negligible as $n \rightarrow \infty$.

The vector $\tilde{\mathbf{c}}(n - 1)$ can be decomposed into two orthogonal components

$$\tilde{\mathbf{c}}(n - 1) = \tilde{\mathbf{c}}_x(n - 1) + \tilde{\mathbf{c}}_x^\perp(n - 1) \quad (10.4.6)$$

one parallel and one orthogonal to the input vector $\mathbf{x}(n)$, as shown in Figure 10.17(a). The response of the *error filter* $\tilde{\mathbf{c}}(n - 1)$ to the input $\mathbf{x}(n)$ is

$$\tilde{y}(n) = \tilde{\mathbf{c}}^H(n - 1)\mathbf{x}(n) = \tilde{\mathbf{c}}_x^H(n - 1)\mathbf{x}(n) \quad (10.4.7)$$

which implies that

$$\tilde{\mathbf{c}}_x(n - 1) = \frac{\tilde{y}^*(n)}{\|\mathbf{x}(n)\|^2} \mathbf{x}(n) \quad (10.4.8)$$

which can be verified by direct substitution in (10.4.7). Note that $\mathbf{x}(n)/\|\mathbf{x}(n)\|$ is a unit vector along the direction of $\mathbf{x}(n)$.

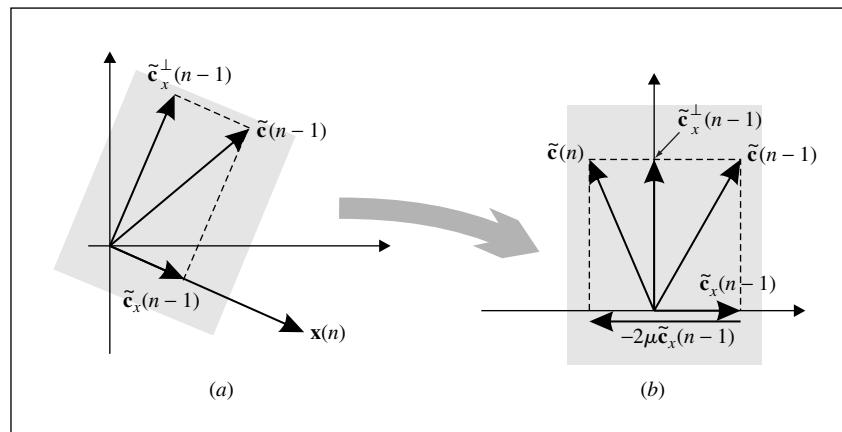


FIGURE 10.17

The geometric approach for the derivation of the LMS algorithm.

If we only know $\mathbf{x}(n)$ and $\tilde{y}(n)$, the best strategy to decrease $\tilde{\mathbf{c}}(n)$ is to choose $\tilde{\mathbf{c}}(n) = \tilde{\mathbf{c}}_x^\perp(n - 1)$, or equivalently subtract $\tilde{\mathbf{c}}_x(n - 1)$ from $\tilde{\mathbf{c}}(n - 1)$. From Figure 10.17(a) note that as long as $\tilde{\mathbf{c}}_x(n - 1) \neq \mathbf{0}$, $\|\tilde{\mathbf{c}}(n)\| = \|\tilde{\mathbf{c}}_x^\perp(n - 1)\| < \|\tilde{\mathbf{c}}_x(n - 1)\|$. This suggests the following adaptation algorithm

$$\tilde{\mathbf{c}}(n) = \tilde{\mathbf{c}}(n - 1) - \tilde{\mu} \frac{\tilde{y}^*(n)}{\|\mathbf{x}(n)\|^2} \mathbf{x}(n) \quad (10.4.9)$$

which guarantees that $\|\tilde{\mathbf{c}}(n)\| < \|\tilde{\mathbf{c}}(n-1)\|$ as long as $0 < \tilde{\mu} < 2$ and $\tilde{y}(n) \neq 0$, as shown in Figure 10.17(b). The best choice clearly is $\tilde{\mu} = 1$.

Unfortunately, the signal $\tilde{y}(n)$ is not available, and we have to replace it with some reasonable approximation. From (10.2.18) and (10.2.10) we obtain

$$\begin{aligned}\tilde{e}(n) &\triangleq e(n) - e_o(n) = y(n) - \hat{y}(n) - y(n) + \hat{y}_o(n) = \hat{y}_o(n) - \hat{y}(n) \\ &= [\mathbf{c}_o^H - \mathbf{c}^H(n-1)]\mathbf{x}(n) = -\tilde{\mathbf{c}}^H(n-1)\mathbf{x}(n) = -\tilde{y}(n)\end{aligned}\quad (10.4.10)$$

where we have used (10.4.7). Using the approximation

$$\tilde{e}(n) = e(n) - e_o(n) \simeq e(n)$$

we combine it with (10.4.10) to get

$$\mathbf{c}(n) = \mathbf{c}(n-1) + \tilde{\mu} \frac{e^*(n)}{\|\mathbf{x}(n)\|^2} \mathbf{x}(n) \quad (10.4.11)$$

which is known as the *normalized LMS algorithm*. Note that the effective step size $\tilde{\mu}/\|\mathbf{x}(n)\|^2$ is time-varying. The LMS algorithm in (10.4.5) follows if we set $\|\mathbf{x}(n)\| = 1$ and choose $\tilde{\mu} = 2\mu$.

LMS algorithm. The LMS algorithm can be summarized as

$$\begin{aligned}\hat{y}(n) &= \mathbf{c}^H(n-1)\mathbf{x}(n) && \text{filtering} \\ e(n) &= y(n) - \hat{y}(n) && \text{error formation} \\ \mathbf{c}(n) &= \mathbf{c}(n-1) + 2\mu \mathbf{x}(n)e^*(n) && \text{coefficient updating}\end{aligned}\quad (10.4.12)$$

where μ is adaptation step size. The algorithm requires $2M+1$ complex multiplications and $2M$ complex additions. Figure 10.18 shows an implementation of an FIR adaptive filter using the LMS algorithm, which is implemented in MATLAB using the function `[yhat, c] = firLms(x, y, M, mu)`. The a posteriori form of the LMS algorithm is developed in Problem 10.9.

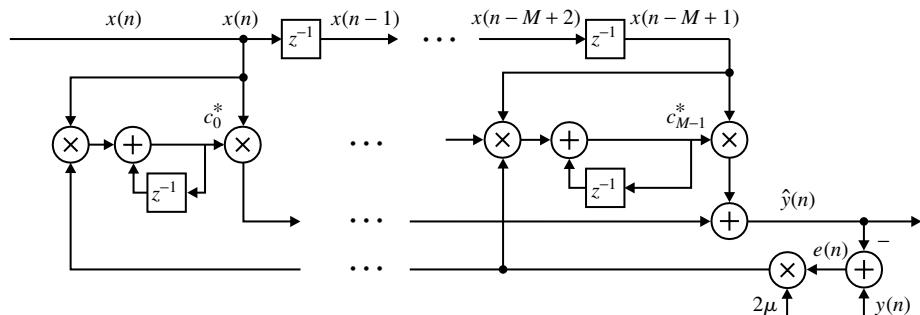


FIGURE 10.18

An FIR adaptive filter realization using the LMS algorithm.

10.4.2 Adaptation in a Stationary SOE

In the sequel, we study the stability and steady-state performance of the LMS algorithm in a stationary SOE; that is, we assume that the input and the desired response processes are jointly stationary. In theory, the goal of the LMS adaptive filter is to identify the optimum filter $\mathbf{c}_o = \mathbf{R}^{-1}\mathbf{d}$ from observations of the input $\mathbf{x}(n)$ and the desired response

$$y(n) = \mathbf{c}_o^H \mathbf{x}(n) + e_o(n) \quad (10.4.13)$$

The optimum error $e_o(n)$ is orthogonal to the vector $\mathbf{x}(n)$; that is, $E\{\mathbf{x}(n)e^*(n)\} = \mathbf{0}$ and acts as measurement or output noise, as shown in Figure 10.19.

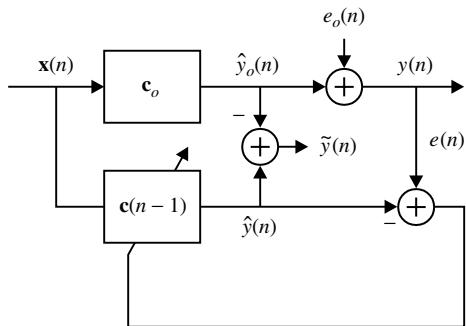


FIGURE 10.19
 LMS algorithm in a stationary SOE.

The first step in the statistical analysis of the LMS algorithm is to determine a difference equation for the coefficient error vector $\tilde{\mathbf{c}}(n)$. To this end, we subtract \mathbf{c}_o from both sides of (10.4.5), to obtain

$$\tilde{\mathbf{c}}(n) = \tilde{\mathbf{c}}(n-1) + 2\mu \mathbf{x}(n) e^*(n) \quad (10.4.14)$$

which expresses the LMS algorithm in terms of the coefficient error vector. We next use (10.4.12) and (10.4.13) in (10.4.14) to eliminate $e(n)$ by expressing it in terms of $\tilde{\mathbf{c}}(n-1)$ and $e_o(n)$. The result is

$$\tilde{\mathbf{c}}(n) = [\mathbf{I} - 2\mu \mathbf{x}(n) \mathbf{x}^H(n)] \tilde{\mathbf{c}}(n-1) + 2\mu \mathbf{x}(n) e_o^*(n) \quad (10.4.15)$$

which is a time-varying forced or nonhomogeneous stochastic difference equation. The irreducible error $e_o(n)$ accounts for measurement noise, modeling errors, unmodeled dynamics, quantization effects, and other disturbances. The presence of $e_o(n)$ prevents convergence because it forces $\tilde{\mathbf{c}}(n)$ to fluctuate around zero. Therefore, the important issue is the BIBO stability of the system (10.4.15). From (10.2.28), we see that $\|\tilde{\mathbf{c}}(n)\|$ is bounded in mean square if we can show that $E\{\tilde{\mathbf{c}}(n)\} \rightarrow 0$ as $n \rightarrow \infty$ and $\text{var}\{\tilde{\mathbf{c}}_k(n)\}$ is bounded for all n . To this end, we develop difference equations for the mean value $E\{\tilde{\mathbf{c}}(n)\}$ and the correlation matrix

$$\Phi(n) \triangleq E\{\tilde{\mathbf{c}}(n) \tilde{\mathbf{c}}^H(n)\} \quad (10.4.16)$$

of the coefficient error vector $\tilde{\mathbf{c}}(n)$. As we shall see, the MSD and the EMSE can be expressed in terms of matrices $\Phi(n)$ and \mathbf{R} . The time evolution of these quantities provides sufficient information to evaluate the stability and steady-state performance of the LMS algorithm.

Convergence of the mean coefficient vector

If we take the expectation of (10.4.15), we have

$$E\{\tilde{\mathbf{c}}(n)\} = E\{\tilde{\mathbf{c}}(n-1)\} - 2\mu E\{\mathbf{x}(n) \mathbf{x}^H(n) \tilde{\mathbf{c}}(n-1)\} \quad (10.4.17)$$

because $E\{\mathbf{x}(n) e_o^*(n)\} = \mathbf{0}$ owing to the orthogonality principle. The computation of the second term in (10.4.17) requires the correlation between the input signal and the coefficient error vector.

If we assume that $\mathbf{x}(n)$ and $\tilde{\mathbf{c}}(n-1)$ are statistically independent, (10.4.17) simplifies to

$$E\{\tilde{\mathbf{c}}(n)\} = (\mathbf{I} - 2\mu \mathbf{R}) E\{\tilde{\mathbf{c}}(n-1)\} \quad (10.4.18)$$

which has the same form as (10.3.11) for the SDA. Therefore, $\tilde{\mathbf{c}}(n)$ converges in the MS sense, that is, $\lim_{n \rightarrow \infty} E\{\tilde{\mathbf{c}}(n)\} = \mathbf{0}$, if the eigenvalues of the system matrix $(\mathbf{I} - 2\mu \mathbf{R})$ are less than 1. Hence, if \mathbf{R} is positive definite and λ_{\max} is its maximum eigenvalue, the condition

$$0 < 2\mu < \frac{1}{\lambda_{\max}} \quad (10.4.19)$$

ensures that the LMS algorithm converges in the MS sense [see the discussion following (10.2.27)].

Independence assumption. The independence assumption between $\mathbf{x}(n)$ and $\tilde{\mathbf{c}}(n-1)$ was critical to the derivation of (10.4.18). To simplify the analysis, we make the following *independence assumptions* (Gardner 1984):

- A1** The sequence of input data vectors $\mathbf{x}(n)$ is independently and identically distributed with zero mean and correlation matrix \mathbf{R} .
A2 The sequences $\mathbf{x}(n)$ and $e_o(n)$ are independent for all n .

From (10.4.15), we see that $\tilde{\mathbf{c}}(n-1)$ depends on $\tilde{\mathbf{c}}(0)$, $\{\mathbf{x}(k)\}_{0}^{n-1}$, and $\{e_o(k)\}_{0}^{n-1}$. Since the sequence $\mathbf{x}(n)$ is IID and the quantities $\mathbf{x}(n)$ and $e_o(n)$ are independent, we conclude that $\mathbf{x}(n)$, $e_o(n)$, and $\tilde{\mathbf{c}}(n-1)$ are mutually independent. This result will be used several times to simplify the analysis of the LMS algorithm.

The independence assumption A1, first introduced in Widrow et al. (1976) and in Mazo (1979), ignores the statistical dependence among successive input data vectors; however, it preserves sufficient statistical information about the adaptation process to lead to useful design guidelines. Clearly, for FIR filtering applications, the independence assumption is violated because two successive input data vectors $\mathbf{x}(n)$ and $\mathbf{x}(n+1)$ have $M - 1$ common elements (shift-invariance property).

Evolution of the coefficient error correlation matrix

The MSD can be expressed in terms of the trace of the correlation matrix[†] $\Phi(n)$, that is,

$$\mathcal{D}(n) = \text{tr}[\Phi(n)] \quad (10.4.20)$$

which can be easily seen by using (10.2.29) and the definition of trace. If we postmultiply both sides of (10.4.15) by their respective Hermitian transposes and take the mathematical expectation, we obtain

$$\begin{aligned} \Phi(n) &= E\{\tilde{\mathbf{c}}(n)\tilde{\mathbf{c}}^H(n)\} \\ &= E\{[\mathbf{I} - 2\mu\mathbf{x}(n)\mathbf{x}^H(n)]\tilde{\mathbf{c}}(n-1)\tilde{\mathbf{c}}^H(n-1)[\mathbf{I} - 2\mu\mathbf{x}(n)\mathbf{x}^H(n)]^H\} \\ &\quad + 2\mu E\{[\mathbf{I} - 2\mu\mathbf{x}(n)\mathbf{x}^H(n)]\tilde{\mathbf{c}}(n-1)e_o(n)\mathbf{x}^H(n)\} \\ &\quad + 2\mu E\{\mathbf{x}(n)e_o^*(n)\tilde{\mathbf{c}}^H(n-1)[\mathbf{I} - 2\mu\mathbf{x}(n)\mathbf{x}^H(n)]^H\} \\ &\quad + 4\mu^2 E\{\mathbf{x}(n)e_o^*(n)e_o(n)\mathbf{x}^H(n)\} \end{aligned} \quad (10.4.21)$$

From the independence assumptions, $e_o(n)$ is independent with $\tilde{\mathbf{c}}(n-1)$ and $\mathbf{x}(n)$. Therefore, the second and third terms in (10.4.21) vanish, and the fourth term is equal to $4\mu^2 P_o \mathbf{R}$. If we expand the first term, we obtain

$$\Phi(n) = \Phi(n-1) - 2\mu[\mathbf{R}\Phi(n-1) + \Phi(n-1)\mathbf{R}] + 4\mu^2 \mathbf{A} + 4\mu^2 P_o \mathbf{R} \quad (10.4.22)$$

$$\text{where } \mathbf{A} \triangleq E\{\mathbf{x}(n)\mathbf{x}^H(n)\tilde{\mathbf{c}}(n-1)\tilde{\mathbf{c}}^H(n-1)\mathbf{x}(n)\mathbf{x}^T(n)\} \quad (10.4.23)$$

and the terms $\mathbf{R}\Phi(n-1)$ and $\Phi(n-1)\mathbf{R}$ have been computed by using the mutual independence of $\mathbf{x}(n)$, $\tilde{\mathbf{c}}(n-1)$, and $e_o(n)$.

The computation of matrix \mathbf{A} can be simplified if we make additional assumptions about the statistical properties of $\mathbf{x}(n)$. As shown in Gardner (1984), development of a recursive relation for the elements of $\Phi(n)$ using only the independence assumptions requires the products with and the inversion of a $M^2 \times M^2$ matrix, where M is the size of $\mathbf{x}(n)$.

The evaluation of this term when $\mathbf{x}(n) \sim \text{IID}$, an assumption that is more appropriate for data transmission applications, is discussed in Gardner (1984). The computation for $\mathbf{x}(n)$ being a *spherically invariant random process (SIRP)* is discussed in Rupp (1993). SIRP models, which include the Gaussian distribution as a special case, provide a good

[†]Note that when (10.4.19) holds, $\lim_{n \rightarrow \infty} E\{\tilde{\mathbf{c}}(n)\} = \mathbf{0}$, and therefore $\Phi(n)$ provides asymptotically the covariance of $\tilde{\mathbf{c}}(n)$.

characterization of speech signals. However, independently of the assumption used, the basic conclusions remain the same.

Assuming that $\mathbf{x}(n)$ is normally distributed, that is, $\mathbf{x}(n) \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$, a significant amount of simplification can be obtained. Indeed, in this case we can use the moment factorization property for normal random variables to express fourth-order moments in terms of second-order moments (Papoulis 1991). As we showed in Section 3.2.3, if z_1, z_2, z_3 , and z_4 are complex-valued, zero-mean, and jointly distributed normal random variables, then

$$E\{z_1 z_2^* z_3 z_4^*\} = E\{z_1 z_2^*\} E\{z_3 z_4^*\} + E\{z_1 z_4^*\} E\{z_2^* z_3\} \quad (10.4.24)$$

or if they are real-valued, then

$$E\{z_1 z_2 z_3 z_4\} = E\{z_1 z_2\} E\{z_3 z_4\} + E\{z_1 z_3\} E\{z_2 z_4\} + E\{z_1 z_4\} E\{z_2 z_3\} \quad (10.4.25)$$

Using direct substitution of (10.4.24) or (10.4.25) in (10.4.23), we can show that

$$\mathbf{A} = \begin{cases} \mathbf{R}\Phi(n-1)\mathbf{R} + \mathbf{R}\text{tr}[\mathbf{R}\Phi(n-1)] & \text{complex case} \\ 2\mathbf{R}\Phi(n-1)\mathbf{R} + \mathbf{R}\text{tr}[\mathbf{R}\Phi(n-1)] & \text{real case} \end{cases} \quad (10.4.26)$$

Finally, substituting (10.4.26) in (10.4.22), we obtain a difference equation for $\Phi(n)$. This is summarized in the following property:

PROPERTY 10.4.1. Using the independence assumptions A1 and A2, and the normal distribution assumption of $\mathbf{x}(n)$, the correlation matrix of the coefficient error vector $\tilde{\mathbf{c}}(n)$ satisfies the difference equation

$$\begin{aligned} \Phi(n) = & \Phi(n-1) - 2\mu[\mathbf{R}\Phi(n-1) + \Phi(n-1)\mathbf{R}] \\ & + 4\mu^2\mathbf{R}\Phi(n-1)\mathbf{R} + 4\mu^2\mathbf{R}\text{tr}[\mathbf{R}\Phi(n-1)] + 4\mu^2P_o\mathbf{R} \end{aligned} \quad (10.4.27)$$

in the complex case and

$$\begin{aligned} \Phi(n) = & \Phi(n-1) - 2\mu[\mathbf{R}\Phi(n-1) + \Phi(n-1)\mathbf{R}] \\ & + 8\mu^2\mathbf{R}\Phi(n-1)\mathbf{R} + 4\mu^2\mathbf{R}\text{tr}[\mathbf{R}\Phi(n-1)] + 4\mu^2P_o\mathbf{R} \end{aligned} \quad (10.4.28)$$

in the real case. Both relations are matrix difference equations driven by the constant term $4\mu^2P_o\mathbf{R}$.

The presence of the term $4\mu^2P_o\mathbf{R}$ in (10.4.27) or (10.4.28) implies that $\Phi(n)$ will never become zero, and as a result the coefficients of the LMS adaptive filter will always fluctuate about their optimum settings, which prevents convergence. It has been shown (Bucklew et al. 1993) that asymptotically $\tilde{\mathbf{c}}(n)$ follows a zero-mean normal distribution. The amount of fluctuation is measured by matrix $\Phi(n)$. In contrast, the absence of a driving term in (10.4.18) allows the convergence of $E\{\mathbf{c}(n)\}$ to the optimum vector \mathbf{c}_o .

Since there are two distinct forms for the difference equation of $\Phi(n)$, we will consider the *real* case (10.4.28) for further discussion. Similar analysis can be done for the complex case (10.4.27), which is undertaken in Problem 10.11. To further simplify the analysis, we transform $\Phi(n)$ to the principal coordinate space of \mathbf{R} using the spectral decomposition

$$\mathbf{Q}^T \mathbf{R} \mathbf{Q} = \Lambda$$

by defining the matrix

$$\Theta(n) \triangleq \mathbf{Q}^T \Phi(n) \mathbf{Q} \quad (10.4.29)$$

which is symmetric and positive definite [when $\Phi(n)$ is positive definite].

If we pre- and postmultiply (10.4.28) by \mathbf{Q}^T and \mathbf{Q} and use $\mathbf{Q}^T \mathbf{Q} = \mathbf{Q} \mathbf{Q}^T = \mathbf{I}$, we obtain

$$\begin{aligned} \Theta(n) = & \Theta(n-1) - 2\mu[\Lambda \Theta(n-1) + \Theta(n-1)\Lambda] \\ & + 8\mu^2\Lambda \Theta(n-1)\Lambda + 4\mu^2\Lambda \text{tr}[\Lambda \Theta(n-1)] + 4\mu^2P_o\Lambda \end{aligned} \quad (10.4.30)$$

which is easier to work with because of the diagonal nature of Λ . For any symmetric and positive definite matrix Θ , we have $|\theta_{ij}(n)|^2 \leq \theta_{ii}\theta_{jj}$. Hence, the convergence of the

diagonal elements ensures the convergence of the off-diagonal elements. This observation and (10.4.30) suggest that to analyze the LMS algorithm, we should extract from (10.4.30) the equations for the diagonal elements

$$\boldsymbol{\theta}(n) \triangleq [\theta_1(n) \ \theta_2(n) \ \cdots \ \theta_M(n)]^T \quad (10.4.31)$$

of $\boldsymbol{\Theta}(n)$ and form a difference equation for the vector $\boldsymbol{\theta}(n)$. Indeed, we can easily show that

$$\boldsymbol{\theta}(n) = \mathbf{B}\boldsymbol{\theta}(n-1) + 4\mu^2 P_o \boldsymbol{\lambda} \quad (10.4.32)$$

where

$$\mathbf{B} \triangleq \boldsymbol{\Lambda}(\rho) + 4\mu^2 \boldsymbol{\lambda} \boldsymbol{\lambda}^T \quad (10.4.33)$$

$$\boldsymbol{\lambda} \triangleq [\lambda_1 \ \lambda_2 \ \cdots \ \lambda_M]^T \quad (10.4.34)$$

$$\boldsymbol{\Lambda}(\rho) \triangleq \text{diag}\{\rho_1, \rho_2, \dots, \rho_M\} \quad (10.4.35)$$

$$\rho_k = 1 - 4\mu\lambda_k + 8\mu^2\lambda_k^2 = (1 - 2\mu\lambda_k)^2 + 4\mu^2\lambda_k^2 > 0 \quad 1 \leq k \leq M \quad (10.4.36)$$

and λ_k are the eigenvalues of \mathbf{R} . The solution of the vector difference equation (10.4.32) is

$$\boldsymbol{\theta}(n) = \mathbf{B}^n \boldsymbol{\theta}(0) + 4\mu^2 P_o \sum_{j=0}^{n-1} \mathbf{B}^j \boldsymbol{\lambda} \quad (10.4.37)$$

and can be easily found by recursion.

The stability of the linear system (10.4.32) is determined by the eigenvalues of the symmetric matrix \mathbf{B} . Using (10.4.33) and (10.4.35), for an arbitrary vector \mathbf{z} , we obtain

$$\mathbf{z}^T \mathbf{B} \mathbf{z} = \mathbf{z}^T \boldsymbol{\Lambda}(\rho) \mathbf{z} + 4\mu^2 (\boldsymbol{\lambda}^T \mathbf{z})^2 = \sum_{k=1}^M \rho_k z_k^2 + 4\mu^2 (\boldsymbol{\lambda}^T \mathbf{z})^2 \quad (10.4.38)$$

where we have used (10.4.36). Hence (10.4.38), for $\mathbf{z} \neq \mathbf{0}$, implies that $\mathbf{z}^T \mathbf{B} \mathbf{z} > 0$, that is, the matrix \mathbf{B} is positive definite. Since matrix \mathbf{B} is symmetric and positive definite, its eigenvalues $\lambda_k(\mathbf{B})$ are real and positive. The system (10.4.37) will be BIBO stable if and only if

$$0 < \lambda_k(\mathbf{B}) < 1 \quad 1 \leq k \leq M \quad (10.4.39)$$

To find the range of μ that ensures (10.4.39), we use the Gershgorin circles theorem (Noble and Daniel 1988), which states that *each eigenvalue of an $M \times M$ matrix \mathbf{B} lies in at least one of the disks with center at the diagonal element b_{kk} and radius equal to the sum of absolute values $|b_{kj}|$, $j \neq k$, of the remaining elements of the row*. Since the elements of \mathbf{B} are positive, we can easily see that

$$\lambda_k(\mathbf{B}) - b_{kk} < \sum_{\substack{j=1 \\ j \neq k}}^M b_{kj} \quad \text{or} \quad \lambda_k(\mathbf{B}) < \rho_k + 4\mu^2 \lambda_k \sum_{i=1}^M \lambda_i$$

using (10.4.33). Hence using (10.4.36), we see the eigenvalues of \mathbf{B} satisfy (10.4.39) if

$$1 - 4\mu\lambda_k + 8\mu^2\lambda_k^2 + 4\mu^2\lambda_k \text{tr} \mathbf{R} < 1$$

$$\text{or} \quad -\mu\lambda_k + 2\mu^2\lambda_k^2 + \mu^2\lambda_k \text{tr} \mathbf{R} < 0$$

which implies that $\mu > 0$ and

$$2\mu < \frac{1}{\lambda_k + \text{tr} \mathbf{R}} < \frac{1}{\text{tr} \mathbf{R}}$$

because $\lambda_k > 0$ for all k . In conclusion, if the adaptation step μ satisfies the condition

$$0 < 2\mu < \frac{1}{\text{tr} \mathbf{R}} \quad (10.4.40)$$

then the system (10.4.37) is stable and therefore the sequence $\boldsymbol{\theta}(n)$ converges.

PROPERTY 10.4.2. When the stability condition (10.4.40) holds, the solution (10.4.37) of the difference equation (10.4.32) can be written as

$$\boldsymbol{\theta}(n) = \mathbf{B}^n[\boldsymbol{\theta}(0) - \boldsymbol{\theta}(\infty)] + \boldsymbol{\theta}(\infty) \quad (10.4.41)$$

where $\boldsymbol{\theta}(0)$ is the initial value and $\boldsymbol{\theta}(\infty)$ is the steady-state value of $\boldsymbol{\theta}(n)$.

Proof. Using the identity

$$\sum_{j=0}^{n-1} \mathbf{B}^j = (\mathbf{I} - \mathbf{B}^n)(\mathbf{I} - \mathbf{B})^{-1} = (\mathbf{I} - \mathbf{B})^{-1} - \mathbf{B}^n(\mathbf{I} - \mathbf{B})^{-1}$$

the solution (10.4.37) can be written as

$$\boldsymbol{\theta}(n) = \mathbf{B}^n[\boldsymbol{\theta}(0) - 4\mu^2 P_o(\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\lambda}] + 4\mu^2 P_o(\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\lambda} \quad (10.4.42)$$

When the eigenvalues of \mathbf{B} are inside the unit circle, we have

$$\lim_{n \rightarrow \infty} \boldsymbol{\theta}(n) \triangleq \boldsymbol{\theta}(\infty) = 4\mu^2 P_o(\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\lambda} \quad (10.4.43)$$

because the first term converges to zero. Substituting (10.4.43) in (10.4.42), we obtain (10.4.41).

Evolution of the mean square error

We next express the MSE as a function of $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$. Using (10.2.10) and (10.2.18), we have

$$e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n) = e_o(n) - \tilde{\mathbf{c}}^H(n-1)\mathbf{x}(n) \quad (10.4.44)$$

where $e_o(n)$ is the optimum filtering error and $\tilde{\mathbf{c}}(n)$ is the coefficient error vector. The (a priori) MSE of the adaptive filter at time n is

$$\begin{aligned} P(n) &\triangleq E\{|e(n)|^2\} \\ &= E\{|e_o(n)|^2\} - E\{\tilde{\mathbf{c}}^H(n-1)\mathbf{x}(n)e_o^*(n)\} - E\{e_o(n)\mathbf{x}^H(n)\tilde{\mathbf{c}}(n-1)\} \\ &\quad + E\{\tilde{\mathbf{c}}^H(n-1)\mathbf{x}(n)\mathbf{x}^H(n)\tilde{\mathbf{c}}(n-1)\} \end{aligned} \quad (10.4.45)$$

Since $\tilde{\mathbf{c}}(n)$ is a random vector, the evaluation of the MSE (10.4.45) requires the correlation between $\mathbf{x}(n)$ and $\tilde{\mathbf{c}}(n-1)$. Using the independence assumptions A1 and A2, we see that the second and third terms in (10.4.45) become zero, as explained before, and the excess MSE is given by the last term

$$P_{\text{ex}}(n) = E\{\tilde{\mathbf{c}}^H(n-1)\mathbf{x}(n)\mathbf{x}^H(n)\tilde{\mathbf{c}}(n-1)\} \quad (10.4.46)$$

If we define the quantities

$$\mathbf{A} \triangleq \tilde{\mathbf{c}}^H(n-1) \quad \text{and} \quad \mathbf{B} \triangleq \mathbf{x}(n)\mathbf{x}^H(n)\tilde{\mathbf{c}}(n-1) \quad (10.4.47)$$

and notice that $\mathbf{AB} = \text{tr}(\mathbf{AB})$ (because \mathbf{AB} is a scalar) and $\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA})$, we obtain

$$\begin{aligned} P_{\text{ex}}(n) &= E\{\text{tr}(\mathbf{AB})\} = E\{\text{tr}(\mathbf{BA})\} = \text{tr}(E\{\mathbf{BA}\}) \\ &= \text{tr}(E\{\mathbf{x}(n)\mathbf{x}^H(n)\}E\{\tilde{\mathbf{c}}(n-1)\tilde{\mathbf{c}}^H(n-1)\}) \end{aligned}$$

because expectation is a linear operation and $\mathbf{x}(n)$ and $\tilde{\mathbf{c}}(n-1)$ have been assumed statistically independent. Therefore, the excess MSE can be expressed as

$$P_{\text{ex}}(n) = \text{tr}[\mathbf{R}\Phi(n-1)] \quad (10.4.48)$$

where $\Phi(n) = E\{\tilde{\mathbf{c}}(n)\tilde{\mathbf{c}}^H(n)\}$ is the correlation matrix of the coefficient error vector. This expression simplifies to

$$P_{\text{ex}}(n) = M\sigma_x^2\sigma_c^2 \quad (10.4.49)$$

if $\mathbf{R} = \sigma_x^2\mathbf{I}$ and $\Phi(n) = \sigma_c^2\mathbf{I}$.

If \mathbf{R} and $\Phi(n)$ are both positive definite, relation (10.4.48) shows that $P_{\text{ex}}(n) > 0$, that is, the MSE attained by the adaptive filter is larger than the optimum MSE P_o of the optimum filter (cost of adaptation).

Next we develop a difference equation for $P_{\text{ex}}(n)$, using, for convenience, the principal coordinate system of the input correlation matrix \mathbf{R} . Since the trace of a matrix remains invariant under an orthogonal transformation, we have

$$P_{\text{ex}}(n) = \text{tr}[\mathbf{R}\Phi(n)] = \text{tr}[\Lambda\Theta(n)] = \lambda^T\theta(n) \quad (10.4.50)$$

where the elements of λ are the eigenvalues of \mathbf{R} and the elements of $\theta(n)$ are the diagonal elements of $\Theta(n)$.

Since the most often observable and important quantity for the operation of an adaptive filter is the MSE, we use our previous results to determine the value of MSE as a function of n , that is, the learning curve of the LMS adaptive filter. To this end, we use the orthogonal decomposition $\mathbf{B} = \mathbf{Q}(\mathbf{B})\Lambda(\mathbf{B})\mathbf{Q}^H(\mathbf{B})$ to express \mathbf{B}^n as

$$\mathbf{B}^n = \mathbf{Q}(\mathbf{B})\Lambda^n(\mathbf{B})\mathbf{Q}^H(\mathbf{B}) = \sum_{k=1}^M \lambda_k^n(\mathbf{B})\mathbf{q}_k(\mathbf{B})\mathbf{q}_k^H(\mathbf{B}) \quad (10.4.51)$$

where $\lambda_k(\mathbf{B})$ are the eigenvalues and $\mathbf{q}_k(\mathbf{B})$ are the eigenvectors of matrix \mathbf{B} . Substituting (10.4.41) and (10.4.51) into (10.4.50) and recalling that $P(n) = P_o + P_{\text{ex}}(n)$, we obtain

$$P(n) = P_o + P_{\text{tr}}(n) + P_{\text{ex}}(\infty) \quad (10.4.52)$$

where $P_{\text{ex}}(\infty)$ is termed the *steady-state excess MSE* and

$$P_{\text{tr}}(n) \triangleq \sum_{k=1}^M \gamma_k(\mathbf{R}, \mathbf{B}) \lambda_k^n(\mathbf{B}) \quad (10.4.53)$$

is termed the *transient MSE* because it dies out exponentially when $0 < \lambda_k(\mathbf{B}) < 1$, $1 \leq k \leq M$. The constants

$$\gamma_k(\mathbf{R}, \mathbf{B}) \triangleq \lambda^T(\mathbf{R})\mathbf{q}_k(\mathbf{B})\mathbf{q}_k^H(\mathbf{B})[\theta(0) - \theta(\infty)] \quad (10.4.54)$$

are determined by the eigenvalues $\lambda_k(\mathbf{R})$ of matrix \mathbf{R} and the eigenvectors $\mathbf{q}_k(\mathbf{B})$ of matrix \mathbf{B} . Since the minimum MSE P_o is available, we need to determine the steady-state excess MSE $P_{\text{ex}}(\infty)$.

PROPERTY 10.4.3. When the LMS adaptive algorithm converges, the steady-state excess MSE is given by

$$P_{\text{ex}}(\infty) = P_o \frac{C(\mu)}{1 - C(\mu)} \quad (10.4.55)$$

$$\text{where } C(\mu) \triangleq \sum_{k=1}^M \frac{\mu \lambda_k}{1 - 2\mu \lambda_k} \quad (10.4.56)$$

and λ_k are the eigenvalues of the input correlation matrix.

Proof. Using (10.4.32) and (10.4.35), we obtain the difference equation

$$\theta_k(n) = \rho_k \theta_k(n-1) + 4\mu^2 \lambda_k P_{\text{ex}}(n-1) + 4\mu^2 P_o \lambda_k \quad (10.4.57)$$

When (10.4.40) holds, (10.4.57) attains the following steady-state form

$$\theta_k(\infty) = \rho_k \theta_k(\infty) + 4\mu^2 \lambda_k P_{\text{ex}}(\infty) + 4\mu^2 P_o \lambda_k$$

whose solution, in conjunction with (10.4.36), gives

$$\theta_k(\infty) = \mu \frac{P_o + P_{\text{ex}}(\infty)}{1 - 2\mu \lambda_k}$$

and
$$P_{\text{ex}}(\infty) = \sum_{k=1}^M \lambda_k \theta_k(\infty) = [P_o + P_{\text{ex}}(\infty)] \sum_{k=1}^M \frac{\mu \lambda_k}{1 - 2\mu \lambda_k}$$

Solving the last equation for $P_{\text{ex}}(\infty)$, we obtain (10.4.55) and (10.4.56).

Solving (10.4.55) for $C(\mu)$ gives

$$C(\mu) = \frac{P_{\text{ex}}(\infty)}{P_o + P_{\text{ex}}(\infty)} \quad (10.4.58)$$

which implies that

$$0 < C(\mu) < 1 \quad (10.4.59)$$

because P_o and $P_{\text{ex}}(\infty)$ are positive quantities. It has been shown that (10.4.59) leads to the tighter bound $0 < 2\mu < 2/(3 \operatorname{tr} \mathbf{R})$ for the adaptation step μ (Horowitz and Senne 1981; Feuer and Weinstein 1985). Therefore, convergence in the MSE imposes a stronger constraint on the step size μ than does (10.4.40), which ensures convergence in the mean.

10.4.3 Summary and Design Guidelines

There are many theoretical and simulation analyses of the LMS adaptive algorithm under a variety of assumptions. In this book, we have focused on results that help us to understand its operation and performance and to develop design guidelines for its practical application. The operation and performance of the LMS adaptive filter are determined by its stability and the properties of its learning curve, which shows the evolution of the MSE as a function of time. The MSE produced by the LMS adaptive algorithm consists of three components [see (10.4.52)]

$$P(n) = P_o + P_{\text{tr}}(n) + P_{\text{ex}}(\infty)$$

where P_o is the optimum MSE, $P_{\text{tr}}(n)$ is the transient MSE, and $P_{\text{ex}}(\infty)$ is the steady-state excess MSE. This equation provides the basis for understanding and evaluating the operation of the LMS adaptive algorithm in a stationary SOE. For convenience, the LMS adaptive filtering algorithm is summarized in Table 10.3.

TABLE 10.3
Summary of the LMS algorithm.

Design parameters
$\mathbf{x}(n)$ = input data vector at time n
$y(n)$ = desired response at time n
$\mathbf{c}(n)$ = filter coefficient vector at time n
M = number of coefficients
μ = step-size parameter
$0 < \mu \ll \frac{1}{\sum_{k=1}^M E\{ x_k(n) ^2\}}$
Initialization
$\mathbf{c}(-1) = \mathbf{x}(-1) = \mathbf{0}$
Computation
For $n = 0, 1, 2, \dots$, compute
$\hat{y}(n) = \mathbf{c}^H(n-1) \mathbf{x}(n)$
$e(n) = y(n) - \hat{y}(n)$
$\mathbf{c}(n) = \mathbf{c}(n-1) + 2\mu \mathbf{x}(n) e^*(n)$

Stability. The LMS adaptive filter converges in the mean-square sense, that is, the transient MSE dies out, if the adaptation step μ satisfies the condition

$$0 < 2\mu < \frac{K}{\text{tr} \mathbf{R}} \quad (10.4.60)$$

where $\text{tr} \mathbf{R}$ is the trace of the input correlation matrix and K is a constant that depends weakly on the statistics of the input data vector. For example, when $\mathbf{x}(n) \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$, we proved that $K = 1$ or $\frac{2}{3}$. In addition, this condition ensures that on average the LMS adaptive filter converges to the optimum filter. We stress that in most practical applications, where the independence assumption does not hold, the *step size* μ should be much smaller than $K / \text{tr} \mathbf{R}$. Therefore, the exact value of K is not important in practice.

Rate of convergence. The transient MSE dies out exponentially without exhibiting any oscillations. This follows from (10.4.53) because when μ satisfies (10.4.40), the eigenvalues of matrix \mathbf{B} are positive and less than 1. The *settling time*, that is, the time taken for the transients to die out, is proportional to the average time constant

$$\tau_{\text{lms,av}} = \frac{1}{\mu \lambda_{\text{av}}} \quad (10.4.61)$$

where $\lambda_{\text{av}} = (\sum_{k=1}^M \lambda_k)/M$ is the average eigenvalue of \mathbf{R} (Widrow et al. 1976). The quantity $P_{\text{tr}}^{\text{total}} = \sum_{n=0}^{\infty} P_{\text{tr}}(n)$, which provides the total transient MSE, can be used as a measure for the speed of adaptation. When $\mu \lambda_k \ll 1$ (see Problem 10.12), we have

$$P_{\text{tr}}^{\text{total}} \triangleq \sum_{n=0}^{\infty} P_{\text{tr}}(n) \simeq \frac{1}{4\mu} \sum_{k=1}^M \Delta\theta_k(0) \quad (10.4.62)$$

where $\Delta\theta_k(0)$ is the initial distance of a coefficient from its optimum setting measured in principal coordinates. As is intuitively expected, *the smaller the step size and the farther the initial coefficients are from their optimum settings, the more iterations it takes for the LMS algorithm to converge*. Furthermore, from the discussion in Section 10.3, it follows that the LMS algorithm will converge faster if the contours of the error surface are circles, that is, when the input correlation matrix is $\mathbf{R} = \sigma_x^2 \mathbf{I}$.

Steady-state excess MSE. The excess MSE after the adaptation has been completed (i.e., the steady-state value) is given by (10.4.55). When $\mu \lambda_k \ll 1$, we may approximate (10.4.55) as follows

$$P_{\text{ex}}(\infty) \simeq P_o \frac{\mu \text{tr} \mathbf{R}}{1 - \mu \text{tr} \mathbf{R}}$$

which allows a much easier interpretation. Solving for $\mu \text{tr} \mathbf{R}$, we obtain $\mu \text{tr} \mathbf{R} \simeq P_{\text{ex}}(\infty) / [P_{\text{ex}}(\infty) + P_o]$ which implies that $0 < \mu \text{tr} \mathbf{R} < 1$. Since $\mu \text{tr} \mathbf{R} \ll 1$, we often use the approximation

$$P_{\text{ex}}(\infty) \simeq \mu P_o \text{tr} \mathbf{R} \quad (10.4.63)$$

which implies that $P_{\text{ex}}(\infty) \ll P_o$, that is, for small values of the step size the excess MSE is much smaller than the optimum MSE. Note that the presence of the irreducible error $e_o(n)$ prevents perfect adaptation as $n \rightarrow \infty$ because $P_o > 0$.

Speed versus quality of adaptation. From the previous discussion we see that there is a tradeoff between rate of convergence (speed of adaptation) and steady-state excess MSE (quality of adaptation, or accuracy of the adaptive filter). The first requirement for an adaptive filter is stability, which is ensured by choosing μ to satisfy (10.4.60). Within this range, decreasing μ to reduce the desired level of misadjustment, according to (10.4.63),

decreases the speed of convergence; see (10.4.62). Conversely, if μ is increased to increase the speed of convergence; this results in an increase in misadjustment. This tradeoff between speed of convergence and misadjustment is a fundamental feature of the LMS algorithm.

FIR filters. In this case, the input is a stationary process $x(n)$ with a Toeplitz correlation matrix \mathbf{R} . Therefore, we have

$$\text{tr } \mathbf{R} = Mr(0) = ME\{|x(n)|^2\} = MP_x \quad (10.4.64)$$

where MP_x is called the *tap input power*. Substituting (10.4.40) into (10.4.64), we obtain

$$0 < 2\mu < \frac{1}{MP_x} = \frac{1}{\text{tap input power}} \quad (10.4.65)$$

which shows that the selection of the step size depends on the input power. Using (10.4.63) and (10.4.64), we see that misadjustment \mathcal{M} is given by

$$\mathcal{M} = \frac{P_{\text{ex}}(\infty)}{P_o} \simeq \mu MP_x \quad (10.4.66)$$

which shows that for given M and P_x the value of misadjustment is proportional to μ . We emphasize that the misadjustment provides a measure of how close an LMS adaptive filter is to the corresponding optimum filter.

The statistical properties of the SOE, that is, the correlation of the input signal and the cross-correlation between input and desired response signals, play a key role in the performance of the LMS adaptive filter.

- First, we should make sure that the relation between $x(n)$ and $y(n)$ can be accurately modeled by a linear FIR filter with M coefficients. Inadequacy of the FIR structure, output observation noise, or lack of correlation between $x(n)$ and $y(n)$ increases the magnitude of the irreducible error. If M is very large, we may want to use a pole-zero IIR filter (Shynk 1989; Treichler et al. 1987). If the relationship between $x(n)$ and $y(n)$ is nonlinear, we certainly need a nonlinear filtering structure (Mathews 1991).
- The LMS algorithm uses a “noisy” instantaneous estimate of the gradient vector. However, when the correlation between input and desired response is weak, the algorithm should make more cautious steps (“wait and average”). Such algorithms update their coefficients every L samples, using all samples between successive updatings to determine the gradient (gradient averaging).
- The eigenvalue structure of \mathbf{R} as measured by its eigenvalue spread ($\lambda_{\max}/\lambda_{\min}$) or equivalently by the *spectral flatness measure (SFM)* (see Section 4.1) has a strong effect on the rate of convergence of the LMS algorithm. In general, the rate of convergence decreases as the eigenvalue spread increases, that is, as the contours of the cost function become more elliptical, or equivalently the input spectrum becomes more nonwhite.

Normalized LMS algorithm. According to (10.4.60), the selection of μ in practical applications is complicated because the power of the input signal either is unknown or varies with time. This problem can be addressed by using the *normalized LMS (NLMS)* algorithm [see (10.4.11)]

$$\mathbf{c}(n) = \mathbf{c}(n-1) + \frac{\tilde{\mu}}{E_M(n)} \mathbf{x}(n) e^*(n) \quad (10.4.67)$$

where $E_M(n) = \|\mathbf{x}(n)\|^2$ and $0 < \tilde{\mu} < 1$. It can be shown that the NLMS algorithm converges in the mean square if $0 < \tilde{\mu} < 1$ (Rupp 1993; Slock 1993), which makes the selection of the step size $\tilde{\mu}$ much easier than the selection of μ in the LMS algorithm.

For FIR filters, the quantity $E_M(n)$ provides an estimate of $ME\{|x(n)|^2\}$ and can be computed recursively by using the sliding-window formula

$$E_M(n) = E_M(n-1) + |x(n)|^2 - |x(n-M)|^2 \quad (10.4.68)$$

where $E_M(-1) = 0$ or a first-order recursive filter estimator. In practice, to avoid division by zero, if $\mathbf{x}(n) = \mathbf{0}$, we set $E_M(n) = \delta + \|\mathbf{x}(n)\|^2$, where δ is a small positive constant.

Other approaches and analyses. The analysis of the LMS algorithm presented in this section is simple, clarifies its performance, and provides useful design guidelines. However, there are many other approaches, which are beyond the scope of this book, that differ in terms of complexity, accuracy, and objectives. Major efforts to remove the independence assumption and replace it with the more realistic statistically dependent input assumption are documented in Macchi (1995), Solo (1997), and Butterweck (1995) and the references therein. Convergence analysis of the LMS algorithm using the *stochastic approximation approach* and a deterministic approach using the *method of ordinary differential equations* are discussed in Solo and Kong (1995), Sethares (1993), and Benveniste et al. (1987). Other types of analyses deal with the determination of the probability densities and the probability of large excursions of the adaptive filter coefficients for various types of input signals (Rupp 1995). The analysis of the convergence properties of the LMS algorithm and its variations is still an active area of research, and new results appear continuously.

10.4.4 Applications of the LMS Algorithm

We now discuss three practical applications in which the LMS algorithm has made a significant impact. In the first case, we consider the previously discussed linear prediction problem and compare the performance of the LMS algorithm with that of the SDA. Table 10.4 provides a summary of the key differences between the SDA and the LMS algorithms. In the second case, we study echo cancelation in full-duplex data transmission, which employs the LMS algorithm in its implementation. In the third case, we discuss the application of adaptive equalization, which is used to minimize intersymbol interference (ISI) in a dispersive channel environment.

TABLE 10.4
Comparison between the SDA and LMS algorithms.

SDA	LMS
Deterministic algorithm: $\lim_{n \rightarrow \infty} \mathbf{c}(n) = \mathbf{c}_o$	Stochastic algorithm: $\lim_{n \rightarrow \infty} E\{\mathbf{c}(n)\} = \mathbf{c}_o$
If converges, it terminates to \mathbf{c}_o	If converges, it fluctuates about \mathbf{c}_o The size of fluctuations is proportional to μ
Noiseless gradient estimate	Noisy gradient estimate
Deterministic steps	Random steps
We can only compare the ensemble average behavior of LMS with the SDA.	

Linear prediction

In Example 10.3.1, the AR(2) model given in (10.3.25) was considered, and the SDA was used to determine the corresponding linear predictor coefficients. We also analyzed the performance of the SDA. In the following example, we perform a similar acquisition of predictor coefficients using the LMS algorithm, and we study the effects of the eigenvalue spread of the input correlation matrix on the convergence of the LMS adaptive algorithm when it is used to update the coefficients.

EXAMPLE 10.4.1. The second-order system in (10.3.25) is repeated here, which generates the signal $x(n)$:

$$x(n) + a_1 x(n - 1) + a_2 x(n - 2) = w(n)$$

where $w(n) \sim \text{WGN}(0, \sigma_w^2)$ and where the coefficients are selected from Table 10.2 for two different eigenvalue spreads. A Gaussian pseudorandom number generator was used to obtain 1000 realizations of $x(n)$ using each set of parameter values given in Table 10.2. These sample realizations were used for statistical analysis.

The second-order LMS adaptive predictor with coefficients $\mathbf{c}(n) = [c_1(n) \ c_2(n)]^T$ is given by [see (10.4.12)]

$$e(n) = x(n) - c_1(n-1)x(n-1) - c_2(n-2)x(n-2) \quad n \geq 0$$

$$c_1(n) = c_1(n-1) + 2\mu e(n)x(n-1)$$

$$c_2(n) = c_2(n-1) + 2\mu e(n)x(n-2)$$

where μ is the step-size parameter. The adaptive predictor was initialized by setting $x(-1) = x(-2) = 0$ and $c_1(-1) = c_2(-1) = 0$. The above adaptive predictor was implemented with $\mu = 0.04$, and the predictor coefficients as well as the MSE were recorded for each realization. These quantities were averaged to study the behavior of the LMS algorithm. These calculations were repeated for $\mu = 0.01$.

In Figure 10.20 we show several plots obtained for $\mathcal{X}(\mathbf{R}) = 1.22$. In plot (a) we show the ensemble averaged trajectory $\{\mathbf{c}(n)\}_{n=0}^{150}$ superimposed on the MSE contours. A trajectory of a simple realization is also shown to illustrate its randomness. In plot (b) the $\mathbf{c}(n)$ learning curve for the averaged value as well as for one single realization is shown. In plot (c) the corresponding learning curves for the MSE are depicted. Finally, in plot (d) we show the effect of step size μ on the MSE learning curve. Similar plots are shown in Figure 10.21 for $\mathcal{X}(\mathbf{R}) = 10$.

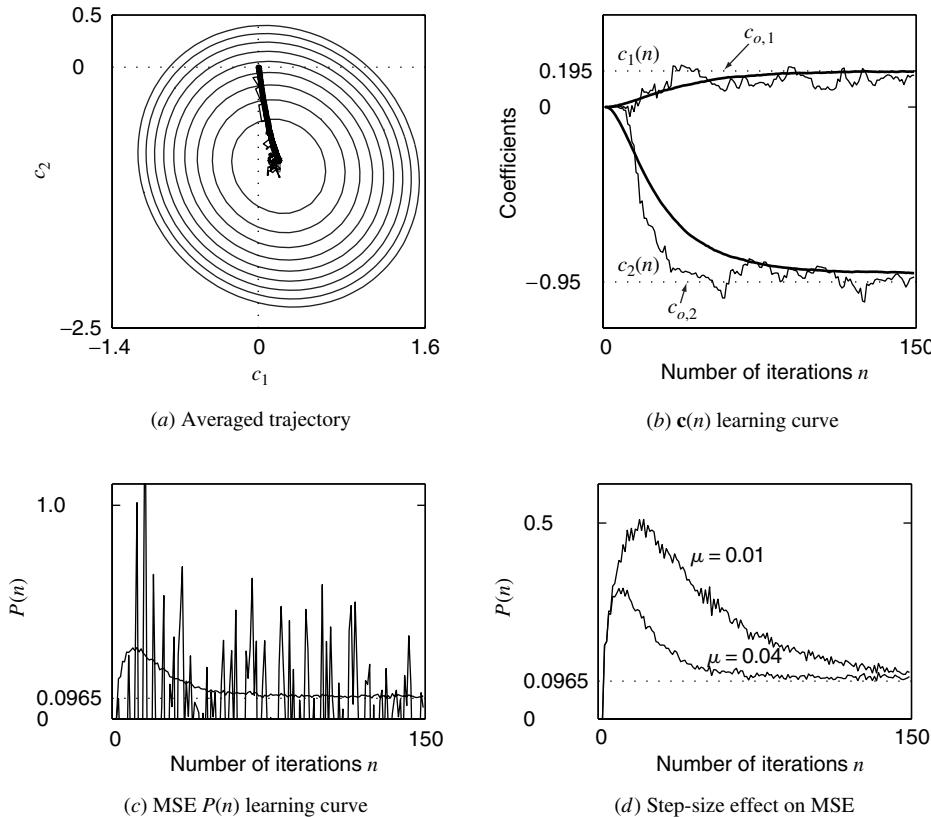
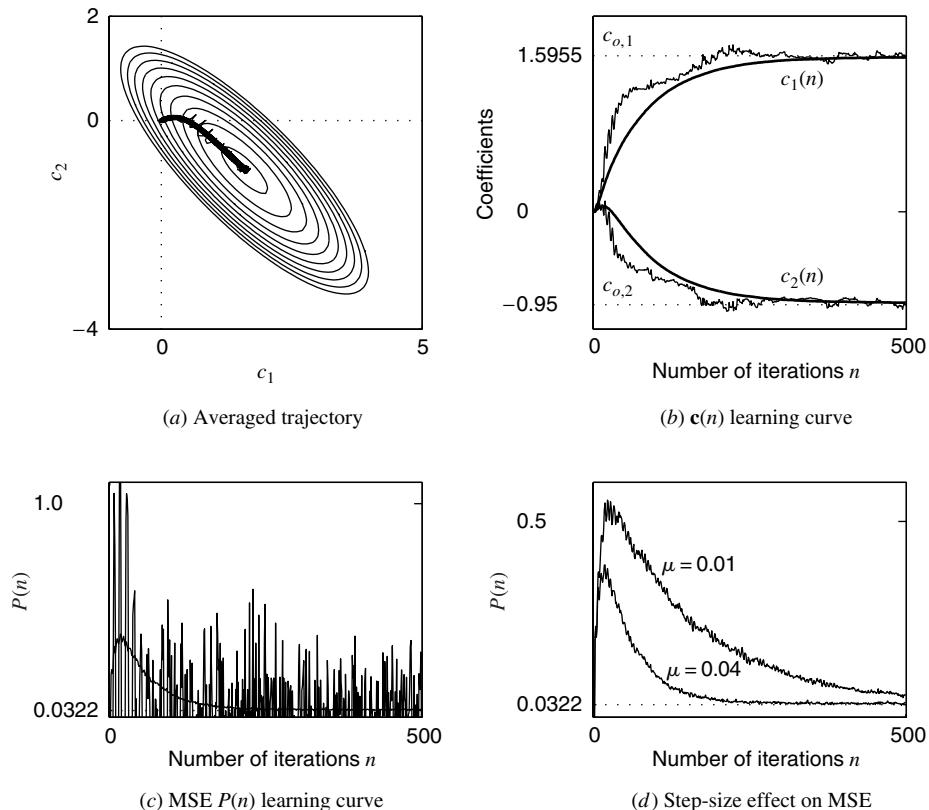


FIGURE 10.20

Performance curves for the LMS used in the linear prediction problem with step-size parameter $\mu = 0.04$ and eigenvalue spread $\mathcal{X}(\mathbf{R}) = 1.22$.

**FIGURE 10.21**

Performance curves for the LMS used in the linear prediction problem with step-size parameter $\mu = 0.04$ and eigenvalue spread $\mathcal{X}(\mathbf{R}) = 10$.

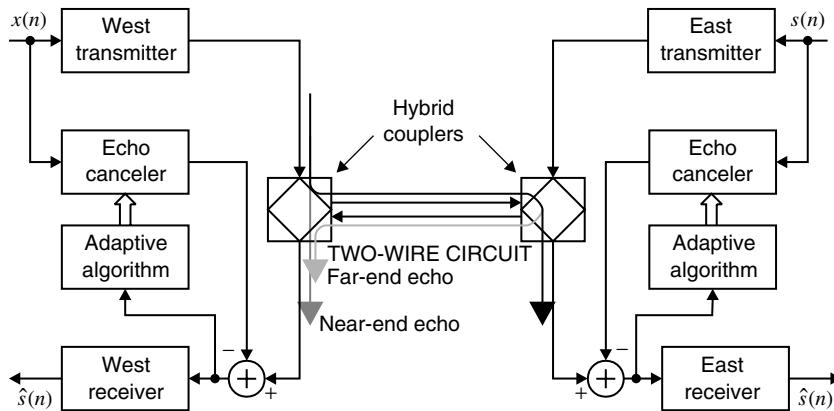
Several observations can be made from these plots:

- The trajectories and the learning curves for a simple realization are clearly random or “noisy,” while the averaging over the ensemble clearly has a smoothing effect.
- The averaged quantities (coefficients and the MSE) converge to the true values, and this convergence rate is in accordance with theory.
- The rate of convergence of the LMS algorithm depends on the step size μ . The smaller the step size, the slower the rate.
- The rate of convergence also depends on the eigenvalue spread $\mathcal{X}(\mathbf{R})$. The larger the spread, the slower the rate. For $\mathcal{X}(\mathbf{R}) = 1.22$, the algorithm converges in about 150 steps while for $\mathcal{X}(\mathbf{R}) = 10$ it requires about 500 steps.

Clearly these observations compare well with the theory.

Echo cancellation in full-duplex data transmission

Figure 10.22 illustrates a system that achieves simultaneous data transmission in both directions (full-duplex) over two-wire circuits using the special two-wire to four-wire interfaces (called *hybrid couplers*) that exist in any telephone set. Although the hybrid couplers are designed to provide perfect isolation between transmitters and receivers, this is not the case in practical systems. As a result, (1) one part of the transmitted signal leaks through the near-end hybrid to its own receiver (near-end echo), and (2) another part is reflected by the far-end hybrid and ends up at its own receiver (far-end echo). The combined echo signal, which can be 30 dB stronger than the signal received from the other end, increases the number of errors. We note that in contrast with acoustic echo cancelation, the delay of echoes in data transmission is immaterial.

**FIGURE 10.22**

Model of a full-duplex data transmission system that uses an echo canceler in the modems.

The best way to address this problem is to form a replica of the echo and then subtract it from the incoming signal. We can model the echoes as the result of an “echo” path between the transmitter and the receiver. For baseband data transmission this echo path is basically linear and varies very slowly with time. Therefore, we can obtain a replica of the echo signal using an FIR LMS adaptive filter (*echo canceler*), as shown in Figure 10.22. The inclusion of the transmitter in the echo path, as long as it involves linear operations, simplifies the implementation and improves the speed of adaptation because the input is an IID binary data sequence of values +1 and -1 with equal probability (Verhoeckx et al. 1979).

Referring to Figure 10.23, if we assume that the echo path has an FIR impulse response, the echo signal is given by

$$y(n) = \mathbf{c}_o^T \mathbf{x}(n) \quad (10.4.69)$$

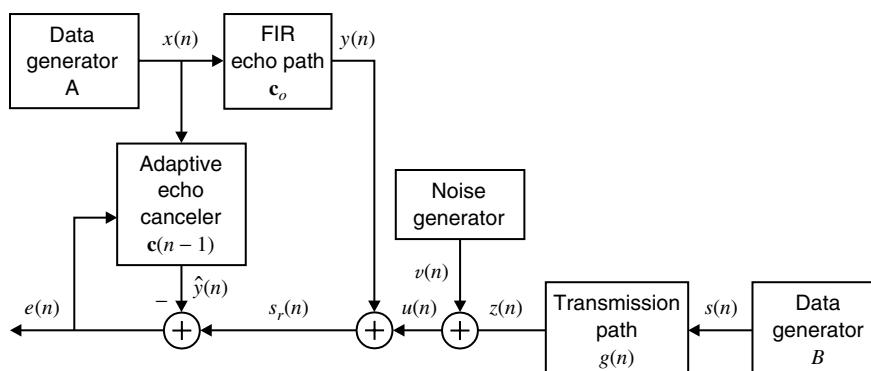
where

$$\mathbf{c}_o = [c_o(0) \ c_o(1) \ \cdots \ c_o(M-1)]^T$$

If $g(n)$ is the impulse response of the transmission path from the far-end transmitter to the near-end receiver, the received signal is given by

$$s_r(n) = y(n) + z(n) + v(n) \triangleq y(n) + u(n) \quad (10.4.70)$$

$$z(n) = \sum_{k=0}^{\infty} g(k)s(n-k)$$

**FIGURE 10.23**

Block diagram of a system for investigating the performance of adaptive echo canceler.

where $s(n)$ is the transmitted data signal and $v(n) \sim \text{WGN}(0, \sigma_v^2)$ is additive noise. The signal $u(n) = z(n) + v(n)$ represents the “uncancelable” signal because it cannot be removed by the canceler.

The LMS adaptive echo canceler is given by

$$\hat{y}(n) = \mathbf{c}^T(n-1)\mathbf{x}(n) \quad (10.4.71)$$

$$e(n) = y(n) - \hat{y}(n) \quad (10.4.72)$$

$$\mathbf{c}(n) = \mathbf{c}(n-1) + 2\mu e(n)\mathbf{x}(n) \quad (10.4.73)$$

where μ is the adaptation step size. The adaptive filter takes advantage of the fact that $x(n)$ is correlated with $y(n)$ but uncorrelated with $s(n)$ and $v(n)$.

The residual (uncanceled) echo is

$$e_r(n) \triangleq y(n) - \hat{y}(n) = [\mathbf{c}_o - \mathbf{c}(n-1)]^T \mathbf{x}(n) \triangleq -\tilde{\mathbf{c}}^T(n-1)\mathbf{x}(n) \quad (10.4.74)$$

and if we assume that $\tilde{\mathbf{c}}(n-1)$ and $\mathbf{x}(n)$ are independent, then

$$P_r(n) = E\{e_r^2(n)\} = E\{\tilde{\mathbf{c}}^T(n-1)\tilde{\mathbf{c}}(n-1)\}$$

because $\mathbf{R} = E\{\mathbf{x}(n)\mathbf{x}^T(n)\} = \mathbf{I}$. Using (10.4.69), (10.4.71), and (10.4.72), we can easily show that

$$\tilde{\mathbf{c}}(n) = \tilde{\mathbf{c}}(n-1) - 2\mu\mathbf{x}(n)\mathbf{x}^T(n)\tilde{\mathbf{c}}(n-1) + 2\mu\mathbf{x}(n)u(n) \quad (10.4.75)$$

If we premultiply (10.4.75) by its transpose and take the mathematical expectation, we obtain

$$P_r(n+1) = (1 - 4\mu + 4\mu^2 M)P_r(n) + 4\mu M\sigma_u^2 \quad (10.4.76)$$

using the independence assumption and the relation $\mathbf{x}^T(n)\mathbf{x}(n) = M$. The solution of (10.4.76), in terms of the residual echo ratio $P_r(n)/\sigma_u^2$, is

$$\frac{P_r(n)}{\sigma_u^2} = (1 - 4\mu + 4\mu^2 M)^n \left[\frac{P_r(0)}{\sigma_u^2} - \frac{\mu M}{1 - \mu M} \right] + \frac{\mu M}{1 - \mu M} \quad (10.4.77)$$

and describes completely the operation of the LMS adaptive echo canceler. Indeed, we draw the following conclusions:

1. The algorithm converges if

$$|1 - 4\mu + 4\mu^2 M| < 1 \quad \text{or} \quad 0 < \mu < \frac{1}{M} \quad (10.4.78)$$

which agrees with (10.4.40) because $\text{tr}\mathbf{R} = M$.

2. After convergence we have

$$P_r(\infty) = \frac{\mu M}{1 - \mu M} \sigma_u^2 \simeq \mu M \sigma_u^2 \quad (10.4.79)$$

which again is in agreement with (10.4.63).

3. If $P_r(n)/\sigma_u^2 \gg \mu M/(1 - \mu M)$, we have

$$\frac{P_r(n)}{P_r(0)} \simeq (1 - 4\mu + 4\mu^2 M)^n \quad (10.4.80)$$

which can be used to find out how many iterations are required for a given echo reduction. For example, we can easily show that to achieve a 20-dB echo reduction requires $n_{20} \simeq 1.15/\mu$ iterations.

From the previous discussion, it should be clear that the step size μ plays a crucial role in the performance of the adaptive echo canceler because it determines both the rate of convergence and the minimum residual echo cancelation that can be attained. Furthermore, we clearly see the tradeoff between fast adaptation and residual echo power.

EXAMPLE 10.4.2. Consider the system shown in Figure 10.23 for investigating the performance of the LMS algorithm in adaptive echo cancelation and to verify the above conclusions. The data generators A (in modem A) and B (in modem B) output symbols $+1$ or -1 with equal probability (i.e., Bernoulli sequence). The FIR filter following data generator A models the echo path, which is assumed to be

$$c_o(n) = -\frac{5}{3}(\frac{1}{2})^n + \frac{8}{3}(\frac{4}{5})^n \quad 0 \leq n \leq M - 1$$

where $M = 20$ is the total length of the echo path. The filter following data generator B models the transmission path between the far-end transmitter and the near-end receiver, which we will assume to be

$$g(n) = \frac{4}{5}(\frac{3}{5})^n \quad n \geq 0$$

The noise generator is a white Gaussian source with $\sigma_v^2 = 1$ and models the transmission noise. Using the equations $\sigma_y^2 = \sum_{k=0}^{N-1} c_o^2(k)$ and $\sigma_u^2 = \sum_{k=0}^{\infty} g^2(k) + \sigma_v^2$, we scale $u(n)$ so that $10 \log(\sigma_y^2/\sigma_u^2) = 30$ dB. The adaptive echo canceler employs the LMS algorithm with $c(0) = \mathbf{0}$. We perform Monte Carlo simulations on this system. Figure 10.24 shows the residual echo ratio $P_r(n)/\sigma_u^2$ evaluated by ensemble averaging over 200 independent trials of the experiment, for two different step sizes in the LMS algorithm [which satisfy (10.4.78)], superimposed on the corresponding theoretical curves computed by using (10.4.79) and (10.4.80). Clearly, the simulations support the theoretical results quite accurately. More detailed discussions of adaptive echo cancelation techniques for both baseband and passband data transmission systems can be found in Gitlin et al. (1992) and in Ling (1993a).

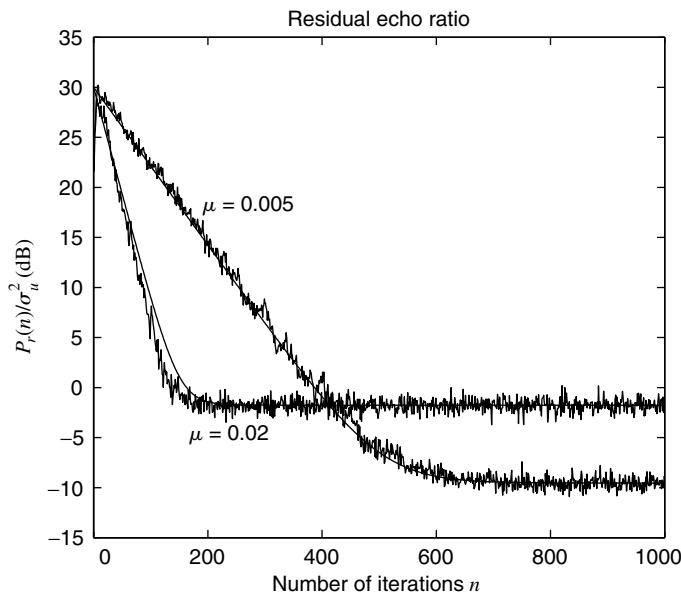


FIGURE 10.24
 Performance analysis of the LMS algorithm in the adaptive echo cancelation that clearly shows the tradeoff between rate of convergence and residual echo power.

Adaptive equalization

In Section 6.8, we discussed the theory and implementation of channel equalization in data transmission systems. When data are transmitted below 2400 bits/s, the ISI is relatively small and does not pose a problem in the operation of a modem. However, for high-speed communication over 2400 bits/s, an equalizer is needed in the modem to compensate for the channel distortion. Since channel characteristics are generally unknown and time-varying,

an adaptive algorithm is required that leads to adaptive equalization. Figure 10.25 describes an application of adaptive filtering to adaptive channel equalization. Initially, coefficients of the equalizer are adjusted, by means of the LMS algorithm, by transmitting a known training sequence of short duration. After this short training period, the actual data sequence $\{y(n)\}$ is transmitted. The slow variation in channel characteristics is then continuously tracked by adjusting coefficients of the equalizer, using the decisions in place of the known training sequence. This approach works well when decision errors are infrequent.

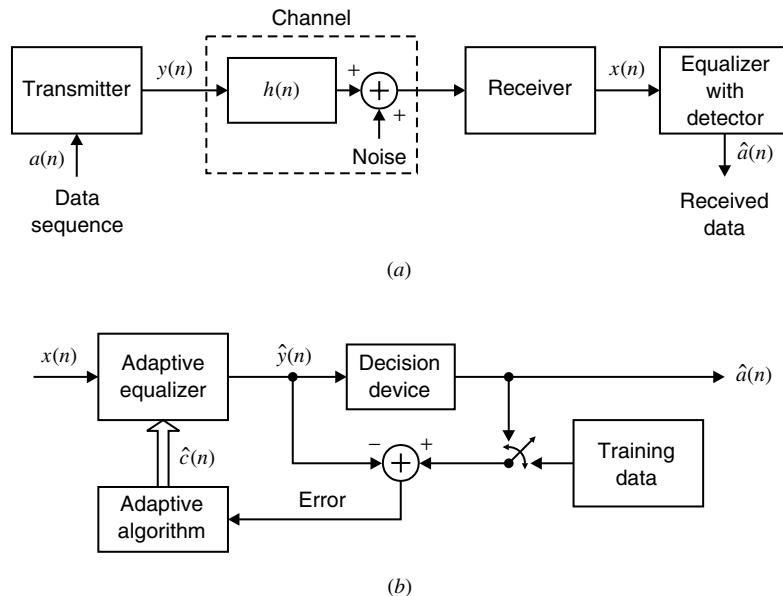


FIGURE 10.25
Model of an adaptive equalizer in a data transmission system.

EXAMPLE 10.4.3. Figure 10.26 shows the block diagram of the system used in the experimental investigation of the performance of the LMS algorithm used in the adaptive equalizer. The data source generates Bernoulli sequence $\{y(n)\}$ with symbols +1 and -1 having zero mean and unit variance. The channel following the source is modeled by the raised cosine impulse response

$$h(n) = \begin{cases} 0.5 \left\{ 1 + \cos \left[\frac{2\pi}{W}(n-2) \right] \right\} & n = 1, 2, 3 \\ 0 & \text{otherwise} \end{cases} \quad (10.4.81)$$

where parameter W is used to control the amount of channel distortion. The amount of channel distortion increases with W . The random noise generator outputs white Gaussian sequence $v(n)$ which models the noise in the channel. The equalizer input is

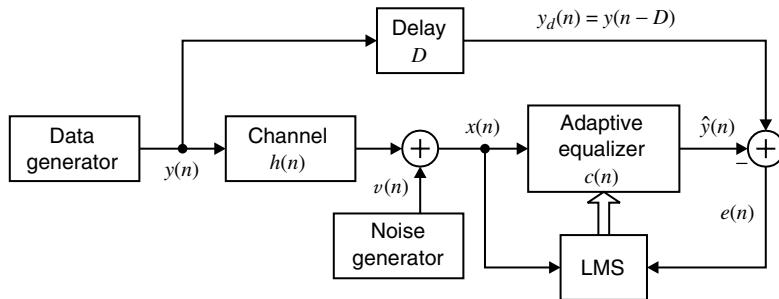
$$x(n) = \sum_{k=1}^3 h(k)y(n-k) + v(n) \quad (10.4.82)$$

Since $y(n)$ is an independent sequence and since $v(n)$ is uncorrelated with $y(n)$, the maximum lag that produces nonzero correlation is 2. Thus the correlation of $x(n)$ is given by

$$r_x(0) = h^2(1) + h^2(2) + h^2(3) + \sigma_v^2$$

$$r_x(1) = h(1)h(2) + h(2)h(3)$$

$$r_x(2) = h(1)h(3)$$

**FIGURE 10.26**

Block diagram of a system for investigating the performance of an adaptive equalizer.

from which an $M \times M$ autocorrelation matrix \mathbf{R} can be constructed for an equalizer of length M . Clearly, parameter W also controls the eigenvalues of \mathbf{R} and hence the ratio $\mathcal{X}(\mathbf{R})$. The design of an MSE equalizer has been discussed in Example 6.8.1. Here we study the performance of the corresponding LMS adaptive equalizer.

The training signal $y(n)$ is delayed by an amount equal to the combined delay introduced by the channel and the equalizer for the desired signal. The impulse response $h(n)$ in (10.4.81) is symmetric with respect to $n = 2$, and assuming that the equalizer is a linear-phase FIR filter, the total delay is equal to $\Delta = (M - 1)/2 + 2$. The error signal $e(n) = y(n - \Delta) - \hat{y}(n)$ is used along with $x(n)$ to implement the LMS algorithm in the adaptive equalizer with $\mathbf{c}(0) = \mathbf{0}$. We performed Monte Carlo simulations using 100 realizations of random sequences with $M = 11$; $\Delta = 7$; $\sigma_v^2 = 0.001$; $W = 2.9$ and $W = 3.5$; and $\mu = 0.01, 0.04$, and 0.08 . The results are shown in Figures 10.27 and 10.28.

Effect of eigenvalue spread. Performance plots of the LMS algorithm for $W = 2.9$ and $W = 3.5$ are shown in Figure 10.27. In plot (a) we depict MSE learning curves from which we observe that the convergence rate of the MSE decreases with W [or equivalently with increase in $\mathcal{X}(\mathbf{R})$], which is to be expected. The steady-state error, on the other hand, increases with W . In plots (b) and (c) we show the ensemble averaged equalizer coefficients. Clearly, the responses are symmetric with respect to $n = 5$ as assumed. Also equalizer coefficients converge to different inverses due to changes in the channel characteristics.

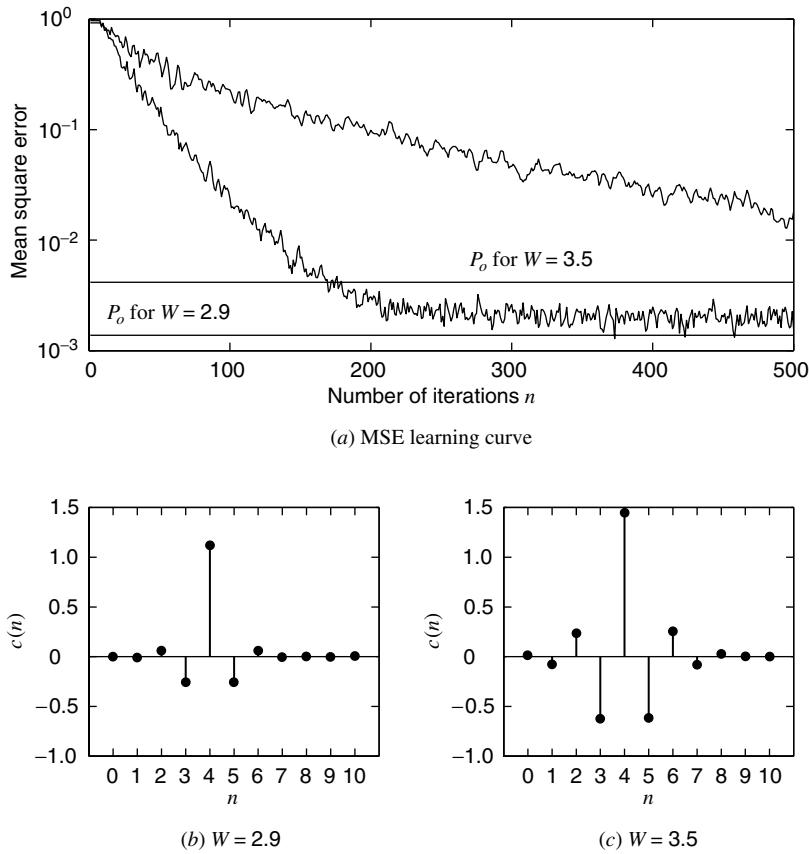
Effect of step size μ . In Figure 10.28 we show the MSE learning curves obtained for $W = 2.9$ and with three different step-size parameter values of $0.01, 0.04$, and 0.08 . It indicates that μ affects the rate of convergence as well as the steady-state value. For $\mu = 0.08$, the algorithm converges in about 100 iterations but has higher steady-state value than the case for $\mu = 0.04$, which requires about 275 iterations for convergence. For $\mu = 0.01$ more than 500 iterations are needed. Finally, Figure 10.29 shows sample realizations of the transmitted, received, and equalized sequences using the discussed LMS equalizer.

10.4.5 Some Practical Considerations

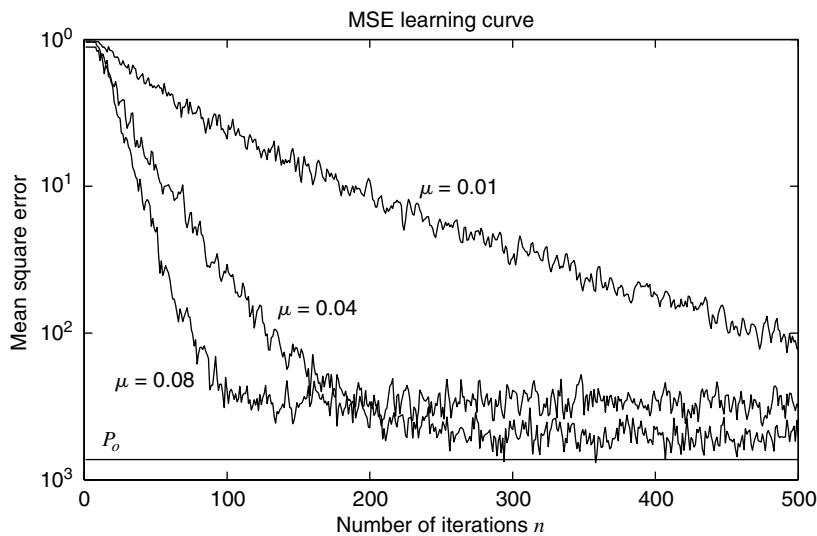
The LMS is the most widely known and used adaptive algorithm because of its simplicity and robustness to disturbances and model errors. We next discuss some issues related to its robustness, finite-word-length effects, and implementation.

Robustness

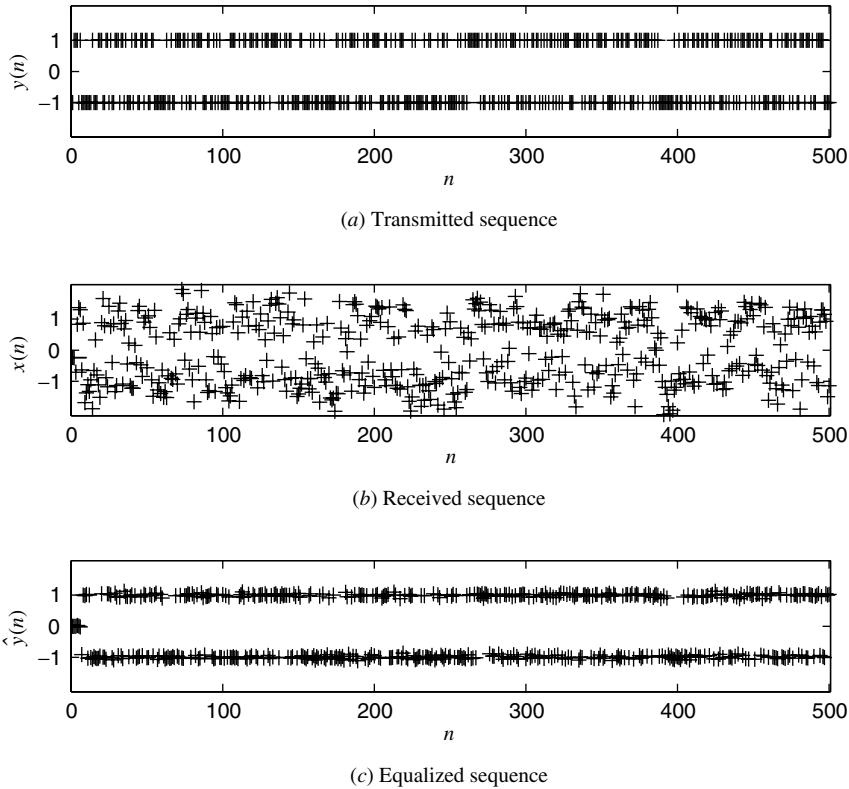
If we assume the model in Figure 10.19, an adaptive filter is said to be *robust* if the effect of the disturbances $\{\mathbf{c}(-1), e_o(n)\}$ on the resulting estimation errors $\{\tilde{\mathbf{c}}(n), e(n)\}$ (or $\{\tilde{\mathbf{c}}(n), \varepsilon(n)\}$), as measured by their energy, is small (Sayed and Rupp 1998). Basically a robust adaptive filter should be insensitive to the initial conditions $\mathbf{c}(-1)$ and the optimum

**FIGURE 10.27**

Performance analysis curves of the LMS algorithm in the adaptive equalizer:
 $\mu = 0.04$.

**FIGURE 10.28**

MSE learning curves of the LMS algorithm in the adaptive equalizer: $W = 2.9$.

**FIGURE 10.29**

Sample realizations of the transmitted, received, and equalized sequences using an FIR LMS equalizer.

residual error $e_o(n)$, which acts as measurement noise. These inputs are collectively called *disturbances*. In practice, $e_o(n)$ accounts not only for measurement noise but also for model mismatching, quantization errors, and other inaccuracies.

If we define the energies of the disturbances and the estimation errors by

$$E_{\text{dist}}(n) = \frac{1}{2\mu} \|\tilde{\mathbf{c}}(-1)\|^2 + \sum_{j=0}^n |e_o(n)|^2 \quad (10.4.83)$$

and

$$E_{\text{error}}(n) = \frac{1}{2\mu} \|\tilde{\mathbf{c}}(n)\|^2 + \sum_{j=0}^n |\tilde{y}(n)|^2 \quad (10.4.84)$$

it can be shown that the coefficient vectors determined by the LMS algorithm satisfy the condition

$$E_{\text{error}}(n) \leq E_{\text{dist}}(n) \quad (10.4.85)$$

assuming that $0 < 2\mu \leq 1/\|\mathbf{x}(n)\|^2$ (Sayed and Kailath 1994; Sayed and Rupp 1996). Equation (10.4.85) shows that the energy of the residuals is always upper-bounded by the energy of the disturbances, which explains the robust behavior of the LMS algorithm.

Furthermore, it can be shown that the LMS algorithm minimizes the maximum possible difference between these two energies, over all disturbances with finite energy, and is optimum according to the H^∞ (or minimax) criterion (Sayed and Rupp 1998; Hassibi et al. 1996).

Finite-precision effects

When we design an LMS adaptive filter for a stationary SOE, we choose the step size μ to provide the desired balance between speed of convergence and misadjustment. If we are not concerned about fast convergence, we can reduce μ so much as to obtain practically insignificant misadjustment. However, in a digital implementation, the adaptation of the LMS algorithm stops (stalls) when the correction term becomes smaller in magnitude than one-half of the least significant bit (LSB), that is,

$$|2\mu e^*(n)x(n-k)| \leq \frac{\text{LSB}}{2} \quad (10.4.86)$$

Therefore, a decrease in μ may result in a performance degradation, unless we increase the number of bits (i.e., the precision) of the filter coefficients. If X_{rms} is the root mean square (rms) amplitude of the input signal, to a good approximation we have

$$|e(n)| \leq \frac{\text{LSB}}{4\mu X_{\text{rms}}} \triangleq \text{DRE} \quad (10.4.87)$$

where DRE is known as the *digital residual error* (Gitlin et al. 1973). We note that for a given number of bits the DRE increases as we decrease the step size μ .

The roundoff numerical errors contribute to the steady-state EMSE a term that is inversely proportional to μ , whereas the quantization of the input data and the filter output contributes a second term that is independent of the step size (Caraiscos and Liu 1984). Hence, in practice the step size of the LMS algorithm cannot be decreased below the level where the degradation effects of quantization and finite-precision arithmetic become significant. Also, the finite-precision effects become more pronounced as the ill conditioning of the input increases (Alexander 1987).

When one or more eigenvalues of the input correlation matrix are zero, the corresponding adaptation modes either do not converge or may result in overflow due to nonlinear quantization effects (Gitlin et al. 1982). These effects can be prevented by using a technique known as *leakage*. The *leaky* LMS algorithm is given by

$$\mathbf{c}(n) = (1 - \gamma\mu)\mathbf{c}(n-1) + \mu e^*(n)\mathbf{x}(n) \quad (10.4.88)$$

where γ is the leakage coefficient. Since μ and γ are very small positive constants, $1 - \gamma\mu$ is slightly less than 1. The updating (10.4.88) is obtained by minimizing the cost function

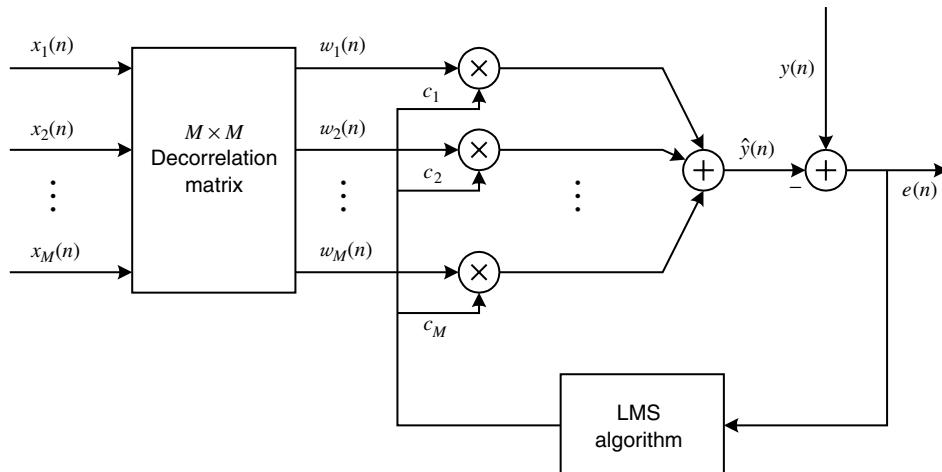
$$P(n) = |e(n)|^2 + \gamma \|\mathbf{c}(n)\|^2 \quad (10.4.89)$$

which includes a penalty term proportional to the size of the coefficient vector. The price of leakage is an increase in computational complexity and some bias in the obtained estimates (see Problem 10.17). More details and practical applications of the leaky LMS algorithm to adaptive equalization are discussed in Gitlin et al. (1992, 1982).

We can simplify the hardware implementation of LMS adaptive filters by using nonlinearities to avoid the multiplications involved in the updating of the filter coefficients. These simplified LMS algorithms update the filter coefficients by using quantized correction terms such as $\mu \text{ sign}\{e(n)\}x(n-k)$, $\mu e(n)\text{sign}\{x(n-k)\}$, or $\mu \text{ sign}\{e(n)x(n-k)\}$; and their performance is degraded by the lower precision. Various signum-based LMS adaptive algorithms are discussed in Claessen and Mecklenbrauker (1981), Duttweiler (1982), and Treichler et al. (1987).

Transform-domain and block LMS algorithms

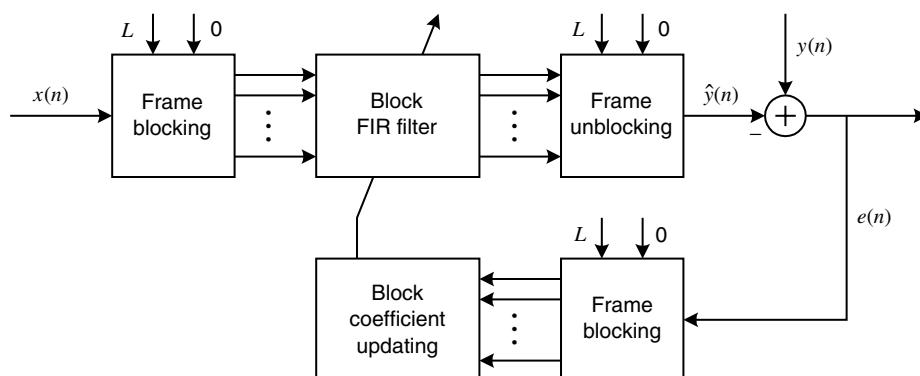
The LMS algorithm attains its best rate of convergence when the input correlation matrix is diagonal with equal eigenvalues. In the case of FIR filters, this implies that the input signal is white noise. When the components of the input data vector are correlated, we can improve the convergence by using an isotropic decorrelating transformation, as shown in Figure 10.30. The transformation matrix can be obtained by using either the triangular or the

**FIGURE 10.30**

Transform domain LMS adaptive filter structure.

orthogonal decomposition of the input correlation matrix as explained in Section 3.5. Since the innovations vector used by the LMS algorithm has uncorrelated components with unit variance, the error performance surface is a hypersphere, and the *transform-domain LMS algorithm* attains its best rate of convergence. In practice, when the input correlation matrix is unknown and possibly time-varying, we can only use suboptimum transforms such as the DFT, the *discrete cosine transform (DCT)*, the *discrete wavelet transform (DWT)*, or some other orthogonal transform. The performance of the obtained adaptive filter depends on the decorrelation properties of the transform, which in turn depends on the properties of the input correlation matrix. Another approach to overcome the problem of slow convergence for highly correlated inputs is found in the family of *affine projection algorithms* discussed in Ozeki and Umeda (1984), Rupp (1995), and Morgan and Kratzer (1996) and the references therein.

In applications that require adaptive filters with a very large number of coefficients, real-time implementation of the LMS algorithm becomes quite involved. For example, acoustic echo cancelers with 8000 coefficients (500 ms sampled at 16 kHz) are typical for teleconference applications (Gilloire et al. 1996). The complexity of such applications can be reduced by using block adaptive filters (see Figure 10.31) that process one block of data at a time in either the time or the frequency domain. The adaptive filter coefficients

**FIGURE 10.31**

Block adaptive filter structure.

are updated once per block and are kept fixed within the block. Such filters have good numerical accuracy, and can be easily pipelined and parallelized, and their complexity can be reduced by computing the involved convolutions and correlations using FFT algorithms. In some applications, such as acoustic echo cancelation, the block-length delay introduced by these filters may create problems. A detailed treatment of block and frequency-domain LMS algorithms is given in Shynk (1992), Gilloire et al. (1996), Haykin (1996), Jenkins and Marshall (1998), and Treichler et al. (1987).

Another approach to reduce complexity and improve convergence is *subband adaptive filtering*, which splits the input signal and the desired response into smaller frequency bands (subbands), subsamples the resulting signals, processes each subband with different LMS filters, and finally interpolates and recombines the subbands to obtain the filter output (Shynk 1992; Gilloire and Vetterli 1992). The improved convergence results because the spectral dynamic range of each subband is smaller than that of the full band. However, the performance of subband adaptive filters is degraded by the cross-talk between adjacent subbands.

10.5 RECURSIVE LEAST-SQUARES ADAPTIVE FILTERS

In this section we use the method of LS to develop adaptive filters, we determine their rate of convergence and misadjustment, and we introduce the *conventional recursive least-squares (CRLS) algorithm* for their implementation. The CRLS algorithm does *not* impose any restrictions on the input data vector; therefore, it can be used for both array processing and FIR filtering applications.

10.5.1 LS Adaptive Filters

LS adaptive filters are designed so that the updating of their coefficients always attains the minimization of the total squared error from the time the filter initiated operation up to the current time. Therefore, the filter coefficients at time index n are chosen to minimize the cost function

$$E(n) = \sum_{j=0}^n \lambda^{n-j} |e(j)|^2 = \sum_{j=0}^n \lambda^{n-j} |y(j) - \mathbf{c}^H \mathbf{x}(j)|^2 \quad (10.5.1)$$

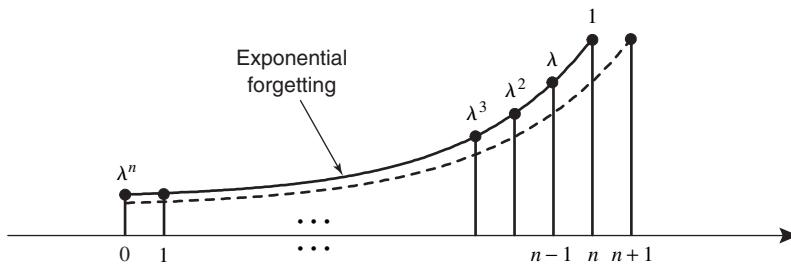
where $e(j)$ is the instantaneous error and the constant λ , $0 < \lambda \leq 1$, is the *forgetting factor*. Note that since the filter coefficients are held constant during the observation interval $0 \leq j \leq n$, the a priori and a posteriori errors are identical. The coefficient vector obtained by minimizing (10.5.1) is denoted by $\mathbf{c}(n)$ and provides the optimum LSE filter at time n . When $\lambda = 1$, we say that the algorithm has *growing memory* because the values of the filter coefficients are a function of all the past input values. The forgetting factor (see Figure 10.32) is used to ensure that data in the distant past are paid less attention (“forgotten”) in order to provide the filter with tracking capability when it operates in a varying SOE (see Section 10.8).

The filter coefficients that minimize the total squared error (10.5.1) are specified by the normal equations

$$\hat{\mathbf{R}}(n)\mathbf{c}(n) = \hat{\mathbf{d}}(n) \quad (10.5.2)$$

$$\text{where } \hat{\mathbf{R}}(n) \triangleq \sum_{j=0}^n \lambda^{n-j} \mathbf{x}(j) \mathbf{x}^H(j) \quad (10.5.3)$$

$$\text{and } \hat{\mathbf{d}}(n) \triangleq \sum_{j=0}^n \lambda^{n-j} \mathbf{x}(j) y^*(j) \quad (10.5.4)$$

**FIGURE 10.32**

Exponential weighting of observations at times n and $n + 1$. Older data are more heavily discounted by the algorithm.

provide exponentially weighted estimates of the input correlation matrix and the cross-correlation vector between input and desired response due to the presence of λ^{n-j} in the cost function (10.5.1). The minimum total squared error is

$$E_{\min}(n) = E_y(n) - \hat{\mathbf{d}}^H(n)\mathbf{c}(n) \quad (10.5.5)$$

where

$$E_y(n) \triangleq \sum_{j=0}^n \lambda^{n-j} |y(j)|^2 \quad (10.5.6)$$

is the energy of the weighted desired response signal. These formulas have been derived in Section 8.2.1.

Suppose now that we wait for some $n > M$, where $\hat{\mathbf{R}}(n)$ is usually nonsingular, we compute $\hat{\mathbf{R}}(n)$ and $\hat{\mathbf{d}}(n)$, and then we solve the normal equations (10.5.2) to determine the filter coefficients $\mathbf{c}(n)$. This approach, which is time-consuming, should be repeated with the arrival of new pairs of observations $\{\mathbf{x}(n), y(n)\}$, that is, at times $n + 1, n + 2$, etc.

A first reduction in computational complexity can be obtained by noticing that (10.5.3) can be expressed as

$$\hat{\mathbf{R}}(n) = \lambda \hat{\mathbf{R}}(n-1) + \mathbf{x}(n)\mathbf{x}^H(n) \quad (10.5.7)$$

which shows that the “new” correlation matrix $\hat{\mathbf{R}}(n)$ can be updated by weighting the “old” correlation matrix $\hat{\mathbf{R}}(n-1)$ with the forgetting factor λ and then incorporating the “new information” $\mathbf{x}(n)\mathbf{x}^H(n)$. Since the outer product $\mathbf{x}(n)\mathbf{x}^H(n)$ is a matrix of rank 1, (10.5.7) provides a rank 1 modification of the correlation matrix. Similarly, using (10.5.4), we can show that

$$\hat{\mathbf{d}}(n) = \lambda \hat{\mathbf{d}}(n-1) + \mathbf{x}(n)y^*(n) \quad (10.5.8)$$

which provides a time update of the cross-correlation vector.

We next show that using these two updatings, we can determine the new coefficient vector $\mathbf{c}(n)$ from the old coefficient vector $\mathbf{c}(n-1)$ and the new observation pair $\{\mathbf{x}(n), y(n)\}$ without solving the normal equations (10.5.2) from scratch.

A priori adaptive LS algorithm. If we solve (10.5.7) for $\hat{\mathbf{R}}(n-1)$ and (10.5.8) for $\hat{\mathbf{d}}(n-1)$ and use the normal equations (10.5.2), we have

$$[\hat{\mathbf{R}}(n) - \mathbf{x}(n)\mathbf{x}^H(n)]\mathbf{c}(n-1) = \hat{\mathbf{d}}(n) - \mathbf{x}(n)y^*(n)$$

or after some simple manipulations

$$\hat{\mathbf{R}}(n)\mathbf{c}(n-1) + \mathbf{x}(n)e^*(n) = \hat{\mathbf{d}}(n) \quad (10.5.9)$$

where

$$e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n) \quad (10.5.10)$$

is the a priori estimation error. If the matrix $\hat{\mathbf{R}}(n)$ is invertible, by multiplying both sides of (10.5.9) by $\hat{\mathbf{R}}^{-1}(n)$ and using (10.5.2), we obtain

$$\mathbf{c}(n-1) + \hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n)e^*(n) = \hat{\mathbf{R}}^{-1}(n)\hat{\mathbf{d}}(n) = \mathbf{c}(n) \quad (10.5.11)$$

If we define the *adaptation gain vector* $\mathbf{g}(n)$ by

$$\hat{\mathbf{R}}(n)\mathbf{g}(n) \triangleq \mathbf{x}(n) \quad (10.5.12)$$

Equation (10.5.11) can be written as

$$\mathbf{c}(n) = \mathbf{c}(n-1) + \mathbf{g}(n)e^*(n) \quad (10.5.13)$$

which shows how to update the old coefficient vector $\mathbf{c}(n-1)$ to obtain the current vector $\mathbf{c}(n)$.

EXAMPLE 10.5.1. It is instructive at this point to derive the LS adaptive filter with a single coefficient. Indeed, since for $M = 1$ the correlation matrix $\hat{\mathbf{R}}(n)$ becomes the scalar $E_x(n)$, we obtain

$$\begin{aligned} E_x(n) &= \lambda E_x(n-1) + |x(n)|^2 \\ e(n) &= y(n) - c^*(n-1)x(n) \\ c(n) &= c(n-1) + \frac{1}{E_x(n)}x(n)e^*(n) \end{aligned}$$

which is like an LMS algorithm with time-varying gain $\mu(n) = 1/E_x(n)$. However, the present algorithm is optimum in the LS sense.

A posteriori adaptive LS algorithm. If we substitute (10.5.7) and (10.5.8) into the normal equations (10.5.2), after some simple manipulations, we obtain

$$\lambda\hat{\mathbf{R}}(n-1)\mathbf{c}(n) - \mathbf{x}(n)\varepsilon^*(n) = \lambda\hat{\mathbf{d}}(n-1) \quad (10.5.14)$$

$$\text{where } \varepsilon(n) = y(n) - \mathbf{c}^H(n)\mathbf{x}(n) \quad (10.5.15)$$

is the a posteriori estimation error. If the matrix $\hat{\mathbf{R}}(n-1)$ is invertible, (10.5.14) gives

$$\mathbf{c}(n) - \lambda^{-1}\hat{\mathbf{R}}^{-1}(n-1)\mathbf{x}(n)\varepsilon^*(n) = \hat{\mathbf{R}}^{-1}(n-1)\hat{\mathbf{d}}(n-1) = \mathbf{c}(n-1)$$

$$\text{or } \mathbf{c}(n) = \mathbf{c}(n-1) + \bar{\mathbf{g}}(n)\varepsilon^*(n) \quad (10.5.16)$$

$$\text{where } \lambda\hat{\mathbf{R}}(n-1)\bar{\mathbf{g}}(n) \triangleq \mathbf{x}(n) \quad (10.5.17)$$

determines the *alternative adaptation gain vector* $\bar{\mathbf{g}}(n)$.

Since recursions (10.5.15) and (10.5.16) are coupled, the a posteriori algorithm is not applicable. However, if we substitute (10.5.16) into (10.5.15), we obtain

$$\begin{aligned} \varepsilon(n) &= y(n) - [\mathbf{c}^H(n-1) + \varepsilon(n)\bar{\mathbf{g}}^H(n)]\mathbf{x}(n) \\ &= e(n) - \varepsilon(n)\bar{\mathbf{g}}^H(n)\mathbf{x}(n) \\ \text{or } \varepsilon(n) &= \frac{e(n)}{\bar{\alpha}(n)} \end{aligned} \quad (10.5.18)$$

$$\text{where } \bar{\alpha}(n) \triangleq 1 + \bar{\mathbf{g}}^H(n)\mathbf{x}(n) = 1 + \lambda^{-1}\mathbf{x}^H(n)\hat{\mathbf{R}}^{-1}(n-1)\mathbf{x}(n) \quad (10.5.19)$$

is known as the *conversion factor*. Hence, we can use (10.5.19) and (10.5.18) to compute the a posteriori error $\varepsilon(n)$ before we update the filter coefficient vector. This trick makes possible the realization and use of the a posteriori LS adaptive filter algorithm. If $\hat{\mathbf{R}}(n-1)$ is positive definite, we have $\bar{\alpha}(n) > 1$ and $|\varepsilon(n)| < |e(n)|$ for all n . Therefore,

$$\sum_n |\varepsilon(n)|^2 < \sum_n |e(n)|^2 \quad (10.5.20)$$

which should be expected[†] because the adaptive filter is designed by minimizing, at each time n , the total squared a posteriori error $\varepsilon(n)$.

[†]The computation of the quantity $\sum_{j=0}^n \lambda^{n-j} |y(j) - \mathbf{c}^H \mathbf{x}(j)|^2$ for $\mathbf{c} = \mathbf{c}(n)$, $\mathbf{c}(j)$, or $\mathbf{c}(j-1)$ gives the block, a posteriori, or a priori total squared error. Clearly, only the block filter performs optimum LS filtering for all data in the interval $0 \leq j \leq n$ (see Problem 10.22).

Also, from (10.5.13), (10.5.16), and (10.5.18) we obtain

$$\mathbf{g}(n) = \frac{\bar{\mathbf{g}}(n)}{\bar{\alpha}(n)} \quad (10.5.21)$$

which shows that the two adaptation gains have the same direction but different lengths. However, from (10.5.13) and (10.5.16) we see that the corrections $\mathbf{g}(n)e^*(n)$ and $\bar{\mathbf{g}}(n)\varepsilon^*(n)$ are equal.

Another conversion factor, defined in terms of the gain vector $\mathbf{g}(n)$, is

$$\alpha(n) \triangleq 1 - \mathbf{x}^H(n)\hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n) = 1 - \mathbf{x}^H(n)\mathbf{g}(n) \quad (10.5.22)$$

and has some interesting interpretations. Using (10.5.21), we have

$$\begin{aligned} \alpha(n) &= 1 - \frac{\mathbf{x}^H(n)\bar{\mathbf{g}}(n)}{\bar{\alpha}(n)} \\ \alpha(n)\bar{\alpha}(n) &= \bar{\alpha}(n) + 1 - [1 + \mathbf{x}^H(n)\bar{\mathbf{g}}(n)] = 1 \end{aligned}$$

or $\alpha(n) = \frac{1}{\bar{\alpha}(n)}$ (10.5.23)

which shows that the two conversion factors are inverses of each other. Since the input correlation matrix is nonnegative definite, that is, $\mathbf{x}^H(n)\hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n) \geq 0$, (10.5.22) implies

$$0 < \alpha(n) \leq 1 \quad (10.5.24)$$

that is, the conversion factor $\alpha(n)$ is bounded by 0 and 1. This bound allows the interpretation of $\alpha(n)$ as an angle variable (Lee et al. 1981), and its monitoring can provide information about the proper operation of RLS algorithms. Also the quantity $1 - \alpha(n)$ can be interpreted as a *likelihood variable* (Lee et al. 1981). It can be shown (see Problem 10.23) that

$$\alpha(n) = \lambda^M \frac{\det \hat{\mathbf{R}}(n-1)}{\det \hat{\mathbf{R}}(n)} \quad (10.5.25)$$

which shows the importance of $\alpha(n)$ or $\bar{\alpha}(n)$ for the invertibility for the estimated correlation matrix.

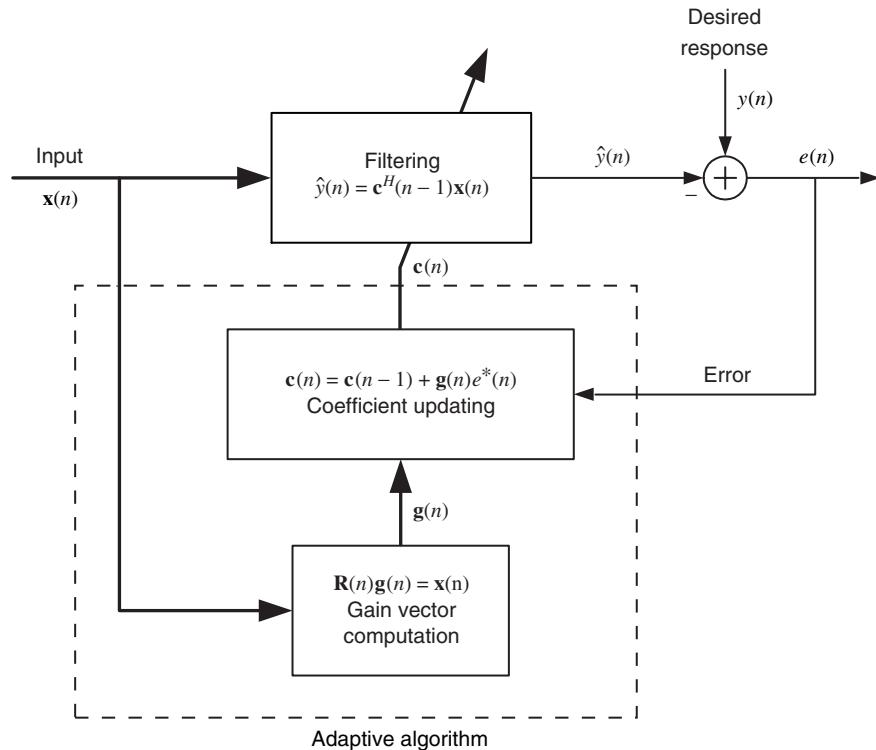
The computational organization of the a priori and a posteriori LS adaptive algorithms is summarized in Table 10.5.

TABLE 10.5
Summary of a priori and a posteriori LS adaptive filter approaches.

	A priori LS adaptive filter	A posteriori LS adaptive filter
Correlation matrix	$\hat{\mathbf{R}}(n) = \lambda\hat{\mathbf{R}}(n-1) + \mathbf{x}(n)\mathbf{x}^H(n)$	$\hat{\mathbf{R}}(n) = \lambda\hat{\mathbf{R}}(n-1) + \mathbf{x}(n)\mathbf{x}^H(n)$
Adaptation gain	$\hat{\mathbf{R}}(n)\mathbf{g}(n) = \mathbf{x}(n)$	$\lambda\hat{\mathbf{R}}(n-1)\bar{\mathbf{g}}(n) = \mathbf{x}(n)$
A priori error	$e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n)$	$e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n)$
Conversion factor	$\alpha(n) = 1 - \mathbf{g}^H(n)\mathbf{x}(n)$	$\bar{\alpha}(n) = 1 + \bar{\mathbf{g}}^H(n)\mathbf{x}(n)$
A posteriori error	$\varepsilon(n) = \alpha(n)e(n)$	$\varepsilon(n) = \frac{e(n)}{\bar{\alpha}(n)}$
Coefficient updating	$\mathbf{c}(n) = \mathbf{c}(n-1) + \mathbf{g}(n)e^*(n)$	$\mathbf{c}(n) = \mathbf{c}(n-1) + \bar{\mathbf{g}}(n)\varepsilon^*(n)$

Figure 10.33 shows a block diagram representation of the a priori LS adaptive filter. There are two important points to be made:

- The adaptation gain is strictly a function of the input signal. The desired response only affects the magnitude and sign of the coefficient correction term through the error.

**FIGURE 10.33**

Basic elements of the a priori LS adaptive filter. Note that the filtering process has no effect on the computation of the gain vector.

- The most demanding computational task in RLS filtering is the computation of the adaptation gain. This involves the solution of a linear system of equations, which requires $O(M^3)$ operations per time update.

10.5.2 Conventional Recursive Least-Squares Algorithm

The major computational load in LS adaptive filters, that is, the computation of the gain vectors

$$\mathbf{g}(n) = \hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n) \quad (10.5.26)$$

or $\bar{\mathbf{g}}(n) = \lambda^{-1}\hat{\mathbf{R}}^{-1}(n-1)\mathbf{x}(n) \quad (10.5.27)$

can be reduced if we can find a recursive formula to update the inverse

$$\mathbf{P}(n) \triangleq \hat{\mathbf{R}}^{-1}(n) \quad (10.5.28)$$

of the correlation matrix. We can develop such an updating by using the rank 1 updating (10.5.7) and the matrix inversion lemma

$$(\lambda\mathbf{R} + \mathbf{x}\mathbf{x}^H)^{-1} = \lambda^{-1}\mathbf{R}^{-1} - \frac{(\lambda^{-1}\mathbf{R}^{-1}\mathbf{x})(\lambda^{-1}\mathbf{R}^{-1}\mathbf{x})^H}{1 + \lambda^{-1}\mathbf{x}^H\mathbf{R}^{-1}\mathbf{x}} \quad (10.5.29)$$

discussed in Appendix A.

Indeed, using (10.5.29), (10.5.7), (10.5.26), and (10.5.19), we can easily show that

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \mathbf{g}(n)\mathbf{g}^H(n) \quad (10.5.30)$$

which provides the desired updating formula. Indeed, given the old matrix $\mathbf{P}(n - 1)$ and the new observations $\{\mathbf{x}(n), y(n)\}$ we compute the new matrix $\mathbf{P}(n)$, using the following procedure

$$\begin{aligned}\bar{\mathbf{g}}(n) &= \lambda^{-1} \mathbf{P}(n - 1) \mathbf{x}(n) \\ \alpha(n) &= 1 + \bar{\mathbf{g}}^H(n) \mathbf{x}(n) \\ \mathbf{g}(n) &= \frac{\bar{\mathbf{g}}(n)}{\alpha(n)} \\ \mathbf{P}(n) &= \lambda^{-1} \mathbf{P}(n - 1) - \mathbf{g}(n) \bar{\mathbf{g}}^H(n)\end{aligned}\tag{10.5.31}$$

which is known as the *conventional recursive LS (CRLS) algorithm*. We again stress that *the CRLS algorithm is valid for both linear combiners and FIR filters because it does not make any assumptions about the nature of the input data vector*. However, for FIR filters we usually assume prewindowing, that is, $\mathbf{x}(-1) = \mathbf{0}$, or equivalently $x(n) = 0$ for $-M \leq n \leq -1$.

Updating of the minimum total squared error. We next derive an update recursion for the minimum total squared error (10.5.5). Using (10.5.6), we can easily see that

$$E_y(n) = \lambda E_y(n - 1) + y(n)y^*(n)\tag{10.5.32}$$

which provides a recursive updating for the energy of the desired response. Substituting (10.5.32) and (10.5.13) into (10.5.5), we obtain

$$E_{\min}(n) = \lambda E_y(n - 1) + y(n)y^*(n) - \hat{\mathbf{d}}^H(n)\mathbf{c}(n - 1) - \hat{\mathbf{d}}^H(n)\mathbf{g}(n)e^*(n)$$

or by using (10.5.8)

$$\begin{aligned}E_{\min}(n) &= \lambda E_y(n - 1) + y(n)y^*(n) - \hat{\mathbf{d}}^H(n)\mathbf{g}(n)e^*(n) \\ &\quad - y(n)\mathbf{x}^H(n)\mathbf{c}(n - 1) - \lambda \hat{\mathbf{d}}^H(n - 1)\mathbf{c}(n - 1)\end{aligned}$$

Rearranging the terms of the last equation and using (10.5.5), we have

$$\begin{aligned}E_{\min}(n) &= \lambda [E_y(n - 1) - \hat{\mathbf{d}}^H(n - 1)\mathbf{c}(n - 1)] + [y(n) - \hat{\mathbf{d}}^H(n)\mathbf{g}(n)]e^*(n) \\ &= \lambda E_{\min}(n - 1) + \{y(n) - [\hat{\mathbf{d}}^H(n)\hat{\mathbf{R}}^{-1}(n)][\hat{\mathbf{R}}(n)\mathbf{g}(n)]\}e^*(n) \\ &= \lambda E_{\min}(n - 1) + [y(n) - \mathbf{c}^H(n)\mathbf{x}(n)]e^*(n)\end{aligned}$$

where the last equation is obtained because the matrix $\hat{\mathbf{R}}(n)$ and its inverse are Hermitian. The last equation leads to

$$E_{\min}(n) = \lambda E_{\min}(n - 1) + \varepsilon(n)e^*(n)\tag{10.5.33}$$

$$= \lambda E_{\min}(n - 1) + \bar{\alpha}(n)|\varepsilon(n)|^2\tag{10.5.34}$$

$$= \lambda E_{\min}(n - 1) + \frac{|\varepsilon(n)|^2}{\alpha(n)}\tag{10.5.35}$$

which provide the desired updating formulas. Since the product $\varepsilon(n)e^*(n)$ is by necessity real, we have $\varepsilon(n)e^*(n) = \varepsilon^*(n)e(n)$. The value of $E_{\min}(n)$ increases with time and reaches a finite limit value only if $\lambda < 1$.

10.5.3 Some Practical Considerations

In the practical implementation of CRLS adaptive filters, we have to deal with the issues of computational complexity, initialization, and finite-word-length effects.

Computational complexity. The complete CRLS algorithm is summarized in Table 10.6. A measure of the computational complexity of the CRLS algorithm is provided by

TABLE 10.6

Practical implementation of the RLS algorithm. To update $\mathbf{P}(n)$, we only compute its upper (low) triangular part and determine the other part using Hermitian symmetry.

Initialization
$\mathbf{c}(-1) = \mathbf{0}$ $\mathbf{P}(-1) = \delta^{-1} \mathbf{I}$
δ = small positive constant
For each $n = 0, 1, 2, \dots$ compute:
Adaptation gain computation
$\tilde{\mathbf{g}}_\lambda(n) = \mathbf{P}(n-1)\mathbf{x}(n)$
$\alpha_\lambda(n) = \lambda + \tilde{\mathbf{g}}_\lambda^H(n)\mathbf{x}(n)$
$\mathbf{g}(n) = \frac{\tilde{\mathbf{g}}_\lambda(n)}{\alpha_\lambda(n)}$
$\mathbf{P}(n) = \lambda^{-1}[\mathbf{P}(n-1) - \mathbf{g}(n)\tilde{\mathbf{g}}_\lambda^H(n)]$
Filtering
$e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n)$
Coefficient updating
$\mathbf{c}(n) = \mathbf{c}(n-1) + \mathbf{g}(n)e^*(n)$

the number of operations (one operation consists of one multiplication and one addition) required to perform one updating. Since $\mathbf{P}(n)$ is Hermitian, it is possible to implement the algorithm so that it will require $2M^2 + 4M$ operations per time updating. The computation of $\tilde{\mathbf{g}}_\lambda(n)$ and the updating of $\mathbf{P}(n)$ require $O(M^2)$ operations. In contrast, all remaining formulas, which involve dot products and vector-by-scalar multiplications, require $O(M)$ operations. The inversion of the correlation matrix $\hat{\mathbf{R}}(n)$ is essentially replaced by the scalar division used to compute $\mathbf{g}(n)$.

Initialization. There are two ways to obtain the values $\mathbf{P}(-1)$ and $\mathbf{c}(-1)$ required to initialize the CRLS algorithm. The most obvious way is to collect an initial block of data $\{\mathbf{x}(n), y(n)\}_{n=0}^{-1}, n_0 > M$, and then compute the exact inverse matrix $\mathbf{P}(-1)$ and the exact LS solution $\mathbf{c}(-1)$.

The approach used in practice is to set $\mathbf{P}(-1) = \delta^{-1} \mathbf{I}$, where δ is a very small positive number (on the order of $0.01\sigma_x^2$) and $\mathbf{c}(-1) = \mathbf{0}$. For FIR filters this corresponds to setting $x(-M+1) = \sqrt{\delta}$ and $x(n) = 0$ for $-M+2 \leq n \leq -1$. For any $n > M$, the normal equations matrix is $\delta\lambda^n \mathbf{I} + \hat{\mathbf{R}}(n)$ and results in a biased estimate of $\mathbf{c}(n)$. However, for large n the choice of δ is unimportant because the algorithm has exponentially forgetting memory for $\lambda < 1$.

It can be shown (see Problem 10.24) that this approach provides a set of coefficients that minimizes the modified cost function

$$E(n) = \delta\lambda^n \|\mathbf{c}\|^2 + \sum_{j=0}^n \lambda^{n-j} |y(j) - \mathbf{c}^H \mathbf{x}(j)|^2 \quad (10.5.36)$$

instead of (10.5.1). This approach amounts to regularization of the LS solution (see Section 8.7.3) and is further discussed in Hubing and Alexander (1991). Note that if we turn off the input, that is, we set $\mathbf{x}(n) = \mathbf{0}$, then (10.5.30) becomes $\mathbf{P}(n) = \lambda^{-1} \mathbf{P}(n-1)$, which is an unstable recursion when $\lambda < 1$.

Finite-word-length effects. There are different RLS algorithms that are algebraically equivalent; that is, they solve the same set of normal equations. Therefore, they have the same rate of convergence and the same insensitivity to variations in the eigenvalue spread of the input correlation matrix with the CRLS algorithm. All RLS algorithms are obtained by exploiting exact mathematical relations between various algorithmic quantities to obtain better computational or numerical properties. Many of these algorithmic quantities have certain physical meanings or theoretical properties. For example, in the CRLS algorithm, the matrix $\mathbf{P}(n)$ is Hermitian and positive definite, the angle variable satisfies $0 < \alpha(n) \leq 1$, and energy $E(n)$ should be always positive. However, when we use finite precision, some of these exact relations, properties, or acceptable ranges for certain algorithmic variables may be violated.

The numerical instability of RLS algorithms can be traced to such forms of *numerical inconsistencies* (Verhaegen 1989; Yang and Böhme 1992; Haykin 1996). The crucial part of the CRLS algorithm is the updating of the inverse correlation matrix $\mathbf{P}(n)$ via (10.5.30). The CRLS algorithm becomes numerically unstable when the matrix $\mathbf{P}(n) = \hat{\mathbf{R}}^{-1}(n)$ loses its Hermitian symmetry or its positive definiteness (Verhaegen 1989). In practice, we can preserve the Hermitian symmetry of $\mathbf{P}(n)$ by computing only its lower (or upper) triangular part, using (10.5.30), and then filling the other part, using the relation $p_{ij}(n) = p_{ji}^*(n)$. Another approach is to replace $\mathbf{P}(n)$ by $[\mathbf{P}(n) + \mathbf{P}^H(n)]/2$ after updating from $\mathbf{P}(n-1)$ to $\mathbf{P}(n)$.

It has been shown that the CRLS algorithm is numerically stable for $\lambda < 1$ and diverges for $\lambda = 1$ (Ljung and Ljung 1985).

10.5.4 Convergence and Performance Analysis

The purpose of any LS adaptive filter, in a stationary SOE, is to identify the optimum filter $\mathbf{c}_o = \mathbf{R}^{-1}\mathbf{d}$ from observations of the input vector $\mathbf{x}(n)$ and the desired response

$$y(n) = \mathbf{c}_o^H \mathbf{x}(n) + e_o(n) \quad (10.5.37)$$

To simplify the analysis we adopt the independence assumptions discussed in Section 10.4.2. The results of the subsequent analysis hold for any LS adaptive filter implemented using the CRLS method or any other algebraically equivalent algorithm. We derive separate results for the growing memory and the fading memory (exponential forgetting) algorithms.

Growing memory ($\lambda = 1$)

In this case all the values of the error signal, from the time the filter starts its operation to the present, have the same influence on the cost function. As a result, the filter loses its tracking ability, which is not important if the filter is used in a stationary SOE.

Convergence in the mean. For $n > M$ the coefficient vector $\mathbf{c}(n)$ is identical to the block LS solution discussed in Section 8.2.2. Therefore

$$E\{\mathbf{c}(n)\} = \mathbf{c}_o \quad \text{for } n > M \quad (10.5.38)$$

that is, the RLS algorithm converges in the mean for $n > M$, where M is the number of coefficients.

Mean square deviation. For $n > M$ we have

$$\Phi(n) = \sigma_o^2 E\{\hat{\mathbf{R}}^{-1}(n)\} \quad (10.5.39)$$

because $\mathbf{c}(n)$ is an exact LS estimate (see Section 8.2.2). The correlation matrix $\hat{\mathbf{R}}(n)$ is described by a complex Wishart distribution, and the expectation of its inverse is

$$E\{\hat{\mathbf{R}}^{-1}(n)\} = \frac{1}{n-M} \mathbf{R}^{-1} \quad n > M \quad (10.5.40)$$

as shown in Muirhead (1982) and Haykin (1996). Hence

$$\Phi(n) = \frac{\sigma_o^2}{n - M} \mathbf{R}^{-1} \quad n > M \quad (10.5.41)$$

and the MSD is

$$\mathcal{D}(n) = \text{tr}[\Phi(n)] = \frac{\sigma_o^2}{n - M} \sum_{i=1}^M \frac{1}{\lambda_i} \quad n > M \quad (10.5.42)$$

where λ_i , the eigenvalues of \mathbf{R} , should not be confused with the forgetting factor λ . From (10.5.42) we conclude that (1) the MSD is magnified by the smallest eigenvalue of \mathbf{R} and (2) the MSD decays almost linearly with time.

A priori excess MSE. We now focus on the a priori LS algorithm because it is widely used in practice and to facilitate a fairer comparison with the (a priori) LMS algorithm. To this end, we note that the a priori excess MSE formula (10.4.48)

$$P_{\text{ex}}(n) = \text{tr}[\mathbf{R}\Phi(n-1)] \quad (10.5.43)$$

derived in Section 10.4.2, under the independence assumption, holds for any a priori adaptive algorithm. Hence, substituting (10.5.41) into (10.5.43), we obtain

$$P_{\text{ex}}(n) = \frac{M}{n - M - 1} \sigma_o^2 \quad n > M \quad (10.5.44)$$

which shows that $P_{\text{ex}}(n)$ tends to zero as $n \rightarrow \infty$.

Exponentially decaying memory ($0 < \lambda < 1$)

In this case the most recent values of the observations have greater influence on the formation of the LS estimate of the filter coefficients. The memory of the filter, that is, the *effective* number of samples used to form the various estimates, is about $1/(1 - \lambda)$ for $0.95 < \lambda < 1$ (see Section 10.8).

Convergence in the mean. We start by multiplying both sides of (10.5.11) by $\hat{\mathbf{R}}(n)$, and then we use (10.5.7) and (10.5.10) to obtain

$$\hat{\mathbf{R}}(n)\mathbf{c}(n) = \lambda \hat{\mathbf{R}}(n-1)\mathbf{c}(n-1) + \mathbf{x}(n)y^*(n) \quad (10.5.45)$$

If we multiply (10.5.7) by \mathbf{c}_o and subtract the resulting equation from (10.5.45), we get

$$\hat{\mathbf{R}}(n)\tilde{\mathbf{c}}(n) = \lambda \hat{\mathbf{R}}(n-1)\tilde{\mathbf{c}}(n-1) + \mathbf{x}(n)e_o^*(n) \quad (10.5.46)$$

where $\tilde{\mathbf{c}}(n) = \mathbf{c}(n) - \mathbf{c}_o$ is the coefficient error vector. Solving (10.5.46) by recursion, we obtain

$$\tilde{\mathbf{c}}(n) = \lambda^n \hat{\mathbf{R}}^{-1}(n) \hat{\mathbf{R}}(0) \tilde{\mathbf{c}}(0) + \hat{\mathbf{R}}^{-1}(n) \sum_{j=0}^n \lambda^{n-j} \mathbf{x}(j) e_o^*(j) \quad (10.5.47)$$

which depends on the initial conditions and the optimum error $e_o(n)$. If we assume that $\hat{\mathbf{R}}(n)$, $\mathbf{x}(j)$, and $e_o(j)$ are independent and we take the expectation of (10.5.47), we obtain

$$E\{\tilde{\mathbf{c}}(n)\} = \delta \lambda^n E\{\hat{\mathbf{R}}^{-1}(n)\} \tilde{\mathbf{c}}(0) \quad (10.5.48)$$

where, as usual, we have set $\hat{\mathbf{R}}(0) = \delta \mathbf{I}$, $\delta > 0$. If the matrix $\hat{\mathbf{R}}(n)$ is positive definite and $0 < \lambda < 1$, then the mean vector $E\{\tilde{\mathbf{c}}(n)\} \rightarrow \mathbf{0}$ as $n \rightarrow \infty$. Hence, the RLS algorithm with exponential forgetting converges asymptotically in the mean to the optimum filter.

Mean square deviation. Using (10.5.46), we obtain the following difference equation for the coefficient error vector

$$\tilde{\mathbf{c}}(n) = \lambda \hat{\mathbf{R}}^{-1}(n) \hat{\mathbf{R}}(n-1) \tilde{\mathbf{c}}(n-1) + \hat{\mathbf{R}}^{-1}(n) \mathbf{x}(n) e_o^*(n)$$

or

$$\tilde{\mathbf{c}}(n) \simeq \lambda \tilde{\mathbf{c}}(n-1) + \hat{\mathbf{R}}^{-1}(n) \mathbf{x}(n) e_o^*(n)$$

because $\hat{\mathbf{R}}^{-1}(n)\hat{\mathbf{R}}(n-1) \simeq \mathbf{I}$ for large n . If we neglect the dependence among $\tilde{\mathbf{c}}(n-1)$, $\hat{\mathbf{R}}(n)$, $\mathbf{x}(n)$, and $e_o(n)$, we have

$$\Phi(n) \simeq \lambda^2 \Phi(n-1) + \sigma_o^2 E\{\hat{\mathbf{R}}^{-1}(n) \mathbf{x}(n) \mathbf{x}^H(n) \hat{\mathbf{R}}^{-1}(n)\} \quad (10.5.49)$$

where $\sigma_o^2 = E\{|e_o(n)|^2\}$.

To make the analysis mathematically tractable, we need an approximation for the inverse matrix $\hat{\mathbf{R}}^{-1}(n)$. To this end, using (10.5.3), we have

$$E\{\hat{\mathbf{R}}(n)\} = \sum_{j=0}^n \lambda^{n-j} E\{\mathbf{x}(n) \mathbf{x}^H(n)\} = \frac{1 - \lambda^{n+1}}{1 - \lambda} \mathbf{R} \simeq \frac{1}{1 - \lambda} \mathbf{R} \quad (10.5.50)$$

where the last approximation holds for $n \gg 1$. If we use the approximation $E\{\hat{\mathbf{R}}(n)\} \simeq \hat{\mathbf{R}}(n)$, we obtain

$$\hat{\mathbf{R}}^{-1}(n) \simeq (1 - \lambda) \mathbf{R}^{-1} \quad (10.5.51)$$

which is more rigorously justified in Eleftheriou and Falconer (1986). Using the last approximation, (10.5.50) becomes

$$\Phi(n) \simeq \lambda^2 \Phi(n-1) + (1 - \lambda)^2 \sigma_o^2 \mathbf{R}^{-1} \quad (10.5.52)$$

which converges because $\lambda^2 < 1$. At steady state we have

$$(1 - \lambda^2) \Phi(\infty) \simeq (1 - \lambda)^2 \sigma_o^2 \mathbf{R}^{-1}$$

because $\Phi(n) \simeq \Phi(n-1)$ for $n \gg 1$. Hence

$$\Phi(\infty) \simeq \frac{1 - \lambda}{1 + \lambda} \sigma_o^2 \mathbf{R}^{-1} \quad (10.5.53)$$

and therefore $\mathcal{D}_\lambda(\infty) = \text{tr}[\Phi(\infty)] = \frac{1 - \lambda}{1 + \lambda} \sigma_o^2 \sum_{i=1}^M \frac{1}{\lambda_i}$ (10.5.54)

which in contrast to (10.5.42) does not converge to zero as $n \rightarrow \infty$. This is explained by noticing that when $\lambda < 1$, the RLS algorithm has finite memory and does not use effectively all the data to form its estimate.

Steady-state *a priori* excess MSE. From (10.5.43) and (10.5.53) we obtain

$$P_{\text{ex}}(\infty) = \text{tr}[\mathbf{R} \Phi(\infty)] \simeq \frac{1 - \lambda}{1 + \lambda} M \sigma_o^2 \quad (10.5.55)$$

which shows that as a result of finite memory, there is a steady-state excess MSE that decreases as λ approaches 1, that is, as the effective memory of the algorithm increases.

Summary

The results of the above analysis are summarized in Table 10.7 for easy reference. We stress at this point that all RLS algorithms, independent of their implementation, have the same performance, assuming that we use sufficient numerical precision (e.g., double-precision floating-point arithmetic). Sometimes, RLS algorithms are said to have optimum learning because at every time instant they minimize the weighted error energy from the start of the operation (Tsypkin 1973). These properties are illustrated in the following example.

EXAMPLE 10.5.2. Consider the adaptive equalizer of Example 10.4.3 shown in block diagram form in Figure 10.26. In this example, we replace the LMS block in Figure 10.26 by the RLS block, and we study the performance of the RLS algorithm and compare it with that of the LMS

TABLE 10.7
Summary of RLS and LMS performance in a stationary SOE.

Property	Growing memory RLS algorithm	Exponential memory RLS algorithm	LMS algorithm
Convergence in the mean	For all $n > M$	Asymptotically for $n \rightarrow \infty$	Asymptotically for $n \rightarrow \infty$
Convergence in MS	Independent of the eigenvalue spread	Independent of the eigenvalue spread	Depends on the eigenvalue spread
Excess MSE	$P_{\text{ex}}(n) = \frac{M\sigma_o^2}{n - M - 1} \rightarrow 0$	$P_{\text{ex}}(\infty) = \frac{1 - \lambda}{1 + \lambda} M\sigma_o^2$	$P_{\text{ex}}(\infty) \simeq \mu\sigma_o^2 \text{tr } \mathbf{R}$

algorithm. The input data source is a Bernoulli sequence $\{y(n)\}$ with symbols $+1$ and -1 having zero mean and unit variance. The channel impulse response is a raised cosine

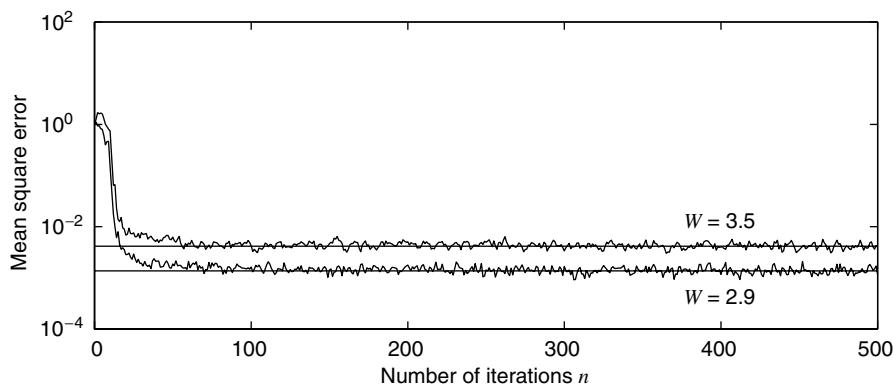
$$h(n) = \begin{cases} 0.5 \left\{ 1 + \cos \left[\frac{2\pi}{W}(n-2) \right] \right\} & n = 1, 2, 3 \\ 0 & \text{otherwise} \end{cases} \quad (10.5.56)$$

where the parameter W controls the amount of channel distortion [or the eigenvalue spread $\mathcal{X}(\mathbf{R})$ produced by the channel]. The channel noise sequence $v(n)$ is white Gaussian with $\sigma_v^2 = 0.001$. The adaptive equalizer has $M = 11$ coefficients, and the input signal $y(n)$ is delayed by $\Delta = 7$ samples. The error signal $e(n) = y(n - \Delta) - \hat{y}(n)$ is used along with $x(n)$ to implement the RLS algorithm given in Table 10.6 with $\mathbf{c}(0) = \mathbf{0}$ and $\delta = 0.001$. We performed Monte Carlo simulations on 100 realizations of random sequences with $W = 2.9$ and $W = 3.5$, and $\lambda = 1$ and 0.8. The results are shown in Figures 10.34 and 10.35.

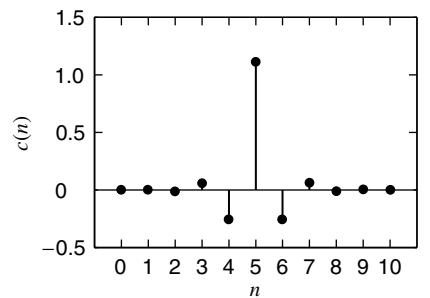
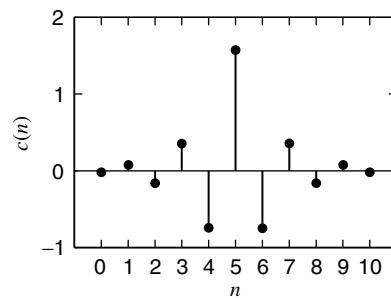
Effect of eigenvalue spread. Performance plots of the RLS algorithm for $W = 2.9$ and $W = 3.5$ are shown in Figure 10.34. In plot (a) we depict MSE learning curves along with the steady-state (or minimum) error. We observe that the MSE convergence rate of the RLS, unlike that for the LMS, does not change with W [or equivalently with change in $\mathcal{X}(\mathbf{R})$]. The steady-state error, on the other hand, increases with W . The important difference between the two algorithms is that the convergence rate is faster for the RLS (compare Figures 10.34 and 10.27). Clearly, this faster convergence of the RLS algorithm is achieved by an increase in computational complexity. In plots (b) and (c) we show the ensemble averaged equalizer coefficients. Clearly, the responses are symmetric with respect to $n = 5$, as assumed. Also equalizer coefficients converge to different inverses due to changes in the channel characteristics.

Effect of forgetting factor λ . In Figure 10.35 we show the MSE learning curves obtained for $W = 2.9$ and with two different factors of 1 and 0.8. For $\lambda = 1$, as explained before, the algorithm has infinite memory and hence the steady-state excess MSE is zero. This fact can be verified in the plot for $\lambda = 1$ in which the MSE converges to the minimum error. For $\lambda = 0.8$, the effective memory is $1/(1 - \lambda) = 5$, which clearly is inadequate for the accurate estimation of the required statistics, resulting in increased excess MSE. Therefore, the algorithm should produce a nonzero excess MSE. This fact can be observed from the plot for $\lambda = 0.8$.

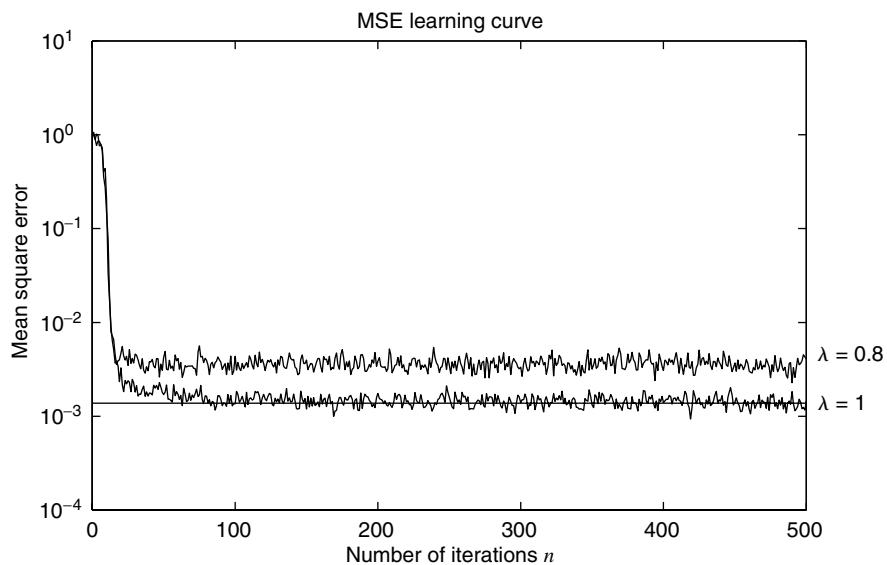
There are two practical issues regarding the RLS algorithm that need an explanation. The first issue relates to the practical value of λ . Although λ can take any value in the interval $0 \leq \lambda \leq 1$, since it influences the effective memory size, the value of λ should be closer to 1. This value is determined by the number of parameters to be estimated and the desired size of the effective memory. Typical values used are between 0.99 and 1 (not 0.8, as we used in this example for demonstration). The second issue deals with the actual computation of matrix $\mathbf{P}(n)$. This matrix must be conjugate symmetric and positive definite. However, an implementation of the CRLS algorithm of Table 10.6 on a finite-precision processor will eventually disturb this symmetry and positive definiteness and would result in an unstable performance. Therefore, it is necessary to force this symmetry either by computing only its lower (or upper) triangular values or by using $\mathbf{P}(n) \leftarrow [\mathbf{P}(n) + \mathbf{P}^H(n)]/2$. Failure to do so generally affects the algorithm performance for $\lambda < 1$.



(a) MSE learning curve

(b) $W = 2.9$ (c) $W = 3.5$ **FIGURE 10.34**

Performance analysis curves of the RLS algorithm in the adaptive equalizer: $\lambda = 1$.

**FIGURE 10.35**

MSE learning curves of the RLS algorithm in the adaptive equalizer: $W = 2.9$.

10.6 RLS ALGORITHMS FOR ARRAY PROCESSING

In this section we show how to develop algorithms for RLS array processing using the QR decomposition. The obtained algorithms (1) are algebraically equivalent to the CRLS algorithm, (2) have very good numerical properties, and (3) are modular and can be implemented using parallel processing. Since there are no restrictions on the input data vector, the algorithms require $O(M^2)$ operations per time update and can be used for both array processing and FIR filtering applications. The method of choice for applications that only require the a priori error $e(n)$ or the a posteriori error $\varepsilon(n)$ is the QR-RLS algorithm using the Givens rotations. For applications that require the coefficient vector $\mathbf{c}(n)$, the Givens rotations-based inverse QR-RLS algorithm is preferred. In Section 10.7 we develop fast algorithms for FIR filters, with a complexity of $O(M)$ operations per time update, by exploiting the shift invariance of the input data vector.

10.6.1 LS Computations Using the Cholesky and QR Decompositions

We start by reformulating the exponentially weighted LS filtering problem in terms of data matrices, as discussed in Section 8.2. If $\mathbf{c}(n)$ is the LS filter coefficient vector at time instant n , we have

$$\varepsilon(j) = y(j) - \mathbf{c}^H(n)\mathbf{x}(j) \quad 0 \leq j \leq n \quad (10.6.1)$$

$$\text{where } \mathbf{x}(j) = [x_1(j) \ x_2(j) \ \cdots \ x_M(j)]^T \quad (10.6.2)$$

for array processing and

$$\mathbf{x}(j) = [x(j) \ x(j-1) \ \cdots \ x(j-M+1)]^T \quad (10.6.3)$$

for FIR filtering. We stress that $\mathbf{c}(n)$ should be held constant during the optimization interval $0 \leq j \leq n$. Using the $(n+1) \times M$ data matrix

$$\begin{aligned} \mathbf{X}^H(n) &\triangleq [\mathbf{x}(0) \ \mathbf{x}(1) \ \cdots \ \mathbf{x}(n)] \\ &= \begin{bmatrix} x_1(0) & x_1(1) & \cdots & x_1(n) \\ x_2(0) & x_2(1) & \cdots & x_2(n) \\ \vdots & \vdots & \ddots & \vdots \\ x_M(0) & x_M(1) & \cdots & x_M(n) \end{bmatrix} \end{aligned} \quad (10.6.4)$$

the $(n+1) \times 1$ desired response vector

$$\mathbf{y}(n) \triangleq [y(0) \ y(1) \ \cdots \ y(n)]^H \quad (10.6.5)$$

and the $(n+1) \times 1$ a posteriori error vector

$$\boldsymbol{\varepsilon}(n) \triangleq [\varepsilon(0) \ \varepsilon(1) \ \cdots \ \varepsilon(n)]^H \quad (10.6.6)$$

we can combine the $n+1$ equations (10.6.1) in a single equation as

$$\boldsymbol{\varepsilon}(n) = \mathbf{y}(n) - \mathbf{X}(n)\mathbf{c}(n) \quad (10.6.7)$$

If we define the $(n+1) \times (n+1)$ exponential weighting matrix

$$\boldsymbol{\Lambda}^2(n) \triangleq \text{diag}\{\lambda^n, \lambda^{n-1}, \dots, 1\} \quad (10.6.8)$$

we can express the total squared error (10.5.1) and the normal equations (10.5.2) in the form required to apply orthogonal decomposition techniques (see Section 8.6). Indeed, we can easily see that the total squared error can be written as

$$E(n) = \sum_{j=0}^n \lambda^{n-j} |\varepsilon(j)|^2 = \|\boldsymbol{\Lambda}(n)\boldsymbol{\varepsilon}(n)\|^2 \quad (10.6.9)$$

and the LS filter coefficients are determined by the normal equations

$$\hat{\mathbf{R}}(n)\mathbf{c}(n) = \hat{\mathbf{d}}(n) \quad (10.6.10)$$

where $\hat{\mathbf{R}}(n) = \sum_{j=0}^n \lambda^{n-j} \mathbf{x}(j) \mathbf{x}^H(j) = [\Lambda(n)\mathbf{X}(n)]^H [\Lambda(n)\mathbf{X}(n)]$ (10.6.11)

and $\hat{\mathbf{d}}(n) = \sum_{j=0}^n \lambda^{n-j} \mathbf{x}(j) y^*(j) = [\Lambda(n)\mathbf{X}(n)]^H [\Lambda(n)\mathbf{y}(n)]$ (10.6.12)

are expressed as a function of the weighted data matrix $[\Lambda(n)\mathbf{X}(n)]$ and the weighted desired response vector $[\Lambda(n)\mathbf{y}(n)]$.

In Chapter 6 we discussed how to solve the normal equations (10.6.10) by using either the Cholesky decomposition

$$\hat{\mathbf{R}}(n) = \tilde{\mathbf{L}}(n)\tilde{\mathbf{L}}^H(n) \quad (10.6.13)$$

or the LDU decomposition

$$\hat{\mathbf{R}}(n) = \mathbf{L}(n)\mathbf{D}(n)\mathbf{L}^H(n) \quad (10.6.14)$$

where $\tilde{\mathbf{L}}(n) = \mathbf{D}^{1/2}(n)\mathbf{L}(n)$.

The Cholesky factor $\tilde{\mathbf{L}}(n)$ can be computed either from matrix $\hat{\mathbf{R}}(n)$ using the Cholesky decomposition algorithm (see Section 6.3) or from data matrix $[\Lambda(n)\mathbf{X}(n)]$ using one of the QR decomposition methods (Givens, Householder, or MGS) discussed in Chapter 8.

Suppose now that the QR decomposition[†] is

$$\mathbf{Q}(n)[\Lambda(n)\mathbf{X}(n)] = \begin{bmatrix} \tilde{\mathbf{R}}(n) \\ \mathbf{0} \end{bmatrix} \quad (10.6.15)$$

where $\tilde{\mathbf{R}}(n)$ is a unique upper triangular matrix with positive diagonal elements and $\mathbf{Q}(n)$ is a unitary matrix. From (10.6.11) and (10.6.15) we have

$$\hat{\mathbf{R}}(n) = \tilde{\mathbf{R}}^H(n)\tilde{\mathbf{R}}(n) \quad (10.6.16)$$

which implies, owing to the uniqueness of the Cholesky decomposition, that $\tilde{\mathbf{L}}(n) = \tilde{\mathbf{R}}^H(n)$. Although the two approaches are *algebraically* equivalent, the QR decomposition (QRD) methods have superior numerical properties because they avoid the squaring operation (10.6.11) (see Section 8.6).

Given the Cholesky factor $\tilde{\mathbf{R}}(n)$, we first solve the lower triangular system

$$\tilde{\mathbf{R}}^H(n)\tilde{\mathbf{k}}(n) \triangleq \hat{\mathbf{d}}(n) \quad (10.6.17)$$

to obtain the partial correlation vector $\tilde{\mathbf{k}}(n)$, using forward elimination. In the case of QRD the vector $\tilde{\mathbf{k}}(n)$ is obtained by transforming $\Lambda(n)\mathbf{y}(n)$ and retaining its first M components, that is,

$$\mathbf{Q}(n)[\Lambda(n)\mathbf{y}(n)] = \mathbf{z}(n) \triangleq \begin{bmatrix} \tilde{\mathbf{k}}(n) \\ \mathbf{z}_2(n) \end{bmatrix} \quad (10.6.18)$$

where $\tilde{\mathbf{k}}(n) = \mathbf{z}^{[M]}(n)$ (see Section 8.6). The minimum LSE is given by

$$E(n) = E_y(n) - \hat{\mathbf{d}}^H(n)\mathbf{c}(n) = \|\mathbf{y}(n)\|^2 - \|\tilde{\mathbf{k}}(n)\|^2 \quad (10.6.19)$$

which was also proved in Section 8.6.

To compute the filter parameters, we can solve the upper triangular system

$$\tilde{\mathbf{R}}(n)\mathbf{c}(n) = \tilde{\mathbf{k}}(n) \quad (10.6.20)$$

by backward elimination. As we discussed in Section 6.3, the solution of (10.6.20) is *not* order-recursive.

[†]To comply with adaptive filtering literature, we express the QR decomposition as $\mathbf{Q}\mathbf{X} = \tilde{\mathbf{R}}$ instead of $\mathbf{Q}^H\mathbf{X} = \tilde{\mathbf{R}}$, which we used in Chapter 8 and is widely used in numerical analysis.

In applications that only require the a posteriori or a priori errors, we can avoid the solution of (10.6.20). Indeed, if we define the LS innovations vector $\tilde{\mathbf{w}}(n)$ by

$$\tilde{\mathbf{R}}^H(n)\tilde{\mathbf{w}}(n) \triangleq \mathbf{x}(n) \quad (10.6.21)$$

$$\text{we obtain} \quad \varepsilon(n) = y(n) - \mathbf{c}^H(n)\mathbf{x}(n) = y(n) - \tilde{\mathbf{k}}^H(n)\tilde{\mathbf{w}}(n) \quad (10.6.22)$$

$$\text{and} \quad e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n) = y(n) - \tilde{\mathbf{k}}^H(n-1)\tilde{\mathbf{w}}(n) \quad (10.6.23)$$

which can be used to compute the errors without knowledge of the parameter vector $\mathbf{c}(n)$. Furthermore, since the lower triangular systems (10.6.17) and (10.6.21) satisfy the optimum nesting property, we can compute both errors in an order-recursive manner.

If we know the factors $\mathbf{L}(n)$ and $\mathbf{D}(n)$ of $\hat{\mathbf{R}}(n)$ at each time instant n , we can use the orthogonal triangular structure shown in Figure 7.1 (see Sections 7.1.5 and 8.5) to compute all $e_m(n)$ and $\varepsilon_m(n)$ for all $1 \leq m \leq M$. A similar structure can be obtained by using the Cholesky factor $\tilde{\mathbf{L}}(n)$ (see Problem 10.26).

From the discussion in Section 10.5.1 we saw that the key part of the CRLS algorithm is the computation of the gain vector

$$\hat{\mathbf{R}}(n)\mathbf{g}(n) = \mathbf{x}(n) \quad (10.6.24)$$

or the alternative gain vector $\lambda\hat{\mathbf{R}}(n-1)\bar{\mathbf{g}}(n) = \mathbf{x}(n)$. Using (10.6.16), (10.6.21), and (10.6.24), we obtain

$$\tilde{\mathbf{R}}(n)\mathbf{g}(n) = \tilde{\mathbf{w}}(n) \quad (10.6.25)$$

which expresses the gain vector in terms of the Cholesky factor $\tilde{\mathbf{R}}(n)$ and the innovations vector $\tilde{\mathbf{w}}(n)$. Similarly with (10.6.20), (10.6.25) lacks the optimum nesting property that is required to obtain an order-recursive algorithm.

To summarize, if we can update the Cholesky factors of either $\hat{\mathbf{R}}(n)$ or $\hat{\mathbf{R}}^{-1}(n)$, we can develop exact RLS algorithms that provide both the filtering errors and the coefficient vector or the filtering error only. The relevant relations are shown in Table 10.8. We stress that the Cholesky decomposition method determines the factors $\tilde{\mathbf{R}}(n)$ and $\tilde{\mathbf{k}}(n)$ by factoring the matrix

$$[\mathbf{X}(n) \mathbf{y}(n)]^H \Lambda^2(n) [\mathbf{X}(n) \mathbf{y}(n)]$$

whereas the QRD methods factor the data matrix $\Lambda(n)[\mathbf{X}(n) \mathbf{y}(n)]$. Since all these algorithms propagate the square roots $\tilde{\mathbf{R}}(n)$ or $\tilde{\mathbf{R}}^{-1}(n)$, the matrices determined by $\hat{\mathbf{R}}(n) = \tilde{\mathbf{R}}^H(n)\tilde{\mathbf{R}}(n)$ and $\hat{\mathbf{R}}^{-1}(n) = \tilde{\mathbf{R}}^{-1}(n)\tilde{\mathbf{R}}^{-H}(n)$ are guaranteed to be Hermitian and are more likely to preserve their positive definiteness. Hence, such algorithms have better numerical properties than the CRLS method.

TABLE 10.8
Triangular decomposition RLS algorithms using coefficient updating and direct error extraction.

Error and coefficients updating	Error-only updating
$\tilde{\mathbf{R}}^H(n)\tilde{\mathbf{w}}(n) = \mathbf{x}(n)$	$\tilde{\mathbf{R}}^H(n)\tilde{\mathbf{k}}(n) \triangleq \hat{\mathbf{d}}(n)$
$\tilde{\mathbf{R}}(n)\mathbf{g}(n) = \tilde{\mathbf{w}}(n)$	$\tilde{\mathbf{R}}^H(n)\tilde{\mathbf{w}}(n) = \mathbf{x}(n)$
$e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n)$	$e(n) = y(n) - \tilde{\mathbf{k}}^H(n-1)\tilde{\mathbf{w}}(n)$
$\mathbf{c}(n) = \mathbf{c}(n-1) + \mathbf{g}(n)e^*(n)$	

10.6.2 Two Useful Lemmas

We next prove two lemmas that are very useful in the development of RLS algorithms using QRD methods. We start with the first lemma, which stems from the algebraic equivalence between the Cholesky and QR decompositions.

LEMMA 10.1. Computing the QRD of the $(n + 1) \times M$ data matrix $\Lambda(n)\mathbf{X}(n)$ is equivalent to evaluating the QRD of the $(M + 1) \times M$ matrix

$$\begin{bmatrix} \sqrt{\lambda}\tilde{\mathbf{R}}(n-1) \\ \mathbf{x}^H(n) \end{bmatrix}$$

Proof. Indeed, if we express $\Lambda(n)\mathbf{X}(n)$ as

$$\Lambda(n)\mathbf{X}(n) = \begin{bmatrix} \sqrt{\lambda}\Lambda(n-1)\mathbf{X}(n-1) \\ \mathbf{x}^H(n) \end{bmatrix} \quad (10.6.26)$$

and define a matrix

$$\bar{\mathbf{Q}}(n-1) \triangleq \begin{bmatrix} \mathbf{Q}(n-1) & \mathbf{0} \\ \mathbf{0}^H & 1 \end{bmatrix} \quad (10.6.27)$$

we obtain

$$\bar{\mathbf{Q}}(n-1)\Lambda(n)\mathbf{X}(n) = \begin{bmatrix} \sqrt{\lambda}\tilde{\mathbf{R}}(n-1) \\ \mathbf{0} \\ \mathbf{x}^H(n) \end{bmatrix} \quad (10.6.28)$$

by using (10.6.15). If we can construct a matrix $\hat{\mathbf{Q}}(n)$ that performs the QRD of the right-hand side of (10.6.28), then the unitary matrix $\mathbf{Q}(n) \triangleq \hat{\mathbf{Q}}(n)\bar{\mathbf{Q}}(n-1)$ performs the QRD of $\Lambda(n)\mathbf{X}(n)$. Since the block of zeros in (10.6.28) has no effect on the construction of matrix $\hat{\mathbf{Q}}(n)$, the construction of $\hat{\mathbf{Q}}(n)$ is equivalent to finding a unitary matrix that performs the QRD of

$$\begin{bmatrix} \sqrt{\lambda}\tilde{\mathbf{R}}(n-1) \\ \mathbf{x}^H(n) \end{bmatrix}$$

The second lemma, known as the *matrix factorization lemma* (Golub and Van Loan 1996; Sayed and Kailath 1994), provides an elegant tool for the derivation of QRD-based RLS algorithms.

LEMMA 10.2. If \mathbf{A} and \mathbf{B} are any two $N \times M$ ($N \leq M$) matrices, then

$$\mathbf{A}^H\mathbf{A} = \mathbf{B}^H\mathbf{B} \quad (10.6.29)$$

if and only if there exists an $N \times N$ unitary matrix \mathbf{Q} ($\mathbf{Q}^H\mathbf{Q} = \mathbf{I}$) such that

$$\mathbf{Q}\mathbf{A} = \mathbf{B} \quad (10.6.30)$$

Proof. From (10.6.30) we have $\mathbf{B}^H\mathbf{B} = \mathbf{A}^H\mathbf{Q}^H\mathbf{Q}\mathbf{A} = \mathbf{A}^H\mathbf{A}$, which proves (10.6.29). To prove the converse, we use the singular value decomposition (SVD) of matrices \mathbf{A} and \mathbf{B}

$$\mathbf{A} = \mathbf{U}_A \Sigma_A \mathbf{V}_A^H \quad (10.6.31)$$

$$\mathbf{B} = \mathbf{U}_B \Sigma_B \mathbf{V}_B^H \quad (10.6.32)$$

where \mathbf{U}_A and \mathbf{U}_B are $N \times N$ unitary matrices, \mathbf{V}_A and \mathbf{V}_B are $M \times M$ unitary matrices, and Σ_A and Σ_B are $N \times M$ matrices consisting of the nonnegative singular values of \mathbf{A} and \mathbf{B} . Using (10.6.29) in conjunction with (10.6.31) and (10.6.32), we obtain

$$\mathbf{V}_A = \mathbf{V}_B \quad (10.6.33)$$

and

$$\Sigma_A = \Sigma_B \quad (10.6.34)$$

If we now define the matrix

$$\mathbf{Q} \triangleq \mathbf{U}_B \mathbf{U}_A^H$$

and use (10.6.33) and (10.6.34), we have

$$\mathbf{Q}\mathbf{A} = \mathbf{U}_B \mathbf{U}_A^H \mathbf{U}_A \Sigma_A \mathbf{V}_A^H = \mathbf{U}_B \Sigma_B \mathbf{V}_B^H = \mathbf{B}$$

which proves the converse of the lemma.

10.6.3 The QR-RLS Algorithm

We next show how to update the factors $\tilde{\mathbf{R}}(n)$ and $\tilde{\mathbf{k}}(n)$ of the extended data matrix $\Lambda(n)[\mathbf{X}(n) \mathbf{y}(n)]$ and then compute the a priori error $e(n)$ or the a posteriori error $\varepsilon(n)$. The findings hold independently of the method we use to construct the orthogonalizing matrix $\mathbf{Q}(n)$.

Suppose now that at time n we know the old Cholesky factors $\tilde{\mathbf{R}}(n-1)$ and $\tilde{\mathbf{k}}(n-1)$, we receive the new data $\{\mathbf{x}(n), y(n)\}$, and we wish to determine the new factors $\tilde{\mathbf{R}}(n)$ and $\tilde{\mathbf{k}}(n)$ without repeating all the work. To this end, we show that if there exists a unitary matrix $\mathbf{Q}(n)$ that annihilates the vector $\mathbf{x}^H(n)$ from the last row of the left-hand side matrix in the relation

$$\mathbf{Q}(n) \begin{bmatrix} \sqrt{\lambda} \tilde{\mathbf{R}}(n-1) & \sqrt{\lambda} \tilde{\mathbf{k}}(n-1) & \mathbf{0} \\ \mathbf{x}^H(n) & y^*(n) & 1 \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{R}}(n) & \tilde{\mathbf{k}}(n) & \tilde{\mathbf{w}}(n) \\ \mathbf{0}^H & \tilde{e}^*(n) & \tilde{\alpha}(n) \end{bmatrix} \quad (10.6.35)$$

then the right-hand side matrix provides the required updates and errors. The scalar $\tilde{\alpha}(n)$ is real-valued because it is equal to the last diagonal element of $\mathbf{Q}(n)$. The meaning and use of $\tilde{\alpha}(n)$ and $\tilde{\mathbf{w}}(n)$, which comprise the last column of $\mathbf{Q}(n)$, will be explained in the sequel.

If we apply Lemma 10.2 with

$$\mathbf{A} = \begin{bmatrix} \sqrt{\lambda} \tilde{\mathbf{R}}(n-1) & \sqrt{\lambda} \tilde{\mathbf{k}}(n-1) & \mathbf{0} \\ \mathbf{x}^H(n) & y^*(n) & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} \tilde{\mathbf{R}}(n) & \tilde{\mathbf{k}}(n) & \tilde{\mathbf{w}}(n) \\ \mathbf{0}^H & \tilde{e}^*(n) & \tilde{\alpha}(n) \end{bmatrix}$$

we obtain[†]

$$\langle \mathbf{B}^H \mathbf{B} \rangle_{11} = \tilde{\mathbf{R}}^H(n) \tilde{\mathbf{k}}(n) = \lambda \tilde{\mathbf{R}}^H(n-1) \lambda \tilde{\mathbf{R}}(n-1) + \mathbf{x}(n) \mathbf{x}^H(n) = \langle \mathbf{A}^H \mathbf{A} \rangle_{11} \quad (10.6.36)$$

$$\langle \mathbf{B}^H \mathbf{B} \rangle_{12} = \tilde{\mathbf{R}}^H(n) \tilde{\mathbf{k}}(n) = \lambda \tilde{\mathbf{R}}^H(n-1) \tilde{\mathbf{k}}(n-1) + \mathbf{x}(n) y^*(n) = \langle \mathbf{A}^H \mathbf{A} \rangle_{12} \quad (10.6.37)$$

$$\langle \mathbf{B}^H \mathbf{B} \rangle_{13} = \tilde{\mathbf{R}}^H(n) \tilde{\mathbf{w}}(n) = \mathbf{x}(n) = \langle \mathbf{A}^H \mathbf{A} \rangle_{13} \quad (10.6.38)$$

$$\langle \mathbf{B}^H \mathbf{B} \rangle_{23} = \tilde{\mathbf{k}}^H(n) \tilde{\mathbf{w}}(n) + \tilde{e}(n) \tilde{\alpha}(n) = y(n) = \langle \mathbf{A}^H \mathbf{A} \rangle_{23} \quad (10.6.39)$$

$$\langle \mathbf{B}^H \mathbf{B} \rangle_{33} = \tilde{\mathbf{w}}^H(n) \tilde{\mathbf{w}}(n) + \tilde{\alpha}^2(n) = 1 = \langle \mathbf{A}^H \mathbf{A} \rangle_{33} \quad (10.6.40)$$

We first note that (10.6.36) is identical to the time updating (10.5.7) of the correlation matrix. Hence, $\tilde{\mathbf{R}}(n)$ is the Cholesky factor of $\hat{\mathbf{R}}(n)$. Also (10.6.37) is identical, due to (10.6.17), to the time updating (10.5.8) of the cross-correlation vector $\hat{\mathbf{d}}(n)$, and (10.6.38) is the definition (10.6.21) of the innovations vector. To uncover the physical meaning of $\tilde{e}(n)$ and $\tilde{\alpha}(n)$, we note that comparing (10.6.39) to (10.6.22) gives

$$\varepsilon(n) = \tilde{e}(n) \tilde{\alpha}(n) \quad (10.6.41)$$

which shows that $\tilde{e}(n)$ is a scaled version of the a posteriori error. Starting with (10.6.40) and using (10.6.20), (10.6.16), and (10.5.22), we obtain

$$\tilde{\alpha}^2(n) = 1 - \tilde{\mathbf{w}}^H(n) \tilde{\mathbf{w}}(n) = 1 - \mathbf{x}^H(n) \mathbf{R}^{-1}(n) \mathbf{x}(n) = \alpha(n) \quad (10.6.42)$$

$$\text{or} \quad \tilde{\alpha}(n) = \sqrt{\alpha(n)} \quad (10.6.43)$$

which shows that $\tilde{\alpha}(n)$ is a normalized conversion factor. Since

$$\varepsilon(n) = \alpha(n) e(n) = \tilde{\alpha}^2(n) e(n) \quad (10.6.44)$$

using (10.6.41), we obtain

$$\tilde{e}(n) = \sqrt{e(n) \varepsilon(n)} \quad (10.6.45)$$

[†] $\langle \cdot \rangle_{ij}$ denotes the ij th element of a block matrix.

that is, $\tilde{e}(n)$ is the geometric mean of the a priori and a posteriori LS errors. Furthermore, (10.6.41) and (10.6.44) give

$$e(n) = \frac{\tilde{e}(n)}{\tilde{\alpha}(n)} \quad (10.6.46)$$

which also can be proved from (10.6.35) directly (see Problem 10.45).

In summary, to determine the updates of $\tilde{\mathbf{R}}(n)$ and $\tilde{\mathbf{k}}(n)$ of the Cholesky factors and the a priori error $e(n)$ we simply need to determine a unitary matrix $\mathbf{Q}(n)$ that annihilates the vector $\mathbf{x}^H(n)$ in (10.6.35). The construction of the matrix $\mathbf{Q}(n)$ is discussed later in Section 10.6.6.

10.6.4 Extended QR-RLS Algorithm

In applications that require the coefficient vector, we need to solve the upper triangular system $\tilde{\mathbf{R}}(n)\mathbf{c}(n) = \tilde{\mathbf{k}}(n)$ by back substitution. This method is not order-recursive and cannot be implemented in parallel. An alternative approach can be chosen by appending one more column to the matrices of the QR algorithm (10.6.35). To simplify the derivation, we combine the first column of (10.6.35) and the new column to construct the formula

$$\mathbf{Q}(n) \begin{bmatrix} \sqrt{\lambda} \tilde{\mathbf{R}}(n-1) & \tilde{\mathbf{R}}^{-H}(n-1)/\sqrt{\lambda} \\ \mathbf{x}^H(n) & \mathbf{0}^H \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{R}}(n) & \mathbf{D}(n) \\ \mathbf{0}^H & \tilde{\mathbf{g}}^H(n) \end{bmatrix} \quad (10.6.47)$$

where $\mathbf{D}(n)$ and $\tilde{\mathbf{g}}(n)$ are yet to be determined. Using Lemma 10.2, we obtain

$$\langle \mathbf{B}^H \mathbf{B} \rangle_{12} = \tilde{\mathbf{R}}^H(n) \mathbf{D}(n) = \mathbf{I} = \langle \mathbf{A}^H \mathbf{A} \rangle_{12} \quad (10.6.48)$$

which implies that $\mathbf{D}(n) = \tilde{\mathbf{R}}^{-H}(n)$ is the Cholesky factor of $\mathbf{R}^{-1}(n)$ and can be updated by using the same orthogonal transformation $\mathbf{Q}(n)$. Furthermore, we have

$$\langle \mathbf{B}^H \mathbf{B} \rangle_{22} = \tilde{\mathbf{R}}^{-1}(n) \tilde{\mathbf{R}}^{-H}(n) + \tilde{\mathbf{g}}(n) \tilde{\mathbf{g}}^H(n) = \frac{1}{\lambda} \tilde{\mathbf{R}}(n-1) \tilde{\mathbf{R}}^{-H}(n-1) = \langle \mathbf{A}^H \mathbf{A} \rangle_{22}$$

which, using (10.6.16), gives

$$\hat{\mathbf{R}}^{-1}(n) = \mathbf{P}(n) = \frac{1}{\lambda} \mathbf{P}(n-1) - \tilde{\mathbf{g}}(n) \tilde{\mathbf{g}}^H(n)$$

Comparing the last equation to (10.5.30) gives

$$\tilde{\mathbf{g}}(n) = \frac{\mathbf{g}(n)}{\sqrt{\alpha(n)}} = \frac{\mathbf{g}(n)}{\tilde{\alpha}(n)} \quad (10.6.49)$$

that is, $\tilde{\mathbf{g}}(n)$ is a scaled version of the RLS gain vector. Using (10.5.13) gives

$$\mathbf{c}(n) = \mathbf{c}(n-1) + \tilde{\mathbf{g}}(n) \tilde{e}^*(n) \quad (10.6.50)$$

which provides a time-updating formula for the filter coefficient vector. This method of updating the coefficient vector $\mathbf{c}(n)$ is known as the *extended QR-RLS algorithm* (Yang and Böhme 1992; Sayed and Kailath 1994). This algorithm is not widely used because the propagation of both $\tilde{\mathbf{R}}(n)$ and $\tilde{\mathbf{R}}^{-H}(n)$ may lead to numerical problems, especially in finite-precision implementations. This problem may be avoided by using the inverse QR-RLS algorithm, discussed next. Other methods of extracting the coefficient vector are discussed in Shepherd and McWhirter (1993).

10.6.5 Inverse QR-RLS Algorithm

From the CRLS algorithm we obtain

$$1 + \frac{1}{\lambda} \mathbf{x}^H(n) \mathbf{P}(n-1) \mathbf{x}(n) = \frac{1}{\alpha(n)} \quad (10.6.51)$$

$$\frac{1}{\lambda} \mathbf{P}(n-1) \mathbf{x}(n) = \frac{\mathbf{g}(n)}{\alpha(n)} \quad (10.6.52)$$

$$\frac{1}{\lambda} \mathbf{P}(n-1) = \mathbf{P}(n) + \frac{\mathbf{g}(n)}{\sqrt{\alpha(n)}} \frac{\mathbf{g}^H(n)}{\sqrt{\alpha(n)}} \quad (10.6.53)$$

which combined with the Cholesky decomposition

$$\mathbf{P}(n) = \hat{\mathbf{R}}^{-1}(n) = \tilde{\mathbf{R}}^{-1}(n) \tilde{\mathbf{R}}^{-H}(n) \quad (10.6.54)$$

leads to the following identity

$$\begin{aligned} & \begin{bmatrix} \frac{1}{\sqrt{\lambda}} \mathbf{x}^H(n) \tilde{\mathbf{R}}^{-1}(n-1) & 1 \\ \frac{1}{\sqrt{\lambda}} \tilde{\mathbf{R}}^{-1}(n-1) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{\lambda}} \tilde{\mathbf{R}}^{-H}(n-1) \mathbf{x}(n) & \frac{1}{\sqrt{\lambda}} \tilde{\mathbf{R}}^{-H}(n-1) \\ 1 & \mathbf{0}^H \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0}^H & \frac{1}{\sqrt{\alpha(n)}} \\ \tilde{\mathbf{R}}^{-1}(n) & \frac{\mathbf{g}(n)}{\sqrt{\alpha(n)}} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{R}}^{-H}(n) \\ \frac{1}{\sqrt{\alpha(n)}} & \frac{\mathbf{g}^H(n)}{\sqrt{\alpha(n)}} \end{bmatrix} \end{aligned} \quad (10.6.55)$$

where $\tilde{\mathbf{R}}^{-1}(n)$ is an upper triangular matrix. From (10.6.55) and Lemma 10.2 there is a unitary matrix $\mathbf{Q}(n)$ such that

$$\mathbf{Q}(n) \begin{bmatrix} \frac{1}{\sqrt{\lambda}} \tilde{\mathbf{R}}^{-H}(n-1) \mathbf{x}(n) & \frac{1}{\sqrt{\lambda}} \tilde{\mathbf{R}}^{-H}(n-1) \\ 1 & \mathbf{0}^H \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{R}}^{-H}(n) \\ \frac{1}{\sqrt{\alpha(n)}} & \frac{\mathbf{g}^H(n)}{\sqrt{\alpha(n)}} \end{bmatrix} \quad (10.6.56)$$

This shows that annihilating the vector $\tilde{\mathbf{R}}^{-H}(n-1) \mathbf{x}(n)/\sqrt{\lambda} = \tilde{\mathbf{w}}(n)/\sqrt{\lambda}$ updates the Cholesky factor $\tilde{\mathbf{R}}^{-H}(n)$, the normalized gain vector $\tilde{\mathbf{g}}(n)$, and the conversion factor $\tilde{\alpha}(n)$. Again, the only requirement of matrix $\mathbf{Q}(n)$ is to annihilate the row vector $\tilde{\mathbf{w}}^H(n)/\sqrt{\lambda}$. This algorithm, like the CRLS method, is initialized by setting $\tilde{\mathbf{P}}^H(-1) = \tilde{\mathbf{R}}^{-H}(-1) = \delta^{-1} \mathbf{I}$, where δ is a very small positive number.

10.6.6 Implementation of QR-RLS Algorithm Using the Givens Rotations

To develop a complete QRD-based RLS algorithm, we need to construct the matrix $\mathbf{Q}(n)$ that annihilates the vector $\mathbf{x}^H(n)$ on the left-hand side of (10.6.35). Since we do not need the vector $\tilde{\mathbf{w}}(n)$ and we can compute $\tilde{\alpha}(n)$ from matrix $\mathbf{Q}(n)$, as we shall see later, we work with the following part

$$\mathbf{Q}(n) \underbrace{\begin{bmatrix} \sqrt{\lambda} \tilde{\mathbf{R}}(n-1) & \sqrt{\lambda} \tilde{\mathbf{k}}(n-1) \\ \mathbf{x}^H(n) & y^*(n) \end{bmatrix}}_{\tilde{\mathbf{R}}(n)} = \begin{bmatrix} \tilde{\mathbf{R}}(n) & \tilde{\mathbf{k}}(n) \\ \mathbf{0}^H & \tilde{e}^*(n) \end{bmatrix} \quad (10.6.57)$$

and show how to annihilate the elements of $\mathbf{x}^H(n)$, one by one, using a sequence of M Givens rotations. We remind the reader that the matrix $\tilde{\mathbf{R}}(n-1)$ is upper triangular. We

start by constructing a Givens rotation matrix $\mathbf{G}^{(1)}(n)$ that operates on the first and last rows of $\bar{\mathbf{R}}(n)$ to annihilate the first element of $\mathbf{x}^H(n)$. More specifically, we wish to find a Givens rotation such that

$$\begin{aligned} & \begin{bmatrix} c_1 & \mathbf{0}^H & s_1^* \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ -s_1 & \mathbf{0}^H & c_1 \end{bmatrix} \begin{bmatrix} \sqrt{\lambda}\tilde{r}_{11}(n-1) & \sqrt{\lambda}\tilde{r}_{12}(n-1) & \cdots & \sqrt{\lambda}\tilde{r}_{1M}(n-1) & \sqrt{\lambda}\tilde{k}_1(n-1) \\ \mathbf{0} & \vdots & & \ddots & \vdots \\ x_1^*(n) & x_2^*(n) & \cdots & x_M^*(n) & y^*(n) \end{bmatrix} \\ &= \begin{bmatrix} \tilde{r}_{11}(n) & \tilde{r}_{12}(n) & \cdots & \tilde{r}_{1M}(n) & \tilde{k}_1(n) \\ \mathbf{0} & \vdots & \ddots & \vdots & \vdots \\ 0 & x_2^{(2)}(n) & \cdots & x_M^{(2)}(n) & y^{(2)}(n) \end{bmatrix} \end{aligned}$$

To this end, we use the first element of the first row and the first element of the last row to determine the rotation parameters c_1 and s_1 , and then we apply the rotation to the remaining M pairs of the two rows. Note that for consistency of notation we define $x_k^{(1)}(n) \triangleq x_k(n)$ and $y^{(1)}(n) \triangleq y(n)$. Then using $\sqrt{\lambda}\tilde{r}_{22}(n)$ and $x_2^{(2)}(n)$, we determine $\mathbf{G}^{(2)}(n)$ and annihilate the second element of the last row by operating on the $M - 1$ pairs of the second row and the last row of the matrix $\mathbf{G}^{(2)}(n)\bar{\mathbf{R}}(n)$.

In general, we use the elements $\sqrt{\lambda}\tilde{r}_{ii}(n)$ and $x_i^{(i)}(n)$ to determine the Givens rotation matrix $\mathbf{G}^{(i)}(n)$ that operates on the i th row and the last row of the rotated matrix $\mathbf{G}^{(i-1)}(n) \cdots \mathbf{G}^{(1)}(n)\bar{\mathbf{R}}(n)$ to annihilate the element $x_i^{(i)}(n)$. Therefore,

$$\begin{aligned} & \begin{bmatrix} c_i & s_i^* \\ -s_i & c_i \end{bmatrix} \begin{bmatrix} 0 & \cdots & 0 & \sqrt{\lambda}\tilde{r}_{ii}(n-1) & \cdots & \sqrt{\lambda}\tilde{r}_{iM}(n-1) & \sqrt{\lambda}\tilde{k}_i(n-1) \\ 0 & \cdots & 0 & x_i^{(i)}(n) & \cdots & x_M^*(n) & y^{(i)}(n) \end{bmatrix} \\ &= \begin{bmatrix} 0 & \cdots & 0 & \tilde{r}_{ii}(n) & \tilde{r}_{i,i+1}(n) & \cdots & \tilde{r}_{iM}(n) & \tilde{k}_i(n) \\ 0 & \cdots & 0 & 0 & x_{i+1}^{(i+1)}(n) & \cdots & x_M^{(i+1)}(n) & y^{(i+1)}(n) \end{bmatrix} \end{aligned} \quad (10.6.58)$$

where

$$c_i = \frac{\sqrt{\lambda}\tilde{r}_{ii}(n-1)}{\tilde{r}_{ii}(n)} \quad s_i = \frac{x_i^{(i)}(n)}{\tilde{r}_{ii}(n)} \quad (10.6.59)$$

and

$$\tilde{r}_{ii}(n) = [\lambda\tilde{r}_{ii}^2(n-1) + |x_i^{(i)}(n)|^2]^{1/2} \quad (10.6.60)$$

Thus, if we perform (10.6.58) for $i = 1, 2, \dots, M$, we annihilate the first M elements in the last row of $\bar{\mathbf{R}}(n)$ and convert $\bar{\mathbf{R}}(n)$ to the triangular matrix shown in (10.6.57). This process requires a total of $M(M + 1)/2$ Givens rotations. The orthogonalization matrix is

$$\mathbf{Q}(n) = \mathbf{G}^{(M)}(n) \cdots \mathbf{G}^{(2)}(n)\mathbf{G}^{(1)}(n) \quad (10.6.61)$$

$$\text{where } \mathbf{G}^{(i)}(n) = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & c_i(n) & s_i^*(n) \\ & & & 1 & & \\ & & & \vdots & \ddots & \vdots \\ & & & & 1 & \\ & & & -s_i(n) & \cdots & c_i(n) \end{bmatrix} \quad (10.6.62)$$

are $(M + 1) \times (M + 1)$ rotation matrices. Note that all off-diagonal elements, except those in the $(i, M + 1)$ and $(M + 1, i)$ locations, are zero.

From (10.6.35) we can easily see that $\tilde{\alpha}(n)$ equals the last diagonal element of $\mathbf{Q}(n)$. Furthermore, taking into consideration the special structure of $\mathbf{G}^{(i)}(n)$ and (10.6.61), we

obtain

$$\tilde{\alpha}(n) = \prod_{i=1}^M c_i(n) \quad (10.6.63)$$

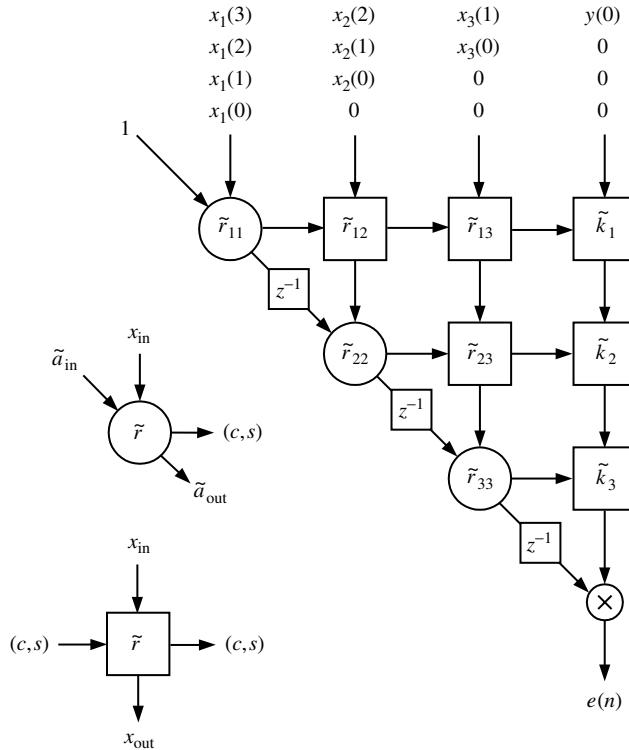
that is, $\tilde{\alpha}(n)$ is the product of the cosine terms in the M Givens rotations. This justifies the interpretation of $\tilde{\alpha}(n)$ and $\alpha(n) = \tilde{\alpha}^2(n)$ as angle variables.

Although the LS solution is not defined if $n < M$, the RLS Givens algorithm may be initialized by setting $\tilde{\mathbf{R}}(0) = \mathbf{0}$ and $\tilde{\mathbf{k}}(n) = \mathbf{0}$. The Givens rotation-based RLS algorithm is summarized in Table 10.9. The algorithm requires about $2M^2$ multiplications, $2M$ divisions, and M square roots per time update.

TABLE 10.9
The Givens rotation-based RLS algorithm.

Initialization
Set all elements $\tilde{r}_{ij}(-1) = 0, \tilde{k}_i(-1) = 0$
Time Recursion: $n = 0, 1, \dots$
$\tilde{e}(n) = y(n) \quad \tilde{\alpha}(n) = 1$
For $i = 1$ to M do
$\tilde{r}_{ii}(n) = \{\lambda\tilde{r}_{ii}^2(n-1) + x_i(n) ^2\}^{1/2}$
$c = \frac{\sqrt{\lambda}\tilde{r}_{ii}(n-1)}{\tilde{r}_{ii}(n)} \quad s = \frac{x_i(n)}{\tilde{r}_{ii}(n)}$
[If $\tilde{r}_{ii}(n) = 0$, set $c = 1$ and $s = 0$]
For $j = i + 1$ to M do
$\bar{x} = cx_j(n) - s\tilde{r}_{ij}(n-1)$
$\tilde{r}_{ij}(n) = c\tilde{r}_{ij}(n-1) + s^*x_j(n)$
$x_j(n) = \bar{x}$
End
$\bar{e} = c\tilde{e}(n) - s\tilde{k}_i(n-1)$
$\tilde{k}_i(n) = c\tilde{k}_i(n-1) + s^*\tilde{e}(n)$
$\tilde{e}(n) = \bar{e}$
$\tilde{\alpha}(n) = c\tilde{\alpha}(n)$
End
$\varepsilon(n) = \tilde{e}(n)\tilde{\alpha}(n)$ or $e(n) = \frac{\tilde{e}(n)}{\tilde{\alpha}(n)}$

The algorithm in Table 10.9 may be implemented in parallel using a triangular array of processors, as illustrated in Figure 10.36 for $M = 3$. At time $n - 1$, the elements of $\tilde{\mathbf{R}}(n - 1)$ and $\tilde{\mathbf{k}}(n - 1)$ are stored in the array elements. The arriving new input data $[\mathbf{x}^H(n) \ y^*(n)]$ are fed from the top and propagate downward. The Givens rotation parameters are calculated in the boundary cells and propagate from left to right. The internal cells receive the rotation parameters from the left, perform the rotation on the data from the top, and pass results to the cells at right and below. The angle variable $\tilde{\alpha}(n)$ is computed along the boundary cells and the a priori or a posteriori error at the last cell. This updating procedure is repeated at each time step upon the arrival of the new data. This structure was derived in McWhirter (1983) by eliminating the linear part used to determine the coefficient vector, by back substitution, from the systolic array introduced in Gentleman and Kung (1981) for the solution of general LS problems. Clearly, the array in Figure 10.36 performs two distinct functions: It propagates the matrix $\tilde{\mathbf{R}}(n)$ and the vector $\tilde{\mathbf{k}}(n)$ that define the LS array processor, and it performs,

**FIGURE 10.36**

Systolic array implementation of the QR-RLS algorithm and functional description of its processing elements.

although in a not-so-obvious way, the filtering operation by providing at the output the error $\varepsilon(n)$ or $e(n)$. Figure 10.36 provides a functional description of the processing elements only. In practice, there are different hardware and software implementations using systolic arrays, wavefront arrays, and CORDIC processors. More detailed descriptions can be found in McWhirter and Proudler (1993), Shepherd and McWhirter (1993), and Haykin (1996).

10.6.7 Implementation of Inverse QR-RLS Algorithm Using the Givens Rotations

If we define the vector

$$\bar{\mathbf{w}}(n) \triangleq \frac{1}{\sqrt{\lambda}} \tilde{\mathbf{R}}^{-H}(n-1) \mathbf{x}(n) \quad (10.6.64)$$

and the scalar $\hat{\alpha}^2(n) \triangleq \frac{1}{\alpha(n)} = 1 + \bar{\mathbf{w}}^H(n) \bar{\mathbf{w}}(n)$ (10.6.65)

we can express (10.6.56) as

$$\mathbf{Q}(n) \begin{bmatrix} \bar{\mathbf{w}}(n) & \frac{1}{\sqrt{\lambda}} \tilde{\mathbf{R}}^{-H}(n-1) \\ 1 & \mathbf{0}^H \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{R}}^{-H}(n) \\ \hat{\alpha}(n) & \tilde{\mathbf{g}}^H(n) \end{bmatrix} \quad (10.6.66)$$

where $\tilde{\mathbf{g}}(n)$ is the normalized gain vector (10.6.49). The matrix $\mathbf{Q}(n)$ will be chosen as a sequence of Givens rotation matrices $\mathbf{G}^{(i)}(n)$ defined in (10.6.62).

We first show that we can determine the angle parameters $c_i(n)$ and $s_i(n)$ of $\mathbf{G}^{(i)}(n)$ using only the elements of $\bar{\mathbf{w}}(n)$. To this end, we choose the angle parameters of the rotation

matrix $\mathbf{G}^{(1)}(n)$ in

$$\mathbf{G}^{(1)}(n) \begin{bmatrix} \bar{w}_1(n) \\ \bar{w}_2(n) \\ \vdots \\ \bar{w}_M(n) \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \bar{w}_2(n) \\ \vdots \\ \bar{w}_M(n) \\ \hat{\alpha}_1(n) \end{bmatrix} \quad (10.6.67)$$

to annihilate the first element $\bar{w}_1(n)$. Note that owing to the structure of $\mathbf{G}^{(1)}(n)$ the remaining elements of $\bar{\mathbf{w}}(n)$ are left unaffected. Since unitary transformations preserve the Euclidean norm of a vector, we can easily see that

$$\hat{\alpha}_1^2(n) = 1 + |\bar{w}_1(n)|^2$$

which expresses $\hat{\alpha}_1(n)$ in terms of $\bar{\mathbf{w}}(n)$. From the first and last equations in (10.6.67), we have the system

$$\begin{aligned} c_1(n)\bar{w}_1(n) + s_1^*(n) &= 0 \\ -s_1(n)\bar{w}_1(n) + c_1(n) &= \hat{\alpha}_1(n) \end{aligned}$$

whose solution

$$c_1(n) = \frac{1}{\hat{\alpha}_1(n)} \quad s_1(n) = -\frac{\bar{w}_1^*(n)}{\hat{\alpha}_1(n)}$$

provides the required parameters. Similarly, we can determine the rotation $\mathbf{G}^{(2)}(n)$ to annihilate the element $\bar{w}_2(n)$ of the vector on the right-hand side of (10.6.67). The required rotation parameters are

$$c_2(n) = \frac{\hat{\alpha}_1(n)}{\hat{\alpha}_2(n)} \quad s_2(n) = -\frac{\bar{w}_2^*(n)}{\hat{\alpha}_2(n)}$$

where $\hat{\alpha}_2^2(n) = 1 + |\bar{w}_1(n)|^2 + |\bar{w}_2(n)|^2 = \hat{\alpha}_1^2(n) + |\bar{w}_2(n)|^2$

provides a recursive formula for the computation of $\hat{\alpha}_i(n)$. The remaining elements of $\bar{\mathbf{w}}(n)$ can be annihilated by continuing in a similar way. In general, for $i = 1, 2, \dots, M$, we have

$$\hat{\alpha}_i(n) = [\hat{\alpha}_{i-1}^2(n) + |\bar{w}_i(n)|^2]^{1/2} \quad \hat{\alpha}_0(n) = 1 \quad (10.6.68)$$

$$c_i(n) = \frac{\hat{\alpha}_{i-1}(n)}{\hat{\alpha}_i(n)} \quad s_i(n) = -\frac{\bar{w}_i^*(n)}{\hat{\alpha}_i(n)} \quad (10.6.69)$$

and $\hat{\alpha}(n) = \hat{\alpha}_M(n)$.

Let us denote by $\tilde{p}_{ij}(n)$ the elements of matrix $\tilde{\mathbf{R}}^{-H}(n)$ and by $g_j^{(i)}(n)$ the elements of vector $\tilde{\mathbf{g}}^H(n)$ after the i th rotation. The first rotation updates the first element of the matrix $\tilde{\mathbf{R}}^{-H}(n-1)/\sqrt{\lambda}$ and modifies the first element of $\tilde{\mathbf{g}}^H(n)$. Indeed, from

$$\mathbf{G}^{(1)}(n) \begin{bmatrix} \frac{1}{\sqrt{\lambda}} \tilde{\mathbf{R}}^{-H}(n-1) \\ \mathbf{0}^H \end{bmatrix} = \begin{bmatrix} \tilde{p}_{11}(n) & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{(1)}(n) & 0 & \cdots & 0 \end{bmatrix} \quad (10.6.70)$$

we obtain

$$\tilde{p}_{11}(n) = \frac{1}{\sqrt{\lambda}} c_1(n) \tilde{p}_{11}(n-1)$$

$$g_1^{(1)}(n) = -\frac{1}{\sqrt{\lambda}} s_1(n) \tilde{p}_{11}(n-1)$$

Multiplication of (10.6.70) by $\mathbf{G}^{(2)}(n)$ updates the second row of $\tilde{\mathbf{R}}^{-H}(n-1)/\sqrt{\lambda}$ and modifies the first two elements of $\tilde{\mathbf{g}}^H(n)$. In general, the i th rotation updates the i th row of

$\tilde{\mathbf{R}}^{-H}(n-1)/\sqrt{\lambda}$ and modifies the i first elements of $\tilde{\mathbf{g}}^H(n)$ using the formulas

$$\tilde{p}_{ij}(n) = \frac{1}{\sqrt{\lambda}}c_i(n)\tilde{p}_{ij}(n-1) + s_i^*(n)g_j^{(i-1)}(n) \quad (10.6.71)$$

$$g_j^{(i)}(n) = c_i(n)g_j^{(i-1)}(n) - \frac{1}{\sqrt{\lambda}}s_i(n)\tilde{p}_{ij}(n-1) \quad (10.6.72)$$

for $1 \leq i \leq M$ and $1 \leq j \leq i$. These recursions are initialized with $g_j^{(i)}(n) = 0$, $1 \leq j \leq M$, $i \leq j$, and provide the required quantities after M rotations. The complete inverse QR-RLS algorithm is summarized in Table 10.10, whereas a systolic array implementation is discussed in Alexander and Ghirnikar (1993).

TABLE 10.10
Summary of the inverse QR-RLS Givens
algorithm.[†]

Initialization
$\mathbf{c}(-1) = \mathbf{x}(-1) = \mathbf{0}$ $\tilde{p}_{ij}(-1) = \delta \gg 1$
Time Recursion: $n = 0, 1, \dots$
$e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n)$
$g_j^{(i)}(n) = 0 \quad 1 \leq j \leq M, i \leq j$
$\hat{\alpha}_0(n) = 1$
For $i = 1$ to M do
$\bar{w}_i(n) = \frac{1}{\sqrt{\lambda}} \sum_{j=1}^i \tilde{p}_{ij}(n-1)x_j(n)$
$\hat{\alpha}_i(n) = [\hat{\alpha}_{i-1}^2(n) + \bar{w}_i(n) ^2]^{1/2}$
$c_i(n) = \frac{\hat{\alpha}_{i-1}(n)}{\hat{\alpha}_i(n)}$ $s_i(n) = -\frac{\bar{w}_i^*(n)}{\hat{\alpha}_i(n)}$
For $j = 1$ to i do
$\tilde{p}_{ij}(n) = \frac{1}{\sqrt{\lambda}}c_i(n)\tilde{p}_{ij}(n-1) + s_i^*(n)g_j^{(i-1)}(n)$
$g_j^{(i)}(n) = c_i(n)g_j^{(i-1)}(n) - \frac{1}{\sqrt{\lambda}}s_i(n)\tilde{p}_{ij}(n-1)$
End
End
$\tilde{e}(n) = \frac{e(n)}{\hat{\alpha}_M(n)}$
For $m = 1$ to M do
$c_m(n) = c_m(n-1) + g_j^{(i)*}(n)\tilde{e}^*(n)$
End

[†]The computations can be done “in-place” using temporary variables as shown in Table 10.9.

10.6.8 Classification of RLS Algorithms for Array Processing

Whereas the CRLS algorithm provides the basis for the introduction and performance evaluation of exact LS adaptive filters for array processing, the Givens rotation-based QR-RLS algorithms provide the most desirable implementation in terms of numerical behavior and ease of hardware implementation. However, there are many more algorithms that have interesting theoretical interpretations or may better serve the needs of particular applications. In general, we have the following types of RLS algorithms.

1. The CRLS algorithm, which is a fixed-order algorithm, updates the inverse $\mathbf{P}(n) = \hat{\mathbf{R}}^{-1}(n)$ of the correlation matrix and then computes the gain vector through a matrix-by-vector multiplication (see Section 10.5).
2. Power-domain square root algorithms propagate either $\tilde{\mathbf{R}}(n)$ or its inverse $\tilde{\mathbf{P}}(n) \triangleq \tilde{\mathbf{R}}^{-1}(n)$, using formulas derived from the Cholesky decomposition of $\hat{\mathbf{R}}(n)$ or $\mathbf{P}(n) = \hat{\mathbf{R}}^{-1}(n)$, respectively. They include two types:
 - a. Algorithms that propagate $\{\tilde{\mathbf{R}}(n), \tilde{\mathbf{k}}(n)\}$ (information filter approach) or $\{\tilde{\mathbf{R}}^{-1}(n), \tilde{\mathbf{k}}(n)\}$ (covariance filter approach[†]) and provide the a priori or a posteriori errors only.
 - b. Algorithms that propagate $\tilde{\mathbf{R}}(n)$ and compute $\mathbf{g}(n)$ by solving (10.6.25) or propagate $\tilde{\mathbf{R}}^{-1}(n)$ and compute $\mathbf{g}(n)$ in a matrix-by-vector multiplication. Both algorithms compute the parameter vector $\mathbf{c}(n)$ and the error $e(n)$ or $\varepsilon(n)$.
3. Amplitude-domain square root algorithms that propagate either $\tilde{\mathbf{R}}(n)$ (*QRD-based RLS*) or its inverse $\tilde{\mathbf{P}}(n) \triangleq \tilde{\mathbf{R}}^{-1}(n)$ (*inverse QRD-based RLS*) working directly with the data matrix $\Lambda(n)[\mathbf{X}(n) \mathbf{y}(n)]$. In both cases, we can develop algorithms providing only the error $e(n)$ or $\varepsilon(n)$ or both the errors and the parameter vector $\mathbf{c}(n)$.

Algorithms that propagate the Cholesky factor $\tilde{\mathbf{R}}(n)$ avoid the loss-of-symmetry problem and have better numerical properties because the condition number of $\tilde{\mathbf{R}}(n)$ equals the square root of the condition number of $\hat{\mathbf{R}}(n)$. Because QRD-based algorithms have superior numerical properties to their Cholesky counterparts, we have focused on RLS algorithms based on the QRD of the data set $\Lambda(n)[\mathbf{X}(n) \mathbf{y}(n)]$. More specifically we discussed QRD-based RLS algorithms using the Givens rotations. Other QRD-based RLS algorithms using the MGS (Ling et al. 1986) and Householder transformations (Liu et al. 1992; Steinhardt 1988; Rader and Steinhardt 1986) also have been developed but are not as widely used.

It is generally accepted that QR decomposition leads to the best methods for solving the LS problem (Golub and Van Loan 1996). It has been shown by simulation that the Givens rotation-based QR-RLS algorithm is numerically stable for $\lambda < 1$ and diverges for $\lambda = 1$ (Yang and Böhme 1992; Haykin 1996). This is the algorithm of choice for applications that require only the a priori or a posteriori errors. Since the extended QR-RLS algorithm propagates both $\tilde{\mathbf{R}}(n)$ and $\tilde{\mathbf{R}}^{-H}(n)$ independently from each other, in finite-precision implementations, the computed values of $\tilde{\mathbf{R}}(n)$ and $\tilde{\mathbf{R}}^{-H}(n)$ deviate from each other's Hermitian inverse. As a result of this numerical inconsistency, the algorithm becomes numerically unstable (Haykin 1996). To avoid this problem, we can use either the QR-RLS algorithm with back substitution “on the fly” or the inverse QR-RLS algorithm (Alexander and Ghirnikar 1993; Pan and Plemmons 1989). The updating of $\mathbf{c}(n)$ with this last algorithm can be implemented in systolic array form without interrupting the adaptation process.

If we factor out the diagonal elements of matrix $\tilde{\mathbf{R}}(n)$, obtained by QRD, we can express $\tilde{\mathbf{R}}(n)$ as

$$\tilde{\mathbf{R}}(n) = \mathbf{D}^{1/2}(n) \tilde{\mathbf{R}}_1(n) \quad (10.6.73)$$

where $\tilde{\mathbf{R}}_1(n)$ is an upper triangular matrix with unit diagonal elements, and

$$\mathbf{D}(n) \triangleq \text{diag}\{\tilde{r}_{11}^2(n), \tilde{r}_{22}^2(n), \dots, \tilde{r}_{MM}^2(n)\} \quad (10.6.74)$$

is a diagonal matrix with positive elements. We can easily see that $\tilde{\mathbf{R}}_1^H(n)$ and $\mathbf{D}(n)$ provide the factors of the LDU decomposition (10.6.14). It turns out that (10.6.73) provides the basis for various QRD-based RLS algorithms that do not require square root operations. In similar manner, the LDU decomposition makes possible the square root-free triangularization of $\hat{\mathbf{R}}(n)$ (see Section 6.3). All algorithms that use the Cholesky factor $\tilde{\mathbf{R}}(n)$ or its inverse $\tilde{\mathbf{R}}^{-1}(n)$ require square root operations, which we can avoid if we use the LDU decomposition factors $\tilde{\mathbf{R}}_1(n)$ and $\mathbf{D}(n)$. Because such algorithms have inferior numerical properties to their square

[†]The terms *information* and *covariance* filtering-type algorithms are used in the context of Kalman filter theory (Bierman 1977; Kailath 1981).

root counterparts and are more prone to overflow and underflow problems, and because square root operations are within the reach of current digital hardware, we concentrate on RLS algorithms that propagate the Cholesky factor or its inverse (Stewart and Chapman 1990). However, square root-free algorithms are very useful for VLSI implementations. The interested reader can find information about such algorithms in Bierman and Thornton (1977), Ljung and Soderstrom (1983), Bierman and Thornton (1977), and Hsieh et al. (1993).

A unified derivation of the various RLS algorithms using a state-space formulation and their correspondence with related Kalman filtering algorithms is given in Sayed and Kailath (1994, 1998) and in Haykin (1996).

All algorithms mentioned above hold for *arbitrary* input data vectors and require $O(M^2)$ arithmetic operations per time update. However, if the input data vector has a shift-invariant structure, all algorithms lead to simplified versions that require $O(M)$ arithmetic operations per time update. These algorithms, which can be used for LS FIR filtering and prediction applications, are discussed in the following section.

10.7 FAST RLS ALGORITHMS FOR FIR FILTERING

In Section 7.3 we exploited the shift invariance of the input data vector

$$\mathbf{x}_{m+1}(n) = \begin{bmatrix} \mathbf{x}_m(n) \\ x(n-m) \end{bmatrix} = \begin{bmatrix} x(n) \\ \mathbf{x}_m(n-1) \end{bmatrix} \quad (10.7.1)$$

to develop a lattice-ladder structure for optimum FIR filters and predictors. The determination of the optimum parameters (see Figure 7.3) required the LDL^H decomposition of the correlation matrix $\mathbf{R}(n)$ and the solution of three triangular systems at each time n . However, for stationary signals the optimum filter is time-invariant, and the coefficients of its direct or lattice-ladder implementation structure are evaluated only once, using the algorithm of Levinson.

The key for the development of order-recursive algorithms was the following order partitioning of the correlation matrix

$$\mathbf{R}_{m+1}(n) = \begin{bmatrix} \mathbf{R}_m(n) & \mathbf{r}_m^b(n) \\ \mathbf{r}_m^{bH}(n) & P_x(n-m) \end{bmatrix} = \begin{bmatrix} P_x(n) & \mathbf{r}_m^{fH}(n) \\ \mathbf{r}_m^f(n) & \mathbf{R}_m(n-1) \end{bmatrix} \quad (10.7.2)$$

which is a result of the shift-invariance property (10.7.1). The same partitioning can be obtained for the LS correlation matrix $\hat{\mathbf{R}}_m(n)$

$$\begin{aligned} \hat{\mathbf{R}}_{m+1}(n) &= \sum_{j=0}^n \lambda^{n-j} \mathbf{x}_{m+1}(j) \mathbf{x}_{m+1}^H(j) \\ &= \begin{bmatrix} \hat{\mathbf{R}}_m(n) & \hat{\mathbf{r}}_m^b(n) \\ \hat{\mathbf{r}}_m^{bH}(n) & E_x(n-m) \end{bmatrix} = \begin{bmatrix} E_x(n) & \hat{\mathbf{r}}_m^{fH}(n) \\ \hat{\mathbf{r}}_m^f(n) & \hat{\mathbf{R}}_m(n-1) \end{bmatrix} \end{aligned} \quad (10.7.3)$$

if we assume that $\mathbf{x}_m(-1) = \mathbf{0}$, a condition known as *prewindowing* (see Section 8.3). This condition is necessary to ensure the presence of the term $\hat{\mathbf{R}}_m(n-1)$ in the lower right corner partitioning of $\hat{\mathbf{R}}_{m+1}(n)$.

The identical forms of (10.7.2) and (10.7.3) imply that the order-recursive relations and the lattice-ladder structure developed in Section 7.3 for optimum FIR filters can be used for *prewindowed* LS FIR filters. Simply, the expectation operator $E\{\cdot\}$ should be replaced by the time-averaging operator $\sum_{j=0}^n \lambda^{n-j}(\cdot)$, and the term *power* should be replaced by the term *energy*, when we go from the optimum MSE to the LSE formulation.

In this section we exploit the shift invariance (10.7.1) and the time updating

$$\hat{\mathbf{R}}_m(n) = \lambda \hat{\mathbf{R}}_m(n-1) + \mathbf{x}_m(n) \mathbf{x}_m^H(n) \quad (10.7.4)$$

to develop the following types of fast algorithms with $O(M)$ complexity:

1. Fast *fixed-order* algorithms for RLS direct-form FIR filters by explicitly updating the gain vectors $\mathbf{g}(n)$ and $\bar{\mathbf{g}}(n)$.
2. Fast *order-recursive* algorithms for RLS FIR lattice-ladder filters by indirect or direct updating of their coefficients.
3. QR decomposition-based RLS lattice-ladder algorithms using the Givens rotation.

All relationships in Section 7.3 are valid for the prewindowed LS problem, but we replace P by E to emphasize the energy interpretation of the cost function. The quantities appearing in the partitionings given by (10.7.3) specify a prewindowed LS forward linear predictor $-\mathbf{a}_m$ and an LS backward linear predictor $-\mathbf{b}_m$. Table 10.11 shows the correspondences between general FIR filtering, FLP, and BLP. Using these correspondences and the normal equations for LS filtering, we can easily obtain the normal equations and the total LSE for the FLP and the BLP, which are also summarized in Table 10.11 (see Problem 10.28). We stress that the predictor parameters $\mathbf{a}_m(n)$ and $\mathbf{b}_m(n)$ are held fixed over the optimization interval $0 \leq j \leq n$.

TABLE 10.11
Summary and correspondences between LS FIR filtering, forward linear prediction, and backward linear prediction.

	FIR filter	FLP	BLP
Input data vector	$\mathbf{x}_m(n)$	$\mathbf{x}_m(n-1)$	$\mathbf{x}_m(n)$
Desired response	$y(n)$	$x(n)$	$x(n-m)$
Coefficient vector	$\mathbf{c}_m(n)$	$-\mathbf{a}_m(n)$	$-\mathbf{b}_m(n)$
Error	$\varepsilon_m(n) = y(n) - \mathbf{c}_m^H(n)\mathbf{x}_m(n)$	$\varepsilon_m^f(n) = x(n) + \mathbf{a}_m^H(n)\mathbf{x}_m(n-1)$	$\varepsilon_m^b(n) = x(n-m) + \mathbf{b}_m^H(n)\mathbf{x}_m(n)$
Cost function	$E_m(n) = \sum_{j=0}^n \lambda^{n-j} \varepsilon_m(j) ^2$	$E_m^f(n) = \sum_{j=0}^n \lambda^{n-j} \varepsilon_m^f(j) ^2$	$E_m^b(n) = \sum_{j=0}^n \lambda^{n-j} \varepsilon_m^b(j) ^2$
Normal equations	$\hat{\mathbf{R}}_m(n)\mathbf{c}_m(n) = \hat{\mathbf{d}}_m(n)$	$\hat{\mathbf{R}}_m(n-1)\mathbf{a}_m(n) = -\hat{\mathbf{r}}_m^f(n)$	$\hat{\mathbf{R}}_m(n)\mathbf{b}_m(n) = -\hat{\mathbf{r}}_m^b(n)$
LSE	$E_m(n) = E_y(n) - \mathbf{c}_m^H(n)\hat{\mathbf{d}}_m(n)$	$E_m^f(n) = E_x(n) + \mathbf{a}_m^H(n)\hat{\mathbf{r}}_m^f(n)$	$E_m^b(n) = E_x(n-m) + \mathbf{b}_m^H(n)\hat{\mathbf{r}}_m^b(n)$
Correlation matrix	$\hat{\mathbf{R}}_m(n) = \sum_{j=0}^n \lambda^{n-j} \mathbf{x}_m(j)\mathbf{x}_m^H(j)$	$\hat{\mathbf{R}}_m(n-1)$	$\hat{\mathbf{R}}_m(n)$
Cross-correlation vectors	$\hat{\mathbf{d}}_m(n) = \sum_{j=0}^n \lambda^{n-j} \mathbf{x}_m(j)y^*(j)$	$\hat{\mathbf{r}}_m^f(n) = \sum_{j=0}^n \lambda^{n-j} \mathbf{x}_m(j-1)x^*(j)$	$\hat{\mathbf{r}}_m^b(n) = \sum_{j=0}^n \lambda^{n-j} \mathbf{x}_m(j)x^*(j-m)$

Table 10.12 summarizes the a priori and a posteriori time updates for the LS FIR filter derived in Section 10.5. If we use the correspondences between general FIR filtering and linear prediction, we can easily deduce similar time-updating recursions for the FLP and the BLP. These updates, which are also discussed in Problem 10.29, are summarized in Table 10.12.

10.7.1 Fast Fixed-Order RLS FIR Filters

The major computational task in RLS filters is the computation of the gain vector $\mathbf{g}(n)$ or $\bar{\mathbf{g}}(n)$. The CRLS algorithm updates the inverse matrix $\hat{\mathbf{R}}^{-1}(n)$ and then determines the gain vector via a matrix-by-vector multiplication that results in $O(M^2)$ complexity. The only way to reduce the complexity from $O(M^2)$ to $O(M)$ is by directly updating the gain vectors. We next show how to develop such algorithms by exploiting the shift-invariant structure of the input data vector shown in (10.7.1).

Summary of LS time-updating relations using a priori and a posteriori errors.

	Equation	A priori time updating	A posteriori time updating
Gain	(a)	$\hat{\mathbf{R}}_m(n)\mathbf{g}_m(n) = \mathbf{x}_m(n)$	$\lambda\hat{\mathbf{R}}_m(n-1)\bar{\mathbf{g}}_m(n) = \mathbf{x}_m(n)$
Filter	(b)	$e_m(n) = y(n) - \mathbf{c}_m^H(n-1)\mathbf{x}_m(n)$	$\varepsilon_m(n) = y(n) - \mathbf{c}_m^H(n)\mathbf{x}_m(n)$
	(c)	$\mathbf{c}_m(n) = \mathbf{c}_m(n-1) + \mathbf{g}_m(n)e_m^*(n)$	$\mathbf{c}_m(n) = \mathbf{c}_m(n-1) + \bar{\mathbf{g}}_m(n)\varepsilon_m^*(n)$
	(d)	$E_m(n) = \lambda E_m(n-1) + \alpha_m(n) e_m(n) ^2$	$E_m(n) = \lambda E_m(n-1) + \frac{ \varepsilon_m(n) ^2}{\alpha_m(n)}$
FLP	(e)	$e_m^f(n) = x(n) + \mathbf{a}_m^H(n-1)\mathbf{x}_m(n-1)$	$\varepsilon_m^f(n) = x(n) + \mathbf{a}_m^H(n)\mathbf{x}_m(n-1)$
	(f)	$\mathbf{a}_m(n) = \mathbf{a}_m(n-1) - \mathbf{g}_m(n-1)e_m^{f*}(n)$	$\mathbf{a}_m(n) = \mathbf{a}_m(n-1) - \bar{\mathbf{g}}_m(n-1)\varepsilon_m^{f*}(n)$
	(g)	$E_m^f(n) = \lambda E_m^f(n-1) + \alpha_m(n-1) e_m^f(n) ^2$	$E_m^f(n) = \lambda E_m^f(n-1) + \frac{ \varepsilon_m^f(n) ^2}{\alpha_m(n-1)}$
BLP	(h)	$e_m^b(n) = x(n-m) + \mathbf{b}_m^H(n-1)\mathbf{x}_m(n)$	$\varepsilon_m^b(n) = x(n-m) + \mathbf{b}_m^H(n)\mathbf{x}_m(n)$
	(i)	$\mathbf{b}_m(n) = \mathbf{b}_m(n-1) - \mathbf{g}_m(n)e_m^{b*}(n)$	$\mathbf{b}_m(n) = \mathbf{b}_m(n-1) - \bar{\mathbf{g}}_m(n)\varepsilon_m^{b*}(n)$
	(j)	$E_m^b(n) = \lambda E_m^b(n-1) + \alpha_m(n) e_m^b(n) ^2$	$E_m^b(n) = \lambda E_m^b(n-1) + \frac{ \varepsilon_m^b(n) ^2}{\alpha_m(n)}$

Fast Kalman algorithm: Updating the gain $\mathbf{g}(n)$

Suppose that we know the gain

$$\mathbf{g}_m(n-1) = \hat{\mathbf{R}}_m^{-1}(n-1)\mathbf{x}_m(n-1) \quad (10.7.5)$$

and we wish to compute the gain

$$\mathbf{g}_m(n) = \hat{\mathbf{R}}_m^{-1}(n)\mathbf{x}_m(n) \quad (10.7.6)$$

at the next time instant by “adjusting” $\mathbf{g}_m(n-1)$, using the new data $\{\mathbf{x}_m(n), y(n)\}$.

If we use the matrix inversion by partitioning formulas (7.1.24) and (7.1.26) for matrix $\hat{\mathbf{R}}_{m+1}(n)$, we have

$$\hat{\mathbf{R}}_{m+1}(n) = \begin{bmatrix} \hat{\mathbf{R}}_m^{-1}(n) & \mathbf{0}_m \\ \mathbf{0}_m^H & 0 \end{bmatrix} + \frac{1}{E_m^b(n)} \begin{bmatrix} \mathbf{b}_m(n) \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{b}_m^H(n) & 1 \end{bmatrix} \quad (10.7.7)$$

$$\text{and } \hat{\mathbf{R}}_{m+1}(n) = \begin{bmatrix} 0 & \mathbf{0}_m^H \\ \mathbf{0}_m & \hat{\mathbf{R}}_m^{-1}(n) \end{bmatrix} + \frac{1}{E_m^f(n)} \begin{bmatrix} 1 \\ \mathbf{a}_m(n) \end{bmatrix} \begin{bmatrix} 1 & \mathbf{a}_m^H(n) \end{bmatrix} \quad (10.7.8)$$

as was shown in Section 7.1.

Using (10.7.7), the first partitioning in (10.7.1), and the definition of $\varepsilon_m^b(n)$ from Table 10.12, we obtain

$$\mathbf{g}_{m+1}(n) = \begin{bmatrix} \mathbf{g}_m(n) \\ 0 \end{bmatrix} + \frac{\varepsilon_m^b(n)}{E_m^b(n)} \begin{bmatrix} \mathbf{b}_m(n) \\ 1 \end{bmatrix} \quad (10.7.9)$$

which provides a *pure order update* of the gain vector $\mathbf{g}_m(n)$. Similarly, using (10.7.8), the second partitioning in (10.7.1), and the definition of $\varepsilon_m^f(n)$ from Table 10.12, we have

$$\mathbf{g}_{m+1}(n) = \begin{bmatrix} 0 \\ \mathbf{g}_m(n-1) \end{bmatrix} + \frac{\varepsilon_m^f(n)}{E_m^f(n)} \begin{bmatrix} 1 \\ \mathbf{a}_m(n) \end{bmatrix} \quad (10.7.10)$$

which provides a *combined order and time update* of the gain vector $\mathbf{g}_m(n)$. This is the key to the development of fast algorithms for updating the gain vector.

Given the gain $\mathbf{g}_m(n-1)$, first we compute $\mathbf{g}_{m+1}(n)$, using (10.7.10). Then we compute $\mathbf{g}_m(n)$ from the first m equations of (10.7.9) as

$$\mathbf{g}_m(n) = \mathbf{g}_{m+1}^{[m]}(n) - g_{m+1}^{(m+1)}(n)\mathbf{b}_m(n) \quad (10.7.11)$$

because

$$g_{m+1}^{(m+1)}(n) = \frac{\varepsilon_m^b(n)}{E_m^b(n)} \quad (10.7.12)$$

from the last equation in (10.7.9). The updatings (10.7.9) and (10.7.10) require time updatings for the predictors $\mathbf{a}_m(n)$ and $\mathbf{b}_m(n)$ and the minimum error energies $E_m^f(n)$ and $E_m^b(n)$, which are given in Table 10.12. The only remaining problem is the coupling between $\mathbf{g}_m(n)$ in (10.7.11) and $\mathbf{b}_m(n)$ in

$$\mathbf{b}_m(n) = \mathbf{b}_m(n-1) - \mathbf{g}_m(n)e_m^{b*}(n) \quad (10.7.13)$$

which can be avoided by eliminating $\mathbf{b}_m(n)$. Carrying out the elimination, we obtain

$$\mathbf{g}_m(n) = \frac{\mathbf{g}_{m+1}^{[m]}(n) - g_{m+1}^{(m+1)}(n)\mathbf{b}_m(n-1)}{1 - g_{m+1}^{(m+1)}(n)e_m^{b*}(n)} \quad (10.7.14)$$

which provides the last step required to complete the updating. This approach, which is known as the *fast Kalman algorithm*, was developed in Falconer and Ljung (1978) using the ideas introduced by Morf (1974). To emphasize the fixed-order nature of the algorithm, we set $m = M$ and drop the order subscript for all quantities of order M . The computational organization of the algorithm, which requires $9M$ operations per time updating, is summarized in Table 10.13.

TABLE 10.13
Fast Kalman algorithm for time updating of LS FIR filters.

Equation	Computation
Old estimates: $\mathbf{a}(n-1), \mathbf{b}(n-1), \mathbf{g}(n-1), \mathbf{c}(n-1), E^f(n-1)$ New data: $\{\mathbf{x}(n), y(n)\}$	
Gain and predictor update	
(a)	$e^f(n) = x(n) + \mathbf{a}^H(n-1)\mathbf{x}(n-1)$
(b)	$\mathbf{a}(n) = \mathbf{a}(n-1) - \mathbf{g}(n-1)e^f(n)$
(c)	$\varepsilon^f(n) = x(n) + \mathbf{a}^H(n)\mathbf{x}(n-1)$
(d)	$E^f(n) = \lambda E^f(n-1) + \varepsilon^f(n)e^f(n)$
(e)	$\mathbf{g}_{M+1}(n) = \begin{bmatrix} 0 \\ \mathbf{g}(n-1) \end{bmatrix} + \frac{\varepsilon^f(n)}{E^f(n)} \begin{bmatrix} 1 \\ \mathbf{a}(n) \end{bmatrix}$
(f)	$e^b(n) = x(n-M) + \mathbf{b}^H(n-1)\mathbf{x}(n)$
(g)	$\mathbf{g}(n) = \frac{\mathbf{g}_{M+1}^{[M]}(n) - g_{M+1}^{(M+1)}(n)\mathbf{b}(n-1)}{1 - g_{M+1}^{(M+1)}(n)e^b(n)}$
(h)	$\mathbf{b}(n) = \mathbf{b}(n-1) - \mathbf{g}(n)e^b(n)$
Filter update	
(i)	$e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n)$
(j)	$\mathbf{c}(n) = \mathbf{c}(n-1) + \mathbf{g}(n)e^*(n)$

The FAEST algorithm: Updating the gain $\bar{\mathbf{g}}(n)$

In a similar way we can update the gain vector

$$\bar{\mathbf{g}}_m(n) = \frac{1}{\lambda} \hat{\mathbf{R}}_m^{-1}(n-1) \mathbf{x}_m(n) \quad (10.7.15)$$

by using (10.7.9) and (10.7.10). Indeed, using (10.7.10) with the lower partitioning (10.7.1) and (10.7.9) with the upper partitioning (10.7.1), we obtain

$$\bar{\mathbf{g}}_{m+1}(n) = \begin{bmatrix} 0 \\ \bar{\mathbf{g}}_m(n-1) \end{bmatrix} + \frac{e_m^f(n)}{\lambda E_m^f(n-1)} \begin{bmatrix} 1 \\ \mathbf{a}_m(n-1) \end{bmatrix} \quad (10.7.16)$$

and $\bar{\mathbf{g}}_{m+1}(n) = \begin{bmatrix} \bar{\mathbf{g}}_m(n) \\ 0 \end{bmatrix} + \frac{e_m^b(n)}{\lambda E_m^b(n-1)} \begin{bmatrix} \mathbf{b}_m(n-1) \\ 1 \end{bmatrix}$ (10.7.17)

which provide a link between $\bar{\mathbf{g}}_m(n-1)$ and $\bar{\mathbf{g}}_m(n)$. From (10.7.17) we obtain

$$\bar{\mathbf{g}}_m(n) = \bar{\mathbf{g}}_{m+1}^{[m]}(n) - \bar{g}_{m+1}^{(m+1)}(n)\mathbf{b}_m(n-1) \quad (10.7.18)$$

because

$$\bar{g}_{m+1}^{(m+1)}(n) = \frac{e_m^b(n)}{\lambda E_m^b(n-1)} \quad (10.7.19)$$

from the last row of (10.7.17). The fundamental difference between (10.7.9) and (10.7.17) is that the presence of $\mathbf{b}_m(n-1)$ in the latter breaks the coupling between gain vector and backward predictor. Furthermore, (10.7.19) can be used to compute $e_m^b(n)$ by

$$e_m^b(n) = \lambda E_m^b(n-1) \bar{g}_{m+1}^{(m+1)}(n) \quad (10.7.20)$$

with only two multiplications.

The time updatings of the predictors using the gain $\bar{\mathbf{g}}_m(n)$, which are given in Table 10.12, require the a posteriori errors that can be computed from the a priori errors by using the conversion factor

$$\bar{\alpha}_m(n) = 1 + \bar{\mathbf{g}}_m^H(n)\mathbf{x}_m(n) \quad (10.7.21)$$

which should be updated in time as well. This can be achieved by a two-step procedure as follows. First, using (10.7.16) and the lower partitioning (10.7.1), we obtain

$$\bar{\alpha}_{m+1}(n) = \bar{\alpha}_m(n-1) + \frac{|e_m^f(n)|^2}{\lambda E_m^f(n-1)} \quad (10.7.22)$$

which is a combined time and order updating. Then we use (10.7.17) and the upper partitioning (10.7.1) to obtain

$$\bar{\alpha}_m(n) = \bar{\alpha}_{m+1}(n) - \bar{g}_{m+1}^{(m+1)}(n)e_m^{b*}(n) \quad (10.7.23)$$

or

$$\bar{\alpha}_m(n) = \bar{\alpha}_{m+1}(n) - \frac{|e_m^b(n)|^2}{\lambda E_m^b(n-1)} \quad (10.7.24)$$

which in conjunction with (10.7.22) provides the required time update $\bar{\alpha}_m(n-1) \rightarrow \bar{\alpha}_{m+1}(n) \rightarrow \bar{\alpha}_m(n)$.

This leads to the *fast a posteriori error sequential technique (FAEST)* algorithm presented in Table 10.14, which was introduced in Carayannis et al. (1983). The FAEST algorithm requires only $7M$ operations per time update and is the most efficient known algorithm for prewindowed RLS FIR filters.

Fast transversal filter (FTF) algorithm. This is an a posteriori type of algorithm obtained from the FAEST by using the conversion factor

$$\alpha_m(n) = 1 - \mathbf{g}_m^H(n)\mathbf{x}_m(n) \quad (10.7.25)$$

instead of the conversion factor $\bar{\alpha}_m(n) = 1/\alpha_m(n)$. Using the Levinson recursions (10.7.9) and (10.7.10) in conjunction with the upper and lower partitionings in (10.7.1), we obtain

$$\alpha_{m+1}(n) = \alpha_m(n) - \frac{|\varepsilon_m^b(n)|^2}{E_m^b(n)} \quad (10.7.26)$$

and

$$\alpha_{m+1}(n) = \alpha_m(n-1) - \frac{|\varepsilon_m^f(n)|^2}{E_m^f(n)} \quad (10.7.27)$$

respectively. To obtain the FTF algorithm, we replace $\bar{\alpha}_m(n)$ in Table 10.14 by $1/\alpha_m(n)$ and Equation (h) by (10.7.27). To obtain $\alpha_m(n)$ from $\alpha_{m+1}(n)$, we cannot use (10.7.26) because

TABLE 10.14
FAEST algorithm for time updating of LS FIR filters.

Equation	Computation
	Old estimates: $\mathbf{a}(n-1)$, $\mathbf{b}(n-1)$, $\mathbf{c}(n-1)$, $\bar{\mathbf{g}}(n-1)$, $E^f(n-1)$, $E^b(n-1)$, $\bar{\alpha}(n-1)$
	New data: $\{\mathbf{x}(n), y(n)\}$
Gain and predictor update	
(a)	$e^f(n) = \mathbf{x}(n) + \mathbf{a}^H(n-1)\mathbf{x}(n-1)$
(b)	$\varepsilon^f(n) = \frac{e^f(n)}{\bar{\alpha}(n-1)}$
(c)	$\mathbf{a}(n) = \mathbf{a}(n-1) - \bar{\mathbf{g}}(n-1)\varepsilon^{f*}(n)$
(d)	$E^f(n) = \lambda E^f(n-1) + \varepsilon^f(n)e^{f*}(n)$
(e)	$\bar{\mathbf{g}}_{M+1}(n) = \begin{bmatrix} 0 \\ \bar{\mathbf{g}}(n-1) \end{bmatrix} + \frac{e^f(n)}{\lambda E^f(n-1)} \begin{bmatrix} 1 \\ \mathbf{a}(n-1) \end{bmatrix}$
(f)	$e^b(n) = \lambda E^b(n-1) \bar{g}_{M+1}^{(M+1)}(n)$
(g)	$\bar{\mathbf{g}}(n) = \bar{\mathbf{g}}_{M+1}^{[M]}(n) - \bar{g}_{M+1}^{(M+1)}(n)\mathbf{b}(n-1)$
(h)	$\bar{\alpha}_{M+1}(n) = \bar{\alpha}(n-1) + \frac{ e^f(n) ^2}{\lambda E^f(n-1)}$
(i)	$\bar{\alpha}(n) = \bar{\alpha}_{M+1}(n) - \bar{g}_{M+1}^{(M+1)*}(n)e^b(n)$
(j)	$\mathbf{b}(n) = \mathbf{b}(n-1) - \bar{\mathbf{g}}(n)\varepsilon^{b*}(n)$
(k)	$\varepsilon^b(n) = \frac{e^b(n)}{\bar{\alpha}(n)}$
(l)	$E^b(n) = \lambda E^b(n-1) + \varepsilon^b(n)e^{b*}(n)$
Filter update	
(m)	$e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n)$
(n)	$\varepsilon(n) = \frac{e(n)}{\bar{\alpha}(n)}$
(o)	$\mathbf{c}(n) = \mathbf{c}(n-1) + \bar{\mathbf{g}}(n)\varepsilon^*(n)$

it requires quantities dependent on $\alpha_m(n)$. To avoid this problem, we replace Equation (i) by the following relation

$$\alpha_m(n) = \frac{\alpha_{m+1}(n)}{1 - \alpha_{m+1}(n)\bar{g}_{m+1}^{(m+1)}(n)e_m^{b*}(n)} \quad (10.7.28)$$

obtained by combining (10.7.24), (10.7.19), and $\bar{\alpha}_m(n) = 1/\alpha_m(n)$. This algorithm, which has the same complexity as FAEST, was introduced in Cioffi and Kailath (1984) using a geometric derivation, and is known as the *fast transversal filter (FTF) algorithm*.

An alternative updating to (10.7.27) can be obtained by noticing that

$$\begin{aligned} \alpha_{m+1}(n) &= \alpha_m(n-1) - \alpha_m^2(n-1) \frac{|e_m^f(n)|^2}{E_m^f(n)} \\ &= \frac{\alpha_m(n-1)}{E_m^f(n)} [E_m^f(n) - \alpha_m(n-1)|e_m^f(n)|^2] \end{aligned} \quad (10.7.29)$$

or equivalently

$$\alpha_{m+1}(n) = \alpha_m(n-1) \frac{\lambda E_m^f(n-1)}{E_m^f(n)} \quad (10.7.29)$$

which can be used instead of (10.7.27) in the FTF algorithm. In a similar way, we can show

that

$$\alpha_{m+1}(n) = \alpha_m(n) \frac{\lambda E_m^b(n-1)}{E_m^b(n)} \quad (10.7.30)$$

which will be used later.

Some practical considerations

Figure 10.37 shows the realization of an adaptive RLS filter using the direct-form structure. The coefficient updating can be done using any of the introduced fast RLS algorithms. Some issues related to the implementation of these filters using multiprocessing are discussed in Problem 10.48.

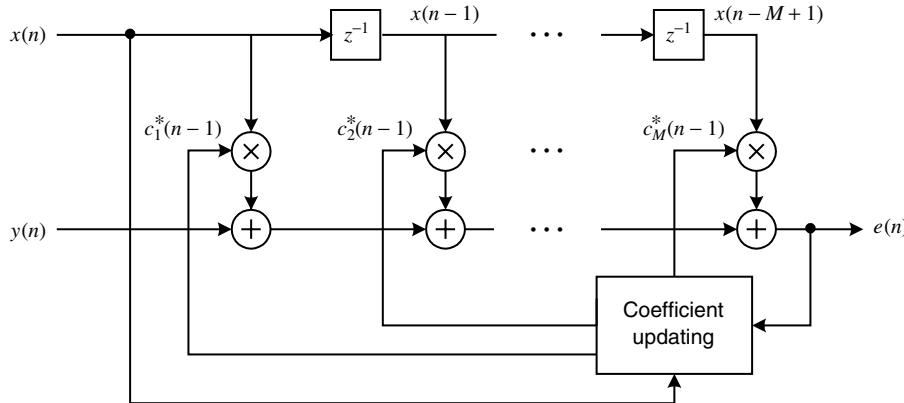


FIGURE 10.37

Implementation of an adaptive FIR filter using a direct-form structure.

In practice, the fast direct-form RLS algorithms are initialized at $n = 0$ by setting

$$\begin{aligned} E^f(-1) &= E^b(-1) = \delta > 0 \\ \alpha(-1) &= 1 \quad \text{or} \quad \bar{\alpha}(-1) = 1 \end{aligned} \quad (10.7.31)$$

and all other quantities equal to zero. The constant δ is chosen as a small positive number on the order of $0.01\sigma_x^2$ (Hubing and Alexander 1991). For $\lambda < 1$, the effects of the initial conditions are quickly “forgotten.” An exact initialization method is discussed in Problem 10.31.

Although the fast direct-form RLS algorithms have the lowest computational complexity, they suffer from numerical instability when $\lambda < 1$ (Ljung and Ljung 1985). When these algorithms are implemented with finite precision, the exact algebraic relations used for their derivation breakdown and lead to numerical problems.

There are two ways to deal with stabilization of the fast direct-form RLS algorithms. In the first approach, we try to identify precursors of ill behavior (warnings) and then use appropriate rescue operations to restore the normal operation of the algorithm (Lin 1984; Cioffi and Kailath 1984). One widely used rescue variable is

$$\eta_m(n) \triangleq \frac{\alpha_{m+1}(n)}{\alpha_m(n)} = \frac{\lambda E_m^b(n-1)}{E_m^b(n)} \quad (10.7.32)$$

which satisfies $0 \leq \eta_m(n) \leq 1$ for infinite-precision arithmetic (see Problem 10.33 for more details).

In the second approach, we exploit the fact that certain algorithmic quantities can be computed in two different ways. Therefore, we could use their difference, which provides a measure of the numerical errors, to change the dynamics of the error propagation system

and stabilize the algorithm. For example, both $e_m^b(n)$ and $\alpha_m(n)$ can be computed either using their definition or simpler order-recursions. This approach has been used to obtain stabilized algorithms with complexities $9M$ and $8M$; however, their performance is highly dependent on proper initialization (Slock and Kailath 1991, 1993).

10.7.2 RLS Lattice-Ladder Filters

The lattice-ladder structure[†] derived in Section 7.3 using the MSE criterion, due to the similarity of (10.7.2) and (10.7.3), holds for the prewindowed LSE criterion as well. This structure, which is depicted in Figure 10.38 for the a posteriori error case, is described by the following equations

$$\varepsilon_0^f(n) = \varepsilon_0^b(n) = x(n) \quad (10.7.33)$$

$$\varepsilon_{m+1}^f(n) = \varepsilon_m^f(n) + k_m^{f*}(n)\varepsilon_m^b(n-1) \quad 0 \leq m < M-1 \quad (10.7.33)$$

$$\varepsilon_{m+1}^b(n) = \varepsilon_m^b(n-1) + k_m^{b*}(n)\varepsilon_m^f(n) \quad 0 \leq m < M-1 \quad (10.7.34)$$

for the lattice part and

$$\varepsilon_0(n) = y(n) \quad (10.7.35)$$

$$\varepsilon_{m+1}(n) = \varepsilon_m(n) - k_m^{c*}(n)\varepsilon_m^b(n) \quad 0 \leq m \leq M-1$$

for the ladder part. The lattice parameters are given by

$$k_m^f(n) = -\frac{\beta_m(n)}{E_m^b(n-1)} \quad (10.7.36)$$

and

$$k_m^b(n) = -\frac{\beta_m^*(n)}{E_m^f(n)} \quad (10.7.37)$$

and the ladder parameters by

$$k_m^c(n) = \frac{\beta_m^c(n)}{E_m^b(n)} \quad (10.7.38)$$

where

$$\beta_m(n) = \mathbf{b}_m^H(n-1)\mathbf{r}_m^f(n) + r_{m+1}^f(n) \quad (10.7.39)$$

and

$$\beta_m^c(n) = \mathbf{b}_m^H(n)\mathbf{d}_m(n) + d_{m+1}(n) \quad (10.7.40)$$

are the partial correlation parameters.

However, as we recall, the time updating of the minimum LSE energies and the partial correlations is possible only if there is a time update for the correlation matrix $\mathbf{R}_m(n)$ and the cross-correlation vector $\mathbf{d}_m(n)$.

The minimum LSE energies can be updated in time using

$$E_m^f(n) = \lambda E_m^f(n-1) + e_m^f(n)\varepsilon_m^{f*}(n) \quad (10.7.41)$$

$$E_m^b(n) = \lambda E_m^b(n-1) + e_m^b(n)\varepsilon_m^{b*}(n) \quad (10.7.42)$$

or their variations, given in Table 10.12.

To update the partial correlation $\beta_m(n)$, we start with the definition (10.7.39) and then use the time-updating formulas for all involved quantities, rearranging and recombining

[†] In Chapter 7 we used the symbol $e(n)$ because we had no need to distinguish between a priori and a posteriori errors. However, since the error $e(n)$ in Section 7.3 is an a posteriori error, we now use the symbol $\varepsilon(n)$.

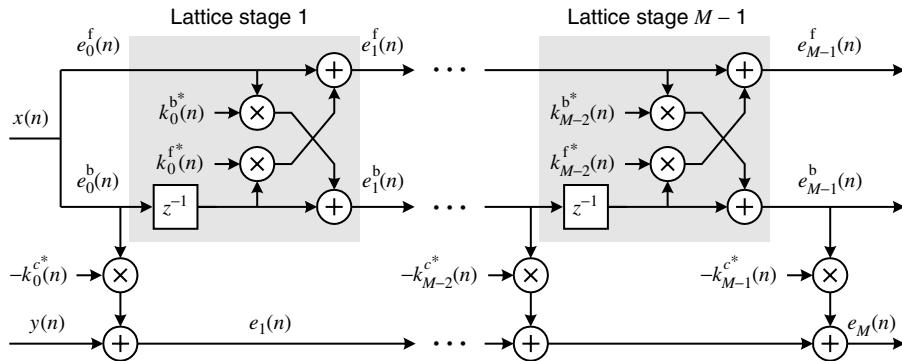


FIGURE 10.38

A posteriori error RLS lattice-ladder filter.

terms as follows:

$$\begin{aligned}
 \beta_m(n+1) &= \mathbf{b}_m^H(n)\mathbf{r}_m^f(n+1) + r_{m+1}^f(n+1) \\
 &= \mathbf{b}_m^H(n)[\lambda\mathbf{r}_m^f(n) + \mathbf{x}_m(n)x^*(n+1)] \\
 &\quad + [\lambda r_{m+1}^f(n) + x(n-m)x^*(n+1)] \\
 &= \lambda\mathbf{b}_m^H(n)\mathbf{r}_m^f(n) + \varepsilon_m^b(n)x^*(n+1) + \lambda r_{m+1}^f(n) \\
 &= \lambda[\mathbf{b}_m^H(n-1) - \varepsilon_m^b(n)\bar{\mathbf{g}}_m(n)]\mathbf{r}_m^f(n) \\
 &\quad + \lambda r_{m+1}^f(n) + \varepsilon_m^b(n)x^*(n+1) \\
 &= \lambda\beta_m(n) + \varepsilon_m^b(n)[x^*(n+1) - \lambda\bar{\mathbf{g}}_m(n)\mathbf{r}_m^f(n)] \\
 &= \lambda\beta_m(n) + \varepsilon_m^b(n)[x^*(n+1) - \mathbf{x}_m^H(n)\mathbf{R}_m^{-1}(n-1)\mathbf{r}_m^f(n)] \\
 &= \lambda\beta_m(n) + \varepsilon_m^b(n)[x^*(n+1) + \mathbf{x}_m^H(n)\mathbf{a}(n)] \\
 &= \lambda\beta_m(n) + \varepsilon_m^b(n)e_m^{f*}(n+1)
 \end{aligned}$$

which provides the desired update formula. The updating

$$\beta_m(n) = \lambda\beta_m(n-1) + \varepsilon_m^b(n-1)e_m^{f*}(n) \quad (10.7.43)$$

$$= \lambda\beta_m(n-1) + \frac{1}{\alpha_m(n-1)}\varepsilon_m^b(n-1)\varepsilon_m^{f*}(n) \quad (10.7.44)$$

is feasible because the right-hand side involves already-known quantities.

In a similar way (see Problem 10.36), we can show that

$$\beta_m^c(n) = \lambda\beta_m^c(n-1) + \varepsilon_m^b(n)e_m^{f*}(n) \quad (10.7.45)$$

$$= \lambda\beta_m^c(n-1) + \frac{1}{\alpha_m(n)}\varepsilon_m^b(n)\varepsilon_m^{f*}(n) \quad (10.7.46)$$

which facilitates the updating of the ladder parameters.

To obtain an a posteriori algorithm, we need the conversion factor $\alpha_m(n)$, which can be obtained using the order-recursive formula (10.7.26). A detailed organization of the a posteriori LS lattice-ladder algorithm, which requires about $20M$ operations per time update, is given in Table 10.15. The initialization of the algorithm is easily obtained from the definitions of the corresponding quantities. The condition $\alpha_0(n-1) = 1$ follows from (10.7.25), and the positive constant δ is chosen to ensure the invertibility of the LS correlation matrix $\mathbf{R}(n)$ (see Section 10.5). The time-updating recursions (c) and (d) can be replaced by order recursions, as explained in Problem 10.37.

TABLE 10.15
**Computational organization of a posteriori
LS lattice-ladder algorithm.**

Equation	Computation
Time initialization ($n = 0$)	
	$E_m^f(-1) = E_m^b(-1) = \delta > 0 \quad 0 \leq m < M - 1$
	$\beta_m(-1) = 0, \varepsilon_m^b(-1) = 0 \quad 0 \leq m < M - 1$
	$\beta_m^c(-1) = 0 \quad 0 \leq m \leq M - 1$
Order initialization	
(a)	$\varepsilon_0^f(n) = \varepsilon_0^b(n) = x(n) \quad \varepsilon_0(n) = y(n) \quad \alpha_0(n-1) = 1$
Lattice part: $m = 0, 1, \dots, M - 1$	
(b)	$\beta_m(n) = \lambda \beta_m(n-1) + \frac{\varepsilon_m^b(n-1) \varepsilon_m^{f*}(n)}{\alpha_m(n-1)}$
(c)	$E_m^f(n) = \lambda E_m^f(n-1) + \frac{ \varepsilon_m^f(n) ^2}{\alpha_m(n-1)}$
(d)	$E_m^b(n) = \lambda E_m^b(n-1) + \frac{ \varepsilon_m^b(n) ^2}{\alpha_m(n)}$
(e)	$k_m^f(n) = \frac{-\beta_m(n)}{E_m^b(n-1)}$
(f)	$k_m^b(n) = \frac{-\beta_m^*(n)}{E_m^f(n)}$
(g)	$\varepsilon_{m+1}^f(n) = \varepsilon_m^f(n) + k_m^{f*}(n) \varepsilon_m^b(n-1)$
(h)	$\varepsilon_{m+1}^b(n) = \varepsilon_m^b(n-1) + k_m^{b*}(n) \varepsilon_m^f(n)$
(i)	$\alpha_{m+1}(n) = \alpha_m(n) - \frac{ \varepsilon_m^b(n) ^2}{E_m^b(n)}$
Ladder part: $m = 1, 2, \dots, M$	
(j)	$\beta_m^c(n) = \lambda \beta_m^c(n-1) + \varepsilon_m^b(n) \varepsilon_m^{f*}(n) / \alpha_m(n)$
(k)	$k_m^c(n) = \frac{\beta_m^c(n)}{E_m^b(n)}$
(l)	$\varepsilon_{m+1}(n) = \varepsilon_m(n) - k_m^{c*}(n) \varepsilon_m^b(n)$

If instead of the a posteriori errors we use the a priori ones, we obtain the following recursions

$$e_0^f(n) = e_0^b(n) = x(n) \quad (10.7.47)$$

$$e_{m+1}^f(n) = e_m^f(n) + k_m^{f*}(n-1) e_m^b(n-1) \quad 0 \leq m < M-1 \quad (10.7.47)$$

$$e_{m+1}^b(n) = e_m^b(n-1) + k_m^{b*}(n-1) e_m^f(n) \quad 0 \leq m < M-1 \quad (10.7.48)$$

for the lattice part and

$$e_0(n) = y(n) \quad (10.7.49)$$

$$e_{m+1}(n) = e_m(n) - k_m^{c*}(n-1) e_m^b(n) \quad 1 \leq m \leq M$$

for the ladder part (see Problem 10.38). As expected, the a priori structure uses the old LS estimates of the lattice-ladder parameters. Based on these recursions, we can develop the a priori error RLS lattice-ladder algorithm shown in Table 10.16, which requires about $20M$ operations per time update.

Computational organization of a priori LS lattice-ladder algorithm.

Equation	Computation		
Time initialization			
	$E_m^f(-1) = E_m^b(-1) = \delta > 0$		
	$\beta_m(-1) = 0$	$e_m^b(-1) = 0$	$0 \leq m < M - 1$
	$\beta_m^c(-1) = 0$	$0 \leq m \leq M - 1$	
Order initialization			
(a)	$e_0^f(n) = e_0^b(n) = x(n)$	$e_0(n) = y(n)$	$\alpha_0(n-1) = 1$
Lattice Part: $m = 0, 1, \dots, M-2$			
(b)	$e_{m+1}^f(n) = e_m^f(n) + k_m^{f*}(n-1)e_m^b(n-1)$		
(c)	$e_{m+1}^b(n) = e_m^b(n-1) + k_m^{b*}(n-1)e_m^f(n)$		
(d)	$\beta_m(n) = \lambda\beta_m(n-1) + \alpha_m(n-1)e_m^b(n-1)e_m^{f*}(n)$		
(e)	$E_m^f(n) = \lambda E_m^f(n-1) + \alpha_m(n-1) e_m^f(n) ^2$		
(f)	$E_m^b(n) = \lambda E_m^b(n-1) + \alpha_m(n) e_m^b(n) ^2$		
(g)	$k_m^f(n) = \frac{-\beta_m(n)}{E_m^b(n-1)}$		
(h)	$k_m^b(n) = \frac{-\beta_m^*(n)}{E_m^f(n)}$		
(i)	$\alpha_m(n) = \alpha_{m-1}(n) - \frac{ e_{m-1}^b(n) ^2}{E_{m-1}^b(n)}$		
Ladder part: $m = 1, 2, \dots, M$			
(j)	$\beta_m^c(n) = \lambda\beta_m^c(n-1) + \alpha_m(n)e_m^b(n)e_m^{c*}(n)$		
(k)	$k_m^c(n) = \frac{\beta_m^c(n)}{E_m^b(n)}$		
(l)	$e_{m+1}(n) = e_m(n) - k_m^{c*}(n-1)e_m^b(n)$		

10.7.3 RLS Lattice-Ladder Filters Using Error Feedback Updatings

The LS lattice-ladder algorithms introduced in the previous section update the partial correlations $\beta_m(n)$ and $\beta_m^c(n)$ and the minimum error energies $E_m^f(n)$ and $E_m^b(n)$, and then compute the coefficients of the LS lattice-ladder filter by division. We next develop two algebraically equivalent algorithms, that is, algorithms that solve the same LS problem, which update the lattice-ladder coefficients *directly*. These algorithms, introduced in Ling et al. (1986), have good numerical properties when implemented with finite-word-length arithmetic.

Starting with (10.7.38) and (10.7.45) we have

$$\begin{aligned} k_m^c(n) &= \frac{\beta_m^c(n)}{E_m^b(n)} = \lambda \frac{\beta_m^c(n-1)}{E_m^b(n-1)} \frac{E_m^b(n-1)}{E_m^b(n)} + \frac{\alpha_m(n)e_m^b(n)e_m^{c*}(n)}{E_m^b(n)} \\ &= \frac{1}{E_m^b(n)} [k_m^c(n-1)\lambda E_m^b(n-1) + \alpha_m(n)e_m^b(n)e_m^{c*}(n)] \end{aligned} \quad (10.7.50)$$

or using

$$\lambda E_m^b(n-1) = E_m^b(n) - \alpha_m(n)e_m^b(n)e_m^{b*}(n)$$

we obtain

$$k_m^c(n) = k_m^c(n-1) + \frac{\alpha_m(n)e_m^b(n)}{E_m^b(n)}[e_m^*(n) - k_m^c(n-1)e_m^{b*}(n)]$$

or

$$k_m^c(n) = k_m^c(n-1) + \frac{\alpha_m(n)e_m^b(n)e_{m+1}^*(n)}{E_m^b(n)} \quad (10.7.51)$$

using (10.7.49). Equation (10.7.51) provides a *direct* updating of the ladder parameters. Similar direct updating formulas can be obtained for the lattice coefficients (see Problem 10.39). Using these updatings, we obtain the a priori RLS lattice-ladder algorithm with error feedback shown in Table 10.17.

TABLE 10.17
Computational organization of a priori RLS lattice-ladder algorithm with direct updating of its coefficients using error feedback formula.

Equation	Computation
Time initialization	
	$E_m^f(-1) = E_m^b(-1) = \delta > 0$
	$k_m^f(-1) = k_m^b(-1) = 0$
	$e_m^b(-1) = 0 \quad k_m^c(-1) = 0$
Order initialization	
(a)	$e_0^f(n) = e_0^b(n) = x(n) \quad e_0(n) = y(n) \quad \alpha_0(n) = 1$
Lattice part: $m = 0, 1, \dots, M-2$	
(b)	$e_{m+1}^f(n) = e_m^f(n) + k_m^{f*}(n-1)e_m^b(n-1)$
(c)	$e_{m+1}^b(n) = e_m^b(n-1) + k_m^{b*}(n-1)e_m^f(n)$
(d)	$E_m^f(n) = \lambda E_m^f(n-1) + \alpha_m(n-1) e_m^f(n) ^2$
(e)	$E_m^b(n) = \lambda E_m^b(n-1) + \alpha_m(n) e_m^b(n) ^2$
(f)	$k_m^f(n) = k_m^f(n-1) - \frac{\alpha_m(n-1)e_m^b(n-1)e_{m+1}^{f*}(n)}{E_m^b(n-1)}$
(g)	$k_m^b(n) = k_m^b(n-1) - \frac{\alpha_m(n-1)e_m^f(n)e_{m+1}^{b*}(n)}{E_m^f(n)}$
(h)	$\alpha_{m+1}(n) = \alpha_m(n) - \frac{ \alpha_m(n)e_m^b(n) ^2}{E_m^b(n)}$
Ladder part: $m = 0, 1, \dots, M-1$	
(i)	$e_{m+1}(n) = e_m(n) - k_m^{c*}(n-1)e_m^b(n)$
(j)	$k_m^c(n) = k_m^c(n-1) + \frac{\alpha_m(n)e_m^b(n)e_{m+1}^*(n)}{E_m^b(n)}$

We note that we first use the coefficient $k_m^c(n-1)$ to compute the higher-order error $e_{m+1}(n)$ by (10.7.49) and then use that error to update the coefficient using (10.7.51). This updating has a feedback-like structure that is sometimes referred to as *error feedback form*. An a posteriori form of the RLS lattice-ladder algorithm with error feedback can be easily obtained as shown in Problem 10.40. Simulation studies (Ling et al. 1986) have shown that

10.7.4 Givens Rotation–Based LS Lattice-Ladder Algorithms

We next show how to implement the LS lattice-ladder computations by using the Givens rotation (see Section 8.6) with and without square roots. The resulting algorithms explore the shift invariance of the input data to reduce the computational complexity from $O(M^2)$ to $O(M)$ operations (Ling 1991; Proudler et al. 1989).

We start by introducing the *angle normalized errors*

$$\tilde{e}_m(n) \triangleq \sqrt{e_m(n)\varepsilon_m(n)} = e_m(n)\sqrt{\alpha_m(n)} \quad (10.7.52)$$

$$\tilde{e}_m^f(n) \triangleq \sqrt{e_m^f(n)\varepsilon_m^f(n)} = e_m^f(n)\sqrt{\alpha_m(n-1)} \quad (10.7.53)$$

$$\tilde{e}_m^b(n) \triangleq \sqrt{e_m^b(n)\varepsilon_m^b(n)} = e_m^b(n)\sqrt{\alpha_m(n)} \quad (10.7.54)$$

which are basically the geometric mean of the corresponding a priori and a posteriori errors [see the discussion following (10.5.24) for the interpretation of $\alpha_m(n)$ as an angle variable]. If we formulate the LS problem in terms of these errors, we do not need to distinguish between a priori and a posteriori error algorithms.

Using the a priori lattice equation (10.7.47) for the forward predictor and the definitions of the angle normalized errors, we obtain

$$\tilde{e}_{m+1}^f(n) = \sqrt{\frac{\alpha_{m+1}(n-1)}{\alpha_m(n-1)}} \tilde{e}_m^f(n) - \frac{\beta_m^*(n-1)}{\sqrt{E_m^b(n-2)}} \sqrt{\frac{\alpha_{m+1}(n-1)}{\alpha_m(n-1)}} \frac{\tilde{e}_m^b(n-1)}{\sqrt{E_m^b(n-2)}} \quad (10.7.55)$$

or by using (10.7.30)

$$\tilde{e}_{m+1}^f(n) = \sqrt{\frac{\lambda E_m^b(n-2)}{E_m^b(n-1)}} \tilde{e}_m^f(n) - \frac{\beta_m^*(n-1)}{\sqrt{E_m^b(n-2)}} \sqrt{\lambda} \frac{\tilde{e}_m^b(n-1)}{\sqrt{E_m^b(n-1)}} \quad (10.7.55)$$

If we define the quantities

$$\tilde{c}_m^b(n) \triangleq \sqrt{\frac{\lambda E_m^b(n-1)}{E_m^b(n)}} \quad (10.7.56)$$

$$\tilde{s}_m^b(n) \triangleq \frac{\tilde{e}_m^b(n)}{\sqrt{E_m^b(n)}} \quad (10.7.57)$$

and $\tilde{k}_m^f(n) \triangleq -\frac{\beta_m^*(n)}{\sqrt{E_m^b(n-1)}} = k_m^f(n)\sqrt{E_m^b(n-1)}$ (10.7.58)

we obtain

$$\tilde{e}_{m+1}^f(n) = \tilde{c}_m^b(n-1)\tilde{e}_m^f(n) + \sqrt{\lambda}\tilde{s}_m^b(n-1)\tilde{k}_m^f(n-1) \quad (10.7.59)$$

which provides the order updating of the angle normalized forward prediction error.

To obtain the update equation for the normalized coefficient $\tilde{k}_m^f(n)$, we start with

$$\beta_m(n) = \lambda\beta_m(n-1) + \alpha_m(n-1)e_m^b(n-1)e_m^{f*}(n) \quad (10.7.60)$$

and using (10.7.58), (10.7.53), and (10.7.54), we obtain

$$\tilde{k}_m^f(n) = \sqrt{\lambda}\tilde{k}_m^f(n-1)\sqrt{\frac{\lambda E_m^b(n-2)}{E_m^b(n-1)}} - \frac{\tilde{e}_m^{b*}(n-1)}{\sqrt{E_m^b(n-1)}}\tilde{e}_m^f(n)$$

or finally

$$\tilde{k}_m^f(n) = \sqrt{\lambda} \tilde{c}_m^b(n-1) \tilde{k}_m^f(n-1) - \tilde{s}_m^{b*}(n-1) \tilde{e}_m^f(n) \quad (10.7.61)$$

with the help of (10.7.56) and (10.7.57).

Using the a priori lattice equation (10.7.48) for the backward predictor and the definitions of the angle normalized errors, we obtain

$$\begin{aligned} \tilde{e}_{m+1}^b(n) &= \sqrt{\frac{\alpha_{m+1}(n)}{\alpha_m(n-1)}} \tilde{e}_m^b(n-1) - \frac{\beta_m(n-1)}{\sqrt{E_m^f(n-1)}} \sqrt{\frac{\alpha_{m+1}(n)}{\alpha_m(n-1)}} \frac{\tilde{e}_m^f(n)}{\sqrt{E_m^f(n-1)}} \\ \text{or } \tilde{e}_{m+1}^b(n) &= \sqrt{\frac{\lambda E_m^f(n-1)}{E_m^f(n)}} \tilde{e}_m^b(n-1) - \frac{\beta_m(n-1)}{\sqrt{E_m^f(n-1)}} \sqrt{\lambda} \frac{\tilde{e}_m^f(n)}{\sqrt{E_m^f(n)}} \end{aligned} \quad (10.7.62)$$

by using (10.7.29). If we define the quantities

$$c_m^f(n) \triangleq \frac{\lambda E_m^f(n-1)}{E_m^f(n)} \quad (10.7.63)$$

$$\tilde{s}_m^f(n) \triangleq \frac{\tilde{e}_m^f(n)}{\sqrt{E_m^f(n)}} \quad (10.7.64)$$

$$\text{and } \tilde{k}_m^b(n) \triangleq -\frac{\beta_m^b(n)}{\sqrt{E_m^b(n)}} = k_m^b(n) \sqrt{E_m^f(n)} \quad (10.7.65)$$

we obtain

$$\tilde{e}_{m+1}^b(n) = \tilde{c}_m^f(n) \tilde{e}_m^b(n-1) + \sqrt{\lambda} \tilde{s}_m^f(n) \tilde{k}_m^b(n-1) \quad (10.7.66)$$

which provides the update equation for the angle normalized backward prediction error. The updating of $\tilde{k}_m^b(n)$ is given by

$$\tilde{k}_m^{b*}(n) = \sqrt{\lambda} \tilde{c}_m^f(n) \tilde{k}_m^b(n-1) - \tilde{s}_m^{f*}(n) \tilde{e}_m^b(n-1) \quad (10.7.67)$$

and can be easily obtained, like (10.7.61), by combining (10.7.60) with (10.7.63) through (10.7.65).

Similar updatings can be easily derived for the ladder part of the filter. Indeed, using (10.7.49), the definitions of the angle normalized errors, and (10.7.30), we have

$$\tilde{e}_{m+1}(n) = \sqrt{\frac{\lambda E_m^b(n-1)}{E_m^b(n)}} \tilde{e}_m(n) - \frac{\beta_m^{c*}(n-1)}{\sqrt{E_{m-1}^b(n-1)}} \sqrt{\lambda} \frac{\tilde{e}_m^b(n)}{\sqrt{E_m^b(n)}}$$

$$\text{or } \tilde{e}_{m+1}(n) = \tilde{c}_m^b(n) \tilde{e}_m(n) - \sqrt{\lambda} \tilde{s}_m^b(n) \tilde{k}_m^c(n-1) \quad (10.7.68)$$

$$\text{where } \tilde{k}_m^c(n) \triangleq \frac{\beta_m^c(n)}{\sqrt{E_m^b(n)}} = k_m^c(n) \sqrt{E_m^b(n)} \quad (10.7.69)$$

is a normalized ladder coefficient. This coefficient can be updated by using the recursion

$$\tilde{k}_m^c(n) = \sqrt{\lambda} \tilde{c}_m^b(n) \tilde{k}_m^c(n-1) + \tilde{s}_m^b(n) \tilde{e}_m^*(n) \quad (10.7.70)$$

which can be obtained, like (10.7.61) and (10.7.67), by using (10.7.45) and related definitions.

If we define the normalized energies

$$\tilde{E}_m^f(n) \triangleq \sqrt{E_m^f(n)} \quad (10.7.71)$$

$$\text{and } \tilde{E}_m^b(n) \triangleq \sqrt{E_m^b(n)} \quad (10.7.72)$$

we can easily show, using (10.7.41) and (10.7.42), that

$$\tilde{E}_m^f(n) = \sqrt{\lambda} \tilde{c}_m^f(n) \tilde{E}_m^f(n-1) + \tilde{s}_m^f(n) \tilde{e}_m^{f*}(n) \quad (10.7.73)$$

and

$$\tilde{E}_m^b(n) = \sqrt{\lambda} \tilde{c}_m^b(n) \tilde{E}_m^b(n-1) + \tilde{s}_m^b(n) \tilde{e}_m^{b*}(n) \quad (10.7.74)$$

which provide time updates for the normalized minimum energies. However, the following recursions

$$\tilde{E}_m^f(n) = \{\lambda[\tilde{E}_m^f(n-1)]^2 + |\tilde{e}_m^f(n)|^2\}^{1/2} \quad (10.7.75)$$

$$\tilde{E}_m^b(n) = \{\lambda[\tilde{E}_m^b(n-1)]^2 + |\tilde{e}_m^b(n)|^2\}^{1/2} \quad (10.7.76)$$

obtained from (10.7.41) and (10.7.42), provide more convenient updatings.

We now have a complete formulation of the LS lattice-ladder recursions using angle normalized errors. To see the meaning and significance of these recursions, we express them in matrix form as

$$\begin{bmatrix} \tilde{e}_{m+1}^f(n) \\ \tilde{k}_m^f(n) \end{bmatrix} = \begin{bmatrix} \tilde{c}_m^b(n-1) & \tilde{s}_m^b(n-1) \\ -\tilde{s}_m^{b*}(n-1) & \tilde{c}_m^b(n-1) \end{bmatrix} \begin{bmatrix} \tilde{e}_m^f(n) \\ \sqrt{\lambda} \tilde{k}_m^f(n-1) \end{bmatrix} \quad (10.7.77)$$

$$\begin{bmatrix} \tilde{e}_{m+1}^b(n) \\ \tilde{k}_m^{b*}(n) \end{bmatrix} = \begin{bmatrix} \tilde{c}_m^f(n) & \tilde{s}_m^f(n) \\ -\tilde{s}_m^{f*}(n) & \tilde{c}_m^f(n) \end{bmatrix} \begin{bmatrix} \tilde{e}_m^b(n-1) \\ \sqrt{\lambda} \tilde{k}_m^{b*}(n-1) \end{bmatrix} \quad (10.7.78)$$

$$\begin{bmatrix} \tilde{e}_{m+1}(n) \\ \tilde{k}_m^{c*}(n) \end{bmatrix} = \begin{bmatrix} \tilde{c}_m^b(n) & -\tilde{s}_m^b(n) \\ \tilde{s}_m^{b*}(n) & \tilde{c}_m^b(n) \end{bmatrix} \begin{bmatrix} \tilde{e}_m(n) \\ \sqrt{\lambda} \tilde{k}_m^{c*}(n-1) \end{bmatrix} \quad (10.7.79)$$

where we see that the updating of the forward predictor parameters and the ladder parameters involves the same matrix delayed by one sample. The different position of the minus sign, due to the different sign used in the definitions of $\tilde{k}_m^f(n)$ and $\tilde{k}_m^c(n)$, is immaterial. Furthermore, it is straightforward to show that

$$|\tilde{c}_m^f(n)|^2 + |\tilde{s}_m^f(n)|^2 = 1 \quad (10.7.80)$$

and

$$|\tilde{c}_m^b(n)|^2 + |\tilde{s}_m^b(n)|^2 = 1 \quad (10.7.81)$$

which imply that the matrices in (10.7.77) through (10.7.79) are the Givens rotation matrices. Therefore, we have obtained a formulation of the LS lattice-ladder algorithm that updates the angle normalized errors and a set of normalized lattice-ladder coefficients using the Givens rotations. Using (10.7.76) and definitions of $\tilde{c}_m^b(n)$ and $\tilde{s}_m^b(n)$, we can show that

$$\begin{bmatrix} \tilde{E}_m^b(n) \\ 0 \end{bmatrix} = \begin{bmatrix} \tilde{c}_m^b(n) & \tilde{s}_m^{b*}(n) \\ -\tilde{s}_m^b(n) & \tilde{c}_m^b(n) \end{bmatrix} \begin{bmatrix} \sqrt{\lambda} \tilde{E}_m^b(n-1) \\ \tilde{e}_m^b(n) \end{bmatrix} \quad (10.7.82)$$

which shows that we can use the BLP Givens rotation to update the normalized energy $\tilde{E}_m^b(n)$. A similar transformation can be obtained for $\tilde{E}_m^f(n)$. However, the energy updatings are usually performed using (10.7.75) and (10.7.76).

The square root-free version of the Givens LS lattice-ladder filter is basically a simple modification of the error feedback form of the a priori LS lattice-ladder algorithm. Indeed, using (10.7.50), we have

$$k_m^c(n) = \frac{\lambda E_m^b(n-1)}{E_m^b(n)} k_m^c(n-1) + \frac{\alpha_m(n) e_m^b(n)}{E_m^b(n)} e_m^*(n)$$

or if we define the quantities

$$c_m^b(n) \triangleq \frac{\lambda E_m^b(n-1)}{E_m^b(n)} = |\tilde{c}_m^b(n)|^2 \quad (10.7.83)$$

and

$$s_m^b(n) \triangleq \frac{\alpha_m(n) e_m^b(n)}{E_m^b(n)} \quad (10.7.84)$$

we obtain

$$k_m^c(n) = c_m^b(n)k_m^c(n-1) + s_m^b(n)e_m^*(n) \quad (10.7.85)$$

which provides the required updating for the ladder parameters.

Similarly, using the error feedback a priori updatings for the lattice parameters, we obtain the recursions

$$k_m^f(n) = c_m^b(n-1)k_m^f(n-1) - s_m^b(n-1)e_m^{f*}(n) \quad (10.7.86)$$

and

$$k_m^b(n) = c_m^f(n)k_m^b(n-1) - s_m^f(n)e_m^{b*}(n-1) \quad (10.7.87)$$

where

$$c_m^f(n) \triangleq \frac{\lambda E_m^f(n-1)}{E_m^f(n)} = |\tilde{c}_m^f(n)|^2 \quad (10.7.88)$$

and

$$s_m^f(n) \triangleq \frac{\alpha_m(n-1)e_m^f(n)}{E_m^f(n)} \quad (10.7.89)$$

are the forward rotation parameters. These recursions constitute the basis for the square root-free Givens LS lattice-ladder algorithm.

Table 10.18 provides the complete computational organizations of the Givens LS lattice-ladder algorithms with and without square roots. The square root algorithm is ini-

TABLE 10.18
Summary of the Givens LS lattice-ladder adaptive filter algorithms.

Equation	Square root form	Square root-free form
Forward rotation parameters		
(a)	$\tilde{E}_m^f(n) = \{\lambda[\tilde{E}_m^f(n-1)]^2 + \tilde{e}_m^f(n) ^2\}^{1/2}$	$E_m^f(n) = \lambda E_m^f(n-1) + \alpha_m(n-1) e_m^f(n) ^2$
(b)	$\tilde{c}_m^f(n) = \frac{\sqrt{\lambda}\tilde{E}_m^f(n-1)}{\tilde{E}_m^f(n)}$	$c_m^f(n) = \frac{\lambda E_m^f(n-1)}{E_m^f(n)}$
(c)	$\tilde{s}_m^f(n) = \frac{\tilde{e}_m^f(n)}{\tilde{E}_m^f(n)}$	$s_m^f(n) = \frac{\alpha_m(n-1)e_m^f(n)}{E_m^f(n)}$
Backward Rotation Parameters		
(d)	$\tilde{E}_m^b(n) = \{\lambda[\tilde{E}_m^b(n-1)]^2 + \tilde{e}_m^b(n) ^2\}^{1/2}$	$E_m^b(n) = \lambda E_m^b(n-1) + \alpha_m(n) e_m^b(n) ^2$
(e)	$\tilde{c}_m^b(n) = \frac{\sqrt{\lambda}\tilde{E}_m^b(n-1)}{\tilde{E}_m^b(n)}$	$c_m^b(n) = \frac{\lambda E_m^b(n-1)}{E_m^b(n)}$
(f)	$\tilde{s}_m^b(n) = \frac{\tilde{e}_m^b(n)}{\tilde{E}_m^b(n)}$	$s_m^b(n) = \frac{\alpha_m(n)e_m^b(n)}{E_m^b(n)}$
Forward predictor rotator		
(g)	$\tilde{e}_{m+1}^f(n) = \tilde{c}_m^b(n-1)\tilde{e}_m^f(n) + \sqrt{\lambda}\tilde{s}_m^b(n-1)\tilde{k}_m^f(n-1)$	$e_{m+1}^f(n) = e_m^f(n) + k_m^{f*}(n-1)e_m^b(n-1)$
(h)	$\tilde{k}_m^f(n) = \sqrt{\lambda}\tilde{c}_m^b(n-1)\tilde{k}_m^f(n-1) - \tilde{s}_m^{b*}(n-1)\tilde{e}_m^f(n)$	$k_m^f(n) = c_m^b(n-1)k_m^f(n-1) - s_m^b(n-1)e_m^{f*}(n)$
Backward predictor rotator		
(i)	$\tilde{e}_{m+1}^b(n) = \tilde{c}_m^f(n)\tilde{e}_m^b(n-1) + \sqrt{\lambda}\tilde{s}_m^f(n)\tilde{k}_m^{b*}(n-1)$	$e_{m+1}^b(n) = e_m^b(n-1) + k_m^{b*}(n-1)e_m^f(n)$
(j)	$\tilde{k}_m^b(n) = \sqrt{\lambda}\tilde{c}_m^f(n)\tilde{k}_m^{b*}(n-1) - \tilde{s}_m^f(n)\tilde{e}_m^b(n-1)$	$k_m^b(n) = c_m^f(n)k_m^b(n-1) - s_m^f(n)e_m^{b*}(n-1)$
Filter rotator		
(k)	$\tilde{e}_{m+1}(n) = \tilde{c}_m^b(n)\tilde{e}_m(n) - \sqrt{\lambda}\tilde{s}_m^b(n)\tilde{k}_m^{c*}(n-1)$	$e_{m+1}(n) = e_m(n) - k_m^{c*}(n-1)e_m^b(n)$
(l)	$\tilde{k}_m^c(n) = \sqrt{\lambda}\tilde{c}_m^b(n)\tilde{k}_m^c(n-1) + \tilde{s}_m^b(n)\tilde{e}_m^*(n)$	$k_m^c(n) = c_m^b(n)k_m^c(n-1) + s_m^b(n)e_m^*(n)$

tialized as usual with $E_m^f(-1) = E_m^b(-1) = \delta > 0$, $\tilde{e}_0^f(n) = \tilde{e}_0^b(n) = x(n)$, $\tilde{e}_0(n) = y(n)$, $\alpha_0(n) = 1$, and all other variables set to zero. The square root-free algorithm is initialized as the a priori algorithm with error feedback. Figure 10.39 shows a single stage of the LS lattice-ladder filter based on Givens rotations with square roots.

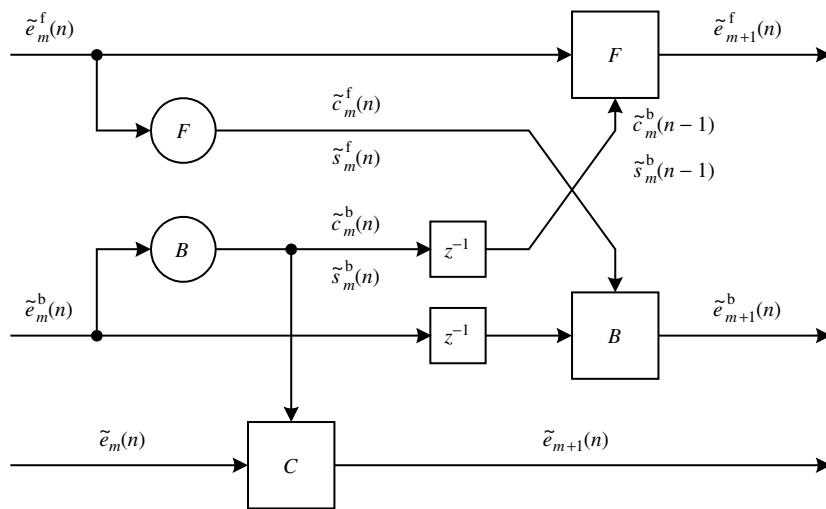


FIGURE 10.39

Block diagram representation of the Givens RLS lattice-ladder stage. Circles denote computing elements that calculate the rotation parameters and squares denote computing elements that perform the rotations.

10.7.5 Classification of RLS Algorithms for FIR Filtering

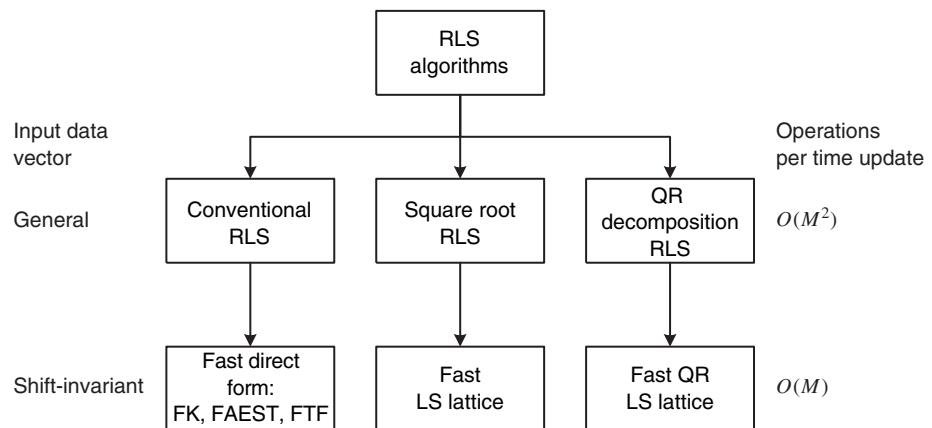
Every exact RLS algorithm discussed in this section consists of two parts: a part that computes the LS forward and backward predictors of the input signal and a part that uses information from the linear prediction part to compute the LS filter. In all cases, information flows from the prediction part to the filtering part, but not vice versa. Therefore, all critical numerical operations take place in the linear prediction section.

For direct-form structures, the prediction problems facilitate the fast computation of the RLS gain vectors.

In the case of lattice-ladder structures, the lattice part (which again solves the linear prediction problem) decorrelates (or orthogonalizes in the LS sense) the input signal vector and creates an orthogonal base consisting of the backward prediction errors $\{e_m^b(n)\}_0^{M-1}$. This orthogonal basis is used by the ladder part to form the LS filtering error. Essentially, the LS lattice part facilitates the triangular UDL decomposition of the inverse correlation matrix $\hat{\mathbf{R}}^{-1}(n)$ or the Gram-Schmidt orthogonalization of the columns of data matrix $\mathbf{X}(n)$. This property makes the RLS lattice-ladder algorithm order-recursive, like its minimum MSE counterpart (see Section 7.3).

The QRD-RLS lattice-ladder algorithms also consist of a lattice part that solves the linear prediction problem and a ladder part that uses information from the lattice to form the LS filtering estimate. The LS lattice produces the triangularization of the inverse correlation matrix $\hat{\mathbf{R}}^{-1}(n)$ whereas the QRD LS lattice produces the upper triangular Cholesky factor of $\hat{\mathbf{R}}(n)$ by applying an orthogonal transformation to data matrix $\mathbf{X}(n)$.

The correspondence of these algorithms to their counterparts for RLS array processing, discussed in Section 10.6, is summarized in Figure 10.40.

**FIGURE 10.40**

Classification of RLS algorithms for array processing and FIR filtering.

It is interesting to note that the RLS lattice-ladder algorithms with error feedback are identical in form to the square root-free Givens rotation-based QRD-RLS lattice-ladder algorithms. This similarity explains the excellent numerical properties of both structures.

The RLS lattice-ladder algorithms (both UDL^H -decomposition based and QR-decomposition based) share the following highly desirable characteristics:

- *Good numerical properties* that originate from the square root decomposition (Cholesky or QR) part of the algorithms.
- *Good convergence properties*, which are inherited from the exact LS minimization performed by all algorithms.
- *Modularity and regularity* that make possible their VLSI and multiprocessing implementation.

It has been shown (Ljung and Ljung 1985) that all RLS lattice-ladder algorithms are numerically stable for $\lambda < 1$. However, they differ in terms of numerical accuracy. It turns out that the lattice-ladder algorithms with error feedback (which are basically equivalent to the square root-free QRD lattice ladder) and the QRD lattice-ladder algorithms have the best numerical accuracy.

10.8 TRACKING PERFORMANCE OF ADAPTIVE ALGORITHMS

Tracking of a time-varying system is an important problem in many areas of application. Consider, for example, a digital communications system in which the channel characteristics may change with time for various reasons. If we want to incorporate an echo canceler in such a system, then clearly the echo canceler must monitor the changing impulse response of the echo path so that it can generate an accurate replica of the echo. This will require the adaptive algorithm of an echo canceler to possess an acceptable tracking capability. Similar situations arise in adaptive equalization, adaptive prediction, adaptive noise canceling, and so on. In all these applications, adaptive filters are forced to operate in a *nonstationary* SOE. In this section, we examine the ability and performance of the LMS and RLS algorithms to track the ever-changing minimum point of the error surface.

As discussed earlier, the tracking mode is a steady-state operation of the adaptive algorithm, and it follows the acquisition mode, which is a transient phenomenon. Therefore, the algorithm must acquire the system parameters before tracking can commence. This has two implications. First, the rate of convergence is generally not related to the tracking

behavior, and as such, we analyze the tracking behavior when the number of iterations (or steps) is relatively large. Second, the time variation of the parameter change should be small enough compared to the rate of convergence that the algorithm can perform adequate tracking; otherwise, it is constantly acquiring the parameters.

10.8.1 Approaches for Nonstationary SOE

To effectively track a nonstationary SOE, adaptive algorithms should use only *local* statistics. There are three practical ways in which this can be achieved.

Exponentially growing window

In this approach, the current data are artificially emphasized by exponentially weighting past data values, as shown in Figure 10.41(a). The error function that is minimized is given by

$$E(n) = \sum_{j=0}^n \lambda^{n-j} |y(j) - \mathbf{c}^H \mathbf{x}(j)|^2 = \lambda E(n-1) + |y(n) - \mathbf{c}^H \mathbf{x}(n)|^2 \quad (10.8.1)$$

where $0 < \lambda < 1$. Clearly, this is the cost function we used in the development of the RLS algorithm, given in Table 10.6, in which λ is termed the forgetting factor. The effective window length is given by

$$L_{\text{eff}} \triangleq \frac{\sum_{n=0}^{\infty} \lambda^n}{\lambda^0} = \frac{1}{1-\lambda} \quad (10.8.2)$$

Hence for good tracking performance λ should be in the range $0.9 \leq \lambda < 1$. Note that $\lambda = 1$ results in a *rectangularly growing window* that uses global statistics and hence will not be able to track parameter changes. Thus the RLS algorithm with exponential forgetting is capable of using the local information needed to adapt in a nonstationary SOE.

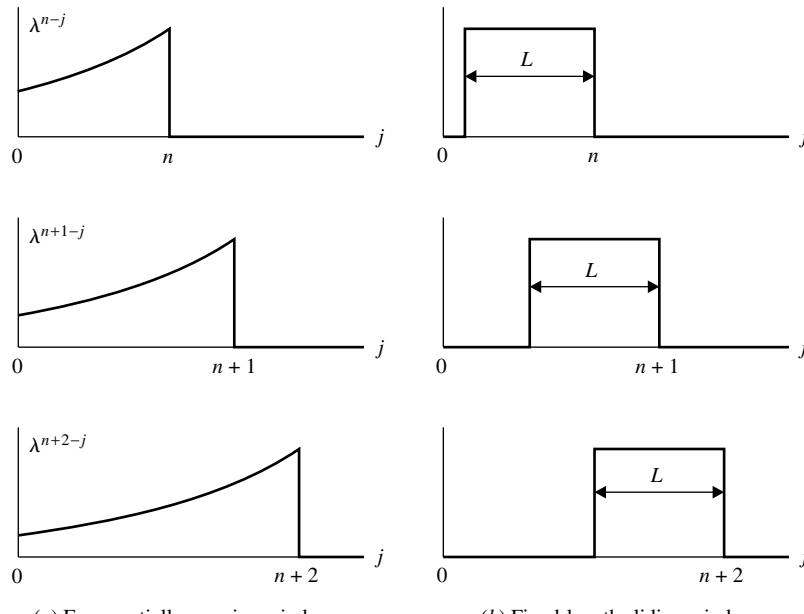


FIGURE 10.41

Illustration of exponentially growing and fixed-length sliding windows.

Fixed-length sliding window

The basic feature of this approach is that the parameter estimates are based only on a finite number of past data values, as shown in Figure 10.41(b). Let us consider a rectangular window of fixed length $L > M$. Then the cost function that is minimized is given by

$$E(n, L) \triangleq \sum_{j=n-L+1}^n |y(j) - \mathbf{c}^H \mathbf{x}(j)|^2 \quad (10.8.3)$$

When a new data value at $n+1$ is added to the sum in (10.8.3), the old data value is discarded, that is, all old data values beyond $n - L + 1$ are discarded. Thus the active number of data values is always a constant equal to L , which makes this as a constant-memory adaptive algorithm. By following the steps given for the RLS adaptive filter in Section 10.5, it is possible to derive a recursive algorithm to determine the filter $\mathbf{c}(n)$ that minimizes the error function in (10.8.3).

Let $\mathbf{c}_{\{n-L\}}(n-1)$ denote the estimate of $\mathbf{c}(n-1)$ based on L data values between $n - L$ and $n - 1$. After the new data value at n is observed, the RLS algorithm in Table 10.6 is applicable with $\lambda = 1$ and with obvious extension of notation. Hence we obtain the algorithm

$$\mathbf{c}_{\{n-L\}}(n) = \mathbf{c}_{\{n-L\}}(n-1) + \mathbf{g}_{\{n-L\}}(n)e^*(n) \quad (10.8.4)$$

$$e(n) = y(n) - \mathbf{c}_{\{n-L\}}^H(n-1)\mathbf{x}(n) \quad (10.8.5)$$

$$\mathbf{g}_{\{n-L\}}(n) = \frac{\bar{\mathbf{g}}_{\{n-L\}}(n)}{\alpha_{\{n-L\}}(n)} \quad (10.8.6)$$

$$\bar{\mathbf{g}}_{\{n-L\}}(n) = \mathbf{P}_{\{n-L\}}(n-1)\mathbf{x}(n) \quad (10.8.7)$$

$$\alpha_{\{n-L\}}(n) = 1 + \bar{\mathbf{g}}_{\{n-L\}}^H(n)\mathbf{x}(n) \quad (10.8.8)$$

$$\mathbf{P}_{\{n-L\}}(n) = \mathbf{P}_{\{n-L\}}(n-1) - \mathbf{g}_{\{n-L\}}(n)\bar{\mathbf{g}}_{\{n-L\}}^H(n) \quad (10.8.9)$$

The above algorithm is based on $L + 1$ data values. To maintain the data window at fixed length L , we have to discard the observation at $n - L$. By using the matrix inversion lemma given in Appendix A, it can be shown that (see Problem 10.51)

$$\mathbf{c}_{\{n-L+1\}}(n) = \mathbf{c}_{\{n-L\}}(n) - \mathbf{g}_{\{n-L+1\}}(n)e^*(n - L) \quad (10.8.10)$$

$$e(n - L) = y(n - L) - \mathbf{c}_{\{n-L\}}^H(n)\mathbf{x}(n - L) \quad (10.8.11)$$

$$\mathbf{g}_{\{n-L+1\}}(n) = \frac{\bar{\mathbf{g}}_{\{n-L+1\}}(n)}{\alpha_{\{n-L+1\}}(n)} \quad (10.8.12)$$

$$\bar{\mathbf{g}}_{\{n-L+1\}}(n) = \mathbf{P}_{\{n-L\}}(n)\mathbf{x}(n - L) \quad (10.8.13)$$

$$\alpha_{\{n-L+1\}}(n) = 1 - \bar{\mathbf{g}}_{\{n-L+1\}}^H(n)\mathbf{x}(n - L) \quad (10.8.14)$$

$$\mathbf{P}_{\{n-L+1\}}(n) = \mathbf{P}_{\{n-L\}}(n) + \mathbf{g}_{\{n-L+1\}}(n)\bar{\mathbf{g}}_{\{n-L+1\}}^H(n) \quad (10.8.15)$$

The overall algorithm for the fixed-memory rectangular window adaptive algorithm is given by (10.8.4) through (10.8.15), which recursively update $\mathbf{c}_{\{n-L\}}(n-1)$ to $\mathbf{c}_{\{n-L+1\}}(n)$. Thus, this algorithm can adapt to the nonstationary SOE using the local information. The fixed-length sliding-window RLS algorithm can be implemented by using a combination of two prewindowed RLS algorithms (Manolakis et al. 1987).

Evolutionary model—Kalman filter

In the first two approaches, adaptation in the nonstationarity SOE was obtained through the local information, either by discarding old data or by deemphasizing it. In the third approach, we assume that we have a statistical model that describes the nonstationarity

of the SOE. This model is in the form of a stochastic difference equation together with appropriate statistical properties. This leads to the well-known Kalman filter formulation in which we assume that the parameter variations are modeled by

$$\mathbf{c}(n) = \boldsymbol{\Xi}(n)\mathbf{c}(n-1) + \mathbf{v}(n) \quad (10.8.16)$$

where $\mathbf{v}(n)$ is a random vector with zero mean and correlation matrix $\boldsymbol{\Sigma}(n)$, and $\boldsymbol{\Xi}(n)$ is the state-transition matrix known for all n . The desired signal $y(n)$ is modeled as

$$y(n) = \mathbf{c}^H(n)\mathbf{x}(n) + \varepsilon(n) \quad (10.8.17)$$

where $\varepsilon(n)$ is the a posteriori estimation error assumed to be zero-mean with variance σ_ε^2 . Thus in this formulation, the parameter vector $\mathbf{c}(n)$ acts as the state of a system while the input data vector $\mathbf{x}(n)$ acts as the time-varying output vector. Now the best linear unbiased estimate $\hat{\mathbf{c}}(n)$ of $\mathbf{c}(n)$ based on past observations $\{y(i)\}_{i=0}^n$ can be obtained by using the Kalman filter equations (Section 7.8). These recursive equations are given by

$$\hat{\mathbf{c}}(n) = \boldsymbol{\Xi}(n)\hat{\mathbf{c}}(n-1) + \mathbf{g}(n)[y(n) - \hat{\mathbf{c}}^H(n-1)\boldsymbol{\Xi}^H(n)\mathbf{x}(n)] \quad (10.8.18)$$

$$\mathbf{g}(n) = \frac{\boldsymbol{\Xi}(n)\mathbf{P}(n-1)\mathbf{x}(n)}{\sigma_\varepsilon^2 + \mathbf{x}^H(n)\mathbf{P}(n-1)\mathbf{x}(n)} \quad (10.8.19)$$

$$\mathbf{P}(n) = \boldsymbol{\Xi}(n)\mathbf{P}(n-1)\boldsymbol{\Xi}^H(n) + \boldsymbol{\Sigma}(n) \quad (10.8.20)$$

$$-\boldsymbol{\Xi}(n)\mathbf{P}(n-1)\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{\sigma_\varepsilon^2 + \mathbf{x}^H(n)\mathbf{P}(n-1)\mathbf{x}(n)}\mathbf{P}(n-1)\boldsymbol{\Xi}^H(n)$$

where $\mathbf{g}(n)$ is the Kalman gain matrix and $\mathbf{P}(n)$ is the error covariance matrix. This approach implies that if the time-varying parameters are modeled as state equations, then the Kalman filter rather than the adaptive filter is a proper solution.

Furthermore, it can be shown that the Kalman filter has a close similarity to the RLS adaptive filters if we make the following appropriate substitutions:

Exponential memory: If we substitute

$$\boldsymbol{\Xi}(n) = \mathbf{I} \quad \sigma_\varepsilon^2 = \lambda \quad \boldsymbol{\Sigma}(n) = \frac{1-\lambda}{\lambda}[\mathbf{I} - \mathbf{g}(n)\mathbf{x}^H(n)]\mathbf{P}(n-1) \quad (10.8.21)$$

then we obtain the exponential memory RLS algorithm given in Table 10.6.

Rectangularly growing memory: If we substitute

$$\boldsymbol{\Xi}(n) = \mathbf{I} \quad \sigma_\varepsilon^2 = 1 \quad \boldsymbol{\Sigma}(n) = 0 \quad (10.8.22)$$

then we obtain the rectangularly growing memory RLS algorithm.

10.8.2 Preliminaries in Performance Analysis

In Sections 10.4 and 10.5.4, we developed and analyzed the LMS and RLS algorithms in stationary environments, respectively. However, these algorithms are generally used in applications (e.g., modems) that are intended to operate continuously in SOE whose characteristics change with time. Therefore, we need to discuss the performance of these two widely used algorithms in such situations. Although we provided various adaptive filtering approaches for time-varying environments above, we now discuss, in the remainder of this section, the ability of these two algorithms to track time-varying parameters. We provide both analytical results, assuming a model of parameter variation, and experimental results, using simulations.

A popular approach for this analytical assessment is to assume a first-order AR model with finite variance [that is we set $\boldsymbol{\Xi}(n) = \rho\mathbf{I}$ in (10.8.16)]. Although higher-order models are also possible, only a few results on the tracking performance using these models are currently available. It is ironic that most analytical results on the tracking performance have

been obtained for the random-walk model (a special case of the first-order AR model), which is unrealistic because of the infinite variance. A tutorial review of the latest results for the general case and additional references are available in Macchi (1996).

In our analysis of tracking characteristics of the LMS and RLS algorithms, we use the first-order AR model and discuss its effect on the tracking performance. The closed-form results will be given using the random-walk model and confirmed using simulated experiments.

Analysis setup

In the tracking analysis, it is desirable to use the *a priori* adaptive filter. Hence we assume that the desired response is generated by the following filter model[†]

$$y(n) = \mathbf{c}_o^H(n-1)\mathbf{x}(n) + v(n) \quad (10.8.23)$$

where $v(n)$ is assumed to be WGN($0, \sigma_v^2$) with $\sigma_v^2 < \infty$. The random processes $\mathbf{x}(n)$ and $v(n)$ are assumed to be independent and stationary. The variation of $\mathbf{c}_o(n)$ is modeled by the first-order AR (or Markov) process

$$\mathbf{c}_o(n) = \rho\mathbf{c}_o(n-1) + \psi(n) \quad (10.8.24)$$

with $0 < \rho < 1$ and creates the nonstationarity of the SOE. The quantity $\psi(n)$ is the uncertainty in the model and assumed to be independent of $\mathbf{x}(n)$ and $v(n)$, with mean $E\{\psi(n)\} = \mathbf{0}$ and correlation $E\{\psi(n)\psi^H(n)\} = \mathbf{R}_\psi$. Tracking is generally achievable if ρ is close to 1. The random-walk model is obtained by using $\rho = 1$ in (10.8.24).

Conjugate transposing and premultiplying both sides of (10.8.23) by $\mathbf{x}(n)$, taking the expectation, and using independence between $\mathbf{x}(n)$ and $v(n)$, we obtain

$$\mathbf{R}\mathbf{c}_o(n-1) = \mathbf{d}(n) \quad (10.8.25)$$

Hence, $\mathbf{c}_o(n-1)$ is the optimum a priori filter and

$$e_o(n) = y(n) - \mathbf{c}_o^H(n-1)\mathbf{x}(n) = v(n) \quad (10.8.26)$$

is the *optimum a priori error*. If $\mathbf{R}_\psi = \mathbf{0}$ and $\rho = 1$, we have $\mathbf{c}_o(n) = \mathbf{c}_o$ for all n , and therefore $y(n)$ is wide-sense stationary (WSS). In this case, we have a stationary environment, and the goal of the adaptive filter is to find the optimum filter \mathbf{c}_o . For $\mathbf{R}_\psi \neq \mathbf{0}$, the adaptive filter should find and track the optimum a priori filter $\mathbf{c}_o(n)$. This setup, which is widely used to analyze the properties of adaptive algorithms, is illustrated in Figure 10.42.

Assumptions

To analyze the tracking performance of adaptive algorithms, we use the assumptions discussed elsewhere and repeated below for convenience.

- A1** The sequence of input data vectors $\mathbf{x}(n)$ is WGN($\mathbf{0}, \mathbf{R}$).
- A2** The desired response $y(n)$ can be modeled as

$$y(n) = \mathbf{c}_o^H(n-1)\mathbf{x}(n) + e_o(n) \quad (10.8.27)$$

where $e_o(n)$ is WGN($0, \sigma_e^2$).

- A3** The time variation of $\mathbf{c}_o(n)$ is described by

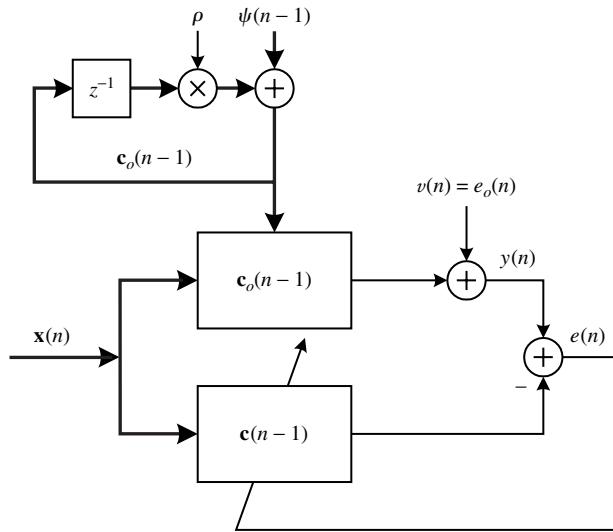
$$\mathbf{c}_o(n) = \rho\mathbf{c}_o(n-1) + \psi(n) \quad (10.8.28)$$

where $0 \leq \rho \leq 1$ and $\psi(n)$ is WGN($\mathbf{0}, \mathbf{R}_\psi$).

- A4** The random sequences $\mathbf{x}(n)$, $e_o(n)$, and $\psi(n)$ are mutually independent.

Through these assumptions, we want to stress that the nonstationarity of the SOE is created solely by $\mathbf{c}_o(n)$ and not by $\mathbf{x}(n)$, which is WSS.

[†]We use this model to make a fair comparison between the adaptive and the optimum filter.

**FIGURE 10.42**

Block diagram of the setup and model used for the analysis of adaptive algorithms.

Although we provide analysis for (10.8.27), many results are given for the random walk model ($\rho = 1$). The case $0 < \rho < 1$, which is straightforward but complicated, is discussed in Solo and Kong (1995). Before we delve into this analysis, we discuss criteria that are used for evaluating the tracking performance.

Degree of nonstationarity

To determine whether an adaptive algorithm can adequately track the changing SOE, one needs to define the speed of variation of the statistics of the adaptive filter environment. This speed is quantified in terms of the *degree of nonstationarity* (DNS), introduced in Macchi (1995, 1996), and is defined by

$$\eta(n) \triangleq \sqrt{\frac{E\{|y_{o,\text{incr}}(n)|^2\}}{P_o(n)}} \quad (10.8.29)$$

where

$$y_{o,\text{incr}}(n) = [\mathbf{c}_o(n) - \mathbf{c}_o(n-1)]^H \mathbf{x}(n) \quad (10.8.30)$$

is the output of the *incremental* filter. The numerator is the power introduced by the variation of the optimum filter, and the denominator is the MMSE, which in the context of (10.8.26) is equal to the power of the output noise. Assuming $\rho = 1$ in (10.8.28), we see that (10.8.30) is given by

$$y_{o,\text{incr}}(n) = \Psi^H \mathbf{x}(n)$$

and hence the numerator in (10.8.29) is given by

$$\begin{aligned} E\{|y_{o,\text{incr}}(n)|^2\} &= E\{\Psi^H \mathbf{x}(n) \mathbf{x}^H(n) \Psi\} = \text{tr}[E\{\Psi^H \mathbf{x}(n) \mathbf{x}^H(n) \Psi\}] \\ &= \text{tr}[E\{\Psi \Psi^H \mathbf{x}(n) \mathbf{x}^H\}] = \text{tr}[E\{\Psi \Psi^H\} E\{\mathbf{x}(n) \mathbf{x}^H\}] \\ &= \text{tr}[\mathbf{R}_\Psi \mathbf{R}] = \text{tr}[\mathbf{R} \mathbf{R}_\Psi] \end{aligned} \quad (10.8.31)$$

where we have used the independence assumption A4. Substituting (10.8.31) in (10.8.29), we obtain

$$\eta(n) \triangleq \sqrt{\frac{\text{tr}[\mathbf{R} \mathbf{R}_\Psi]}{P_o(n)}} \quad (10.8.32)$$

Smaller values of η ($\ll 1$) imply that the adaptive algorithm can track time variations of the nonstationary SOE. On the contrary, if $\eta > 1$, then the statistical variations of the

SOE are too fast for the adaptive algorithm to keep up with the SOE and lead to massive misadjustment errors. In such situations, an adaptive filter should not be used.

Mean square deviation (MSD)

We defined the MSD $\mathcal{D}(n)$ in (10.2.29) as a performance measure for adaptive filters in the steady-state environment. It is also used for measuring the tracking performance. Consider the coefficient error vector $\tilde{\mathbf{c}}(n)$, which can be written as

$$\begin{aligned}\tilde{\mathbf{c}}(n) &= \mathbf{c}(n) - \mathbf{c}_o(n) \\ &= [\mathbf{c}(n) - E\{\mathbf{c}(n)\}] + [E\{\mathbf{c}(n)\} - \mathbf{c}_o(n)]\end{aligned}\quad (10.8.33)$$

$$\triangleq \tilde{\mathbf{c}}_1(n) + \tilde{\mathbf{c}}_2(n) \quad (10.8.34)$$

where $\tilde{\mathbf{c}}_1(n)$ is the fluctuation of the adaptive filter parameter vector about its mean (estimation error) and $\tilde{\mathbf{c}}_2(n)$ is the bias of $\mathbf{c}(n)$ with respect to the true vector $\mathbf{c}_o(n)$ (systematic or lag error). Using the independence assumption of the previous section that $\mathbf{x}(n)$ and $\mathbf{c}(n-1)$ are statistically independent, we can show that (Macchi 1996)

$$E\{\tilde{\mathbf{c}}_1^H(n)\tilde{\mathbf{c}}_2(n)\} = 0 \quad (10.8.35)$$

which by using (10.2.29) and (10.8.34) leads to

$$\mathcal{D}(n) = \mathcal{D}_1(n) + \mathcal{D}_2(n) \quad (10.8.36)$$

The first MSD term is due to the parameter estimation error and is called the *estimation variance*. The second MSD term is due to the parameter lag error and is termed *lag variance*, and its presence indicates the nonstationary environment.

Misadjustment and lowest excess MSE

The second performance measure, defined in (10.2.38), is the (a priori) misadjustment $\mathcal{M}(n)$, which is the ratio of the excess MSE $P_{\text{ex}}(n)$ to the MMSE $P_o(n)$. The a priori excess MSE is given by

$$P_{\text{ex}}(n) = E\{|\tilde{\mathbf{c}}^H(n-1)\mathbf{x}(n)|^2\} = E\{|\tilde{\mathbf{c}}_1^H(n-1)\mathbf{x}(n) + \tilde{\mathbf{c}}_2^H(n-1)\mathbf{x}(n)|^2\} \quad (10.8.37)$$

which under the independence assumption and (10.8.35) can be written as

$$P_{\text{ex}}(n) = P_{\text{ex},1}(n) + P_{\text{ex},2}(n) \quad (10.8.38)$$

where the first term, $P_{\text{ex},1}(n)$, is excess MSE due to estimation error and is termed the *estimation noise* while the second term, $P_{\text{ex},2}(n)$, is the excess MSE due to lag error and is called the *lag noise*. Therefore, we can also write the misadjustment $\mathcal{M}(n)$ as

$$\mathcal{M}(n) = \mathcal{M}_1(n) + \mathcal{M}_2(n) \quad (10.8.39)$$

where $\mathcal{M}_1(n)$ is the *estimation misadjustment* and $\mathcal{M}_2(n)$ is the *lag misadjustment*.

In the context of the first-order Markov model, the best performance obtained by any a priori adaptive filter occurs if $\mathbf{c}(n) = \rho\mathbf{c}_o(n-1)$. This observation makes possible the computation of a lower bound for the excess MSE of any a priori adaptive algorithm. From (10.8.34) and (10.8.24), we have

$$\begin{aligned}\tilde{\mathbf{c}}(n) &= \mathbf{c}(n) - \mathbf{c}_o(n) = [\mathbf{c}(n) - \rho\mathbf{c}_o(n-1)] - \boldsymbol{\psi}(n) \\ &\triangleq \hat{\mathbf{c}}(n) - \boldsymbol{\psi}(n)\end{aligned}\quad (10.8.40)$$

and hence

$$\begin{aligned}P_{\text{ex}}(n) &= E\{|\tilde{\mathbf{c}}^H(n-1)\mathbf{x}(n)|^2\} \\ &= E\{|\hat{\mathbf{c}}^H(n-1)\mathbf{x}(n) - \boldsymbol{\psi}^H(n-1)\mathbf{x}(n)|^2\}\end{aligned}\quad (10.8.41)$$

$$\begin{aligned}&= E\{|\hat{\mathbf{c}}^H(n-1)\mathbf{x}(n)|^2\} + E\{|\boldsymbol{\psi}^H(n-1)\mathbf{x}(n)|^2\} \\ &\quad + 2E\{\hat{\mathbf{c}}^H(n-1)\mathbf{x}(n)\mathbf{x}^H(n)\boldsymbol{\psi}(n-1)\}\end{aligned}\quad (10.8.42)$$

Since the term $\hat{\mathbf{c}}(n)$ does not depend on $\psi(n)$ and since the random sequences $\mathbf{x}(n)$ and $\psi(n - 1)$ are assumed independent, the last term in (10.8.42) is zero. Hence,

$$P_{\text{ex}}(n) \geq E\{|\psi^H(n - 1)\mathbf{x}(n)|^2\} \quad (10.8.43)$$

which provides a lower bound for the excess MSE of any a priori adaptation algorithm. Because $\psi(n)$ and $\mathbf{x}(n)$ are assumed independent, we obtain

$$E\{|\psi^H(n - 1)\mathbf{x}(n)|^2\} = \text{tr}(\mathbf{R}\mathbf{R}_\psi) \quad (10.8.44)$$

Similarly, neglecting the dependence between $\mathbf{x}(n)$ and $\tilde{\mathbf{c}}(n - 1)$, we have

$$E\{|\tilde{\mathbf{c}}^H(n - 1)\mathbf{x}(n)|^2\} = \text{tr}[\mathbf{R}\Phi(n - 1)] \quad (10.8.45)$$

which provides the a priori excess MSE. Furthermore, it can be shown that the DNS places a lower limit on the misadjustment, that is,

$$\mathcal{M}(n) = \frac{P_{\text{ex}}(n)}{P_o(n)} \geq \frac{E\{|\psi^H(n - 1)\mathbf{x}(n)|^2\}}{P_o(n)} = \frac{\text{tr}(\mathbf{R}\mathbf{R}_\psi)}{\sigma_v^2} = \eta^2(n) \quad (10.8.46)$$

10.8.3 LMS Algorithm

Using the LMS algorithm (10.4.12), the error vector in (10.8.34), and the Markov model in (10.8.28) with $\rho = 1$, we can easily obtain

$$\tilde{\mathbf{c}}(n) = [\mathbf{I} - 2\mu\mathbf{x}(n)\mathbf{x}^H(n)]\tilde{\mathbf{c}}(n - 1) + 2\mu\mathbf{x}(n)e_o^*(n) - \psi(n) \quad (10.8.47)$$

which, compared to (10.4.15), has one extra input. Since $\mathbf{x}(n)$, $e_o(n)$, and $\psi(n)$ are mutually independent, $\psi(n)$ adds only an extra term $\sigma_\psi^2 \mathbf{I}$ to the correlation of $\tilde{\mathbf{c}}(n)$.

Misadjustment. To determine the misadjustment, we perform orthogonal transformation of the correlation matrix of $\tilde{\mathbf{c}}(n)$. When we transform (10.4.28) to (10.4.30), using the orthogonal transformation (10.4.29), the presence of the diagonal matrix $\sigma_\psi^2 \mathbf{I}$ changes only the diagonal components with the addition of the term σ_ψ^2 . Indeed, we can easily show that

$$\theta_k(n) = \rho_k \theta_k(n - 1) + 4\mu^2 \lambda_k P_{\text{ex}}(n - 1) + 4\mu^2 P_o \lambda_k + \sigma_\psi^2 \quad (10.8.48)$$

where $P_o(n) = P_o = \sigma_v^2$ for large n . Clearly, (10.8.48) converges under the same conditions as (10.4.40). At steady state we have

$$\theta_k(\infty) = \rho_k \theta_k(\infty) + 4\mu^2 \lambda_k P_{\text{ex}}(\infty) + 4\mu^2 P_o \lambda_k + \sigma_\psi^2 \quad (10.8.49)$$

or using (10.4.36), we have

$$\theta_k(\infty) = \mu \frac{P_o + P_{\text{ex}}(\infty)}{1 - 2\mu\lambda_k} + \frac{1}{4\mu\lambda_k} \frac{\sigma_\psi^2}{1 - 2\mu\lambda_k} \quad (10.8.50)$$

which in conjunction with (10.4.55) and (10.4.56) gives

$$P_{\text{ex}}(\infty) = \frac{C(\mu)}{1 - C(\mu)} \sigma_v^2 + \frac{1}{4\mu} \frac{D(\mu)}{1 - C(\mu)} \sigma_\psi^2 \quad (10.8.51)$$

where $D(\mu) \triangleq \sum_{k=1}^M \frac{1}{1 - 2\mu\lambda_k}$ (10.8.52)

If $\mu\lambda_k \ll 1$, we have $C(\mu) \simeq \mu \text{tr}(\mathbf{R})$ and $D(\mu) \simeq M$, which lead to

$$P_{\text{ex}}(\infty) \simeq \mu \sigma_v^2 \text{tr}(\mathbf{R}) + \frac{1}{4\mu} M \sigma_\psi^2 \quad (10.8.53)$$

or $\mathcal{M}(\infty) \simeq \mu \text{tr}(\mathbf{R}) + \frac{1}{4\mu} M \frac{\sigma_\psi^2}{\sigma_v^2}$ (10.8.54)

Hence in the steady state, the misadjustment can be approximated by two terms. The first term is estimation misadjustment, which increases with μ , while the second term is the lag misadjustment, which decreases with μ . Therefore, an optimum value of μ exists that minimizes $\mathcal{M}(\infty)$, given by

$$\mu_{\text{opt}} \simeq \frac{\sigma_\psi}{2\sigma_v} \sqrt{\frac{M}{\text{tr}(\mathbf{R})}} \quad (10.8.55)$$

$$\text{or} \quad \mathcal{M}_{\min}(\infty) \simeq \frac{\sigma_\psi}{\sigma_v} \sqrt{M \text{tr}(\mathbf{R})} \quad (10.8.56)$$

MSD. To determine the MSD, consider (10.8.47). For small step size μ , the system matrix $[\mathbf{I} - 2\mu\mathbf{x}(n)\mathbf{x}^H(n)]$ is very close to the identity matrix. Hence using the direct averaging method due to Kushner (1984), we can obtain a close solution of $\tilde{\mathbf{c}}(n)$ by solving (10.8.47) in which the system matrix is replaced by its average $[\mathbf{I} - 2\mu\mathbf{R}]$, that is,

$$\tilde{\mathbf{c}}(n) = [\mathbf{I} - 2\mu\mathbf{R}]\tilde{\mathbf{c}}(n-1) + 2\mu\mathbf{x}(n)e_o^*(n) - \boldsymbol{\psi}(n) \quad (10.8.57)$$

where we have kept the same notation. Taking the covariance of both sides of (10.8.57), we obtain

$$\Phi(n) = [\mathbf{I} - 2\mu\mathbf{R}]\Phi(n-1)[\mathbf{I} - 2\mu\mathbf{R}] + 4\mu^2\sigma_v^2\mathbf{R} + \mathbf{R}_\psi \quad (10.8.58)$$

The approximate steady-state solution of (10.8.58) is given by

$$\mathbf{R}\Phi + \Phi\mathbf{R} \simeq 2\mu\sigma_v^2\mathbf{R} + \frac{\mathbf{R}_\psi}{2\mu} \quad (10.8.59)$$

where the second-order term $4\mu^2\mathbf{R}\Phi\mathbf{R}$ is ignored for small values of μ . After premultiplying (10.8.59) by \mathbf{R}^{-1} , we obtain

$$\Phi + \mathbf{R}^{-1}\Phi\mathbf{R} \simeq 2\mu\sigma_v^2 + \frac{\mathbf{R}^{-1}\mathbf{R}_\psi}{2\mu} \quad (10.8.60)$$

Taking the trace of (10.8.60) and using $\text{tr}(\mathbf{R}^{-1}\Phi\mathbf{R}) = \text{tr}(\Phi)$, we obtain

$$\text{tr}(\Phi) \simeq \mu M \sigma_v^2 + \frac{\text{tr}(\mathbf{R}^{-1}\mathbf{R}_\psi)}{4\mu} \quad (10.8.61)$$

By following the development in (10.8.28), it can be shown that (Problem 10.52) $\mathcal{D}(\infty) = \text{tr}(\Phi)$. Hence

$$\mathcal{D}(\infty) \simeq \mu M \sigma_v^2 + \frac{\text{tr}(\mathbf{R}^{-1}\mathbf{R}_\psi)}{4\mu} \quad (10.8.62)$$

As expected, the MSD has two terms: The estimation deviation is linearly proportional to μ while the lag deviation is inversely proportional to μ . The optimum value of the step size μ is obtained when both deviations are equal and is given by

$$\mu_{\text{opt}} \simeq \frac{1}{2} \sqrt{\frac{\text{tr}(\mathbf{R}^{-1}\mathbf{R}_\psi)}{M\sigma_v^2}} \quad (10.8.63)$$

$$\text{or} \quad \mathcal{D}_{\min}(\infty) = \sqrt{M\sigma_v^2 \text{tr}(\mathbf{R}^{-1}\mathbf{R}_\psi)} \quad (10.8.64)$$

EXAMPLE 10.8.1. To study the tracking performance of the LMS algorithm, we will simulate a slowly time-varying SOE whose parameters follow an almost random-walk behavior. The simulation setup is shown in Figure 10.42 and given by (10.8.27) and (10.8.28). The simulation parameters are as follows:

$$\begin{aligned} \mathbf{c}_o(n) \text{ model parameters:} \quad & \mathbf{c}_o(0) = \begin{bmatrix} -0.8 \\ 0.95 \end{bmatrix} \quad M = 2 \quad \rho = 0.999 \\ & \boldsymbol{\psi}(n) \sim \text{WGN}(\mathbf{0}, \mathbf{R}_\psi) \quad \mathbf{R}_\psi = (0.01)^2 \mathbf{I} \end{aligned}$$

Signal $\mathbf{x}(n)$ parameters: $\mathbf{x}(n) \sim \text{WGN}(\mathbf{0}, \mathbf{R}) \quad \mathbf{R} = \mathbf{I}$
 Noise $v(n)$ parameters: $v(n) \sim \text{WGN}(0, \sigma_v^2) \quad \sigma_v = 0.1$

For these values, the degree of nonstationarity from (10.8.32) is given by

$$\eta(n) = \frac{\sqrt{\text{tr}[\mathbf{R}\mathbf{R}_\psi]}}{\sigma_v} = 0.1414 < 1$$

which means that the LMS can track the time variations of the SOE.

Three different adaptations (slow, matched, and fast) of the LMS algorithm were designed. Their adaptation results are shown in Figures 10.43 through 10.48. From (10.8.55) and (10.8.63), the optimum performance is obtained when

$$\mu_{\text{opt}} = 0.05$$

for which $\mathcal{M}_{\min}(\infty) = 0.2$ and $\mathcal{D}_{\min}(\infty) = 0.002$. Hence, the following values for μ were selected for simulation:

Slow:	$\mu = 0.01$
Matched:	$\mu = 0.1$
Fast:	$\mu = 0.3$

Figure 10.43 shows the matched adaptation of parameter coefficients while Figure 10.44 shows the resulting $\mathcal{D}(n)$ and $\mathcal{M}(n)$. Clearly, the LMS tracks the varying coefficients nicely with expected small misregistration and deviation errors. Figure 10.45 shows the slow adaptation of parameter coefficients while Figure 10.46 shows the resulting $\mathcal{D}(n)$ and $\mathcal{M}(n)$. In this case, although the LMS algorithm tracks with bounded error variance, the tracking is not very good and the resulting misregistration errors are large. Finally, Figure 10.47 shows the fast adaptation of parameter coefficients while Figure 10.48 shows the resulting $\mathcal{D}(n)$ and $\mathcal{M}(n)$. In this case, although the algorithm is able to keep track of the slowly varying coefficients, the resulting variance is large and hence the estimation errors are large. Once again, the total errors are large compared to those for the matched case.

10.8.4 RLS Algorithm with Exponential Forgetting

Consider again the model given in Figure 10.42 and described in the analysis setup.

Misadjustment. To determine the misadjustment in tracking, we first evaluate the excess MSE caused by lag, that is, by the deviation between $E\{\mathbf{c}(n)\}$ and the optimum a priori filter $\mathbf{c}_o(n)$. Combining

$$\mathbf{c}(n) = \mathbf{c}(n-1) + \hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n)e^*(n) \quad (10.8.65)$$

with $e^*(n) = e_o^*(n) - \mathbf{x}^H(n)[\mathbf{c}(n-1) - \mathbf{c}_o(n-1)] \quad (10.8.66)$

and taking the expectation result in

$$E\{\mathbf{c}(n)\} = E\{\mathbf{c}(n-1)\} + E\{\hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n)\mathbf{x}^H(n)\}[E\{\mathbf{c}(n-1)\} - \mathbf{c}_o(n-1)] \quad (10.8.67)$$

because the expectation of $\hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n)e_o^*(n)$ vanishes. Using the approximation $E\{\hat{\mathbf{R}}^{-1}(n)\cdot\mathbf{x}(n)\mathbf{x}^H(n)\} \simeq (1-\lambda)\mathbf{I}$, we have

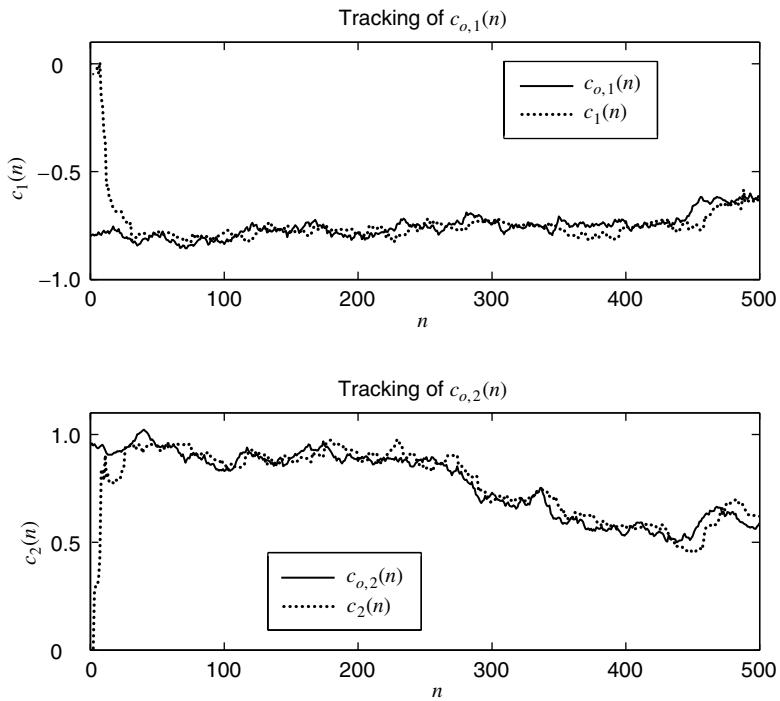
$$\tilde{\mathbf{c}}_{\text{lag}}(n) \simeq \lambda\tilde{\mathbf{c}}_{\text{lag}}(n) + \mathbf{c}_o(n-1) - \mathbf{c}_o(n) \quad (10.8.68)$$

or

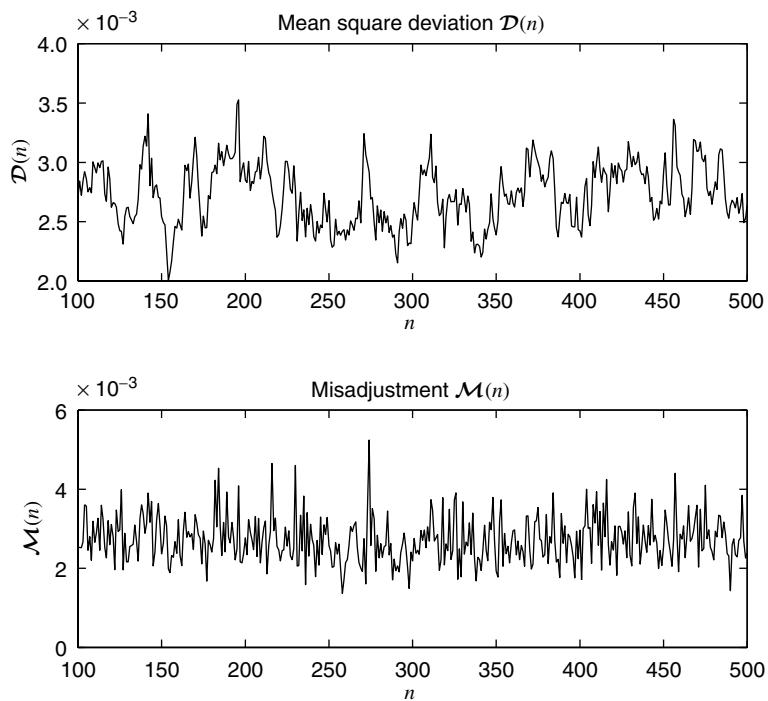
$$\tilde{\mathbf{c}}_{\text{lag}}(n) \simeq \lambda\tilde{\mathbf{c}}_{\text{lag}}(n-1) - \boldsymbol{\psi}(n) \quad (10.8.69)$$

for the random-walk ($\rho = 1$) model. The covariance matrix is

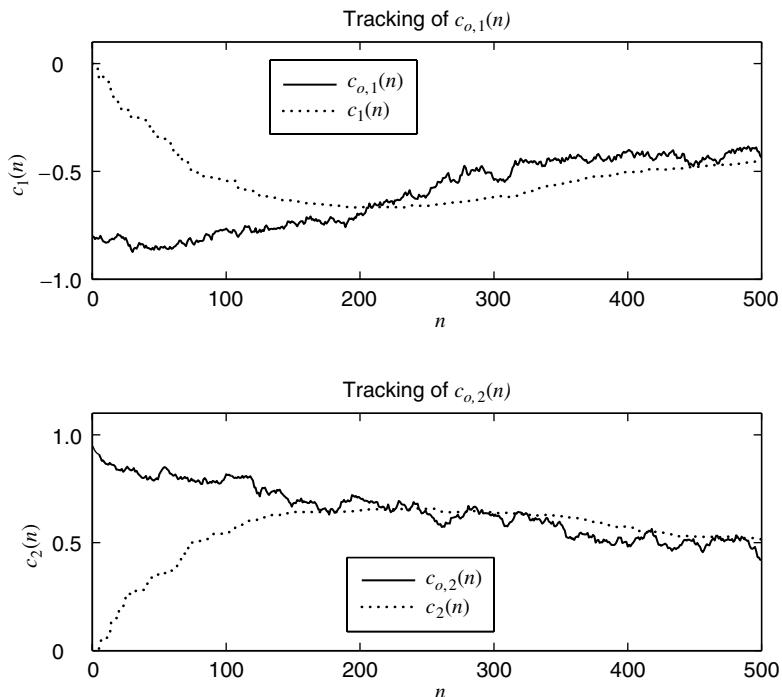
$$\Phi_{\text{lag}}(n) \simeq \lambda^2\Phi_{\text{lag}}(n-1) + \mathbf{R}_\psi \quad (10.8.70)$$

**FIGURE 10.43**

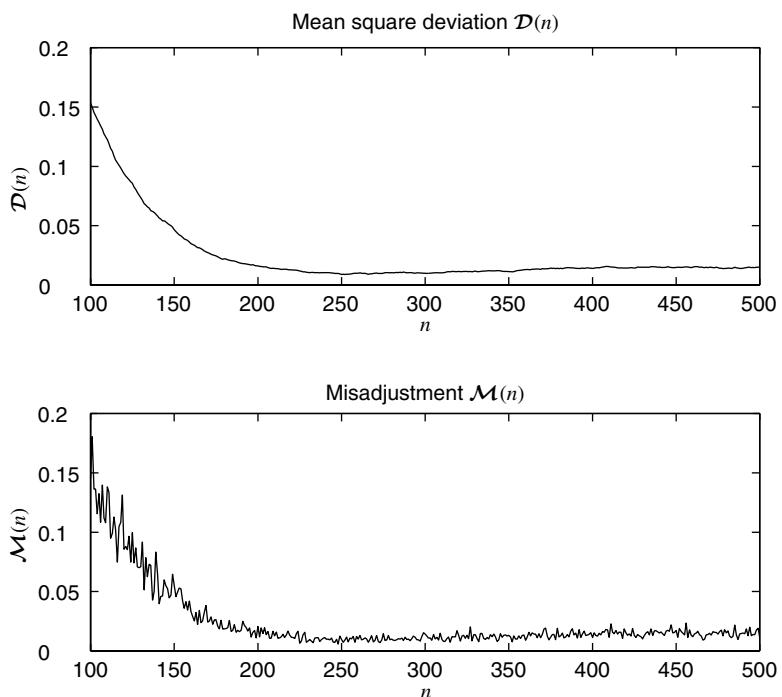
Matched adaptation of slowly time-varying parameters: LMS algorithm with $\mu = 0.1$.

**FIGURE 10.44**

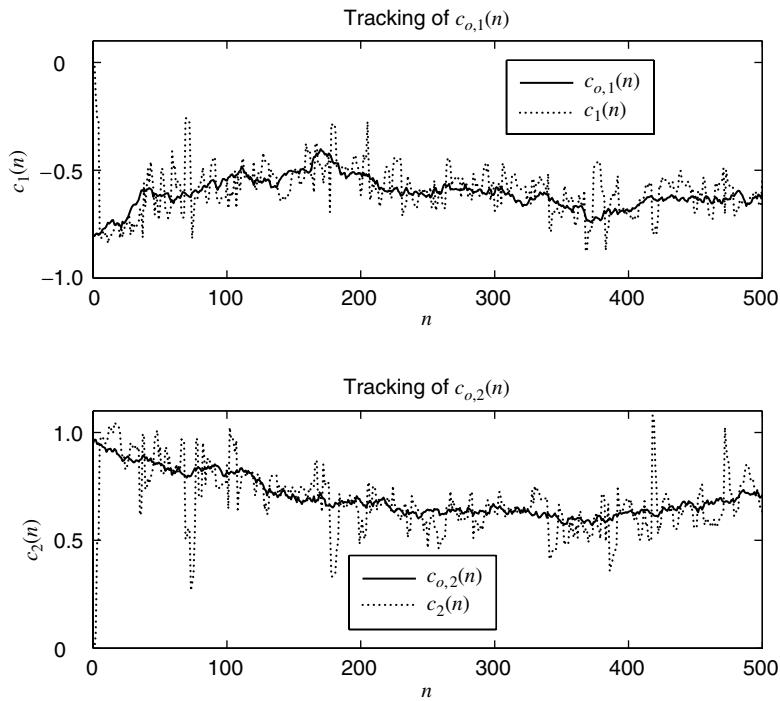
Learning curves of LMS algorithm with matched adaptation.

**FIGURE 10.45**

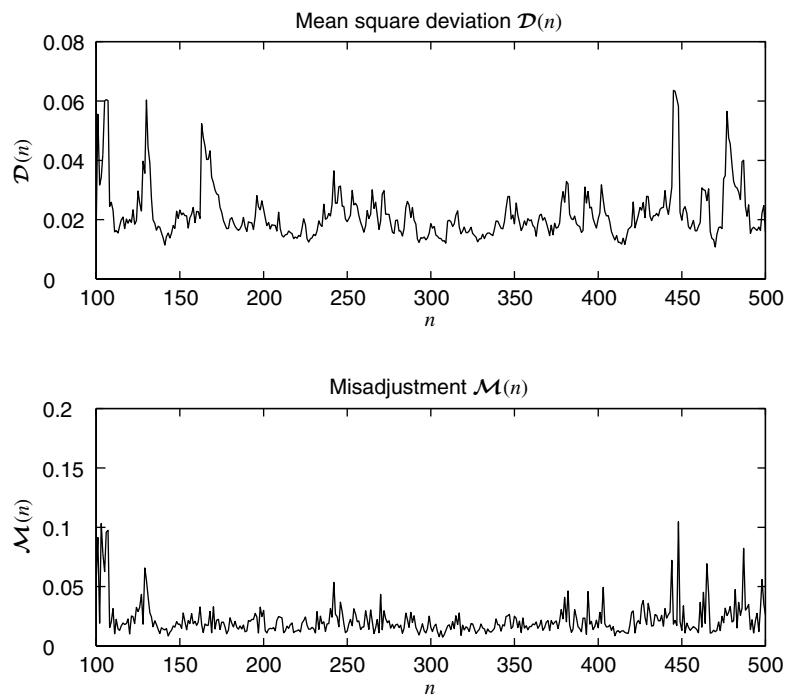
Slow adaptation of slowly time-varying parameters: LMS algorithm with $\mu = 0.01$.

**FIGURE 10.46**

Learning curves of LMS algorithm for slow adaptation.

**FIGURE 10.47**

Fast adaptation of slowly time-varying parameters: LMS algorithm with $\mu = 0.3$.

**FIGURE 10.48**

Learning curves of LMS algorithm for fast adaptation.

and in steady state (assuming $0 < \lambda < 1$)

$$\Phi_{\text{lag}}(\infty) \simeq \frac{1}{(1-\lambda)^2} \mathbf{R}_\psi \quad (10.8.71)$$

The lag excess MSE is

$$P_{\text{lag}}(\infty) = \text{tr}[\mathbf{R}\Phi(\infty)] \simeq \frac{1}{(1-\lambda)^2} \text{tr}[\mathbf{R}\mathbf{R}_\psi] \simeq \frac{1}{2(1-\lambda)} \text{tr}[\mathbf{R}\mathbf{R}_\psi] \quad (10.8.72)$$

because $(1-\lambda)^2 = (1+\lambda)(1-\lambda) \simeq 2(1-\lambda)$ for $\lambda \simeq 1$.

The excess MSE due to estimation is $[(1-\lambda)/2]M\sigma_v^2$, hence the total excess MSE is

$$P_{\text{ex}}(\infty) \simeq \frac{1-\lambda}{2} M \sigma_v^2 + \frac{1}{2(1-\lambda)} \sigma_\psi^2 \text{tr}(\mathbf{R}) \quad (10.8.73)$$

if $\mathbf{R}_\psi = \sigma_\psi^2 \mathbf{I}$. Finally, the misadjustment is given by

$$\mathcal{M}(\infty) \simeq \frac{1-\lambda}{2} M + \frac{\sigma_\psi^2 \text{tr}(\mathbf{R})}{2(1-\lambda)\sigma_v^2} \quad (10.8.74)$$

The first term in (10.8.74) is the estimation misadjustment, which is linearly proportional to $1-\lambda$, while the second term is the lag misadjustment, which is inversely proportional to $1-\lambda$. The optimum value of λ is given by

$$\lambda_{\text{opt}} \simeq 1 - \frac{\sigma_\psi}{\sigma_v} \sqrt{\frac{1}{M} \text{tr}(\mathbf{R})} \quad (10.8.75)$$

and the minimum misadjustment is given by

$$\mathcal{M}_{\min}(\infty) \simeq \frac{\sigma_\psi}{\sigma_v} \sqrt{M \text{tr}(\mathbf{R})} \quad (10.8.76)$$

MSD. An analysis similar to the MSD development of the LMS algorithm can be done to obtain

$$D(\infty) \simeq \frac{1-\lambda}{2} \sigma_v^2 \text{tr}(\mathbf{R}^{-1}) + \frac{\sigma_\psi^2}{2(1-\lambda)} \quad (10.8.77)$$

with

$$\lambda_{\text{opt}} \simeq 1 - \frac{\sigma_\psi}{\sigma_v} \sqrt{\frac{1}{\text{tr}(\mathbf{R}^{-1})}} \quad (10.8.78)$$

and

$$\mathcal{D}_{\min}(\infty) \simeq \frac{\sigma_\psi \sigma_v}{2} \sqrt{\text{tr}(\mathbf{R}^{-1})} \quad (10.8.79)$$

which again highlights the dependence of tracking abilities on λ .

EXAMPLE 10.8.2. To study the tracking performance of the RLS algorithm, we again simulate the slowly time-varying SOE given in Example 10.8.1 whose parameters are repeated here:

$$\begin{aligned} \mathbf{c}_o(n) \text{ model parameters: } \mathbf{c}_o(0) &= \begin{bmatrix} -0.8 \\ 0.95 \end{bmatrix} & M &= 2 & \rho &= 0.999 \\ \boldsymbol{\psi}(n) &\sim \text{WGN}(\mathbf{0}, \mathbf{R}_\psi) & \mathbf{R}_\psi &= (0.01)^2 \mathbf{I} \end{aligned}$$

Signal $\mathbf{x}(n)$ parameters: $\mathbf{x}(n) \sim \text{WGN}(\mathbf{0}, \mathbf{R})$ $\mathbf{R} = \mathbf{I}$

Noise $v(n)$ parameters: $v(n) \sim \text{WGN}(0, \sigma_v^2)$ $\sigma_v = 0.1$

For these values, the degree of nonstationarity is $\eta(n) = 0.1414$, which means that the RLS can track the time variations of the SOE.

Three different adaptations (slow, matched, and fast) of the RLS algorithm were designed. Their adaptation results are shown in Figures 10.49 through 10.54. From (10.8.75) and (10.8.77), the optimum misadjustment performance is obtained when

$$\lambda_{\text{opt}} = 0.9 \quad \text{with } \mathcal{M}_{\min}(\infty) = 0.2$$

while from (10.8.78) and (10.8.79), the optimum deviation performance is obtained when

$$\lambda_{\text{opt}} = 0.93 \quad \text{with } \mathcal{D}_{\min}(\infty) = 0.007$$

Hence, the following values for λ were selected for simulation:

Slow:	$\lambda = 0.99$
Matched:	$\lambda = 0.9$
Fast:	$\lambda = 0.5$

Figure 10.49 shows the matched adaptation of parameter coefficients while Figure 10.50 shows the resulting $\mathcal{D}(n)$ and $\mathcal{M}(n)$. Clearly, the RLS tracks the varying coefficients nicely with expected small misregistration and deviation errors. Figure 10.51 shows the slow adaptation of parameter coefficients while Figure 10.52 shows the resulting $\mathcal{D}(n)$ and $\mathcal{M}(n)$. In this case, although the RLS algorithm tracks with bounded error variance, the tracking is not very good and the resulting misregistration errors are large. Finally, Figure 10.53 shows the fast adaptation of parameter coefficients while Figure 10.54 shows the resulting $\mathcal{D}(n)$ and $\mathcal{M}(n)$. In this case, although the algorithm is able to keep track of the slowly varying coefficients, the resulting variance is large and hence the estimation errors are large. Once again, the total errors are large compared to those for the matched case.

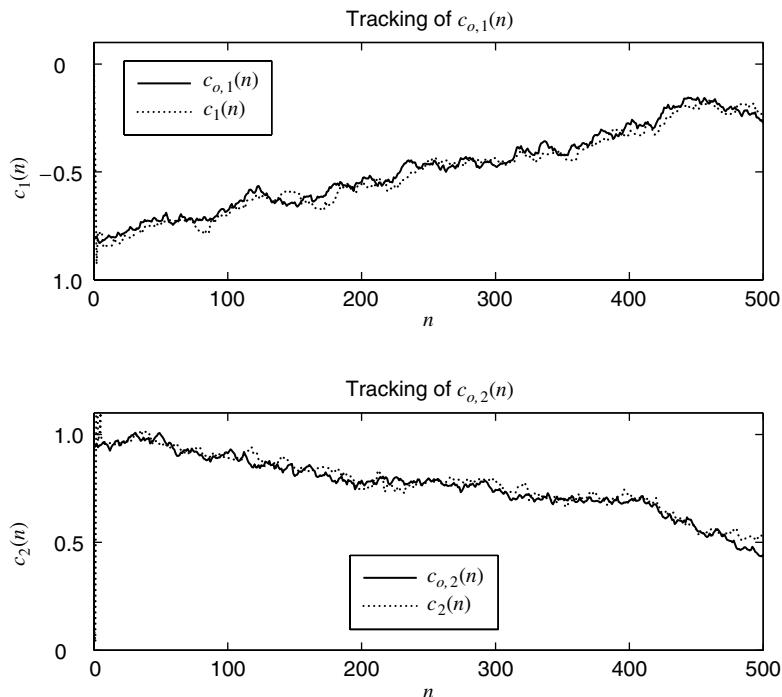
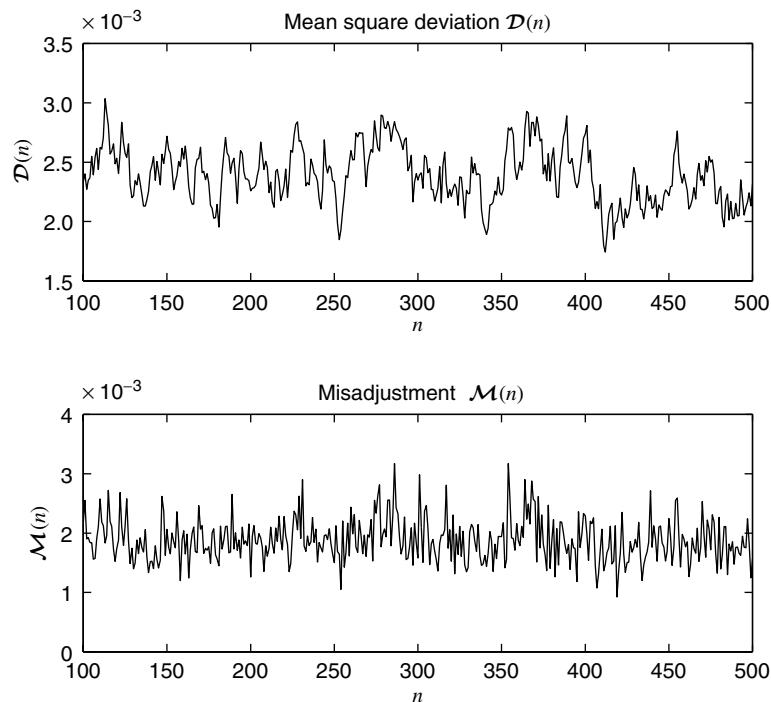


FIGURE 10.49

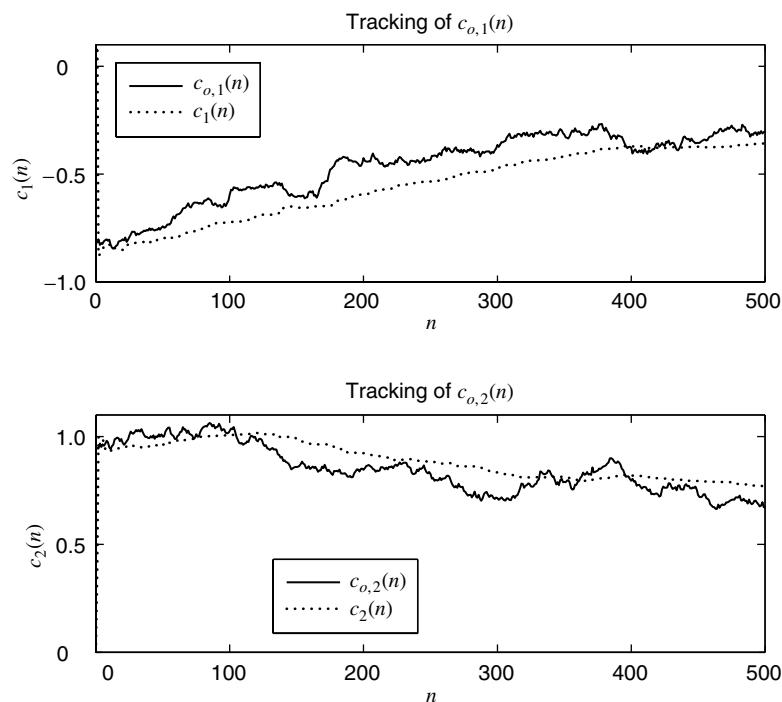
Matched adaptation of slowly time-varying parameters: RLS algorithm with $\lambda = 0.9$.

10.8.5 Comparison of Tracking Performance

When the optimum filter drifts like a random walk with small increment variance σ_ψ^2 , the tracking performance for the LMS algorithm is given by (10.8.54) and (10.8.62) while that for the RLS algorithm is given by (10.8.74) and (10.8.77). Whether the LMS or the RLS algorithm is better depends on matrices \mathbf{R} and \mathbf{R}_ψ . A general comparison is difficult to make, but some guidelines have been developed for particular cases. It has been shown that (Haykin 1996)

**FIGURE 10.50**

Learning curves of RLS algorithm for matched adaptation.

**FIGURE 10.51**

Slow adaptation of slowly time-varying parameters: RLS algorithm with $\lambda = 0.99$.

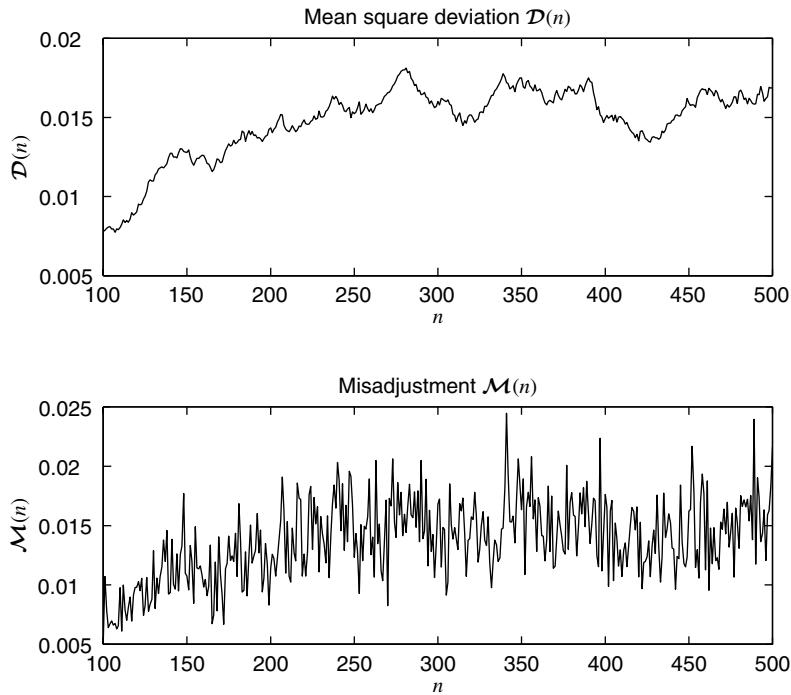


FIGURE 10.52
Learning curves of RLS algorithm for slow adaptation.

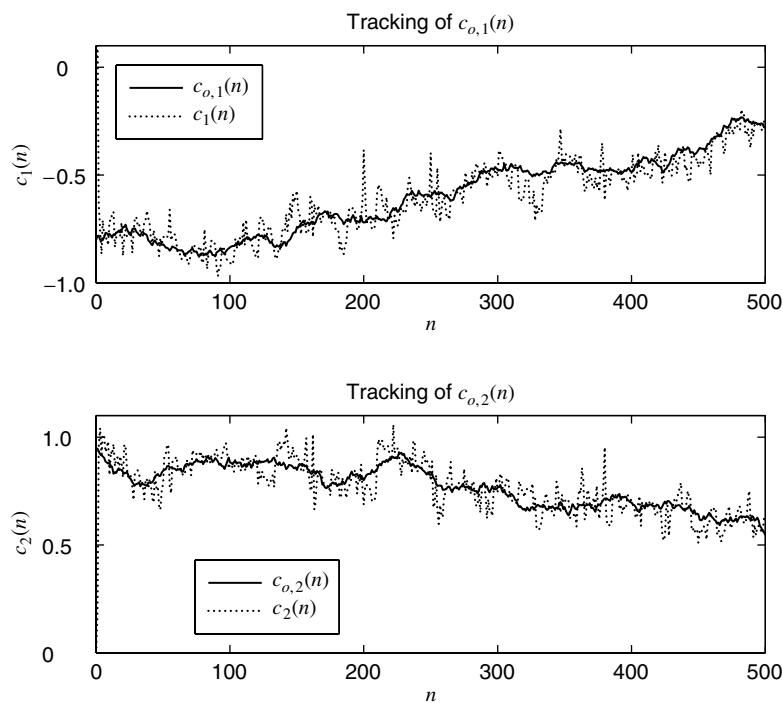


FIGURE 10.53
Fast adaptation of slowly time-varying parameters: RLS algorithm with $\lambda = 0.5$.

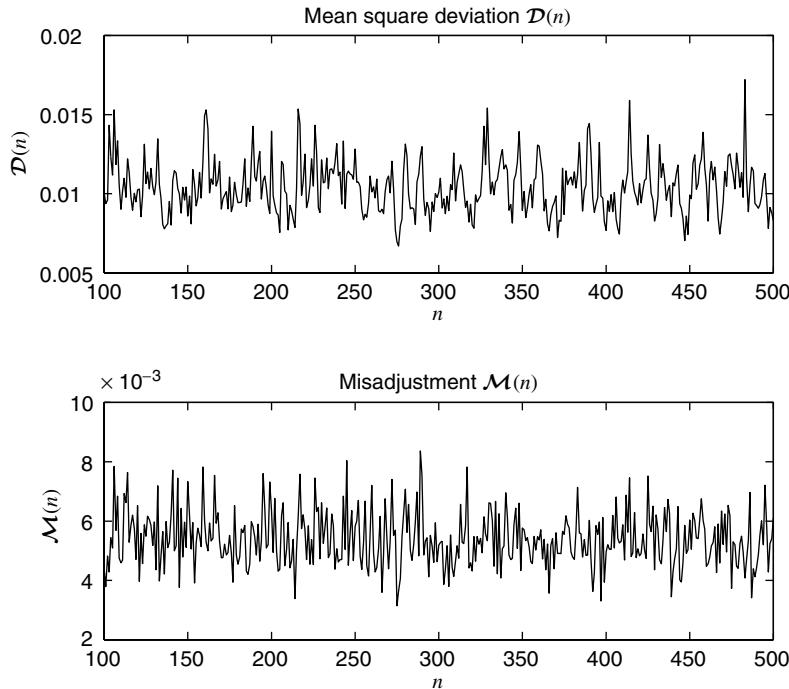


FIGURE 10.54

Learning curves of RLS algorithm for fast adaptation.

- When $\mathbf{R}_\psi = \sigma_\psi^2 \mathbf{I}$, then both the LMS and RLS algorithms produce essentially the same minimum levels of MSD and misadjustment. However, this analysis is true only asymptotically and for slowly varying parameters (small σ_ψ^2).
- When $\mathbf{R}_\psi = \alpha \mathbf{R}$ where α is a constant, then the LMS algorithm produces smaller values of the minimum levels of MSD and misadjustment than the RLS algorithm does.
- When $\mathbf{R}_\psi = \beta \mathbf{R}^{-1}$ where β is a constant, then the RLS algorithm is better than the LMS algorithm in producing the smaller values of the minimum levels of MSD and misadjustment.

In summary, we should state that in practice the comparison of the acquisition and tracking performance of LMS and RLS adaptive filters is a very complicated subject. Although the previous analysis provides some insight only extensive simulations in the context of a specific application can help to choose the appropriate algorithm.

10.9 SUMMARY

In this chapter we discussed the theory of operation, design, performance evaluation, implementation, and applications of adaptive filters. The most significant attribute of an adaptive filter is its ability to incrementally adjust its coefficients so as to improve a predefined criterion of performance over time.

We basically developed and analyzed two families of adaptive filtering algorithms:

- The family of LMS FIR adaptive filters, which are based on a stochastic version of the steepest-descent optimization algorithm.
- The family of RLS FIR adaptive filters, which are based on a stochastic version of the Newton-type optimization algorithms.

Both types of approaches can be used to develop adaptive algorithms for direct-form and lattice-ladder FIR filter structures.

For LMS adaptive filters we focused on direct-form structures because those are the most widely used and studied. However, we briefly discussed transform-domain and subband implementations because they offer a viable solution for applications that require adaptive filters with very long impulse responses.

All RLS FIR adaptive filters discussed in this chapter exhibit identical performance if they are implemented using infinite-precision arithmetic. However, they differ in terms of computational complexity and performance under finite-word-length implementations. The various types of RLS algorithms are summarized in Figure 10.40. We stress that algorithms for array processing can be used for FIR filtering (shift-invariant input data vector), but not vice versa. However, such a practice is not recommended because the computational complexity is much higher. The LMS algorithm (Section 10.4), the CRLS algorithm (Section 10.5), and the QR decomposition-based algorithms (Section 10.6) are general and can be used for both array processing and FIR filtering applications. In contrast, the fast RLS algorithms in Section 10.7 can be used only for FIR filtering and prediction applications. The steady-state performance of LMS and RLS algorithms in a stationary environment is discussed in Sections 10.4 and 10.5, whereas their tracking performance in a nonstationary environment is analyzed in Section 10.8.

The treatment of adaptive filters in this chapter has been quite extensive, in both number of topics and depth. However, the following important topics have been omitted:

- IIR adaptive filters (Treichler et al. 1987; Johnson 1984; Shynk 1989; Regalia 1995; Netto et al. 1995; Williamson 1998). Although adaptive IIR filters have the potential to offer the same performance as FIR filters with less computational complexity, they are not widely used in practical applications. The main reasons are related to the nonquadratic nature of their performance error surface (see Section 6.2) and the additional stability problems caused by the presence of poles in their system function.
- Adaptive filters using nonlinear filtering structures and neural networks (Grant and Mulgrew 1995; Haykin 1996; Mathews 1991). The need for such filters arises in applications involving nonlinear input-output relationships, nonlinear detectors (e.g., data equalization), and non-Gaussian or impulsive noise. The optimization required in some of these cases can be performed using genetic optimization algorithms (Tang et al. 1996).
- FIR direct-form and lattice-ladder LS adaptive filters for multichannel signals (Slock 1993; Ling 1993b; Carayannis et al. 1986).

PROBLEMS

10.1 Consider the process $x(n)$ generated using the AR(3) model

$$x(n) = -0.729x(n-3) + w(n)$$

where $w(n) \sim \text{WGN}(0, 1)$. We want to design a linear predictor of $x(n)$ using the SDA algorithm. Let

$$\hat{y}(n) = \hat{x}(n) = c_{0,1}x(n-1) + c_{0,2}x(n-2) + c_{0,3}x(n-3)$$

- Determine the 3×3 autocorrelation matrix \mathbf{R} of $x(n)$, and compute its eigenvalues $\{\lambda_i\}_{i=1}^3$.
- Determine the 3×1 cross-correlation vector \mathbf{d} .
- Choose the step size μ so that the resulting response is overdamped. Now implement the SDA

$$\mathbf{c}_k = [c_{k,1} \ c_{k,2} \ c_{k,3}]^T = \mathbf{c}_{k-1} + 2\mu(\mathbf{d} - \mathbf{R}\mathbf{c}_{k-1})$$

and plot the trajectories of $\{c_{k,i}\}_{i=1}^3$ as a function of k .

- Repeat part (c) by choosing μ so that the response is underdamped.

- 10.2** In the SDA algorithm, the index k is an iteration index and not a time index. However, we can treat it as a time index and use the instantaneous filter coefficient vector \mathbf{c}_k to filter data at $n = k$. This will result in an *asymptotically optimum* filter whose coefficients will converge to the optimum one. Consider the process $x(n)$ given in Problem 10.1.

- (a) Generate 500 samples of $x(n)$ and implement the asymptotically optimum filter. Plot the signal $\hat{y}(n)$.
- (b) Implement the optimum filter \mathbf{c}_0 on the same sequence, and plot the resulting $\hat{y}(n)$.
- (c) Comment on the above two plots.

- 10.3** Consider the AR(2) process $x(n)$ given in Example 10.3.1. We want to implement the Newton-type algorithm for faster convergence using

$$\mathbf{c}_k = \mathbf{c}_{k-1} - \mu \mathbf{R}^{-1} \nabla P(\mathbf{c}_{k-1})$$

- (a) Using $a_1 = -1.5955$ and $a_2 = 0.95$, implement the above method for $\mu = 0.1$ and $\mathbf{c}_0 = \mathbf{0}$. Plot the locus of $c_{k,1}$ versus $c_{k,2}$.
- (b) Repeat part (a), using $a_1 = -0.195$ and $a_2 = 0.95$.
- (c) Repeat parts (a) and (b), using the optimum step size for μ that results in the fastest convergence.

- 10.4** Consider the adaptive linear prediction of an AR(2) process $x(n)$ using the LMS algorithm in which

$$x(n) = 0.95x(n-1) - 0.9x(n-2) + w(n)$$

where $w(n) \sim \text{WGN}(0, \sigma_w^2)$. The adaptive predictor is a second-order one given by $\mathbf{a}(n) = [a_1(n) \ a_2(n)]^T$.

- (a) Implement the LMS algorithm given in Table 10.3 as a MATLAB function

$$[c, e] = \text{lplms}(x, y, mu, M, c0).$$

which computes filter coefficients in c and the corresponding error in e , given signal x , desired signal y , step size mu , filter order M , and the initial coefficient vector $c0$.

- (b) Generate 500 samples of $x(n)$, and obtain linear predictor coefficients using the above function. Use step size μ so that the algorithm converges in the mean. Plot predictor coefficients as a function of time along with the true coefficients.
- (c) Repeat the above simulation 1000 times to obtain the learning curve, which is obtained by averaging the squared error $|e(n)|^2$. Plot this curve and compare its steady-state value with the theoretical MSE.

- 10.5** Consider the adaptive echo canceler given in Figure 10.23. The FIR filter $c_0(n)$ is given by

$$c_0(n) = (0.9)^n \quad 0 \leq n \leq 2$$

In this simulation, ignore the far-end signal $u(n)$. The data signal $x(n)$ is a zero-mean, unit-variance white Gaussian process, and $y(n)$ is its echo.

- (a) Generate 1000 samples of $x(n)$ and determine $y(n)$. Use these signals to obtain a fourth-order LMS echo canceler in which the step size μ is chosen to satisfy (10.4.40) and $\mathbf{c}(0) = \mathbf{0}$. Obtain the final echo canceler coefficients and compare them with the true ones.
- (b) Repeat the above simulation 500 times, and obtain the learning curve. Plot this curve along with the actual MSE and comment on the plot.
- (c) Repeat parts (a) and (b), using a third-order echo canceler.
- (d) Repeat parts (a) and (b), using one-half the value of μ used in the first part.

- 10.6** The normalized LMS (NLMS) algorithm is given in (10.4.67), in which the effective step size is time-varying and is given by $\tilde{\mu}/\|\mathbf{x}(n)\|^2$, where $0 < \tilde{\mu} < 1$.

- (a) Modify the function `firlms` to implement the NLMS algorithm and obtain the function

$$[c, e] = \text{nfirlms}(x, y, mu, M, c0).$$

- (b) Choose $\tilde{\mu} = 0.1$ and repeat Problem 10.4. Compare your results in terms of convergence speed.
- (c) Choose $\tilde{\mu} = 0.1$ and repeat Problem 10.5(a) and (b). Compare your results in terms of convergence speed.

10.7 Another variation of the LMS algorithm is called the *sign-error* LMS algorithm, in which the coefficient update equation is given by

$$\mathbf{c}(n) = \mathbf{c}(n - 1) + 2\mu \operatorname{sgn}[e(n)] \mathbf{x}(n)$$

where

$$\operatorname{sgn}[e(n)] = \begin{cases} 1 & \operatorname{Re}[e(n)] > 0 \\ 0 & \operatorname{Re}[e(n)] = 0 \\ -1 & \operatorname{Re}[e(n)] < 0 \end{cases}$$

The advantage of this algorithm is that the multiplication is replaced by a sign change, and if μ is chosen as a negative power of 2, then the multiplication is replaced by a shifting operation that is easy and fast to implement. Furthermore, since $\operatorname{sgn}(x) = x/|x|$, the effective step size $\tilde{\mu}$ is inversely proportional to the magnitude of the error.

- (a) Modify the function `firlms` to implement the sign-error LMS algorithm and obtain the function

$$[c, e] = \text{sefirlms}(x, y, mu, M, c0).$$

- (b) Repeat Problem 10.4 and compare your results in terms of convergence speed.
- (c) Repeat Problem 10.5(a) and (b) and compare your results in terms of convergence speed.

10.8 Consider an AR(1) process $x(n) = ax(n - 1) + w(n)$, where $w(n) \sim \text{WGN}(0, \sigma_w^2)$. We wish to design a one-step first-order linear predictor using the LMS algorithm

$$\begin{aligned}\hat{x}(n) &= \hat{a}(n - 1)x(n - 1) \\ e(n) &= x(n) - \hat{x}(n) \\ \hat{a}(n) &= \hat{a}(n - 1) + 2\mu e(n)x(n - 1)\end{aligned}$$

where μ is the adaptation step size.

- (a) Determine the autocorrelation $r_x(l)$, the optimum first-order linear predictor, and the corresponding MMSE.
- (b) Using the independence assumption, first determine and then solve the difference equation for $E\{\hat{a}(n)\}$.
- (c) For $a = \pm 0.95$, $\mu = 0.025$, $\sigma_x^2 = 1$, and $0 \leq n < N = 500$, determine the ensemble average of $E\{\hat{a}(n)\}$ using 200 independent runs and compare with the theoretical curve obtained in part (b).
- (d) Using the independence assumption, first determine and then solve the difference equation for $P(n) = E\{e^2(n)\}$.
- (e) Repeat part (c) for $P(n)$ and comment upon the results.

10.9 Using the a posteriori error $\varepsilon(n) = y(n) - \mathbf{c}^H(n)\mathbf{x}(n)$, derive the coefficient updating formulas for the a posteriori error LMS algorithm. Note: Refer to Equations (10.2.20) to (10.2.22).

10.10 Solve the interference cancelation problem described in Example 6.4.1, using the LMS algorithm, and compare its performance to that of the optimum canceler.

10.11 Repeat the convergence analysis of the LMS algorithm for the complex case, using formula (10.4.27) instead of (10.4.28).

10.12 Consider the total transient excess MSE, defined by

$$P_{\text{tr}}^{(\text{total})} = \sum_{n=0}^{\infty} P_{\text{tr}}(n)$$

in Section 10.4.3.

- (a) Show that $P_{\text{tr}}^{(\text{total})}$ can be written as $P_{\text{tr}}^{(\text{total})} = \lambda^T (\mathbf{I} - \mathbf{B})^{-1} \Delta\theta(0)$, where $\Delta\theta(0)$ is the initial (i.e., at $n = 0$) deviation of the filter coefficients from their optimum setting.
- (b) Starting with the formula in step (a), show that

$$P_{\text{tr}}^{(\text{total})} = \frac{1}{4\mu} \frac{\sum_{i=1}^M \frac{\Delta\theta_i(0)}{1 - 2\mu\lambda_i}}{1 - \sum_{i=1}^M \frac{\mu\lambda_i}{1 - 2\mu\lambda_i}}$$

- (c) Show that if $\mu\lambda_k \ll 1$, then

$$P_{\text{tr}}^{(\text{total})} \simeq \frac{1}{4\mu} \frac{\sum_{i=1}^M \Delta\theta_i(0)}{1 - \mu \text{tr}(\mathbf{R})} \simeq \frac{1}{4\mu} \sum_{i=1}^M \Delta\theta_i(0)$$

which is formula (10.4.62), discussed in Section 10.4.3.

- 10.13** The frequency sampling structure for the implementation of an FIR filter $H(z) = \sum_{n=0}^{M-1} h(n)z^{-n}$ is specified by the following relation

$$H(z) = \frac{1 - z^{-M}}{M} \sum_{k=0}^{M-1} \frac{H(e^{j2\pi k/M})}{1 - e^{j2\pi k/M} z^{-1}} \triangleq H_1(z) H_2(z)$$

where $H_1(z)$ is a comb filter with M zeros equally spaced on the unit circle and $H_2(z)$ is a filter bank of resonators. Note that $\tilde{H}(k) \triangleq H(e^{j2\pi k/M})$, the DFT of $\{h(n)\}_0^{M-1}$, provides coefficients of the filter. Derive an LMS-type algorithm to update these coefficients, and sketch the resulting adaptive filter structure.

- 10.14** There are applications in which the use of a non-MSE criterion may be more appropriate. To this end, suppose that we wish to design and study the behavior of an “LMS-like” algorithm that minimizes the cost function $P^{(k)} = E\{e^{2k}(n)\}$, $k = 1, 2, 3, \dots$, using the model defined in Figure 10.19.

- (a) Use the instantaneous gradient vector to derive the coefficient updating formula for this LMS-like algorithm.
 (b) Using the assumptions introduced in Section 10.4.2 show that

$$E\{\tilde{\mathbf{c}}(n)\} = [\mathbf{I} - 2\mu k(2k-1)E\{e_o^{2(k-1)}(n)\}\mathbf{R}]E\{\tilde{\mathbf{c}}(n-1)\}$$

where \mathbf{R} is the input correlation matrix.

- (c) Show that the derived algorithm converges in the mean if

$$0 < 2\mu < \frac{1}{k(2k-1)E\{e_o^{2(k-1)}(n)\}\lambda_{\max}}$$

where λ_{\max} is the largest eigenvalue of \mathbf{R} .

- (d) Show that for $k = 1$ the results in parts (a) to (c) reduce to those for the standard LMS algorithm.

- 10.15** Consider the noise cancelation system shown in Figure 10.6. The useful signal is a sinusoid $s(n) = \cos(\omega_0 n + \phi)$, where $\omega_0 = \pi/16$ and the phase ϕ is a random variable uniformly distributed from 0 to 2π . The noise signals are given by $v_1(n) = 0.9 v_1(n-1) + w(n)$ and $v_2(n) = -0.75 v_2(n-1) + w(n)$, where the sequences $w(n)$ are WGN(0, 1).

- (a) Design an optimum filter of length M and choose a reasonable value for M_o by plotting the MMSE as a function of M .
 (b) Design an LMS filter with M_o coefficients and choose the step size μ to achieve a 10 percent misadjustment.
 (c) Plot the signals $s(n)$, $s(n) + v_1(n)$, $v_2(n)$, the clean signal $e_o(n)$ using the optimum filter, and the clean signal $e_{\text{lms}}(n)$ using the LMS filter, and comment upon the obtained results.

10.16 A modification of the LMS algorithm, known as the *momentum LMS (MLMS)*, is defined by

$$\mathbf{c}(n) = \mathbf{c}(n-1) + 2\mu e^*(n)\mathbf{x}(n) + \alpha[\mathbf{c}(n-1) - \mathbf{c}(n-2)]$$

where $|\alpha| < 1$ (Roy and Shynk 1990).

- (a) Rewrite the previous equation to show that the algorithm has the structure of a low-pass ($0 < \alpha < 1$) or a high-pass ($-1 < \alpha < 0$) filter.
- (b) Explain intuitively the effect of the momentum term $\alpha[\mathbf{c}(n-1) - \mathbf{c}(n-2)]$ on the filter's convergence behavior.
- (c) Repeat the computer equalization experiment in Section 10.4.4, using both the LMS and the MLMS algorithms for the following cases, and compare their performance:
 - i. $W = 3.1, \mu_{\text{lms}} = \mu_{\text{mlms}} = 0.01, \alpha = 0.5$.
 - ii. $W = 3.1, \mu_{\text{lms}} = 0.04, \mu_{\text{mlms}} = 0.01, \alpha = 0.5$.
 - iii. $W = 3.1, \mu_{\text{lms}} = \mu_{\text{mlms}} = 0.04, \alpha = 0.2$.
 - iv. $W = 4, \mu_{\text{lms}} = \mu_{\text{mlms}} = 0.03, \alpha = 0.3$.

10.17 In Section 10.4.5 we presented the leaky LMS algorithm [see (10.4.88)]

$$\mathbf{c}(n) = (1 - \alpha\mu)\mathbf{c}(n-1) + \mu e^*(n)\mathbf{x}(n)$$

where $0 < \alpha \ll 1$ is the leakage coefficient.

- (a) Show that the coefficient updating equation can be obtained by minimizing

$$P(n) = |e(n)|^2 + \alpha\|\mathbf{c}(n)\|^2$$

- (b) Using the independence assumptions, show that

$$E\{\mathbf{c}(n)\} = [\mathbf{I} - \mu(\mathbf{R} + \alpha\mathbf{I})]E\{\mathbf{c}(n-1)\} + \mu\mathbf{d}$$

where $\mathbf{R} = E\{\mathbf{x}(n)\mathbf{x}^H(n)\}$ and $\mathbf{d} = E\{\mathbf{x}(n)y^*(n)\}$.

- (c) Show that if $0 < \mu < 2/(\alpha + \lambda_{\max})$, where λ_{\max} is the maximum eigenvalue of \mathbf{R} , then

$$\lim_{n \rightarrow \infty} E\{\mathbf{c}(n)\} = (\mathbf{R} + \alpha\mathbf{I})^{-1}\mathbf{d}$$

that is, in the steady state $E\{\mathbf{c}(\infty)\} \neq \mathbf{c}_o = \mathbf{R}^{-1}\mathbf{d}$.

10.18 There are various communications and speech signal processing applications that require the use of filters with linear phase (Manolakis et al. 1984). For simplicity, assume that m is even.

- (a) Derive the normal equations for an optimum FIR filter that satisfies the constraints
 - i. $\mathbf{c}_m^{(\text{lp})} = \mathbf{J}\mathbf{c}_m^{(\text{lp})}$ (linear phase)
 - ii. $\mathbf{c}_m^{(\text{cgd})} = -\mathbf{J}\mathbf{c}_m^{(\text{cgd})}$ (constant group delay).
- (b) Show that the obtained optimum filters can be expressed as $\mathbf{c}_m^{(\text{lp})} = \frac{1}{2}(\mathbf{c}_m + \mathbf{J}\mathbf{c}_m)$ and $\mathbf{c}_m^{(\text{cgd})} = \frac{1}{2}(\mathbf{c}_m - \mathbf{J}\mathbf{c}_m)$, where \mathbf{c}_m is the unconstrained optimum filter.
- (c) Using the results in part (b) and the algorithm of Levinson, derive lattice-ladder structure for the constrained optimum filters.
- (d) Repeat parts (a), (b), and (c) for the linear predictor with linear phase, which is specified by $\mathbf{a}_m^{(\text{lp})} = \mathbf{J}\mathbf{a}_m^{(\text{lp})}$.
- (e) Develop an LMS algorithm for the linear-phase filter $\mathbf{c}_m^{(\text{lp})} = \mathbf{J}\mathbf{c}_m^{(\text{lp})}$ and sketch the resulting structure. Can you draw any conclusions regarding the step size and the misadjustment of this filter compared to those of the unconstrained LMS algorithm?

10.19 In this problem, we develop and analyze by simulation an LMS-type adaptive lattice predictor introduced in Griffiths (1977). We consider the all-zero lattice filter defined in (7.5.7), which is completely specified by the lattice parameters $\{k_m\}_0^{M-1}$. The input signal is assumed wide-sense stationary.

- (a) Consider the cost function

$$P_m^{\text{f b}} = E\{|e_m^{\text{f}}(n)|^2 + |e_m^{\text{b}}(n)|^2\}$$

which provides the total prediction error power at the output of the m th stage, and show that

$$\frac{\partial P_m^{\text{fb}}}{\partial k_{m-1}^*} = 2E\{e_m^*(n)e_{m-1}^b(n-1) + e_{m-1}^*(n)e_m^b(n)\}$$

- (b) Derive the updating formula using the LMS-type approach

$$k_m(n) = k_m(n-1) - 2\mu(n)[e_m^*(n)e_{m-1}^b(n-1) + e_{m-1}^*(n)e_m^b(n)]$$

where the normalized step size $\mu(n) = \bar{\mu}/E_{m-1}^b(n)$ is computed in practice by using the formula

$$E_{m-1}(n) = \alpha E_{m-1}(n-1) + (1-\alpha)[|e_{m-1}^f(n)|^2 + |e_{m-1}^b(n-1)|^2]$$

where $0 < \alpha < 1$. Explain the role and proper choice of α , and determine the proper initialization of the algorithm.

- (c) Write a MATLAB function to implement the derived algorithm, and compare its performance with that of the LMS algorithm in the linear prediction problem discussed in Example 10.4.1.

10.20 Consider a signal $x(n)$ consisting of a harmonic process plus white noise, that is,

$$x(n) = A \cos(\omega_1 n + \phi) + w(n)$$

where ϕ is uniformly distributed from 0 to 2π and $w(n) \sim \text{WGN}(0, \sigma_w^2)$.

- (a) Determine the output power $\sigma_y^2 = E\{y^2(n)\}$ of the causal and stable filter

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n-k)$$

and show that we can cancel the harmonic process using the ideal notch filter

$$H(e^{j\omega}) = \begin{cases} 1 & \omega = \omega_1 \\ 0 & \text{otherwise} \end{cases}$$

Is the obtained ideal notch filter practically realizable? That is, is the system function rational? Why?

- (b) Consider the second-order notch filter

$$H(z) = \frac{D(z)}{A(z)} = \frac{1 + a z^{-1} + z^{-2}}{1 + a \rho z^{-1} + \rho^2 z^{-2}} = \frac{D(z)}{D(z/\rho)}$$

where $-1 < \rho < 1$ determines the steepness of the notch and $a = -2 \cos \omega_0$ its frequency. We fix ρ , and we wish to design an adaptive filter by adjusting a .

- i. Show that for $\rho \approx 1$, $\sigma_y^2 = A^2|H(e^{j\omega_1})|^2 + \sigma_w^2$, and plot σ_y^2 as a function of the frequency ω_0 for $\omega_1 = \pi/6$.
- ii. Evaluate $d\sigma_y^2(a)/da$ and show that the minimum of $\sigma_y^2(a)$ occurs for $a = -2 \cos \omega_1$.
- (c) Using a direct-form II structure for the implementation of $H(z)$ and the property $dY(z)/da = [dH(z)/da]X(z)$, show that the following relations

$$\begin{aligned} s_2(n) &= -a(n-1)\rho s_2(n-1) - \rho^2 s_2(n-2) + (1 - gr)s_1(n-1) \\ g(n) &= s_2(n) - \rho s_2(n-2) \\ s_1(n) &= -a(n-1)\rho s_1(n-1) - \rho^2 s_1(n-2) + x(n) \\ y(n) &= s_1(n) + a(n-1)s_1(n-1) + s_1(n-2) \\ a(n) &= a(n-1) - 2\mu y(n)g(n) \end{aligned}$$

constitute an adaptive LMS notch filter. Draw its block diagram realization.

- (d) Simulate the operation of the obtained adaptive filter for $\rho = 0.9$, $\omega_1 = \pi/6$, and SNR 5 and 15 dB. Plot $\omega_0(n) = \arccos[-a(n)/2]$ as a function of n , and investigate the tradeoff between convergence rate and misadjustment by experimenting with various values of μ .

- 10.21** Consider the AR(2) process given in Problem 10.4. We will design the adaptive linear predictor using the RLS algorithm. The adaptive predictor is a second-order one given by $\mathbf{c}(n) = [c_1(n) \ c_2(n)]^T$.
- (a) Develop a MATLAB function to implement the RLS algorithm given in Table 10.6

```
[c, e] = rls(x, y, lambda, delta, M, c0);
```

which computes filter coefficients in \mathbf{c} and the corresponding error in e given signal x , desired signal y , forgetting factor λ , initialization parameter δ , filter order M , and the initial coefficient vector \mathbf{c}_0 . To update $\mathbf{P}(n)$, compute only the upper or lower triangular part and determine the other part by using Hermitian symmetry.

- (b) Generate 500 samples of $x(n)$ and obtain linear predictor coefficients using the above function. Use a very small value for δ (for example, 0.001) and various values of $\lambda = 0.99, 0.95, 0.9, \text{ and } 0.8$. Plot predictor coefficients as a function of time along with the true coefficients for each λ , and discuss your observations. Also compare your results with those in Problem 10.4.
- (c) Repeat each simulation above 1000 times to get corresponding learning curves, which are obtained by averaging respective squared errors $|e(n)|^2$. Plot these curves and compare their steady-state value with the theoretical MSE.

- 10.22** Consider a system identification problem where we observe the input $x(n)$ and the noisy output $y(n) = y_o(n) + v(n)$, for $0 \leq n \leq N - 1$. The unknown system is specified by the system function

$$H_o(z) = \frac{0.0675 + 0.1349z^{-1} + 0.0675z^{-2}}{1 - 1.1430z^{-1} + 0.4128z^{-2}}$$

and $x(n) \sim \text{WGN}(0, 1)$, $v(n) \sim \text{WGN}(0, 0.01)$, and $N = 300$.

- (a) Model the unknown system using an LS FIR filter, with $M = 15$ coefficients, using the no-windowing method. Compute the total LSE E_{ls} in the interval $n_0 \leq n \leq N - 1$ for $n_0 = 20$.
- (b) Repeat part (a) for $0 \leq n \leq n_0 - 1$ (do not compute E_{ls}). Use the vector $\mathbf{c}(n_0)$ and the matrix $\mathbf{P}(n_0) = \hat{\mathbf{R}}^{-1}(n_0)$ to initialize the CRLS algorithm. Compute the total errors $E_{\text{apr}} = \sum_{n=n_0}^{N-1} e^2(n)$ and $E_{\text{apost}} = \sum_{n=n_0}^{N-1} \varepsilon^2(n)$ by running the CRLS for $n_0 \leq n \leq N - 1$.
- (c) Order the quantities E_{ls} , E_{apr} , E_{apost} by size and justify the resulting ordering.

- 10.23** Prove Equation (10.5.25) using the identity $\det(\mathbf{I}_1 + \mathbf{AB}) = \det(\mathbf{I}_2 + \mathbf{BA})$, where identity matrices \mathbf{I}_1 and \mathbf{I}_2 and matrices \mathbf{A} and \mathbf{B} have compatible dimensions. Hint: Put (10.5.7) in the form $\mathbf{I}_1 + \mathbf{AB}$.

- 10.24** Derive the normal equations that correspond to the minimization of the cost function (10.5.36), and show that for $\delta = 0$ they are reduced to the standard set (10.5.2) of normal equations. For the situation described in Problem 10.22, run the CRLS algorithm for various values of δ and determine the range of values that provides acceptable performance.

- 10.25** Modify the CRLS algorithm in Table 10.6 so that its coefficients satisfy the linear-phase constraint $\mathbf{c} = \mathbf{J}\mathbf{c}^*$. For simplicity, assume that $M = 2L$; that is, the filter has an even number of coefficients.

- 10.26** Following the approach used in Section 7.1.5 to develop the structure shown in Figure 7.1, derive a similar structure based on the Cholesky (*not* the LDL^H) decomposition.

- 10.27** Show that the partitioning (10.7.3) of $\hat{\mathbf{R}}_{m+1}(n)$ to obtain the same partitioning structure as (10.7.2) is possible only if we apply the prewindowing condition $\mathbf{x}_m(-1) = \mathbf{0}$. What is the form of the partitioning if we abandon the prewindowing assumption?

- 10.28** Derive the normal equations and the LSE formulas given in Table 10.11 for the FLP and the BLP methods.

10.29 Derive the FLP and BLP a priori and a posteriori updating formulas given in Table 10.12.

615

PROBLEMS

10.30 Modify Table 10.14 for the FAEST algorithm, to obtain a table for the FTF algorithm, and write a MATLAB function for its implementation. Test the obtained function, using the equalization experiment in Example 10.5.2.

10.31 If we wish to initialize the fast RLS algorithms (fast Kalman, FAEST, and FTF) using an exact method, we need to collect a set of data $\{\mathbf{x}(n), \mathbf{y}(n)\}_0^{n_0}$ for any $n_0 > M$.

- Identify the quantities needed to start the FAEST algorithm at $n = n_0$. Form the normal equations and use the LDL^H decomposition method to determine these quantities.
- Write a MATLAB function `faestexact.m` that implements the FAEST algorithm using the exact initialization procedure described in part (a).
- Use the functions `faest.m` and `faestexact.m` to compare the two different initialization approaches for the FAEST algorithm in the context of the equalization experiment in Example 10.5.2. Use $n_0 = 1.5M$ and $n_0 = 3M$. Which value of δ gives results closest to the exact initialization method?

10.32 Using the order-recursive approach introduced in Section 7.3.1, develop an order-recursive algorithm for the solution of the normal equations (10.5.2). Note: In Section 7.3.1 we could not develop a closed-form algorithm because some recursions required the quantities $\mathbf{b}_m(n-1)$ and $E_m^b(n-1)$. Here we can avoid this problem by using time recursions.

10.33 In this problem we discuss several quantities that can serve to warn of ill behavior in fast RLS algorithms for FIR filters.

- Show that the variable

$$\eta_m(n) \triangleq \frac{\alpha_{m+1}(n)}{\alpha_m(n)} = \frac{\lambda E_m^b(n-1)}{E_m^b(n)} = 1 - g_{m+1}^{(m+1)}(n)e_m^{b*}(n)$$

satisfies the condition $0 \leq \eta_m(n) \leq 1$.

- Prove the relations

$$\alpha_m(n) = \lambda^m \frac{\det \hat{\mathbf{R}}_m(n-1)}{\det \hat{\mathbf{R}}_m(n)} \quad E_m^f(n) = \frac{\det \hat{\mathbf{R}}_{m+1}(n)}{\det \hat{\mathbf{R}}_m(n-1)} \quad E_m^b(n) = \frac{\det \hat{\mathbf{R}}_{m+1}(n)}{\det \hat{\mathbf{R}}_m(n)}$$

- Show that

$$\alpha_m(n) = \lambda^m \frac{E_m^b(n)}{E_m^f(n)}$$

and use it to explain why the quantity $\eta_m^\alpha(n) = E_m^f(n) - \lambda^m E_m^b(n)$ can be used as a warning variable.

- Explain how the quantities

$$\eta_{\bar{g}}(n) \triangleq \bar{g}_{M+1}^{(M+1)}(n) - \frac{e^b(n)}{\lambda E^b(n-1)}$$

and $\eta_b(n) \triangleq e^b(n) - \lambda E^b(n-1) \bar{g}_{M+1}^{(M+1)}(n)$

can be used as warning variables.

10.34 When the desired response is $y(j) = \delta(j-k)$, that is, a spike at $j = k$, $0 \leq k \leq n$, the LS filter $\mathbf{c}_m^{(k)}$ is known as a *spiking filter* or as an LS inverse filter (see Section 8.3).

- Determine the normal equations and the LSE $E_m^{(k)}(n)$ for the LS filter $\mathbf{c}_m^{(k)}$.
- Show that $\mathbf{c}_m^{(n)} = \mathbf{g}_m(n)$ and $E_m^{(n)}(n) = \alpha_m(n)$ and explain their meanings.
- Use the interpretation $\alpha_m(n) = E_m^{(n)}(n)$ to show that $0 \leq \alpha_m(n) \leq 1$.
- Show that $\mathbf{a}_m(n) = \sum_{k=0}^n \mathbf{c}_m^{(k)}(n-1)x(k)$ and explain its meaning.

- 10.35** Derive Equations (10.7.33) through (10.7.35) for the a posteriori LS lattice-ladder structure, shown in Figure 10.38, starting with the partitionings (10.7.1) and the matrix by inversion by partitioning relations (10.7.7) and (10.7.8).

Prove relations (10.7.45) and (10.7.46) for the updating of the ladder partial correlation coefficient $\beta_m^c(n)$.

- 10.37** In Section 7.3.1 we derived order-recursive relations for the FLP, BLP, and FIR filtering MMSEs.

- (a) Following the derivation of (7.3.36) and (7.3.37), derive similar order-recursive relations for $E_m^f(n)$ and $E_m^b(n)$.
- (b) Show that we can obtain a complete LS lattice-ladder algorithm by replacing, in Table 10.15, the time-recursive updatings of $E_m^f(n)$ and $E_m^b(n)$ with the obtained order-recursive relations.
- (c) Write a MATLAB function for this algorithm, and verify it by using the equalization experiment in Example 10.5.2.

- 10.38** Derive the equations for the a priori RLS lattice-ladder algorithm given in Table 10.16, and write a MATLAB function for its implementation. Test the function by using the equalization experiment in Example 10.5.2.

- 10.39** Derive the equations for the a priori RLS lattice-ladder algorithm with error feedback (see Table 10.7), and write a MATLAB function for its implementation. Test the function by using the equalization experiment in Example 10.5.2.

- 10.40** Derive the equations for the a posteriori RLS lattice-ladder algorithm with error feedback (Ling et al. 1986) and write a MATLAB function for its implementation. Test the function by using the equalization experiment in Example 10.5.2.

- 10.41** The a posteriori and the a priori RLS lattice-ladder algorithms need the conversion factor $\alpha_m(n)$ because the updating of the quantities $E_m^f(n)$, $E_m^b(n)$, $\beta_m(n)$, and $\beta_m^c(n)$ requires both the a priori and a posteriori errors. Derive a double (a priori and a posteriori) lattice-ladder RLS filter that avoids the use of the conversion factor by updating both the a priori and the a posteriori prediction and filtering errors.

- 10.42** Program the RLS Givens lattice-ladder filter with square roots (see Table 10.18), and study its use in the adaptive equalization experiment of Example 10.5.2.

- 10.43** Derive the formulas and program the RLS Givens lattice-ladder filter without square roots (see Table 10.18), and study its use in the adaptive equalization experiment of Example 10.5.2.

- 10.44** In this problem we discuss the derivation of the normalized lattice-ladder RLS algorithm, which uses a smaller number of time and order updating recursions and has better numerical behavior due to the normalization of its variables. *Note:* You may find useful the discussion in Carayannis et al. (1986).

- (a) Define the energy and angle normalized variables

$$\bar{e}_m^f(n) = \frac{\varepsilon_m^f(n)}{\sqrt{\alpha_m(n)}\sqrt{E_m^f(n)}} \quad \bar{e}_m^b(n) = \frac{\varepsilon_m^b(n)}{\sqrt{\alpha_m(n)}\sqrt{E_m^b(n)}} \quad \bar{e}_m(n) = \frac{\varepsilon_m(n)}{\sqrt{\alpha_m(n)}\sqrt{E_m(n)}}$$

$$\bar{k}_m(n) = \frac{\beta_m(n)}{\sqrt{E_m^f(n)}\sqrt{E_m^b(n-1)}} \quad \bar{k}_m^c(n) = \frac{\beta_m^c(n)}{\sqrt{E_m(n)}\sqrt{E_m^b(n)}}$$

and show that the normalized errors and the partial correlation coefficients $\bar{k}_m(n)$ and $\bar{k}_m^c(n)$ have magnitude less than 1.

(b) Derive the following normalized lattice-ladder RLS algorithm:

$$E_0^f(-1) = E_0(-1) = \delta > 0$$

For $n = 0, 1, 2, \dots$

$$E_0^f(n) = \lambda E_0^f(n-1) + |x(n)|^2, \quad E_0(n) = \lambda E_0(n-1) + |y(n)|^2$$

$$\bar{e}_0^f(n) = \bar{e}_0^b(n) = \frac{x(n)}{\sqrt{E_0^f(n)}}, \quad \bar{e}_0(n) = \frac{y(n)}{\sqrt{E_0(n)}}$$

For $m = 0$ to $M - 1$

$$\begin{aligned} \bar{k}_m(n) &= \sqrt{1 - |\bar{e}_m^f(n)|^2} \sqrt{1 - |\bar{e}_m^b(n-1)|^2} \bar{k}_m(n-1) + \bar{e}_m^{f*}(n) \bar{e}_m^b(n-1) \\ \bar{e}_{m+1}^f(n) &= \left(\sqrt{1 - |\bar{e}_m^b(n-1)|^2} \sqrt{1 - |\bar{k}_m(n)|^2} \right)^{-1} [\bar{e}_m^f(n) - \bar{k}_m(n) \bar{e}_m^b(n-1)] \\ \bar{e}_{m+1}^b(n) &= \left(\sqrt{1 - |\bar{e}_m^f(n)|^2} \sqrt{1 - |\bar{k}_m(n)|^2} \right)^{-1} [\bar{e}_m^b(n-1) - \bar{k}_m(n) \bar{e}_m^f(n)] \\ \bar{k}_m^c(n) &= \sqrt{1 - |\bar{e}_m(n)|^2} \sqrt{1 - |\bar{e}_m^b(n)|^2} \bar{k}_m^c(n-1) + \bar{e}_m^{*}(n) \bar{e}_m^b(n) \\ \bar{e}_{m+1}^c(n) &= \left(\sqrt{1 - |\bar{e}_m^b(n)|^2} \sqrt{1 - |\bar{k}_m^c(n)|^2} \right)^{-1} [\bar{e}_m(n) - \bar{k}_m^c(n) \bar{e}_m^b(n)] \end{aligned}$$

(c) Write a MATLAB function to implement the derived algorithm, and test its validity by using the equalization experiment in Example 10.5.2.

10.45 Prove (10.6.46) by direct manipulation of (10.6.35).

10.46 Derive the formulas for the QR-RLS lattice predictor (see Table 10.18), using the approach introduced in Section 10.6.3 (Yang and Böhme 1992).

10.47 Demonstrate how the systolic array in Figure 10.55, which is an extension of the systolic array structure shown in Figure 10.36, can be used to determine the LS error $e(n)$ and the LS

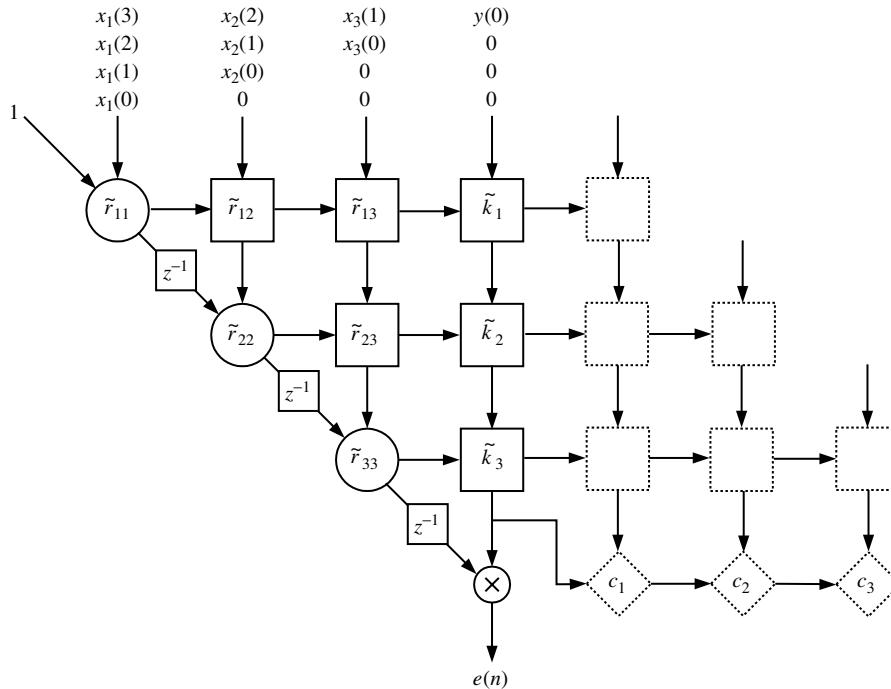


FIGURE 10.55

Systolic array implementation of the extended QR-RLS algorithm.

coefficient vector $\mathbf{c}(n)$. Determine the functions to be assigned to the dotted-line computing elements and the inputs with which they should be supplied.

10.48 The implementation of adaptive filters using multiprocessing involves the following steps: (1) partitioning of the overall computational job into individual tasks, (2) allocation of computational and communications tasks to the processors, and (3) synchronization and control of the processors. Figure 10.56 shows a cascade multiprocessing architecture used for adaptive filtering. To avoid *latency* (i.e., a delay between the filter's input and output that is larger than the sampling interval), each processor should complete its task in time less than the sampling period and use results computed by the preceding processor and the scalar computational unit at the previous sampling interval. This is accomplished by the unit delays inserted between the processors.

- (a) Explain why the fast Kalman algorithm, given in Table 10.13, does not satisfy the multiprocessing requirements.
- (b) Prove the formulas

$$\mathbf{b}(n) = \frac{\mathbf{b}(n-1) - \mathbf{g}_{M+1}^{[M]}(n)e^{\mathbf{b}^*(n)}}{1 - g_{M+1}^{(M+1)}(n)e^{\mathbf{b}^*(n)}} \quad (k)$$

$$\mathbf{g}(n) = \mathbf{g}_{M+1}^{[M]}(n) - g_{M+1}^{(M+1)}(n)\mathbf{b}(n) \quad (l)$$

and show that they can be used to replace formulas (g) and (h) in Table 10.13.

- (c) Rearrange the formulas in Table 10.13 as follows: (e), (k), (l), (a), (b), (c), (d), (f). Replace n by $n - 1$ in (e), (l), and (k). Show that the resulting algorithm complies with the multiprocessing architecture shown in Figure 10.56.
- (d) Draw a block diagram of a single multiprocessing section that can be used in the multiprocessing architecture shown in Figure 10.56. Each processor in Figure 10.56 can be assigned to execute one or more of the designed sections. Note: You may find useful the discussions in Lawrence and Tewksbury (1983) and in Manolakis and Patel (1992).
- (e) Figure 10.57 shows an alternative implementation of a multiprocessing section that can be used in the architecture of Figure 10.56. Identify the input-output quantities and the various multiplier factors.

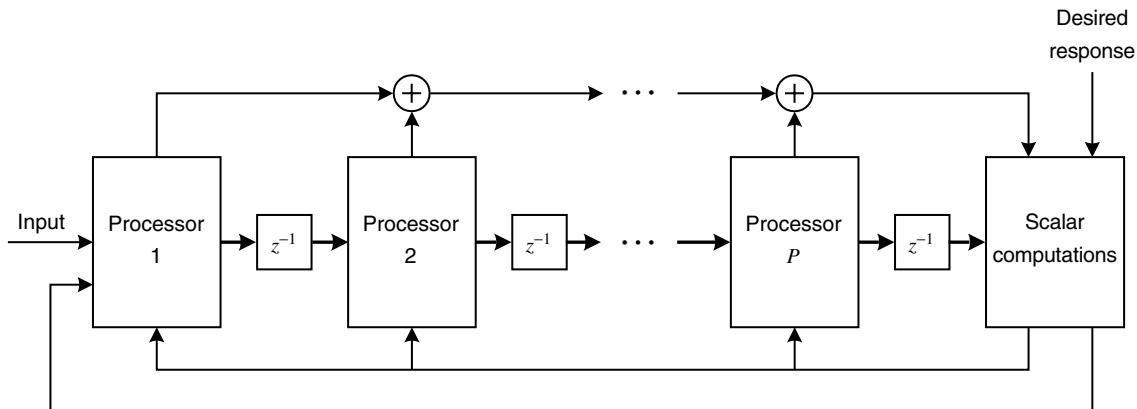
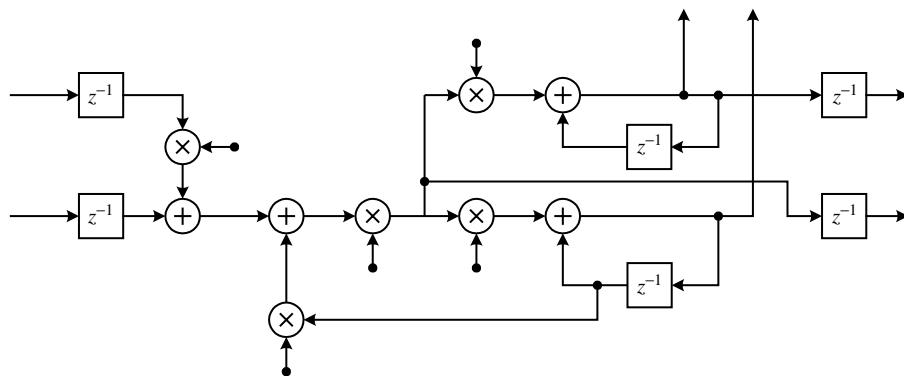


FIGURE 10.56

Cascade multiprocessing architecture for the implementation of FIR adaptive filters.

10.49 Show that the LMS algorithm in Table 10.13 satisfies the multiprocessing architecture in Figure 10.56.

10.50 Show that the a priori RLS linear prediction lattice (i.e., without the ladder part) algorithm with error feedback complies with the multiprocessing architecture of Figure 10.56. Explain

**FIGURE 10.57**

Section for the multiprocessing implementation of the fast Kalman algorithm.

why the addition of the ladder part violates the multiprocessing architecture. Can we rectify these violations? (See Lawrence and Tewksbury 1983.)

10.51 The fixed-length sliding window RLS algorithm is given in (10.8.4) through (10.8.10).

- Derive the above equations of this algorithm (see Manolakis et al. 1987).
- Develop a MATLAB function to implement the algorithm

[c, e] = slwrls(x, y, L, delta, M, c0);

where L is the fixed length of the window.

- Generate 500 samples of the following nonstationary process

$$x(n) = \begin{cases} w(n) + 0.95x(n-1) - 0.9x(n-2) & 0 \leq n < 200 \\ w(n) - 0.95x(n-1) - 0.9x(n-2) & 200 \leq n < 300 \\ w(n) + 0.95x(n-1) - 0.9x(n-2) & n \geq 300 \end{cases}$$

where $w(n)$ is a zero-mean, unit-variance white noise process. We want to obtain a second-order linear predictor using adaptive algorithms. Use the sliding window RLS algorithm on the data and choose $L = 50$ and 100 . Obtain plots of the filter coefficients and mean square error.

- Now use the growing memory RLS algorithm by choosing $\lambda = 1$. Compare your results with the sliding-window RLS algorithm.
- Finally, use the exponentially growing memory RLS by choosing $\lambda = (L-1)/(L+1)$ that produces the same MSE. Compare your results.

10.52 Consider the definition of the MSD $\mathcal{D}(n)$ in (10.2.29) and that of the trace of a matrix (A.2.16).

- Show that $D(n) = \text{tr}\{\Phi(n)\}$, where $\Phi(n)$ is the correlation matrix of $\tilde{c}(n)$.
- For the evolution of the correlation matrix in (10.8.58), show that

$$\mathcal{D}(\infty) \simeq \mu M \sigma_v^2 + \frac{\text{tr}(\mathbf{R}^{-1} \mathbf{R}_\psi)}{4\mu}$$

10.53 Consider the analysis model given in Figure 10.42. Let the parameters of this model be as follows:

$$\mathbf{c}_o(n) \text{ model parameters: } \mathbf{c}_o(0) = \begin{bmatrix} 0.9 \\ -0.8 \end{bmatrix} \quad M = 2 \quad \rho = 0.95$$

$$\boldsymbol{\psi}(n) \sim \text{WGN}(\mathbf{0}, \mathbf{R}_\psi) \quad \mathbf{R}_\psi = (0.01)^2 \mathbf{I}$$

Signal $\mathbf{x}(n)$ parameters: $\mathbf{x}(n) \sim \text{WGN}(\mathbf{0}, \mathbf{R}) \quad \mathbf{R} = \mathbf{I}$

Noise $v(n)$ parameters: $v(n) \sim \text{WGN}(0, \sigma_v^2) \quad \sigma_v = 0.1$

Simulate the system, using three values of μ that show slow, matched, and optimum adaptations of the LMS algorithm.

- Obtain the tracking plots similar to Figure 10.43 for each of the above three adaptations.
- Obtain the learning curve plots similar to Figure 10.44 for each of the above three adaptations.

- 10.54** Consider the analysis model given in Figure 10.42. Let the parameters of this model be as follows

$$\mathbf{c}_o(n) \text{ model parameters: } \mathbf{c}_o(0) = \begin{bmatrix} 0.9 \\ -0.8 \end{bmatrix} \quad M = 2 \quad \rho = 0.95$$

$$\boldsymbol{\psi}(n) \sim \text{WGN}(\mathbf{0}, \mathbf{R}_\psi) \quad \mathbf{R}_\psi = (0.01)^2 \mathbf{I}$$

Signal $\mathbf{x}(n)$ parameters: $\mathbf{x}(n) \sim \text{WGN}(\mathbf{0}, \mathbf{R}) \quad \mathbf{R} = \mathbf{I}$

Noise $v(n)$ parameters: $v(n) \sim \text{WGN}(0, \sigma_v^2) \quad \sigma_v = 0.1$

Simulate the system, using three values of μ that show slow, matched, and optimum adaptations of the RLS algorithm.

- Obtain the tracking plots similar to Figure 10.49 for each of the above three adaptations.
- Obtain the learning curve plots similar to Figure 10.50 for each of the above three adaptations.
- Compare your results with those obtained in Problem 10.53.

- 10.55** Consider the time-varying adaptive equalizer shown in Figure 10.58 in which the time variation of the channel impulse response is given by

$$h(n) = \rho h(n-1) + \sqrt{1-\rho} \eta(n)$$

with $\rho = 0.95 \quad \eta(n) \sim \text{WGN}(0, \sqrt{10}) \quad h(0) = 0.5$

Let the equalizer be a single-tap equalizer and $v(n) \sim \text{WGN}(0, 0.1)$.

- Simulate the system for three different adaptations; that is, choose μ for slow, matched, and fast adaptations of the LMS algorithm.
- Repeat part (a), using the RLS algorithm.

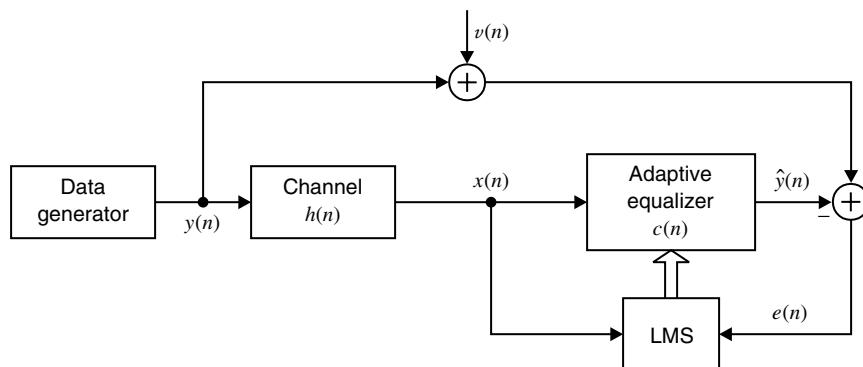


FIGURE 10.58

Adaptive channel equalizer system with time-varying channel in Problem 10.55.

Array Processing

The subject of array processing is concerned with the extraction of information from signals collected using an array of sensors. These signals propagate spatially through a medium, for example, air or water, and the resulting wavefront is sampled by the sensor array. The information of interest in the signal may be either the content of the signal itself (communications) or the location of the source or reflection that produces the signal (radar and sonar). In either case, the sensor array data must be processed to extract this useful information. The methods utilized in most cases are extensions of the statistical and adaptive signal processing techniques discussed in previous chapters, such as spectral estimation and optimum and adaptive filtering, extended to sensor array applications.

Sensor arrays are found in a wide range of applications, including radar, sonar, seismology, biomedicine, communications, astronomy, and imaging. Each of these individual fields contains a wealth of research into the various methods for the processing of array signals. Generally, the type of processing is dictated by the particular application. However, an underlying set of principles and techniques is common to a diverse set of applications. In this chapter, we focus on the fundamentals of array processing with emphasis on optimum and adaptive techniques. To simplify the discussion, we concentrate on linear arrays, where the sensors are located along a line. The extension of this material to other array configurations is fairly straightforward in most cases. The intent of this chapter is to first give the uninitiated reader some exposure to the basic principles of array processing and then apply adaptive processing techniques to the array processing problem. For a more detailed treatment of array processing methods, see Monzingo and Miller (1980), Hudson (1981), Compton (1988), and Johnson and Dudgeon (1993).

The chapter begins in Section 11.1 with a brief background in some array fundamentals, including spatially propagating signals, modulation and demodulation, and the array signal model. In Section 11.2, we introduce the concept of beamforming, that is, the spatial discrimination or filtering of signals collected with a sensor array. We look at conventional, that is, nonadaptive, beamforming and touch upon many of the common considerations for an array that affect its performance, for example, element spacing, resolution, and sidelobe levels. In Section 11.3, we look at the optimum beamformer, which is based on a priori knowledge of the data statistics. Within this framework, we discuss some of the specific aspects of adaptive processing that affect performance in Section 11.4. Then, in Section 11.5, we discuss adaptive array processing methods that estimate the statistics from actual data, first block-adaptive and then sample-by-sample adaptive methods. Section 11.6 discusses other adaptive array processing techniques that were born out of practical considerations for various applications. The determination of the angle of arrival of a spatial signal is the topic of Section 11.7. In Section 11.8, we give a brief description of space-time adaptive processing.

11.1 ARRAY FUNDAMENTALS

The information contained in a spatially propagating signal may be either the location of its source or the content of the signal itself. If we are interested in obtaining this information, we generally must deal with the presence of other, undesired signals. Much as a frequency-selective filter emphasizes signals at a certain frequency, we can choose to focus on signals from a particular direction. Clearly, this task can be accomplished by using a single sensor, provided that it has the ability to spatially discriminate; that is, it passes signals from certain directions while rejecting those from other directions. Such a single-sensor system, shown in Figure 11.1(a), is commonly found in communications and radar applications in which the signals are collected over a continuous spatial extent or *aperture* using a parabolic dish. The signals are reflected to the antenna in such a way that signals from the direction in which the dish is pointed are emphasized. The ability of a sensor to spatially discriminate, known as *directivity*, is governed by the shape and physical characteristics of its geometric structure. However, such a single-sensor system has several drawbacks. Since the sensor relies on mechanical pointing for directivity, it can extract and track signals from only one direction at a time; it cannot look in several directions simultaneously. Also, such a sensor cannot adapt its response, which would require physically changing the aperture, in order to reject potentially strong sources that may interfere with the extraction of the signals of interest.

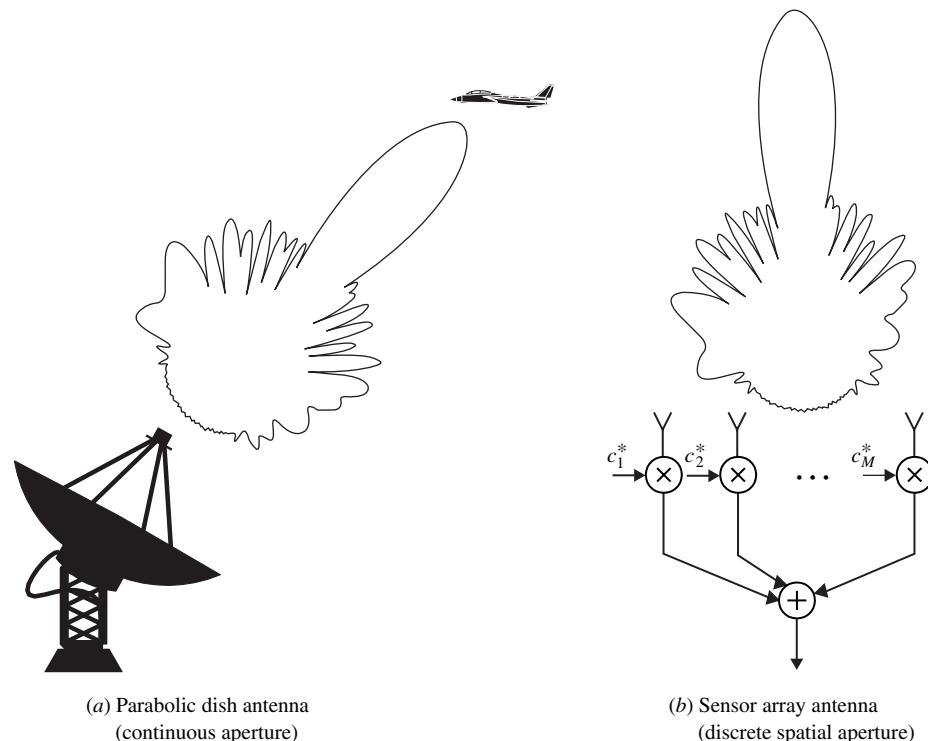


FIGURE 11.1
Comparison of a single, directive antenna with multiple sensors that make up an antenna array. In both cases, the response is designed to emphasize signals from a certain direction through spatial filtering, either continuous or discrete.

An array of sensors has the ability to overcome these shortcomings of a single sensor. Figure 11.1 (b) illustrates the use of a sensor array. The sensor array signals are combined in such a way that a particular direction is emphasized. However, the direction in which the

array is focused or pointed is almost independent of the orientation of the array. Therefore, the sensors can be combined in distinct, separate ways so as to emphasize different directions, all of which may contain signals of interest. Since the various weighted summations of the sensors simply amount to processing the same data in different ways, these multiple sources can be extracted simultaneously. Also arrays have the ability to adjust the overall rejection level in certain directions to overcome strong interference sources. In this section, we discuss some fundamentals of sensor arrays. First, we give a brief description of spatially propagating signals and the modulation and demodulation operations. Then we develop a signal model, first for an arbitrary array and then by simplifying to the case of a uniform linear array. In addition, we point out the interpretation of a sensor array as a mechanism for the spatial sampling of a spatially propagating signal.

11.1.1 Spatial Signals

In their most general form, spatial signals are signals that propagate through space. These signals originate from a source, travel through a propagation medium, say, air or water, and arrive at an array of sensors that spatially samples the waveform. A processor can then take the data collected by the sensor array and attempt to extract information about the source, based on certain characteristics of the propagating wave. Since space is three-dimensional, a spatial signal at a point specified by the vector \mathbf{r} can be represented either in Cartesian coordinates (x, y, z) or in spherical coordinates $(R, \phi_{az}, \theta_{el})$ as shown in Figure 11.2. Here, $R = \|\mathbf{r}\|$ represents range or the distance from the origin, and ϕ_{az} and θ_{el} are the azimuth and elevation angles, respectively.

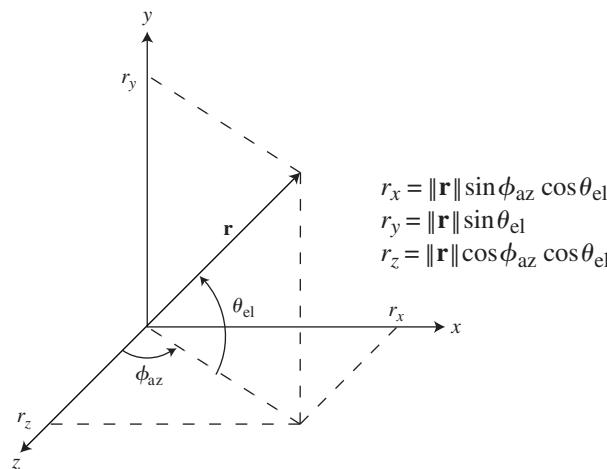


FIGURE 11.2
Three-dimensional space describing azimuth, elevation, and range.

The propagation of a spatial signal is governed by the solution to the wave equation. For electromagnetic propagating signals, the wave equation can be deduced from Maxwell's equations (Ishimaru 1990), while for sound waves the solution is governed by the basic laws of acoustics (Kino 1987; Jensen et al. 1994). However, in either case, for a propagating wave emanating from a source located at \mathbf{r}_0 , one solution is a single-frequency wave given by

$$s(t, \mathbf{r}) = \frac{A}{\|\mathbf{r} - \mathbf{r}_0\|^2} e^{j2\pi F_c \left(t - \frac{\|\mathbf{r} - \mathbf{r}_0\|}{c} \right)} \quad (11.1.1)$$

where A is the complex amplitude, F_c is the carrier frequency of the wave, and c is the speed of propagation of the wave. The speed of propagation is determined by the type of wave (electromagnetic or acoustic) and the propagation medium. For the purposes of this discussion, we ignore the singularity at the source (origin); that is, $s(t, \mathbf{r}_0) = \infty$. This equation suppresses the dependencies on ϕ_{az} and θ_{el} since the wave propagates radially from the source. At any point in space, the wave has the temporal frequency F_c . In (11.1.1) and for the remainder of this chapter, we will assume a lossless, nondispersive propagation medium, that is, a medium that does not attenuate the propagating signal further than predicted by the wave equation, and the propagation speed is uniform so that the wave travels according to (11.1.1). A dispersive medium adds a frequency dependence to the wave propagation (Jensen et al. 1994). Clearly, the signal travels in time where the spatial propagation is determined by the direct coupling between space and time in order to satisfy (11.1.1). We can then define the wavelength of the propagating wave as

$$\lambda = \frac{c}{F_c} \quad (11.1.2)$$

which is the distance traversed by the wave during one temporal period.

Two other simplifying assumptions will be made for the remainder of this chapter. First, the propagating signals are assumed to be produced by a point source; that is, the size of the source is small with respect to the distance between the source and the sensors that measure the signal. Second, the source is assumed to be in the “far field,” i.e., at a large distance from the sensor array, so that the spherically propagating wave can be reasonably approximated with a plane wave. This approximation again requires the source to be far removed from the array so that the curvature of the wave across the array is negligible. This concept is illustrated in Figure 11.3. Multiple sources are treated through superposition of the various spatial signals at the sensor array. Although each individual wave radiates from its source, generally the origin ($\mathbf{r} = \mathbf{0}$) is reserved for the position of the sensor array since this is the point in space at which the collection of waves is measured. For more details on spatially propagating signals, see Johnson and Dudgeon (1993).

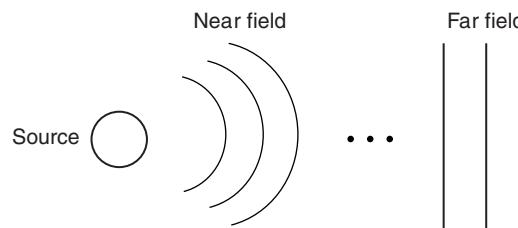


FIGURE 11.3
Plane wave approximation in the far field of the source.

Let us now consider placing a linear array in three-dimensional space in order to sense the propagating waves. The array consists of a series of elements located on a line with uniform spacing. Such an array is known as a *uniform linear array (ULA)*. For convenience, we choose the coordinate system for our three-dimensional space as in Figure 11.2 such that the ULA lies on the x axis. In addition, we have a wave originating from a point \mathbf{r} in this three-dimensional space that is located in the far field of the array such that the propagating signal can be approximated by a plane wave at the ULA. The plane wave impinges on the ULA as illustrated in Figure 11.4. As we will see, the differences in distance between the sensors determine the relative delays in arrival of the plane wave. The point from which the wave originates can be described by its distance from the origin $\|\mathbf{r}\|$ and its azimuth and elevation angles ϕ_{az} and θ_{el} , respectively. If the distance between elements of the ULA is d , then the difference in propagation distance between neighboring elements for a plane wave arriving from an azimuth ϕ_{az} and elevation θ_{el} is

$$d_x = \|\mathbf{r}\| \sin \phi_{az} \cos \theta_{el} \quad (11.1.3)$$

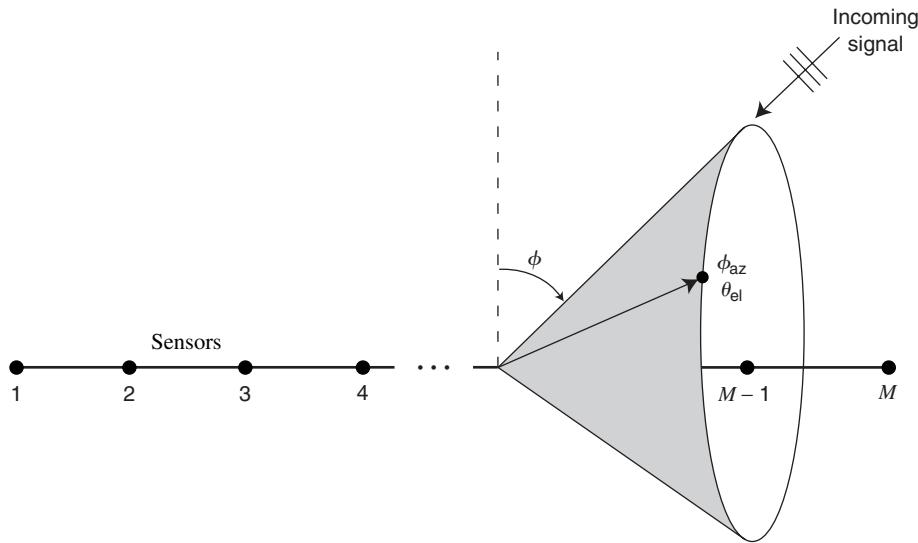


FIGURE 11.4

Cone angle ambiguity surface for a uniform linear array.

These differences in the propagation distance that the plane wave must travel to each of the sensors are a function of a general angle of arrival with respect to the ULA ϕ . If we consider the entire three-dimensional space, we note that equivalent delays are produced by any signal arriving from a cone about the ULA. Therefore, any signal arriving at the ULA on this surface has the same set of relative delays between the elements. This conical ambiguity surface is illustrated in Figure 11.4. For this reason, the angle of incidence to a linear array is commonly referred to as the *cone angle*, ϕ_{cone} . We see that the cone angle is related to the physical angles, azimuth and elevation defined in Figure 11.4, by

$$\sin \phi = \sin \phi_{\text{az}} \cos \theta_{\text{el}} \quad (11.1.4)$$

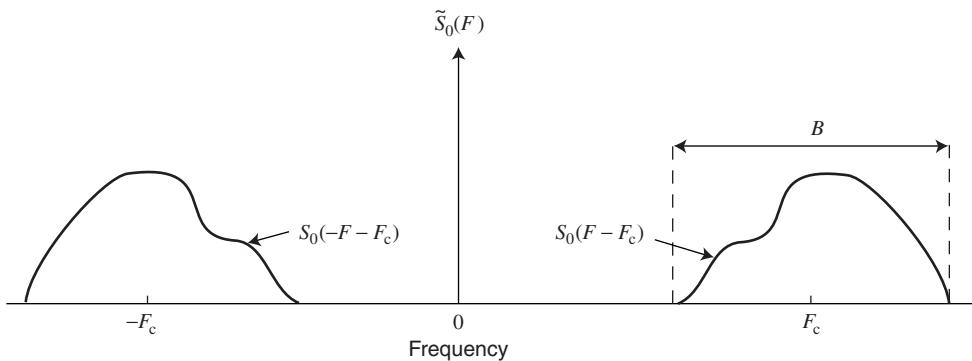
where $\phi = 90^\circ - \phi_{\text{cone}}$. In this manner, we can take a given azimuth and elevation pair and determine their corresponding cone angle. For the remainder of this chapter, we use the terms *angle of arrival* and simply *angle* interchangeably.

11.1.2 Modulation-Demodulation

The spatial propagation of signals was described by (11.1.1) using a propagation speed c and a center frequency F_c . For a general class of signals, the signal of interest $s_0(t)$ has a bandwidth that is a small fraction of the center frequency and is modulated up to the center frequency. Since the propagating wave then “carries” certain information to the receiving point in the form of a temporal signal, F_c is commonly referred to as the *carrier frequency*. The process of generating the signal $\tilde{s}_0(t)$ from $s_0(t)$ in order to transmit this information is accomplished by mixing the signal $s_0(t)$ with the carrier waveform $\cos 2\pi F_c t$ in an operation known as *modulation*. The propagating signal is then produced by a high-gain transmitter. The signal travels through space until it arrives at a sensor that measures the signal. Let us denote the received propagating signal as

$$\tilde{s}_0(t) = s_0(t) \cos 2\pi F_c t = \frac{1}{2} s_0(t) (e^{j2\pi F_c t} + e^{-j2\pi F_c t}) \quad (11.1.5)$$

where we say that the signal $s_0(t)$ is carried by the propagating waveform $\cos 2\pi F_c t$. The spectrum of $\tilde{s}_0(t)$ is made up of two components: the spectrum of the signal $s_0(t)$ shifted to F_c and shifted to $-F_c$ and reflected about $-F_c$. This spectrum $\tilde{S}_0(F)$ is shown in Figure 11.5. Here we indicate the signal $s_0(t)$ has a bandwidth B . The baseband signal $s_0(t)$,

**FIGURE 11.5**

Spectrum of a bandpass signal.

although originating as a real signal prior to modulation, has a nonsymmetric spectrum due to the asymmetric[†] spectral response of the propagation medium about frequency F_c . The received signal $\tilde{s}_0(t)$, though, is real-valued; that is, its spectrum exhibits even symmetry about $F = 0$. This fact is consistent with actual, physical signals that *are* real-valued as they are received and measured by a sensor.

The reception of spatially propagating signals with a sensor is only the beginning of the process of forming digital samples for both the in-phase and quadrature components of the sensor signal. Upon reception of the signal $\tilde{s}_0(t)$, the signal is mixed back to baseband in an operation known as *demodulation*. Included in the m th sensor signal is thermal noise due to the electronics of the sensor $w_m(t)$

$$\tilde{x}_m(t) = \tilde{s}_0(t) * h_m(t, \phi_s) + \tilde{w}_m(t) \quad (11.1.6)$$

where $h_m(t, \phi_s)$ is the combined temporal and spatial impulse response of the m th sensor. The angle ϕ_s is the direction from which $\tilde{s}_0(t)$ was received. In the case of an omnidirectional sensor with an equal response in all directions, the impulse response no longer is dependent on the angle of the signal. The demodulation process involves multiplying the received signal by $\cos 2\pi F_c t$ and $-\sin 2\pi F_c t$ to form both the in-phase and quadrature channels, respectively. Note the quadrature component is 90° out of phase of the in-phase component. The entire process is illustrated in Figure 11.6 for the m th sensor. This structure is referred to as the *receiver* of the m th channel.

Following demodulation, the signals in each channel are passed through a low-pass filter to remove any high-frequency components. The cutoff frequency of this low-pass filter determines the bandwidth of the receiver. Throughout this chapter, we assume a perfect or ideal low-pass filter, that is, a response of 1 in the passband and 0 in the stopband. In practice, the characteristics of the actual, nonideal low-pass filter can impact the performance of the resulting processor. Following the low-pass filtering operation, the signals in both the in-phase and quadrature channels are critically (Nyquist) sampled at the receiver bandwidth B . Oversampling at greater than the receiver bandwidth is also possible but is not considered here. More details on the signals at the various stages of the receiver, including the sensor impulse response, are covered in the next section on the array signal model. The output of the receiver is a complex-valued, discrete-time signal for the m th sensor with the in-phase and quadrature channels generating the real and imaginary portions of the signal

$$x_m(n) = x_m^{(I)}(n) + j x_m^{(Q)}(n) \quad (11.1.7)$$

For more details on the complex representation of bandpass signals, sampling, and the modulation and demodulation process, see Section 2.1. We should also mention that the sampling process in many systems is implemented using a technique commonly referred

[†]The asymmetry can arise from dispersive effects in the transmission medium.

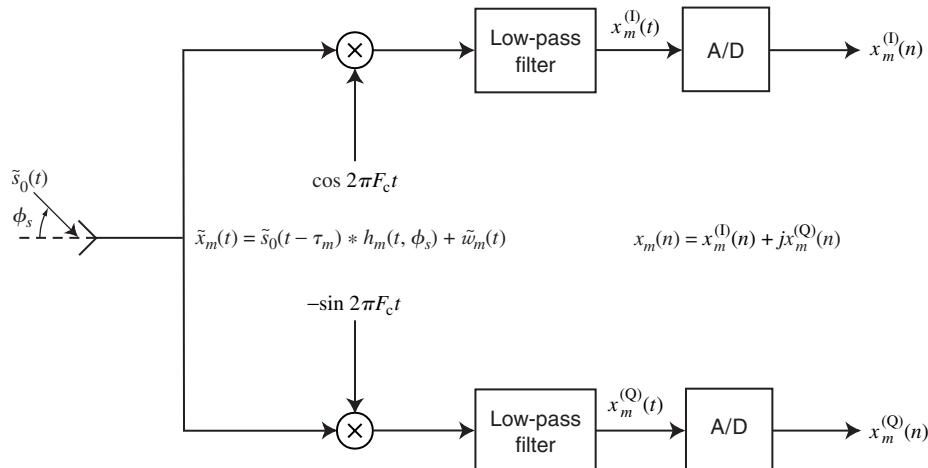


FIGURE 11.6

Block diagram of propagating signal arriving at a sensor with a receiver.

to as *digital in-phase/quadrature* or simply *digital IQ* (Rader 1984), rather than the more classical method outlined in this section. The method is more efficient as it only requires a single analog-to-digital (A/D) converter, though at a higher sampling rate.[†] See Rader (1984) for more details.

11.1.3 Array Signal Model

We begin by developing a model for a single spatial signal in noise received by a ULA. Consider a signal received by the ULA from an angle ϕ_s as in Figure 11.6. Each sensor receives the spatially propagating signal and converts its measured energy to voltage. This voltage signal is then part of the receiver channel from Figure 11.6. In addition, the receiver contains noise due to internal electronics known as *thermal noise*.[‡] Recall from (11.1.6) that $\tilde{x}_m(t)$ is the continuous-time signal in the m th sensor containing both the received carrier-modulated signals and thermal noise. The signal $x_m(t)$ is then obtained by demodulating $\tilde{x}_m(t)$ to baseband and low-pass filtering to the receiver bandwidth, while $x_m(n)$ is its discrete-time counterpart. Since the model is assumed to be linear, the extension to multiple signals, including interference sources, is straightforward.

The discrete-time signals from a ULA may be written as a vector containing the individual sensor signals

$$\mathbf{x}(n) = [x_1(n) \ x_2(n) \ \cdots \ x_M(n)]^T \quad (11.1.8)$$

where M is the total number of sensors. A single observation or measurement of this signal vector is known as an *array snapshot*. We begin by examining a single, carrier-modulated signal $\tilde{s}_0(t) = s_0(t) \cos 2\pi F_c t$ arriving from angle ϕ_s that is received by the m th sensor. We assume that the signal $s_0(t)$ has a deterministic amplitude and random, uniformly distributed phase. The \sim symbol is used to indicate that the signal is a passband or carrier-modulated signal. Here $s_0(t)$ is the baseband signal, and F_c is the carrier frequency. This signal is received by the m th sensor with a delay τ_m

$$\tilde{x}_m(t) = h_m(t, \phi_s) * \tilde{s}_0(t - \tau_m) + \tilde{w}_m(t) \quad (11.1.9)$$

[†]This digital IQ technique is very important for adaptive processing as I/Q channel mismatch can limit performance. One A/D converter avoids this source of mismatch.

[‡]Another source of noise may be external background noise. Many times this is assumed to be isotropic so that the overall noise signal is uncorrelated from sensor to sensor.

where $h_m(t, \phi)$ is the impulse response of the m th sensor as a function of both time and angle ϕ , and $\tilde{w}_m(t)$ is the sensor noise. Note that the relative delay at the m th sensor τ_m is also a function of ϕ_s . We have temporarily suppressed this dependence to simplify the notation. Usually, we set $\tau_1 = 0$, in which case the delays to the remaining sensors ($m = 2, 3, \dots, M$) are simply the differences in propagation time of $\tilde{s}_0(t)$ to these sensors with respect to the first sensor. The sensor signal can also be expressed in the frequency domain as

$$\begin{aligned}\tilde{X}_m(F) &= H_m(F, \phi_s) \tilde{s}_0(F) e^{-j2\pi F \tau_m} + \tilde{W}_m(F) \\ &= H_m(F, \phi_s) [S_0(F - F_c) + S_0^*(-F - F_c)] e^{-j2\pi F \tau_m} + \tilde{W}_m(F)\end{aligned}\quad (11.1.10)$$

by using (11.1.5) and taking the Fourier transform of (11.1.9). Following demodulation and ideal low-pass filtering of the signal from the m th sensor, as shown in Figure 11.6, the spectrum of the signal is

$$X_m(F) = H_m(F + F_c, \phi_s) S_0(F) e^{-j2\pi(F+F_c)\tau_m} + W_m(F) \quad (11.1.11)$$

where $X_m(F) = X_m^{(I)}(F) + jX_m^{(Q)}(F)$. The second term $S_0^*(-F - 2F_c)$ has been removed through the ideal low-pass filtering operation. This ideal low-pass filter has a value of unity across its passband so that $W_m(F) = \tilde{W}_m(F + F_c)$ for $|F| < B/2$.

We now make a critical, simplifying assumption: The bandwidth of $s_0(t)$ is small compared to the carrier frequency; this is known as the *narrowband assumption*. This assumption allows us to approximate the propagation delays of a particular signal between sensor elements with a phase shift. There are numerous variations on this assumption, but in general it holds for cases in which the signal bandwidth is less than some small percentage of the carrier frequency, say, less than 1 percent. The ratio of the signal bandwidth to the carrier frequency is referred to as the *fractional bandwidth*. However, the fractional bandwidth for which the narrowband assumption holds is strongly dependent on the length of the array and the strength of the received signals. Thus, we might want to consider the *time-bandwidth product (TBWP)*, which is the maximum amount of time for a spatial signal to propagate across the entire array ($\phi_s = \pm 90^\circ$). If $\text{TBWP} \ll 1$, then the narrowband assumption is valid. The effects of bandwidth on performance are treated in Section 11.4.2.

In addition to the narrowband assumption, we assume that the response of the sensor is constant across the bandwidth of the receiver, that is, $H_m(F + F_c, \phi_s) = H_m(F_c, \phi_s)$ for $|F| < B/2$. Thus, the spectrum in (11.1.11) simplifies to

$$X_m(F) = H_m(F_c, \phi_s) S_0(F) e^{-j2\pi F_c \tau_m} + W_m(F) \quad (11.1.12)$$

and the discrete-time signal model is obtained by sampling the inverse Fourier transform of (11.1.12)

$$x_m(n) = H_m(F_c, \phi_s) s_0(n) e^{-j2\pi F_c \tau_m} + w_m(n) \quad (11.1.13)$$

The term $w_m(n)$ corresponds to $W_m(F)$, the sensor thermal noise across the bandwidth of the receiver of the m th sensor. Furthermore, we assume that the power spectral density of this noise is flat across this bandwidth; that is, the discrete-time noise samples are uncorrelated. Also, the thermal noise in all the sensors is mutually uncorrelated.[†] If we further assume that each of the sensors in the array has an equal, omnidirectional response at frequency F_c , that is, $H_m(F_c, \phi_s) = H(F_c, \phi_s) = \text{constant}$, for $1 \leq m \leq M$, then the constant sensor responses can be absorbed into the signal term[‡]

$$s(n) = H(F_c) s_0(n) \quad (11.1.14)$$

[†]In actual systems, thermal noise samples are temporally correlated through the use of antialiasing filters prior to digital sampling. In addition, the thermal noise between sensors may be correlated due to mutual coupling of the sensors.

[‡]In many systems, we can compensate for differences in responses by processing signals from the sensors in an attempt to make their responses as similar as possible. When the data from the sensors are used to perform this compensation, the process is known as *adaptive channel matching*.

For the remainder of the chapter, we use the signal $s(n)$ as defined in (11.1.14). Using (11.1.8) and (11.1.13), we can then write the full-array discrete-time signal model as

$$\mathbf{x}(n) = \sqrt{M} \mathbf{v}(\phi_s) s(n) + \mathbf{w}(n) \quad (11.1.15)$$

where $\mathbf{v}(\phi) = \frac{1}{\sqrt{M}} [1 \ e^{-j2\pi F_c \tau_2(\phi)} \ \dots \ e^{-j2\pi F_c \tau_M(\phi)}]^T$ (11.1.16)

is the *array response vector*. We have chosen to measure all delays relative to the first sensor [$\tau_1(\phi) = 0$] and are now indicating the dependence of these delays on ϕ . We use the normalization of $1/\sqrt{M}$ for mathematical convenience so that the array response vector has unit norm, that is, $\|\mathbf{v}(\phi)\|^2 = \mathbf{v}^H(\phi) \mathbf{v}(\phi) = 1$. The factor is compensated for with the \sqrt{M} term in (11.1.15). The assumption of equal, omnidirectional sensor responses is necessary to simplify the analysis but should always be kept in mind when considering experimentally collected data for which this assumption certainly will not hold exactly. The other critical assumption made is that we have perfect knowledge of the array sensor locations, which also must be called into question for actual sensors and the data collected with them.

Up to this point, we have not made any assumptions about the form of the array, so that the array signal model we have developed holds for arbitrary arrays. Now we wish to focus our attention on the ULA, which is an array that has all its elements on a line with equal spacing between the elements. The ULA is shown in Figure 11.7, and the interelement spacing is denoted by d . Consider the single propagating signal that impinges on the ULA from an angle ϕ . Since all the elements are equally spaced, the spatial signal has a difference in propagation paths between any two successive sensors of $d \sin \phi$ that results in a time delay of

$$\tau(\phi) = \frac{d \sin \phi}{c} \quad (11.1.17)$$

where c is the rate of propagation of the signal. As a result, the delay to the m th element with respect to the first element in the array is

$$\tau_m(\phi) = (m - 1) \frac{d \sin \phi}{c} \quad (11.1.18)$$

and substituting into (11.1.16), we see the array response vector for a ULA is

$$\mathbf{v}(\phi) = \frac{1}{\sqrt{M}} [1 \ e^{-j2\pi[(d \sin \phi)/\lambda]} \ \dots \ e^{-j2\pi[(d \sin \phi)/\lambda](M-1)}]^T \quad (11.1.19)$$

since $F_c = c/\lambda$.

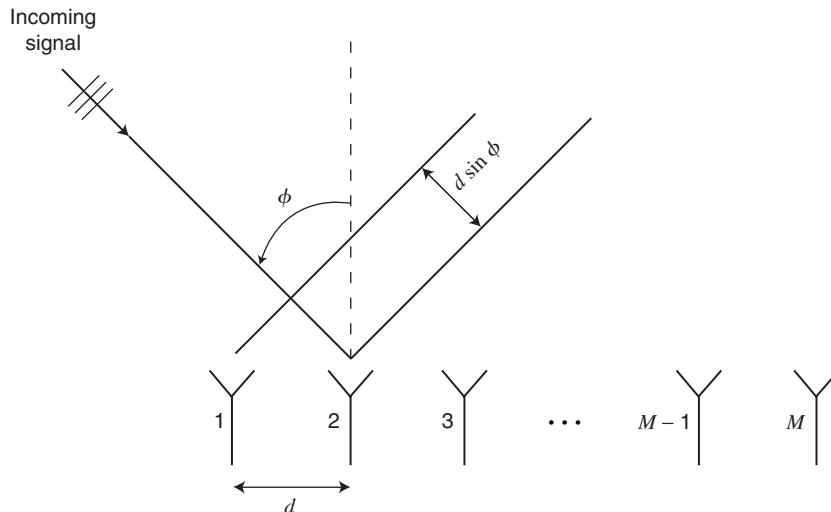


FIGURE 11.7

Plane wave impinging on a uniform linear array.

11.1.4 The Sensor Array: Spatial Sampling

In general, we can think of a sensor array as a mechanism for spatially sampling wavefronts propagating at a certain operating (carrier) frequency. Since in most instances the user either controls or has knowledge of the operating frequency, the sensor array provides a reliable means of interrogating the incoming wavefront for information. Similar to temporal sampling, the sensor array provides discrete (spatially sampled) data that can be used without loss of information, provided certain conditions are met. Namely, the sampling frequency must be high enough so as not to create spatial ambiguities or, in other words, to avoid spatial aliasing. The advantages of discrete-time processing and digital filtering have been well documented (Oppenheim and Schafer 1989; Proakis and Manolakis 1996). In the case of the spatial processing of signals, spatial sampling using an array provides the capability to change the characteristics of a discrete spatial filter, which is not possible for a continuous spatial aperture.

An arbitrary array performs its sampling in multiple dimensions and along a nonuniform grid so that it is difficult to compare to discrete-time sampling. However, a ULA has a direct correspondence to uniform, regular temporal sampling, since it samples uniformly in space on a linear axis. Thus, for a ULA we can talk about a *spatial sampling frequency* U_s defined by

$$U_s = \frac{1}{d} \quad (11.1.20)$$

where the *spatial sampling period* is determined by the interelement spacing d and is measured in cycles per unit of length (meters). Recall from (11.1.19) that the measurements made with a ULA on a narrowband signal correspond to a phase progression across the sensors determined by the angle of the incoming signal. As with temporal signals, the phase progression for uniform sampling is a consequence of the frequency; that is, consecutive samples of the same signal differ only by a phase shift of $e^{j2\pi F}$, where F is the frequency. In the case of a spatially propagating signal, this frequency is given by

$$U = \frac{\sin \phi}{\lambda} \quad (11.1.21)$$

which can be thought of as the *spatial frequency*. The *normalized spatial frequency* is then defined by

$$u \triangleq \frac{U}{U_s} = \frac{d \sin \phi}{\lambda} \quad (11.1.22)$$

Therefore, we can rewrite the array response vector from (11.1.19) in terms of the normalized spatial frequency as

$$\mathbf{v}(\phi) = \mathbf{v}(u) = \frac{1}{\sqrt{M}} [1 \ e^{-j2\pi u} \ \dots \ e^{-j2\pi u(M-1)}]^T \quad (11.1.23)$$

which we note is simply a Vandermonde vector (Strang 1998), that is, a vector whose elements are successive integer powers of the same number, in this case $e^{-j2\pi u}$.

The interelement spacing d is simply the spatial sampling interval, which is the inverse of the sampling frequency. Therefore, similar to Shannon's theorem for discrete-time sampling, there are certain requirements on the spatial sampling frequency to avoid aliasing. Since normalized frequencies are unambiguous for $-\frac{1}{2} \leq u < \frac{1}{2}$ and the full range of possible unambiguous angles is $-90^\circ \leq \phi \leq 90^\circ$, the sensor spacing must be

$$d \leq \frac{\lambda}{2} \quad (11.1.24)$$

to prevent spatial ambiguities. Since lowering the array spacing below this upper limit only provides redundant information and directly conflicts with the desire to have as much aperture as possible for a fixed number of sensors, we generally set $d = \lambda/2$. This tradeoff is further explored using beampatterns in the next section.

In many applications, the desired information to be extracted from an array of sensors is the content of a spatially propagating signal from a certain direction. The content may be a message contained in the signal, such as in communications applications, or merely the existence of the signal, as in radar and sonar. To this end, we want to linearly combine the signals from all the sensors in a manner, that is, with a certain weighting, so as to examine signals arriving from a specific angle. This operation, shown in Figure 11.8, is known as *beamforming* because the weighting process emphasizes signals from a particular direction while attenuating those from other directions and can be thought of as casting or forming a beam. In this sense, a beamformer is a spatial filter; and in the case of a ULA, it has a direct analogy to an FIR frequency-selective filter for temporal signals, as discussed in Section 1.5.1. Beamforming is commonly referred to as “electronic” steering since the weights are applied using electronic circuitry following the reception of the signal for the purpose of steering the array in a particular direction.[†] This can be contrasted with mechanical steering, in which the antenna is physically pointed in the direction of interest. For a complete tutorial on beamforming see Van Veen and Buckley (1988, 1998).

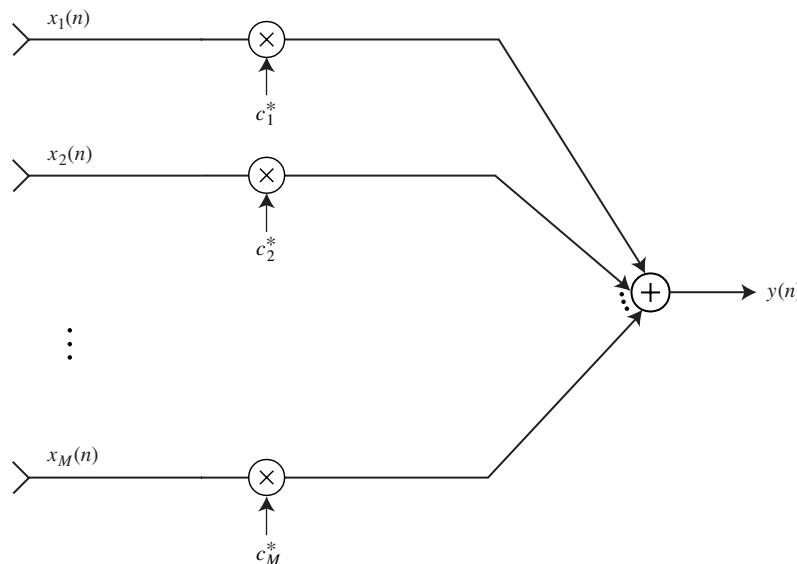


FIGURE 11.8
Beamforming operation.

In its most general form, a beamformer produces its output by forming a weighted combination of signals from the M elements of the sensor array, that is,

$$y(n) = \sum_{m=1}^M c_m^* x_m(n) = \mathbf{c}^H \mathbf{x}(n) \quad (11.2.1)$$

where

$$\mathbf{c} = [c_1 \ c_2 \ \cdots \ c_M]^T \quad (11.2.2)$$

is the column vector of beamforming weights. The beamforming operation for an M element array is illustrated in Figure 11.8.

[†] In general, performance does degrade as the angle to which the array is steered approaches $\phi = -90^\circ$ or $\phi = 90^\circ$. Although the array is optimized at broadside ($\phi = 0^\circ$), it certainly can steer over a wide range of angles about broadside for which performance degradation is minimal.

Beam response

A standard tool for analyzing the performance of a beamformer is the response for a given weight vector \mathbf{c} as a function of angle ϕ , known as the *beam response*. This angular response is computed by applying the beamformer \mathbf{c} to a set of array response vectors from all possible angles, that is, $-90^\circ \leq \phi < 90^\circ$,

$$C(\phi) = \mathbf{c}^H \mathbf{v}(\phi) \quad (11.2.3)$$

Typically, in evaluating a beamformer, we look at the quantity $|C(\phi)|^2$, which is known as the *beampattern*. Alternatively, the beampattern can be computed as a function of normalized spatial frequency u from (11.1.22). For a ULA with $\lambda/2$ element spacing, the beampattern as a function of u can be efficiently computed using the FFT for $-\frac{1}{2} \leq u < \frac{1}{2}$ at points separated by $1/N_{\text{fft}}$ where $N_{\text{fft}} \geq M$ is the FFT size. Thus, a beampattern can be computed in MATLAB with the command `C=fftshift(fft(c,N_fft))/sqrt(M)`, where the FFT size is selected to display the desired level of detail. To compute the corresponding angles of the beampattern, we can simply convert spatial frequency to angle as

$$\phi = \arcsin \frac{\lambda}{d} u \quad (11.2.4)$$

A sample beampattern for a 16-element uniform array with uniform weighting ($c_m = 1/\sqrt{M}$) is shown in Figure 11.9, which is plotted on a logarithmic scale in decibels. The large mainlobe is centered at $\phi = 0^\circ$, the direction in which the array is steered. Also notice the unusual sidelobe structure created by the nonlinear relationship between angle and spatial frequency in (11.2.4) at angles away from broadside ($\phi = 0^\circ$).

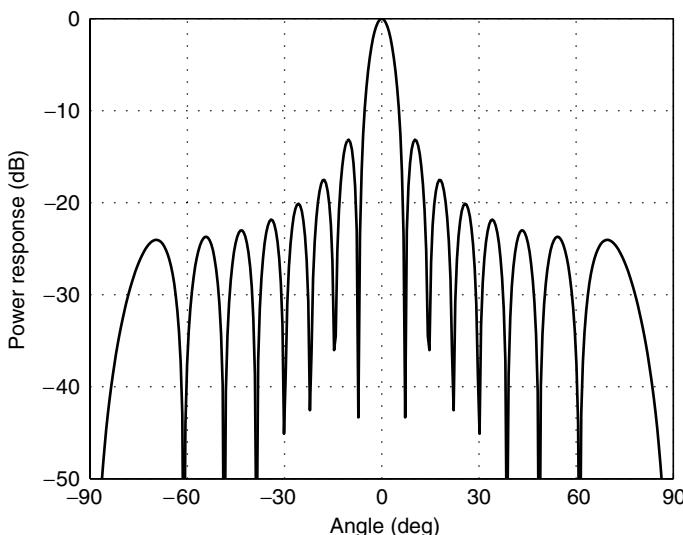


FIGURE 11.9

A sample beampattern of a spatial matched filter for an $M = 16$ element ULA steered to $\phi = 0^\circ$.

Important note. The beampattern is the spatial frequency response of a given beamformer. It should not be confused with the *steered response*, which is the response of the array to a certain set of spatial signals impinging on the array as we steer the array to all possible angles. Since this operation corresponds to measuring the power as a function of spatial frequency or angle, the steered response might be better defined as the *spatial power spectrum*

$$R(\phi) = E\{|\mathbf{c}^H(\phi)\mathbf{x}(n)|^2\} \quad (11.2.5)$$

where the choice of the beamformer $\mathbf{c}(\phi)$ determines the type of spatial spectrum, say, conventional or minimum-variance. Various spectrum estimation techniques were discussed in Chapters 5 and 9, several of which can be extended for the estimation of the spatial spectrum from measurements in practical applications. One interpretation of the estimation of the spectrum was made as a bank of frequency-selective filters at the frequencies at which the spectrum is computed. Similarly, the computation of the spatial spectrum can be thought of as the output of a bank of beamformers steered to the angles at which the spatial spectrum is computed.

Output signal-to-noise ratio

We now look at the signal-to-noise ratio (SNR) of the beamformer output and determine the improvement in SNR with respect to each element, known as the *beamforming gain*. Let us consider the signal model for a ULA from (11.1.15), which consists of a signal of interest arriving from an angle ϕ_s and thermal sensor noise $\mathbf{w}(n)$. The beamformer or spatial filter \mathbf{c} is applied to the array signal $\mathbf{x}(n)$ as

$$y(n) = \mathbf{c}^H \mathbf{x}(n) = \sqrt{M} \mathbf{c}^H \mathbf{v}(\phi_s) s(n) + \bar{w}(n) \quad (11.2.6)$$

where $\bar{w}(n) = \mathbf{c}^H \mathbf{w}(n)$ is the noise at the beamformer output and is also temporally uncorrelated. The beamformer output power is

$$P_y = E\{|y(n)|^2\} = \mathbf{c}^H \mathbf{R}_x \mathbf{c} \quad (11.2.7)$$

where

$$\mathbf{R}_x = E\{\mathbf{x}(n)\mathbf{x}^H(n)\} \quad (11.2.8)$$

is the correlation matrix of the array signal $\mathbf{x}(n)$. Recall from (11.1.15) and (11.1.23) that the signal for the m th element is given by

$$x_m(n) = e^{-j2\pi(m-1)u_s} s(n) + w_m(n) \quad (11.2.9)$$

where u_s is the normalized spatial frequency of the array signal produced by $s(n)$. The signal $s(n)$ is the signal of interest within a single sensor including the sensor response $H_m(F_c)$ from (11.1.14). Therefore, the signal-to-noise ratio in each element is given by

$$\text{SNR}_{\text{elem}} \triangleq \frac{\sigma_s^2}{\sigma_w^2} = \frac{|e^{-j2\pi(m-1)u_s} s(n)|^2}{E\{|w_m(n)|^2\}} \quad (11.2.10)$$

where $\sigma_s^2 = E\{|s(n)|^2\}$ and $\sigma_w^2 = E\{|w_m(n)|^2\}$ are the element level signal and noise powers, respectively. Recall that the signal $s(n)$ has a deterministic amplitude and random phase. We assume that all the elements have equal noise power σ_w^2 so that the SNR does not vary from element to element. This SNR_{elem} is commonly referred to as the *element level SNR* or the *SNR per element*.

Now if we consider the signals at the output of the beamformer, the signal and noise powers are given by

$$P_s = E\{|\sqrt{M}[\mathbf{c}^H \mathbf{v}(\phi_s)]s(n)|^2\} = M\sigma_s^2 |\mathbf{c}^H \mathbf{v}(\phi_s)|^2 \quad (11.2.11)$$

$$P_n = E\{|\mathbf{c}^H \mathbf{w}(n)|^2\} = \mathbf{c}^H \mathbf{R}_n \mathbf{c} = \|\mathbf{c}\|^2 \sigma_w^2 \quad (11.2.12)$$

because $\mathbf{R}_n = \sigma_w^2 \mathbf{I}$. Therefore, the resulting SNR at the beamformer output, known as the *array SNR*, is

$$\text{SNR}_{\text{array}} = \frac{P_s}{P_n} = \frac{M |\mathbf{c}^H \mathbf{v}(\phi_s)|^2 \sigma_s^2}{\|\mathbf{c}\|^2 \sigma_w^2} = \frac{|\mathbf{c}^H \mathbf{v}(\phi_s)|^2}{\|\mathbf{c}\|^2} M \text{SNR}_{\text{elem}} \quad (11.2.13)$$

which is simply the product of the beamforming gain and the element level SNR. Thus, the *beamforming gain* is given by

$$G_{\text{bf}} \triangleq \frac{\text{SNR}_{\text{array}}}{\text{SNR}_{\text{elem}}} = \frac{|\mathbf{c}^H \mathbf{v}(\phi_s)|^2}{\|\mathbf{c}\|^2} M \quad (11.2.14)$$

The beamforming gain is strictly a function of the angle of arrival ϕ_s of the desired signal, the beamforming weight vector \mathbf{c} , and the number of sensors M .

11.2.1 Spatial Matched Filter

Recall the array signal model of a single signal, arriving from a direction ϕ_s , with sensor thermal noise

$$\begin{aligned}\mathbf{x}(n) &= \sqrt{M} \mathbf{v}(\phi_s) s(n) + \mathbf{w}(n) \\ &= [s(n) e^{-j2\pi u_s} s(n) \dots e^{-j2\pi(M-1)u_s} s(n)]^T + \mathbf{w}(n)\end{aligned}\quad (11.2.15)$$

where the components of the noise vector $\mathbf{w}(n)$ are uncorrelated and have power σ_w^2 , that is, $E\{\mathbf{w}(n)\mathbf{w}^H(n)\} = \sigma_w^2 \mathbf{I}$. The individual elements of the array contain the same signal $s(n)$ with different phase shifts corresponding to the differences in propagation times between elements. Ideally, the signals from the M array sensors are added coherently, which requires that each of the relative phases be zero at the point of summation; that is, we add $s(n)$ with a perfect replica of itself. Thus, we need a set of complex weights that results in a perfect phase alignment of all the sensor signals. The beamforming weight vector that phase-aligns a signal from direction ϕ_s at the different array elements is the *steering vector*, which is simply the array response vector in that direction, that is,

$$\mathbf{c}_{\text{mf}}(\phi_s) = \mathbf{v}(\phi_s) \quad (11.2.16)$$

The steering vector beamformer is also known as the *spatial matched filter*[†] since the steering vector is matched to the array response of signals impinging on the array from an angle ϕ_s . As a result, ϕ_s is known as the *look direction*. The use of the spatial matched filter is commonly referred to as *conventional beamforming*.

The output of the spatial matched filter is

$$\begin{aligned}y(n) &= \mathbf{c}_{\text{mf}}^H(\phi_s) \mathbf{x}(n) = \mathbf{v}^H(\phi_s) \mathbf{x}(n) \\ &= \frac{1}{\sqrt{M}} [1 \ e^{j2\pi u_s} \ \dots \ e^{j2\pi(M-1)u_s}] \\ &\quad \times \left\{ \begin{bmatrix} s(n) \\ e^{-j2\pi u_s} s(n) \\ \vdots \\ e^{-j2\pi(M-1)u_s} s(n) \end{bmatrix} + \mathbf{w}(n) \right\} \\ &= \frac{1}{\sqrt{M}} [s(n) + s(n) + \dots + s(n)] + \bar{w}(n) \\ &= \sqrt{M} s(n) + \bar{w}(n)\end{aligned}\quad (11.2.17)$$

where again $\bar{w}(n) = \mathbf{c}_{\text{mf}}^H(\phi_s) \mathbf{w}(n)$ is the beamformer output noise. Examining the array SNR of the spatial matched filter output, we obtain

$$\begin{aligned}\text{SNR}_{\text{array}} &= \frac{P_s}{P_n} = \frac{M \sigma_s^2}{E\{|\mathbf{v}^H(\phi_s) \mathbf{w}(n)|^2\}} \\ &= \frac{M \sigma_s^2}{\mathbf{v}^H(\phi_s) \mathbf{R}_n \mathbf{v}(\phi_s)} = M \frac{\sigma_s^2}{\sigma_w^2} = M \cdot \text{SNR}_{\text{elem}}\end{aligned}\quad (11.2.18)$$

since $P_s = M \sigma_s^2$ and $\mathbf{R}_n = \sigma_w^2 \mathbf{I}$. Therefore, the beamforming gain is

$$G_{\text{bf}} = M \quad (11.2.19)$$

that is, equal to the number of sensors. In the case of spatially white noise, the spatial matched filter is optimum in the sense of maximizing the SNR at the output of the beamformer. Thus,

[†]The spatial matched filter should not be confused with the optimum matched filters discussed in Section 6.9 that depend on the correlation of the data. However, it is optimum in the case of spatially uncorrelated noise.

the beamforming gain of the spatial matched filter is known as the *array gain* because it is the maximum possible gain of a signal with respect to sensor thermal noise for a given array. Clearly from this perspective, the more elements in the array, the greater the beamforming gain. However, physical reality places limitations on the number of elements that can be used. The spatial matched filter maximizes the SNR because the individual sensor signals are coherently aligned prior to their combination. However, as we will see, other sources of interference that have spatial correlation require other types of adaptive beamformers that maximize the signal-to-interference-plus-noise ratio (SINR).

The beampattern of the spatial matched filter can serve to illustrate several key performance metrics of an array. A sample beampattern of a spatial matched filter was shown in Figure 11.9 for $\phi_s = 0^\circ$. The first and most obvious attribute is the large lobe centered on ϕ_s , known as the mainlobe or *mainbeam*, and the remaining, smaller peaks are known as *sidelobes*. The value of the beampattern at the desired angle $\phi = \phi_s$ is equal to 1 (0 dB) due to the normalization used in the computation of the beampattern. A response of less than 1 in the look direction corresponds to a direct loss in desired signal power at the beamformer output. The sidelobe levels determine the rejection of the beamformer to signals not arriving from the look direction. The second attribute is the *beamwidth*, which is the angular span of the mainbeam. The resolution of the beamformer is determined by this mainlobe width, with smaller beamwidths resulting in better angular resolution. The beamwidth is commonly measured from the half-power (-3-dB) points $\Delta\phi_{3\text{dB}}$ or from null to null of the mainlobe $\Delta\phi_{\text{nn}}$. Using the beampattern, we next set out to examine the effects of the number of elements and their spacing on the array performance in the context of the spatial matched filter. However, in the following example, we first illustrate the use of a spatial matched filter to extract a signal from noise.

EXAMPLE 11.2.1. A signal received by a ULA with $M = 20$ elements and $\lambda/2$ spacing contains both a signal of interest at $\phi_s = 20^\circ$ with an array SNR of 20 dB and thermal sensor noise with unit power ($\sigma_w^2 = 1$). The signal of interest is an impulse present only in the 100th sample and is produced by the sequence of MATLAB commands

```
u_s=(d/lambda)* sin(phi_s*pi/180); s=zeros(M,N);
s(:,100)=(10^(SNR/20))*exp(-j*2*pi*u_s*[0:(M-1)]/M)/sqrt(M);
```

The uncorrelated noise samples with a Gaussian distribution are generated by

```
w=(randn(M,N)+j*randn(M,N))/sqrt(2);
```

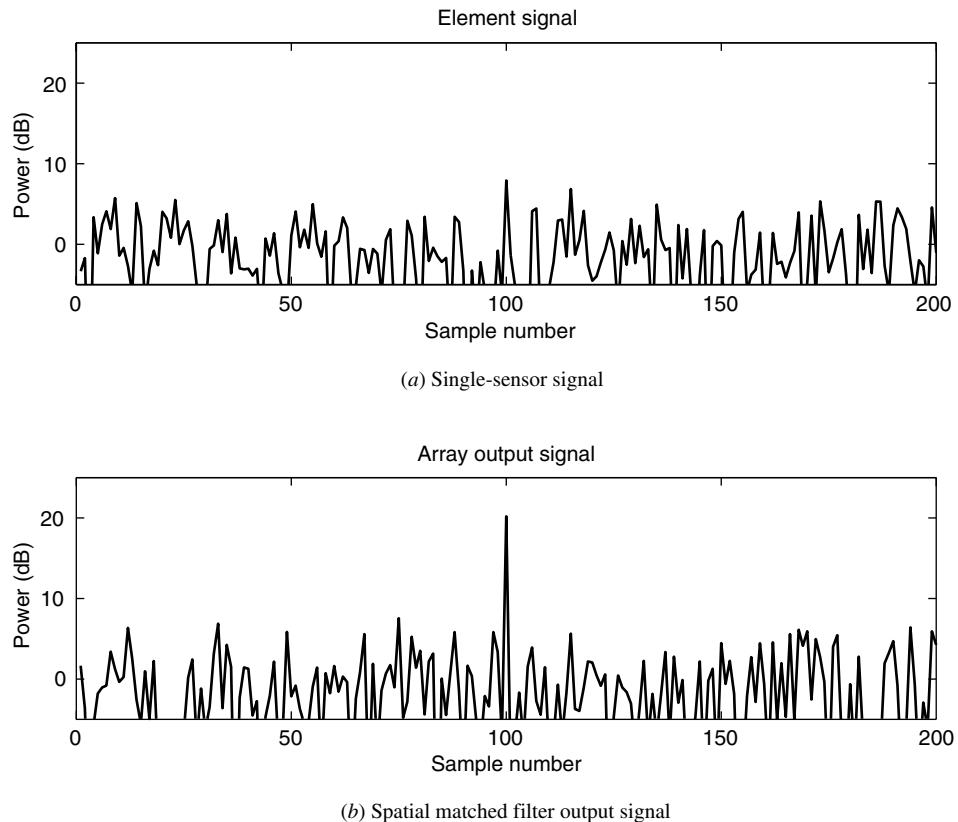
The two signals are added to produce the overall array signal $x = s + w$. Examining the signal at a single sensor in Figure 11.10(a), we see that the signal is not visible at $n = 100$ since the element level SNR is only 7 dB (full-array SNR minus M in decibels). The output power of this sample for a given realization can be more or less than the expected SNR due to the addition of the noise. However, when we apply a spatial matched filter using

```
c_mf=exp(-j*2*pi*u_s*[0:(M-1)]/M)/sqrt(M);
y=c_mf'*x;
```

we can clearly see the signal of interest since the array SNR is 20 dB. As a rule of thumb, we require the array SNR to be at least 10 to 12 dB to clearly observe the signal.

Element spacing

In Section 11.1.4, we determined that the element spacing must be $d \leq \lambda/2$ to prevent spatial aliasing. Here, we relax this restriction and look at various element spacings and the resulting array characteristics, namely, their beampatterns. In Figure 11.11, we show the beampatterns of spatial matched filters with $\phi_s = 0^\circ$ for ULAs with element spacings of $\lambda/4$, $\lambda/2$, λ , and 2λ (equal-sized apertures of 10λ with 40, 20, 10, and 5 elements, respectively).

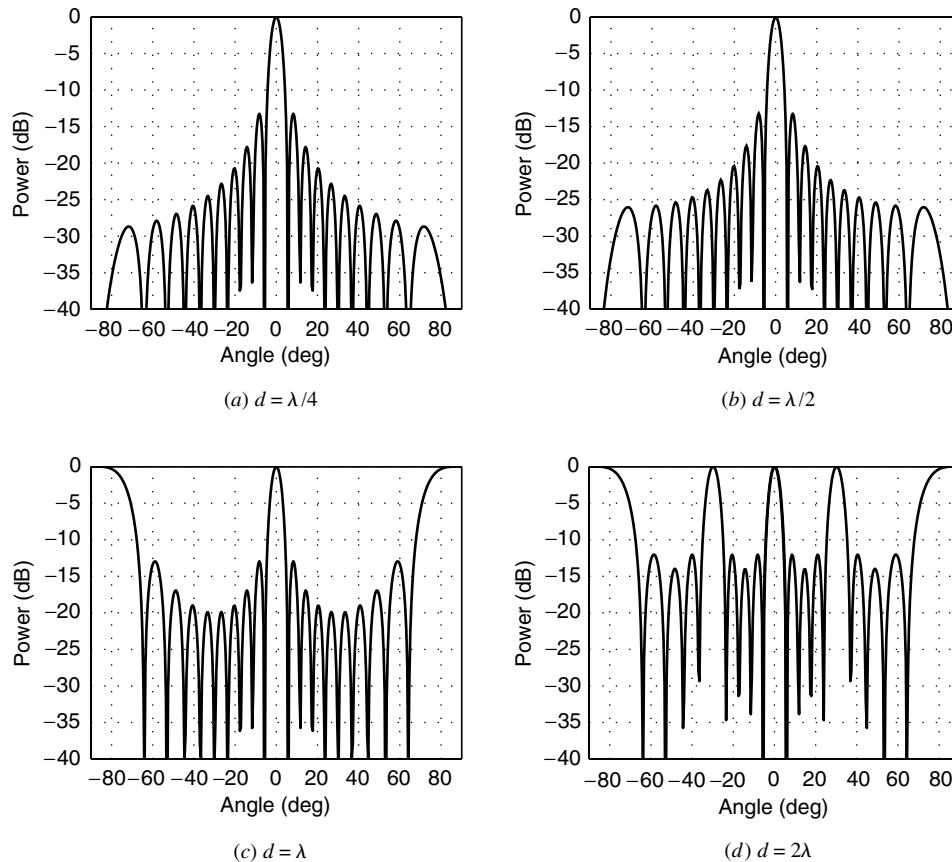
**FIGURE 11.10**

The spatial signals from (a) an individual sensor and (b) the output of a spatial matched filter beamformer.

We note that the beampatterns for $\lambda/4$ and $\lambda/2$ spacing are identical with equal-sized mainlobes and the first sidelobe having a height of -13 dB. The oversampling for the array with an element spacing of $\lambda/4$ provides no additional information and therefore does not improve the beamformer response in terms of resolution. In the case of the undersampled arrays ($d = \lambda$ and 2λ), we see the same structure (beamwidth) around the look direction but also note the additional peaks in the beampattern (0 dB) at $\pm 90^\circ$ for $d = \lambda$ and in even closer for $d = 2\lambda$. These additional lobes in the beampattern are known as *grating lobes*. Grating lobes create spatial ambiguities; that is, signals incident on the array from the angle associated with a grating lobe will look just like signals from the direction of interest. The beamformer has no means of distinguishing signals from these various directions. In certain applications, grating lobes may be acceptable if it is determined that it is either impossible or very improbable to receive returns from these angles; for example, a communications satellite is unlikely to receive signals at angles other than those corresponding to the ground below. The benefit of the larger element spacing is that the resulting array has a larger aperture and thus better resolution, which is our next topic of discussion. The topic of larger apertures with element spacing greater than $\lambda/2$ is commonly referred to as a *thinned array* and is addressed in Problem 11.5.

Array aperture and beamforming resolution

The aperture is the finite area over which a sensor collects spatial energy. In the case of a ULA, the aperture is the distance between the first and last elements. In general, the designer of an array yearns for as much aperture as possible. The greater the aperture, the

**FIGURE 11.11**

Beampatterns of a spatial matched filter for different element spacings with an equal-sized aperture $L = 10\lambda$.

finer the resolution of the array, which is its ability to distinguish between closely spaced sources. As we will see in Section 11.7, improved resolution results in better angle estimation capabilities. The angular resolution of a sensor array is measured in beamwidth $\Delta\phi$, which is commonly defined as the angular extent between the nulls of the mainbeam $\Delta\phi_{nn}$ or the half-power points of the mainbeam (-3 dB) $\Delta\phi_{3\text{dB}}$. As a general rule of thumb, the -3 -dB beamwidth for an array with an aperture length of L is quoted in radians as

$$\Delta\phi_{3\text{dB}} \approx \frac{\lambda}{L} \quad (11.2.20)$$

although the actual -3 -dB points of a spatial matched filter yield a resolution of $\Delta\phi_{3\text{dB}} = 0.89 \lambda/L$ (the resolution of the conventional matched filter near broadside, $\phi = 0^\circ$). The approximation in (11.2.20) is intended for the full range of prospective beamformers, not just spatial matched filters.[†] Since the resolution is dependent on the operating frequency F_c or equivalently on the wavelength, the aperture is often measured in wavelengths rather than in absolute length in meters. At large operating frequencies, say, $F_c = 10$ GHz or $\lambda = 3$ cm (X band in radar terminology), it is possible to populate a physical aperture of fixed length with a large number of elements, as opposed to lower operating frequencies, say, $F_c = 300$ MHz or $\lambda = 1$ m.

[†]Tapered beamformers, as discussed in Section 11.2.2, may considerably exceed this approximation, particularly for large tapers.

We illustrate the effect of aperture on resolution, using a few representative beampatterns. Figure 11.12 shows beampatterns for $M = 4, 8, 16$, and 32 with interelement spacing fixed at $d = \lambda/2$ (nonaliasing condition). Therefore, the corresponding apertures in wavelengths are $D = 2\lambda, 4\lambda, 8\lambda$, and 16λ . Clearly, increasing the aperture yields better resolution, with a factor-of-2 improvement for each of the successive twofold increases in aperture length. The level of the first sidelobe is always about -13 dB below the mainlobe peak.

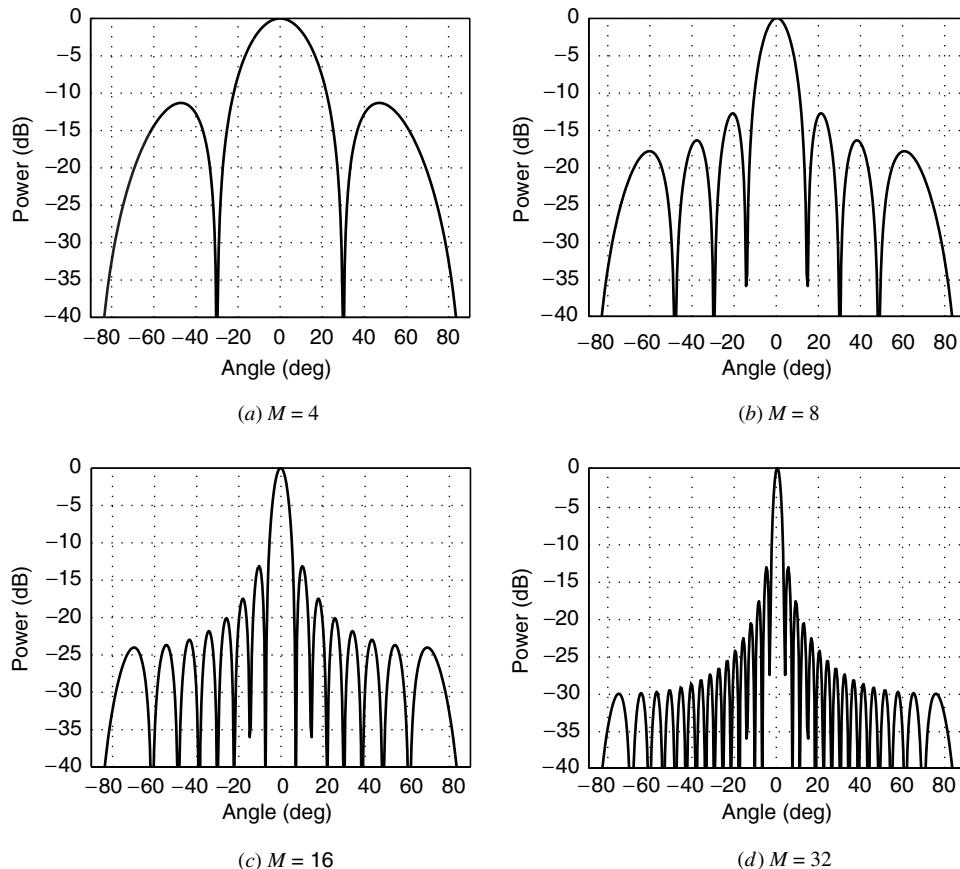


FIGURE 11.12

Beampatterns of a spatial matched filter for different aperture sizes with a common element of spacing of $d = \lambda/2$.

11.2.2 Tapered Beamforming

The spatial matched filter would be perfectly sufficient if the only signal present, aside from the sensor thermal noise, were the signal of interest. However, in many instances we must contend with other, undesired signals that hinder our ability to extract the signal of interest. These signals may also be spatially propagating at the same frequency as the operating frequency of the array. We refer to such signals as *interference*. These signals may be present due to hostile adversaries that are attempting to prevent us from receiving the signal of interest, for example, jammers in radar or communications; or they might be incidental signals that are present in our current operating environment, such as transmissions by other users in a communications system or radar clutter. In Sections 11.3, 11.5, and 11.6, we outline ways in which we can overcome these interferers by using adaptive methods.

However, there are also nonadaptive alternatives that can be employed in certain cases, namely, the use of a taper with the spatial matched filter.

Consider the ULA signal model from (11.2.15), but now including an interference signal $\mathbf{i}(n)$ made up of P interference sources

$$\mathbf{x}(n) = \mathbf{s}(n) + \mathbf{i}(n) + \mathbf{w}(n) = \sqrt{M}\mathbf{v}(\phi_s)s(n) + \sqrt{M} \sum_{p=1}^P \mathbf{v}(\phi_p)i_p(n) + \mathbf{w}(n) \quad (11.2.21)$$

where $\mathbf{v}(\phi_p)$ and $i_p(n)$ are the array response vector and actual signal due to the p th interferer, respectively. If we have a ULA with $\lambda/2$ element spacing, the beampattern of the spatial matched filter, as shown in Figure 11.13, may have sidelobes that are high enough to pass these interferers through the beamformer with a high enough gain to prevent us from observing the desired signal. For this array, if an interfering source were present at $\phi = 20^\circ$ with a power of 40 dB, the power of the interference at the output of the spatial matched filter would be 20 dB because the sidelobe level at $\phi = 20^\circ$ is only -20 dB. Therefore, if we were trying to receive a weaker signal from $\phi_s = 0^\circ$, we would be unable to extract it because of sidelobe leakage from this interferer.

The spatial matched filter has weights all with a magnitude equal to $1/\sqrt{M}$. The look direction is determined by a linear phase shift across the weights of the spatial matched filter. However, the sidelobe levels can be further reduced by tapering the magnitudes of the spatial matched filter. To this end, we employ a tapering vector \mathbf{t} that is applied to the spatial matched filter to realize a low sidelobe level beamformer

$$\mathbf{c}_{\text{tbf}}(\phi_s) = \mathbf{t} \odot \mathbf{c}_{\text{mf}}(\phi_s) \quad (11.2.22)$$

where \odot represents the *Hadamard product*, which is the element-by-element multiplication of the two vectors (Strang 1998). We refer to this beamformer as the *tapered beamformer*.

The determination of a taper can be thought of as the design of the desired beamformer where \mathbf{c}_{mf} simply determines the desired angle. The weight vector of the spatial matched filter from (11.2.16) has unit norm; that is, $\mathbf{c}_{\text{mf}}^H \mathbf{c}_{\text{mf}} = 1$. Similarly, the tapered beamformer \mathbf{c}_{tbf} is normalized so that

$$\mathbf{c}_{\text{tbf}}^H(\phi_s) \mathbf{c}_{\text{tbf}}(\phi_s) = 1 \quad (11.2.23)$$

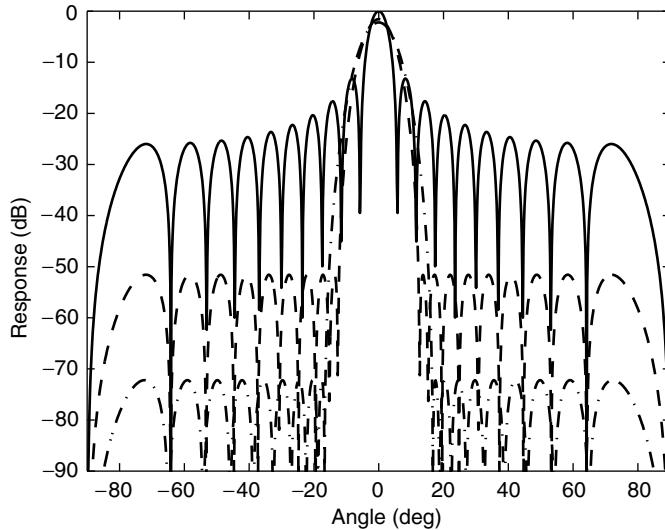
The choices for tapers, or windows, were outlined in Section 5.1 in the context of spectral estimation. Here, we use Dolph-Chebyshev tapers simply for illustration purposes. This taper produces a constant sidelobe level (equiripples in the stopband in spectral estimation), which is often a desirable attribute of a beamformer. The best taper choice is driven by the actual application. The beampatterns of the ULA are used again, but this time the beampatterns of tapered beamformers are also shown in Figure 11.13. The sidelobe levels of the tapers were chosen to be -50 and -70 dB.[†] The same 40-dB interferer would have been reduced to -10 and -30 dB at the beamformer output, respectively.

However, the use of tapers does not come without a cost. The peak of the beampattern is no longer at 0 dB. This loss in gain in the current look direction is commonly referred to as a *tapering loss* and is simply the beampattern evaluated at ϕ_s :

$$L_{\text{taper}} \triangleq |C_{\text{tbf}}(\phi_s)|^2 = |\mathbf{c}_{\text{tbf}}^H(\phi_s) \mathbf{v}(\phi_s)|^2 \quad (11.2.24)$$

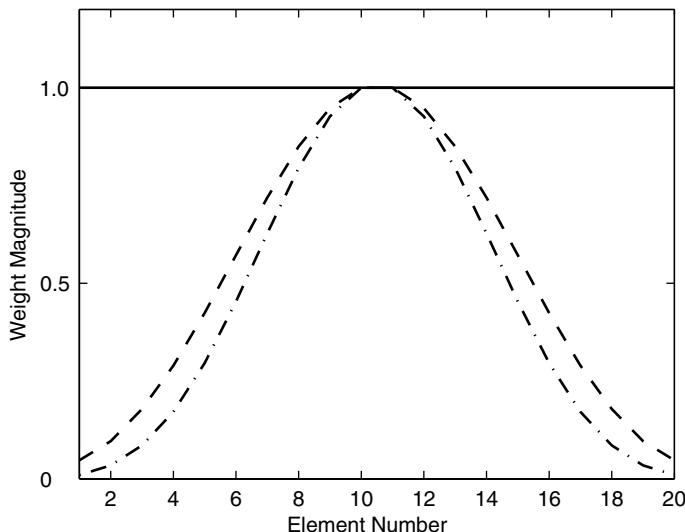
Since the tapering vector was normalized as in (11.2.23), the tapering loss is in the range $0 \leq L_{\text{taper}} \leq 1$ with $L_{\text{taper}} = 1$ corresponding to no loss (untapered spatial matched filter). The tapering loss is the loss in SNR of the desired signal at the beamformer output that cannot be recovered. More significantly, notice that the mainlobes of the beampatterns in Figure 11.13 are much broader for the tapered beamformers. The consequence is a loss

[†] In practice the tapering sidelobe levels are limited by array element location errors due to uncertainty. This limit is often at -30 dB but may be even higher. For illustration purposes we will ignore these limits in this chapter.

**FIGURE 11.13**

Beampatterns of beamformers with $M = 20$ with no taper (solid line), -50 -dB taper (dashed line), and -70 -dB taper (dash-dot line).

in resolution that becomes more pronounced as the tapering is increased to achieve lower sidelobe levels. This phenomenon was also treated within the context of spectral estimation in Section 5.1. However, its interpretation for an array can better be understood by examining plots of the magnitude of the taper vector \mathbf{t} , shown in Figure 11.14 for the -50 - and -70 -dB Dolph-Chebyshev tapers. Note that the elements on the ends of the array are given less weighting as the tapering level is increased. The tapered array in effect deemphasizes these end elements while emphasizing the center elements. Therefore, the loss in resolution for a tapered beamformer might be interpreted as a loss in the effective aperture of the array imparted by the tapering vector.

**FIGURE 11.14**

The magnitude levels of the tapered beamforming weights as a function of element number for $M = 20$ with no taper (solid line), -50 -dB taper (dashed line), and -70 -dB taper (dash-dot line).

EXAMPLE 11.2.2. We illustrate the use of tapers with the spatial matched filter for the extraction of a radar signal in the presence of a jamming interference source using a ULA with $M = 20$ elements with $\lambda/2$ spacing. The desired radar signal is known as a *target* and is present for only one sample in time. Here the target signal is at time sample (range gate) $n = 100$ and is at $\phi = 0^\circ$ with an array SNR of 20 dB. The jammer transmits a high-power, uncorrelated waveform (white noise). The angle of the jammer is $\phi_j = 20^\circ$, and its strength is 40 dB. The additive, sensor thermal noise has unit power (0 dB). We generate the jammer signal for $N = 200$ samples with the MATLAB commands

```
v_i = exp(-j*pi*[0:M-1]'*sin(phi_i*pi/180))/sqrt(M);
i_x=(10^(40/20))*v_i*(randn(1,N)+j*randn(1,N))/sqrt(2)
```

Similarly, the unit power thermal noise signal is produced by

```
w=(randn(M,N)+j*randn(M,N))/sqrt(2)
```

Two beamformers (steered to $\phi = 0^\circ$) are applied to the resulting array returns: a spatial matched filter and a tapered beamformer with a -50-dB sidelobe level. The resulting beamformer output signals are shown in Figure 11.15. The spatial matched filter is unable to reduce the jammer sufficiently to observe the target signal at $n = 100$. However, the tapered beamformer is able to attenuate the jammer signal below the thermal noise level and the target is easily extracted. The target signal is approximately 18.5 dB with the -1.5 dB loss due to the tapering loss in (11.2.24).

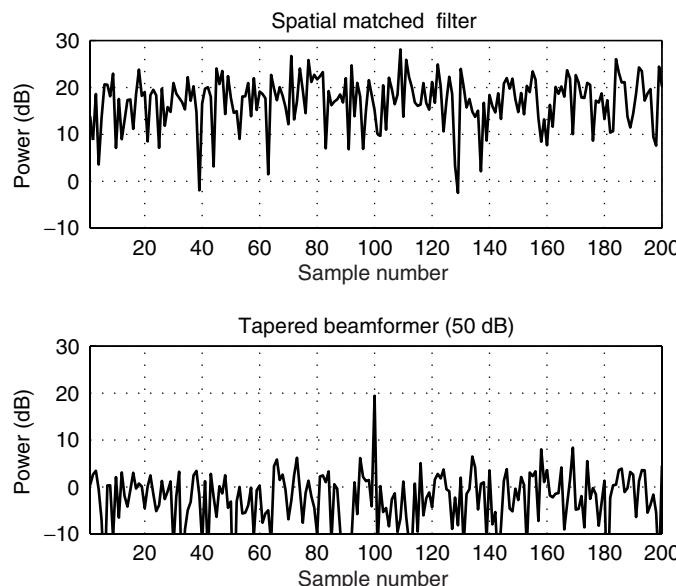


FIGURE 11.15

The output signals of a spatial matched filter and a tapered beamformer (-50-dB).

11.3 OPTIMUM ARRAY PROCESSING

So far, we have only considered beamformers whose weights are determined independently of the data to be processed. If instead we base the actual beamforming weights on the array data themselves, then the result is an *adaptive array* and the operation is known as *adaptive beamforming*. Ideally, the beamforming weights are adapted in such a way as to optimize the spatial response of the resulting beamformer based on a certain criterion. To this end, the

criterion is chosen to enhance the desired signal while rejecting other, unwanted signals. This weight vector is similar to the optimum matched filter from Chapter 6. However, the manner in which it is implemented, namely, the methodology of how this equation is successfully applied to the array processing problem, is the topic of this and the next three sections.

This section focuses on optimum array processing methods that make use of the a priori known statistics of the data to derive the beamforming weights. Implicit in the optimization is the a priori knowledge of the true statistics of the array data. In Section 11.5, we discuss techniques for implementing these methods that estimate the unknown statistics from the data. We will use the general term *adaptive* to refer to beamformers that use an estimated correlation matrix computed from array snapshots, while reserving the term *optimum* for beamformers that optimize a certain criterion based on knowledge of the array data statistics. We begin by discussing the array signal model that contains interference in addition to the desired signal and noise. We then proceed to derive the *optimum beamformer*, where the optimality criterion is the maximization of the theoretical signal-to-interference-plus-noise ratio. In addition, we give an alternate implementation of the optimum beamformer: the generalized sidelobe canceler. This structure also gives an intuitive understanding of the optimum beamformer. Various issues associated with the optimum beamformer, namely, the effect of signal mismatch and bandwidth on the performance of an optimum beamformer, are discussed in Section 11.4.

The signal of interest is seldom the only array signal aside from thermal noise present. The array must often contend with other, undesired signals that interfere with our ability to extract this signal of interest, as described in Section 11.2.2. Often the interference is so powerful that even a tapered beamformer is unable to sufficiently suppress it to extract the signals of interest. The determination of the presence of signals of interest is known as *detection*, while the inference of their parameters, for example, the angle of arrival ϕ_s , is referred to as *estimation*. The topic of detection is not explicitly treated here. Rather, we seek to maximize the *visibility* of the desired signal at the array output, that is, the ratio of the signal power to that of the interference plus noise, to facilitate the detection process. There are several textbooks devoted to the subject of detection theory (Scharf 1991; Poor 1994; Kay 1998) to which the interested reader is referred. Parameter estimation methods to determine the angle of the desired signal are the topic of Section 11.7.

Consider an array signal that consists of the desired signal $\mathbf{s}(n)$, an interference signal $\mathbf{i}(n)$, along with sensor thermal noise $\mathbf{w}(n)$, that is,

$$\mathbf{x}(n) = \mathbf{s}(n) + \mathbf{i}(n) + \mathbf{w}(n) = \sqrt{M} \mathbf{v}(\phi_s) s(n) + \mathbf{i}(n) + \mathbf{w}(n) \quad (11.3.1)$$

where $s(n)$ is a signal with deterministic amplitude σ_s and uniformly distributed random phase. The interference-plus-noise component of the array signal is

$$\mathbf{x}_{i+n}(n) = \mathbf{i}(n) + \mathbf{w}(n) \quad (11.3.2)$$

which are both modeled as zero-mean stochastic processes. The interference has spatial correlation according to the angles of the contributing interferers, while the thermal noise is spatially uncorrelated. The interference component of the signal may consist of several sources, as modeled in (11.2.21). The sensor thermal noise is assumed to be uncorrelated with power σ_w^2 . The assumption is made that *all of these three components are mutually uncorrelated*. As a result, the array correlation matrix is

$$\mathbf{R}_x = E\{\mathbf{x}(n)\mathbf{x}^H(n)\} = M\sigma_s^2 \mathbf{v}(\phi_s) \mathbf{v}^H(\phi_s) + \mathbf{R}_i + \mathbf{R}_n \quad (11.3.3)$$

where σ_s^2 is the power of the signal of interest and \mathbf{R}_i and \mathbf{R}_n are the interference and noise correlation matrices, respectively. The interference-plus-noise correlation matrix is the sum of these latter two matrices

$$\mathbf{R}_{i+n} = \mathbf{R}_i + \sigma_w^2 \mathbf{I} \quad (11.3.4)$$

where $\mathbf{R}_n = \sigma_w^2 \mathbf{I}$ since the sensor thermal noise is spatially uncorrelated.

The ultimate goal of the prospective adaptive beamformer is to combine the sensor signals in such a way that the interference signal is reduced to the level of the thermal noise while the desired signal is preserved. Stated another way, we would like to maximize the ratio of the signal power to that of the interference plus noise, known as the *signal-to-interference-plus-noise ratio (SINR)*. Maximizing the SINR is the optimal criterion for most detection and estimation problems. Simply stated, *maximizing the SINR seeks to improve the visibility of the desired signal as much as possible in a background of interference*. This criterion should not be confused with maximizing the SNR (spatial matched filter) in the absence of interference.

At the input of the array, that is, in each individual sensor, the SINR is given by

$$\text{SINR}_{\text{elem}} = \frac{\sigma_s^2}{\sigma_i^2 + \sigma_w^2} \quad (11.3.5)$$

where σ_s^2 , σ_i^2 , and σ_w^2 are the signal, interference, and thermal noise powers in each individual element. The SINR at the beamformer output, following the application of the beamforming weight vector \mathbf{c} , is

$$\text{SINR}_{\text{out}} = \frac{|\mathbf{c}^H \mathbf{s}(n)|^2}{E\{|\mathbf{c}^H \mathbf{x}_{i+n}(n)|^2\}} = \frac{M\sigma_s^2 |\mathbf{c}^H \mathbf{v}(\phi_s)|^2}{\mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c}} \quad (11.3.6)$$

We wish to maximize this array output SINR. First, note that the interference-plus-noise correlation matrix can be factored as

$$\mathbf{R}_{i+n} = \mathbf{L}_{i+n} \mathbf{L}_{i+n}^H \quad (11.3.7)$$

where \mathbf{L}_{i+n} is the Cholesky factor of the correlation matrix.[†] See Section 6.3 for details. Thus, defining

$$\tilde{\mathbf{c}} = \mathbf{L}_{i+n}^H \mathbf{c} \quad \tilde{\mathbf{v}}(\phi_s) = \mathbf{L}_{i+n}^{-1} \mathbf{v}(\phi_s) \quad (11.3.8)$$

we can rewrite (11.3.6) as

$$\text{SINR}_{\text{out}} = \frac{M\sigma_s^2 |\tilde{\mathbf{c}}^H \tilde{\mathbf{v}}(\phi_s)|^2}{\tilde{\mathbf{c}}^H \tilde{\mathbf{c}}} \quad (11.3.9)$$

Using the Schwartz inequality

$$\tilde{\mathbf{c}}^H \tilde{\mathbf{v}}(\phi_s) \leq \|\tilde{\mathbf{c}}\| \|\tilde{\mathbf{v}}(\phi_s)\| \quad (11.3.10)$$

and substituting (11.3.10) into (11.3.9), we find that

$$\text{SINR}_{\text{out}} \leq M\sigma_s^2 \frac{\|\tilde{\mathbf{c}}\|^2 \|\tilde{\mathbf{v}}(\phi_s)\|^2}{\|\tilde{\mathbf{c}}\|^2} = M\sigma_s^2 \|\tilde{\mathbf{v}}(\phi_s)\|^2 \quad (11.3.11)$$

Thus, the maximum SINR is found by satisfying the upper bound for (11.3.11), which yields

$$\text{SINR}_{\text{out}}^{\max} = M\sigma_s^2 \tilde{\mathbf{v}}^H(\phi_s) \tilde{\mathbf{v}}(\phi_s) = M\sigma_s^2 [\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)] \quad (11.3.12)$$

We also see that the same maximum SINR is obtained if we set $\tilde{\mathbf{c}} = \alpha \tilde{\mathbf{v}}(\phi_s)$ where α is an arbitrary constant. In other words, the SINR is maximized when these two vectors are parallel to each other and α can be chosen to satisfy other requirements. Therefore, using (11.3.8), we can solve for the optimum weight vector (Brynn 1962; Capon et al. 1967; Brennan and Reed 1973)

$$\mathbf{c}_o = \alpha \mathbf{L}_{i+n}^{-H} \tilde{\mathbf{v}}(\phi_s) = \alpha \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s) \quad (11.3.13)$$

where α is an arbitrary constant. Thus, the optimum beamforming weights are proportional to $\mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)$. The proportionality constant α in (11.3.13) can be set in a variety of ways.

[†]Note that any square root factorization $\mathbf{R}_{i+n} = \mathbf{R}_{i+n}^{1/2} \mathbf{R}_{i+n}^{H/2}$ of the correlation matrix can be chosen.

Table 11.1 gives various normalizations for the optimum beamformer. The normalization we adopt throughout this chapter is to constrain the optimum beamformer to have unity gain in the look direction, that is, $\mathbf{c}_o^H \mathbf{v}(\phi_s) = 1$. Therefore,

$$\mathbf{c}_o^H \mathbf{v}(\phi_s) = \alpha [\mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)]^H \mathbf{v}(\phi_s) = 1 \quad (11.3.14)$$

and the resulting optimum beamformer is given by

$$\mathbf{c}_o = \frac{\mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)}{\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)} \quad (11.3.15)$$

In general, the normalization of the optimum beamformer is arbitrary and is dictated by the use of the output, for example, measure residual interference power or detection. In any case, the SINR is maximized independently of the normalization. The most commonly used normalizations are listed in Table 11.1.

TABLE 11.1
Optimum weight normalizations for unit gain in look direction, unit gain on noise, and unit gain on interference-plus-noise constraints.

Constraint	Mathematical formulation	Optimum beamformer normalization
MVDR (unit gain in look direction)	$\mathbf{c}_o^H \mathbf{v}(\phi_s) = 1$	$\alpha = [\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)]^{-1}$
Unit noise gain	$\mathbf{c}_o^H \mathbf{c}_o = 1$	$\alpha = [\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-2} \mathbf{v}(\phi_s)]^{-1/2}$
Unit gain on interference-plus-noise*	$\mathbf{c}_o^H \mathbf{R}_{i+n} \mathbf{c}_o = 1$	$\alpha = [\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)]^{-1/2}$

*This normalization is commonly referred to as the adaptive matched filter normalization (Robey et al. 1992). Its use is primarily for detection purposes. Since the output level of the interference-plus-noise has a set power of unity, a constant detection threshold can be used for all angles.

Alternately, the optimum beamformer can be derived by solving the following constrained optimization problem: Minimize the interference-plus-noise power at the beamformer output

$$P_{i+n} = E\{|\mathbf{c}^H \mathbf{x}_{i+n}(n)|^2\} = \mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c} \quad (11.3.16)$$

subject to a look-direction distortionless response constraint, that is,

$$\min P_{i+n} \quad \text{subject to} \quad \mathbf{c}^H \mathbf{v}(\phi_s) = 1 \quad (11.3.17)$$

The solution of this constrained optimization problem is found by using Lagrange multipliers (see Appendix B and Problem 11.7) and results in the same weight vector as (11.3.15). This formulation has led to the commonly used term *minimum-variance distortionless response (MVDR) beamformer*. For a discussion of minimum-variance beamforming, see Van Veen (1992). The optimum beamformer passes signals impinging on the array from angle ϕ_s while rejecting significant energy (interference) from all other angles. This beamformer can be thought of as an optimum spatial matched filter since it provides maximum interference rejection, while matching the response of signals impinging on the array from a direction ϕ_s . The optimal weights balance the rejection of interference with the thermal noise gain so that the output thermal noise does not cause a reduction in the output SINR.

The optimum beamformer maximizes the SINR given by (11.3.12), which is independent of the normalization. Another useful metric is a measure of the performance relative to the interference-free case, that is, $\mathbf{x}(n) = \mathbf{s}(n) + \mathbf{w}(n)$. To gauge the performance of the beamformer independently of the desired signal power, we simply normalize the SINR

by the hypothetical array output SNR had there been no interference present, which from (11.2.18) is $\text{SNR}_0 = M\sigma_s^2/\sigma_w^2$. The resulting measure is known as the *SINR loss*, which for the optimum beamformer, by substituting into (11.3.12), is

$$L_{\text{sincr}}(\phi_s) \triangleq \frac{\text{SINR}_{\text{out}}(\phi_s)}{\text{SNR}_0} = \sigma_w^2 \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s) \quad (11.3.18)$$

The SINR loss is always between 0 and 1, taking on the maximum value when the performance is equal to the interference-free case. Typically, the SINR loss is computed across all angles for a given interference scenario. In this sense, the SINR loss of the optimum beamformer provides a measure of the residual interference remaining following optimum processing and informs us of our loss in performance due to the presence of interference. We also notice that (11.3.18) is the reciprocal of the minimum-variance power spectrum of the interference plus noise. Minimum-variance power spectrum estimation was discussed in Section 9.5.

EXAMPLE 11.3.1. To demonstrate the optimum beamformer, we consider a scenario in which there are three interference sources and compare it to a conventional beamformer (spatial matched filter). The array is a 20-element ULA with $\lambda/2$ element spacing. These interferers are at the following angles with the corresponding interference-to-noise ratios (INRs) in decibels: $\phi = 20^\circ$ and INR = 35 dB, $\phi = -30^\circ$ and INR = 70 dB, and $\phi = 50^\circ$ and INR = 50 dB. The optimum beamformer is first computed using (11.3.15) for a look direction of $\phi_s = 0^\circ$. The beampattern of this optimum beamformer is computed by using (11.2.3) and is plotted in Figure 11.16(a). Notice the nulls at the angles of the interference ($\phi = -30^\circ, 20^\circ, 50^\circ$). These nulls are deep enough that the interference at the beamformer output is below the sensor thermal noise level. The conventional beamformer, however, cannot place nulls on the interferers since it is independent of the data. We also perform optimum beamforming across all angles $-90^\circ < \phi < 90^\circ$ and compute the corresponding SINR loss due to the interference using (11.3.18). The SINR loss is plotted in Figure 11.16(b). The notches at the interference angles are simply the negative of the INR of the interferers corresponding to significant losses in performance. However, these performance losses are limited to these angles. The SINR loss at all other angles is almost at its maximum value of 1 (0 dB). The SINR loss of the conventional beamformer is significantly worse at all angles because of the strong interference that makes its way to the beamformer output through its sidelobes.

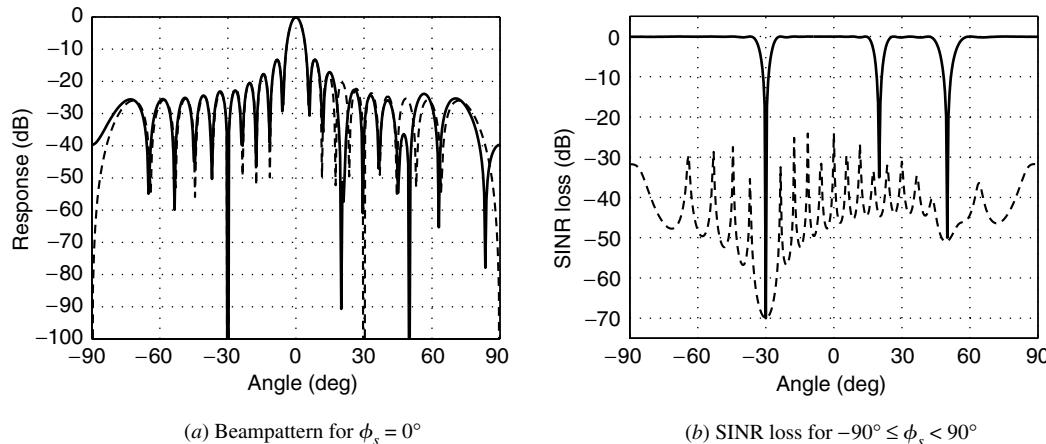


FIGURE 11.16

Beampattern (steered to $\phi = 0^\circ$) and SINR loss plots versus angle. Solid line is the optimum beamformer, and dashed line is the conventional beamformer.

EXAMPLE 11.3.2. We revisit the problem from Example 11.2.2 with a jammer at $\phi_i = 20^\circ$ except the jammer power is now 70 dB. Clearly, the -50-dB tapered beamformer is no longer

capable of sufficiently suppressing this jammer. Rather, we compute the optimum beamformer using (11.3.15), where $\mathbf{R}_{i+n} = 10^7 \mathbf{v}(\phi_i) \mathbf{v}^H(\phi_i) + \mathbf{I}$. First, we examine the beampattern of the optimum beamformer steered to $\phi = 0^\circ$ in Figure 11.17(a). Notice the null on the jammer at $\phi = 20^\circ$ with a depth of greater than -150 dB. We also plot the SINR loss in Figure 11.17(b) as we scan the look direction from -90° to 90° . Almost no SINR loss is experienced at angles away from the jammer, while at the jammer angle $\phi = 20^\circ$, the SINR loss corresponds to the jammer power (70 dB). As a similar exercise to that in Example 11.2.2, we can produce a target signal at $\phi = 0^\circ$ and attempt to extract it, using both a spatial matched filter and an optimum beamformer. The output signals are shown in Figure 11.17(c) and (d), respectively. The optimum beamformer is able to successfully extract the signal whereas the output of the spatial matched filter is dominated by interference. Notice that we do not suffer the taper loss on the target as we did for the tapered beamformer due to the $\mathbf{c}_o^H \mathbf{v}(\phi_s) = 1$ constraint in the optimum beamformer.

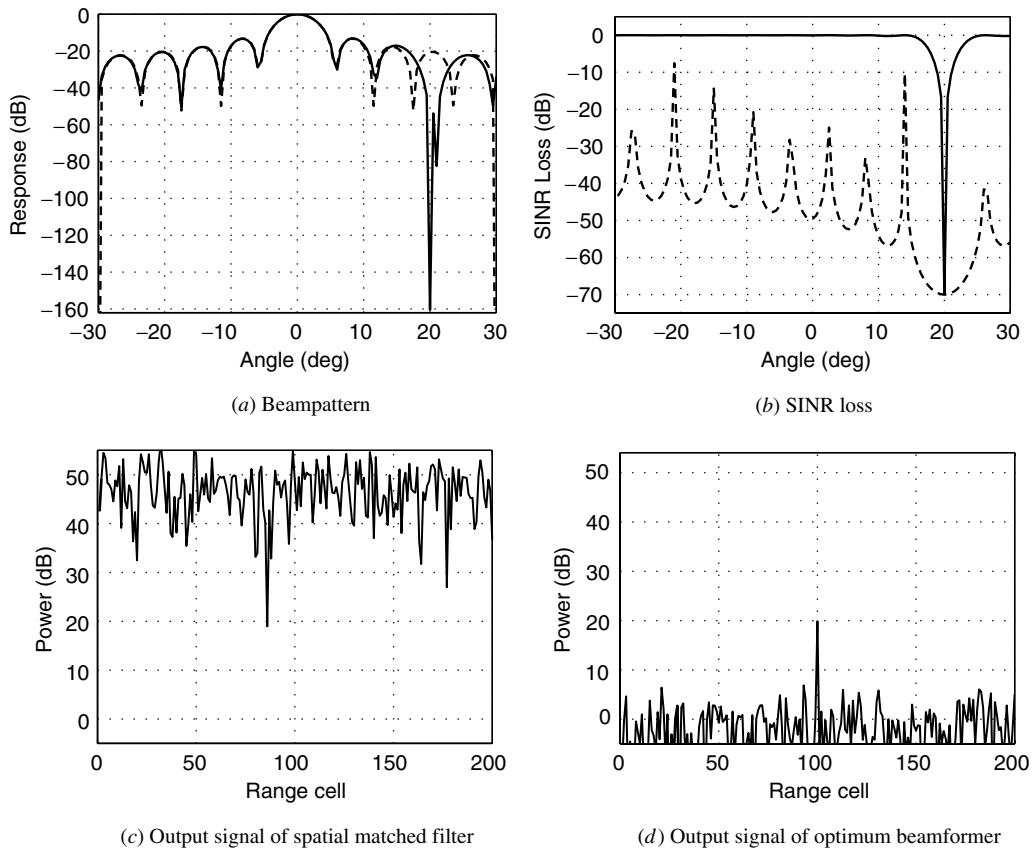


FIGURE 11.17
(a) Beampattern and (b) SINR loss of optimum beamformer (solid line) versus spatial matched filter (dashed line), along with (c) the output signals of a spatial matched filter and (d) the optimum beamformer.

11.3.2 Eigenanalysis of the Optimum Beamformer

In many cases, significant insight can be gained by considering the optimum beamformer in terms of the eigenvalues and eigenvectors of the interference-plus-noise correlation matrix

$$\mathbf{R}_{i+n} = \sum_{m=1}^M \lambda_m \mathbf{q}_m \mathbf{q}_m^H \quad (11.3.19)$$

where the eigenvalues have been ordered from largest to smallest, that is, $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_M$. If the rank of the interference is P , then $\lambda_m = \sigma_w^2$ for $m > P$; that is, the remainder of the eigenvalues is equal to the thermal noise power. The eigenvectors are orthonormal ($\mathbf{q}_m^H \mathbf{q}_k = 0$ for $k \neq m$, $\mathbf{q}_m^H \mathbf{q}_m = 1$) and form a basis for the interference-plus-noise subspace that can be split into interference and noise subspaces given by

$$\text{Interference subspace: } \{\mathbf{q}_m \mid 1 \leq m \leq P\} \quad \text{Noise subspace: } \{\mathbf{q}_m \mid P < m \leq M\} \quad (11.3.20)$$

The inverse of \mathbf{R}_{i+n} can also be written in terms of the eigenvalues and eigenvectors, λ_m and \mathbf{q}_m , of the correlation matrix \mathbf{R}_{i+n} , that is,

$$\mathbf{R}_{i+n}^{-1} = \sum_{m=1}^M \frac{1}{\lambda_m} \mathbf{q}_m \mathbf{q}_m^H \quad (11.3.21)$$

We further assume that the rank of the interference is less than the total number of sensors, that is, $P < M$. In this case, the smallest eigenvalues of \mathbf{R}_{i+n} are noise eigenvalues and are equal to the thermal noise power $\lambda_m = \sigma_w^2$ for $m > P$. Substituting (11.3.21) into the optimum beamformer weights in (11.3.15), we have

$$\begin{aligned} \mathbf{c}_o &= \alpha \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s) = \alpha \sum_{m=1}^M \frac{1}{\lambda_m} \mathbf{q}_m \mathbf{q}_m^H \mathbf{v}(\phi_s) \\ &= \alpha \left[\frac{1}{\sigma_w^2} \mathbf{v}(\phi_s) - \frac{1}{\sigma_w^2} \mathbf{v}(\phi_s) + \sum_{m=1}^M \frac{\mathbf{q}_m^H \mathbf{v}(\phi_s)}{\lambda_m} \mathbf{q}_m \right] \\ &= \frac{\alpha}{\sigma_w^2} \left\{ \mathbf{v}(\phi_s) - \sum_{m=1}^M \frac{\lambda_m - \sigma_w^2}{\lambda_m} [\mathbf{q}_m^H \mathbf{v}(\phi_s)] \mathbf{q}_m \right\} \end{aligned} \quad (11.3.22)$$

where $\alpha = [\mathbf{v}(\phi_s)^H \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)]^{-1}$. The resulting beam response is

$$C_o(\phi) = \frac{\alpha}{\sigma_w^2} \left\{ C_q(\phi) - \sum_{m=1}^M \frac{\lambda_m - \sigma_w^2}{\lambda_m} [\mathbf{q}_m^H \mathbf{v}(\phi_s)] Q_m(\phi) \right\} \quad (11.3.23)$$

$$\text{where } C_q(\phi) = \mathbf{v}^H(\phi_s) \mathbf{v}(\phi) = \mathbf{c}_{mf}^H \mathbf{v}(\phi) = C_{mf}(\phi) \quad (11.3.24)$$

is the response of the spatial matched filter $\mathbf{c}_{mf}(\phi_s) = \mathbf{v}(\phi_s)$ [see (11.2.16)] and is known as the *quiescent response* of the optimum beamformer. However,

$$Q_m(\phi) = \mathbf{q}_m^H \mathbf{v}(\phi) \quad (11.3.25)$$

is the beam response of the m th eigenvector, known as an *eigenbeam*. Thus, the response of the optimum beamformer consists of weighted eigenbeams subtracted from the quiescent response. The weights for the eigenbeams are determined by the corresponding eigenvalue, the noise power, and the cross-product of the look-direction steering vector and the respective eigenvector. Examining the term $(\lambda_m - \sigma_w^2)/\lambda_m$, we see clearly that for strong interferers $\lambda_m \gg \sigma_w^2$ and $(\lambda_m - \sigma_w^2)/\lambda_m \approx 1$, and the eigenbeam is subtracted from the quiescent response weighted by $\mathbf{q}_m^H \mathbf{v}(\phi)$. This subtraction of properly weighted interference eigenvectors places nulls in the directions of the interference sources. The term $\mathbf{q}_m^H \mathbf{v}(\phi_s)$ in (11.3.23) scales the interference eigenbeam to the quiescent response of the spatial matched filter in the direction of the corresponding interferer. Thus, the null depth for an interferer of the beampattern $|C_o(\phi)|^2$ is determined by the response of the eigenbeam to the quiescent response and the strength of the interferer relative to the noise level. However, for the noise eigenvalues $\lambda_m = \sigma_w^2$ and $(\lambda_m - \sigma_w^2)/\lambda_m = 0$. Therefore, the noise eigenvectors have no effect on the optimum beamformer. Interestingly, for the case of noise only and thus all noise eigenvalues, that is, no interference present, the optimum beamformer reverts to the

spatial matched filter $\mathbf{c}_o(\phi_s) = \mathbf{c}_{\text{mf}}(\phi_s) = \mathbf{v}(\phi_s)$, which is the beamformer that maximizes the SNR.

11.3.3 Interference Cancelation Performance

The interference cancelation performance of the optimum beamformer can be determined by examining the beam response at the angles of the interferers. The beam response at these angles indicates the depth of the null that the optimum beamformer places on the interferer. Using the MVDR optimum beamformer from (11.3.15), we see that the response in the direction of an interferer ϕ_p of an optimum beamformer that is steered in direction ϕ_s is

$$C_o(\phi_p) = \mathbf{c}_o^H \mathbf{v}(\phi_p) = \alpha \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_p) \quad (11.3.26)$$

where ϕ_p is the angle of the p th interferer and $\alpha = [\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)]^{-1}$. Now we note that \mathbf{R}_{i+n} can be split into a component due to the p th interferer and the correlation matrix of the remaining interference-plus-noise \mathbf{Q}_{i+n}

$$\mathbf{R}_{i+n} = \mathbf{Q}_{i+n} + M\sigma_p^2 \mathbf{v}(\phi_p) \mathbf{v}^H(\phi_p) \quad (11.3.27)$$

where σ_p^2 is the power of the p th interferer in a single element. Using the matrix inversion lemma (Appendix A), we obtain

$$\mathbf{R}_{i+n}^{-1} = \mathbf{Q}_{i+n}^{-1} - M\sigma_p^2 \frac{\mathbf{Q}_{i+n}^{-1} \mathbf{v}(\phi_p) \mathbf{v}^H(\phi_p) \mathbf{Q}_{i+n}^{-1}}{1 + M\sigma_p^2 \mathbf{v}^H(\phi_p) \mathbf{Q}_{i+n}^{-1} \mathbf{v}(\phi_p)} \quad (11.3.28)$$

Substituting (11.3.28) into (11.3.26), we find the optimum beamformer response to be (Richmond 1999)

$$\begin{aligned} C_o(\phi_p) &= \alpha \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_p) \\ &= \alpha \mathbf{v}^H(\phi_s) \mathbf{Q}_{i+n}^{-1} \mathbf{v}(\phi_p) \\ &\quad - \alpha \mathbf{v}^H(\phi_s) \mathbf{Q}_{i+n}^{-1} \mathbf{v}(\phi_p) \mathbf{v}^H(\phi_p) \mathbf{Q}_{i+n}^{-1} \mathbf{v}(\phi_p) \\ &\quad \times \left[\frac{M\sigma_p^2}{1 + M\sigma_p^2 \mathbf{v}^H(\phi_p) \mathbf{Q}_{i+n}^{-1} \mathbf{v}(\phi_p)} \right] \\ &= \underbrace{\frac{\mathbf{v}^H(\phi_s) \mathbf{Q}_{i+n}^{-1} \mathbf{v}(\phi_p)}{\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)}}_{\text{term 1}} \underbrace{\frac{1}{1 + M\sigma_p^2 \mathbf{v}^H(\phi_p) \mathbf{Q}_{i+n}^{-1} \mathbf{v}(\phi_p)}}_{\text{term 2}} \end{aligned} \quad (11.3.29)$$

We notice that the optimum beamformer response is made up of the product of two terms. The first term is the response at angle ϕ_p of an optimum beamformer steered in direction ϕ_s formed in the absence of this interferer ($\sigma_p^2 = 0$), that is, the sidelobe level of the optimum beamformer had this interference not been present. However, the power of the interferer is many times significantly greater than this sidelobe level, and the optimum beamformer cancels the interferer by placing a null at the angle of the interferer. The second term produces the null at the angle ϕ_p . By examining this term, it is apparent that the depth of the null is determined by the power of the interferer $M\sigma_p^2$. Clearly, the larger the power of the interferer, the smaller this term becomes and the deeper the null depth of the optimum beamformer is at ϕ_p . The factor $\mathbf{v}^H(\phi_p) \mathbf{Q}_{i+n}^{-1} \mathbf{v}(\phi_p)$ is the amount of energy received from ϕ_p not including the interferer and has as a lower bound equal to the thermal noise power (spatially white). Since the power response of the beamformer is $|C_o(\phi_p)|^2$, the null depth is actually proportional to $M^2\sigma_p^4$, or twice the power of the interferer at the array output, in decibels (Compton 1988).

In the derivation of the optimum beamformer, we used the vector $\mathbf{v}(\phi_s)$ that was matched to the array response of a desired signal arriving from an angle ϕ_s . The resulting beamformer weight vector \mathbf{c}_o has unity gain in this direction; that is, $\mathbf{c}_o^H \mathbf{v}(\phi_s) = 1$, owing to the normalization of the weights. However, the sidelobes of the beamformer are still at the same levels as the spatial matched filter (nonadaptive beamformer) from (11.2.16), although with a different structure, as can be seen from a sample beampattern of the optimum beamformer shown in Figure 11.18(a).

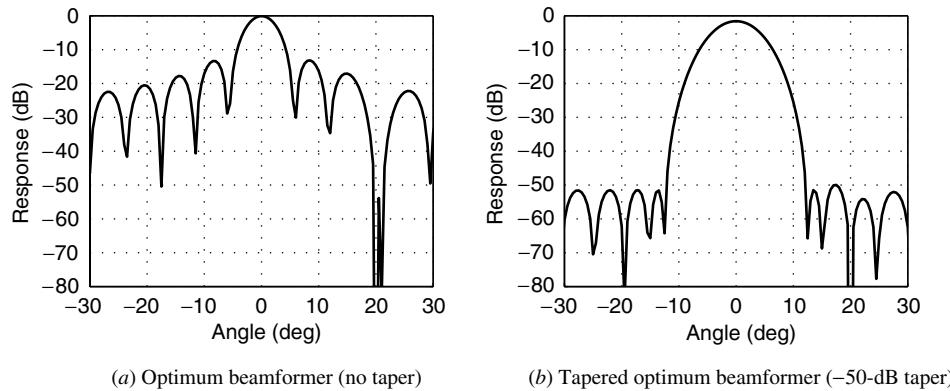


FIGURE 11.18

Beampatterns of an optimum beamformer (a) without and (b) with tapering (-50-dB Dolph-Chebyshev taper) steered to $\phi = 0^\circ$.

The optimum beamformer uses the interference-plus-noise correlation matrix \mathbf{R}_{i+n} . Now, although the beamformer weights must be estimated from intervals of the data that contain only interference (no desired signal present), they are presumably applied to segments that contain both interference and a desired signal. What happens when we are searching an angular region for potential desired signals? A desired signal at an angle ϕ_1 may be easily found by using an adaptive beamformer directed to this angle ($\phi_s = \phi_1$), assuming the signal strength after beamforming is significantly larger than the sensor thermal noise. However, we will also be searching other angles for potential desired signals. If we are looking at one of these other angles, say, $\phi_2 \neq \phi_1$, we want to avoid concluding a signal is present when it may actually be due to sidelobe leakage of the signal at ϕ_1 . This problem is best illustrated by using the beampattern of an optimum beamformer with an interferer at $\phi = 20^\circ$ in Figure 11.18(a). The optimum beamformer is steered to an angle $\phi_s = 0^\circ$. Let us assume another signal is present at $\phi_1 = -20^\circ$ that was not part of the interference (not accounted for in the interference correlation matrix). The gain of the optimum beamformer at $\phi = -20^\circ$ is approximately -20 dB. If the strength of this signal is significantly greater than 20 dB, the optimum beamformer steered to $\phi_s = 0^\circ$ will pass this sidelobe signal with sufficient strength that we may erroneously conclude a signal is present at $\phi_s = 0^\circ$. This problem is commonly referred to as a *sidelobe target* or *desired signal problem*.

The sidelobe signal problem described above can be cured, at least partially, by reducing the sidelobe levels of the beamformer to levels that sufficiently reject these sidelobe signals. As we described in Section 11.2.2, the application of a taper to a spatial matched filter resulted in a low sidelobe beampattern. The same principle applies to the optimum beamformer. We define a tapered array response vector at an angle ϕ_s as

$$\mathbf{v}_t(\phi_s) = \mathbf{c}_{\text{tbf}}(\phi_s) = \mathbf{t} \odot \mathbf{c}_{\text{mf}}(\phi_s) \quad (11.3.30)$$

where \mathbf{t} is the tapering vector and \odot is the Hadamard or element-by-element product. The tapering vector is normalized such that $\mathbf{v}_t^H(\phi_s)\mathbf{v}_t(\phi_s) = 1$ as in (11.2.23). The resulting low sidelobe adaptive beamformer is given by substituting $\mathbf{v}_t(\phi_s)$ for $\mathbf{v}(\phi_s)$ in (11.3.15)

$$\mathbf{c}_{\text{to}} = \frac{\mathbf{R}_{i+n}^{-1}\mathbf{v}_t(\phi_s)}{\mathbf{v}_t^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}_t(\phi_s)} \quad (11.3.31)$$

We again use the Dolph-Chebyshev taper for illustration purposes because this choice of taper provides a uniform sidelobe level. Other choices include the window functions discussed in Chapter 5 in the context of spectrum estimation. Consider the optimum beamformer with an interferer at $\phi = 20^\circ$ from Figure 11.18(a) with a potential signal leaking through the sidelobe at $\phi = -20^\circ$. If instead we use a tapered optimum beamformer from (11.3.31) with a -50 -dB sidelobe taper, a potential signal at $\phi = -20^\circ$ receives a -50 -dB level of attenuation. Figure 11.18(b) shows the beampattern of this tapered optimum beamformer. The sidelobe levels are significantly reduced while the null on the interferer at $\phi = 20^\circ$ has been maintained.

The adaptive beamformer given by (11.3.31) is no longer optimal in any sense [unless it were somehow possible for our desired signal to be spatially matched to $\mathbf{v}_t(\phi_s)$]. However, the resulting adaptive beamformer still provides rejection of unwanted interferers via spatial nulling through the use of \mathbf{R}_{i+n}^{-1} in (11.3.31). In addition, the low sidelobe levels of the beampattern reject signals not contained in the interference that are present at angles other than the angle of look ϕ_s . The penalty to be paid for the robustness provided by these low sidelobes is a small tapering loss in the direction of the look ϕ_s given by

$$L_{\text{taper}} = |\mathbf{c}_{\text{to}}^H \mathbf{v}(\phi_s)|^2 = \left| \frac{\mathbf{v}_t^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s)}{\mathbf{v}_t^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}_t(\phi_s)} \right|^2 = \frac{|\mathbf{v}^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}_t(\phi_s)|^2}{\mathbf{v}_t^H(\phi_s)\mathbf{R}_{i+n}^{-2}\mathbf{v}_t(\phi_s)} \quad (11.3.32)$$

and a widening of the mainlobe beamwidth, as can be seen in the beampattern in Figure 11.18. This tapering loss indicates a mismatch between the true signal and the constraint in the optimum beamformer.

11.3.5 The Generalized Sidelobe Canceler

We have shown that the optimum MVDR beamformer maximizes the output SINR and can be formulated as a constrained optimization given by

$$\min \mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c} \quad \text{subject to} \quad \mathbf{c}^H \mathbf{v}(\phi_s) = 1 \quad (11.3.33)$$

which results in the MVDR beamformer weight vector

$$\mathbf{c}_o = \frac{\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s)}{\mathbf{v}^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s)} \quad (11.3.34)$$

This problem formulation can be broken up into constrained and unconstrained components that give rise to both an alternate implementation and a more intuitive interpretation of the optimum beamformer. The resulting structure, known as the *generalized sidelobe canceler* (GSC) (Griffiths and Jim 1982), uses a preprocessing stage to transform the optimization from constrained to unconstrained (Applebaum and Chapman 1976; Griffiths and Jim 1982). The GSC structure is illustrated in Figure 11.19.

Consider the array signal $\mathbf{x}(n)$ from (11.3.1) consisting of a signal component $\mathbf{s}(n)$ and an interference-plus-noise component $\mathbf{x}_{i+n}(n)$. We are interested in forming the optimum beamformer steered to the angle ϕ_s . Let us start by forming a nonadaptive spatial matched filter in this direction $\mathbf{c}_{\text{mf}} = \mathbf{v}(\phi_s)$. The resulting output is the main channel signal given by

$$y_0(n) = \mathbf{c}_{\text{mf}}^H(\phi_s)\mathbf{x}(n) = \mathbf{v}^H(\phi_s)\mathbf{x}(n) = s_0(n) + i_0(n) + w_0(n) \quad (11.3.35)$$

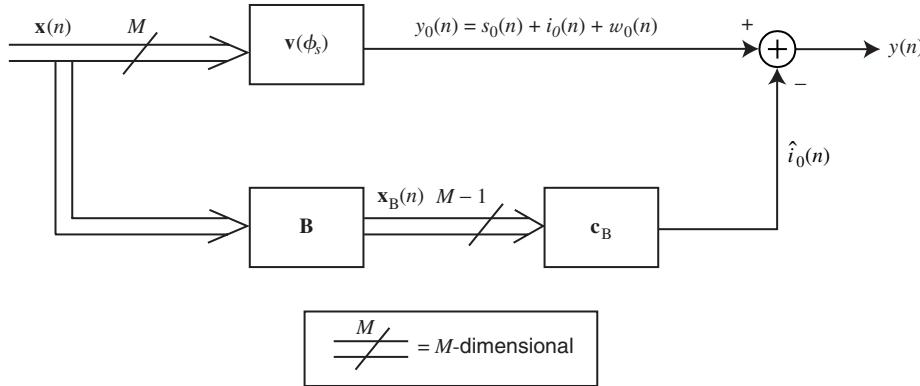


FIGURE 11.19
 Generalized sidelobe canceler.

This nonadaptive beamformer makes up the upper branch of the GSC. In addition, let us form a lower branch consisting of $M - 1$ channels in which the unconstrained optimization is performed. To prevent signal cancellation according to the unity-gain constraint in (11.3.33), we must ensure that these $M - 1$ channels do not contain any signals[†] from ϕ_s . To this end, we form an $(M - 1) \times M$ signal blocking matrix \mathbf{B} that is orthogonal to the look-direction constraint $\mathbf{v}(\phi_s)$

$$\mathbf{B}^H \mathbf{v}(\phi_s) = \mathbf{0} \quad (11.3.36)$$

The resulting output of the blocking matrix is the $(M - 1) \times 1$ vector signal

$$\mathbf{x}_B(n) = \mathbf{B}^H \mathbf{x}(n) \quad (11.3.37)$$

Thus, several choices for the blocking matrix exist that can perform this projection onto the $(M - 1)$ -dimensional subspace orthogonal to $\mathbf{v}(\phi_s)$. One choice uses a set of $M - 1$ beams that are each chosen to satisfy this constraint. The spatial frequency of the ULA for an angle ϕ_s is

$$u_s = \frac{d}{\lambda} \sin \phi_s \quad (11.3.38)$$

For $\mathbf{v}(\phi_s)$, spatial matched filters at the frequencies

$$u_m = u_s + \frac{m}{M} \quad (11.3.39)$$

for $m = 1, 2, \dots, M - 1$ are mutually orthogonal as well as orthogonal to $\mathbf{v}(\phi_s)$, that is,

$$\mathbf{v}^H(\phi_m) \mathbf{v}(\phi_s) = 0 \quad (11.3.40)$$

where ϕ_m is the angle corresponding to the spatial frequency u_m given by (11.3.39). Thus, we can construct a beampspace signal blocking matrix from these $M - 1$ steering vectors

$$\mathbf{B} = [\mathbf{v}(u_1) \mathbf{v}(u_2) \cdots \mathbf{v}(u_{M-1})] \quad (11.3.41)$$

An alternative signal blocking matrix can be implemented, assuming the array is presteered to the angle ϕ_s (Griffiths and Jim 1982). Presteering is accomplished by phase-shifting each element of the array by the corresponding steering vector element to this angle without actually forming a summation. Then any blocking matrix for which the elements of each column sum to zero will satisfy (11.3.36).

Once the nonadaptive preprocessing has been performed for the upper and lower branches of the GSC, an unconstrained optimization can be performed in the lower branch.

[†]Although the optimum beamformer was formulated for a signal-free interference-plus-noise correlation matrix \mathbf{R}_{i+n} , it is possible that the presence of the desired signal in the data is unavoidable.

Using the $M - 1$ channels in the lower branch, we want to estimate the undesired portion of the upper branch signal $y_0(n)$ due to interference. This interference is presumed to arrive at the array from a different angle than ϕ_s so that it must be contained in the lower branch signal as well. Thus, we need to compute an adaptive weight vector for the lower branch channels that forms an estimate of the interference in the upper branch. The estimated interference is subtracted from the upper branch. This problem is the classical MMSE filtering problem (see Chapter 6), whose solution is given by the Wiener-Hopf equation

$$\mathbf{c}_B = \mathbf{R}_B^{-1} \mathbf{r}_B \quad (11.3.42)$$

where $\mathbf{R}_B = E\{\mathbf{x}_B(n)\mathbf{x}_B^H(n)\}$ is the lower branch correlation matrix and $\mathbf{r}_B = E\{\mathbf{x}_B y_0^*(n)\}$ is the cross-correlation vector between the upper and lower branch signals. The resulting estimate of the upper branch interference signal is

$$\hat{i}_0(n) = \mathbf{c}_B^H \mathbf{x}_B(n) \quad (11.3.43)$$

and the output of the GSC is

$$y(n) = y_0(n) - \hat{i}_0(n) = y_0(n) - \mathbf{c}_B^H \mathbf{x}_B(n) \quad (11.3.44)$$

As we stated earlier, the GSC is equivalent to the optimum beamformer; that is, it maximizes the SINR at its output for signals arriving at angle ϕ_s . The power of the GSC formulation lies in its interpretation. Whereas for the optimum beamformer, the interference was canceled by forming spatial nulls in the directions of interferers, the GSC can be visualized as estimating the interference component in the upper branch from the lower-branch signals. Of course, the GSC also forms spatial nulls in the directions of the interferers. In terms of an alternate implementation, one must consider that if we are to steer the array to a number of different angles, each direction will require the formation of a new blocking matrix and the computation of a different correlation matrix and cross-correlation vector for the GSC. On the other hand, the optimum beamformer formulation has the same correlation matrix independent of the direction to which it is steered and, therefore, is often preferred for implementation purposes.

11.4 PERFORMANCE CONSIDERATIONS FOR OPTIMUM BEAMFORMERS

In this section we look at some considerations that influence the performance of an optimum beamformer. These considerations are also applicable to the adaptive methods in Section 11.5 that are derived from the optimum beamformer. Since the optimum beamformer serves as an upper bound on the performance of any adaptive method, these considerations can serve as adjustments to this performance bound for the adaptive counterparts to the optimum beamformer.

Two major factors that affect the performance of an optimum beamformer are:

- Mismatch of the actual signal to the assumed signal model used by the optimum beamformer
- Bandwidth of the signal that violates the narrowband assumption.

In the first section, we look at the effects of differences in the actual signal from that assumed for the optimum beamformer, known as *signal mismatch*. In virtually all array processing implementations, some level of mismatch will exist, due to either uncertainty in the exact angle of arrival of the signal of interest or the fact that the locations and characteristics of the individual sensors differ from our assumptions. As we will see, these errors that produce a signal mismatch can have profound implications on performance, particularly when the signal of interest is present in the correlation matrix. Next, we look at the effects of wider bandwidths on the performance of the optimum beamformer. In many applications, certain requirements necessitate the use of larger bandwidths. Their impact and possible means of correction are discussed in this section.

In our formulation of the optimum beamformer, we assumed that a signal arriving at the array from an angle ϕ_s would produce a response equal to the ideal steering vector for a ULA [see (11.1.19)]. Thus, the optimum beamformer constrained its response to be spatially “matched” to the array response of the signal $\mathbf{v}_s = \mathbf{v}(\phi_s) = \mathbf{v}_0$ where $\phi_0 = \phi_s$

$$\mathbf{c}_o^H \mathbf{v}_0 = \mathbf{c}_o^H \mathbf{v}_s = 1 \quad (11.4.1)$$

that is, to pass it with unity gain. The vector \mathbf{v}_0 is the assumed array response. However, in reality, the signal may exhibit a different response across the array or may arrive from another angle $\phi_s \neq \phi_0$. The differences in response arise due to distortion of the waveform during propagation, amplitude and phase mismatches between the individual sensors, or errors in the assumed locations of the sensors.[†] These mismatches manifest themselves in a deviation of the array response from that assumed for a ULA in (11.1.19). However, if the angle of arrival of the signal differs from the assumed angle, the result is an array response as in (11.1.19), but for the angle ϕ_s as opposed to the steering angle ϕ_0 . In either case, the beamformer is mismatched with the signal of interest and is no longer optimum. In this section, we examine the effect of these mismatches on the performance of the optimum beamformer, for the case of the signal of interest contained in the correlation matrix and absent from it. As we will see, the inclusion of this signal of interest in the correlation matrix has profound implications on the performance of a mismatched optimum beamformer. The analysis that follows was originally reported by Cox (1973).

Consider the case of an array signal consisting of a signal of interest $\mathbf{s}(n)$, interference $\mathbf{i}(n)$, and thermal noise $\mathbf{w}(n)$

$$\mathbf{x}(n) = \mathbf{s}(n) + \mathbf{i}(n) + \mathbf{w}(n) \quad (11.4.2)$$

where the noise is assumed to be uncorrelated, that is, $\mathbf{R}_n = \sigma_w^2 \mathbf{I}$. Now let us assume that the signal of interest is given by

$$\mathbf{s}(n) = \sqrt{M} s(n) \mathbf{u}_s \quad (11.4.3)$$

where \mathbf{u}_s , with unit norm ($\mathbf{u}_s^H \mathbf{u}_s = 1$), is the true array response to the signal of interest. For generality, \mathbf{u}_s may be either an ideal or a nonideal array response for a ULA of a signal arriving from angle ϕ_s , but in either case it is mismatched with the assumed response

$$\mathbf{u}_s \neq \mathbf{v}_0 \quad (11.4.4)$$

The correlation matrix of the signal $\mathbf{x}(n)$ is made up of components due to the signal and the interference-plus-noise

$$\mathbf{R}_x = E\{\mathbf{x}(n)\mathbf{x}^H(n)\} = M\sigma_s^2 \mathbf{u}_s \mathbf{u}_s^H + \mathbf{R}_{i+n} \quad (11.4.5)$$

where the signal power is $\sigma_s^2 = |s(n)|^2$. The optimum beamformer with an MVDR constraint for the signal $\mathbf{s}(n)$ in (11.3.15) is

$$\mathbf{c}_o = \frac{\mathbf{R}_{i+n}^{-1} \mathbf{u}_s}{\mathbf{u}_s^H \mathbf{R}_{i+n}^{-1} \mathbf{u}_s} \quad (11.4.6)$$

However, the true array response \mathbf{u}_s is unknown. This optimum beamformer in (11.4.6) yields the maximum output SINR given by

$$\text{SINR}_o = M\sigma_s^2 \mathbf{u}_s^H \mathbf{R}_{i+n}^{-1} \mathbf{u}_s = \text{SNR}_0 \cdot L_{\text{sinr}} \quad (11.4.7)$$

where $\text{SNR}_0 = M\sigma_s^2/\sigma_w^2$ is the matched filter SNR in the absence of interference from (11.2.18) (best performance possible) and $L_{\text{sinr}} = \sigma_w^2 \mathbf{u}_s^H \mathbf{R}_{i+n}^{-1} \mathbf{u}_s$ is the SINR loss from

[†]Similar losses also result from using a tapered steering vector. This loss was shown for the tapered optimum beamformer for the case of the signal of interest not present in the correlation matrix. As we will show in this section, the inclusion of the signal of interest in the correlation matrix can cause substantial losses in such a tapered beamformer.

(11.3.18) due to the presence of the interference. Thus, we can evaluate the losses due to signal mismatch and the inclusion of the signal of interest in the correlation matrix with respect to the maximum SINR in (11.4.7).

Loss due to signal mismatch

First, let us consider a mismatched signal $\mathbf{v}_0 \neq \mathbf{u}_s$ without the signal of interest present in the correlation matrix. The mismatch arises due to our lack of knowledge of the true array response to the signal of interest \mathbf{u}_s . The computation of the beamformer weights, assuming the array response to the signal to be \mathbf{v}_0 with an MVDR normalization, is given by

$$\mathbf{c}_1 = \frac{\mathbf{R}_{i+n}^{-1} \mathbf{v}_0}{\mathbf{v}_0^H \mathbf{R}_{i+n}^{-1} \mathbf{v}_0} \quad (11.4.8)$$

The SINR at the beamformer output for this weight vector is given by

$$\begin{aligned} \text{SINR}_1 &= \frac{|\mathbf{c}_1^H \mathbf{s}(n)|^2}{\mathbf{c}_1^H \mathbf{R}_{i+n} \mathbf{c}_1} = M \sigma_s^2 \frac{|\mathbf{v}_0^H \mathbf{R}_{i+n}^{-1} \mathbf{u}_s|^2}{\mathbf{v}_0^H \mathbf{R}_{i+n}^{-1} \mathbf{v}_0} \\ &= M \sigma_s^2 \mathbf{u}_s^H \mathbf{R}_{i+n}^{-1} \mathbf{u}_s \frac{|\mathbf{v}_0^H \mathbf{R}_{i+n}^{-1} \mathbf{u}_s|^2}{(\mathbf{v}_0^H \mathbf{R}_{i+n}^{-1} \mathbf{v}_0)(\mathbf{u}_s^H \mathbf{R}_{i+n}^{-1} \mathbf{u}_s)} \\ &= \text{SINR}_o \cdot \cos^2(\mathbf{v}_0, \mathbf{u}_s; \mathbf{R}_{i+n}^{-1}) \end{aligned} \quad (11.4.9)$$

where the term $\cos(\cdot)$ measures the cosine of a generalized angle between two vectors \mathbf{a} and \mathbf{b} weighted by matrix \mathbf{Z} (Cox 1973)

$$\cos^2(\mathbf{a}, \mathbf{b}; \mathbf{Z}) \triangleq \frac{|\mathbf{a}^H \mathbf{Z} \mathbf{b}|^2}{(\mathbf{a}^H \mathbf{Z} \mathbf{a})(\mathbf{b}^H \mathbf{Z} \mathbf{b})} \quad (11.4.10)$$

This term can be shown to have limits of $0 \leq \cos^2(\mathbf{a}, \mathbf{b}; \mathbf{Z}) \leq 1$ through the Schwartz inequality. The SINR from (11.4.9) can be rewritten as

$$\text{SINR}_1 = \text{SNR}_0 \cdot L_{\text{sinr}} \cdot L_{\text{sm}} \quad (11.4.11)$$

where we define the signal mismatch (sm) loss to be

$$L_{\text{sm}} \triangleq \cos^2(\mathbf{v}_0, \mathbf{u}_s; \mathbf{R}_{i+n}^{-1}) \quad (11.4.12)$$

Therefore, the SINR in (11.4.9) is a result of reducing the maximum SNR for a matched filter by the SINR loss due to the interference L_{sinr} as well as the loss due to the mismatch L_{sm} .

To gain some insight into the loss due to mismatch, consider the eigendecomposition of \mathbf{R}_{i+n}^{-1} given by

$$\mathbf{R}_{i+n}^{-1} = \sum_{m=1}^M \frac{1}{\lambda_m} \mathbf{q}_m \mathbf{q}_m^H \quad (11.4.13)$$

where λ_m and \mathbf{q}_m are the eigenvalue and eigenvector pairs, respectively. The largest eigenvalues and their corresponding eigenvectors are due to interference, while the small eigenvalues and eigenvectors are due to noise only. Since the eigenvectors form a basis for the M -dimensional vector space, any vector, say, \mathbf{v}_0 or \mathbf{u}_s , can be written as a linear combination of these eigenvectors. The product of the matrix \mathbf{R}_{i+n}^{-1} with any vector closely aligned with an interference eigenvector will suffer significant degradation. Therefore, the mismatch loss in (11.4.12) should be relatively small for the case of \mathbf{u}_s not closely aligned with interferers. Otherwise, if the signal lies near any of the interference eigenvectors, the beamformer will be more sensitive to signal mismatch.

Intuitively, performance degradation due to a mismatch in the optimum beamformer is relatively insensitive for small mismatches. The beamformer in (11.4.8) attempts to remove

any energy that is not contained in its unity-gain constraint for \mathbf{v}_0 . Since the signal with an array response \mathbf{u}_s is not contained in the correlation matrix, the only losses incurred are due to the degree of mismatch between \mathbf{u}_s and \mathbf{v}_0 and the similarity of \mathbf{u}_s to interference components that are nulled through the use of \mathbf{R}_{i+n}^{-1} . However, most importantly, the loss due to mismatch is independent of the signal strength σ_s^2 .

Loss due to signal in the correlation matrix

To implement the optimum beamformer in (11.4.6) in practice, we must assume that we can estimate \mathbf{R}_{i+n} without the presence of the signal $\mathbf{s}(n)$. However, in many applications the signal is present all the time so that an estimate of a signal-free correlation matrix is not possible. In this case, the optimum beamformer must be constructed with the correlation matrix from (11.4.5) and is given by

$$\mathbf{c}_2 = \frac{\mathbf{R}_x^{-1}\mathbf{v}_0}{\mathbf{v}_0^H\mathbf{R}_x^{-1}\mathbf{v}_0} \quad (11.4.14)$$

Although this beamformer differs from the beamformer \mathbf{c}_1 in (11.4.8) that does not include the signal of interest in the correlation matrix, it produces an identical beamforming weight vector in the case when it is perfectly matched to the signal of interest, that is, $\mathbf{v}_0 = \mathbf{u}_s$ (see Problem 11.10). Thus, the beamformer in (11.4.14) also maximizes the SINR in the case of a perfectly matched signal. However, we want to examine the sensitivity of this beamformer to signal mismatches. The SINR of the beamformer from (11.4.14) with the signal present (sp) in the correlation matrix can be shown to be (Cox 1973)

$$\begin{aligned} \text{SINR}_2 &= \frac{|\mathbf{c}_2^H \mathbf{s}(n)|^2}{\mathbf{c}_2^H \mathbf{R}_{i+n} \mathbf{c}_2} = M\sigma_s^2 \frac{|\mathbf{v}_0^H \mathbf{R}_x^{-1} \mathbf{u}_s|^2}{\mathbf{v}_0^H \mathbf{R}_x^{-1} \mathbf{R}_{i+n} \mathbf{R}_x^{-1} \mathbf{v}_0} \\ &= \frac{\text{SINR}_1}{1 + (2\text{SINR}_o + \text{SINR}_o^2) \cdot \sin^2(\mathbf{v}_0, \mathbf{u}_s; \mathbf{R}_{i+n}^{-1})} \\ &= \text{SNR}_0 \cdot L_{\text{sinr}} \cdot L_{\text{sm}} \cdot L_{\text{sp}} \end{aligned} \quad (11.4.15)$$

where SINR_1 is the SINR of the mismatched beamformer in (11.4.9). The $\sin(\cdot)$ term measures the sine of the generalized angle between \mathbf{v}_0 and \mathbf{u}_s and is related to the $\cos(\cdot)$ term from (11.4.10) by

$$\sin^2(\mathbf{v}_0, \mathbf{u}_s; \mathbf{R}_{i+n}^{-1}) = 1 - \cos^2(\mathbf{v}_0, \mathbf{u}_s; \mathbf{R}_{i+n}^{-1}) \quad (11.4.16)$$

Thus, the SINR of a beamformer constructed with the signal of interest present in the correlation matrix suffers an additional loss L_{sp} , beyond the losses associated with the interference L_{sinr} and the mismatch L_{sm} between \mathbf{u}_s and \mathbf{v}_0 , which is given by

$$L_{\text{sp}} = \frac{1}{1 + (2\text{SINR}_o + \text{SINR}_o^2) \sin^2(\mathbf{v}_0, \mathbf{u}_s; \mathbf{R}_{i+n}^{-1})} \quad (11.4.17)$$

Unlike the mismatch loss from (11.4.12), the loss due to the signal presence in the correlation matrix with signal mismatch is related to the signal strength σ_s^2 . In fact, (11.4.17) shows a strong dependence on the signal strength through the terms SINR_o and SINR_o^2 in the denominator. This dependence on signal strength is weighted by the sine term in (11.4.16) that measures the amount of mismatch. Thus for large signals, the losses can be significant. In fact, it can be shown that the losses resulting from strong signals present in the correlation matrix can cause the output SINR to be lower than if the signal had been relatively weak. This phenomenon along with possible means of alleviating the losses is explored in Problem 11.11.

We have shown a high sensitivity to mismatch of strong signals of interest when they are present in the correlation matrix used to compute the beamforming weights in (11.4.14). Since, in practice, a certain level of mismatch is always present, it may sometimes be advisable to use conventional, nonadaptive beamformers when the signal is present at

all times and does not allow the estimation of a signal-free correlation matrix \mathbf{R}_{i+n} . If the performance of such nonadaptive beamformers is deemed unacceptable, then special measures such as diagonal loading, which is described in Section 11.5.2, must be taken to design a robust beamformer that is less sensitive to mismatch (Cox et al. 1987).

11.4.2 Effect of Bandwidth

So far, we have relied on the narrowband assumption, meaning that the bandwidth B of the received signals is small with respect to the carrier frequency F_c . Previously, we gave a rule of thumb for this assumption, namely, that the *fractional bandwidth*, defined as

$$\bar{B} = \frac{B}{F_c} \quad (11.4.18)$$

is small, say, $\bar{B} \ll 1$ percent. Another measure is the space-time-bandwidth product, which for an array of length L is

$$\text{TBWP} = \frac{LB}{c} \quad (11.4.19)$$

where the time L/c is the maximum amount of time for a plane wave to propagate across the entire array, that is, the maximum propagation delay between the first and last elements ($\phi = \pm 90^\circ$, $\sin \phi = 1$).

However, many real-world applications require increased bandwidths, which cause this assumption to be violated (Buckley 1987; Zatman 1998). The question then is, What is the effect of bandwidth on the performance of an array? Let us begin by examining the narrowband steering vector for a ULA from (11.1.19)

$$\mathbf{v}(\phi) = \frac{1}{\sqrt{M}} [1 \ e^{-j2\pi[(d \sin \phi)/\lambda]} \ \dots \ e^{-j2\pi[(d \sin \phi)/\lambda](M-1)}]^T \quad (11.4.20)$$

which assumes that λ is constant, that is, the array receives signals only from a frequency F_c . Relaxing this assumption and substituting $\lambda = c/F$ from (11.1.2) gives us a steering vector that makes no assumptions about the bandwidth of the incoming signal

$$\mathbf{v}(\phi, F) = \frac{1}{\sqrt{M}} [1 \ e^{-j2\pi[(d \sin \phi)/c]F} \ \dots \ e^{-j2\pi[(d \sin \phi)/c](M-1)F}]^T \quad (11.4.21)$$

When we demodulate the received signals by the carrier frequency F_c , we are making an implicit narrowband assumption that allows us to model the time delay between sensor elements as a phase shift. Therefore, a wideband signal arriving from an angle ϕ appears to the narrowband receiver as if it were arriving from an angular region centered at ϕ (provided the spectrum of the incoming signal is centered about F_c), since the approximation of the delay between elements as a single phase shift no longer holds. This phenomenon is known as *dispersion* since the incoming wideband signal appears to disperse in angle across the array.

Let us examine the impact of a wideband interference signal on the performance of an adaptive array. The correlation matrix of a single interference source impinging on the array from an angle ϕ is found by integrating over the bandwidth of the received signal

$$\mathbf{R}_i = \frac{\sigma_p^2}{B} \int_{F_c-B/2}^{F_c+B/2} \mathbf{v}(\phi, F) \mathbf{v}^H(\phi, F) dF \quad (11.4.22)$$

where the assumption is made that the spectral response of the signal is flat over the bandwidth, that is, $|R(F)|^2 = 1$ for $F_c - B/2 \leq F \leq F_c + B/2$. Now, focusing on the individual elements of the correlation matrix, namely the (m, n) th element

$$\begin{aligned}
 \langle \mathbf{R}_i \rangle_{m,n} &= \frac{\sigma_p^2}{B} \int_{F_c-B/2}^{F_c+B/2} e^{j2\pi m[(d \sin \phi)/c]F} e^{-j2\pi n[(d \sin \phi)/c]F} dF \\
 &= \frac{\sigma_p^2}{B} \int_{F_c-B/2}^{F_c+B/2} e^{j2\pi(m-n)[(d \sin \phi)/c]F} dF \\
 &= \sigma_p^2 e^{j2\pi(m-n)[(d \sin \phi)/c]F_c} \frac{2 \sin \left[2\pi(m-n) \frac{d \sin \phi}{c} \frac{B}{2} \right]}{2\pi(m-n) \frac{d \sin \phi}{c} B} \\
 &= \sigma_p^2 e^{j2\pi(m-n)[(d \sin \phi)/\lambda]} \text{sinc} \left[(m-n) \frac{d \sin \phi}{c} B \right]
 \end{aligned} \tag{11.4.23}$$

where $\text{sinc}(x) = \sin(\pi x)/(\pi x)$. We notice that each element is made up of two terms. The first term is simply the cross-correlation between the m th and n th sensor array elements for a narrowband signal arriving from ϕ

$$\langle \mathbf{R}_i^{(\text{nb})} \rangle_{m,n} = \sigma_p^2 e^{j2\pi(m-n)[(d \sin \phi)/\lambda]} \tag{11.4.24}$$

where the superscript indicates that this is the narrowband correlation matrix. The second term represents the dispersion across the array caused by the bandwidth of the interferer and is given by

$$\langle \mathbf{R}_d \rangle_{m,n} = \text{sinc} \left[(m-n) \frac{d \sin \phi}{c} B \right] \tag{11.4.25}$$

Using (11.4.25), we can construct a matrix that models this dispersion across the entire array, which we refer to as the *dispersion matrix*. Dispersion creates decorrelation of the signal across the array, and this term represents the temporal autocorrelation of the impinging signal. Therefore, we can write the wideband correlation matrix as the Hadamard product of the narrowband correlation and the dispersion matrices

$$\mathbf{R}_i^{(\text{wb})} = \mathbf{R}_i^{(\text{nb})} \odot \mathbf{R}_d \tag{11.4.26}$$

where the Hadamard product is a point-by-point multiplication (Strang 1998).

The dispersion produced by a wideband signal can be compensated or corrected for at its specific angle by using a technique known as *time-delay steering*. Notice that the dispersion term in (11.4.25) is $\langle \mathbf{R}_d \rangle_{m,n} = 1$ for $\phi = 0^\circ$ since the argument of the sinc function is zero. Therefore, for signals arriving from $\phi = 0^\circ$ no dispersion can occur. In other words, for signals arriving from broadside to the array ($\phi = 0^\circ$), the delay between elements is zero, independent of the frequency of the signal or its bandwidth. The dispersion becomes worse as the value of the angle is increased. This suggests a simple remedy to correct for dispersion: refocus the array to angle ϕ . Steering the array in this direction involves time-delays each element to compensate for its delay between elements explicitly. This time-delay steering can be implemented in analog or digitally and is illustrated in Figure 11.20. The time-delay steered array signal is

$$\mathbf{x}_{\text{td}}(t) = [x_1(t) \ x_2(t - \tau_2) \ \cdots \ x_M(t - \tau_M)]^T \tag{11.4.27}$$

where $\tau_m = (d/\lambda)(m-1) \sin \phi$ with λ being the wavelength of the center frequency F_c . Thus, a signal arriving from this angle will have no delay between elements following the time-delay steering. A convenient means of modeling time-delay steering in the discrete-time signal is through application of the matrix

$$\mathbf{V} = \text{diag}\{\mathbf{v}(\phi)\} \tag{11.4.28}$$

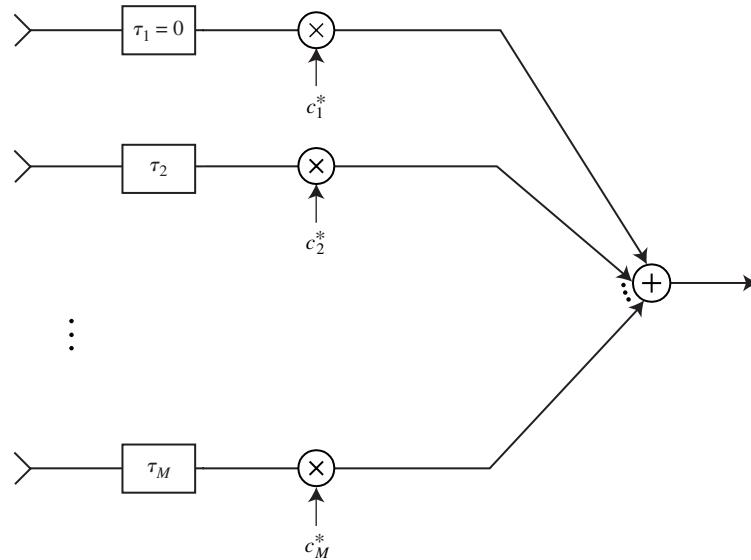


FIGURE 11.20

Time-delay steering prior to beamforming (referenced to the first element, $\tau_1 = 0$).

to the array signal $\mathbf{x}(n)$ as

$$\mathbf{x}_{\text{td}}(n) = \mathbf{V}^H \mathbf{x}(n) \quad (11.4.29)$$

The resulting interference correlation matrix is

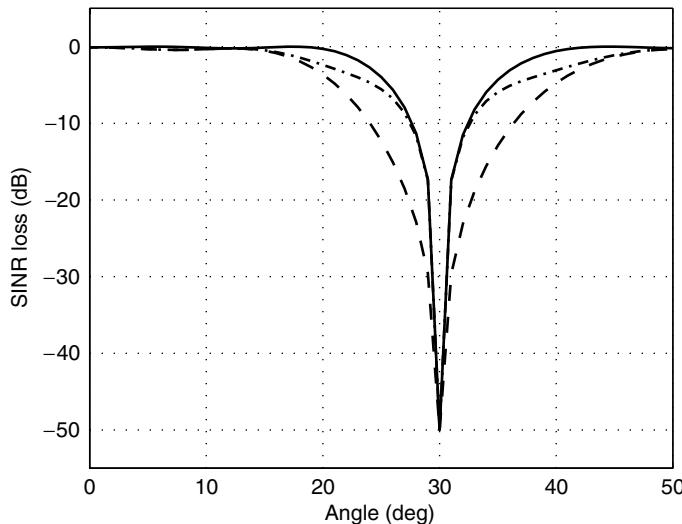
$$\mathbf{R}_i^{(td)} = \mathbf{V}^H \mathbf{R}_i^{(wb)} \mathbf{V} \quad (11.4.30)$$

Time-delay steering will focus signals from angle ϕ but may in fact increase the amount of dispersion from other angles. However, if we are not looking at these other angles, this effect may not be noticed. The underlying phenomenon that is occurring is that an optimum beamformer is forced to use additional adaptive degrees of freedom to cancel the dispersed, wideband interference signals. As long as the optimum beamformer has sufficient degrees of freedom, the effect of dispersion at other angles may not be evident.

EXAMPLE 11.4.1. Consider the radar interference scenario with a single jammer at an angle $\phi = 30^\circ$ with a jammer-to-noise ratio JNR = 50 dB. Again, we have an $M = 10$ element array with $\lambda/2$ spacing. The center frequency of the array is $F_c = 1$ GHz, and the bandwidth is $B = 10$ MHz for a fractional bandwidth of $\bar{B} = 1$ percent. The SINR loss of an optimum beamformer is found by substituting the wideband correlation matrix from (11.4.26) into the SINR loss in (11.3.18)

$$L_{\text{sinr}}(\phi_s) = \mathbf{v}^H(\phi_s) [\mathbf{R}_{i+n}^{(\text{wb})}]^{-1} \mathbf{v}(\phi_s) \quad (11.4.31)$$

where the wideband interference-plus-noise correlation matrix is $\mathbf{R}_{i+n}^{(wb)} = \mathbf{R}_i^{(wb)} + \sigma_w^2 \mathbf{I}$ since the thermal noise is uncorrelated. Scanning across all angles, we can compute the SINR loss, which is shown in Figure 11.21 along with the SINR loss had the signal been narrowband. Notice the increased width of the SINR loss notch centered about $\phi = 30^\circ$, which corresponds to a dropoff in performance in the vicinity of the jammer with respect to the narrowband case. However, at angles farther from the jammer there is no impact on performance; that is, $L_{\text{sir}}(\phi_s) \approx 0$ dB. Next we look at the performance of an optimum beamformer that incorporates time-delay steering prior to adaptation. In this case, using $\mathbf{R}_i^{(td)} + \sigma_w^2 \mathbf{I}$ from (11.4.30) in place of $\mathbf{R}_{i+n}^{(wb)}$ in the SINR loss equation, we can compute the SINR loss of the optimum beamformer using time-delay steering, which is also plotted in Figure 11.21. The notch around the jammer at $\phi = 30^\circ$ has been restored to the narrowband case for angles immediately surrounding $\phi = 30^\circ$. At the

**FIGURE 11.21**

SINR loss for wideband jammer with JNR = 50 dB at an angle of $\phi = 30^\circ$. The carrier frequency is $F_c = 1$ GHz, and the bandwidth is $B = 10$ MHz. Solid line is the narrowband signal, dashed line is the wideband signal, and dash-dot line is the wideband signal with time-delay steering.

angles a little farther away, the performance is still worse than that for the narrowband case but still significantly better than that without time-delay steering.

11.5 ADAPTIVE BEAMFORMING

So far, we have only considered the optimum beamformer but have not concerned ourselves with how such a beamformer would be implemented in practice. Optimality was only achieved because we assumed perfect knowledge of the second-order statistics of the interference at the array, that is, the interference-plus-noise correlation matrix \mathbf{R}_{i+n} . In this section, we describe the use of adaptive methods that are based on collected data from which the correlation matrix is estimated. We look at two types of methods: block adaptive and sample-by-sample adaptive. A block adaptive implementation of the optimum beamformer uses a “block” of data to estimate the adaptive beamforming weight vector and is known as *sample matrix inversion (SMI)*. The SMI adaptive beamformer is examined in Section 11.5.1 along with the sidelobe levels and training issues associated with the SMI adaptive beamformer. Next we introduce the use of diagonal loading within the context of the block adaptive SMI beamformer in Section 11.5.2. In Section 11.5.3, we discuss sample-by-sample adaptive methods. These methods, as the block adaptive methods, base their estimates of the statistics on the data, but update these statistics with each new sample and are extensions of the adaptive filtering techniques from Chapter 10 to array processing.

11.5.1 Sample Matrix Inversion

In practice, the correlations are unknown and must be estimated from the data. Thus, we turn to the maximum-likelihood (ML) estimate of the correlation matrix given by the average of outer products of the array snapshots (Goodman 1963)

$$\hat{\mathbf{R}}_{i+n} = \frac{1}{K} \sum_{k=1}^K \mathbf{x}_{i+n}(n_k) \mathbf{x}_{i+n}^H(n_k) \quad (11.5.1)$$

where the indices n_k define the K samples of $\mathbf{x}_{i+n}(n)$ for $1 \leq n \leq N$ that make up the *training set*. Many applications may dictate that the collected snapshots be split into training data and data to be processed. The ML estimate of the correlation matrix implies that as $K \rightarrow \infty$, then $\hat{\mathbf{R}}_{i+n} \rightarrow \mathbf{R}_{i+n}$; and it is known as the *sample correlation matrix*. The total number of snapshots K used to compute the sample correlation matrix is referred to as the *sample support*. The larger the sample support, the better the estimate $\hat{\mathbf{R}}_{i+n}$ of the correlation matrix for stationary data. Proceeding by substituting the sample correlation matrix from (11.5.1) into the optimum beamformer weight computation in (11.3.15) results in the adaptive beamformer (Reed et al. 1974)

$$\mathbf{c}_{\text{sni}} = \frac{\hat{\mathbf{R}}_{i+n}^{-1} \mathbf{v}(\phi_s)}{\mathbf{v}^H(\phi_s) \hat{\mathbf{R}}_{i+n}^{-1} \mathbf{v}(\phi_s)} \quad (11.5.2)$$

known as the *sample matrix inversion* adaptive beamformer.[†] As for the optimum beamformer, an SMI adaptive beamformer can be implemented with low sidelobe control through the use of tapers. Simply substitute a tapered steering vector from (11.3.20) for $\mathbf{v}(\phi_s)$ in (11.5.2)

$$\mathbf{c}_{\text{tsni}} = \frac{\hat{\mathbf{R}}_{i+n}^{-1} \mathbf{v}_t(\phi_s)}{\mathbf{v}_t^H(\phi_s) \hat{\mathbf{R}}_{i+n}^{-1} \mathbf{v}_t(\phi_s)} \quad (11.5.3)$$

Similarly, all the adaptive processing methods that will be discussed in Section 11.6, that is, the linearly constrained beamformer, all the partially adaptive beamformers, and the sidelobe canceler, can be implemented in a similar fashion by substituting the appropriate sample correlation matrix for its theoretical counterpart.

Of course, we cannot expect to substitute an estimate $\hat{\mathbf{R}}_{i+n}$ of the true correlation matrix \mathbf{R}_{i+n} into the adaptive weight equation without experiencing a loss in performance. We begin by computing the output SINR of the SMI adaptive beamformer

$$\begin{aligned} \text{SINR}_{\text{sni}} &= \frac{M\sigma_s^2 |\mathbf{c}_{\text{sni}}^H \mathbf{v}(\phi_s)|^2}{E\{|\mathbf{c}_{\text{sni}}^H \mathbf{x}_{i+n}(n)|^2\}} = \frac{M\sigma_s^2 |\mathbf{c}_{\text{sni}}^H \mathbf{v}(\phi_s)|^2}{\mathbf{c}_{\text{sni}}^H \hat{\mathbf{R}}_{i+n} \mathbf{c}_{\text{sni}}} \\ &= M\sigma_s^2 \frac{[\mathbf{v}^H(\phi_s) \hat{\mathbf{R}}_{i+n}^{-1} \mathbf{v}(\phi_s)]^2}{\mathbf{v}^H(\phi_s) \hat{\mathbf{R}}_{i+n}^{-1} \hat{\mathbf{R}}_{i+n} \hat{\mathbf{R}}_{i+n}^{-1} \mathbf{v}(\phi_s)} \end{aligned} \quad (11.5.4)$$

Comparing this to the SINR obtained with the optimum beamformer from (11.3.12), we obtain the loss associated with the SMI adaptive beamformer relative to the optimum beamformer

$$L_{\text{sni}} = \frac{\text{SINR}_{\text{sni}}}{\text{SINR}_o} = \frac{[\mathbf{v}^H(\phi_s) \hat{\mathbf{R}}_{i+n}^{-1} \mathbf{v}(\phi_s)]^2}{[\mathbf{v}^H(\phi_s) \hat{\mathbf{R}}_{i+n}^{-1} \mathbf{R}_{i+n} \hat{\mathbf{R}}_{i+n}^{-1} \mathbf{v}(\phi_s)][\mathbf{v}^H(\phi_s) \hat{\mathbf{R}}_{i+n}^{-1} \mathbf{v}(\phi_s)]} \quad (11.5.5)$$

This SMI loss is dependent on the array data used to compute $\hat{\mathbf{R}}_{i+n}$, which implies that L_{sni} , like the data, is a random variable. In fact, it can be shown that L_{sni} follows a beta distribution given by (Reed et al. 1974)

$$p_\beta(L_{\text{sni}}) = \frac{K!}{(M-2)!(K+1-M)!} (1 - L_{\text{sni}})^{M-2} (L_{\text{sni}})^{K+1-M} \quad (11.5.6)$$

assuming a complex Gaussian distribution for the sensor thermal noise and the interference signals. Here M is the number of sensors in the array, and K is the number of snapshots

[†]An adaptive beamformer that is very similar to the SMI adaptive beamformer is known as the *adaptive matched filter (AMF)* (Robey et al. 1992). The difference between the two is actually in the normalization. The AMF requires $\mathbf{c}^H \hat{\mathbf{R}}_{i+n} \mathbf{c} = 1$ rather than $\mathbf{c}^H \mathbf{v}(\phi_s) = 1$ so that the interference-plus-noise has unit power at the beamformer output. As a result, it is straightforward to choose a detection threshold for the output of an AMF beamformer. For this reason, this method is discussed primarily within the context of *adaptive detection*. It is straightforward to show that the relation between the AMF and SMI adaptive weights, as they are defined in (11.5.2), is $\mathbf{c}_{\text{amf}} = [\mathbf{v}^H(\phi_s) \hat{\mathbf{R}}_{i+n}^{-1} \mathbf{v}(\phi_s)]^{1/2} \mathbf{c}_{\text{sni}}$.

used to estimate \mathbf{R}_{i+n} . Taking the expectation of this loss yields

$$E\{L_{smi}\} = \frac{K+2-M}{K+1} \quad (11.5.7)$$

which can be used to determine the sample support required to limit the losses due to correlation matrix estimation to a level considered acceptable. From (11.5.7), we can deduce the SMI loss will be approximately -3 dB for $K = 2M$ and approximately -1 dB for $K = 5M$.

EXAMPLE 11.5.1. In this example, we study the SMI adaptive beamformer and the loss associated with the number of snapshots used for training. SMI adaptive beamformers are produced with sample supports of $K = 1.5M, 2M$, and $5M$. Consider a ULA with $M = 20$ elements with an interference source at $\phi_i = 20^\circ$ and a power of 50 dB. The thermal noise has unit variance $\sigma_w^2 = 1$. We can generate the interference-plus-noise signal \mathbf{x}_{i+n} as

```
v_i = exp(-j*pi*[0:M-1]'.*sin(phi_i*pi/180))/sqrt(M);
x_ipn=(10^(40/20))*v_i*(randn(1,N)+j*randn(1,N))/sqrt(2) + ...
(randn(1,N)+j*randn(1,N))/sqrt(2);
```

The sample correlation matrix is then found from (11.5.2). We compute the SINR at an angle of ϕ by first computing the SMI adaptive weight vector from (11.5.3), and the SINR from (11.5.4) using the actual correlation matrix \mathbf{R}_{i+n} , computed by

```
R_ipn = (10^(40/10))*v_i*v_i' + eye(M);
```

and a signal of interest with $M\sigma_s^2 = 1$. We repeat this across all angles $-90^\circ \leq \phi < 90^\circ$ and average over 100 realizations of \mathbf{x}_{i+n} . The resulting average SINR for the various sample supports is shown in Figure 11.22 along with the SINR for the optimum beamformer computed from (11.3.12). Note that for a signal of interest with $M\sigma_s^2 = 1$ and unit-variance noise, the SINR of the optimum beamformer is equal to its SINR loss. The jammer null is at $\phi = 20^\circ$, as expected for all the beamformers. However, we notice that the SINR of the SMI adaptive beamformers is less than the optimum beamformer SINR by approximately 4, 3, and 1 dB for the sample supports of $K = 1.5M, 2M$, and $5M$, respectively. These losses are consistent with the SMI loss predicted by (11.5.7).

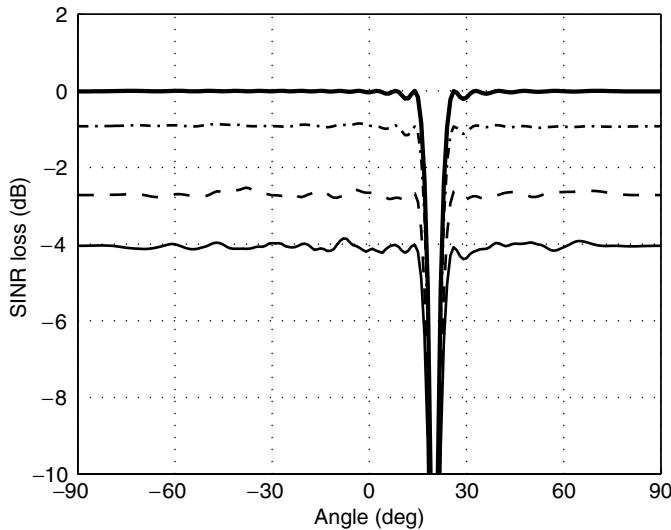
Sidelobe levels of the SMI adaptive beamformer

In addition to affecting the SINR of the beamformer output, the use of array snapshots to estimate \mathbf{R}_{i+n} has implications for the sidelobe levels of the resulting adaptive beamformer. The following analysis follows directly from Kelly (1989). Consider a signal received from a direction other than the direction of look ϕ_s . The response to such a signal determines the sidelobe level of adaptive beamformer at this angle. For the MVDR optimum beamformer from (11.3.15), the sidelobe level (SLL) at an angle ϕ_u is given by

$$\text{SLL}_o = |C_o(\phi_u)|^2 = \frac{|\mathbf{v}^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_u)|^2}{|\mathbf{v}^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s)|^2} \quad (11.5.8)$$

where ϕ_s is the beamformer steering angle or look direction. Likewise, we can also define the SINR of a signal $\mathbf{s}(n) = \sigma_u \mathbf{v}(\phi_u)$ received from an angle ϕ_u in the sidelobes of the optimum beamformer steered to ϕ_s

$$\begin{aligned} \text{SINR}_o(\phi_s, \phi_u) &= \frac{|\mathbf{c}_o^H(\phi_s)\mathbf{s}(n)|^2}{E\{|\mathbf{c}_o^H(\phi_s)\mathbf{x}_{i+n}(n)|^2\}} = \frac{\sigma_u^2 |\mathbf{v}^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_u)|^2}{\mathbf{v}^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s)} \\ &= \frac{\text{SINR}_o(\phi_u, \phi_u)|\mathbf{v}^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_u)|^2}{[\mathbf{v}^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s)][\mathbf{v}^H(\phi_u)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_u)]} \\ &= \text{SINR}_o(\phi_u, \phi_u) \cos^2(\mathbf{v}(\phi_s), \mathbf{v}(\phi_u); \mathbf{R}_{i+n}^{-1}) \end{aligned} \quad (11.5.9)$$

**FIGURE 11.22**

SINR loss for SMI adaptive beamformer with different numbers of training snapshots. Thin solid line has 30 snapshots ($K = 1.5M$), dashed line has 40 snapshots ($K = 2M$), and dash-dot line has 100 snapshots ($K = 5M$). Thick, solid line is SINR loss for the optimum beamformer.

since $\text{SINR}_o(\phi_u, \phi_u) = \sigma_u^2 \mathbf{v}^H(\phi_u) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_u)$, which is the maximum output SINR possible for a signal at angle ϕ_u , that is, the SINR if the optimum beamformer had been properly steered in this direction. The term

$$\begin{aligned} \cos(\mathbf{v}(\phi_s), \mathbf{v}(\phi_u); \mathbf{R}_{i+n}^{-1}) &= \frac{\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_u)}{[\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)]^{1/2} [\mathbf{v}^H(\phi_u) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_u)]^{1/2}} \\ &= \frac{\tilde{\mathbf{v}}^H(\phi_s) \tilde{\mathbf{v}}(\phi_u)}{[\tilde{\mathbf{v}}^H(\phi_s) \tilde{\mathbf{v}}(\phi_s)]^{1/2} [\tilde{\mathbf{v}}^H(\phi_u) \tilde{\mathbf{v}}(\phi_u)]^{1/2}} \end{aligned} \quad (11.5.10)$$

where

$$\tilde{\mathbf{v}}(\phi) = \mathbf{L}_{i+n}^{-1} \mathbf{v}(\phi) \quad (11.5.11)$$

measures the cosine of a generalized angle between vectors $\mathbf{v}(\phi_s)$ and $\mathbf{v}(\phi_u)$ (Cox 1973). This last quantity is the cosine of the angle between the whitened vectors $\tilde{\mathbf{v}}(\phi_s)$ and $\tilde{\mathbf{v}}(\phi_u)$ at the respective angles of ϕ_s and ϕ_u . The matrix \mathbf{L}_{i+n} is simply the Cholesky factor of the correlation matrix, that is, $\mathbf{R}_{i+n} = \mathbf{L}_{i+n} \mathbf{L}_{i+n}^H$. The sidelobe level of the optimum beamformer from (11.5.8) can also be written in terms of the SINR from (11.5.9)

$$\text{SLL}_o = |C_o(\phi_u)|^2 = \frac{\text{SINR}_o(\phi_s, \phi_u)}{\text{SINR}_o(\phi_s, \phi_s)} \quad (11.5.12)$$

$$\text{From (11.5.9), } \cos^2(\mathbf{v}(\phi_s), \mathbf{v}(\phi_u); \mathbf{R}_{i+n}^{-1}) = \frac{\text{SINR}_o(\phi_s, \phi_u)}{\text{SINR}_o(\phi_u, \phi_u)} \quad (11.5.13)$$

which is not the same as the sidelobe level in (11.5.12). However, this cosine term is a measure of the attenuation provided by an optimum beamformer steered in the direction ϕ_s as opposed to the maximum SINR provided by steering to angle ϕ_u . Thus, $\cos^2(\mathbf{v}(\phi_s), \mathbf{v}(\phi_u); \mathbf{R}_{i+n}^{-1})$ can be thought of as the sidelobe level at an angle ϕ_u of an optimum beamformer steered to ϕ_s in the absence of interference at ϕ_u . As a result, this term serves as an upper bound on the sidelobe level.

$$\begin{aligned}
\text{SINR}_{\text{smi}}(\phi_s, \phi_u) &= \frac{|\mathbf{c}_{\text{smi}}^H(\phi_s)\mathbf{s}(n)|^2}{E\{|\mathbf{c}_{\text{smi}}^H(\phi_s)\mathbf{x}_{i+n}(n)|^2\}} \\
&= \frac{\sigma_u^2 |\mathbf{v}^H(\phi_s)\hat{\mathbf{R}}_{i+n}^{-1}\mathbf{v}(\phi_u)|^2}{\mathbf{v}^H(\phi_s)\hat{\mathbf{R}}_{i+n}^{-1}\mathbf{R}_{i+n}\hat{\mathbf{R}}_{i+n}^{-1}\mathbf{v}(\phi_s)} \\
&= \text{SINR}_o(\phi_u, \phi_u) \\
&\quad \times \frac{|\mathbf{v}^H(\phi_s)\hat{\mathbf{R}}_{i+n}^{-1}\mathbf{v}(\phi_u)|^2}{[\mathbf{v}^H(\phi_s)\hat{\mathbf{R}}_{i+n}^{-1}\mathbf{R}_{i+n}\hat{\mathbf{R}}_{i+n}^{-1}\mathbf{v}(\phi_s)][\mathbf{v}^H(\phi_u)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_u)]} \\
&= \text{SINR}_o(\phi_u, \phi_u)L(\phi_s, \phi_u)
\end{aligned} \tag{11.5.14}$$

where

$$\begin{aligned}
L(\phi_s, \phi_u) &= \frac{\text{SINR}_{\text{smi}}(\phi_s, \phi_u)}{\text{SINR}_o(\phi_u, \phi_u)} \\
&= \frac{|\mathbf{v}^H(\phi_s)\hat{\mathbf{R}}_{i+n}^{-1}\mathbf{v}(\phi_u)|^2}{[\mathbf{v}^H(\phi_s)\hat{\mathbf{R}}_{i+n}^{-1}\mathbf{R}_{i+n}\hat{\mathbf{R}}_{i+n}^{-1}\mathbf{v}(\phi_s)][\mathbf{v}^H(\phi_u)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_u)]}
\end{aligned} \tag{11.5.15}$$

This term is bounded by $0 < L(\phi_s, \phi_u) < 1$ and can be interpreted as the loss of a signal received from the sidelobe angle ϕ_u processed with an SMI adaptive beamformer steered to ϕ_s relative to the maximum SINR possible for this signal. The term in the denominator of (11.5.15) is the SINR of the optimum, not the SMI adaptive beamformer. It is evident that as the number of array snapshots $K \rightarrow \infty$, $\hat{\mathbf{R}}_{i+n} \rightarrow \mathbf{R}_{i+n}$, $L(\phi_s, \phi_u) \rightarrow \cos^2(\mathbf{v}(\phi_s), \mathbf{v}(\phi_u); \mathbf{R}_{i+n}^{-1})$ from (11.5.15). The sidelobe level, however, of the SMI adaptive beamformer is

$$\text{SLL}_{\text{smi}} = |C_{\text{smi}}(\phi_u)|^2 = \frac{|\mathbf{v}^H(\phi_s)\hat{\mathbf{R}}_{i+n}^{-1}\mathbf{v}(\phi_u)|^2}{|\mathbf{v}^H(\phi_s)\hat{\mathbf{R}}_{i+n}^{-1}\mathbf{v}(\phi_s)|^2} \tag{11.5.16}$$

However, unlike the sidelobe level of the optimum beamformer in (11.5.12) which could be related to the SINR of signals in the sidelobes, such a relation does not hold for the SMI adaptive beamformer because

$$\mathbf{v}^H(\phi_s)\hat{\mathbf{R}}_{i+n}^{-1}\mathbf{R}_{i+n}\hat{\mathbf{R}}_{i+n}^{-1}\mathbf{v}(\phi_s) \neq \mathbf{v}^H(\phi_s)\hat{\mathbf{R}}_{i+n}^{-1}\mathbf{v}(\phi_s) \tag{11.5.17}$$

Asymptotically, this relation holds, but for finite sample support it does not. Nonetheless, we can draw some conclusions about the anticipated sidelobe levels using $L(\phi_s, \phi_u)$. The loss in SINR of the sidelobe signal $L(\phi_s, \phi_u)$ is a random variable with a probability distribution (Boroson 1980)

$$\begin{aligned}
p(L, \Theta) &= \sum_{j=0}^J \binom{J}{j} \cos^2(\mathbf{v}(\phi_s), \mathbf{v}(\phi_u); \mathbf{R}_{i+n}^{-1})^{J-j} \\
&\quad \times \sin^2(\mathbf{v}(\phi_s), \mathbf{v}(\phi_u); \mathbf{R}_{i+n})^j p_\beta(L, J+1, M-1)
\end{aligned} \tag{11.5.18}$$

where $\sin^2(\mathbf{v}(\phi_s), \mathbf{v}(\phi_u); \mathbf{R}_{i+n}) = 1 - \cos^2(\mathbf{v}(\phi_s), \mathbf{v}(\phi_u); \mathbf{R}_{i+n}^{-1})$. Recall that $\cos^2(\mathbf{v}(\phi_s), \mathbf{v}(\phi_u); \mathbf{R}_{i+n}^{-1})$ depends on the true correlation matrix. The term J is given by

$$J = K + 1 - M \tag{11.5.19}$$

and $p_\beta(x, l, m)$ is the beta probability distribution given by

$$p_\beta(x, l, m) = \frac{(l+m-1)!}{(l-1)!(m-1)!} x^{l-1} (1-x)^{m-1} \tag{11.5.20}$$

From this probability distribution, we can compute the expected value of the loss of a signal received in the sidelobes of the SMI adaptive beamformer

$$E\{L(\phi_s, \phi_u)\} = \frac{1}{K+1}[1 + (K+1-M)\cos^2(\mathbf{v}(\phi_s), \mathbf{v}(\phi_u); \mathbf{R}_{i+n}^{-1})] \quad (11.5.21)$$

For the case of perfect alignment ($\phi_u = \phi_s$), equation (11.5.21) measures the loss in SINR in the look direction since $\cos^2(\cdot) = 1$ and $E\{L(\phi_s, \phi_s)\} = L_{\text{smi}} = (K+2-M)/(K+1)$ from (11.5.21), which is the standard SMI SINR loss. In the opposite extreme, if ϕ_u is the angle of a null in the corresponding optimum beamformer, then $\cos^2(\cdot) = 0$ and

$$E\{L(\phi_s, \phi_u)\} = \frac{1}{K+1} \quad (11.5.22)$$

The expected value of this loss can be interpreted as a bound on the sidelobe level provided that no interference sources were present at angle ϕ_u . The implication of this equation is a lower bound on the sidelobe level that can be achieved by using an SMI adaptive beamformer. Note that all this analysis also applies for tapered SMI adaptive beamformers when we substitute $\mathbf{v}_t(\phi_s) \rightarrow \mathbf{v}(\phi_s)$ as we did for the weights in (11.5.3). As a rule of thumb, we can use (11.5.22) to determine the sample support required for the desired sidelobe level. For example, if we were to design an adaptive beamformer with -40 -dB sidelobe levels, we would require on the order of $K = 10,000$ snapshots.

EXAMPLE 11.5.2. We want to explore the effect of the number of training samples on the sidelobe levels of the SMI adaptive beamformer. To this end, we generate an interference signal at $\phi_i = 10^\circ$ with a power of 70 dB and noise with unit variance ($\sigma_w^2 = 1$) for a ULA with $M = 40$ elements. The interference-plus-noise signal \mathbf{x}_{i+n} is generated by

```
v_i = exp(-j*pi*[0:M-1]'*sin(phi_i*pi/180))/sqrt(M);
x_ipn=(10^(70/20))*v_i*(randn(1,N)+j*randn(1,N))/sqrt(2) + ...
(randn(1,N)+j*randn(1,N))/sqrt(2);
```

The sample correlation matrix is computed using (11.5.1). Then the SMI adaptive beamformer weights are computed from (11.5.2) with a look direction of $\phi_s = 0^\circ$. We can compute the beampattern of the SMI adaptive beamformer using (11.2.3). The resulting beampatterns averaged over 100 realizations for SMI adaptive beamformers with sample support of $K = 100$ and $K = 1000$ are shown in Figure 11.23 for $-10^\circ < \phi < 90^\circ$ along with the beampattern of an optimum beamformer computed using the weight vector in (11.3.15) and a true correlation

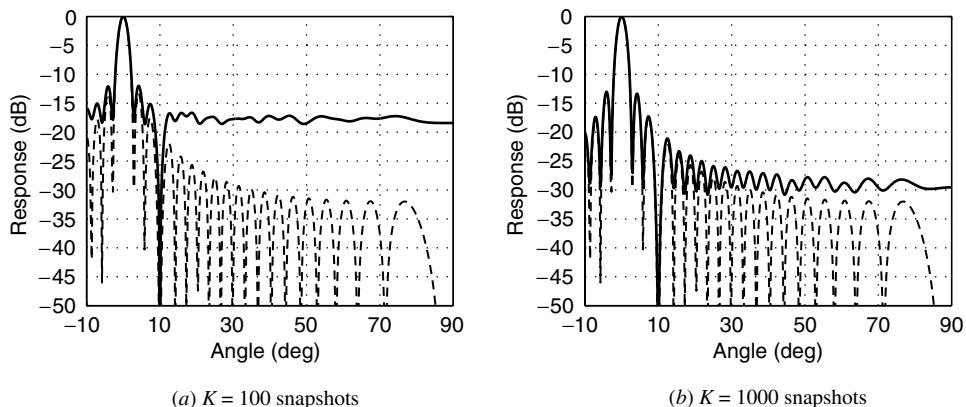


FIGURE 11.23

Beampatterns of an SMI adaptive beamformer for (a) $K = 100$ snapshots and (b) $K = 1000$ snapshots. The dashed line is the quiescent response (optimum beamformer), and the solid line is the SMI adaptive beamformer.

```
R_ipn = (10^(70/10))*v_i*v_i' + eye(M);
```

Clearly, the sidelobe levels of the SMI adaptive beamformer are limited by the sample support available for training. For the case of $K = 100$, the sidelobe level is approximately -18 dB, whereas for $K = 1000$, the sidelobe level is approximately -30 dB.

Training issues

To implement the SMI adaptive beamformer, we need an estimate of the interference-plus-noise correlation matrix, which of course requires that no desired signal $s(n)$ be present. The use of \mathbf{R}_{i+n} provided an attractive theoretical basis for the derivation of the optimum beamformer and its subsequent adaptive implementation with the SMI technique. Although it can be shown that the use of a correlation matrix containing the desired signal produces equivalent adaptive weights in the case of perfect steering, this can almost never be accomplished in practice. Usually, we do not have perfect knowledge of the exact array sensor locations and responses. Coupled with the fact that often the angle of the desired signal is not known exactly for cases when we are searching for its actual direction, the presence of the desired signal in the training set results in the cancellation and subsequent loss in performance.

How do we get a signal-free estimate of the correlation matrix from array data in practice? In many applications, such as in certain radar and communications systems, we control when the desired signal is present since it is produced by a transmission that we initiate. In the case of jamming, common to both these applications, we can choose not to transmit for a period of time in order to collect data with which we can estimate \mathbf{R}_{i+n} . This type of training is often termed *listen-only*. For other types of interference that are only present at the same time as the desired signal, such as clutter in radar and reverberations in active sonar, the training can be accomplished using a technique known as *split window*. If we use a training set consisting of data samples around the sample of interest (before and after), we can exclude the sample of interest, and possibly some of its neighboring samples, to avoid the inclusion of the desired signal in the training set. This method has significant computational implications because it requires a different correlation matrix and therefore a separate computation of the adaptive beamforming weights for each sample under consideration. This problem can be alleviated somewhat by using matrix update methods, as discussed in Chapter 10; nonetheless, the increase in cost cannot be considered insignificant.

Certain methods have been proposed for the purposes of reducing the computations associated with estimating the correlation matrix. One such method is to assume the correlation matrix is Toeplitz for a ULA. Of course, this assumption is valid if the array consists of elements with equal responses $H_k(F, \phi)$ as a function of both frequency and angle from (11.1.14). However, in practice, this assumption almost never holds. The fact that the spatial signals are measured using different sensors, all with different responses, coupled with the limits on mechanical precision of the sensor placement in the array inevitably will cause these assumptions to be violated. As a result, constraining the correlation matrix to be Toeplitz, which is akin to averaging the correlations down the diagonals of the correlation matrix, will cause performance degradation that can be significant. These methods are well suited for temporal signals that are measured with a common sensor and are sampled at a rate that is very accurately controlled via a single analog-to-digital converter. Unfortunately with arrays, the spatial sampling process is not nearly as precise, and the use of multiple sensors for measurements can produce vastly different signal characteristics.

11.5.2 Diagonal Loading with the SMI Beamformer

Clearly, the ability of an SMI adaptive beamformer to achieve a desired sidelobe level relies on the availability of sufficient sample support K . However, for many practical applications,

owing to either the nonstationarity of the interference or operational considerations, a limited number of samples are available to train the SMI adaptive beamformer. How, then, can we achieve this desired low sidelobe behavior? First, recall that the beam response of an optimum beamformer can be written in terms of its eigenvalues and eigenvectors as in (11.3.23). Likewise, for the SMI adaptive beamformer

$$C_{\text{smi}}(\phi) = \frac{\alpha}{\hat{\lambda}_{\min}} \left\{ C_q(\phi) - \sum_{m=1}^M \frac{\hat{\lambda}_m - \hat{\lambda}_{\min}}{\hat{\lambda}_m} [\hat{\mathbf{q}}_m^H \mathbf{v}(\phi_s)] \hat{\mathcal{Q}}_m(\phi) \right\} \quad (11.5.23)$$

where $\hat{\lambda}_m$ and $\hat{\mathbf{q}}_m$ are the eigenvalues and eigenvectors of $\hat{\mathbf{R}}_{i+n}$, respectively, and $C_q(\phi)$ and $\hat{\mathcal{Q}}_m(\phi)$ are the beampatterns of the quiescent weight vector and the m th eigenvector, known as an eigenbeam, respectively. Therefore, $C_{\text{smi}}(\phi)$ is simply $C_q(\phi)$ minus weighted eigenbeams that place nulls in the directions of interferers. The weights on the eigenbeams are determined by the ratio $(\hat{\lambda}_m - \hat{\lambda}_{\min})/\hat{\lambda}_m$. The noise eigenvectors are chosen to fill the remainder of the interference-plus-noise space that is not occupied by the interference. Ideally, these eigenvectors should have no effect on the beam response because the eigenvalues of the true correlation matrix $\lambda_m = \lambda_{\min} = \sigma_w^2$ for $m > P$. However, this relation does not hold for the sample correlation matrix for which the eigenvalues vary about the noise power σ_w^2 and asymptotically approach this expected value for increasing sample support. Therefore, the eigenbeams affect the beam response in a manner determined by their deviation from the noise floor σ_w^2 . Since, as in the case of the sample correlation matrix, these eigenvalues are random variables that vary according to the sample support K , the beam response suffers from the addition of randomly weighted eigenbeams. The result is a higher sidelobe level in the adaptive beampattern.

A means of reducing the variation of the eigenvalues is to add a weighted identity matrix to the sample correlation matrix (Hudson 1981, Carlson 1988)

$$\hat{\mathbf{R}}_l = \hat{\mathbf{R}}_{i+n} + \sigma_l^2 \mathbf{I} \quad (11.5.24)$$

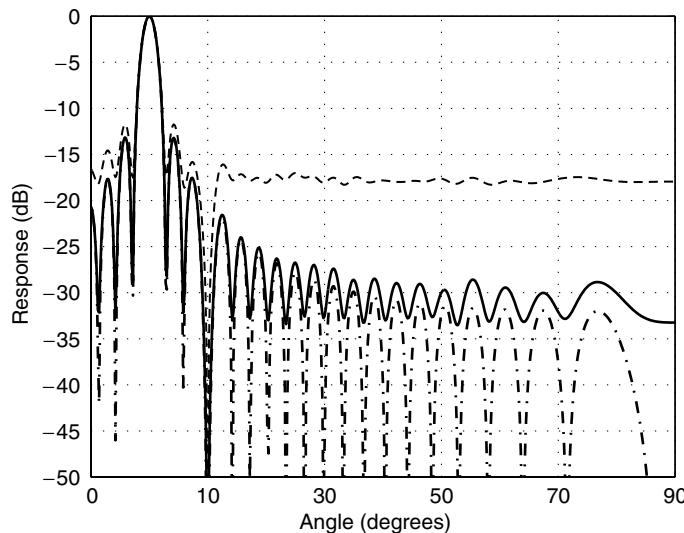
a technique that is known as *diagonal loading*. The result of diagonal loading of the correlation matrix is to add the loading level to all the eigenvalues. This, in turn, produces a bias in these eigenvalues in order to reduce their variation. To obtain the diagonally loaded SMI adaptive beamformer, simply substitute $\hat{\mathbf{R}}_l$ into (11.5.2)

$$\mathbf{c}_{l\text{smi}} = \frac{\hat{\mathbf{R}}_l^{-1} \mathbf{v}(\phi_s)}{\mathbf{v}^H(\phi_s) \hat{\mathbf{R}}_l^{-1} \mathbf{v}(\phi_s)} \quad (11.5.25)$$

The bias in the eigenvalues produces a slight bias in the adaptive weights that reduces the output SINR. However, this reduction is very modest when compared to the substantial gains in the quality of the adaptive beampattern.

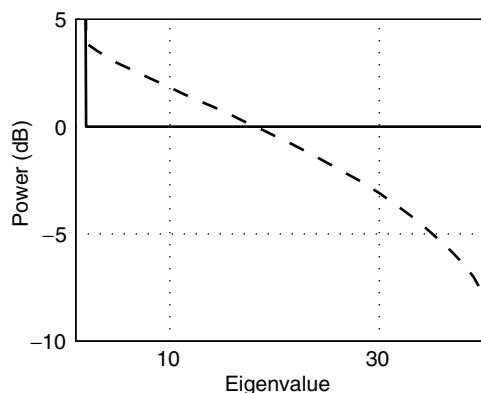
Recommended loading levels are $\sigma_w^2 \leq \sigma_l^2 < 10\sigma_w^2$. The maximum loading level is dependent on the application, but the minimum should be at least equal to the noise power in order to achieve substantial improvements. The loading causes a reduction in the nulling of weak interferers, that is, interferers with powers that are relatively close to the noise power. The effect on strong interferers is minimal since their eigenvalues only experience a minor increase. One added benefit of diagonal loading is that it provides a robustness to signal mismatch, as described in Section 11.4.1.

EXAMPLE 11.5.3. In this example, we explore the use of diagonal loading of the sample correlation matrix to control the sidelobe levels of the SMI adaptive beamformer using the same set of parameters as in Example 11.5.2. The beampatterns for the SMI adaptive beamformer and the diagonally loaded SMI adaptive beamformer are shown in Figure 11.24 along with the beampattern for the optimum beamformer for $-10^\circ < \phi < 90^\circ$. The diagonal loading level was set to 5 dB above the thermal noise power, that is, $\sigma_l^2 = 10^{0.5}$, and the sample support was $K = 100$. The sidelobe levels of the diagonally loaded SMI adaptive beamformer

**FIGURE 11.24**

Beampatterns of an SMI adaptive beamformer for $K = 100$ snapshots without diagonal loading (dashed line), and with $\sigma_1^2 = 5$ dB diagonal loading (solid line). The beampattern of the optimum beamformer is also shown with the dash-dot line.

are very close to those of the optimum beamformer that used a known correlation matrix, while for the SMI adaptive beamformer the sidelobes are at approximately -18 dB. To gain some insight into the higher sidelobe levels of the SMI adaptive beamformer, we compute the eigenvalues of the SMI adaptive beamformer without diagonal loading using the MATLAB command `lambda = eig(Rhat)`; where `Rhat` is the sample correlation matrix from (11.5.1). The eigenvalues of the sample and true correlation matrix are shown in Figure 11.25. The largest eigenvalue, corresponding to the 70-dB jammer, is approximately 70 dB but cannot be observed on this plot. We notice that for $K = 100$ training samples, the noise eigenvalues of $\hat{\mathbf{R}}_{i+n}$ are significantly different from those of \mathbf{R}_{i+n} , with larger than a 10-dB difference in some cases. As we stated earlier, the effect on the beampattern of the SMI adaptive beamformer is to add a random pattern weighted by this difference in eigenvalues. In the case of diagonal loading, the eigenvalues have as a lower bound the loading level σ_1^2 which, in turn, reduces these errors that are added to the beampatterns. The cost of the diagonal loading is to limit our ability to cancel weak interference with power less than the loading level. However, in the case of strong interference, almost no loss in terms of interference cancellation is experienced by introducing diagonal loading.

**FIGURE 11.25**

Noise eigenvalues of the SMI adaptive beamformer without diagonal loading $\sigma_w^2 = 1$ (dashed line) and the optimum beamformer (solid line).

11.5.3 Implementation of the SMI Beamformer

Although the SMI adaptive beamformer is formulated in terms of an estimated correlation matrix, the actual implementation, as with the least-squares methods discussed in Chapter 8, is usually in terms of the data samples directly. In other words, the actual estimate of the correlation matrix is never formed explicitly. Methods that are implemented on the data directly are commonly referred to as *amplitude domain* techniques, whereas if the sample correlation matrix had been formed, the implementation would be said to be performed in the *power domain*. The explicit computation of the sample correlation matrix is undesirable, first and foremost because the squaring of the data requires a large increase in the dynamic range of any processor. Numerical errors in the data are squared as well, and for a large number of training samples this computation may be prohibitively expensive. In this section, we give a brief discussion of the implementation considerations for the SMI adaptive beamformer, where the implementation is strictly in the amplitude domain. The incorporation of diagonal loading in this setting is also discussed since its formulation was given in the power domain.

The SMI beamformer is based on the estimated correlation matrix from (11.5.1). This sample correlation matrix may be written equivalently as

$$\hat{\mathbf{R}}_{i+n} = \frac{1}{K} \sum_{k=1}^K \mathbf{x}(n_k) \mathbf{x}^H(n_k) = \frac{1}{K} \mathbf{X}^H \mathbf{X} \quad (11.5.26)$$

where \mathbf{X} is the data matrix formed with the array snapshots that make up the training set for the SMI adaptive weights, presumably containing only interference and noise, that is, no desired signals. This data matrix is

$$\mathbf{X}^H = [\mathbf{x}(n_1) \ \mathbf{x}(n_2) \ \cdots \ \mathbf{x}(n_K)] \quad (11.5.27)$$

$$= \begin{bmatrix} x_1(n_1) & x_1(n_2) & \cdots & x_1(n_K) \\ x_2(n_1) & x_2(n_2) & \cdots & x_2(n_K) \\ \vdots & \vdots & \ddots & \vdots \\ x_M(n_1) & x_M(n_2) & \cdots & x_M(n_K) \end{bmatrix} \quad (11.5.28)$$

where n_k , for $k = 1, 2, \dots, K$, are the array snapshot indices of the training set. As was shown in Chapter 8, we can perform a QR decomposition on the data matrix to obtain the upper triangular factor

$$\mathbf{X} = \mathbf{Q} \mathcal{R}_x \quad (11.5.29)$$

where \mathbf{Q} is a $K \times M$ orthonormal matrix and \mathcal{R}_x is the $M \times M$ upper triangular factor. If we define the lower triangular factor as

$$\mathbf{L}_x \triangleq \frac{1}{\sqrt{K}} \mathcal{R}_x^H \quad (11.5.30)$$

the sample correlation matrix can then be written as

$$\hat{\mathbf{R}}_{i+n} = \frac{1}{K} \mathbf{X}^H \mathbf{X} = \frac{1}{K} \mathcal{R}_x^H \mathcal{R}_x = \mathbf{L}_x \mathbf{L}_x^H \quad (11.5.31)$$

since $\mathbf{Q}^H \mathbf{Q} = \mathbf{I}$. The SMI adaptive weights from (11.5.2) are then found to be

$$\mathbf{c}_{smi} = \frac{\hat{\mathbf{R}}_{i+n}^{-1} \mathbf{v}(\phi_s)}{\mathbf{v}^H(\phi_s) \hat{\mathbf{R}}_{i+n}^{-1} \mathbf{v}(\phi_s)} = \frac{\mathbf{L}_x^{-H} \mathbf{L}_x^{-1} \mathbf{v}(\phi_s)}{|\mathbf{L}_x^{-1} \mathbf{v}(\phi_s)|^2} \quad (11.5.32)$$

The implementation of diagonal loading with the SMI adaptive beamformer is also possible in the amplitude domain. Recall that the diagonally loaded correlation matrix from (11.5.24) is given by

$$\hat{\mathbf{R}}_l = \hat{\mathbf{R}}_{i+n} + \sigma_l^2 \mathbf{I} = \frac{1}{K} \mathbf{X}^H \mathbf{X} + \sigma_l^2 \mathbf{I} \triangleq \frac{1}{K} \mathbf{X}_l^H \mathbf{X}_l \quad (11.5.33)$$

where \mathbf{X}_l is the “diagonally loaded” data matrix. Of course, data matrix \mathbf{X} is not a square matrix, and thus it is not actually diagonally loaded. Instead, we append the data matrix with the square root of the loading matrix as

$$\mathbf{X}_l^H = [\mathbf{X}^H \sqrt{K}\sigma_l \mathbf{I}] \quad (11.5.34)$$

The resulting diagonally loaded SMI adaptive weights are found by substituting \mathbf{X}_l for \mathbf{X} in the amplitude-domain implementation of the SMI adaptive beamformer given above. The practical implementation of the SMI adaptive beamformer is performed in the following steps:

1. Compute the QR factorization of data matrix $\mathbf{X} = \mathbf{Q}\mathcal{R}_x$.
2. Find the Cholesky factor by normalizing the upper triangular factor $\mathbf{L}_x = (1/\sqrt{K})\mathcal{R}_x^H$.
3. Solve for \mathbf{z}_1 from $\mathbf{L}_x \mathbf{z}_1 = \mathbf{v}(\phi_s)$.
4. Solve for \mathbf{z}_2 from $\mathbf{L}_x^H \mathbf{z}_2 = \mathbf{z}_1$.
5. The SMI adaptive weight vector is given by $\mathbf{c}_{\text{smi}} = \mathbf{z}_2 / \|\mathbf{z}_1\|^2$.

11.5.4 Sample-by-Sample Adaptive Methods

The SMI adaptive beamformer is a least-squares (LS) block adaptive technique similar to the LS methods discussed in Chapter 8. However, the optimum beamformer can also be implemented by using methods that compute the beamforming weights on a sample-by-sample basis; that is, the weights are updated for each new sample. Such methods are referred to as *sample-by-sample adaptive* and are simply extensions of the adaptive filtering methods from Chapter 10. The manner in which sample adaptive beamformers differ from adaptive filters is that rather than solve an unconstrained LS problem, adaptive beamformers solve a constrained LS problem. The implication of this constraint is that rather than have an estimated cross-correlation in the normal equations $\hat{\mathbf{R}}(n)\mathbf{c} = \hat{\mathbf{d}}(n)$, we have the deterministic steering vector $\mathbf{v}(\phi_s)$. Unlike the cross-correlation vector, the steering vector is known a priori and is *not* estimated from the data. We briefly discuss both techniques based on recursive least-squares (RLS) and steepest-descent methods. Since the derivation of the methods follows that for the adaptive filters in Chapter 10 quite closely, we only give a brief sketch of the algorithms along with some discussion.

An important consideration for sample adaptive methods is whether or not these techniques are appropriate for array processing applications. The problem with these methods is the amount of time required for the adaptive weights to converge. In many applications, the delay associated with the convergence of the adaptive beamformer is not acceptable. For example, a radar system might be attempting to find targets at close ranges. Range corresponds to the time delay associated with the propagation of the radar signal. Therefore, close ranges are the first samples received by the array during a collection period. A sample adaptive method that uses the close ranges to train (converge) could not find the targets at these ranges (samples). In fact, the time needed for convergence may not be insignificant, thus creating a large blind interval that is often unacceptable. However, the sample-by-sample adaptive techniques are appropriate for array processing applications in which the operating environment is nonstationary. Since the sample-by-sample adaptive beamformer alters its weights with each new sample, it can dynamically update its response for such a changing scenario.

Another important distinction between sample and block adaptive methods is the inclusion of the signal of interest in each sample and thus in the correlation matrix. Therefore, for sample adaptive methods, we cannot use a signal-free version of the correlation matrix, that is, the interference-plus-noise correlation matrix \mathbf{R}_{i+n} , but rather must use the whole correlation matrix \mathbf{R}_x . The inclusion of the signal in the correlation matrix has profound effects on the robustness of the adaptive beamformer in the case of signal mismatch. This effect was discussed in the context of the optimum beamformer in Section 11.4.1.

Recursive least-squares methods

We will not spend a lot of time discussing recursive least-squares (RLS) methods for adaptive beamforming since this topic is treated in Chapter 10. For further details on RLS methods used in array processing, the interested reader is referred to Schreiber (1986), McWhirter and Shepherd (1989), Yang and Böhme (1992). An important difference between the methods discussed here and those in Section 10.6 is in the normal equations that solve a constrained rather than an unconstrained optimization. The output signal of the adaptive beamformer is

$$y(n) = \mathbf{c}^H \mathbf{x}(n) \quad (11.5.35)$$

However, $y(n)$ is not the desired response. In Section 10.6, we developed techniques based on the normal equations $\hat{\mathbf{R}}\mathbf{c} = \hat{\mathbf{d}}$, where $\hat{\mathbf{d}}$ is the estimated cross-correlation. However, for the adaptive beamformer we use the steering vector $\mathbf{v}(\phi_s)$, which is deterministic, in place of $\hat{\mathbf{d}}$. Algorithms based on RLS methods can be implemented such that the output $y(n)$ is computed directly (direct output extraction) or the adaptive beamformer weights are computed and then applied to determine the output (see Section 10.6). The simplifications for the beamformer case are discussed in Yang and Böhme (1992) and Haykin (1996).

The RLS methods are based on the update equation of the estimate of the correlation matrix

$$\hat{\mathbf{R}}_x(n+1) = \lambda \hat{\mathbf{R}}_x(n) + \mathbf{x}(n+1)\mathbf{x}^H(n+1) \quad (11.5.36)$$

where $0 < \lambda \leq 1$ is a scalar sometimes referred to as the forgetting factor. From the updated sample correlation matrix, an update for its inverse can be found by using the matrix inversion lemma from Appendix A. The adaptive beamformer weight vector is then found by modifying the solution to the MVDR adaptive weights with the updated inverse sample correlation matrix. In practice, these updatings are implemented by slightly modifying any of the algorithms described in Section 10.6.

Steepest-descent methods

The LMS algorithm from Section 10.4 is based on the method of steepest descent. However, the desired response used to form the LMS adaptive weights is not clear for the adaptive beamforming application. Instead, there is the steering vector $\mathbf{v}(\phi_s)$ that specifies the direction to which the adaptive beamformer is steered, namely, the angle ϕ_s . The resulting constrained optimization produced the optimum MVDR beamformer from Section 11.3. The sample adaptive implementation of this constrained optimization problem based on steepest descent was first proposed by Frost. The resulting algorithm uses a projection operation to separate the constrained optimization into a data-independent component and an adaptive portion that performs an unconstrained optimization (Frost 1972). The original algorithm was formulated using multiple linear constraints, as will be discussed in Section 11.6.1. However, in this section we focus on its implementation with the single unity-gain look-direction constraint for the MVDR beamformer. Note that the separation of the constrained and unconstrained components proposed by Frost provided the motivation for the generalized sidelobe canceler (GSC) structure (Griffiths and Jim 1982) discussed in Section 11.3.5. Below, we simply give the procedure for implementing the Frost algorithm. The interested reader is referred to Frost (1972) for further details.

Formally, the MVDR adaptive beamformer is attempting to solve the following constrained optimization

$$\min \mathbf{c}^H \mathbf{R}_x \mathbf{c} \quad \text{subject to} \quad \mathbf{c}^H \mathbf{v}(\phi_s) = 1 \quad (11.5.37)$$

where the entire correlation matrix \mathbf{R}_x including the desired signal is used in place of the interference-plus-noise correlation matrix \mathbf{R}_{i+n} since we assume the signal of interest is always present. The correlation matrix is unknown and must be estimated from the data. To start the algorithm, we can form an $M \times M$ projection matrix \mathbf{P} that projects onto a

subspace orthogonal to the data-independent steering vector $\mathbf{v}(\phi_s)$. This projection matrix is given by (see Chapter 8)

$$\mathbf{P} = \mathbf{I} - \mathbf{v}(\phi_s)\mathbf{v}^H(\phi_s) \quad (11.5.38)$$

We can then define the nonadaptive beamformer weight vector as

$$\mathbf{c}_{\text{na}} = \mathbf{v}(\phi_s) \quad (11.5.39)$$

which is simply the spatial matched filter from Section 11.2. The update equation for the sample adaptive beamformer based on Frost's steepest-descent (sd) algorithm is then written as

$$\mathbf{c}_{\text{sd}}(n+1) = \mathbf{c}_{\text{na}} + \mathbf{P}[\mathbf{c}_{\text{sd}}(n) - \mu y^*(n)\mathbf{x}(n)] \quad (11.5.40)$$

where μ is the step-size parameter and

$$y(n) = \mathbf{c}_{\text{sd}}^H \mathbf{x}(n) \quad (11.5.41)$$

is the output of the steepest-descent sample adaptive beamformer.

Since the projection matrix \mathbf{P} maintains orthogonality between the \mathbf{c}_{na} and the adapted portion of (11.5.40), the nonadaptive beamformer weights from (11.5.39) maintain the unity-gain constraint from (11.5.37). In fact, since the adaptation is performed on the component orthogonal to \mathbf{c}_{na} in an unconstrained manner, the Frost algorithm is essentially using an LMS adaptive filter in the GSC architecture from Section 11.3.5. The convergence of the Frost adaptive beamformer, as for the LMS adaptive filter, is controlled by the step-size parameter μ . In order for the adaptive beamformer weights to converge, the step-size parameter must be chosen to be

$$0 < \mu < \frac{1}{\tilde{\lambda}_{\max}} \quad (11.5.42)$$

where $\tilde{\lambda}_{\max}$ is the maximum eigenvalue of the matrix

$$\tilde{\mathbf{R}} = \mathbf{P}\mathbf{R}_x\mathbf{P} \quad (11.5.43)$$

More details about the algorithm can be found in Frost (1972).

The sample adaptive beamformer based on the Frost algorithm maintains a look direction of ϕ_s through the constraint $\mathbf{c}_{\text{sd}}^H \mathbf{v}(\phi_s) = 1$. This constraint is easily seen by interpreting the adaptive weight update equation in (11.5.40) as the steering vector $\mathbf{c}_{\text{na}} = \mathbf{v}(\phi_s)$ updated by a component orthogonal to $\mathbf{v}(\phi_s)$. In the case of a signal received from a direction ϕ_s , the adaptive beamformer will immediately track this signal, since it is constrained to observe signals at ϕ_s and is not part of the adaptation. The convergence of this sample adaptive beamformer in terms of interference rejection is very similar to the LMS algorithm. See Chapter 10 for details on the LMS algorithm.

11.6 OTHER ADAPTIVE ARRAY PROCESSING METHODS

In this section, we consider various other adaptive array processing methods. First, we look at the use of multiple constraints in an adaptive array beyond the single constraint of distortionless response for the MVDR optimum beamformer. Second, we consider partially adaptive arrays that are methods that perform deterministic preprocessing prior to adaptation, in order to reduce the adaptive degrees of freedom. These methods are commonly used in practice for both computational reasons as well as limited sample support. Third, we describe the sidelobe canceler that was the first proposed adaptive array processing method. In addition to its historical significance, the sidelobe canceler is still a viable technique for certain array processing applications. Throughout this section, we use the word *adaptive* to indicate that the various methods are based on training data. However, the derivations are all

in terms of known statistics. Although none of these methods can really be called optimum, each one satisfies an optimization criterion in the case of known statistics. The implementation of the methods using actual data samples in place of assuming known statistics follows directly from the techniques described in Section 11.5.

11.6.1 Linearly Constrained Minimum-Variance Beamformers

In Section 11.3, we discussed the optimum beamformer that maximizes the signal-to-interference-plus-noise ratio (SINR). This optimum beamformer was also formulated as the solution to a constrained optimization problem, namely,

$$\min \mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c} \quad \text{subject to} \quad \mathbf{c}^H \mathbf{v}(\phi_s) = 1 \quad (11.6.1)$$

where $\mathbf{v}(\phi_s)$ is the array response vector for a signal arriving from an angle ϕ_s . Due to this alternate formulation, the optimum beamformer is commonly referred to as the minimum-variance distortionless response (MVDR) beamformer.

However, some applications may require additional conditions on the beamformer. As with the optimum beamformer, we want to minimize the output power $\mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c}$, but with additional constraints on the response of the beamformer. The imposition of further constraints on the minimum-variance beamformer results in suboptimum performance in terms of SINR. However, if designed properly, the constraints should have little effect on SINR while yielding some desirable attributes. One common use of constraints is for the case when the angle of an interference source ϕ_i is known a priori. In this case, we want to reject all energy received from this angle, that is,

$$\mathbf{c}^H \mathbf{v}(\phi_i) = 0 \quad (11.6.2)$$

The result of the null constraint is an adaptive beamformer that rejects all energy from the angle ϕ_i . Another type of constraint is to require the beamformer to pass signals not only from the angle ϕ_s , but also from another angle ϕ_1 . As for the MVDR beamformer, this constraint is formulated as

$$\mathbf{c}^H \mathbf{v}(\phi_1) = 1 \quad (11.6.3)$$

In this manner, multiple angles can be specified to pass signals of interest with unity gain. Such amplitude constraints can also be used to preserve the response of the beamformer in an angular region about ϕ_s (Steele 1983, Takao et al. 1976). These additional constraints help to make the resulting adaptive beamformer more robust to signal mismatches, as discussed in Section 11.4.1, that result from the actual angle of the desired signal ϕ_0 slightly differing from its presumed angle ϕ_s . Therefore, if we choose a pair of angles slightly offset from ϕ_s

$$\phi_1 = \phi_s - \Delta\phi \quad \phi_2 = \phi_s + \Delta\phi \quad (11.6.4)$$

the response of the beamformer steered to ϕ_s broadens. The effect in terms of mainlobe width is similar to tapering the MVDR beamformer when the angle offset $\Delta\phi$ is small. An alternative approach to robust adaptive beamforming is the use of derivative constraints. See Applebaum and Chapman (1976), Er and Cantoni (1983), and Steele (1983) for details.

Once we have determined a set of constraints, for example, the desired responses at a set of angles, we can solve for the constrained adaptive beamformer. The result is known as the *linearly constrained minimum-variance (LCMV) beamformer* (Applebaum and Chapman 1976; Buckley 1987). As we stated earlier, we want to minimize the output energy of the beamformer subject to a set of constraints. This problem is formulated as

$$\min \mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c} \quad \text{subject to} \quad \mathbf{C}^H \mathbf{c} = \boldsymbol{\delta} \quad (11.6.5)$$

where \mathbf{C} is known as the constraint matrix and $\boldsymbol{\delta}$ is the constraint response vector. For example, if we want to pass signals from an angle ϕ_s as well as preserve its response with

a pair of amplitude constraints at the angles $\phi_s \pm \Delta\phi$, the constraint matrix and constraint response vectors are given by

$$\mathbf{C} = [\mathbf{v}(\phi_s) \ \mathbf{v}(\phi_s - \Delta\phi) \ \mathbf{v}(\phi_s + \Delta\phi)] \quad \boldsymbol{\delta} = [1 \ 1 \ 1]^T \quad (11.6.6)$$

As for the MVDR beamformer, the solution for the LCMV beamformer is found by using Lagrange multipliers (see Appendix B). The LCMV beamformer weight vector is given by

$$\mathbf{c}_{\text{lcmv}} = \mathbf{R}_{i+n}^{-1} \mathbf{C} (\mathbf{C}^H \mathbf{R}_{i+n}^{-1} \mathbf{C})^{-1} \boldsymbol{\delta} \quad (11.6.7)$$

As for the MVDR beamformer, the LCMV beamformer can also be formulated using a generalized sidelobe canceler architecture (Griffiths and Jim 1982), discussed in Section 11.3.5. In fact, the MVDR beamformer is simply a special case of the LCMV beamformer with $\mathbf{C} = \mathbf{v}(\phi_s)$ and $\boldsymbol{\delta} = 1$.

In this section, we have described the use of *linear* constraints in a minimum-variance beamformer. However, the use of *quadratic constraints* within the context of a minimum-variance beamformer is also possible. The primary motivation for using these quadratic constraints is for robustness purposes against signal mismatch, as discussed in Section 11.4.1. One such quadratic constraint adds a constraint on the norm of the weight vector of the adaptive beamformer in addition to the MVDR constraint (Cox et al. 1987; Maksym 1979)

$$\min \mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c} \quad \text{subject to} \quad \mathbf{c}^H \mathbf{v}(\phi_s) = 1 \quad \text{and} \quad \|\mathbf{c}\|^2 \leq \kappa^2 \quad (11.6.8)$$

whose solution is given by

$$\mathbf{c} = \eta (\mathbf{R}_{i+n} + \sigma_\kappa^2 \mathbf{I})^{-1} \mathbf{v}(\phi_s) \quad (11.6.9)$$

where η is a constant and σ_κ^2 is a scaling term on the identity matrix. Thus, minimizing the norm of the adaptive beamforming weight vector is equivalent to adding a weighted identity matrix to the interference-plus-noise correlation matrix. The solution to this quadratic constraint bears a striking resemblance to diagonal loading as discussed in the context of the SMI adaptive beamformer, in Section 11.5.2. In fact, the use of some level of diagonal loading is generally a recommended practice for implementing an adaptive beamformer to reduce its sensitivity to mismatch, and for the purposes of low sidelobe levels.

11.6.2 Partially Adaptive Arrays

The optimum beamformer maximizes output SINR by placing a null in the direction of any interference sources while maintaining gain in the direction of interest. Recall the optimum beamforming weights from (11.3.13)

$$\mathbf{c}_o = \alpha \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s) \quad (11.6.10)$$

where we choose the MVDR normalization $\alpha = [\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)]^{-1}$. The correlation matrix \mathbf{R}_{i+n} is an $M \times M$ matrix where M is the number of elements in the ULA. The optimum weights adapt to the statistics of the data in an M -dimensional space where M is referred to as the *adaptive degrees of freedom*. However, in many applications, the number of elements in the array exceeds the adaptive degrees of freedom that can be practically implemented. The implementation of such a beamformer requires the estimation of the correlation matrix from collected data. As shown in Section 11.5, the estimation of \mathbf{R}_{i+n} requires a certain number of data samples to maintain a desired level of performance. Many times, the number of data samples is limited, due to either finite regions over which the data are stationary or restrictions on the length of the collection interval. Likewise, the number of adaptive degrees of freedom that can be implemented may be limited for computational reasons. These restrictions motivate the use of methods that reduce the degrees of freedom

prior to adaptation. An array implemented using a reduced number of degrees of freedom is referred to as a *partially adaptive array*.

Consider an array signal vector $\mathbf{x}(n)$ consisting of a desired signal, interference, and noise components

$$\mathbf{x}(n) = \mathbf{s}(n) + \mathbf{i}(n) + \mathbf{w}(n) = \sqrt{M}\mathbf{v}(\phi_s)s(n) + \sqrt{M} \sum_{p=1}^P \mathbf{v}(\phi_p)i_p(n) + \mathbf{w}(n) \quad (11.6.11)$$

where ϕ_p and $i_p(n)$ are the angle and signal, respectively, of the p th interferer with a total of P interferers. Usually, the number of interferers is limited; yet the number of elements in the array M may be quite large, that is, $M \gg P$. In general, one adaptive degree of freedom is required for each interferer.[†] Therefore, we only require some number of adaptive degrees of freedom $Q > P$, not the full dimensionality provided by the number of elements M . We want to use a large number of elements in order to have an aperture that achieves the desired angular resolution. Therefore, we do not want to limit the number of elements in order to reduce the degrees of freedom; rather, we want to project the array data into a lower-dimensional subspace in which we can perform our optimization (Morgan 1978). The projection is accomplished using a nonadaptive preprocessor and is modeled as a rank-reducing transformation matrix \mathbf{T} with dimensions $M \times Q$ applied to the array signal

$$\tilde{\mathbf{x}}(n) = \mathbf{T}^H \mathbf{x}(n) \quad (11.6.12)$$

where $\tilde{\mathbf{x}}(n)$ is a signal vector of dimension Q . Likewise, the interference-plus-noise signal in the lower-dimensional space is

$$\tilde{\mathbf{x}}_{i+n}(n) = \mathbf{T}^H \mathbf{x}_{i+n}(n) \quad (11.6.13)$$

and has a correlation matrix

$$\tilde{\mathbf{R}}_{i+n} = E\{\tilde{\mathbf{x}}_{i+n}(n)\tilde{\mathbf{x}}_{i+n}^H(n)\} = \mathbf{T}^H \mathbf{R}_{i+n} \mathbf{T} \quad (11.6.14)$$

The partially adaptive beamforming weights are then given by

$$\tilde{\mathbf{c}} = \alpha \tilde{\mathbf{R}}_{i+n}^{-1} \tilde{\mathbf{v}}(\phi_s) \quad (11.6.15)$$

where

$$\tilde{\mathbf{v}}(\phi_s) = \mathbf{T}^H \mathbf{v}(\phi_s) \quad (11.6.16)$$

is the projection of the M -dimensional steering vector $\mathbf{v}(\phi_s)$ onto the same Q -dimensional subspace. The output of the partially adaptive beamformer is then obtained by applying the beamforming weights in (11.6.15) to the reduced-dimension array signal from (11.6.12)

$$y(n) = \tilde{\mathbf{c}}^H \tilde{\mathbf{x}}(n) \quad (11.6.17)$$

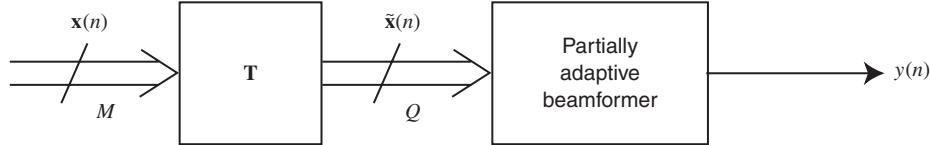
The resulting partially adaptive beamformer, shown in Figure 11.26, is no longer optimal in the sense of the full M -dimensional beamformer, but is optimal given the nonadaptive preprocessing transformation onto the Q -dimensional subspace. Thus, the overall performance of the partially adaptive beamformer is governed by how much information was preserved by the nonadaptive preprocessor \mathbf{T} . The performance of the partially adaptive beamformer can be assessed relative to the full-dimensional processor by reconstructing the effective $M \times 1$ beamforming weight vector with the transformation matrix and the partially adaptive (pa) weights

$$\mathbf{c}_{\text{pa}} = \mathbf{T} \tilde{\mathbf{c}} \quad (11.6.18)$$

In addition, we must consider the effect of this preprocessing transformation on the noise correlation matrix. For the array signal, we have assumed that the noise has a power of σ_w^2 and is uncorrelated, that is, $\mathbf{R}_n = \sigma_w^2 \mathbf{I}$. Therefore, the noise following the application of the preprocessing transformation has a correlation matrix given by

$$\tilde{\mathbf{R}}_n = \mathbf{T}^H \mathbf{R}_n \mathbf{T} = \sigma_w^2 \mathbf{T}^H \mathbf{T} \quad (11.6.19)$$

[†]The assumption is that the interferers are narrowband and are well separated in angle.

**FIGURE 11.26**

Partially adaptive array using data transformation.

In the case of an SMI adaptive beamformer, this different structure of the noise correlation matrix has implications for diagonal loading. The diagonal loading of the sample correlation matrix of the full array was performed by adding a weighted diagonal matrix to the sample correlation matrix in (11.5.24). For a partially adaptive array that already has had a preprocessing transform performed, the diagonal loading of a sample correlation matrix becomes

$$\hat{\mathbf{R}}_l = \hat{\mathbf{R}}_{i+n} + \sigma_l^2 \mathbf{T}^H \mathbf{T} \quad (11.6.20)$$

where σ_l^2 is the loading level. Since the thermal noise is not necessarily uncorrelated after the preprocessing transformation, diagonal loading must account for the transformed noise correlation. Otherwise, performance degradation can occur.

So far, we have only stated that the adaptation for a partially adaptive array must take place in a lower-dimensional space using a nonadaptive preprocessor, but we have not given any explicit means of performing this task. Below we discuss two commonly used preprocessing methods used for partially adaptive arrays.

Subarray partially adaptive arrays

Many times, the number of elements in an array can be very large. Thus, one means of reducing the adaptive degrees of freedom is to split the array into a number of smaller arrays, process the smaller arrays in a nonadaptive manner, and perform adaptation on the outputs of these smaller arrays. Let us consider the case in which we are looking for signals from a direction ϕ_s , and the full-dimensional steering vector is $\mathbf{v}_M(\phi_s)$, where we use the subscript M to denote the length of the steering vector. The full array may be divided into Q equal-sized intervals[†] of nonoverlapping subarrays of length

$$\tilde{M} = \frac{M}{Q} \quad (11.6.21)$$

where we have assumed that M is an integer multiple of Q . The rank-reducing transformation for the subarrays then can be written as a sparsely populated matrix made up of length- \tilde{M} steering vectors $\mathbf{v}_{\tilde{M}}(\phi_s)$

$$\mathbf{T} = \begin{bmatrix} \mathbf{v}_{\tilde{M}}(\phi_s) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{v}_{\tilde{M}}(\phi_s) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{v}_{\tilde{M}}(\phi_s) \end{bmatrix} \quad (11.6.22)$$

Each subarray consists of an \tilde{M} -dimensional conventional beamformer steered to ϕ_s and can be viewed as a highly directional element as opposed to the omnidirectional elements assumed for the individual sensors of the array.

Beamspace partially adaptive arrays

Another approach to constructing a partially adaptive beamformer is to produce a set of beams using the full array. The ensuing adaptation is performed in a reduced-dimension

[†] Subarrays need not necessarily have equal length or be nonoverlapping. This restriction is placed on the formulation only to simplify the discussion.

beamspace, that is, a space spanned by the nonadaptive beams. If we use B beams, the rank-reducing transformation matrix is

$$\mathbf{T} = [\mathbf{v}(\phi_1) \mathbf{v}(\phi_2) \cdots \mathbf{v}(\phi_B)] \quad (11.6.23)$$

where $\phi_1, \phi_2, \dots, \phi_B$ are the angles of these beams. These beamformers are typically steered in directions around the angle of interest, ϕ_s . For example, if the angle of interest is $\phi_s = 0^\circ$, beams might be steered to angles $\phi = -5^\circ, -4^\circ, \dots, 0^\circ, \dots, 4^\circ, 5^\circ$. The spacing of the beams depends on the full aperture of the array and the angular extent of interest. One can also steer beams in other directions away from the angle of interest, which may contain interference sources that we will want to cancel in the partially adaptive processor.

We have modeled the rank-reducing transformation as a matrix. Usually, the rank of the reduced-dimension space is dictated by the number of digital channels that can be formed due to hardware limitations. Therefore, the rank reduction process is performed prior to sampling using analog beamformers, either across a reduced or full array aperture for the subarray or beamspace partially adaptive array processors, respectively.

11.6.3 Sidelobe Cancelers

The *sidelobe canceler* is actually one of the first implementations of an adaptive array (Howells 1959), and it was originally proposed by Howells and Applebaum. The method uses a main channel along with a single auxiliary, or an array of auxiliary channels, as shown in Figure 11.27. The main channel generally has a high gain in the direction of the desired signal and is produced by either a highly directional sensor, for example, a parabolic dish, or the output of a nonadaptive beamformer, such as a spatial matched filter. The auxiliary channels, however, are low-gain elements often with omnidirectional responses that are used to augment the main channel. The auxiliary channels can be in a ULA configuration. The idea behind the sidelobe canceler is that interference is assumed to be present in both main and auxiliary channels, but the desired signal, though present in the main channel due to its high gain in the direction of the signal, is below the sensor thermal noise in the auxiliary channels. The auxiliary channels are used to form an estimate of the main channel interference that can be used for cancellation purposes. The philosophy behind the sidelobe canceler is shown in Figure 11.28, using representative beampatterns weighted by their directional gains.

Consider a main channel (mc) signal

$$x_{\text{mc}}(n) = g_s s(n) + i_{\text{mc}}(n) + w_{\text{mc}}(n) \quad (11.6.24)$$

consisting of the desired signal $s(n)$ with a gain of g_s , an interference signal $i_{\text{mc}}(n)$ that may be due to several interferers arriving from various angles, and noise $w_{\text{mc}}(n)$ that is

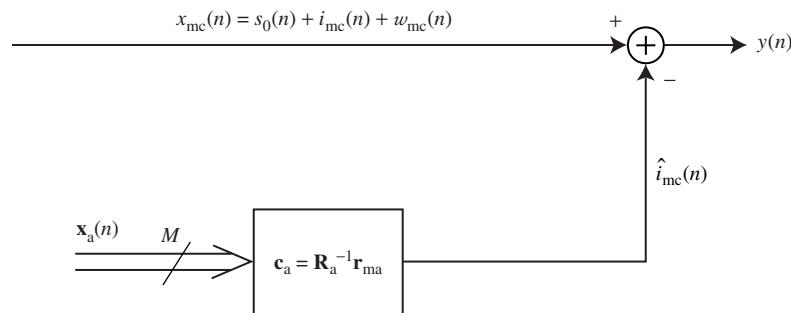


FIGURE 11.27
Sidelobe canceler.

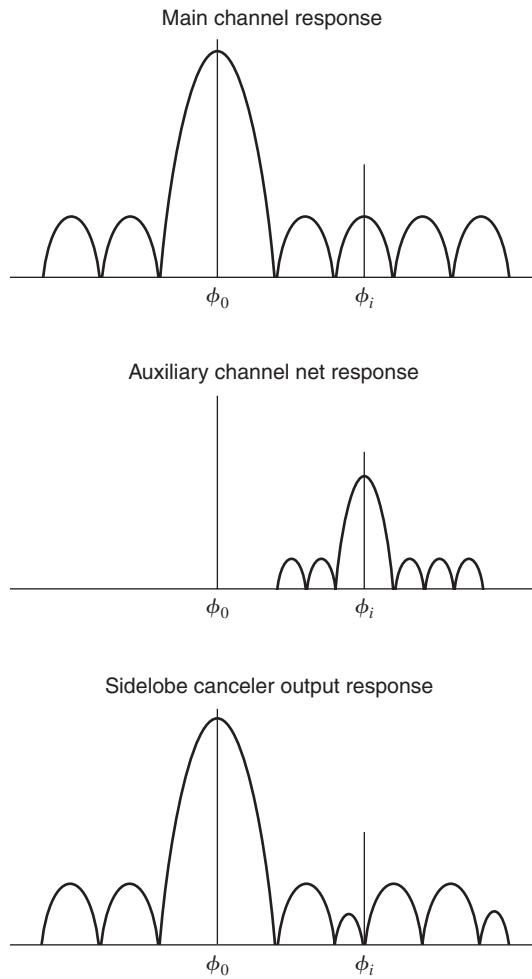


FIGURE 11.28
 Illustration of the sidelobe canceler channel and auxiliary channel beampatterns.

temporally uncorrelated. All three of these signals are assumed to be mutually uncorrelated. The interference in this main channel is often so strong that it dominates the desired signal even though it has a large gain in the direction of this desired signal. However, the auxiliary channel signals may be written as a signal vector

$$\mathbf{x}_a(n) = s(n)\mathbf{v}(\phi_s) + \sum_{p=1}^P i_p(n)\mathbf{v}(\phi_p) + \mathbf{w}(n) \quad (11.6.25)$$

where $\mathbf{v}(\phi)$ is the array response vector at an angle ϕ that was given by (11.1.19) for the case of a ULA. The desired signal impinges on the auxiliary array from the angle ϕ_s , and the sensor thermal noise $\mathbf{w}(n)$ is temporally and spatially uncorrelated. Recall that $s(n)$ is usually considered weak enough that it is well below the sensor noise power. The interference can be made up of several sources. Here we have chosen a model consisting of P interferers with signals $i_p(n)$ and angles of arrival of ϕ_p . Note that the main channel interference $i_{mc}(n)$ is made up of contributions from the same P interferers weighted by the spatial response of the main channel in the directions of the interference sources. These angles of arrival of the interferers, as well as the exact response of the main channel in these directions, are generally unknown and lead to an adaptive solution for the auxiliary channel weight vector.

The sidelobe canceler estimates the interference in the main channel by using the auxiliary channels. As illustrated in Figure 11.27, the auxiliary channels are combined by using a set of adaptive weights to form an estimate of the interference in the main channel.

$$\hat{i}_{mc}(n) = \mathbf{c}_a^H \mathbf{x}_a(n) \quad (11.6.26)$$

where the adaptive weight vector \mathbf{c}_a is chosen so as to minimize the output power. Of course, the implicit assumption has been made that the signal of interest is below the thermal noise level in $\mathbf{x}_a(n)$. Otherwise, if $s(n)$ is strong enough in the auxiliary channels, then the sidelobe canceler will cancel this signal of interest in addition to the interference. The output signal is then obtained by subtracting the estimate of the interference from the main channel

$$y(n) = x_{mc}(n) - \hat{i}_{mc}(n) \quad (11.6.27)$$

The output power is given by

$$P_{out} = \sigma_m^2 - E\{|\mathbf{c}_a^H \mathbf{x}_a(n)|^2\} = \sigma_m^2 - \mathbf{c}_a^H \mathbf{R}_a \mathbf{c}_a \quad (11.6.28)$$

where $\mathbf{R}_a = E\{\mathbf{x}_a(n)\mathbf{x}_a^H(n)\} \quad (11.6.29)$

is the auxiliary array correlation matrix. The solution for these weights is simply the linear MMSE estimator from Section 6.2, given by

$$\mathbf{c}_a = \mathbf{R}_a^{-1} \mathbf{r}_{ma} \quad (11.6.30)$$

where $\mathbf{r}_{ma} = E\{\mathbf{x}_a(n)x_{mc}^*(n)\} \quad (11.6.31)$

is the cross-correlation vector between the auxiliary array and the main channel. The output signal of the sidelobe canceler is

$$y(n) = x_{mc}(n) - \mathbf{c}_a^H \mathbf{x}_a(n) \quad (11.6.32)$$

Hence, the minimum output power is obtained by substituting (11.6.30) into (11.6.28)

$$P_{out}^{(min)} = \sigma_m^2 - \mathbf{r}_{ma}^H \mathbf{R}_a^{-1} \mathbf{r}_{ma} \quad (11.6.33)$$

Of course, all this analysis has considered the case in which the signal of interest is below the thermal noise level in the auxiliary array. Larger signal amplitudes will result in the cancellation of the signal of interest using a sidelobe canceler structure. This topic is treated in Problem 11.15.

11.7 ANGLE ESTIMATION

In this section, we consider the topic of angle estimation, that is, given a spatially propagating signal $\mathbf{s}(n)$, the determination of its angle of arrival at the array. In the formulation of the beamformers in Sections 11.2 through 11.6, the assumption was always made that the beamformer was steered to the angle of the desired signal. However, in practice, the actual angle from which the signal arrives is not precisely known. Instead, an amount of uncertainty exists with respect to the exact angle, even when the signal of interest is within the beam. The beamformer is steered to angle ϕ_0 while the actual signal arrives from ϕ_s . The purpose of an angle estimation algorithm is to attempt to determine this angle ϕ_s . We begin with a discussion of the maximum-likelihood (ML) angle estimator. Next we give a brief sketch of the Cramér-Rao lower bound on angle accuracy, which provides a measure against which the performance of any algorithm can be compared. Then we consider angle estimation algorithms, commonly referred to as *beamsplitting*. In the case of a ULA, a spatially propagating signal is equivalent to a complex exponential in the temporal domain. Hence, we briefly discuss the use of the frequency estimation techniques from Section 9.6 that were based on the model of a complex exponential contained in noise.

In this section, we give a brief discussion of the maximum-likelihood estimator of the angle of a signal arriving at a ULA. Consider a spatially propagating signal of interest

$$\mathbf{s} = \sqrt{M}\sigma_s \mathbf{v}(\phi_s) \quad (11.7.1)$$

where M is the number of sensors in the ULA, σ_s is the complex amplitude of the signal, and ϕ_s is the angle of the signal. The complex signal amplitude has a deterministic magnitude and uniformly distributed random phase. The signal is received by the ULA along with interference \mathbf{i} and spatially uncorrelated thermal noise \mathbf{w} , that is,

$$\mathbf{x} = \sqrt{M}\sigma_s \mathbf{v}(\phi_s) + \mathbf{i} + \mathbf{w} = \sqrt{M}\sigma_s \mathbf{v}(\phi_s) + \mathbf{x}_{i+n} \quad (11.7.2)$$

We have dropped the discrete-time index n since we are assuming the signal is present[†] and we are interested in a single snapshot only. The interference-plus-noise correlation matrix of the snapshot \mathbf{x} is given by

$$\mathbf{R}_{i+n} = E\{\mathbf{x}_{i+n} \mathbf{x}_{i+n}^H\} = \mathbf{R}_i + \sigma_w^2 \mathbf{I} \quad (11.7.3)$$

Furthermore, we assume that the interference-plus-noise signal \mathbf{x}_{i+n} has a complex Gaussian density function with zero mean. Thus, the probability density function of the snapshot \mathbf{x} is a complex Gaussian function with a mean determined by the signal of interest

$$p(\mathbf{x}; \sigma_s, \phi_s) = \frac{1}{\pi^M \det(\mathbf{R}_{i+n})} \exp\{-[\mathbf{x} - \sqrt{M}\sigma_s \mathbf{v}(\phi_s)]^H \mathbf{R}_{i+n}^{-1} [\mathbf{x} - \sqrt{M}\sigma_s \mathbf{v}(\phi_s)]\} \quad (11.7.4)$$

The peak in this probability density function corresponds to the mean given by the signal of interest $\sqrt{M}\sigma_s \mathbf{v}(\phi_s)$, which is the “most likely” event. The ML angle estimate is the angle $\hat{\phi}_s$ for which this probability density function of the snapshot takes on its maximum value, that is,

$$\hat{\phi}_s = \arg \max_{\phi} p(\mathbf{x}; \sigma_s, \phi_s) \quad (11.7.5)$$

The resulting ML estimator of ϕ_s is then given by (Kay 1993)

$$\hat{\phi}_s = \arg \max_{\phi} \frac{|\mathbf{v}^H(\phi) \mathbf{R}_{i+n}^{-1} \mathbf{x}|^2}{\mathbf{v}^H(\phi) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi)} \quad (11.7.6)$$

Interestingly, this ML estimate can be interpreted as

$$\hat{\phi}_s = \arg \max_{\phi} |\mathbf{c}_{amf}^H(\phi) \mathbf{x}|^2 \quad (11.7.7)$$

where $\mathbf{c}_{amf}(\phi)$ is the optimum beamformer given by (11.3.13) with adaptive matched filter (AMF) normalization from Table 11.1

$$\mathbf{c}_{amf}(\phi) = \frac{\mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi)}{\sqrt{\mathbf{v}^H(\phi) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi)}} \quad (11.7.8)$$

as shown in Robey et al. (1992). This normalization is in contrast to MVDR normalized optimum beamformer in (11.3.15) that we have considered for the remainder of this chapter. Therefore, the ML angle estimator is the angle to which an AMF normalized optimum beamformer is steered that maximizes the output power for a given snapshot \mathbf{x} . In terms of the angle accuracy that might be achieved, the ML estimator can be approximated by forming a dense grid of optimum beamformers in angle with angular spacing at the desired

[†]In many applications, this assumption may be based on an up-front processing stage that determines the presence of the signal, known as *detection*.

minimum acceptable accuracy (Baranowski and Ward 1997). In many applications, we might want to achieve a much finer resolution than the beamwidth of the ULA, say, one-tenth of a beamwidth accuracy, known as 10:1 beamsplitting. Thus, this level of angle accuracy would require the computation of roughly $10M$ AMF optimum beamformers, where M is the number of sensors in the ULA. Generally, this requirement is computationally excessive, and we desire an alternative angle estimation algorithm that can achieve performance comparable to the ML estimator. This topic is addressed in Section 11.7.3. However, let us first consider the performance of the ML angle estimator that can be used as a bound for other angle estimation algorithms, which is the topic of the next section.

11.7.2 Cramér-Rao Lower Bound on Angle Accuracy

The *Cramér-Rao bound* (CRB) places a lower bound on the performance of an unbiased estimator (Kay 1993). We provide a sketch of the derivation of the CRB for angle accuracy (Ward 1996). This derivation is a simplification of the derivation by Ward (1996) that was done for two-dimensional angle and frequency estimation. Note that the CRB provides the minimum variance of an unbiased estimator. If an estimator can achieve the CRB, then it is the maximum-likelihood estimator. The CRB is found by solving for the diagonal elements of the inverse of the Fisher information matrix. For more details see Kay (1993) and Ward (1996).

Let us start by redefining the beamformer for a ULA from the spatial matched filter in (11.1.19) that has its phase center moved from the first element to the center of the array

$$\begin{aligned} \mathbf{v}_\Sigma(\phi) &= e^{-j2\pi \frac{M-1}{2} \frac{d}{\lambda} \sin \phi} \mathbf{v}(\phi) \\ &= \frac{1}{\sqrt{M}} \left[e^{-j2\pi \frac{M-1}{2} \frac{d}{\lambda} \sin \phi} \ e^{-j2\pi \frac{M-3}{2} \frac{d}{\lambda} \sin \phi} \ \dots \ e^{j2\pi \frac{M-1}{2} \frac{d}{\lambda} \sin \phi} \right]^T \end{aligned} \quad (11.7.9)$$

which we will refer to as the *sum beamformer*.[†] This choice of a phase center provides the tightest bound on accuracy (Rife and Boorstyn 1974). We can define a second beamformer based on the derivative of $\mathbf{v}_\Sigma(\phi)$ given by

$$\mathbf{v}_\Delta(\phi) = j\boldsymbol{\delta} \odot \mathbf{v}_\Sigma(\phi) \quad (11.7.10)$$

$$\text{where } \boldsymbol{\delta} = \left[-\frac{M-1}{2} \ -\frac{M-3}{2} \ \dots \ \frac{M-1}{2} \right]^T \quad (11.7.11)$$

which can be thought of as a difference taper. The steering vector $\mathbf{v}_\Delta(\phi)$, however, provides a difference pattern beamformer steered to the angle ϕ , as is commonly used in monopulse radar (Levanon 1988) for angle estimation purposes. For this reason, we refer to it as the *difference beamformer*. In relation to the sum beamformer, we can easily verify that

$$\mathbf{v}_\Delta^H(\phi) \mathbf{v}_\Sigma(\phi) = 0 \quad (11.7.12)$$

that is, the two beamformers are orthogonal to each other. The fact that the two beamformers are orthogonal to each other means that, in terms of the signal \mathbf{s} , the two beamformers can make two independent measurements of the signal. These independent measurements allow for the discrimination of the angle.

Using these two steering vectors $\mathbf{v}_\Delta(\phi)$ and $\mathbf{v}_\Sigma(\phi)$, we can form an adaptive sum beamformer

$$\mathbf{c}_\Sigma(\phi) = \mathbf{R}_{i+n}^{-1} \mathbf{v}_\Sigma(\phi) \quad (11.7.13)$$

[†]We use the term *beamformer* for interpretation of the Cramér-Rao bound only. No actual beams are formed since the CRB is only a performance bound and not a processing technique.

and an adaptive difference beamformer

$$\mathbf{c}_\Delta(\phi) = \mathbf{R}_{i+n}^{-1} \mathbf{v}_\Delta(\phi) \quad (11.7.14)$$

which both have not been normalized to satisfy any particular criteria. Proceeding, we can compute the power of the interference-plus-noise output of these two beamformers

$$P_\Sigma = \mathbf{c}_\Sigma^H \mathbf{R}_{i+n} \mathbf{c}_\Sigma \quad P_\Delta = \mathbf{c}_\Delta^H \mathbf{R}_{i+n} \mathbf{c}_\Delta \quad (11.7.15)$$

Similarly, we can measure the normalized cross-correlation $\rho_{\Sigma\Delta}$ of the interference-plus-noise outputs of these adaptive sum and difference beamformers \mathbf{R}_{i+n}

$$\rho_{\Sigma\Delta}^2 = \frac{|\mathbf{c}_\Sigma^H \mathbf{R}_{i+n} \mathbf{c}_\Delta|^2}{P_\Sigma P_\Delta} \quad (11.7.16)$$

Using (11.7.15) and (11.7.16), the CRB on angle estimation for a ULA is given by [†]

$$\sigma_\phi^2 \geq \frac{1}{2\pi^2 \cdot \text{SNR}_0 \cdot P_\Delta (1 - \rho_{\Sigma\Delta}^2) \cos^2 \phi} \quad (11.7.17)$$

where SNR_0 is the SNR for a spatial matched filter from (11.2.16) in the absence of interference, that is, noise only, which is given by

$$\text{SNR}_0 = M \frac{\sigma_s^2}{\sigma_w^2} \quad (11.7.18)$$

The CRB on angle accuracy has several interesting interpretations. First and foremost, as the signal power increases in value, SNR_0 increases; as a result, angle accuracy improves. Intuitively, this result makes sense as the stronger the signal of interest, the better the angle estimate should be. Likewise, the term $\cos^2 \phi$ simply represents the increase in beamwidth of the ULA as we steer away from broadside ($\phi = 0^\circ$). The interpretation of the other terms P_Δ and $1 - \rho_{\Sigma\Delta}^2$ may be less obvious, but also provides insight. Here P_Δ provides a measure of the received power aligned with the adaptive difference beamformer. On the other hand, $\rho_{\Sigma\Delta}$ is the cross-correlated energy between the adaptive sum and difference beamformers. Ideally, $\rho_{\Sigma\Delta}$ is zero, since \mathbf{c}_Σ and \mathbf{c}_Δ beamformers are derived from \mathbf{v}_Σ and \mathbf{v}_Δ , respectively, which are orthogonal to each other. In the case of the two adaptive beamformers, the adaptation will remove this orthogonality, but the beamformers should be different enough that $\rho_{\Sigma\Delta} \ll 1$. Otherwise, angle accuracy will suffer.

11.7.3 Beamsplitting Algorithms

Let us consider the scenario with a single beamformer steered to an angle ϕ_0 with our signal of interest at angle ϕ_s . The beamformer passes all signals within its beamwidth with only slight attenuation of signals that are not directly at the center of the beam steered to ϕ_0 . Clearly, this single beamformer cannot discriminate between signals received within its beamwidth. However, we desire a more accurate estimate of the angle of the signal of interest than simply the beamwidth of our beamformer. Thus, any angle estimator must achieve finer accuracy than the beamwidth, and as a result angle estimation algorithms are commonly referred to as *beamsplitting algorithms*.

To construct an angle estimation algorithm, it is necessary to obtain different measurements of the signal of interest in order to determine its angle. These measurements allow an angle estimation algorithm to discriminate between returns that arrive at the array from different angles. To this end, we use a set of beamformers steered in the general direction of

[†]This formulation assumes unit-variance thermal noise power. Therefore, if signals have different thermal noise power, the correlation matrix must be normalized by the thermal noise power prior to computing P_Δ and $\rho_{\Sigma\Delta}$.

the signal of interest but with different spatial responses, that is, beampatterns. One means of obtaining different measurements of the signal of interest is to slightly offset the steering direction of two beamformers. For example, we might form two beams at angles

$$\phi_1 = \phi_0 - \Delta\phi \quad \phi_2 = \phi_0 + \Delta\phi \quad (11.7.19)$$

where $\Delta\phi$ is a fraction of a beamwidth, for example, half a beamwidth. Let the weight vectors for these two beamformers be \mathbf{c}_1 and \mathbf{c}_2 , respectively. These two beamformers can be either nonadaptive, as in the case of the conventional beamformers discussed in Section 11.2, or one of the various adaptive beamformers from Section 11.3, 11.5, or 11.6. Ideally, a pair of adaptive beamformers is used for applications in which interference is encountered. Since the two beamformers are slightly offset from angle ϕ_0 , they may be thought of as “left” and “right” beamformers. Using the beamformer weight vectors, we can then form the ratio

$$\gamma_x = \frac{\mathbf{c}_1^H \mathbf{x}}{\mathbf{c}_2^H \mathbf{x}} \quad (11.7.20)$$

where recall that \mathbf{x} is the snapshot under consideration that contains the signal of interest $\mathbf{s} = \sqrt{M}\sigma_s \mathbf{v}(\phi_s)$. Similarly, we can also hypothesize this ratio for any angle ϕ to form a discrimination function

$$\gamma(\phi) = \frac{\mathbf{c}_1^H \mathbf{v}(\phi)}{\mathbf{c}_2^H \mathbf{v}(\phi)} \quad (11.7.21)$$

Comparing the value of the measured ratio in (11.7.20) for the snapshot \mathbf{x} to this angular discrimination function in (11.7.21), we obtain an estimate of the angle of the signal of interest ϕ_s . The key requirement for the discrimination function is that it be monotonic over the angular region in which it is used; that is, there is a one-to-one correspondence of the function in (11.7.21) and every angle in this region. The angular region typically encompasses the beamwidths of the two beamformers. This requirement on the discrimination function $\gamma(\phi)$ means that the two beamformers \mathbf{c}_1 and \mathbf{c}_2 must have different spatial responses.

We have simply given an example of how an angle estimation algorithm might be constructed. The topic of angle estimation is a very large area, and the choice of algorithm should be determined by the particular application. In the example given, we constructed a beamsplitting algorithm with left and right beams. Similarly, we could have chosen sum and difference beams, as is commonly done in radar in a technique known as monopulse (Sherman 1984). In fact, sum and difference beams can be formed from left and right beams by taking their sum and difference, respectively. In this case, a simple linear transformation exists that provides a mapping between the two beam strategies, and as a result one would anticipate equivalent performance. For further material on angle estimation algorithms, the interested reader is referred to Davis et al. (1974), McGarty (1974), Zoltowski (1992), and Nickel (1993).

11.7.4 Model-Based Methods

In Section 9.6, we discussed frequency estimation techniques based on a model of a complex exponential contained in noise. Certainly all these techniques could also be applied to the angle estimation problem, particularly for a ULA that has a direct correspondence to a discrete-time uniformly sampled signal. In this case, the angle is determined by the spatial frequency of the ULA. These methods are commonly referred to as *superresolution* techniques because they are able to achieve better resolution than traditional, nonadaptive methods. In fact, many of these techniques were originally proposed for array processing applications. However, certain considerations must be taken into account when one is trying

to apply these methods for use with a sensor array. First, a certain amount of uncertainty exists with respect to the exact spatial location of all the sensors. All these methods exploit the structure imposed by regular sampling where knowledge of the sampling instance is very precise. In the case of a temporally sampled signal, this assumption is very reasonable; but in the case of an array with these uncertainties, the validity of this assumption must be called into question. In addition, for a sensor array, all the signals are measured by different sensors with slightly different characteristics, as opposed to a temporally sampled signal for which all the samples are measured by the same sensor (analog-to-digital converter). Although these channel mismatches can be corrected for in theory, a perfect correction is never possible. For this reason, caution is in order when using these model-based methods for the purposes of angle estimation with an array.

11.8 SPACE-TIME ADAPTIVE PROCESSING

Space-time adaptive processing (STAP) is concerned with the two-dimensional processing of signals in both the spatial and temporal domains. The topic of STAP has received a lot of attention recently as it is a natural extension of array processing (Ward 1994, 1995; Klemm 1999). Although discussions of STAP date back to the early 1970s (Brennan and Reed 1973), the realization of STAP in an actual system was not possible until just recently, due to advances that were necessary in computing technology. We give a brief overview of the principles of STAP and cite some of the considerations for its practical implementation. Although STAP has also been proposed for use in communications systems (Paulraj and Papadias 1997), we primarily discuss it in the context of the airborne radar application for the purposes of clutter cancelation (Brennan and Reed 1973; Ward 1995; Klemm 1999).

A general STAP architecture is shown in Figure 11.29. Consider a ULA of sensors as we have discussed throughout this chapter. We choose a ULA for the sake of simplicity, but note that STAP techniques can be extended for arbitrary array configurations. In addition, the signal from each sensor consists of a set of time samples or delays that make up a

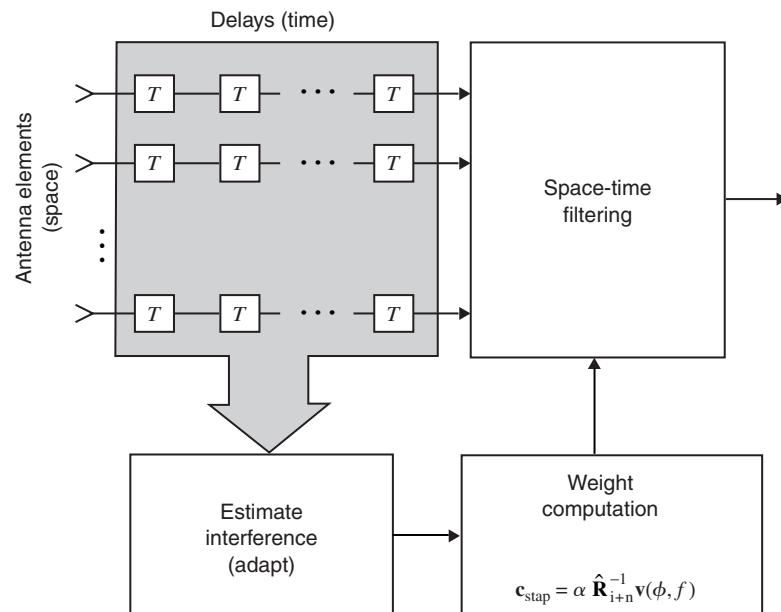


FIGURE 11.29
 Space-time adaptive processing.

time window. In radar applications, the time samples represent the returns from a set of transmitted pulses. For an airborne radar that moves with a certain velocity, the reflected signals from moving and nonmoving objects have a single frequency across the pulses. The pulse frequency results in a complex exponential across the pulses. The frequency is produced by the relative velocity of the objects with respect to the array and is known as the *Doppler frequency*. Thus, we wish to construct a space-time model for a signal received from a certain angle ϕ_s at a frequency f_s . We model the spatial component of the signal using the *spatial (sp) steering vector* for a ULA with M sensors from (11.1.23)

$$\mathbf{v}_{\text{sp}}(\phi_s) = \frac{1}{\sqrt{M}}[1 \ e^{-j2\pi[(d/\lambda)\sin\phi]} \ \dots \ e^{-j2\pi[(d/\lambda)\sin\phi](M-1)}]^T \quad (11.8.1)$$

Likewise, the temporal component of a signal that is a complex exponential can be modeled using a data window frequency vector, which technically is a *temporal frequency steering vector*. This temporal steering vector is given by

$$\mathbf{v}_{\text{time}}(f) = \frac{1}{\sqrt{L}}[1 \ e^{-j2\pi f} \ \dots \ e^{-j2\pi f(L-1)}]^T \quad (11.8.2)$$

where L is the number of time samples or pulses. Both \mathbf{v}_{sp} and \mathbf{v}_{time} have unit norm, that is, $\mathbf{v}_{\text{sp}}^H \mathbf{v}_{\text{sp}} = 1$ and $\mathbf{v}_{\text{time}}^H \mathbf{v}_{\text{time}} = 1$. Using these two one-dimensional steering vectors, we can form the two-dimensional $LM \times 1$ steering vector known as a *space-time steering vector*

$$\mathbf{v}_{\text{st}}(\phi, f) = \mathbf{v}_{\text{time}}(f) \otimes \mathbf{v}_{\text{sp}}(\phi) \quad (11.8.3)$$

where \otimes is the Kronecker product (Golub and Van Loan 1996). This vector, like the two one-dimensional steering vectors, has unit norm. Using this space-time steering vector, we can then model a spatially propagating signal arriving at the ULA from an angle ϕ_s with a frequency f_s as

$$\mathbf{s}(n) = \sqrt{LM} \mathbf{v}_{\text{st}}(\phi_s, f_s) s(n) \quad (11.8.4)$$

Of course, this signal of interest $\mathbf{s}(n)$ is not the only signal since the ULA at the very least will have thermal noise from the sensors. However, let us consider the case where in addition to the signal of interest, the ULA receives other spatially propagating signals that constitute interference $\mathbf{i}(n)$. Thus, the overall space-time signal in the ULA is

$$\mathbf{x}(n) = \mathbf{s}(n) + \mathbf{i}(n) + \mathbf{w}(n) \quad (11.8.5)$$

where $\mathbf{w}(n)$ is the sensor thermal noise space-time signal that is both temporally and spatially uncorrelated, that is, $E\{\mathbf{w}(n)\mathbf{w}^H(n)\} = \sigma_w^2 \mathbf{I}$. The interference component is made up of spatially propagating signals that may be temporally uncorrelated or consist of complex exponentials in the time domain, just as the signal of interest. In the case of an airborne radar, jamming interference is temporally uncorrelated while spatially correlated; that is, the jamming signal consists of uncorrelated noise that arrives from a certain angle ϕ . However, clutter returns are produced by reflections of the radar signal from the ground and have both spatial and temporal correlation. Due to the nature of the airborne radar problem, these clutter returns exhibit a certain structure that can be exploited for the purposes of implementing a STAP algorithm (Ward 1995; Klemm 1999).

As we did for the optimum beamformer, we want to find the optimum STAP weight vector. The optimality condition is again the maximization of the output SINR. The space-time interference-plus-noise correlation matrix is

$$\mathbf{R}_{i+n} = E\{\mathbf{x}_{i+n}(n)\mathbf{x}_{i+n}^H(n)\} \quad (11.8.6)$$

where

$$\mathbf{x}_{i+n}(n) = \mathbf{i}(n) + \mathbf{w}(n) \quad (11.8.7)$$

is the interference-plus-noise component of the signal. The availability of data that do not contain the signal of interest is a training issue for the implementation of the STAP algorithm that we do not consider here. See Borsari and Steinhardt (1995) and Rabideau and Steinhardt (1999).

The optimum STAP weight vector is found in a similar fashion to the optimum beamformer in Section 11.3. Using a unit gain on target constraint, the optimum STAP weight vector is

$$\mathbf{c}_{\text{stap}} = \frac{\mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s, f_s)}{\mathbf{v}^H(\phi_s, f_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s, f_s)} \quad (11.8.8)$$

where the space-time steering vector $\mathbf{v}(\phi_s, f_s)$ specifies the angle and frequency of the presumed signal of interest $\mathbf{s}(n)$. The implementation of STAP requires the estimation of \mathbf{R}_{i+n} from data samples in order to compute the sample correlation matrix $\hat{\mathbf{R}}_{i+n}$. This block adaptive implementation is also known as sample matrix inversion (SMI). SMI was discussed in the context of adaptive beamforming in Section 11.5.

The adaptive degrees of freedom of full STAP as specified in (11.8.8) are LM . For most applications, computational considerations as well as a limited amount of data to train the adaptive weights make the implementation of fully dimensional STAP impractical. Thus, we must consider reduced-dimension versions of STAP (Ward 1994, 1995). To this end, a preprocessing stage precedes the adaptation that reduces the degrees of freedom to an acceptable level. The most commonly considered approaches use a partially adaptive array implementation, as discussed in Section 11.6.2, either beamspace or subarrays, to reduce the spatial degrees of freedom. Temporal degrees of freedom can be reduced by using a frequency-selective temporal or Doppler filter (Ward and Steinhardt 1994). Alternatively, a subset of the total number of pulses can be used where the subsets of pulses are then combined following adaptive processing (Baranowski 1995).

A brief mention should be given to the application of STAP to the communications problem (Paulraj and Papadias 1997). Unlike in the radar application, it is generally not possible to separate the signal of interest from the interference-plus-noise in communications applications. In addition, although as in radar the signals are spatially propagating and thus arriving at the sensor array at a specific angle, for communications the temporal signals are not necessarily complex exponential signals. Instead, many times the signals consist of coded sequences. In this case, STAP must incorporate the codes rather than complex exponentials into its processing for the proper extraction of these signals.

11.9 SUMMARY

In this chapter, we have given a brief overview of array processing, starting with array fundamentals and covering optimum and adaptive beamforming. Throughout the chapter, we focused on the ULA, but it is possible to extend these concepts to other array configurations. The background material included spatially propagating signals, the concepts of modulation and demodulation, and the model for a spatial signal received by a ULA. We drew the analogy between the ULA, in terms of spatial sampling, and the discrete-time sampling of temporal signals. Next, we introduced the topic of conventional beamforming for which we discussed the spatial matched filter, which maximizes the SNR in the absence of interference, and tapered low sidelobe beamformers. Within this context, we looked at the characteristics of an array in terms of resolution and ambiguities known as grating lobes.

The remainder of the chapter dealt with optimum and adaptive beamforming techniques related to the processing introduced earlier in the book for use with discrete-time signals. These methods are concerned with adapting to the characteristics of the data, either assuming knowledge of these characteristics (optimum) or estimating them from the data (adaptive). One might say that the *fundamental equation* in adaptive beamforming is $\mathbf{c} = \mathbf{R}^{-1} \mathbf{v}(\phi_s)$, where $\mathbf{v}(\phi_s)$ determines the direction ϕ_s in which we are steering and \mathbf{R}^{-1} , the inverse of the array correlation matrix, performs the adaptation to the array data. Within this context, we looked at various issues, such as sidelobe levels and interference cancelation, and the

effects of signal mismatch and bandwidth. The more advanced topics of angle estimation and STAP were also discussed.

Throughout this chapter, we have tried to remain general in our treatment of the array processing principles. Ultimately, specific issues and concerns related to the application will dictate the type of processing that is needed. Areas in which arrays are commonly employed include radar, sonar, and communications. In parts of the chapter, we have used examples based on radar applications since they tend to be the easiest to simplify and describe. Other important issues not discussed that arise in radar include the nonstationarity of the signals as well as training strategies. The sonar application is rich with issues that make the implementation of adaptive arrays a very challenging task. Propagation effects, including signal multipath, lead to complicated models that must be used to estimate the steering vectors. In addition, signals of interest tend to be present at all times, so that the adaptive beamformer must be trained with the signal of interest present. As we described in Section 11.4.1, this situation leads to a heightened sensitivity to signal mismatch. For more details see Baggeroer et al. (1993). Arrays for communications applications have also become a very popular field owing to the rapid growth of the wireless communications industry. The fundamental issue for wireless communications is the number of users that can be simultaneously accommodated. The limitations arise from the interference produced by other users. Arrays can help to increase the capacity in terms of the number of users. For more details, see Litva and Lo (1996).

We have presented some material on the more advanced topics of angle estimation and STAP. Another extension of adaptive beamforming is the subject of adaptive detection. This topic is concerned with the determination of the presence of signals of interest in which the detector is determined adaptively from the data. References on this subject include Kelly (1986), Steinhardt (1992), Robey et al. (1992), Bose and Steinhardt (1995), Scharf and McWhorter (1997), Kreithen and Steinhardt (1996), Conte et al. (1996), and Richmond (1997).

PROBLEMS

- 11.1** Consider a narrowband spatially propagating signal with a speed of propagation c . The signal impinges on an $M = 2$ element ULA from an angle $\phi = 0^\circ$ with a spacing d between the elements. For illustration purposes, let the temporal content of the signal be a pulse.
- Let the time of arrival of the pulse at the first sensor be $t = 0$. At what time does the signal arrive at the second sensor?
 - Do any other angles ϕ produce the same delay between the two sensors? Why?
 - Suppose now that we only have a single sensor. Can we determine the angle from which a signal impinges on this sensor?
- 11.2** We want to investigate the use of a mechanically steered versus an electronically steered array. Consider a spatial matched filter with $M = 20 \lambda/2$ -spaced elements. Now consider that the array is steered to $\phi = 45^\circ$. In the case of mechanical steering, the pointing direction is always broadside to the array. To compute the beampattern of the mechanically steered array, simply take the beampattern computed at $\phi = 0^\circ$ and shift it by the mechanical steering angle, that is, $\phi' = \phi + \phi_{\text{mech}}$. However, the beampattern of an electronically steered array is simply the beampattern of the spatial matched filter steered to the desired angle. Compare the beampattern of the mechanically steered array to that of the electronically steered array. What do you observe? Repeat this for $\phi = 60^\circ$ steering, both electronic and mechanical.
- 11.3** In this problem, we want to explore the use of beampatterns and steered or spatial responses of a ULA. Consider a signal $\mathbf{x}(n)$ consisting of two spatially propagating signals from $\phi_1 = -10^\circ$ and $\phi_2 = 30^\circ$, both made of random, complex Gaussian noise. The respective powers of the two signals are 20 and 25 dB. The number of sensors in the ULA is 50, and its thermal noise level is 0 dB. The ULA has interelement spacing of $d = \lambda/2$.
- Compute one realization of $\mathbf{x}(n)$ for $N = 1000$ samples, that is, $1 \leq n \leq N$. Using a spatial matched filter, compute a steered response for this signal from the beamformer

- outputs, and plot it in decibels versus angle. What do you observe? Compare the result to the expected steered response using the true correlation matrix.
- (b) Compute and plot the beampattern for the spatial matched filter steered to $\phi = 30^\circ$. How can you relate the power levels you observed in part (a) at the angles of the two signals to the beampattern?
- (c) Change the power level of the signal at $\phi = -10^\circ$ to 60 dB, and compute the steered response. What do you observe? What do you recommend in order to distinguish these two signals? Implement your idea and plot the estimated steered response from the beamformer outputs.
- 11.4** Suppose that we have an $M = 30$ element ULA with a thermal noise level of 0 dB.
- Generate a realization of the ULA signal $\mathbf{x}(n)$ consisting of two random, complex Gaussian signals at $\phi = 0^\circ$ and $\phi = 3^\circ$ both with power 20 dB, along with the sensor thermal noise. The interelement spacing is $d = \lambda/2$. Let the number of samples you generate be $N = 1000$. Compute and plot the steered response of $\mathbf{x}(n)$ using a spatial matched filter. What do you observe?
 - Repeat part (a) for an $M = 60$ element ULA. What do you observe?
 - Now using the $M = 30$ element ULA again, but with interelement spacing $d = \lambda$, compute the steered response and comment on the result.
 - Compute the beampatterns for the spatial matched filter steered to $\phi = 0^\circ$ for the three array configurations in parts (a), (b), and (c).
- 11.5** In this problem, we want to investigate the use of randomly thinned arrays. Note that the $M = 30$ element ULA with $d = \lambda$ spacing from Problem 11.4 is simply the $M = 60$ element ULA with every other element deleted. Such an array is often referred to as a *thinned array*. Using an $M = 60$ element array, randomly thin the array. (*Hint:* Use a random number generator.) First thin to 75 percent (45 elements) and then to 50 percent (30 elements). Compute and plot the steered response, using a spatial matched filter for the signal in Problem 11.4. Note that the spatial matched filter must take into account the positions of the elements; that is, it is no longer a Vandermonde steering vector. Compute the beampatterns of these two randomly thinned arrays. Repeat this process 3 times. What do you observe?
- 11.6** The spatial matched filter from (11.2.16) is the beamformer that maximizes the SNR in the absence of interference. For this spatial matched filter, the beamforming or array gain was shown to be $G_{bf} = M$. Suppose now that we have an $M = 20$ element ULA in which the elements have unequal gain. In other words, the spatial matched filter no longer has the same amplitude in each element. Find the spatial matched filter for the case when all even-numbered elements have a unity gain, while all the odd-numbered elements have a gain of $\frac{3}{4}$. What is the beamforming gain for this array? The noise has equal power in all elements.
- 11.7** The optimum beamformer weights with MVDR normalization are found by solving the following optimization
- $$\min P_{i+n} \quad \text{subject to} \quad \mathbf{c}^H \mathbf{v}(\phi_s) = 1$$
- Using Lagrange multipliers discussed in Appendix B, show that the MVDR optimum beamformer weight vector is
- $$\mathbf{c}_o = \frac{\mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)}{\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)}$$
- 11.8** In this problem, we want to investigate the different normalizations of the optimum beamformer from Table 11.1. We refer to the three normalizations as MVDR ($\alpha = [\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)]^{-1}$), adaptive matched filter or AMF ($\alpha = [\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)]^{-1/2}$), and unit gain on noise ($\alpha = [\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-2} \mathbf{v}(\phi_s)]^{-1/2}$). Let the interference-plus-noise signal consist of two interference sources at $\phi = 45^\circ$ and $\phi = 20^\circ$ with powers of 30 and 15 dB, respectively. The noise power is $\sigma_w^2 = 1$. Now compute the steered response of the optimum beamformers with the three different normalizations between $-90^\circ < \phi < 90^\circ$ (using 1° increments), using the true

correlation matrix. What is the difference in the outputs of the optimum beamformers with the different normalizations? For what purposes are the different normalizations useful?

- 11.9** The generalized sidelobe canceler (GSC) was derived as an alternative implementation of the MVDR optimum beamformer. Show that the overall end-to-end weights associated with the GSC are equivalent to the MVDR optimum beamformer weight vector in (11.3.15).

- 11.10** In the formulation of the optimum beamformer, we used the interference-plus-noise correlation matrix \mathbf{R}_{i+n} . However, in many applications it is not possible to have interference-plus-noise-only data, and the signal of interest is always present. Thus, the beamformer must be implemented using the correlation matrix

$$\mathbf{R}_x = \mathbf{R}_{i+n} + \sigma_s^2 \mathbf{v}(\phi_s) \mathbf{v}^H(\phi_s)$$

Show that the use of this correlation matrix will have no effect on the optimum beamformer weight vector for the case of no signal mismatch. *Hint:* Use the matrix inversion lemma (see Appendix A).

- 11.11** In this problem, we want to look at the effect of signal mismatch on the performance of the optimum beamformer. Of course, the resulting beamformer is no longer really optimum, but instead is optimized to our presumptions about the signal. Consider the case with three interference sources at $\phi = 5^\circ, 20^\circ$, and -30° with powers of 25, 35, and 50 dB, respectively. Compute the optimum beamformer steered to $\phi = 0^\circ$. Now consider the case where the signal of interest is not at $\phi = 0^\circ$ but rather at $\phi = -1^\circ$. The array consists of an $M = 50$ element ULA with a noise power of $\sigma_w^2 = 1$.

- (a) Find the signal mismatch loss when the signal of interest is not in the correlation matrix. Vary the strength of the signal from 0 to 30 dB.
- (b) Find the signal mismatch loss when the signal is in the correlation matrix. Vary the strength of the signal from 0 to 30 dB.

- 11.12** Let us again consider a set of three interference sources at $\phi = 5^\circ, 20^\circ$, and -30° with powers of 25, 35, and 50 dB, respectively. Now consider the case where the signal of interest is not at $\phi = 0^\circ$ but rather at $\phi = -1^\circ$ and has a power of $M\sigma_s^2 = 20$ dB. The array consists of an $M = 50$ element ULA with a noise power of $\sigma_w^2 = 1$. However, instead of computing the optimum beamformer with the correlation matrix \mathbf{R}_{i+n} , use the diagonally loaded interference-plus-noise matrix

$$\mathbf{R}_l = \mathbf{R}_{i+n} + \sigma_l^2 \mathbf{I}$$

where σ_l^2 is the loading level.

- (a) Find the signal mismatch loss when the signal of interest is not in the correlation matrix. Compute and plot the mismatch loss varying the diagonal loading level from 0 to 20 dB.
- (b) Find the signal mismatch loss when the signal is in the correlation matrix. Compute and plot the mismatch loss varying the diagonal loading level from 0 to 20 dB.

- 11.13** The Frost sample-by-sample adaptive beamformer was derived for the MVDR beamformer. Extend the Frost sample-by-sample adaptive beamformer for the case of multiple constraints in an LCMV adaptive beamformer.

- 11.14** The LCMV beamformer weight vector is given in (11.6.7) and was found by using Lagrange multipliers, which are discussed in Appendix B. Verify this result; that is, using Lagrange multipliers, show that the LCMV beamformer weight vector is given by

$$\mathbf{c}_{lcmv} = \mathbf{R}_{i+n}^{-1} \mathbf{C} (\mathbf{C}^H \mathbf{R}_{i+n}^{-1} \mathbf{C})^{-1} \boldsymbol{\delta}$$

where \mathbf{C} and $\boldsymbol{\delta}$ are defined as in Section 11.6.1.

- 11.15** Let us consider the sidelobe canceler, as discussed in Section 11.6.3. We restrict the problem to a single interferer that has an angle ϕ_i with respect to a ULA that makes up the auxiliary

$$x_{\text{mc}}(n) = g_s s(n) + i_{\text{mc}}(n) + w_{\text{mc}}(n)$$

where $i_{\text{mc}}(n) = g_i i(n)$ is the temporally uncorrelated signal $i(n)$ with unit variance that has a main channel gain of g_i . The main channel thermal noise $w_{\text{mc}}(n)$ is temporally uncorrelated with a power of σ_0^2 . The auxiliary channels make up an M -element ULA with thermal noise σ_w^2 . The auxiliary channel signal vector is given by

$$\mathbf{x}(n) = s(n)\mathbf{v}(\phi_s) + \sigma_i i(n)\mathbf{v}(\phi_i) + \mathbf{w}(n)$$

where ϕ_s and ϕ_i are the angles of the signal of interest and the interferer with respect to the ULA, respectively.

- (a) Form the expressions for the auxiliary channel correlation matrix \mathbf{R}_a and cross-correlation vector \mathbf{r}_{ma} that include the signal of interest in the auxiliary channels.
- (b) Compute the output power of the interference-plus-noise.
- (c) Compute the output power of the signal. What conclusions can you draw from your answer?

11.16 The MVDR optimum beamformer is simply a special case of the LCMV beamformer. In this case, the constraint matrix is $\mathbf{C} = \mathbf{v}(\phi_s)$ and the constraint response vector is $\delta = 1$.

- (a) Using the LCMV weight vector given in (11.6.7), substitute this constraint and constraint response and verify that the resulting beamformer weight vector is equal to the MVDR optimum beamformer.
- (b) Find an expression for the interference-plus-noise output power of the LCMV beamformer.

11.17 The optimum beamformer could also be formulated as the constrained optimization problem that resulted in the MVDR beamformer. This beamformer can be implemented as a generalized sidelobe canceler (GSC), as shown in Section 11.3.5. Similarly, the LCMV beamformer can be implemented in a GSC architecture. Derive the formulation of a GSC with multiple linear constraints.

11.18 Consider the case of an $M = 20$ element array with $d = \lambda/2$ interelement spacing and thermal noise power $\sigma_w^2 = 1$. An interference source is present at $\phi = 30^\circ$ with a power of 50 dB. Generate one realization of 1000 samples of this interferer. In addition, a signal of interest is present at $\phi_s = 0^\circ$ with a power of $\sigma_s = 100$ (20 dB) in the $n = 100$ th sample only.

- (a) Using an SMI adaptive beamformer for the full array, compute the output signal. Is the signal of interest visible?
- (b) Using a partially adaptive beamformer with $Q = 4$ nonoverlapping subarrays with $\tilde{M} = 5$ elements, compute the output of an SMI adaptive beamformer. What can you say about the signal of interest now?
- (c) Repeat part (b) with $Q = 2$ and $\tilde{M} = 10$. What are your observations now?

11.19 Consider the case of an $M = 40$ element array with $d = \lambda/2$ interelement spacing and thermal noise power $\sigma_w^2 = 1$. An interference source is present at $\phi = 20^\circ$ with a power of 50 dB. Generate one realization of 1000 samples of this interferer. In addition, a signal of interest is present at $\phi_s = 0^\circ$ with a power of $\sigma_s = 100$ (20 dB) in the $n = 100$ th sample only.

- (a) Using an SMI adaptive beamformer for the full array, compute the output signal. Is the signal of interest visible?
- (b) Using a beamspace partially adaptive beamformer consisting of 11 beams at the angles $-5^\circ \leq \phi \leq 5^\circ$ at 1° increments, compute the output of a partially adaptive SMI beamformer. What can you say about the signal of interest now?
- (c) Repeat part (b) with beams only at $\phi = -1^\circ, 0^\circ$, and 1° . What are your observations now?

11.20 Compute the SINR loss for a partially adaptive beamformer with a general preprocessing transformation \mathbf{T} . You need to start with the general definition of SINR loss

$$L_{\text{sincr}} \triangleq \frac{\text{SINR}_{\text{out}}(\phi_s)}{\text{SNR}_0}$$

where $\text{SINR}_{\text{out}}(\phi_s)$ is the output SINR of the partially adaptive beamformer at angle ϕ_s and SNR_0 is the SNR of the spatial matched filter in the absence of interference.

- 11.21** Consider the case of an interference source at $\phi = 30^\circ$ with a power of 40 dB. The ULA is a 20-element array with $d = \lambda/2$ interelement spacing and has unit-variance thermal noise ($\sigma_w^2 = 1$).

- (a) Compute the SINR loss for the optimum beamformer for $-90^\circ \leq \phi \leq 90^\circ$.
- (b) Let us consider the case of the GSC formulation of the optimum beamformer. If we choose to use a beamspace blocking matrix \mathbf{B} in (11.3.41), what are the spatial frequencies of the spatial matched filters in this blocking matrix for an optimum beamformer steered to $\phi = 0^\circ$?
- (c) To implement a reduced-rank or partially adaptive beamformer, use only the two spatial matched filters in the beamspace blocking matrix with spatial frequencies closest to the interference source (spatial frequency $u = \frac{1}{2} \sin \phi_i = 0.25$). Compute the SINR loss of this partially adaptive beamformer, and compare it to the SINR loss of the optimum beamformer found in part (a).

Further Topics

The distinguishing feature of this book, up to this point, is the reliance on random process models having finite variance and short memory and specified by their second-order moments. This chapter deviates from this path by focusing on further topics where there is an explicit or implicit need for higher-order moments, long memory, or high variability.

In the first part (Section 12.1), we introduce the area of *higher-order statistics (HOS)* with emphasis on the concepts of cumulants and polyspectra. We define cumulants and polyspectra; we analyze the effect of linear, time-invariant systems upon the HOS of the input process; and we derive the HOS of linear processes. Higher-order moments, unlike second-order moments, are shown to contain phase information and can be used to solve problems in which phase is important.

In the second part (Sections 12.2 through 12.4), we illustrate the importance of HOS for the blind deconvolution of non-minimum-phase systems, and we show how the underlying theory can be used to design unsupervised adaptive filters for symbol-spaced and fractionally spaced equalization of data communication channels.

In the third part (Sections 12.5 and 12.6), we introduce two types of random signal models characterized by long memory: fractional and self-similar, or random, fractal models. We conclude with rational and fractional models with symmetric α -stable (S α S) excitations and self-similar processes with S α S increments. These models have long memory and find many applications in the analysis and modeling of signals with long-range dependence and impulsive or spiky behavior.

12.1 HIGHER-ORDER STATISTICS IN SIGNAL PROCESSING

The statistics of a Gaussian process are completely specified by its second-order moments, that is, correlations and power spectral densities (see Section 3.3). Since non-Gaussian processes do *not* have this property, their higher-order statistics contain additional information that can be used to measure their deviation from normality. In this section we provide some background definitions and properties of higher-order moments, and we discuss their transformation by linear, time-invariant systems. More detailed treatments can be found in Mendel (1991), Nikias and Raghubeer (1987), Nikias and Mendel (1993), and Rosenblatt (1985).

12.1.1 Moments, Cumulants, and Polyspectra

The first four moments of a complex-valued stationary stochastic process are defined by

$$r_x^{(1)} \triangleq E\{x(n)\} = \mu_x \quad (12.1.1)$$

$$r_x^{(2)}(l_1) \triangleq E\{x^*(n)x(n+l_1)\} = r_x(l_1) \quad (12.1.2)$$

$$r_x^{(3)}(l_1, l_2) \triangleq E\{x^*(n)x(n+l_1)x(n+l_2)\} \quad (12.1.3)$$

$$r_x^{(4)}(l_1, l_2, l_3) \triangleq E\{x^*(n)x^*(n+l_1)x(n+l_2)x(n+l_3)\} \quad (12.1.4)$$

although other definitions are possible by conjugating different terms. We note that the first two moments are the mean and the autocorrelation sequence, respectively.

In Section 3.2.4 we showed that the cumulant of a linear combination of IID random variables can be determined by a linear combination of their cumulants. In addition, in Section 3.1.2, we noted that the kurtosis of a random variable measures its deviation from Gaussian behavior. For these reasons, we usually prefer to work with cumulants instead of moments. Since higher-order cumulants are invariant to a shift of the mean value, we define them under a zero-mean assumption.

The first four cumulants of a zero-mean stationary process are defined by

$$\kappa_x^{(1)} = E\{x(n)\} = \mu_x = 0 \quad (12.1.5)$$

$$\kappa_x^{(2)}(l_1) = E\{x^*(n)x(n+l_1)\} = r_x(l_1) \quad (12.1.6)$$

$$\kappa_x^{(3)}(l_1, l_2) = E\{x^*(n)x(n+l_1)x(n+l_2)\} \quad (12.1.7)$$

$$\begin{aligned} \kappa_x^{(4)}(l_1, l_2, l_3) &= E\{x^*(n)x^*(n+l_1)x(n+l_2)x(n+l_3)\} \\ &\quad - \kappa_x^{(2)}(l_2)\kappa_x^{(2)}(l_3-l_1) - \kappa_x^{(2)}(l_3)\kappa_x^{(2)}(l_2-l_1) \end{aligned} \quad (12.1.8)$$

(complex-valued case)

$$\begin{aligned} \kappa_x^{(4)}(l_1, l_2, l_3) &= E\{x(n)x(n+l_1)x(n+l_2)x(n+l_3)\} - \kappa_x^{(2)}(l_1)\kappa_x^{(2)}(l_3-l_2) \\ &\quad - \kappa_x^{(2)}(l_2)\kappa_x^{(2)}(l_3-l_1) - \kappa_x^{(2)}(l_3)\kappa_x^{(2)}(l_2-l_1) \end{aligned} \quad (12.1.9)$$

(real-valued case)

and can be obtained by using the cumulant-generating function discussed in Section 3.1.2 (Mendel 1991). It can be shown that

$$\kappa_x^{(k)}(l_1, l_2, \dots, l_{k-1}) = \mu_x^{(k)}(l_1, l_2, \dots, l_{k-1}) - \mu_g^{(k)}(l_1, l_2, \dots, l_{k-1}) \quad k = 3, 4 \quad (12.1.10)$$

where $x(n)$ is a non-Gaussian process and $g(n)$ is a Gaussian process with the same mean and autocorrelation sequence. The negative terms in (12.1.8) and (12.1.9) express the fourth-order cumulant of the Gaussian process in terms of second-order ones. In this sense, in addition to higher-order correlations, cumulants measure the distance of a process from Gaussianity. Note that if $x(n)$ is Gaussian, $\kappa_x^{(k)}(l_1, l_2, \dots, l_{k-1}) = 0$ for all $k \geq 3$ even if Equation (12.1.10) holds only for $k = 3, 4$.

If we assume that $\mu_x = 0$ and set $l_1 = l_2 = l_3 = 0$ in (12.1.6) through (12.1.8), we obtain

$$\kappa_x^{(2)}(0) = E\{|x(n)|^2\} = \sigma_x^2 \quad (12.1.11)$$

$$\kappa_x^{(3)}(0, 0) = \gamma_x^{(3)} \quad (12.1.12)$$

$$\kappa_x^{(4)}(0, 0, 0) = E\{|x(n)|^4\} - 2\sigma_x^4 \quad \text{complex} \quad (12.1.13)$$

$$= E\{x^4(n)\} - 3\sigma_x^4 \quad \text{real} \quad (12.1.14)$$

which provide the variance, unnormalized skewness, and unnormalized kurtosis of the process (see Section 3.1.2).

If the probability distribution of a process is symmetric (e.g., uniform, Gaussian, Laplace), its third-order cumulants are zero. In such cases, we need to consider fourth-order cumulants. Higher-order cumulants ($k > 4$) are seldom used in practice.

If the cumulants are absolutely summable, we can define the k th-order *cumulant spectra*, *higher-order spectra*, or *polyspectra* as the $(k-1)$ -dimensional Fourier transform of the k th-order cumulant. More specifically, the power spectral density (PSD), bispectrum, and trispectrum of a zero-mean stationary process are defined by

$$R_x^{(2)}(e^{j\omega}) \triangleq \sum_{l_1=-\infty}^{\infty} \kappa_x^{(2)}(l_1) e^{-j\omega l_1} = R_x(e^{j\omega}) \quad (12.1.15)$$

$$R_x^{(3)}(e^{j\omega_1}, e^{j\omega_2}) \triangleq \sum_{l_1=-\infty}^{\infty} \sum_{l_2=-\infty}^{\infty} \kappa_x^{(3)}(l_1, l_2) e^{-j(\omega_1 l_1 + \omega_2 l_2)} \quad (\text{bispectrum})$$

(12.1.16)

$$\text{and } R_x^{(4)}(e^{j\omega_1}, e^{j\omega_2}, e^{j\omega_3}) \triangleq \sum_{l_1=-\infty}^{\infty} \sum_{l_2=-\infty}^{\infty} \sum_{l_3=-\infty}^{\infty} \kappa_x^{(4)}(l_1, l_2, l_3) e^{-j(\omega_1 l_1 + \omega_2 l_2 + \omega_3 l_3)} \quad (\text{trispectrum})$$

(12.1.17)

where ω_1 , ω_2 , and ω_3 are the frequency variables. In contrast to the PSD, which is real-valued and nonnegative, both the bispectrum and the trispectrum are complex-valued. Since the higher-order cumulants of a Gaussian process are zero, its bispectrum and trispectrum are zero as well.

Many symmetries exist in the arguments of cumulants and polyspectra of both real and complex stochastic processes (Rosenblatt 1985; Nikias and Mendel 1993). For example, from the obvious symmetry

$$r_x^{(3)}(l_1, l_2) = r_x^{(3)}(l_2, l_1) \quad (12.1.18)$$

$$\text{we obtain } R_x^{(3)}(e^{j\omega_1}, e^{j\omega_2}) = R_x^{(3)}(e^{j\omega_2}, e^{j\omega_1}) \quad (12.1.19)$$

which is a basic property of the bispectrum.

For real-valued processes, we have the additional symmetries

$$\begin{aligned} r_x^{(3)}(l_1, l_2) &= r_x^{(3)}(-l_2, l_1 - l_2) = r_x^{(3)}(-l_1, l_2 - l_1) \\ &= r_x^{(3)}(l_2 - l_1, -l_1) = r_x^{(3)}(l_1 - l_2, -l_2) \end{aligned} \quad (12.1.20)$$

$$\begin{aligned} r_x^{(4)}(l_1, l_2, l_3) &= r_x^{(4)}(l_2, l_1, l_3) = r_x^{(4)}(l_1, l_3, l_2) \\ &= r_x^{(4)}(-l_1, l_2 - l_1, l_3 - l_1) \end{aligned} \quad (12.1.21)$$

which can be used to simplify the computation of cumulants. It can be shown that the nonredundant region for $r_x^{(3)}(l_1, l_2)$ is the wedge $\{(l_1, l_2) : 0 \leq l_2 \leq l_1 \leq \infty\}$ and for $r_x^{(4)}(l_1, l_2, l_3)$ is the cone $\{(l_1, l_2, l_3) : 0 \leq l_3 \leq l_2 \leq l_1 \leq \infty\}$. The symmetries of cumulants impose symmetry properties upon the polyspectra. Indeed, by using (12.1.20) it can be shown that

$$\begin{aligned} R_x^{(3)}(e^{j\omega_1}, e^{j\omega_2}) &= R_x^{(3)}(e^{j\omega_2}, e^{j\omega_1}) = R_x^{(3)}(e^{j\omega_1}, e^{-j\omega_1-j\omega_2}) \\ &= R_x^{(3)}(e^{-j\omega_1-j\omega_2}, e^{j\omega_2}) = R_x^{(3)*}(e^{-j\omega_1}, e^{-j\omega_2}) \end{aligned} \quad (12.1.22)$$

which implies that the nonredundant region for the bispectrum is the triangle with vertices at $(0, 0)$, $(2\pi/3, 2\pi/3)$, and $(\pi, 0)$. The trispectrum of a real-valued process has 96 symmetry regions (Pflug et al. 1992).

Finally, we note that if in (12.1.7) we replace $x(n + l_1)$ by $y(n + l_1)$ and $x(n + l_2)$ by $z(n + l_2)$, we can define the cross-cumulant and then take its Fourier transform to find the cross-bispectrum. These quantities are useful for joint signal analysis (Nikias and Mendel 1993).

12.1.2 Higher-Order Moments and LTI Systems

Consider a BIBO stable linear, time-invariant (LTI) system with impulse response $h(n)$ and input-output relation given by

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) \quad (12.1.23)$$

If the input $x(n)$ is stationary with zero mean, the output autocorrelation is

$$r_y(l) = \sum_{k_0} \sum_{k_1} h(k_1)h^*(k_0)r_x(l - k_1 - k_0) \quad (12.1.24)$$

where the range of summations, which is from $-\infty$ to ∞ , is dropped for convenience (see Section 3.4). Also we have

$$R_y(e^{j\omega}) = |H(e^{j\omega})|^2 R_x(e^{j\omega}) \quad (12.1.25)$$

which shows that the output PSD is *insensitive* to the phase response of the system.

Using (12.1.23) and (12.1.7), we can show that the input and output third-order cumulants are related by

$$\kappa_y^{(3)}(l_1, l_2) = \sum_{k_0} \sum_{k_1} \sum_{k_2} h^*(k_0)h(k_1)h(k_2)\kappa_x^{(3)}(l_1 - k_1 + k_0, l_2 - k_2 + k_0) \quad (12.1.26)$$

To obtain the fourth-order cumulant of the output, we first determine its fourth-order moment

$$\begin{aligned} r_y^{(4)}(l_1, l_2, l_3) &= \sum_{k_0} \sum_{k_1} \sum_{k_2} \sum_{k_3} h^*(k_0)h^*(k_1)h(k_2)h(k_3) \\ &\quad \times r_x^{(4)}(l_1 - k_1 + k_0, l_2 - k_2 + k_0, l_3 - k_3 + k_0) \end{aligned} \quad (12.1.27)$$

using (12.1.23) and (12.1.4) (see Problem 12.1). Then using (12.1.8), (12.1.9), and (12.1.24), we have

$$\begin{aligned} \kappa_y^{(4)}(l_1, l_2, l_3) &= \sum_{k_0} \sum_{k_1} \sum_{k_2} \sum_{k_3} h^*(k_0)h^*(k_1)h(k_2)h(k_3) \\ &\quad \times \kappa_x^{(4)}(l_1 - k_1 + k_0, l_2 - k_2 + k_0, l_3 - k_3 + k_0) \end{aligned} \quad (12.1.28)$$

which holds for both real- and complex-valued processes. An interesting interpretation of this relationship in terms of convolutions is given in Mendel (1991) and in Therrien (1992).

We now compute the bispectrum of the output signal $y(n)$ in terms of the bispectrum of the input $x(n)$ and the frequency response $H(e^{j\omega})$ of the system. Indeed, taking the two-dimensional Fourier transform of (12.1.26) and interchanging the order of summations, we have

$$\begin{aligned} R_y^{(3)}(e^{j\omega_1}, e^{j\omega_2}) &= \sum_{l_1} \sum_{l_2} \kappa_y^{(3)}(l_1, l_2)e^{-j(\omega_1 l_1 + \omega_2 l_2)} \\ &= \sum_{l_1} \sum_{l_2} \sum_{k_0} \sum_{k_1} \sum_{k_2} h^*(k_0)h(k_1)h(k_2) \\ &\quad \times \kappa_x^{(3)}(l_1 - k_1 + k_0, l_2 - k_2 + k_0)e^{-j(\omega_1 l_1 + \omega_2 l_2)} \\ &= \sum_{k_0} h^*(k_0)e^{j(\omega_1 + \omega_2)k_0} \sum_{k_1} h(k_1)e^{-j\omega_1 k_1} \sum_{k_2} h(k_2)e^{-j\omega_2 k_2} \\ &\quad \times \sum_{l_1} \sum_{l_2} \kappa_x^{(3)}(l_1 - k_1 + k_0, l_2 - k_2 + k_0)e^{-j\omega_1(l_1 - k_1 + k_0)}e^{-j\omega_2(l_2 - k_2 + k_0)} \end{aligned}$$

Rearranging terms and using (12.1.16), we obtain

$$R_y^{(3)}(e^{j\omega_1}, e^{j\omega_2}) = H^*(e^{j\omega_1 + j\omega_2})H(e^{j\omega_1})H(e^{j\omega_2})R_x^{(3)}(e^{j\omega_1}, e^{j\omega_2}) \quad (12.1.29)$$

which shows that the bispectrum, in contrast to the PSD in (12.1.25), is sensitive to the phase of the system.

In a similar, but more complicated way, we can show that the output trispectrum is given by

$$\begin{aligned} R_y^{(4)}(e^{j\omega_1}, e^{j\omega_2}, e^{j\omega_3}) &= H^*(e^{j\omega_1 + j\omega_2 + j\omega_3})H^*(e^{-j\omega_1})H(e^{j\omega_2}) \\ &\quad \times H(e^{j\omega_3})R_x^{(4)}(e^{j\omega_1}, e^{j\omega_2}, e^{j\omega_3}) \end{aligned} \quad (12.1.30)$$

which again shows that the trispectrum is sensitive to the phase of the system.

12.1.3 Higher-Order Moments of Linear Signal Models

695

SECTION 12.1

Higher-Order Statistics in
Signal Processing

In Section 4.1 we discussed linear signal models and the innovations representation of stationary processes with given second-order moments. This representation is given by

$$x(n) = \sum_{k=0}^{\infty} h(k)w(n-k) \quad (12.1.31)$$

$$r_x(l) = \sigma_w^2 \sum_{k=0}^{\infty} h(k)h^*(k-l) \quad (12.1.32)$$

$$R_x(e^{j\omega}) = \sigma_w^2 |H(e^{j\omega})| \quad (12.1.33)$$

where $w(n)$ is a white noise process and $H(z)$ is a minimum phase. If $w(n)$ is Gaussian, $x(n)$ is Gaussian and this representation provides a complete description of the process.

If the excitation $w(n)$ is IID and *non-Gaussian* with cumulants

$$\kappa_w^{(k)}(l_1, l_2, \dots, l_{k-1}) = \begin{cases} \gamma_w^{(k)} & l_1 = l_2 = \dots = l_{k-1} = 0 \\ 0 & \text{otherwise} \end{cases} \quad (12.1.34)$$

the output of the linear model is also non-Gaussian. The cumulants and the polyspectra of process $x(n)$ are

$$\kappa_x^{(k)}(l_1, l_2, \dots, l_{k-1}) = \gamma_w^{(k)} \sum_{n=0}^{\infty} h(n)h(n+l_1)\cdots h(n+l_{k-1}) \quad (12.1.35)$$

and

$$R_x^{(k)}(e^{j\omega_1}, e^{j\omega_2}, \dots, e^{j\omega_{k-1}}) = \gamma_w^{(k)} H(e^{j\omega_1})H(e^{j\omega_2})\cdots H(e^{j\omega_{k-1}})H^*(e^{j\sum_{i=1}^{k-1}\omega_i}) \quad (12.1.36)$$

respectively. The cases for $k = 3, 4$ follow easily from Equations (12.1.26), (12.1.28) to (12.1.30), and (12.1.34). A general derivation is discussed in Problem 12.2.

Setting $k = 3$ into (12.1.36), we obtain

$$\angle R_x^{(3)}(e^{j\omega_1}, e^{j\omega_2}) = \angle \gamma_w^{(3)} - \angle H(e^{j\omega_1+j\omega_2}) + \angle H(e^{j\omega_1}) + \angle H(e^{j\omega_2}) \quad (12.1.37)$$

which shows that we can use the bispectrum of the output to determine the phase response of the system if the input is a non-Gaussian IID process. From (12.1.33) we see that this is not possible using the PSD.

EXAMPLE 12.1.1. For $0 < a < 1$ and $0 < b < 1$, consider the MA(2) systems

$$\begin{aligned} H_{\min}(z) &= (1 - az^{-1})(1 - bz^{-1}) \\ H_{\max}(z) &= (1 - az)(1 - bz) \\ H_{\text{mix}}(z) &= (1 - az)(1 - bz^{-1}) \end{aligned}$$

which obviously are minimum-phase, maximum-phase, and mixed-phase, respectively. All these systems have the same output complex PSD

$$R_x(z) = \sigma_w^2 H_{\min}(z)H_{\min}(z^{-1}) = \sigma_w^2 H_{\max}(z)H_{\max}(z^{-1}) = \sigma_w^2 H_{\text{mix}}(z)H_{\text{mix}}(z^{-1})$$

and hence the same autocorrelation. As a result, we cannot correctly identify the phase response of an MA(2) model using the PSD (or equivalently the autocorrelation) of the output signal. However, we can correctly identify the phase by using the bispectrum. The output third-order cumulant $\kappa_x^{(3)}(l_1, l_2)$ for the above MA(2) models can be computed by using either the complex bispectrum (the z transform of the third-order cumulant)

$$R_x^{(3)}(z_1, z_2) = \gamma_w^{(3)} H(z_1)H(z_2)H(-z_1^{-1}z_2^{-1})$$

$$\text{or the formula } \kappa_x^{(3)}(l_1, l_2) = \gamma_w^{(3)} \sum_{n=0}^2 h(n)h(n+l_1)h(n+l_2) \quad (12.1.38)$$

for all values of l_1, l_2 that lead to overlapping terms in the summation. The results are summarized in Table 12.1. The values shown are for the principal region (see Figure 12.1); the remaining ones are computed using the symmetry relations (12.1.20). Using the formula

$$R_x^{(3)}(e^{j\omega_1}, e^{j\omega_2}) = \gamma_w^{(3)} H(e^{j\omega_1})H(e^{j\omega_2})H^*(e^{j(\omega_1+\omega_2)}) \quad (12.1.39)$$

we can numerically compute the bispectrum, using the DFT (see Problem 12.4). The results are plotted in Figure 12.2. We see that the cumulants and bispectra of the three systems are different. Hence, the third-order moments can be used to identify both the magnitude and the phase response of the MA(2) model.

TABLE 12.1
Minimum-, maximum-, and mixed-phase MA(2) systems with the same autocorrelation (or PSD) but with different third-order moments ($0 < a < 1, 0 < b < 1$) (Nikias and Raghuveer 1987).

Cumulants	Minimum-phase MA(2)	Maximum-phase MA(2)	Mixed-phase MA(2)
$\kappa_x^{(3)}(0, 0)$	$1 - (a + b)^3 + a^3b^3$	$1 - (a + b)^3 + a^3b^3$	$(1 + ab)^3 - a^3 - b^3$
$\kappa_x^{(3)}(1, 1)$	$(a + b)^2 - (a + b)a^2b^2$	$-(a + b) + ab(a + b)^2$	$-a(1 + ab)^2 + (1 + ab)b^2$
$\kappa_x^{(3)}(2, 2)$	a^2b^2	ab	$-ab^2$
$\kappa_x^{(3)}(1, 0)$	$-(a + b) + ab(a + b)^2$	$(a + b)^2 - (a + b)a^2b^2$	$a^2(1 + ab) - (1 + ab)^2b$
$\kappa_x^{(3)}(2, 0)$	ab	a^2b^2	$-a^2b$
$\kappa_x^{(3)}(2, 1)$	$-(a + b)ab$	$-(a + b)ab$	$ab(1 + ab)$
$r_x(0)$	$1 + a^2b^2 + (a + b)^2$	$1 + a^2b^2 + (a + b)^2$	$1 + a^2b^2 + (a + b)^2$
$r_x(1)$	$-(a + b)(1 + ab)$	$-(a + b)(1 + ab)$	$-(a + b)(1 + ab)$
$r_x(2)$	ab	ab	ab

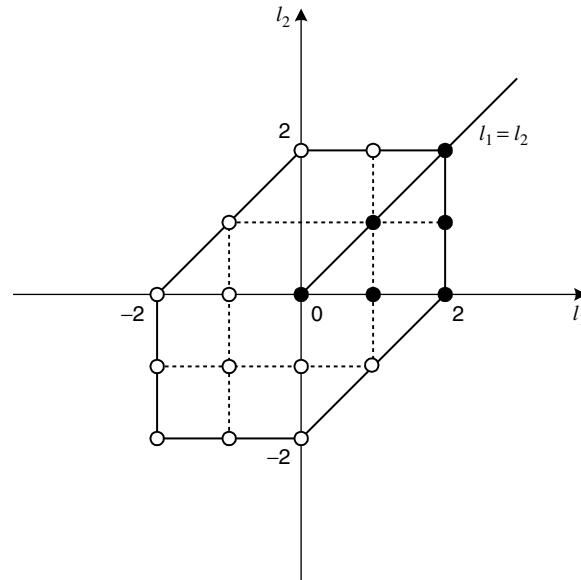
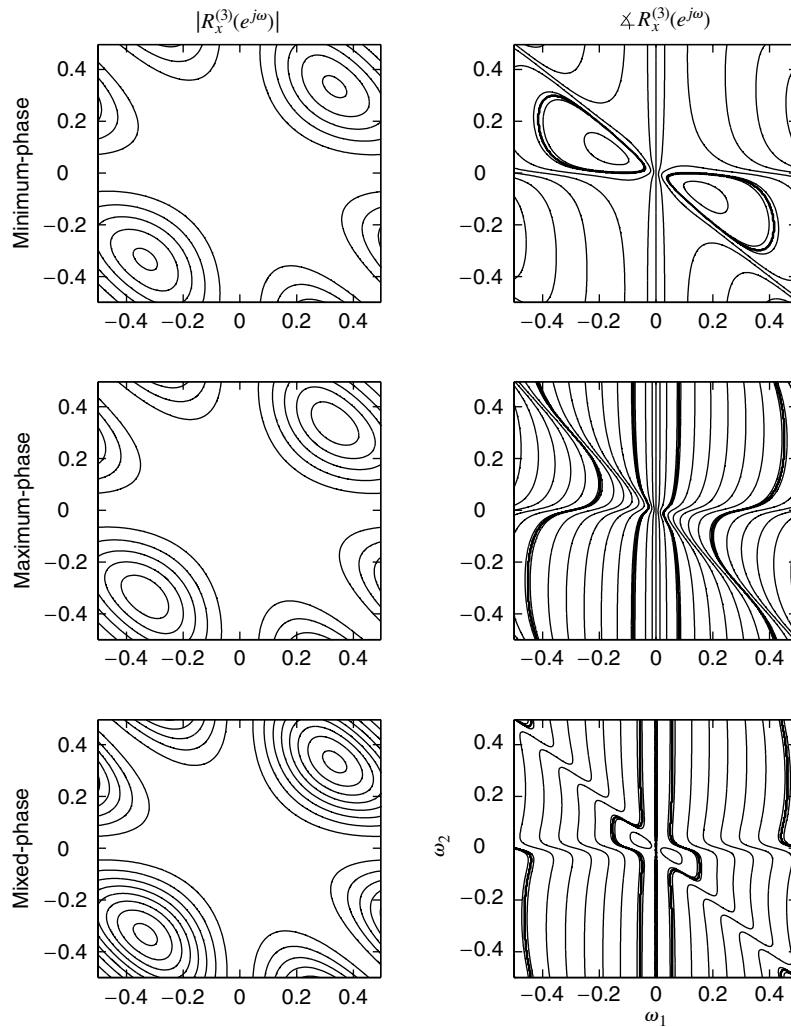


FIGURE 12.1
Region of support for the third-order cumulant of the MA(2) model. The solid circles indicate the primary samples, which can be utilized to determine the remaining samples using symmetry relations.

From the previous example and the general discussion of higher-order moments and their transformation by linear systems, we conclude that HOS can be useful when we deal with non-Gaussian signals or Gaussian signals that have passed through nonlinear

**FIGURE 12.2**

Bispectrum magnitude and phase for minimum-, maximum-, and mixed-phase MA(2) models with $a = 0.5$ and $b = 0.9$.

systems. More specifically, the use of HOS is beneficial in the following cases: suppression of additive Gaussian colored noise, identification of the phase response of a system using output data only, and characterization of non-Gaussian processes or nonlinear systems. More details and applications are discussed in Nikias and Mendel (1993). However, note that the application of HOS-based methods to real-world problems is very difficult because (1) the computation of reliable estimates of higher-order moments requires a large amount of data and (2) the assessment and interpretation of the results require a solid statistical background and extensive practical experience.

12.2 BLIND DECONVOLUTION

In Section 6.7, we discussed optimum inverse filtering and deconvolution using the minimum mean square error (MMSE) criterion under the assumption that all required statistical moments are known. In the case of *blind deconvolution* (see Figure 6.23), the goal is to retrieve the input of a system $G(z)$ by using only the output signal and possibly some

statistical information about the input. The most critical requirement is that the input signal $w(n)$ be IID, which is a reasonable assumption for many applications of practical interest. In this case, we have

$$R_x(e^{j\omega}) = \sigma_w^2 |G(e^{j\omega})|^2 \quad (12.2.1)$$

which can be used to determine, at least in principle, the magnitude response $|G(e^{j\omega})|$ from the output PSD $R_x(e^{j\omega})$. In general, it is impossible to obtain the phase response of the system from $R_x(e^{j\omega})$ without additional information. For example, if we know that $G(z) = 1/A(z)$ is a minimum-phase AP(P) system, we can uniquely identify it from $r_x(l)$ or $R_x(e^{j\omega})$, using the method of linear prediction. However, if the system is not minimum-phase, the method of linear prediction will identify it as minimum-phase, leading to erroneous results.

The importance of the input probability density function in deconvolution applications is illustrated in the following figures. Figure 12.3 shows a random sequence generated by filtering white Gaussian noise with a minimum-phase system $H(z)$ and the sequences obtained by deconvolution of this sequence with the minimum-phase, maximum-phase, and

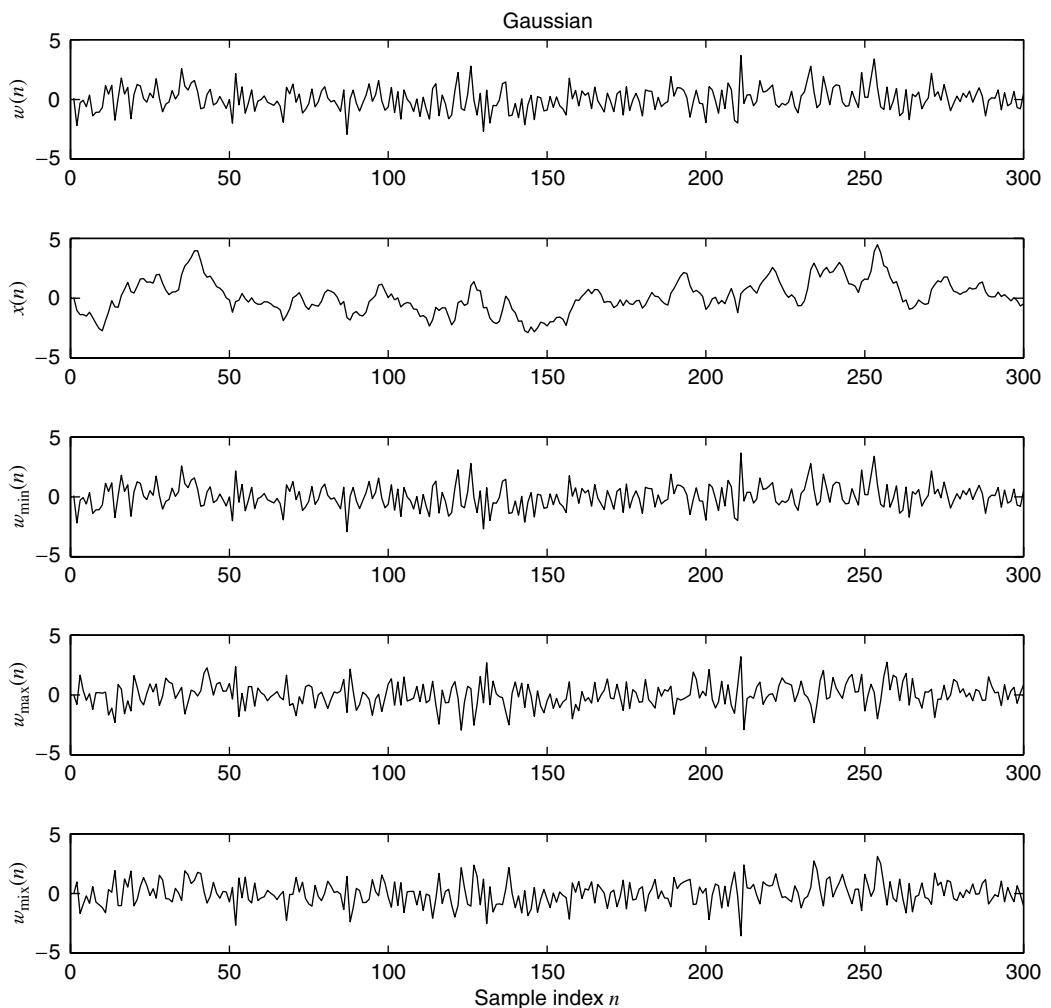


FIGURE 12.3

A minimum-phase Gaussian random sequence and its deconvolution by the corresponding minimum-, maximum-, and mixed-phase inverse systems.

mixed-phase inverse systems corresponding to $H(z)$. The three deconvolved sequences, which look visually similar, are all uncorrelated and statistically indistinguishable, because in the Gaussian case uncorrelatedness implies statistical independence. Figure 12.4 shows the results of the same experiment repeated with the same systems and a white noise sequence with an exponential probability density function. It is now clear that only the minimum-phase inverse system provides the correct answer, although all three deconvolved sequences have the same second-order statistics (Donoho 1981). More details about the generation of these figures and further discussion of their meaning are given in Problem 12.5.

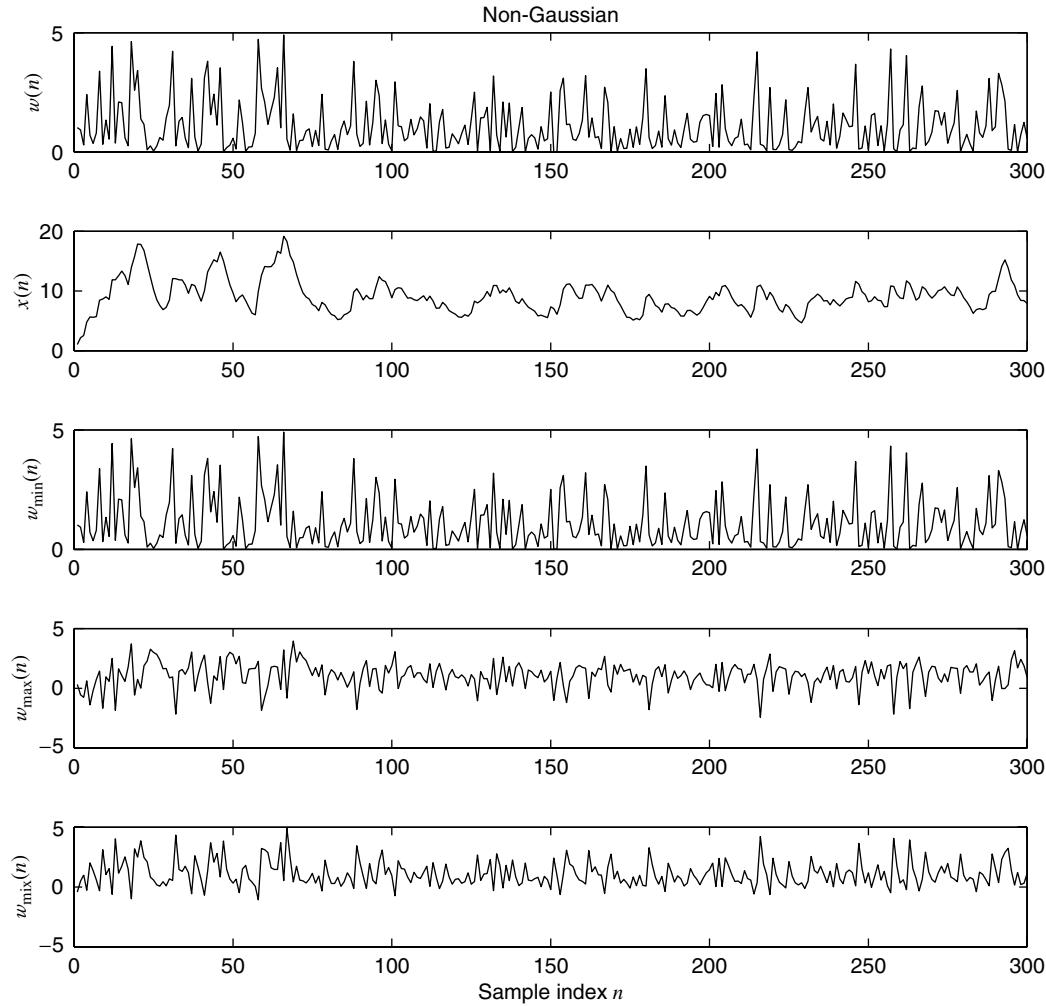


FIGURE 12.4

A minimum-phase non-Gaussian random sequence and its deconvolution by the corresponding minimum-, maximum-, and mixed-phase inverse systems.

We conclude that complete identification of $G(z)$, and therefore correct retrieval of the input signal $w(n)$, requires the identification of the phase response $\angle G(e^{j\omega})$ of the system; failure to do so may lead to erroneous results.

If the input $w(n)$ is IID and non-Gaussian, the bispectrum of the output is [see (12.1.36)]

$$R_x^{(3)}(e^{j\omega_1}, e^{j\omega_2}) = \kappa_w^{(3)} G(e^{j\omega_1})G(e^{j\omega_2})G^*(e^{j\omega_1+j\omega_2}) \quad (12.2.2)$$

and it can be used to determine both the magnitude and phase response of $G(e^{j\omega})$ from the

magnitude and argument of the bispectrum. If the bispectrum is identically zero, we can use some nonzero higher-order spectrum. Therefore, HOS can be used in many ways to obtain unbiased estimates of $\angle G(e^{j\omega})$ provided that the polyspectra are not all identically zero, or equivalently the input probability density function (pdf) is non-Gaussian (Matsuoka and Ulrych 1984; Mendel 1991; Nikias and Mendel 1993). We emphasize that, in practice, polyspectra estimators have high variance, and therefore reliable phase estimation requires very long data records. In conclusion, *blind deconvolution is always possible provided a stable inverse $1/G(z)$ exists and $w(n)$ is non-Gaussian*. If $w(n)$ is Gaussian, we cannot correctly identify the phase response of the inverse system using only the second-order moments of the output signal.

As we have already mentioned, MMSE linear prediction solves the blind deconvolution problem for minimum-phase systems with Gaussian inputs using the autocorrelation of the output signal. In essence, the inverse system retrieves the input by restoring its flat PSD, which has been colored by the system $G(z)$. This suggests the following question: *Is it possible to uniquely determine the inverse system $h(n)$ by restoring some property of the input signal (besides spectral flatness) that has been distorted by the system $G(z)$?* To address this question, let us consider the effects of an LTI system upon the probability density function of the input signal. We recall that

- If the input pdf is Gaussian, then the output pdf is Gaussian. In general, if the input pdf is stable, then the output pdf is also stable. This follows from the fact that only stable random variables are invariant under linear transformations (see Section 3.2.4). However, we limit our discussion to Gaussian signals because they have finite variance.
- If the input pdf is non-Gaussian, then the output pdf tends to Gaussian as a result of the central limit theorem (see Section 3.3.7). The “Gaussianization” capability of the system depends on the length and amplitude of its impulse response.[†] This is illustrated in Example 3.2.4, which shows that the sum of uniform random variables becomes “more Gaussian” as their number increases.

We see that filtering of a non-Gaussian IID sequence increases its Gaussianity. The only system that does *not* alter a non-Gaussian input pdf has impulse response with one nonzero sample, that is, $b_0\delta(n - n_0)$. In any other case, the input and output distributions are different, except if the input is Gaussian. A strict proof is provided by the following theorem (Kagan et al. 1973).

THEOREM 12.1. Consider a random variable x defined by the linear combination of IID random variables w_k

$$x = \sum_k c_k w_k \quad (12.2.3)$$

with coefficients such that $\sum_k |c_k|^2 < \infty$. The random variable x is Gaussian if and only if (a) x has finite variance, (b) $x \stackrel{d}{=} w_k$ for all k , (c) at least two coefficients c_k are not zero.

If we define the overall system (see Figure 12.5)

$$c(n) = g(n) * h(n) \quad (12.2.4)$$

the signals $y(n)$ and $w(n)$ can have the same non-Gaussian distribution if and only if $c(n)$ has only one nonzero coefficient. Hence, if we know the input pdf, we can determine the inverse system $h(n)$ by restoring the pdf of $y(n)$ to match the pdf of the input $w(n)$. However, it turns out that instead of restoring the pdf, that is, all moments (Benveniste et al. 1980), we only need to restore the moments up to order 4 (Shalvi and Weinstein 1990). This is shown in the following theorem.

[†]In many practical applications (e.g., seismology), the underlying data are non-Gaussian; however, unavoidable filtering operations (e.g., recording instruments) tend to “Gaussianize” their distribution. As a result, many times the non-Gaussianity of the data becomes apparent after proper deconvolution (Donoho 1981).

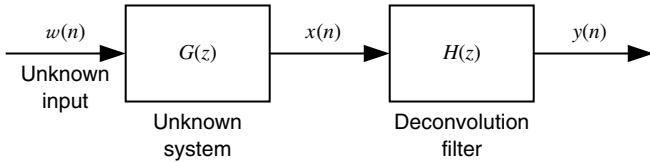


FIGURE 12.5
 Basic blind deconvolution model.

THEOREM 12.2. Consider a stable LTI system

$$y(n) = \sum_k c(k)w(n-k) \quad (12.2.5)$$

with an IID input $w(n)$ that has finite moments up to order 4. Then we have

$$E\{|y(n)|^2\} = E\{|w(n)|^2\} \sum_k |c(k)|^2 \quad (12.2.6)$$

$$E\{y^2(n)\} = E\{w^2(n)\} \sum_k c^2(k) \quad (12.2.7)$$

and

$$\kappa_y^{(4)} = \kappa_w^{(4)} \sum_k |c(k)|^4 \quad (12.2.8)$$

where $\kappa_y^{(4)} = E\{|y(n)|^4\} - 2E^2\{|y(n)|^2\} - |E\{y^2(n)\}|^2$ (12.2.9)

is the fourth-order cumulant of $y(n)$ and $\kappa_w^{(4)}$ is the fourth-order cumulant of $w(n)$.

Proof. Relations (12.2.6) and (12.2.7) can be easily shown by using (12.2.5) and the independence assumption. To prove (12.2.8), we start with (12.2.5); then by interchanging the order between expectation and summations, we have

$$\begin{aligned} E\{|y(n)|^4\} &= E \left\{ \left| \sum_k c(k)w(n-k) \right|^4 \right\} \\ &= \sum_{k_1} \sum_{k_2} \sum_{k_3} \sum_{k_4} c(k_1)c^*(k_2)c(k_3)c^*(k_4) \\ &\quad \times \underbrace{E\{w(n-k_1)w^*(n-k_2)w(n-k_3)w^*(n-k_4)\}}_W \end{aligned} \quad (12.2.10)$$

where $W = \begin{cases} E\{|w(n)|^4\} & k_1 = k_2 = k_3 = k_4 \\ E^2\{|w(n)|^2\} & k_1 = k_2 \neq k_3 = k_4, k_1 = k_4 \neq k_2 = k_3 \\ |E\{w^2(n)\}|^2 & k_1 = k_3 \neq k_2 = k_4 \\ 0 & \text{otherwise} \end{cases}$ (12.2.11)

by invoking the independence assumption. If we substitute (12.2.11) into (12.2.10), we obtain

$$\begin{aligned} E\{|y(n)|^4\} &= E\{|w(n)|^4\} \sum_k |c(k)|^2 \\ &\quad + 2E^2\{|w(n)|^2\} \left[\left[\sum_k |c(k)|^2 \right]^2 - \sum_k |c(k)|^4 \right] \\ &\quad + |E\{w^2(n)\}|^2 \left[\left[\sum_k c^2(k) \right]^2 - \sum_k |c(k)|^4 \right] \end{aligned} \quad (12.2.12)$$

Finally, substituting (12.2.6) and (12.2.7) into (12.2.12) and rearranging the various terms, we obtain (12.2.8).

We now use the previous theorem to derive necessary and sufficient conditions for blind deconvolution (Shalvi and Weinstein 1990).

THEOREM 12.3. Consider the blind deconvolution model shown in Figure 12.5 where $c(n) = g(n) * h(n)$. If $E\{|y(n)|^2\} = E\{|w(n)|^2\}$, then

1. $|\kappa_y^{(4)}| \leq |\kappa_w^{(4)}|$, that is, the kurtosis of the output is less than or equal to the kurtosis of the input.
2. $|\kappa_y^{(4)}| = |\kappa_w^{(4)}|$ if and only if $c(n) = e^{j\theta}\delta(n - n_0)$. Hence, if the kurtosis of the output is equal to the kurtosis of the input, the inverse system is given by $H(z) = e^{j\theta}z^{-n_0}/G(z)$.

Proof. The proof can be easily obtained by using the inequality

$$\sum_k |c(k)|^4 \leq \left[\sum_k |c(k)|^2 \right]^2 \quad (12.2.13)$$

where equality holds if and only if $c(k)$ has at most one nonzero component. The condition $E\{|y(n)|^2\} = E\{|w(n)|^2\}$ in conjunction with (12.2.6) implies that $\sum_k |c(k)|^2 = 1$. Therefore, $\sum_k |c(k)|^4 \leq 1$ and $|\kappa_y^{(4)}| \leq |\kappa_w^{(4)}|$ due to (12.2.8). Clearly, if $\sum_k |c(k)|^2 = 1$, we can have $\sum_k |c(k)|^4 = 1$ if and only if $c(n) = e^{j\theta}\delta(n - n_0)$.

This theorem shows that a necessary and sufficient condition for the correct recovery of the inverse system $h(n)$, that is, for successful blind deconvolution, is that $E\{|y(n)|^2\} = E\{|w(n)|^2\}$ and $|\kappa_y^{(4)}| = |\kappa_w^{(4)}|$. Therefore, we can determine $h(n)$ by solving the following constrained optimization problem:

$$\max_{h(n)} |\kappa_y^{(4)}| \quad \text{subject to} \quad E\{|y(n)|^2\} = E\{|w(n)|^2\} \quad (12.2.14)$$

It has been shown that for FIR inverse filters of sufficient length, $\kappa_y^{(4)}$ has no spurious local maxima over $E\{|y(n)|^2\} = E\{|w(n)|^2\}$, and therefore gradient search algorithms converge to the correct solution regardless of initialization (Shalvi and Weinstein 1990). We should stress that the IID property of the input $w(n)$ is a key requirement for blind deconvolution methods to work.

By using the normalized cumulants $\tilde{\kappa}_y^{(4)} = \kappa_y^{(4)} / \sigma_y^4$ it has been shown for real signals (Donoho 1981) that

$$\tilde{\kappa}_y^{(4)} = \tilde{\kappa}_w^{(4)} \frac{\sum_k |c(k)|^4}{\left[\sum_k |c(k)|^2 \right]^2} \quad (12.2.15)$$

which implies that $|\kappa_y^{(4)}| \leq |\kappa_w^{(4)}|$, a result attributed to Granger (1976). Furthermore, Donoho (1981) showed that if $\tilde{\kappa}_w^{(4)} \neq 0$, then maximization of $|\kappa_y^{(4)}|$ provides a solution to the blind deconvolution problem (Tugnait 1992). An elaborate discussion of cumulant maximization criteria and algorithms for blind deconvolution is given in Cadzow (1996). A review of various approaches for blind system identification and deconvolution is given in Abed-Meraim et al. (1997). In the next section, we apply these results to the design of adaptive filters for blind equalization.

12.3 UNSUPERVISED ADAPTIVE FILTERS—BLIND EQUALIZERS

All the adaptive filters we have discussed so far require the availability of a desired response signal that is used to “supervise” their operation. What we mean by that is that, at each time instant, the adaptive filter compares its output with the desired response and uses this

information to improve its performance. In this sense, the desired response serves as a training signal that provides the feedback needed by the filter to improve its performance. However, as we discussed in Sections 1.4.1 and 10.1, there are applications such as blind equalization and blind deconvolution in which the availability of a desired response signal is either impossible or impractical. In this section we discuss adaptive filters that circumvent this problem; that is, they can operate *without* a desired response signal. These filters are called *unsupervised adaptive filters* to signify the fact that they operate without “supervision,” that is, without a desired response signal. Clearly, unsupervised adaptive filters need additional information to make up for the lack of a desired response signal. This information depends on the particular application and has a big influence on the design and performance of the adaptive algorithm. The most widely used unsupervised adaptive filters are application-specific and operate by exploiting (1) the higher-order statistics, (2) the cyclostationary statistics, or (3) some invariant property of the input signal. Most unsupervised adaptive filtering algorithms have been developed in the context of blind equalization, which provides the most important practical application of these filters.

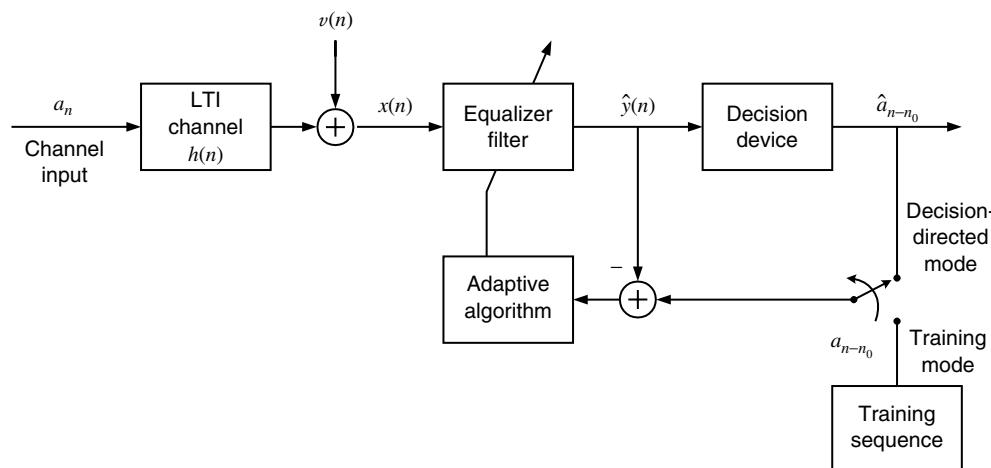


FIGURE 12.6

Conventional channel equalizer with training and decision-directed modes of operation.

12.3.1 Blind Equalization

Figure 12.6 shows the traditional approach to adaptive equalization.[†] When the adaptive equalizer starts its operation, the transmitter sends a known training sequence over the unknown channel. Since the training sequence can be used as a desired response signal, we can adjust the equalizer's coefficients by using the standard LMS or RLS algorithms. The LMS equalization algorithm with a training sequence is

$$\mathbf{c}(n) = \mathbf{c}(n-1) + 2\mu \mathbf{x}(n)e^*(n) \quad (12.3.1)$$

$$\text{where } e(n) = a_{n-n_0} - \hat{y}(n) = a_{n-n_0} - \mathbf{c}^H(n-1)\mathbf{x}(n) \quad (12.3.2)$$

is the a priori error. If, at the end of the training period, the MSE $E\{|e(n)|^2\}$ is so small that $\hat{y}(n) \simeq a_{n-n_0}$, then we can replace a_{n-n_0} by the decision $\hat{a}_{n-n_0} \triangleq Q[\hat{y}(n)]$ and switch the equalizer to decision-directed mode. The resulting algorithm is

$$\mathbf{c}(n) = \mathbf{c}(n-1) + 2\mu \mathbf{x}(n)\{Q[\hat{y}(n)] - \hat{y}(n)\}^* \quad (12.3.3)$$

and its performance depends on how close $\mathbf{c}(n)$ is to the optimum setting \mathbf{c}_o according to the

[†]This approach has been discussed in Sections 6.8 and 10.4.4.

MSE or the zero-forcing criterion. If $\mathbf{c}(0)$ is close to \mathbf{c}_o , the intersymbol interference (ISI) is significantly reduced (i.e., the eye is open), the decision device makes correct decisions with low probability of error, and the algorithm is likely to converge to \mathbf{c}_o . However, if $\mathbf{c}(0)$ is not close to \mathbf{c}_o , that is, when the eye is closed (which is when we need an equalizer), then the error surface can be multimodal and the decision-directed equalizer fails to converge or converges to a local minimum (Mazo 1980).

The training session should be repeated each time the channel response changes or after system breakdowns, which results in a reduction of the data throughput. However, there are digital communications applications in which the start-up and retraining of the adaptive equalizer have to be accomplished *without* a training sequence. Adaptive equalizers that operate without the aid of a training signal are known as *blind equalizers*, although the term *unsupervised* would be more appropriate. The need for blind equalization is enormous in digital point-to-multipoint and broadcast networks, such as high-definition and cable television. In all these applications, the transmitter should be able to send its content unaffected by the joining or withdrawal of client receivers or their need for training data (Treichler et al. 1998).

Clearly, blind equalization is a special case of blind deconvolution with input from a finite alphabet. When we deal with blind equalization, we should recall the following facts:

1. The second-order statistics of the output provide information about the magnitude response of an LTI channel. Therefore, mixed-phase channels *cannot* be identified using second-order statistics only.
2. Mixed-phase LTI channels with IID Gaussian inputs cannot be identified from their output because all statistical information is contained in the second-order moments.
3. The inverse of a mixed-phase LTI channel is IIR and unstable. Hence, only an FIR causal approximation can be used for its equalization.
4. Channels with zeros on the unit circle cannot be equalized by using zero-forcing equalizers (Section 6.8).
5. Since $|H(e^{j\omega})|^2 = |H(e^{j\omega})e^{j\theta}|^2$ and for perfect equalization $H(z)C(z) = b_0z^{-n_0}$, $b_0 \neq 0$, the channel can be identified up to a rotational factor and a constant time shift.
6. The structure of the finite symbol alphabet improves the detection process, which can be thought as an unsupervised pattern classification problem (Fukunaga 1990).

All equalizers (blind or not blind) use the second-order statistics (autocorrelation or power spectrum) of the channel output, to obtain information about the channel's magnitude response. However, blind equalizers need additional information to determine the phase response of the channel and to compensate for the absence of the desired response sequence. Phase information can be obtained from the HOS or the second-order and higher-order cyclostationary moments of the channel output. The cyclostationarity property results from the modulation of the transmitted signal (Gardner 1991).

The above types of information can be exploited, either individually or in combination, to obtain various blind equalization algorithms. The available blind equalization methods can be categorized into two groups:

1. *HOS-based methods*. These can be further divided into two groups:
 - a. *Implicit HOS algorithms* implicitly explore HOS by iteratively minimizing a *non-MSE* criterion, which does not require the desired response but reflects the amount of residual ISI in the received signal.
 - b. *Explicit HOS algorithms* compute explicitly the block estimates of the power spectrum to determine the magnitude response and block estimates of the trispectrum, to determine the phase response of the channel.
2. *Cyclostationary statistics-based methods*, which exploit the second-order cyclostationary statistics of the received signal.

Since the number of samples required to estimate the m th-order moment, for a given level of bias and variance, increases almost exponentially with order m (Brillinger 1980), both implicit and explicit HOS-based methods have a slow rate of convergence. Indeed, since channel identification requires at least fourth-order moments, HOS-based algorithms require a large number, typically several thousand, of data samples (Ding 1994).

Explicit HOS methods originated in geophysics to solve blind deconvolution problems with non-Gaussian inputs (Wiggins 1978; Donoho 1981; Godfrey and Rocca 1981). A complete discussion of the application of HOS techniques to blind equalization is given in Hatzinakos and Nikias (1991). Because HOS algorithms require a large number of data samples and have high computational complexity, they are not used in practice for blind equalization applications. In contrast to symbol rate blind equalizers that require the use of HOS, the input of fractionally spaced equalizers (which is sampled higher than the symbol rate) contains additional cyclostationarity-based second-order statistics (SOS) that can be exploited to identify the channel. Since SOS requires fewer data samples for estimation, we can exploit cyclostationarity to obtain algorithms that converge faster than HOS-based algorithms. Furthermore, channel identification using cyclic SOS does not preclude inputs with Gaussian or nearly Gaussian statistics. More information about these methods can be found in Gardner (1991), Ding (1994), Tong et al. (1994a, b), and Moulines et al. (1995). We focus on implicit HOS methods because they are easy to implement and are widely used in practice.

12.3.2 Symbol Rate Blind Equalizers

The basic structure of a blind equalization system is shown in Figure 12.7. The key element is a scalar zero-memory nonlinear function $\tilde{\psi}$, which serves to generate a desired response signal $\tilde{\psi}[\hat{y}(n)]$ for the adaptive algorithm.

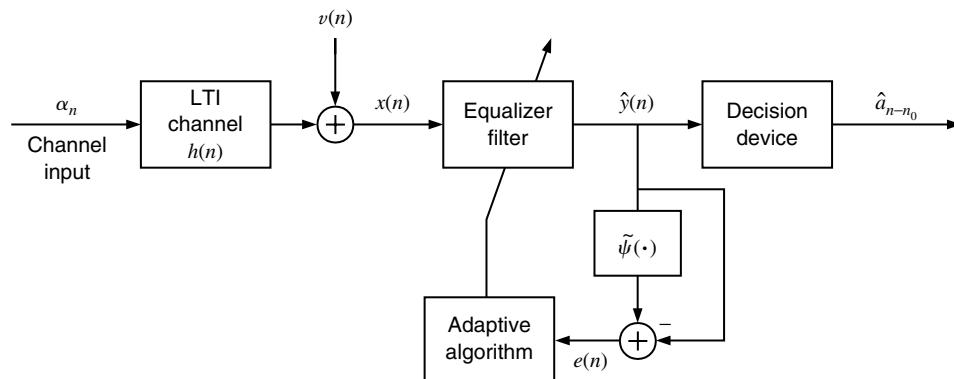


FIGURE 12.7

Basic elements of an adaptive blind equalization system.

We wish to find the function $\tilde{\psi}$ that provides a good estimate of the desired response a_n . To this end, suppose that we have a good initial guess $c(n)$ of the equalizer coefficients. Then we assume that the convolution of the channel and equalizer impulse responses can be decomposed as

$$h(n) * c(n) = \delta(n) + h_{\text{ISI}}(n) \quad (12.3.4)$$

where $h_{\text{ISI}}(n)$ is the component creating the ISI. The output of the equalizer is

$$\begin{aligned}\hat{y}(n) &= c(n) * x(n) = c(n) * [h(n) * a_n + v(n)] \\ &= a_n + h_{\text{ISI}}(n) * a_n + c(n) * v(n) \triangleq a_n + \tilde{v}(n)\end{aligned}\quad (12.3.5)$$

where $h_{\text{ISI}}(n) * a_n$ is the residual ISI and $c(n) * v(n)$ is additive noise. By invoking the central limit theorem, we can show that the convolutional noise $\tilde{v}(n)$ can be modeled as white Gaussian noise (Godfrey and Rocca 1981; Haykin 1996). Since a_n is IID and since a_n and $\tilde{v}(n)$ are statistically independent, the minimum MSE estimate $z(n)$ of a_n based on $\hat{y}(n)$ is

$$z(n) = E\{a_n | \hat{y}(n)\} \triangleq \tilde{\psi}[\hat{y}(n)] \quad (12.3.6)$$

which is a nonlinear function of $\hat{y}(n)$ because a_n has a non-Gaussian distribution. Then the a priori error is

$$e(n) = \tilde{\psi}[\hat{y}(n)] - \hat{y}(n) \quad (12.3.7)$$

where $\hat{y}(n) = \sum_{k=-L}^L c_k^*(n-1)x(n-k) \triangleq \mathbf{c}^H(n-1)\mathbf{x}(n)$ (12.3.8)

is the output of the equalizer. This leads to the following a priori stochastic gradient algorithm for blind equalization

$$\mathbf{c}(n) = \mathbf{c}(n-1) + \mu \mathbf{x}(n) e^*(n) \quad (12.3.9)$$

where μ is the adaptation step size.

Another approach used to derive (12.3.9) is to start with the cost function

$$P(n) \triangleq E\{\Psi[\hat{y}(n)]\} \quad (12.3.10)$$

where $\psi(y) \triangleq \tilde{\psi}(y) - y$ (12.3.11)

is the derivative $\psi'(y) \triangleq \Psi'(y) = \frac{\partial \Psi(y)}{\partial y}$ (12.3.12)

of a nonlinear function Ψ . The nonlinearity of Ψ creates the dependence of the cost function on the HOS of $\hat{y}(n)$ and a_n . The cost function (12.3.10) should not require the input sequence a_n ; it should reflect the amount of current ISI, and its minimum should correspond to the minimum ISI or minimum MSE condition. In contrast to the MSE criterion, which depends on the SOS and is a quadratic (convex) function of the equalizer parameters, the cost function (12.3.10) is nonconvex and may have local minima. If we compute the gradient of $P(n)$ with respect to \mathbf{c} and drop the expectation operation, we obtain the stochastic gradient algorithm (12.3.9).

Equations (12.3.8), (12.3.7), and (12.3.9) provide the general form of LMS-type blind equalization algorithms. Different choices for the nonlinear function $\tilde{\psi}$ result in various algorithms for blind equalization. Because the output $\hat{y}(n)$ is approximately a Bussgang process, these algorithms are sometimes called *Bussgang algorithms* for blind equalization (Haykin 1996). A process is called *Bussgang* (Bussgang 1952; Bellini 1986) if it satisfies the property

$$E\{\hat{y}(n)\hat{y}^*(n-l)\} = E\{\hat{y}(n)\tilde{\psi}[\hat{y}^*(n-l)]\} \quad (12.3.13)$$

that is, its autocorrelation is equal to the cross-correlation between the process and a non-linear transformation of the process.

Sato algorithm. The first blind equalizer was introduced by Sato (1975) for one-dimensional multilevel pulse amplitude modulation (PAM) signals. It uses the error function

$$\psi_1(n) = R_1 \operatorname{sgn}[\hat{y}(n)] - \hat{y}(n) = e(n) \quad (12.3.14)$$

where

$$R_1 \triangleq \frac{E\{|a_n|^2\}}{E\{|a_n|\}} \quad (12.3.15)$$

and $\text{sgn}(x)$ is the signum function. Integration of $\psi_1(n)$ gives

$$\Psi_1[\hat{y}(n)] = \frac{1}{2}[R_1 - \hat{y}(n)]^2 \quad (12.3.16)$$

whose expectation provides the cost function for the Sato algorithm. The complex version of the algorithm, used for quadrature amplitude modulation (QAM) constellations, uses the error

$$e(n) = R_1 \text{csgn}[\hat{y}(n)] - \hat{y}(n) \quad (12.3.17)$$

where

$$\text{csgn}(x) = \text{csgn}(x_r + jx_i) = \text{sgn}(x_r) + j\text{sgn}(x_i) \quad (12.3.18)$$

is the complex signum function.

Godard algorithms. The most widely used algorithms, in practical blind equalization applications, were developed by Godard (1980) for QAM signal constellations. Godard replaced the function Ψ_1 with the more general function

$$\Psi_p[\hat{y}(n)] = \frac{1}{2p}[R_p - |\hat{y}(n)|^p]^p \quad (12.3.19)$$

where p is a positive integer and R_p is the positive real constant

$$R_p \triangleq \frac{E\{|a_n|^{2p}\}}{E\{|a_n|^p\}} \quad (12.3.20)$$

which is known as the *dispersion of order p*. The family of Godard stochastic gradient algorithms is described by

$$\mathbf{c}(n) = \mathbf{c}(n-1) + \mu \mathbf{x}(n)e^*(n) \quad (12.3.21)$$

where

$$e(n) = \hat{y}(n)|\hat{y}(n)|^{p-2}[R_p - |\hat{y}(n)|^p] \quad (12.3.22)$$

is the error signal. This is an LMS-type algorithm obtained by computing the gradient of (12.3.19) and dropping the expectation operator.

Other algorithms for blind equalization include (Ding 1998) the extensions of the Sato algorithm in Benveniste et al. (1980), the stop-and-go algorithms (Picchi and Prati 1987), and the Shalvi and Weinstein algorithms (Shalvi and Weinstein 1990).

12.3.3 Constant-Modulus Algorithm

The Godard algorithm for $p = 2$ was independently introduced by Treichler and Agee (1983) with the name *constant-modulus algorithm (CMA)* and used the property restoral approach. The resulting cost function

$$P(n) = E\{[R_2 - |\hat{y}(n)|^2]^2\} \quad (12.3.23)$$

depends on the amount of ISI plus noise at the output of the equalizer. Godard (1980) has shown that the coefficient values that minimize (12.3.23) are close to the values that minimize[†] the MSE $E\{[|a_n|^2 - |\hat{y}(n)|^2]^2\}$. The criterion is independent of the carrier phase because if we replace $\hat{y}(n)$ by $\hat{y}(n)e^{j\phi}$ in (12.3.23), then $P(n)$ remains unchanged. As a result, the adaptation of the CMA can take place independently of and simultaneously with

[†] More precisely, we wish to minimize $E\{[|a_{n-n_0}|^2 - |\hat{y}(n)|^2]^2\}$ for a particular choice of the delay n_0 . As we have seen in Section 6.8, the value of n_0 has a critical effect on the performance of the equalizer.

operation of the carrier recovery system. The CMA is summarized in Table 12.2. Note that for 128-QAM, $R_2 = 110$. If we choose $R_2 \neq 110$, the CMA converges to a linearly scaled 128-QAM constellation that satisfies (12.3.23). However, choosing an unreasonable value for R_2 may cause problems when we switch to decision-directed mode (Gitlin et al. 1992).

TABLE 12.2
Summary of Godard or constant-modulus algorithm.

Operation	Equation
Equalizer	$\hat{y}(n) = \sum_{k=-L}^L c_k^*(n-1)x(n-k)$
Error	$e(n) = \hat{y}(n)[R_2 - \hat{y}(n) ^2]$
Updating	$\mathbf{c}(n) = \mathbf{c}(n-1) + \mu \mathbf{x}(n)e^*(n)$
Godard constant	$R_2 \triangleq \frac{E\{ a(n) ^4\}}{E\{ a(n) ^2\}}$

Because of its practical success and its computational simplicity, the CMA is widely used in blind equalization and blind array signal processing systems.

The CMA in Table 12.2 performs a stochastic gradient minimization of the constant-modulus performance surface (12.3.1). In contrast to the unimodal MSE performance surface of trained equalizers, the constant-modulus performance surface of blind equalizers is multimodal. The multimodality of the error surface and the lack of a desired response signal have profound effects on the convergence properties of the CMA (Johnson et al. 1998). A detailed analysis of the local convergence of the CMA algorithm is provided in Ding et al. (1991).

1. *Initialization.* Since the CMA error surface is nonconvex, the algorithm may converge to undesirable minima, which indicates the importance of the initialization procedure. In practice, almost all blind equalizers are initialized using the *tap-centering* approach: All coefficients are set to zero except for the center (reference) coefficient, which is set larger than a certain constant.
2. *Convergence rate.* The trained LMS algorithm has a bounded convergence rate $(1 - 2\mu\lambda_{\max})^{-1} < \tau < (1 - 2\mu\lambda_{\min})^{-1}$, because the Hessian matrix (which determines the curvature) of the quadratic error surface is constant. Since the error surface of the constant-modulus criterion is multimodal and includes saddle points, the convergence rate of the CMA is slow at the neighborhood of saddle points and comparable to that of the trained LMS in the neighborhood of a local minimum.
3. *Excess MSE.* In the trained LMS algorithm, the excess MSE is determined by the step size, attainable MMSE, number of filter coefficients, and power of the input signal. In addition, the excess MSE of the CMA depends on the kurtosis of the source signal (Fijalkow et al. 1998).

EXAMPLE 12.3.1. To illustrate the key characteristics of the adaptive blind symbol or baud-spaced equalizer (BSE) using the CMA algorithm, we used BERGULATOR, a public-domain interactive MATLAB-5 program that allows experimentation with the constant-modulus criterion and various implementations of the CMA (Schniter 1998). The system function of the channel is $H(z) = 1 + 0.5z^{-1}$; the input is an IID sequence with four equispaced levels (PAM); the SNR = 50 dB; the equalizer has two coefficients c_0 and c_1 ; and the step size of the CMA is $\mu = 0.005$. Figure 12.8 shows contours of the constant-modulus criterion surface in the equalizer coefficient space, where the location of the MMSE is indicated by the asterisk* and the local

MSE locations by \times . Since the constant-modulus surface is multimodal, the equalizer converges at a different minimum depending on the initial starting point. This is illustrated by the two different coefficient trajectories shown in Figure 12.8, which demonstrates the importance of initialization in adaptive algorithms with nonquadratic cost functions. Figure 12.9 shows the learning curves for smoothed versions of the error and the square of the error for the trajectories in Figure 12.8.

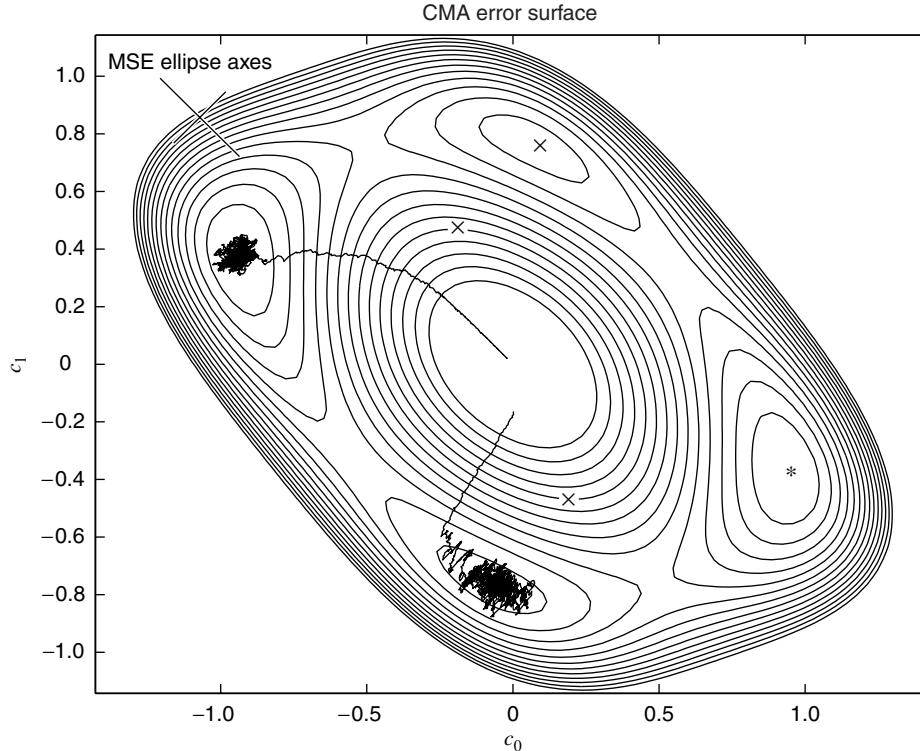


FIGURE 12.8

Contours of the constant-modulus cost function and coefficient trajectories for a blind BSE using the CMA.

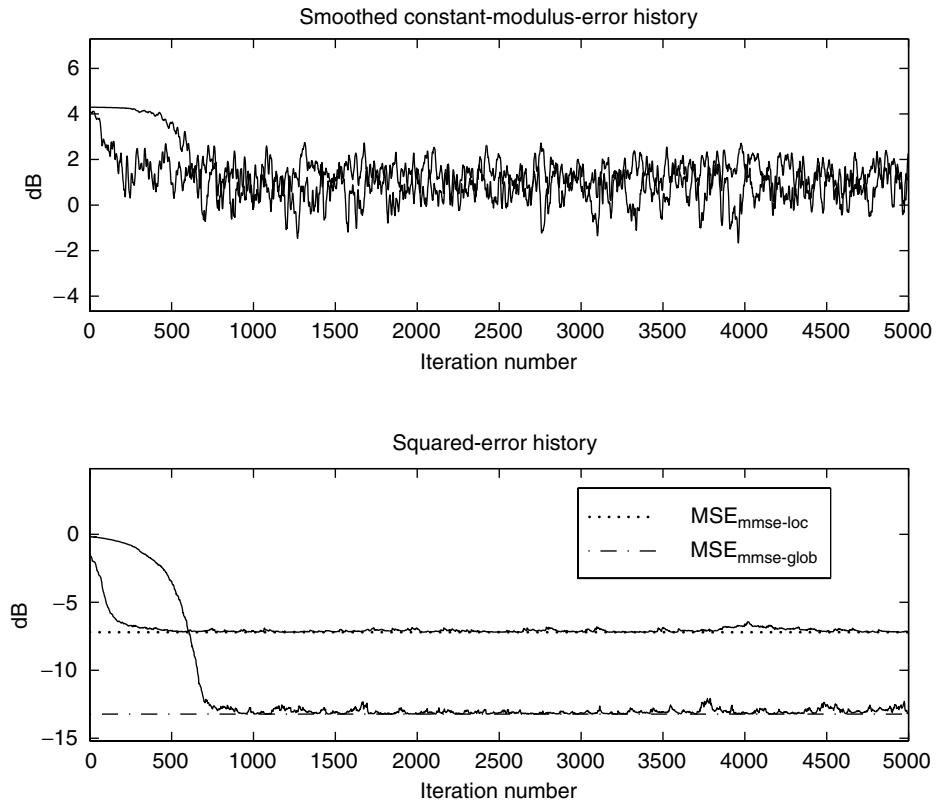
12.4 FRACTIONALLY SPACED EQUALIZERS

The input to a *fractionally spaced equalizer (FSE)* (see Figure 12.10) is obtained by sampling the channel output at a rate faster than the symbol or baud rate $\mathcal{R}_B = 1/T_B$, where T_B is the symbol duration. For simplicity and because they are extensively used in practice, we focus on $T_B/2$ spaced FSE. However, all results can be extended to any rational fraction of T_B . One of the most attractive features of an FSE is that under ideal conditions, a finite impulse response (FIR) FSE can perfectly equalize an FIR channel (Johnson et al. 1998). Referring[†] to Figure 6.26 (a), we see that the continuous-time output of the channel is

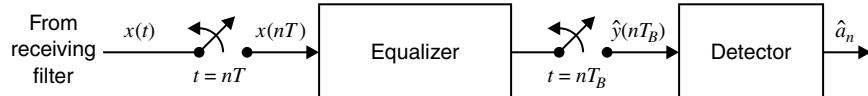
$$\tilde{x}(t) = \sum_{k=-\infty}^{\infty} a_k \tilde{h}_r(t - kT_B - t_0) + \tilde{v}(t) \quad (12.4.1)$$

where $\tilde{h}_r(t)$ is the continuous-time impulse response and where we have incorporated the channel delay t_0 in $\tilde{h}_r(t)$. The discrete-time model of Figure 6.30 is no longer valid since

[†]The material in this section requires familiarity with the notation and concepts developed in Section 6.8.

**FIGURE 12.9**

Learning curves for a blind BSE using the CMA for the two coefficient trajectories in Figure 12.8.

**FIGURE 12.10**

Block diagram of data communications receiver with a fractionally spaced equalizer.

$T = T_B/2$. However, if we extend the development leading to Figure 6.30 for $t = nT_B/2$, we obtain the discrete-time signal

$$x(n) = \sum_{k=0}^{\infty} a_k h_r(n-2k) + v(n) \quad (12.4.2)$$

where $h_r(n)$ is the equivalent discrete-time impulse response and $v(n)$ is the equivalent white Gaussian noise (WGN). The output of an FIR $T_B/2$ spaced FSE is

$$y_f(n) = \sum_{k=0}^{2M-1} c_k x(n-k) \quad (12.4.3)$$

where we have chosen the even-order $2M$ for simplicity. If we decimate the output of the

equalizer by retaining the *odd*-indexed samples $2n + 1$, we have

$$\begin{aligned}\hat{y}(n) &\triangleq y_f(2n + 1) = \sum_{k=0}^{2M-1} c_k x(2n + 1 - k) \\ &= \sum_{k=0}^{M-1} c_{2k} x(2n + 1 - 2k) + \sum_{k=0}^{M-1} c_{2k+1} x(2n - 2k)\end{aligned}$$

or

$$\hat{y}(n) = \sum_{k=0}^{M-1} c_k^e x^e(n - k) + \sum_{k=0}^{M-1} c_k^o x^o(n - k) \quad (12.4.4)$$

$$\text{where } c_k^e = c_{2k}, \quad c_k^o = c_{2k+1}, \quad x^e(n) = x(2n), \quad x^o(n) = x(2n + 1) \quad (12.4.5)$$

are known as the *even* (e) and *odd* (o) parts of the equalizer impulse responses and the received sequences, respectively. Equation (12.4.4) expresses the decimated symbol rate output of the equalizer as the sum of two symbol rate convolutions involving the even and odd two-channel subequalizers.

If we define the even and odd symbol rate subchannels

$$h^e(n) = h_r(2n) \quad \text{and} \quad h^o(n) = h_r(2n + 1) \quad (12.4.6)$$

we can show that the combined impulse response $\tilde{h}(n)$ from the transmitted symbols a_n to the symbol rate output $\hat{y}(n)$ of the FSE is given by

$$\tilde{h}(n) = c_n^e * h^o(n) + c_n^o * h^e(n) \quad (12.4.7)$$

in the time domain or

$$\tilde{H}(z) = C^e(z)H^o(z) + C^o(z)H^e(z) \quad (12.4.8)$$

in the z domain. The resulting two-channel system model is illustrated in Figure 12.11.

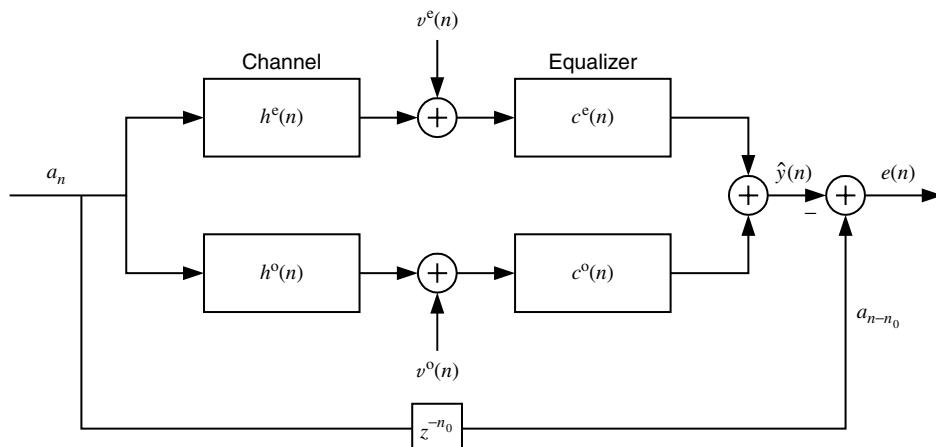


FIGURE 12.11

Two-channel representation of a $T_b/2$ spaced equalizer.

12.4.1 Zero-Forcing Fractionally Spaced Equalizers

If we define the $(M + L - 1) \times M$ even subchannel matrix (we assume a $2L$ FIR channel)

$$\mathbf{H}_e \triangleq \begin{bmatrix} h^e(0) & 0 & \cdots & 0 \\ h^e(1) & h^e(0) & \cdots & \vdots \\ \vdots & h^e(1) & \ddots & 0 \\ h^e(L-1) & \vdots & \ddots & h^e(0) \\ 0 & h^e(L-1) & \ddots & h^e(1) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & h^e(L-1) \end{bmatrix} \quad (12.4.9)$$

the even subequalizer vector

$$\mathbf{c}_e \triangleq [c_0^e \ c_1^e \ \cdots \ c_{M-1}^e]^T \quad (12.4.10)$$

and their counterparts \mathbf{H}_o and \mathbf{c}_o , we can express the convolution equation (12.4.7) in matrix form as

$$\tilde{\mathbf{h}} = \mathbf{H}\mathbf{c} \quad (12.4.11)$$

$$\text{where } \mathbf{H} \triangleq [\mathbf{H}_e \ \mathbf{H}_o] \quad \mathbf{c} \triangleq \begin{bmatrix} \mathbf{c}_e \\ \mathbf{c}_o \end{bmatrix} \quad (12.4.12)$$

and $\tilde{\mathbf{h}} \triangleq [\tilde{h}(0) \ \tilde{h}(1) \ \cdots \ \tilde{h}(M+L-1)]^T$ is the symbol-spaced overall system response. In the absence of noise, the system is free of ISI if $\tilde{\mathbf{h}}$ is equal to

$$\delta_{n_0} \triangleq [0 \ \cdots \ 0 \ 1 \ 0 \ \cdots \ 0]^T \quad (12.4.13)$$

where n_0 , $0 \leq n_0 \leq M + L - 1$, indicates the location of the nonzero coefficient. Equivalently, the z domain zero-ISI condition from (12.4.8) is given by

$$z^{-n_0} = \tilde{H}(z) = C^e(z)H^o(z) + C^o(z)H^e(z) \quad (12.4.14)$$

The zero-forcing FIR equalizer is specified by the system of linear equations $\mathbf{H}\mathbf{c} = \delta_{n_0}$, which has a solution if \mathbf{H} is full row rank. This condition is also known as *strong perfect equalization*, and it holds if the number of columns is equal to or larger than the number of rows, that is, if $2M \geq M + L - 1$ or $M \geq L - 1$. Furthermore, the $T_B/2$ spaced full-rank condition implies that the system functions $H_e(z)$ and $H_o(z)$ have no common roots. These topics are discussed in detail in Johnson et al. (1998).

The main advantage of the zero-forcing FSE over the corresponding synchronous equalizer is that, in the absence of noise, a zero-ISI elimination is possible using a finite-order FSE. In the case of the synchronous equalizer, a similar zero-ISI elimination is possible only when the equalizer is of infinite length.

12.4.2 MMSE Fractionally Spaced Equalizers

When the channel noise $v(n)$ is present, then perfect equalization, even for an FSE, is not possible. Hence, the emphasis shifts to the best possible compromise between ISI and noise amplification (which is present in a zero-forcing equalizer) in a minimum MSE sense. This is obtained by minimizing the mean square value of the data symbol error

$$e(n) \triangleq \hat{y}(n) - a_{n-n_0} \quad (12.4.15)$$

for a particular choice of delay n_0 . To obtain an expression for $\hat{y}(n)$ using the vector $\tilde{\mathbf{h}}$ in (12.4.11), we first define

$$\mathbf{a}_n \triangleq [a_n \ a_{n-1} \ \cdots \ a_{n-(M+L-1)}]^T \quad (12.4.16)$$

and

$$\mathbf{v}(n) = [v(n-1) \ v(n-3) \ \cdots \ v(n-2L+1) \ v(n) \ v(n-2) \ \cdots \ v(n-2L+2)]^T \quad (12.4.17)$$

where the samples of the noise sequence are arranged as odd samples followed by the even samples so as to be consistent with the definitions of \mathbf{H} and \mathbf{c} . We then substitute (12.4.2) into (12.4.4) and obtain

$$\hat{y}(n) = \mathbf{a}_n^T \mathbf{H} \mathbf{c} + \mathbf{v}^T(n) \mathbf{c} \quad (12.4.18)$$

Using δ_{n_0} in (12.4.13), we see the desired symbol a_{n-n_0} is equal to $\mathbf{a}_n^T \delta_{n_0}$. Hence from (12.4.15) and (12.4.18), the symbol error is

$$e(n) = \mathbf{a}_n^T (\mathbf{H} \mathbf{c} - \delta_{n_0}) + \mathbf{v}^T(n) \mathbf{c} \quad (12.4.19)$$

Assuming that the symbol sequence $\{a_n\}$ is IID with variance σ_a^2 and is uncorrelated with the noise sequence $v(n) \sim \text{WN}(0, \sigma_v^2)$, the mean square value of the error $e(n)$ is given by

$$\text{MSE}(\mathbf{c}, n_0) = E\{|e(n)|^2\} = \sigma_a^2 (\mathbf{H} \mathbf{c} - \delta_{n_0})^H (\mathbf{H} \mathbf{c} - \delta_{n_0}) + \sigma_v^2 \mathbf{c}^H \mathbf{c} \quad (12.4.20)$$

which is a function of two minimizing parameters \mathbf{c} and n_0 . Following our development in Section 6.2 on linear MSE estimation, the equalizer coefficient vector that minimizes (12.4.20) is given by

$$\hat{\mathbf{c}} = \left(\mathbf{H}^H \mathbf{H} + \frac{\sigma_v^2}{\sigma_a^2} \mathbf{I} \right)^{-1} \mathbf{H}^H \delta_{n_0} \quad (12.4.21)$$

which is the classical Wiener filter. Also compare (12.4.21) with the frequency-domain Wiener filter given in (6.8.29). The corresponding minimum MSE with respect to $\hat{\mathbf{c}}$ is given by

$$\min_{\hat{\mathbf{c}}} \text{MSE}(\mathbf{c}, n_0) = \text{MSE}(n_0) = \delta_{n_0}^T \left[\mathbf{I} - \mathbf{H} \left(\mathbf{H}^H \mathbf{H} + \frac{\sigma_v^2}{\sigma_a^2} \mathbf{I} \right)^{-1} \mathbf{H}^H \right] \delta_{n_0} \quad (12.4.22)$$

Finally, the optimum value of n_0 is obtained by determining the index of the minimum diagonal element of the matrix in square brackets in (12.4.22), that is,

$$\hat{n}_0 = \arg \min_{n_0} \left\{ \left[\mathbf{I} - \mathbf{H} \left(\mathbf{H}^H \mathbf{H} + \frac{\sigma_v^2}{\sigma_a^2} \mathbf{I} \right)^{-1} \mathbf{H}^H \right]_{n_0, n_0} \right\} \quad (12.4.23)$$

Once again, similar to the synchronous equalizer, the MMSE fractionally spaced equalizer is more robust to both the channel noise and the large amount of ISI. Additionally, it provides insensitivity to sampling phase and an ability to function as a matched filter in the presence of severe noise. Therefore, in practice, FSEs are preferred to synchronous equalizers.

12.4.3 Blind Fractionally Spaced Equalizers

Fractionally spaced equalizers have just about dominated practical equalization applications because they are insensitive to sampling phase, they can function as matched filters, they can compensate severe band-edge delay distortion, they provide reduced noise enhancement, and they can perfectly equalize an FIR channel under ideal conditions (Gitlin et al. 1992; Johnson et al. 1998).

The CMA for an FSE is given by

$$\hat{y}(n) = \sum_{k=0}^{M-1} c_k^e(n-1) x^o(n-k) + \sum_{k=0}^{M-1} c_k^o(n-1) x^e(n-k) \triangleq \mathbf{c}^T(n-1) \mathbf{x}(n) \quad (12.4.24)$$

$$e(n) = \hat{y}(n)[R_2 - |\hat{y}(n)|^2] \quad (12.4.25)$$

$$\mathbf{c}(n) = \mathbf{c}(n-1) + \mu \mathbf{x}(n)e^*(n) \quad (12.4.26)$$

where $\mathbf{c}(n-1)$ and $\mathbf{x}(n)$ are concatenated even and odd sample vectors. The blind FSE adaptive structure is shown in Figure 12.12. The value of R_2 depends on the input symbol constellation. This algorithm and its convergence are discussed in Johnson et al. (1998).

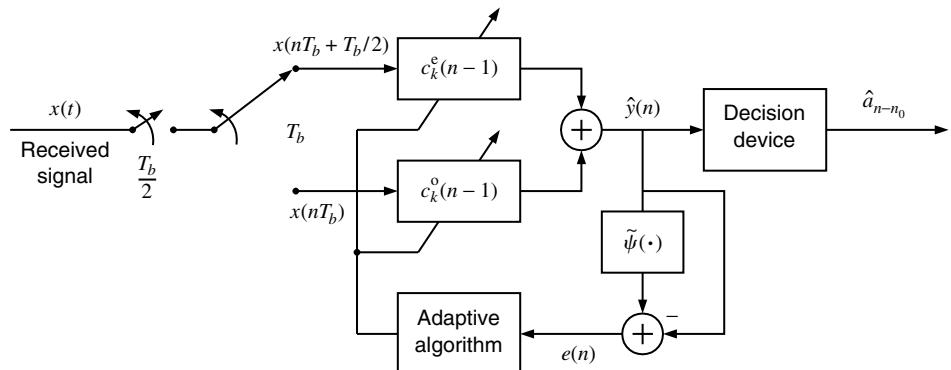


FIGURE 12.12

Basic elements of an FS adaptive blind equalization system.

EXAMPLE 12.4.1. To illustrate the superiority of the blind FSE over the blind BSE, we have used the BERGULATOR to simulate a 16-QAM data transmission system. The channel system function is $H(z) = 0.2 + 0.5z^{-1} + z^{-2} - 0.1z^{-3}$, the SNR = 20dB, and the equalizer has $M = 8$ coefficients. Figure 12.13 shows the constellation of the received signal at the input of the equalizer, where it is clear that the combined effect of ISI and noise makes detection extremely

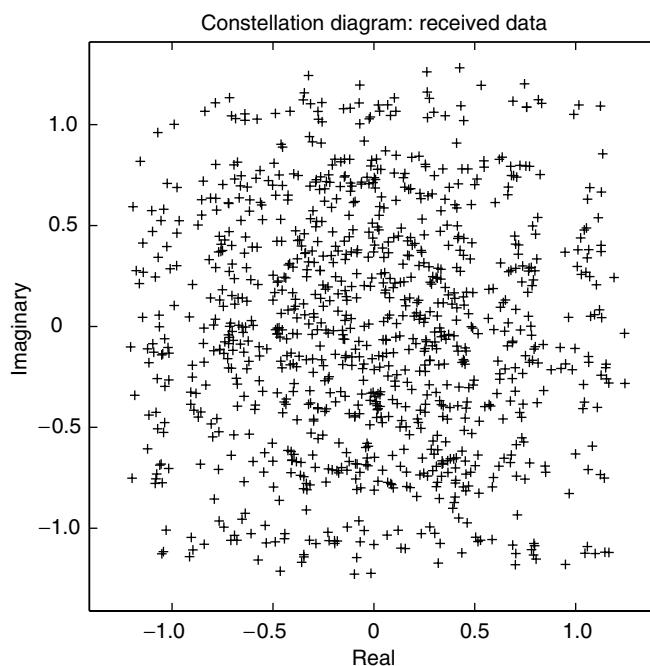
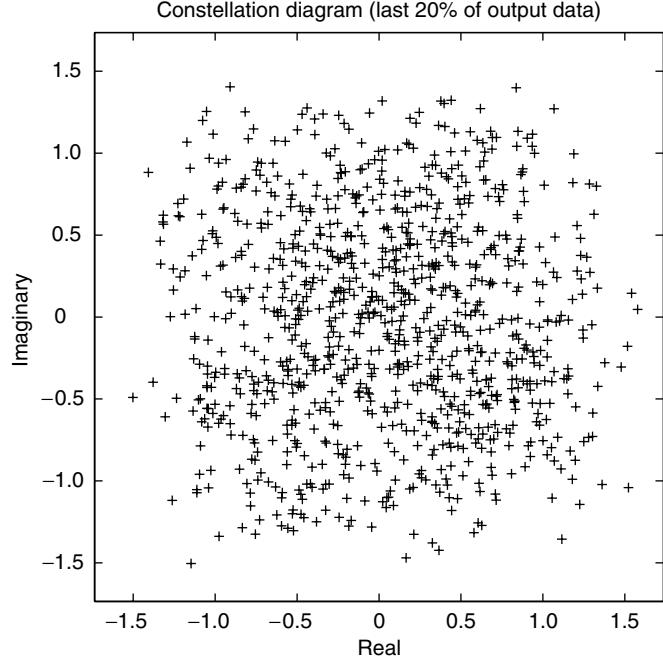
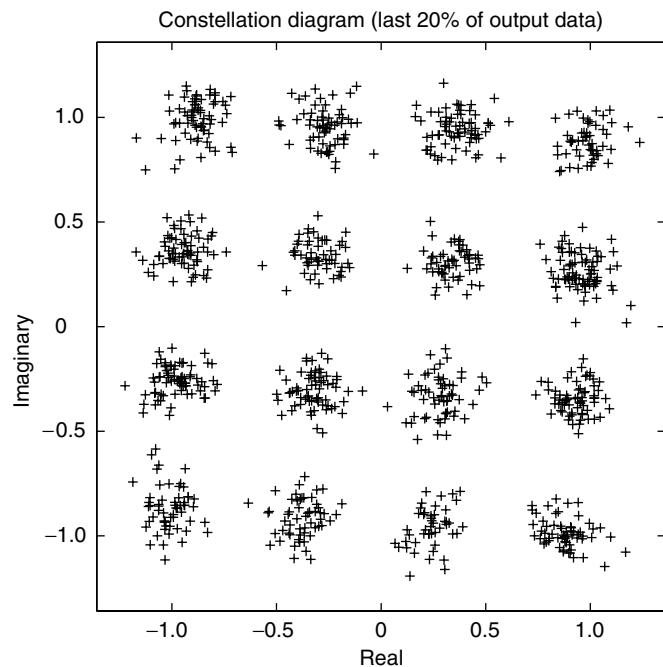


FIGURE 12.13

Constellation of the received signal symbols at the input of the equalizer.

**FIGURE 12.14**

Constellation of the equalized signal symbols at the output of the BSE equalizer.

**FIGURE 12.15**

Constellation of the equalized signal symbols at the output of the FSE.

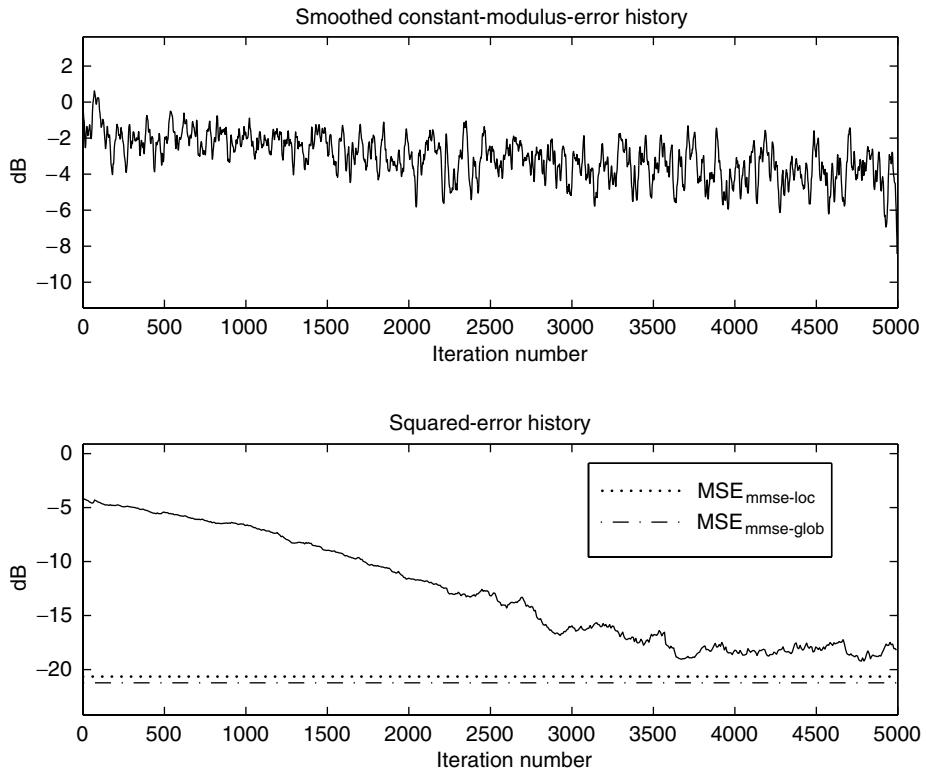


FIGURE 12.16
Learning curves for the blind FSE adaptive equalizer using the CMA.

12.5 FRACTIONAL POLE-ZERO SIGNAL MODELS

In this section we show how to obtain models with hyperbolically decaying autocorrelation, and hence long memory, by introducing fractional poles at zero frequency (fractional pole models) or nonzero frequency (harmonic fractional pole models). Cascading fractional with rational models results in mixed-memory models, known as *fractional pole-zero models*. We explore the properties of both types of models and introduce techniques for their practical implementation. Special emphasis is placed on the generation of discrete fractional pole noise, which is the random process generated by exciting a fractional pole model with white Gaussian noise. We conclude with a brief introduction to pole-zero and fractional pole models with S&S IID inputs, which result in processes with high variability and short or long memory, respectively. Fractional models are widely used in areas such as hydrology, data network traffic analysis, heart rate analysis, and economics.

12.5.1 Fractional Unit-Pole Model

The impulse response and the autocorrelation sequence of a pole-zero model decay exponentially with time, that is, they are geometrically bounded as

$$|h(n)| \leq C_h \zeta^{-n} \quad |\rho(l)| \leq C_\rho \zeta^{-l} \quad (12.5.1)$$

where $C_h, C_\rho > 0$ and $0 < \zeta < 1$ (see Chapter 4). To get a long impulse response or a long autocorrelation, at least one of the poles should move very close to the unit circle. However, in many applications we need models whose autocorrelation decays more slowly

than ζ^{-l} as $l \rightarrow \infty$, that is, models with long memory (see Section 3.2.4). In this section, we introduce a class of models, known as fractional pole models, whose autocorrelation asymptotically exhibits a geometric decay.

We have seen in Chapter 4 that by restricting some “integral” poles to being on the unit circle, we obtain models that are useful in modeling some types of nonstationary behavior. The *fractional pole model* $\text{FP}(d)$ was introduced in Granger and Joyeux (1980) and Hosking (1981), and is defined by

$$H_d(z) = \sum_{k=0}^{\infty} h_d(k) z^{-k} \triangleq \frac{1}{(1-z^{-1})^d} \quad (12.5.2)$$

where d is a nonintegral, that is, a fractional parameter. See Figure 12.17.

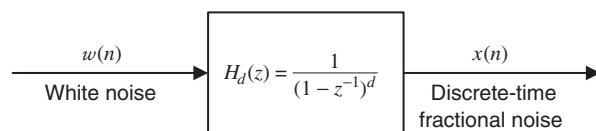


FIGURE 12.17

Block diagram representation of the discrete-time fractional noise model.

The characteristics of the model depend on the value of parameter d . Since d is not an integer, $H_d(z)$ is not a rational function. It is the nonrationality that gives this model its long-memory properties. Although we can approximate a fractional model by a PZ(P, Q) model, the orders P and Q that are needed to obtain a good approximation can be very large. This makes the estimation of pole-zero model parameters very difficult, and in practice it is better to use an $\text{FP}(d)$ model.

Impulse response. To obtain the impulse response of the fractional pole model, we expand the system function $H_d(z) = (1 - z^{-1})^{-d}$ in a power series using the binomial series expansion. This gives

$$H_d(z) = \frac{1}{(1 - z^{-1})^d} = 1 + dz^{-1} + \frac{d(d+1)}{2!} z^{-2} + \dots \quad (12.5.3)$$

The impulse response is given by

$$h_d(n) = \frac{d(d-1)\cdots(d+n-1)}{n!} = \frac{(d+n-1)!}{n!(d-1)!} = \frac{\Gamma(n+d)}{\Gamma(n+1)\Gamma(d)} \quad (12.5.4)$$

for $n \geq 0$ and $h_d(n) = 0$ for $n < 0$. $\Gamma(\cdot)$ is the gamma function defined as

$$\Gamma(\alpha) \triangleq \begin{cases} \int_0^\infty t^{\alpha-1} e^{-t} dt & \alpha > 0 \\ \infty & \alpha = 0 \\ \alpha^{-1} \Gamma(1+\alpha) & \alpha < 0 \end{cases} \quad (12.5.5)$$

with $\Gamma(\alpha+1) = \alpha\Gamma(\alpha)$ for any α and $\Gamma(n+1) = n!$ for n an integer. Note that $h_d(n)$ can be easily computed by using the recursion

$$h_d(n) = \frac{d+n-1}{n} h_d(n-1) \quad n = 1, 2, \dots \quad (12.5.6)$$

with $h_d(0) = 1$.

The system function of the inverse model is

$$H_I(z) \triangleq \sum_{n=0}^{\infty} h_I(n)z^{-n} = \frac{1}{H_d(z)} = (1 - z^{-1})^d \quad (12.5.7)$$

$$\text{Hence } h_I(n) = \frac{(-d + n - 1)!}{n!(-d - 1)!} = \frac{\Gamma(n - d)}{\Gamma(n + 1)\Gamma(-d)} = h_{-d}(n) \quad (12.5.8)$$

As expected, $h_I(n)$ is obtained from $h(n)$ by simply replacing d by $-d$.

Minimum-phase. To understand the behavior of the model, we look at the impulse response as $n \rightarrow \infty$. Using Sterling's approximation (Abramowitz and Stegun 1970)

$$\frac{(n + d - 1)!}{n!} \sim n^{d-1} \quad \text{as } n \rightarrow \infty \quad (12.5.9)$$

$$\text{we have } h(n) \sim \frac{1}{(d - 1)!} n^{d-1} \quad \text{as } n \rightarrow \infty \quad (12.5.10)$$

As a result of this geometric decay, the sum $\sum_{n=0}^{\infty} |h(n)|$ does not exist for $d > 0$. Therefore, the system is not BIBO stable. However, if $d < \frac{1}{2}$, the sum $\sum_{n=0}^{\infty} h^2(n) < \infty$, and the input $w(n)$ has finite variance, then the output of the system

$$x(n) = \sum_{k=0}^{\infty} h_d(k)w(n - k) \quad (12.5.11)$$

exists in the mean square sense. In a similar way, the output of the inverse system exists in mean square if $d > -\frac{1}{2}$. In view of this mean square convergence, we say that the fractional pole model is minimum-phase if $-\frac{1}{2} < d < \frac{1}{2}$, even if $h_d(n)$ does not converge absolutely.

Spectrum. The complex power spectrum of the model is $R_x(z) = \sigma_w^2 R_h(z)$, where

$$R_h(z) = H(z)H(z^{-1}) = \frac{1}{(1 - z^{-1})^d(1 - z)^d} \quad (12.5.12)$$

For $z = e^{j\omega}$ we obtain the power spectrum

$$R_h(e^{j\omega}) = \frac{1}{[2 \sin(\omega/2)]^{2d}} \quad -\pi < \omega \leq \pi \quad (12.5.13)$$

We see that $R_h(0) = \sum_{l=-\infty}^{\infty} r(l)$ is finite only if $d \leq 0$. Also as the frequency $\omega \rightarrow 0$, the power spectrum becomes

$$R_h(e^{j\omega}) \sim \frac{1}{\omega^{2d}} \quad \text{as } \omega \rightarrow 0 \quad (12.5.14)$$

because $\sin \theta \simeq \theta$ as $\theta \rightarrow 0$.

Autocorrelation. The autocorrelation $r_x(l) = \sigma_w^2 r_h(l)$ of the model can be found by using the inverse Fourier transform of $R_h(e^{j\omega})$, that is,

$$r_h(l) = \frac{1}{2\pi} \int_{-\pi}^{\pi} R_h(e^{j\omega}) e^{-j\omega l} d\omega = \frac{1}{2\pi} \int_0^{\pi} (\cos \omega l) \left(2 \sin \frac{\omega}{2}\right)^{-2d} d\omega \quad (12.5.15)$$

Using the identity (Gradshteyn and Ryzhik 1994)

$$\int_0^{\pi} \cos ax \sin^{v-1} x dx = \frac{\pi \cos(a\pi/2)\Gamma(v+1)2^{1-v}}{v\Gamma[(v+a+1)/2]\Gamma[(v-a+1)/2]}$$

$$\text{we obtain } r_h(l) = \frac{(-1)^l \Gamma(1-2d)}{\Gamma(1+l-d) \Gamma(1-l-d)} \quad l = 0, 1, 2, \dots \quad (12.5.16)$$

for the autocorrelation and

$$r_h(l) = \frac{r_h(l)}{r_h(0)} = \frac{\Gamma(1-d)\Gamma(l+d)}{\Gamma(d)\Gamma(l+1-d)} = \frac{(d+l-1)!}{(d-1)!(l-d)!} \quad (12.5.17)$$

for the normalized autocorrelation. Using Sterling's formula, we obtain the following asymptotic approximation

$$\rho_h(l) \sim C_d l^{2d-1} \quad \text{as } l \rightarrow \infty \quad (12.5.18)$$

which again verifies the long memory of the model. From (12.5.16) and the definition of power spectrum, we have

$$r_h(0) = \sum_{n=0}^{\infty} h^2(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 d\omega = \frac{\Gamma(1-2d)}{\Gamma^2(1-d)} \quad (12.5.19)$$

Thus, for $d < \frac{1}{2}$ we have $\int_{-\pi}^{\pi} |H(e^{j\omega})|^2 d\omega < \infty$. Hence, the inverse transform $h(n)$ converges in mean square.

Partial autocorrelation. To determine the partial autocorrelation sequence, we can show, using (12.5.17) and the algorithm of Levinson-Durbin, that the AP(m) model parameters are given by

$$a_k^{(m)} = \binom{m}{k} \frac{(k-d-1)!(m-d-k)!}{(-d-1)!(m-d)!} \quad (12.5.20)$$

Therefore, since $k_m = -a_m^{(m)}$, we have

$$k_m = \frac{d}{m-d} \quad m = 1, 2, 3, \dots \quad (12.5.21)$$

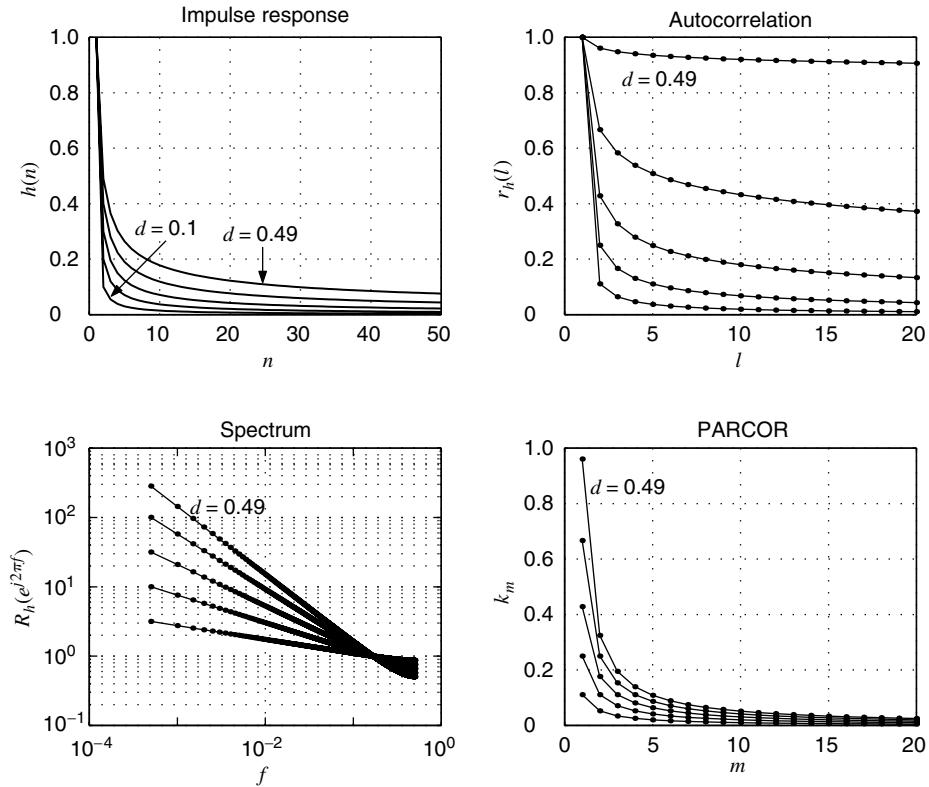
The details of the derivation are the subject of Problem 12.6.

Model memory. From Equations (12.5.10), (12.5.18), and (12.5.14) and from the long-memory definitions in Section 3.4, we conclude that the minimum-phase fractional pole model has long memory. More specifically, we arrive at the following conclusions:

- *Long memory.* For $0 < d < \frac{1}{2}$ the autocorrelation and partial correlation sequences decay monotonically and hyperbolically to zero. Although $\sum_{l=-\infty}^{\infty} |\rho(l)| = \infty$ and $R(e^{j\omega}) \rightarrow \infty$ as $\omega \rightarrow 0$, the integral (12.5.19) of $R(e^{j\omega})$ is finite. The spectrum is dominated by low-frequency components (low-pass), and the divergence at $\omega = 0$ causes the long-memory behavior. The system acts as a *fractional integrator*.
- *Short memory.* For $-\frac{1}{2} < d < 0$ the autocorrelation and partial autocorrelation sequences decay monotonically and hyperbolically to zero. In this case $\sum_{l=-\infty}^{\infty} |\rho(l)| < \infty$, $R(e^{j0}) = \sum_{l=-\infty}^{\infty} \rho(l) = 0$, and the spectrum is dominated by high-frequency components (high-pass). Sometimes we say that this model exhibits short-memory behavior. The system acts as a *fractional differentiator*.

Figures 12.18 and 12.19 show the impulse response, autocorrelation, partial autocorrelation, and power spectrum of the FP(d) model for various values of d . The short-memory and long-memory behavior of the model, as a function of parameter d , is clearly evident.

Discrete-time fractional pole noise. If we drive an FP(d) model with white Gaussian noise (see Figure 12.20), the resulting process is known as *discrete-time fractional Gaussian noise (DTFGN)*. Since the impulse response of an FP(d) system decays hyperbolically, its system function cannot be accurately approximated by a rational function. Hence, its practical implementation is not straightforward. Short sequences can be generated using the LDL^H or Cholesky decompositions of the process correlation using the (12.5.16) matrix, as explained in Section 3.5. This approach guarantees that the correlation of the generated

**FIGURE 12.18**

Impulse response, autocorrelation, partial autocorrelation, and power spectrum of the FP(d) model for $d = 0.1, 0.2, 0.3, 0.4, 0.49$.

sequence matches the theoretical autocorrelation. Since the correlation matrix is Toeplitz, its triangular factors can be computed efficiently by using the Schür algorithm (see Section 7.7). Careful inspection of Figure 12.19 shows that the impulse response of the inverse system decays extremely rapidly. Therefore, we can obtain a very accurate recursive implementation of the FP(d) system by following the approach discussed in Example 4.5.1.

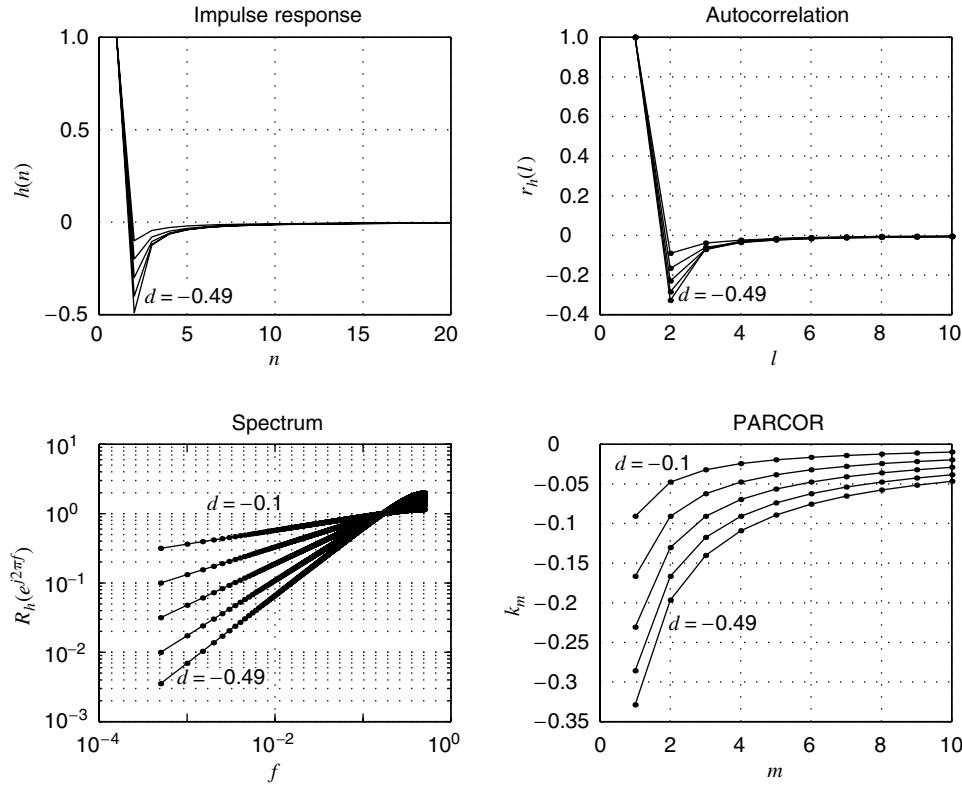
A practical algorithm for the generation of DTFGN is derived in Hosking (1984) using the following result: For any stationary process with zero mean value, the conditional mean and variance of $x(n)$ given $\{x(j)\}_0^{n-1}$ are given by

$$\mu_x(n) = E\{x(n)|x(n-1), \dots, x(0)\} = - \sum_{j=1}^n a_j^{(n)*} x(n-j) \triangleq -\mathbf{a}_n^H \mathbf{x}_n \quad (12.5.22)$$

$$\text{and} \quad v_x(n) = \text{Var}\{x(n)|x(n-1), \dots, x(0)\} = \sigma_x^2 \prod_{j=1}^n (1 - |k_j|^2) \quad (12.5.23)$$

where \mathbf{a}_n is the forward linear predictor (FLP) with lattice parameters k_j and $\sigma_x^2 = E\{|x(n)|^2\}$ (Ramsey 1974). This result implies that we can use the Levinson-Durbin algorithm to recursively determine $\mu_x(n)$ and $v_x(n)$, starting at $n = 0$ and generating $x(n) \sim \text{WGN}[\mu_x(n), v_x(n)]$ at each step. For the FP(d) model this algorithm is simplified because $k_m = d/(m-d)$ is known. The algorithm is initialized with $x(0) \sim \text{WGN}(0, \sigma_x^2)$ and continues with repeating the following recursions

$$k_n = \frac{d}{n-d} \quad (12.5.24)$$

**FIGURE 12.19**

Impulse response, autocorrelation, partial autocorrelation, and power spectrum of the $\text{FP}(d)$ model for $d = -0.1, -0.2, -0.3, -0.4, -0.49$.

$$\mathbf{a}_{n+1} = \begin{bmatrix} \mathbf{a}_n \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{J} \mathbf{a}_n^* \\ 1 \end{bmatrix} k_n \quad (12.5.25)$$

$$\mu_x(n+1) = \mathbf{a}_{n+1}^H \mathbf{x}_{n+1} \quad (12.5.26)$$

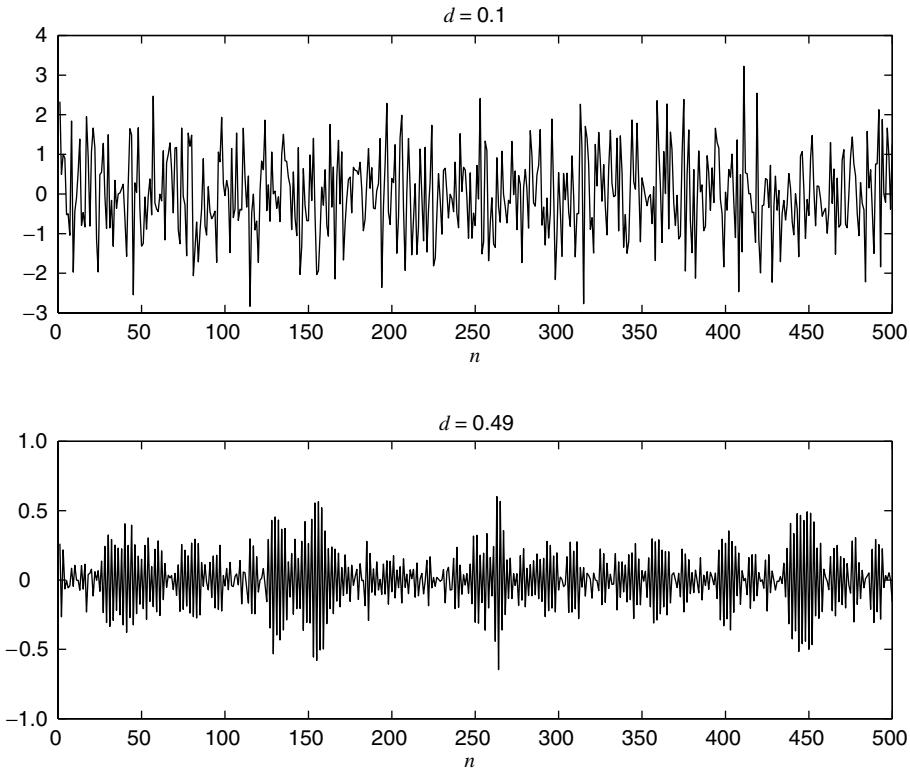
$$v_x(n+1) = v_x(n)(1 - |k_n|^2) \quad (12.5.27)$$

$$x(n+1) \sim \text{WGN}[\mu_x(n+1), v_x(n+1)] \quad (12.5.28)$$

for $n = 1, 2, \dots, N$. The algorithm is implemented by the function $\mathbf{x} = \text{dtfgn}(d, N)$. Figure 12.20 shows sample realizations of discrete fractional noise, for various values of d , generated by using the above algorithm. A simplified, numerically robust algorithm, using the lattice structure, is introduced in Problem 12.9. The estimation of long memory is discussed in Section 12.6.

12.5.2 Fractional Pole-Zero Models: $\text{FPZ}(P, d, Q)$

Since the behavior of the $\text{FP}(d)$ model is controlled by the single parameter d , it is not flexible enough to model the wide variety of short-term (small-lag) autocorrelation structures encountered in practical applications. A more powerful model capable of modeling both short-term and long-term correlation structures can be obtained by cascading a $\text{PZ}(P, Q)$ model (to handle short memory) with an $\text{FP}(d)$ (to handle long memory). This can be viewed

**FIGURE 12.20**

Sample realizations of discrete-time fractional Gaussian noise for two different values of d .

as filtering discrete-time fractional noise with a pole-zero filter. The resulting model is known as the *fractional pole-zero model* and is denoted by $\text{FPZ}(P, d, Q)$. The system function is

$$H_{\text{fpz}}(z) = \frac{1}{(1 - z^{-1})^d} \frac{D(z)}{A(z)} \quad (12.5.29)$$

The $\text{FPZ}(P, d, Q)$ is minimum-phase if $-\frac{1}{2} < d < \frac{1}{2}$ and $\text{PZ}(P, Q)$ is minimum-phase.

With regard to the long-range behavior of the model, we can show that as $l \rightarrow \infty$,

$$\rho(l) \sim C_\rho l^{2d-1} \quad (12.5.30)$$

where $C_\rho \neq 0$, and as $\omega \rightarrow 0$

$$R(e^{j\omega}) = \frac{1}{|1 - e^{-j\omega}|^{2d}} \frac{|D(e^{j\omega})|^2}{|A(e^{j\omega})|^2} \sim \frac{|D(0)|^2}{|A(0)|^2} \frac{1}{\omega^{2d}} \quad (12.5.31)$$

Parameter d controls the impulse response and the autocorrelation of the model at large lags and the spectrum at low frequencies. Parameters a_k and d_k control the impulse response and the autocorrelation of the model at small lags, and the spectrum at high frequencies.

Autoregressive fractionally integrated moving-average models. Fractional pole-zero models driven by white noise [*autoregressively fractionally integrated moving-average models (ARFIMA) models*] generate random signals whose samples are significantly dependent even if they are too far apart. In practice (e.g., geophysics, hydrology, economics) there are many time series in which the dependence between samples that are too far away, though small, is still too significant to be ignored. Such signals with *long-term persistence* can be effectively modeled using ARFIMA models, because of their flexibility in dealing with both short-term and long-term correlation structures. An alternative family of random fractal models for modeling long memory behavior is discussed in the next section.

Harmonic fractional pole-zero models. The FP(d) models with $0 < d < \frac{1}{2}$ exhibit long memory, but their spectrum peaks at zero frequency and their autocorrelation does not have any periodicity. We next discuss a class of *harmonic* models with long memory, periodic autocorrelations, and power spectra that resonate at any frequency in the interval $0 \leq \omega \leq \pi$. Such models are more appropriate for the modeling of data with strong periodicities because they exhibit long memory and pseudoperiodic behavior.

Let $e^{\pm j\theta}$ be a pair of complex conjugate poles on the unit circle and at angles $\pm\theta$ from the real axis. Then we have $(1 - e^{j\theta}z^{-1})(1 - e^{-j\theta}z^{-1}) = 1 - (2 \cos \theta)z^{-1} + z^{-2}$. The *harmonic fractional pole model*, denoted by HFP(d, θ), is a causal system defined by

$$H_{\theta,d}(z) = \frac{1}{(1 - 2z^{-1} \cos \theta + z^{-2})^d} = \sum_{n=0}^{\infty} h_{\theta,d}(n)z^{-n} \quad (12.5.32)$$

where d is a fractional parameter and θ is an angle controlling the location of the peak of the spectrum. For $\theta = 0$, Equation (12.5.32) reduces to a standard FP($2d$) model. The properties of this model are discussed in Problem 12.10. The minimum-phase HFP(d, θ) model can be cascaded with a minimum-phase PZ(P, Q) model to obtain an HFPZ(P, d, Q, θ) model that offers greater flexibility in controlling both the short-term and long-term correlation structure.

12.5.3 Symmetric α -Stable Fractional Pole-Zero Processes

Up to this point we have studied linear signal models driven by a sequence of IID Gaussian or non-Gaussian random variables with *finite* variance. However, many practical time series including isolated sharp spikes or bursts of spikes can be better described by random signal models with *infinite* variance. To ensure that some signal samples take large values with high probability, we need a probability density function with fat or heavy tails. We focus on the family of S α S random variables because of their heavy tails and the fact that they are invariant under linear transformations.

As we have seen in Chapters 4 and 5, the linear process

$$x(n) = \sum_{k=0}^{\infty} h(k) w(n-k) \quad (12.5.33)$$

is strictly stationary if (1) $w(n) \sim \text{IID}(0, \sigma_w^2)$ with $\sigma_w^2 < \infty$ (finite variance) and (2) $\sum_{k=-\infty}^{\infty} |h(k)| < \infty$, that is, the system is BIBO stable. However, to ensure stationarity when the input is S α S with $\sigma_w = \infty$ (power law tails), the sequence $|h(k)|$ should decay exponentially. Since the impulse response of a stable pole-zero system decays exponentially, its response to an S α S IID sequence is strictly stationary and S α S stable.

So far, we have discussed the properties of fractional pole-zero models and their response to white noise with finite variance. The following proposition specifies under what conditions the output of a PZ($0, d, 0$) model with stable excitation is defined.

THEOREM 12.4. Consider the following fractional pole model FP(d)

$$x(n) = \sum_{k=0}^{\infty} h(k)w(n-k) \quad \text{with} \quad h(k) = \frac{(d+k-1)!}{k!(d-1)!} \quad (12.5.34)$$

where $w(n)$ is IID and S α S. A necessary condition for the series (12.5.34) to converge is

$$-\infty < d < 1 - \frac{1}{\alpha} \quad (12.5.35)$$

When (12.5.35) holds, the series converges in the following sense:

1. $0 < \alpha \leq 1$: absolutely almost surely
2. $1 < \alpha \leq 2$: absolutely almost surely if $d \leq 0$ and absolutely surely if $d > 0$ and $\mu = 0$

Proof. See Samorodnitsky and Taqqu (1994).

We note that because both $h(n)$ and the tails of the input distribution decay as a power law, the stability of the model depends on α . Recall that no dependence on the input signal exists if the input signal has finite variance, because for $E\{w^2(n)\} < \infty$ the stability requirement is $\sum_{k=-\infty}^{\infty} |h(k)|^2 < \infty$.

The output of the inverse model $g(n) = h(n)|_{d \leftarrow -d}$ is defined for $-\infty < -d < 1 - 1/\alpha$ or $-(1 - 1/\alpha) < d < \infty$; hence, the model is minimum-phase if

$$-\left(1 - \frac{1}{\alpha}\right) < d < 1 - \frac{1}{\alpha} \quad (12.5.36)$$

The stability and minimum-phase regions for the FP(d) model with S α S IID excitations are shown in Figure 12.21. Theorem 12.4 applies for the model FPZ(P, d, Q) assuming it is stable, because it behaves asymptotically as the PZ($0, d, 0$) model.

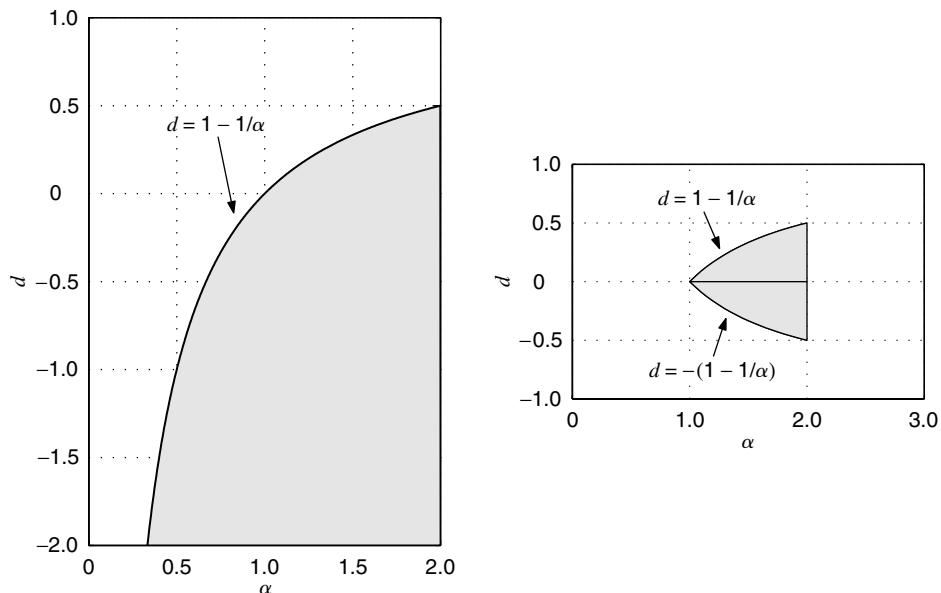


FIGURE 12.21

Stability (left) and minimum-phase (right) regions for a fractional pole model driven by an S α S IID sequence.

Although a linear stable process is strictly stationary, it is not second-order stationary because $E\{|x(n)|^2\} = \infty$. Therefore, the autocorrelation and the PSD of the process $x(n)$ do not exist. However, we can use the normalized autocorrelation of the signal model (12.5.33)

$$\rho(l) = \frac{\sum_{n=-\infty}^{\infty} h(n)h(n-l)}{\sum_{n=-\infty}^{\infty} h^2(n)} \quad (12.5.37)$$

and its Fourier transform to characterize the linear stable process $x(n)$. Clearly, this is a legitimate characterization for processes with finite variance and provides a reasonable characterization for stable linear processes because of the IID nature of the excitation $w(n)$. We can estimate $\rho(l)$ from a set of data $\{x(n)\}_0^{N-1}$ using the consistent estimator (Brockwell

$$\hat{\rho}(l) = \frac{\sum_{n=0}^{N-1+|l|} x(n)x(n-l)}{\sum_{n=0}^{N-1} x^2(n)} \quad (12.5.38)$$

12.6 SELF-SIMILAR RANDOM SIGNAL MODELS

In this section, we introduce the family of statistically self-similar or random fractal models, which are based on self-similar stochastic processes. Any segment of a self-similar process looks similar, in a statistical sense, to a scaled version of a larger segment of the process. Because of their practical importance, we focus on self-similar processes with stationary increments. We show that the stationary-increments requirement leads to processes whose autocorrelation sequences decay hyperbolically, that is, to models with long memory. We mainly focus on the fractional Brownian motion (nonstationary) and the fractional Gaussian noise (stationary) models, as well as their properties, simulation, and applications. However, we provide a brief introduction to self-similar processes with S&S increments, which result in random signal models with long memory and high variability.

12.6.1 Self-Similar Stochastic Processes

Each time a geologist takes a photograph of a geological object, say, a fossil, she or he includes in the picture an object with *known scale* (e.g., a coin or a ruler), because without the scale, it is impossible to determine whether the photograph covers 10 cm or 10 m. For this reason we say that geological phenomena are *scale-invariant*, or that they do not have a *characteristic scale*.

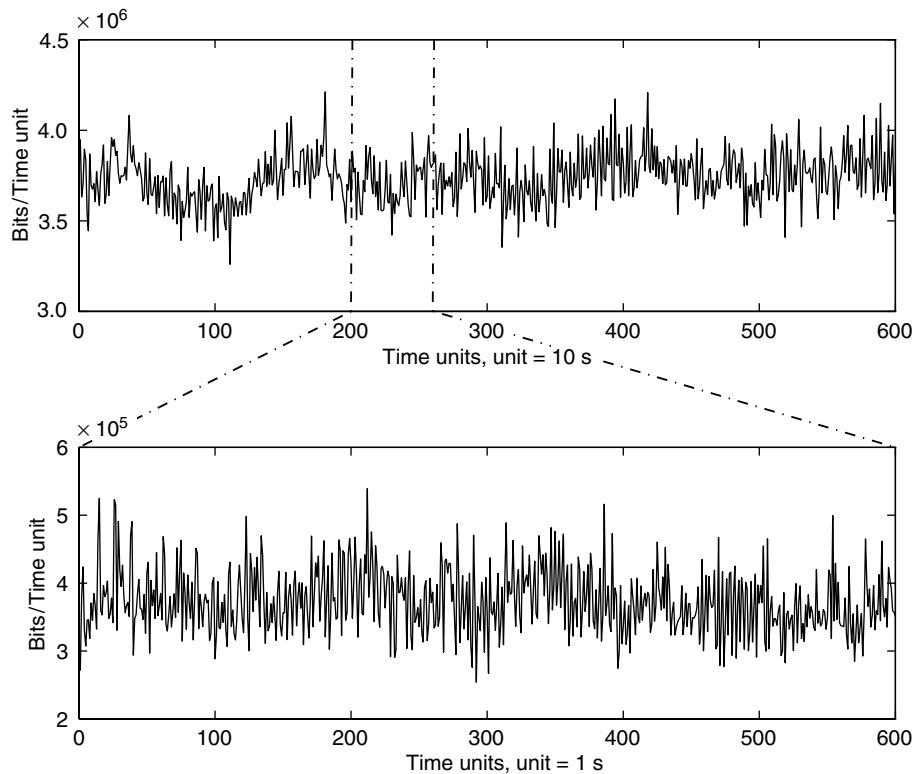
If we can reproduce an object by magnifying some portion of it, we say that the object is *scale-invariant*, or *self-similar*. Thus, self-similarity is invariance with respect to scaling. Such self-similar geometric objects are known as *fractals* (Mandelbrot 1982).

A signal $x(t)$ is self-similar if[†] $x(ct) = c^H x(t)$. It can be easily seen that a signal described by a power law $x(t) = \alpha t^\beta$ is self-similar. However, such signals are of limited interest. A more interesting and useful type of signal is that exhibiting a weaker, that is, statistical, version of self-similarity. A random signal is called (*statistically*) *self-similar* if its statistical properties are scale-invariant, meaning that its statistics do not change under magnification or reduction. Self-similar random signals are also known as *random fractals*.

Statistical self-similarity means that small fluctuations at small scales become larger fluctuations at larger scales. Therefore, as we analyze more and more data, these ever-larger fluctuations increase the value of the measured variance, which in the limit becomes infinite. This increase of variance with the length of the data has been observed in the analysis of various practical time series that exhibit self-similar behavior. Figure 12.22 provides a visual illustration of the self-similar behavior of the variable-rate video traffic time series (Garrett and Willinger 1994).

These ideas can be formalized within the context of the theory of stochastic processes by using the following definition.

[†]The superscript H is an index and not a conjugate transposition operator. For lack of better notation, we will continue to use the accepted notation.

**FIGURE 12.22**

Pictorial illustration of self-similarity for the variable-bit-rate video traffic time series. The bottom series is obtained from the top series by expanding the segment between the two vertical lines. Although the two series have lengths of 600 and 60s, they are remarkably similar visually and statistically. (Courtesy of M. Garrett and M. Vetterli.)

DEFINITION 12.1. A continuous-time stochastic process $x(t)$ is said to be (statistically) *self-similar* with (self-similarity) index[†] H (H -ss) if and only if, for any scaling parameter $c > 0$, the processes $x(ct)$ and $c^H x(t)$ are statistically equivalent, that is, they have the same finite-dimensional distributions. Symbolically

$$x(ct) \stackrel{d}{=} c^H x(t) \quad (12.6.1)$$

where the symbol $\stackrel{d}{=}$ denotes equality in distribution and, more specifically, equality of finite-dimensional joint probability distributions.

It should be emphasized that individual realizations of the process are not necessarily deterministically scale-invariant. The above definition of self-similarity has several implications, which can be summarized as follows:

- A change in the time scale is *statistically* equivalent to a change in the amplitude scale. Hence, the statistic of $x(t)$ is invariant under the transformation

$$x(t) \rightarrow c^{-H} x(ct) \quad (12.6.2)$$

- To obtain statistically equivalent processes, the time axis must be scaled *differently* from the amplitude axis. In the language of fractals, we say that the graphs $\{t, x(t)\}$ and $\{t, c^{-H} x(ct)\}$, $0 \leq t < \infty$, are *statistically self-affine* because the scaling factor is

[†]Also known as the *Hurst exponent*.

different for the time and amplitude axes. An example of such a self-similar process for $H = \frac{1}{2}$ is shown in Figure 12.23. This process, whose distribution at each t is Gaussian, is known as (ordinary) *Brownian motion*. (A detailed discussion of Brownian motion is given in the next section.) The time trace shown in the top plot in Figure 12.23 is generated as a discrete equivalent of $x(t)$, using 16,384 samples over unit time interval. When it is plotted as a continuous curve, we lose sight of its discrete nature and view it as a fractal curve that is indistinguishable from a continuous Brownian—a true fractal curve possessing self-similarity at all levels of magnification. Statistical self-affinity of $x(t)$ is evident as we zoom into it. The zooming area in the top plot is shown as a box, and the scaled curve is shown in the middle plot. Note that we scaled the middle one-fourth of the time axis while the amplitude axis was magnified by 2 since $4^H = 4^{1/2} = 2$. This retained the statistical similarity of the middle curve to the original one. Further scaling of time axis by 4 and the amplitude axis by 2 is shown in the bottom plot of Figure 12.23. Once again the resulting plot is statistically similar to the original one. This Brownian motion displayed at different levels of resolution demonstrates the concept of statistical self-affinity.

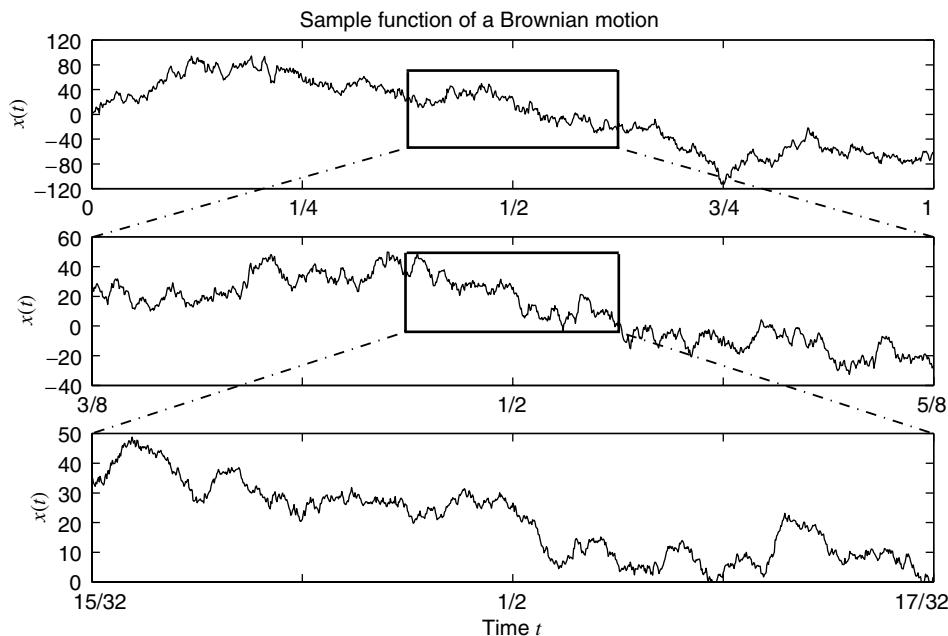


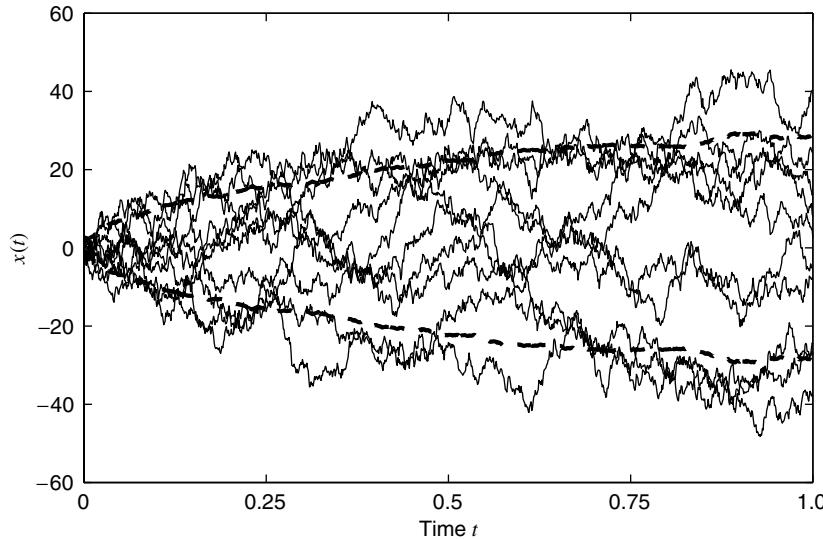
FIGURE 12.23

Statistical self-affine property of the Brownian motion trace.

- If we set $ct = 1$ in (12.6.2), we have

$$x(t) \stackrel{d}{=} t^H x(1) \quad t > 0 \quad (12.6.3)$$

Therefore, self-similar processes cannot be stationary, except for $H = 0$. This nonstationarity of the Brownian motion trace $x(t)$ of Figure 12.23 is shown in Figure 12.24, which illustrates the spreading of signal values about the mean value of zero as time increases. For display purposes, 10 sample functions of $x(t)$ are shown, all of which begin at $x(0) = 0$. This spreading is in a statistical sense, in that some traces return to zero and some cases return even more than once. To determine this statistical spreading, 100 sample functions were used, and the sample standard deviation $\sigma_x(t)$ at each t was

**FIGURE 12.24**

The diffusion property of the Brownian motion trace.

computed. This $\pm\sigma_x(t)$, shown as dashed lines in Figure 12.24, clearly indicates the diffusion (or nonstationarity) property of Brownian motion. Note that since the standard deviation is proportional to $E\{|x(t + \Delta) - x(t)|\}$, we have

$$\sigma_x(t) \propto t^H = t^{1/2} \quad (12.6.4)$$

for the Brownian motion, and the dashed line in Figure 12.24 confirms it.

- For *strict-sense* self-similar processes, all finite-dimensional distributions are equal. However, for *wide-sense* self-similar processes, only second-order moments are equal. From (12.6.2) these moments are given as

$$\mu_x(t) \triangleq E\{x(t)\} = c^{-H}\mu_x(ct) \quad (12.6.5)$$

$$r_x(t_1, t_2) \triangleq E\{x(t_1)x(t_2)\} = c^{-2H}r_x(ct_1, ct_2) \quad (12.6.6)$$

Clearly, for Gaussian processes the two types of self-similarity are equivalent.

Because of their practical importance, we focus on self-similar stochastic processes that have stationary increments.

DEFINITION 12.2. A real-valued process $x(t)$ has *stationary increments* if

$$x(t + \tau) - x(\tau) \stackrel{d}{=} x(t) - x(0) \quad \text{for all } \tau \quad (12.6.7)$$

In practical applications, the nature of processes with stationary increments is analyzed using a quantity known as the *semivariogram*, defined by

$$v_x(\tau) \triangleq \frac{1}{2}E\{[x(t + \tau) - x(t)]^2\} \quad (12.6.8)$$

which, for stationary processes, reduces to

$$v_x(\tau) = 2[r_x(0) - r_x(\tau)] \quad (12.6.9)$$

We next turn our attention to self-similar processes with stationary increments

DEFINITION 12.3. A continuous-time stochastic process is *self-similar with stationary increments (H-ssi)* if and only if

- It is *self-similar* with index H .
- It has *stationary increments*.

As shown in the following theorem, the requirements for self-similarity and stationary increments completely specify the second-order moments of the underlying process $x(t)$.

THEOREM 12.5. The mean value, variance, and autocorrelation of an H -sssi process are given by, respectively

$$\mu_x(t) = 0 \quad (12.6.10)$$

$$\sigma_x^2(t) = t^{2H} \sigma_H^2 \quad (12.6.11)$$

$$r_x(t_1, t_2) = \frac{1}{2} \sigma_H^2 (|t_1|^{2H} - |t_1 - t_2|^{2H} + |t_2|^{2H}) \quad (12.6.12)$$

where $\sigma_H^2 = E\{x^2(1)\}$.

Proof. From (12.6.2) we have, for $t = 0$,

$$x(0) \stackrel{d}{=} c^{-H} x(c0) = c^{-H} x(0) \Rightarrow x(0) = 0 \quad (12.6.13)$$

Also from (12.6.2) and (12.6.3), we conclude that

$$\mu_x(t) = E\{x(t)\} = E\{c^{-H} x(ct)\} = c^{-H} E\{x(ct)\} = t^H E\{x(1)\} \quad (12.6.14)$$

Using the stationary increment property (12.6.7), (12.6.13), and (12.6.14), we obtain

$$E\{x(t + \tau) - x(\tau)\} = E\{x(t) - x(0)\} = E\{x(t)\} = t^H E\{x(1)\} \quad (12.6.15)$$

Using the self-similarity definition, however, we have

$$E\{x(t + \tau) - x(\tau)\} = [(t + \tau)^H - \tau^H] E\{x(1)\} \quad (12.6.16)$$

Comparing (12.6.15) and (12.6.16), we conclude that $E\{x(1)\} = 0$; hence from (12.6.14)

$$\mu_x(t) = 0$$

which proves (12.6.10). Similarly, since $x(t) \stackrel{d}{=} t^H x(1)$, for $t > 0$, we have

$$\sigma_x^2(t) = E\{x^2(t)\} = t^{2H} E\{x^2(1)\} = t^{2H} \sigma_H^2$$

which proves (12.6.11). Finally, again using stationarity of the increments and (12.6.11), we obtain

$$E\{[x(t_1) - x(t_2)]^2\} = E\{[x(t_1 - t_2) - x(0)]^2\} = \sigma_H^2 (t_1 - t_2)^{2H} \quad (12.6.17)$$

$$\begin{aligned} \text{or } E\{[x(t_1) - x(t_2)]^2\} &= E\{x^2(t_1)\} + E\{x^2(t_2)\} - 2E\{x(t_1)x(t_2)\} \\ &= \sigma_H^2 t_1^{2H} + \sigma_H^2 t_2^{2H} - 2r_x(t_1, t_2) \end{aligned} \quad (12.6.18)$$

where $r_x(t_1, t_2)$ is the autocorrelation function of $x(t)$. Combining the last two equations, we obtain

$$r_x(t_1, t_2) = \frac{1}{2} \sigma_H^2 [t_1^{2H} - (t_1 - t_2)^{2H} + t_2^{2H}] \quad (12.6.19)$$

which completes the proof of the theorem.

Self-similar processes with stationary increments are well-defined if $H > 0$ and $x(0) = 0$ with probability 1 (Vervaat 1987). For $H = 1$, $x(t) = |t|x(1)$; that is, the realizations are lines through the origin, and the process is of no interest. For $H = 0$, we have $x(t) = 0$, which is a trivial process. For $H < 0$ the process is not mean square continuous, and for $H > 1$ the increments are nonstationary. The permissible range of H is determined by the existence of moments: If $x(t)$ is H -sssi with finite variance, then $0 < H \leq 1$ (Samorodnitsky and Taqqu 1994).

The autocorrelation (12.6.12) shows that H -sssi processes are nonstationary. Despite this nonstationarity, we can define a time-averaged spectrum. Since small scales correspond to large frequencies and large scales to small frequencies, the amplitude of the fluctuations is small at high frequencies and large at low frequencies. In light of the previous discussion, it should not come as a surprise that the power spectrum of self-similar processes follows

a power law, that is, is proportional to $1/|F|^\beta$. Indeed, it has been shown (Flandrin 1989) that the time-averaged power spectrum of an H -sssi process is given by

$$R_x(F) = \frac{\sigma_H^2}{|F|^{2H+1}} \quad (12.6.20)$$

where F is the frequency in cycles per unit of time. As we can easily see, $R_x(cF) = c^{-(2H+1)} R_x(F)$, which shows that the process is wide-sense self-similar.

12.6.2 Fractional Brownian Motion

If we restrict the probability distribution of an H -sssi process to being Gaussian, we obtain a unique process known as the *fractional Brownian motion*, abbreviated as FBM (Mandelbrot and Van Ness 1968). These FBMs have Hurst exponents in the range $0 < H < 1$. The (ordinary) Brownian motion of Figure 12.23 is a special case of FBM when $H = \frac{1}{2}$.

DEFINITION 12.4. A Gaussian H -sssi process, $0 < H \leq 1$, is called *fractional Brownian motion* (FBM) and is denoted by $B_H(t)$.

There are several equivalent definitions of FBM process, which are summarized by the following theorem (Samorodnitsky and Taqqu 1994; Beran 1994).

THEOREM 12.6. If $0 < H \leq 1$ and $\sigma_H^2 = E\{x^2(1)\}$, the following statements are equivalent

1. $B_H(t)$ is Gaussian and H -sssi.
2. $B_H(t)$ is fractional Brownian motion with self-similarity index H .
3. $B_H(t)$ is Gaussian and has mean zero for $H < 1$ and autocorrelation function

$$r_{B_H}(t_1, t_2) = \frac{1}{2}\sigma_H^2(|t_1|^{2H} - |t_1 - t_2|^{2H} + |t_2|^{2H}) \quad (12.6.21)$$

EXAMPLE 12.6.1. In Figure 12.25 we show time traces of FBMs for $H = 0.2, 0.5$, and 0.8 . Clearly, in these traces there is a qualitative difference between each trace that is very noticeable. For a low value of $H = 0.2$, the trace shows more fractured or crinkled behavior. This behavior occurs for $0 < H < 0.5$, and the corresponding traces have tendencies to turn back upon themselves (negative correlation). The corresponding property is known as *antipersistence*. A

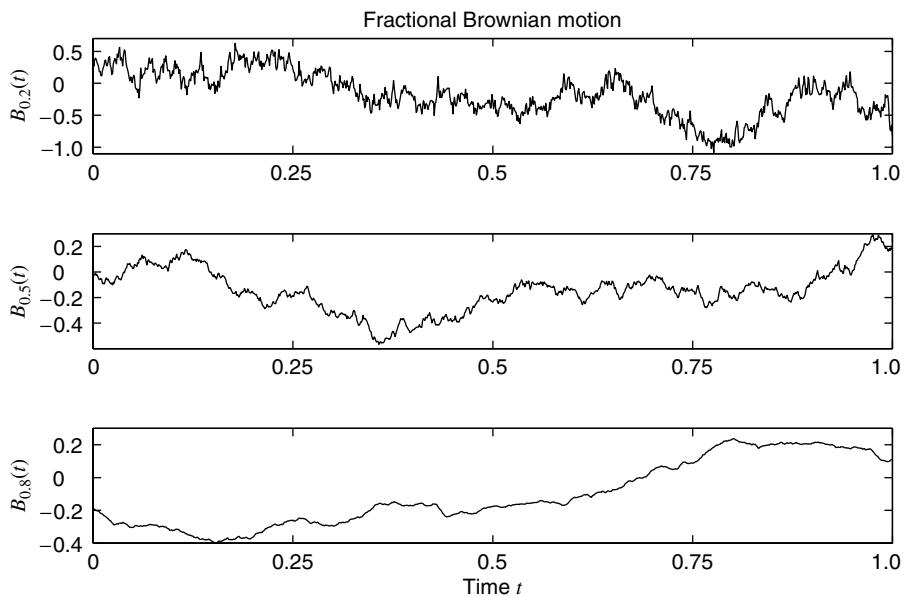


FIGURE 12.25

Time traces of fractional Brownian motion for $H = 0.2$, $H = 0.5$, and $H = 0.8$.

stock market fluctuation is a good example of this process. As H increases, the amount of crinkle reduces. For $H = 0.5$ we have the (ordinary) Brownian motion for which the correlation is zero, and the trace shows no preferred tendency to turn back or persist in the same direction (neutral in persistence). For a high value of $H = 0.8$, the trace is smoother, and in fact for $0.5 < H < 1$, the FBM traces show persistence in the direction in which they are moving (positive correlation). This property is known as *persistence*. Typical coastlines (boundaries between land and water) are good examples of such traces.

From the above example, we note that the fractal behavior of traces diminishes as H increases from 0 to 1. Hence there must be an inverse relationship between H and the fractal dimension D (also known as the Hausdorff dimension). The concept of dimension is closely related to the property of self-similarity or scaling. For the purpose of discussion, let us consider our natural Euclidean dimensions. In one dimension, a line segment possesses a scaling property. If it is subdivided into N identical line segments, then each segment is scaled down by the ratio $r = 1/N$ from the whole, or $Nr = 1$. A square is a two-dimensional plane that possesses the scaling property. If it is subdivided into N equal squares, then each square side is scaled down by a factor of $r = 1/\sqrt{N}$, or $Nr^2 = 1$. Carrying this analysis to the cube in three dimensions, we observe that if a cube is subdivided into N identical cubes, then each subcube edge is scaled down by the factor $r = 1/\sqrt[3]{N}$, or $Nr^3 = 1$. Now we can generalize this analysis to an arbitrary noninteger dimension D . If a D -dimensional object is subdivided into N identical copies of itself, then the side of each copy is scaled down by the ratio $r = 1/\sqrt[D]{N}$, or $Nr^D = 1$. Thus we obtain

$$D = \frac{\log N}{\log (1/r)} \quad (12.6.22)$$

The above approach can also be used to determine the fractal dimension D of the FBM traces and to relate it to the Hurst exponent H . One interesting technique for determining the fractal dimension is known as *box counting*. The basic idea is to compute the total number N of enclosing boxes (or rectangles) needed to cover all identical subtraces that have been scaled down by the ratio r from the whole trace and then use formula (12.6.22) to estimate the fractal dimension. Refer to the top plot of Figure 12.23. The enclosing box shows that if the whole trace is divided into 4 identical subtraces, then the box height is scaled down by $(\frac{1}{4})^{1/2} = \frac{1}{2}$. Thus the area of each rectangular box is

$$\left(\frac{1}{4}\right)\left(\frac{1}{2}\right) = \frac{1}{4^{3/2}} = \frac{1}{4^{1+1/2}} \quad (12.6.23)$$

However, we have to relate the scaling of smaller (identical) *square* boxes to the original box since the original is a square box of unit side length (this implicitly assumes that the amplitude axis in Figure 12.23 is unity, which is not unreasonable since we are using fractions). The smaller square boxes of side length $r = \frac{1}{4}$ have area equal to $1/4^2$. Thus the number of square boxes required to cover each subinterval is (note the box counting)

$$\frac{1/4^{3/2}}{1/4^2} = \frac{1}{4^{1/2-1}} \quad (12.6.24)$$

Since there are 4 subintervals in Figure 12.23, the total number of square boxes required to cover the whole trace is

$$N = 4 \left(\frac{1}{4^{1/2-1}} \right) = \frac{1}{4^{1/2-2}} \quad (12.6.25)$$

Hence substituting (12.6.25) into (12.6.22), and using $r = \frac{1}{4}$, we obtain

$$D = \frac{\log (1/4^{1/2-2})}{\log (1/\frac{1}{4})} = \frac{\log 4^{2-1/2}}{\log 4} = 2 - \frac{1}{2} = 1.5 \quad (12.6.26)$$

which is the fractal dimension of (ordinary) Brownian motion. Generalizing to $0 < H < 1$, we can show that (see Problem 12.12)

$$D = 2 - H \quad (12.6.27)$$

Thus the sample paths of fractional Brownian motion are fractal curves with Hausdorff dimension $D = 2 - H$ (Falconer 1990). Referring to Figure 12.25, we see the fractal dimensions of the fractional Brownian motions are $D = 1.8$ for $H = 0.2$ (antipersistent), $D = 1.5$ for $H = 0.5$ (Brownian motion), and $D = 0.8$ for $H = 1.2$ (persistent). Thus the more wiggly the trace, the higher the dimension.

Continuous-time fractional pole systems. In Section 12.5 we used a discrete-time fractional pole to obtain a system with long memory. When this system is driven by a WGN process, the result is a long-memory process called discrete-time FGN. This leads to the following question: Can we use a continuous-time fractional pole to obtain a long-memory system that could be used to generate a long-memory process in general and fractional Brownian motion in particular? The answer is yes, so now we provide an intuitive engineering explanation.

For any $d > 0$, we have the following Laplace transform pair (Abramowitz and Stegun 1970)

$$h_d(t) = \frac{1}{\Gamma(d)} t^{d-1} u(t) \iff H_d(s) = \frac{1}{s^d} \quad (12.6.28)$$

where $\Gamma(\cdot)$ is the gamma function. Note that for $d = 1$, $h_1(t)$ corresponds to an ideal integrator. However, for fractional d , the function $h_d(t)$ has a hyperbolic decay. The result is a system with long memory called the *fractional integrator*. These topics are the subject of a discipline known as fractional calculus (Oldham and Spanier 1974).

The output of the fractional integrator is provided by the convolution integral

$$x(t) = \frac{1}{\Gamma(d)} \int (t - \tau)^{d-1} u(t - \tau) w(\tau) d\tau \quad (12.6.29)$$

which satisfies the scaling property

$$y(t) = \frac{1}{\Gamma(d)} \int (t - \tau)^{d-1} w(c\tau) d\tau = \frac{c^{-d}}{\Gamma(d)} \int (ct - \lambda)^{d-1} w(\lambda) d\lambda = c^{-d} x(ct) \quad (12.6.30)$$

where $\lambda = c\tau$. Linear systems that satisfy (12.6.30) are said to be *linear, scale-invariant* systems (Wornell 1996). We emphasize that while linear, shift-invariant systems with rational system functions have memory that decays exponentially, linear, scale-invariant systems exhibit self-similarity and long (hyperbolically decaying) memory.

Intuition suggests that the output of scale-invariant systems, driven by white noise, should exhibit statistical self-similarity. Indeed, it can be shown that linear, scale-invariant systems can be used to generate fractional Brownian motion processes (Samorodnitsky and Taqqu 1994). More specifically, the fractional Brownian motion process can be generated by passing white noise through a linear, scale-invariant system

$$B_H(t) = \int_{-\infty}^{\infty} h_t(\tau) w(\tau) d\tau \quad (12.6.31)$$

with the following causal impulse response

$$h_t(\tau) = \frac{1}{C(H)} \{ [t - \tau]_+^{H-1/2} - [(-\tau)_+]^{H-1/2} \} \quad (12.6.32)$$

where $C(H) = \left\{ \int_0^{\infty} [(1 + \tau)^{H-1/2} - \tau^{H-1/2}]^2 d\tau + \frac{1}{2H} \right\}^{1/2}$ (12.6.33)

and $u_+ = \begin{cases} u & \text{if } u \geq 0 \\ 0 & \text{if } u < 0 \end{cases}$ (12.6.34)

We note that the change from the impulse response (12.6.28) to (12.6.32) was introduced by Mandelbrot (1982) to ensure that $B_H(t)$ has the required properties (Wornell 1993; Kasdin 1995). An equivalent *harmonizable* representation of fractional Brownian motion in the frequency domain is also derived in Samorodnitsky and Taqqu (1994).

12.6.3 Fractional Gaussian Noise

The discrete fractional Gaussian noise is a stationary sequence obtained by periodically sampling the fractional Brownian motion process $B_H(t)$ and then computing the first difference. The resulting random sequence is $x(nT) \triangleq B_H(nT) - B_H(nT - T)$, where T is the sampling interval. Since the fractional Brownian motion process is statistically scale-invariant, we set $T = 1$. Therefore, the *discrete fractional Gaussian noise* process is defined by

$$x(n) \triangleq B_H(n) - B_H(n-1) \quad (12.6.35)$$

and it is simply referred to as *FGN*.

We next determine the second-order moments, that is, the autocorrelation and PSD of the FGN process.

THEOREM 12.7. The autocorrelation sequence of the discrete fractional Gaussian noise is

$$r_x(l) = \frac{1}{2}\sigma_H^2(|l-1|^{2H} - 2|l|^{2H} + |l+1|^{2H}) \quad (12.6.36)$$

Since the correlation depends only on the distance l between the samples, the process is wide-sense stationary.

Proof. Using (12.6.21) and (12.6.35), we can easily show that

$$\begin{aligned} E\{x(n)x(n-l)\} &= E\{[B_H(n) - B_H(n-1)][B_H(n-l) - B_H(n-l-1)]\} \\ &= \frac{1}{2}\sigma_H^2[(l-1)^{2H} - 2l^{2H} + (l+1)^{2H}] \end{aligned}$$

which leads to (12.6.36).

Figure 12.26 shows the autocorrelation sequence for various values of the self-similarity index H . Note that for $H = \frac{1}{2}$ we have $r_x(l) = \delta(l)$, which shows that the FGN process is white noise.

THEOREM 12.8. The power spectrum of the FGN process $x(n)$ is given by

$$R_x(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r_x(l)e^{-j\omega l} = \sigma_H^2 C_H |1 - e^{-j\omega}|^2 \sum_{k=-\infty}^{\infty} \frac{1}{|\omega + 2\pi k|^{2H+1}} \quad (12.6.37)$$

where

$$C_H = 2H\Gamma(2H)\sin(\pi H) \quad (12.6.38)$$

is a constant dependent on the self-similarity index.

Proof. A rigorous proof can be found in Samorodnitsky and Taqqu (1994). Here we provide a more heuristic proof. The sequence $x(n)$ is obtained by sampling the FBM process $B_H(t)$ every $T = 1$ time unit, that is, evaluating $s(n) = B_H(nT)$, and then computing the first difference $x(n) = s(n) - s(n-1)$. From the sampling theorem (see Chapter 2) we have

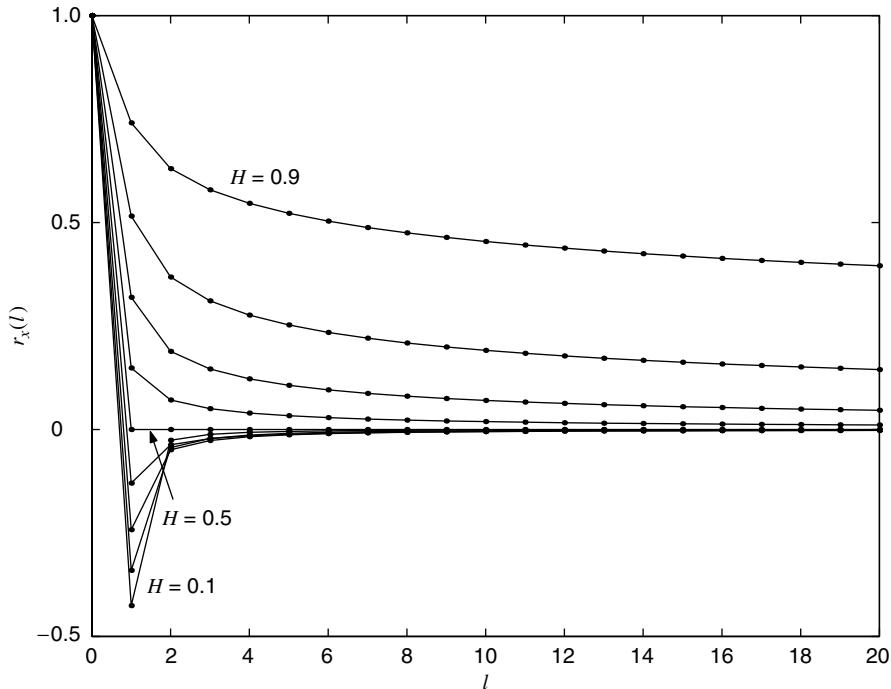
$$R_s(e^{j\omega}) = \frac{1}{T} \sum_{k=-\infty}^{\infty} R_B\left(\frac{\omega}{T} + \frac{2\pi k}{T}\right)$$

where $\omega = \Omega T$. The frequency response of the first-difference filter is $H(e^{j\omega}) = 1 - e^{-j\omega}$, or

$$|H(e^{j\omega})|^2 = 2(1 - \cos \omega) = 4 \sin^2\left(\frac{\omega}{2}\right)$$

Since, from (12.6.20),

$$R_B(F) = C_H \frac{\sigma_H^2}{|F|^{2H+1}}$$

**FIGURE 12.26**

Autocorrelation sequence of FGN for $H = 0.1$ to $H = 0.9$ at 0.1 increments.

the power spectrum of $x(n)$ is

$$R_x(e^{j\omega}) = |H(e^{j\omega})|^2 R_s(e^{j\omega}) = 2\sigma_H^2 C_H (1 - \cos \omega) \frac{1}{T} \sum_{k=-\infty}^{\infty} \frac{1}{|\omega + 2\pi k|^{2H+1}}$$

which results in (12.6.37) for $T = 1$.

Figure 12.27 shows the PSD of FGN for various values of the self-similarity index H . Note that for $H = \frac{1}{2}$ we have a flat PSD, which shows that the FGN process is white noise.

Self-similarity. The discrete FGN process is asymptotically (i.e., at large scales) self-similar. Indeed, the autocorrelation

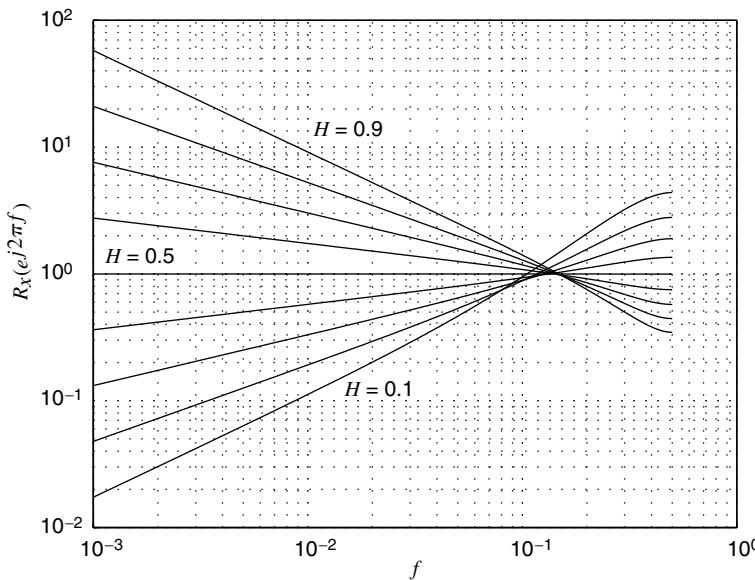
$$r(l) \sim \sigma_H^2 H(2H-1)|l|^{2H-2} \quad \text{as } |l| \rightarrow \infty, H \neq \frac{1}{2}$$

decays hyperbolically for large lags, and the PSD

$$R(e^{j\omega}) \sim C_H \frac{\sigma_H^2}{|\omega|^{2H-1}} \quad \text{as } |\omega| \rightarrow 0, H \neq \frac{1}{2}$$

follows a power law as the frequency becomes very small, that is, as the period becomes very large.

Process memory. The FGN process has long memory for $\frac{1}{2} < H < 1$ because the summation $\sum_{l=-\infty}^{\infty} r(l) = \infty$, or equivalently $R(e^{j\omega}) \rightarrow \infty$ as $|\omega| \rightarrow 0$. In this case the autocorrelation decays very slowly, the frequency response resembles a low-pass filter, and the resulting realizations look smooth. In contrast, the process exhibits short memory for $0 < H < \frac{1}{2}$, because $\sum_{l=-\infty}^{\infty} |r(l)| < \infty$ and $\sum_{l=-\infty}^{\infty} r(l) = 0$, or equivalently $R(e^{j\omega}) \rightarrow 0$ as $|\omega| \rightarrow 0$. In addition, for $0 < H < \frac{1}{2}$, the correlation is negative, that is,

**FIGURE 12.27**

PSD function of FGN for $H = 0.1$ to $H = 0.9$ at 0.1 increments.

$r(l) < 0$ for $l \neq 0$; and the process exhibits negative dependence, or antipersistence. In this case the autocorrelation decays very rapidly, the frequency response resembles a high-pass filter, and the resulting realizations look rough.

Comparison between FGN and FPN. The discrete-time FGN and FPN processes have been independently introduced and have been developed using different approaches. However, close inspection of their second-order statistics reveals some interesting similarities and differences, which are summarized in Table 12.3. The most interesting feature is that both processes become asymptotically self-similar at large scales.

12.6.4 Simulation of Fractional Brownian Motions and Fractional Gaussian Noises

Although statistical self-similar processes are relatively easy to describe notationally [see (12.6.1)], they are not easy to generate since there is no explicit (or compact) mathematical formula to do so. The FBMs and FGNs are special cases of these processes that have independent increments with underlying distribution that is Gaussian. Although an explicit formula exists for FBM [see (12.6.31)], the additional complication is that we cannot generate a continuous trace (this would require infinitely long memory). We can only hope to generate an approximate, sampled version of the process on a computer. Thus, as explained before, these simulations are not self-similar at all scales. Nevertheless, we provided plots of these processes in Figure 12.25 for various values of the self-similarity index H . This can be done via techniques that either use properties of the processes or employ indirect approaches. In this section, we provide a brief summary of some of these techniques. For more detailed discussion see Samorodnitsky and Taqqu (1994) and Barnsley et al. (1988). We begin with the simulation of ordinary Brownian motion, which is easy to generate.

Cumulative-sum method. This technique is a direct method that is suitable for generating FBM for $H = 0.5$. We note that the increments of this process not only are stationary but also are uncorrelated with one another. These increments then form the WGN process,

which can be simulated on a computer. Thus by integrating WGN we can obtain the ordinary Brownian motion. For discrete FBM, this requires taking the cumulative sum of the generated WGN sequence. Therefore, the steps in generating the ordinary Brownian motion are as follows:

1. Subdivide the time axis into a sufficiently fine grid. Let the number of grid points be N .
2. Generate N independent Gaussian random numbers with mean 0 and variance σ^2 . In MATLAB this can be done using the `randn(N, 1)` function.
3. Obtain a cumulative sum of the random numbers obtained in step 2 above. In MATLAB use the `cumsum` function. The resulting sequence is a discrete approximation of ordinary Brownian motion.

TABLE 12.3

Similarities and differences between discrete fractional Gaussian noise and discrete fractional pole noise.

	Discrete fractional Gaussian noise	Discrete fractional pole noise
Definition	$x(n) \triangleq B_H(n) - B_H(n-1)$	$x(n) \triangleq \sum_{k=0}^{\infty} \frac{\Gamma(k+d)}{\Gamma(k+1)\Gamma(d)} w(n-k)$ $w(n) \sim WN(0, \sigma^2) \quad -\frac{1}{2} < d < \frac{1}{2}$
Autocorrelation	$r(l) = \frac{1}{2} \sigma_H^2 (l-1 ^{2H} - 2 l ^{2H} + l+1 ^{2H})$	$r(l) = \frac{\sigma^2 (-1)^l \Gamma(1-2d)}{\Gamma(1+l-d) \Gamma(1-l-d)}$
Power spectrum	$R(e^{j\omega}) = \sum_{k=-\infty}^{\infty} \frac{C_H \sigma_H^2 (1 - \cos \omega)}{ \omega + 2\pi k ^{2H+1}}$	$R(e^{j\omega}) = \frac{\sigma^2}{[2 \sin(\omega/2)]^{2d}}$
Self-similarity (as $ l \rightarrow \infty$)	$r(l) \sim \sigma_H^2 H (2H-1) l ^{2H-2} \quad H \neq \frac{1}{2}$	$r(l) \sim C_d l ^{2d-1} \Rightarrow d = H - \frac{1}{2}$
Self-similarity (as $ \omega \rightarrow 0$)	$R(e^{j\omega}) \sim C_H \frac{\sigma_H^2}{ \omega ^{2H-1}} \quad H \neq \frac{1}{2}$	$R(e^{j\omega}) \sim \frac{\sigma^2}{ \omega ^{2d}} \Rightarrow d = H - \frac{1}{2}$
Long-memory	$\sum_{l=-\infty}^{\infty} r(l) = R(0) = \infty$ $\frac{1}{2} < H < 1$	$0 < d < \frac{1}{2}$
Short-memory	$\sum_{l=-\infty}^{\infty} r(l) = R(0) = 0$ $0 < H < \frac{1}{2}$ $\sum_{l=-\infty}^{\infty} r(l) < \infty \quad r(l) < 0, \text{ for } l \neq 0$	$-\frac{1}{2} < d < 0$
Partial correlation		$k_m = \frac{d}{m-d} \quad m = 1, 2, 3, \dots$

A MATLAB function to implement the above steps is explored in Problem 12.13. Since it is difficult to generate properly correlated random numbers, the cumulative-sum method is not suitable for H other than 0.5.

Spectral synthesis method. This method can be used to generate FBM with an index $0 < H < 1$. The basic principle behind the spectral synthesis approach is that if we can construct its spectral density function $R_x(F)$, then we can obtain the corresponding FBM through inverse transformation. From (12.6.20), we have

$$R_x(F) \propto \frac{1}{|F|^\beta} \quad \beta = 2H + 1 \quad (12.6.39)$$

Also, similar to the spectral density function relation (5.3.2) for discrete-time stochastic processes, we have

$$R_x(F) = \lim_{T \rightarrow \infty} \left[\frac{1}{T} |X_c(F)|^2 \right] \quad (12.6.40)$$

Thus from (12.6.39) and (12.6.40) it is possible to obtain a frequency-domain method for approximating samples of an FBM with $0 < H < 1$. Let $\{x(n)\}$ be the sample functions of an FBM with Hurst parameter H . Then its DTFT magnitude $|X(e^{j\omega})|$ has the form

$$|X(e^{j\omega})| \propto \frac{1}{|\omega|^{\beta/2}} \quad -\pi < \omega \leq \pi \quad (12.6.41)$$

Since this is a continuous function, we use the DFT approach to obtain samples in the time domain. If we sample $X(e^{j\omega})$ at N equispaced frequencies $\omega_k = 2\pi k/N$, $0 \leq k \leq N-1$, then the DFT magnitude has the form

$$|\tilde{X}(k)| \propto \begin{cases} \frac{1}{k^{\beta/2}} & 0 \leq k \leq \frac{N}{2} \\ |\tilde{X}(N-k)| & \frac{N}{2} < k \leq N-1 \end{cases} \quad (12.6.42)$$

The phase of $\tilde{X}(k)$ can be chosen to be random, uniformly distributed over $[-\pi, \pi]$ subject to the constraint of odd symmetry. Finally, taking the IDFT of $\tilde{X}(k)$ results in a sequence that approximates samples of the FBM with $H = (\beta - 1)/2$. The steps of this spectral synthesis method can be summarized as follows:

1. Given H , determine $\beta = 2H + 1$.
2. Choose sufficiently large N , and use a suitable proportionality constant to generate $|\tilde{X}(k)|$ according to (12.6.42).
3. Randomize phase $\theta(k)$; that is, generate phase values according to

$$\theta(k) = \begin{cases} \text{uniform random number over } [-\pi, \pi] & 0 \leq k \leq \frac{N}{2} \\ -\theta(N-k) & \frac{N}{2} < k \leq N-1 \end{cases} \quad (12.6.43)$$

4. Assemble $\tilde{X}(k) = |\tilde{X}(k)| \exp j\theta(k)$, $0 \leq k \leq N-1$, and determine the IDFT to obtain $x(n)$.

One major problem with this technique is that the resulting sequence is periodic with period N due to the DFT operation (or sampling in the frequency domain). Therefore, to avoid these boundary problems, a middle third of the sequence is used as a representative FBM trace. The FBM traces shown in Figure 12.25 were generated using the above steps. A MATLAB function to implement the above steps is explored in Problem 12.14.

Note that the corresponding FGN sequence is obtained by taking a first-order difference of the generated FBM sequence, that is,

$$w(n) = x(n) - x(n-1) \quad 1 \leq n \leq N-1 \quad (12.6.44)$$

Random midpoint replacement method. This is another direct method to produce FBM and is based on the scaling property of the increments [from (12.6.11)] that

$$\text{var} [\Delta B_H(t)] = |\Delta t|^{2H} \sigma_H^2 \quad (12.6.45)$$

The approach is to begin generating random sequence values at the endpoints of the interval and then successively decimate the interval and generate a random value at the midpoint of the smaller interval according to (12.6.45). Therefore, this method can be implemented recursively. To generate an FBM over the interval $[0, 1]$ with parameter H , the following steps can be used:

- Choose $B_H(0) = 0$ and select $B_H(1)$ equal to a Gaussian random number with mean 0 and variance σ_H^2 since

$$\sigma_H^2 = E\{B_H^2(1)\}$$

Clearly, $\text{var}[B_H(1) - B_H(0)] = 1^{2H}\sigma_H^2 = \sigma_H^2$.

- For the first stage, set $B_H(\frac{1}{2})$ to be the average of $B_H(0)$ and $B_H(1)$ plus some independent Gaussian number offset δ_1 with mean zero and variance $\sigma_{\delta_1}^2$, that is,

$$B_H\left(\frac{1}{2}\right) = \frac{1}{2}[B_H(1) - B_H(0)] + \delta_1 \quad (12.6.46)$$

Thus $B_H(\frac{1}{2}) - B_H(0)$ and $B_H(1) - B_H(\frac{1}{2})$ have mean 0 and variance

$$\text{var}\left[B_H\left(\frac{1}{2}\right) - B_H(0)\right] = \frac{1}{4} \text{var}[B_H(1) - B_H(0)] + \text{var}(\delta_1) \quad (12.6.47)$$

$$\left(\frac{1}{2}\right)^{2H} \sigma_H^2 = \frac{1}{4} \sigma_H^2 + \text{var}(\delta_1)$$

$$\text{or} \quad \text{var}(\delta_1) = \left[\left(\frac{1}{2}\right)^{2H} - \frac{1}{4} \left(\frac{1}{2^0}\right)^{2H}\right] \sigma_H^2 \quad (12.6.48)$$

- At the second stage, we generate $B_H(\frac{1}{4})$ and $B_H(\frac{3}{4})$, using the above method specialized to $\Delta t = \frac{1}{4}$, that is,

$$B_H\left(\frac{1}{4}\right) = \frac{1}{2} \left[B_H\left(\frac{1}{2}\right) - B_H(0) \right] + \delta_{21}$$

$$B_H\left(\frac{3}{4}\right) = \frac{1}{2} \left[B_H(1) - B_H\left(\frac{1}{2}\right) \right] + \delta_{22}$$

$$\text{with} \quad \text{var}(\delta_{21}) = \text{var}(\delta_{22}) = \left[\left(\frac{1}{2^2}\right)^{2H} - \frac{1}{4} \left(\frac{1}{2^1}\right)^{2H}\right] \sigma_H^2 \quad (12.6.49)$$

- Continuing in this fashion, at stage r we generate 2^{r-1} midpoints as the average of their respective endpoints plus a Gaussian random number offset $\delta_{r,k}$, $k = 1, 2, \dots, 2^{r-1}$, with variance

$$\text{var}(\delta_{r,k}) = \left[\left(\frac{1}{2^r}\right)^{2H} - \frac{1}{4} \left(\frac{1}{2^{r-1}}\right)^{2H}\right] \sigma_H^2 = \left(\frac{1}{2^r}\right)^{2H} (1 - 2^{2H-2}) \sigma_H^2 \quad (12.6.50)$$

$$= \frac{1}{2^{2H}} \text{var}(\delta_{r-1,k}) \quad (12.6.51)$$

Thus, as expected for an FBM, at time scale $1/2^r$ we add randomness with mean 0 and variance proportional to $(1/2^r)^{2H}$ according to (12.6.50). Also from (12.6.51) we can recursively generate the variance at each stage.

- Stop the procedure when a sufficient number of trace points are generated.

This method also suffers from a few shortcomings. The most troublesome problem is that once a given midpoint is generated, its value remains unchanged in all later stages. Thus points generated at different stages have different statistical properties in their neighborhood. This produces a visible trace that does not seem to go away even if more stages are added, and the artifact is more pronounced as $H \rightarrow 1$. A MATLAB function implementing the above steps in a recursive fashion is explored in Problem 12.15. Once again, the corresponding FGN sequence is obtained by taking a first-order difference of the generated FBM sequence.

The generation of one- and higher-dimensional FBM is a very popular subject in engineering, sciences, and computer graphics. More information and additional references can be found in Mandelbrot (1982), Maeder (1995), Peitgen et al. (1988), and Samorodnitsky and Taqqu (1994) and in the vast literature on fractals.

The estimation of the self-similarity index H or the long-memory parameter $d = H - \frac{1}{2}$ is a very difficult task. A summary of the most widely used methods, including an empirical evaluation, is provided in Taqqu et al. (1995). Additional information can be found in Beran (1994), Beran et al. (1995), and Brockwell and Davis (1991). We next present two simple methods that exploit the definition of self-similarity in the time and frequency domains (Pentland 1984; Beran 1994).

For any self-similar process $x(n)$ and any integer $\Delta > 0$, the increments $\Delta x(n) \triangleq x(n + \Delta) - x(n)$ have zero mean and satisfy the relation

$$E\{[\Delta x(n)]^2\} = C\Delta^{2H} \quad (12.6.52)$$

where C is a constant. Taking the natural logarithm of both sides, we have

$$\ln E\{[\Delta x(n)]^2\} = \ln C + 2H \ln \Delta \quad (12.6.53)$$

which can be used to estimate H using linear regression on a log-log plot. The expectation on the left side of (12.6.53) can be estimated by using the mean value of $[\Delta x(n)]^2$.

In practice, to avoid the influence of outliers, we use the quantity $E\{|\Delta x(n)|\}$, which leads to

$$\ln E\{|\Delta x(n)|\} = \ln C + H \ln \Delta \quad (12.6.54)$$

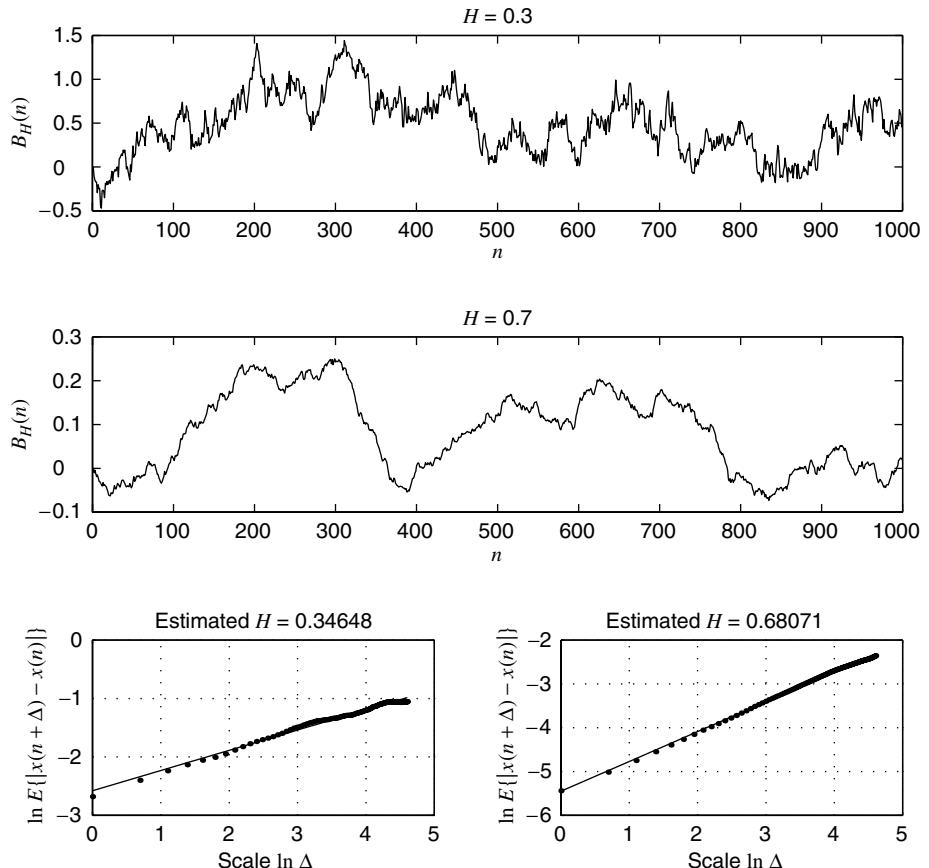
where C is a constant. The expectation in (12.6.52) is estimated by the mean absolute value, and H is determined by linear regression. This approach is illustrated in Figure 12.28, which shows the estimation of the self-similarity index H for two realizations of an FBM process (for details see Problem 12.16). We note that in practice the range of scales extends from 1 to $0.1N$, where N is the length of the used data record.

We have seen that for $|f| \rightarrow 0$, the PSD of FBM, FGN, and FPN follows a power law $1/f^\beta$, where $\beta = 2H + 1$ (FBM), $\beta = 2H - 1$ (FGN), and $\beta = 2d = 2H - 1$ (FPN). Therefore, another method for estimating the long-memory parameter H , is to compute an estimate of the PSD (see Chapter 5), and then determine H by linear regression of the logarithm of the PSD on the logarithm of the frequency. In practice, we only use the lowest 10 percent of the PSD frequencies for the linear regression because the power law relationship holds as $|f| \rightarrow 0$ (Taqqu et al. 1995). The PSD estimation of power law processes using the multitaper PSD estimation method is discussed in McCoy et al. (1998), which shows that using this method provides better estimates of long memory than the traditionally used periodogram estimator.

In practice, data are *scale-limited*: The sampling interval determines the lowest scale, and the data record length determines the highest scale. Furthermore, the scaling behavior for a certain statistical moment may change from one range of scales to another. When we try to make predictions from an adoption of a scale-invariant model, there are certain discrepancies between theory and practice. In theory, the power increases with wavelength without limit, and the variance increases with profile length without limit. In practice, the power for long wavelengths is not as large as predicted by extrapolating the power law trend observed at short wavelengths (frequency domain), and the variance does not increase without bounds as the profile length increases (spatial domain).

12.6.6 Fractional Lévy Stable Motion

If we assume that the probability density function of the stationary increments is S α S, the resulting self-similar process is known as *fractional Lévy stable motion (FLSM)*. However, unlike the FBM process, the second-order moments of the FLSM process do not exist because S α S distributions have infinite variance. The realizations of FLSM resemble more

**FIGURE 12.28**

Sample realizations of an FBM process and log-log plots for estimation of H using linear regression.

spiky versions of FBM realizations because of the heavy tails of the stable distribution. Hence, FLSM processes provide an excellent model for signals with long memory and high variability.

Formally, an FLSM process $L_{H,\alpha}(t)$ is best formulated in terms of its increment process $x_{H,\alpha}(n)$, known as *fractional Lévy stable noise (FLSN)*. The FLSN is defined by the stochastic integral (Samorodnitsky and Taqqu 1994)

$$x_{H,\alpha}(n) = L_{H,\alpha}(n+1) - L_{H,\alpha}(n) = C \int_{-\infty}^n [(n+1-s)^{H-1/\alpha} - (n-s)^{H-1/\alpha}] w_\alpha(s) ds \quad (12.6.55)$$

where C is a constant, α is the characteristic exponent of the S α S distribution, and $w_\alpha(s)$ is white noise from an S α S distribution. Notice that for $\alpha = 2$, Equation (12.6.55) provides an integral description of FGN. From Figure 12.29, which shows several realizations of FLSM for $H = 0.7$ and various values of α , it is evident that the lower the value of α , the more impulsive the process becomes. The techniques described above for generating FBM can be modified to simulate FLSM, by replacing the Gaussian random generator with the S α S one described in Chambers et al. (1976) and Samorodnitsky and Taqqu (1994).

The long-memory parameter H can be estimated by using (12.6.54) and linear regression. The PSD method cannot be used because the second-order moments of the FLSM process do not exist. The estimation of the characteristic exponent α of the S α S increments

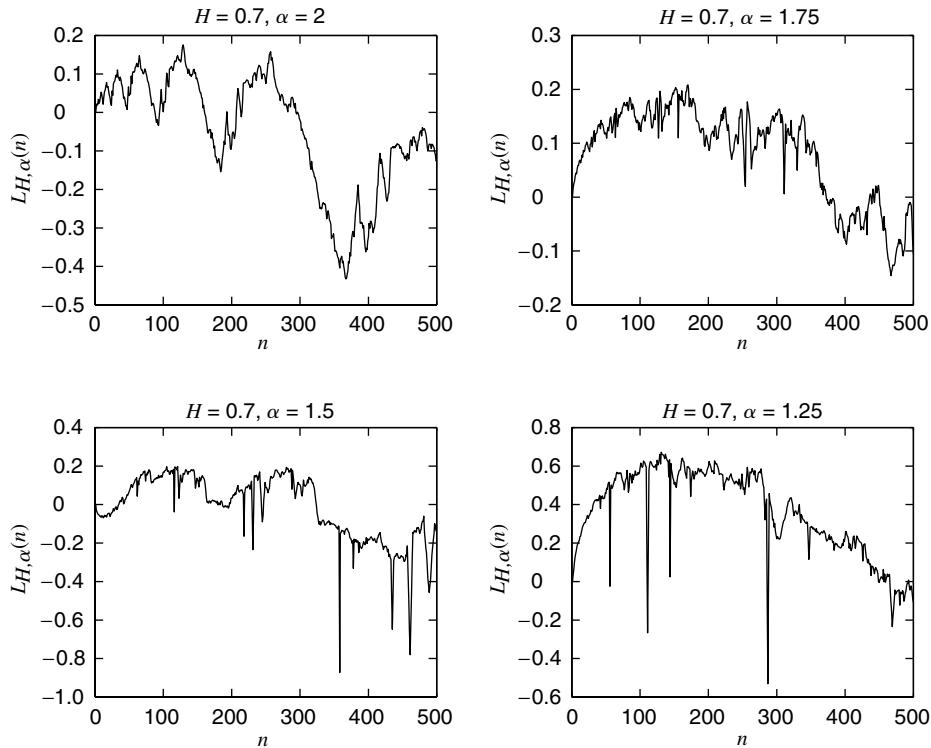


FIGURE 12.29

Sample realizations of the FLSM process for $H = 0.7$ and various values of α . The spikes increase as we go from a Gaussian ($\alpha = 2$) to a Cauchy ($\alpha = 1$) distribution.

is a very difficult task because (1) S α S distributions have infinite variance and (2) the increments are not IID owing to the long-range dependence structure. Further discussion of these topics, which are beyond the scope of this book, is provided in Adler et al. (1998), McCulloch (1986), and Koutrouvelis (1980).

Some interesting applications of FLSM to the modeling and interpolation of natural signals and images are discussed in Kogon and Manolakis (1994, 1996), Peng et al. (1993), Painter (1996), and Stuck and Kleiner (1974).

12.7 SUMMARY

In this chapter we introduced the basic concepts of three very important areas of statistical and adaptive signal processing that are the subject of extensive research. The goal was to help appreciate the limits of second-order statistical techniques, open a window to the exciting world of modern signal processing, and help the navigation through the ever-increasing literature.

In Section 12.1 we introduced the basics of higher-order statistics and pointed out the situations in which their use may be beneficial. In general, the advantages of HOS become more evident as the non-Gaussianity and nonlinearity of the underlying models increase. Also HOS is of paramount importance when we deal with non-minimum-phase systems. Concise reviews of several aspects of HOS are given in Swami (1998) and Tugnait (1998), and a comprehensive bibliography is given in Swami et al. (1997).

Section 12.2 provided a brief introduction to the principles of blind deconvolution and demonstrated that the blind deconvolution of non-minimum-phase systems requires the use

of HOS. In Sections 12.3 and 12.4 we introduced the concept of unsupervised adaptive filters, which operate without using a desired response signal; and we illustrated their application to both symbol-spaced and fractionally spaced blind equalization systems. A brief overview of current research in channel estimation and equalization is provided in Giannakis (1998). There are three types of unsupervised adaptive filtering algorithms: algorithms that use HOS either implicitly or explicitly, algorithms that use cyclostationary statistics, and algorithms that use information-theoretic concepts (Bell and Sejnowski 1995; Pham and Garrat 1997). We have focused on the widely used family of Bussgang-type algorithms that make implicit use of HOS.

In the last part of this chapter, we provided an introduction to random signal models with long memory and low or high variability. More specifically, we discussed fractional pole models with Gaussian or $S\alpha S$ IID excitations and self-similar process models with Gaussian (FBM, FGN) or $S\alpha S$ (FLSM, FLSN) stationary increments. The recent discovery that Ethernet traffic data are self-similar and $S\alpha S$ (Willinger et al. 1994) established long-memory models as a very useful tool in communication systems engineering. Finally, we note that the wavelet transform, which decomposes a signal into a superposition of scaled and shifted versions of a single basis function known as the *mother wavelet*, provides a natural tool for the analysis of linear self-similar systems and self-similar random signals. The discrete wavelet transform facilitates, to a useful degree, the whitening of self-similar processes and can be used to synthesize various types of practical self-similar random signals (Mallat 1998; Wornell 1996).

PROBLEMS

- 12.1** Prove (12.1.27), which relates the output and input fourth-order cumulants of a linear, time-invariant system.
- 12.2** (a) Derive (12.1.35) and (12.1.36).
 (b) Using the formulas for the cumulant of the sum of IID random variables, developed in Section 3.2.4, determine $\kappa_y^{(4)}$ and compare with the result obtained in (a).
- 12.3** If $x(n)$ is a stationary Gaussian process, show that $E\{x^2(n)x^2(n-l)\} = \rho_x^2(l)$ and explain how it can be used to investigate the presence of nonlinearities.
- 12.4** In this problem we use an MA(2) model to explore some properties of cumulants and bispectra.
 (a) Write a MATLAB function `k=cuma(b)` that computes the cumulant $\kappa_x^{(3)}(l_1, l_2)$ of the MA(2) model $H(z) = b_0 + b_1 z^{-1} + b_2 z^{-2}$ for $-L \leq l_1, l_2 \leq L$.
 (b) Use the functions `k=cuma(b)`, `X=fft(x)`, and `X=shiftfft(X)` to compute the bispectra of the three MA(2) models in Table 12.1. Plot your results and compare with those in Figure 12.2.
 (c) Compute the bispectra of the models using the formula
- $$R_x^{(3)}(e^{j\omega_1}, e^{j\omega_2}) = \kappa_w^{(3)} H(e^{j\omega_1}) H(e^{j\omega_2}) H^*(e^{j(\omega_1+\omega_2)})$$
- for $\omega_1 = \omega_2 = 2\pi k/N$, $0 \leq k \leq N-1$. Compare with the results in part b and Figure 12.2.
 (d) Show that the bispectrum can be computed in MATLAB using the following segment of code:
- ```

H=freqz(h,1,N,'whole') ;
Hc=conj(H) ;
R3x=(H'*H').*hankel(Hc,Hc([N,1:N-1])) ;
R3x=shiftfft(R3x) ;

```
- 12.5** Using the minimum-, maximum-, and mixed-phase systems discussed in Example 12.1.1, write a MATLAB program to reproduce the results shown in Figures 12.3 and 12.4. Use  $a = 0.4$ ,  $b = 0.8$ , and  $N = 300$  samples.

- 12.6** Use the Levinson-Durbin algorithm, developed in Chapter 7, to derive expressions (12.5.20), direct-form coefficients, and (12.5.21) for the lattice parameters of the fractional pole model.

743

---

PROBLEMS

- 12.7** Consider the FPZ( $1, d, 0$ ) model

$$H_{\text{fpz}}(z) = \frac{1}{(1 - z^{-1})^d} \frac{1}{(1 + az^{-1})}$$

where  $-\frac{1}{2} < d < \frac{1}{2}$  and  $-1 < a < 1$ . Compute and plot the impulse response, autocorrelation, and spectrum for  $a = \pm 0.9$  and  $d = \pm 0.2, \pm 0.4$ . Identify which models have long memory and which have short memory.

- 12.8** Compute and plot the PSD of the FGN process, using the following approaches, and compare the results.

- (a) The definition  $R_x(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r_x(l)e^{-j\omega l}$  and formula (12.6.36) for the autocorrelation.
- (b) The theoretical formula (12.6.37).

- 12.9** Use the algorithm of Schür to develop a more efficient implementation of the fractional pole noise generation method described by Equations (12.5.24) to (12.5.28).

- 12.10** In this problem we study the properties of the harmonic fractional unit-pole model specified by the system function given by (12.5.32). The impulse response is given by (Gray et al. 1989)

$$h_{\theta,d}(n) = \sum_{k=0}^{\lfloor n/2 \rfloor} \frac{(-1)^k \Gamma(d+n-k)(2 \cos \theta)^{n-2k}}{k!(n-2k)!\Gamma(d)}$$

where  $\Gamma(\cdot)$  is the gamma function.

- (a) Compute and plot  $h_{\theta,d}(n)$  for various values of  $\theta$  and  $d$ .
- (b) Demonstrate the validity of the above formula by evaluating  $h_{\theta,d}(n)$  from  $H_{\theta,d}(z)$  for the same values of  $\theta$  and  $d$ .
- (c) Illustrate that the model is minimum-phase if  $|\cos \theta| < 1$  and  $-\frac{1}{2} < d < \frac{1}{2}$  or  $\cos \theta = \pm 1$  and  $-\frac{1}{4} < d < \frac{1}{4}$ .
- (d) Illustrate that the harmonic minimum-phase model, like the FPZ( $0, d, 0$ ) one, exhibits long-memory behavior only for positive values of  $d$ .
- (e) Show that for  $0 < d < \frac{1}{4}$  and  $\cos \theta = 1$ , the autocorrelation equals that of the FPZ( $0, 2d, 0$ ) model [multiplied by  $(-1)^l$  if  $\cos \theta = -1$ ]. When  $|\cos \theta| < 1$  and  $0 < d < \frac{1}{2}$ , illustrate numerically that the autocorrelation can be approximated by  $\rho(l) \sim -l^{2d-1} \sin(\theta l - \pi d)$  as  $l \rightarrow \infty$ .
- (f) Compute and plot the spectrum of the model for  $\theta = \pi/3$  and various values of  $d$ .
- (g) Generate and plot realizations of Gaussian HFPZ noise for  $\theta = \pi/6$  and  $d = -0.3, 0.1$ , and 0.4.

- 12.11** Determine the variogram of the process  $x(n)$  obtained by exciting the system

$$H(z) = \frac{1}{(1 - z^{-1})(1 - az^{-1})} \quad |a| < 1$$

with white noise  $w(n) \sim \text{WGN}(0, \sigma_w^2)$ .

- 12.12** Following the steps leading to (12.6.26), show that the fractal (Haussdorff) dimension  $D$  is related to the Hurst exponent  $H$  by

$$D = 2 - H$$

- 12.13** Develop a MATLAB function to generate the ordinary Brownian motion trace according to the steps given for the cumulative sum method in Section 12.6.3. The format of the function should be `x = obm_cumsum(N)`.

- (a) Generate 16,384 samples of the Brownian motion  $x(t)$  over  $0 \leq t \leq 1$ .
- (b) Investigate the self-affine property of  $x(t)$  by reproducing a figure similar to Figure 12.23.

- 12.14** Develop a MATLAB function to generate the fractional Brownian motion trace according to the steps given for the spectral synthesis method in Section 12.6.3. The format of the function should be `x = fbm_spectral(H, N)`.
- Generate 1024 samples of the FBM  $B_H(t)$  over  $0 \leq t \leq 1$  for  $H = 0.3$ . Investigate the self-affine property of  $B_{0.3}(t)$ .
  - Generate 1024 samples of the FBM  $B_H(t)$  over  $0 \leq t \leq 1$  for  $H = 0.7$ . Investigate the self-affine property of  $B_{0.7}(t)$ .
- 12.15** Develop a MATLAB function to generate the fractional Brownian motion trace according to the steps given for the random midpoint replacement method in Section 12.6.3. The format of the function should be `x = fbm_replace(N)`.
- Generate 1024 samples of the FBM  $B_H(t)$  over  $0 \leq t \leq 1$  for  $H = 0.5$ . Compare visually  $B_{0.5}(t)$  with that obtained by using the cumulative-sum method. Comment on your observations.
  - Generate 1024 samples of the FBM  $B_H(t)$  over  $0 \leq t \leq 1$  for  $H = 0.99$ . Investigate the artifact discussed in the chapter for  $H \rightarrow 1$ .
- 12.16** Based on Equation (12.6.54), develop a MATLAB function `[H, sigmaH] = est_H_mad(x)` that computes an estimate of the self-similarity index  $H$  and the variance  $\sigma_H^2$  of an FBM process.
- Use function `x = fbm_replace(N)` to generate  $N = 1024$  samples of an FBM process with  $H = 0.3$ , and use the function `[H, sigmaH] = est_H_mad(x)` to estimate  $H$  and  $\sigma_H$ .
  - Repeat the previous task for  $H = 0.7$ .
  - Perform a Monte Carlo simulation using 100 trials and compute the mean and standard deviation of the estimates for  $H$  and  $\sigma_H$  in (a) and (b).
- 12.17** Repeat Problem 12.16 by developing a function that estimates the self-similarity index  $H$  by determining the slope of the first 10 percent values of the periodogram in a log-log plot.

## Matrix Inversion Lemma

The matrix inversion lemma is a useful formula that is employed extensively in signal processing. The purpose of this formula is to express the inverse of a matrix in terms of the inverse of one of its additive components, so as to facilitate an efficient computation of the inverse. To motivate this lemma, consider the inverse of the following scalar quantity

$$(a + xy)^{-1} = \frac{1}{a + xy} \quad a + xy \neq 0, a \neq 0$$

in terms of the inverse of  $a$ . Since  $a + xy \neq 0$  and  $a \neq 0$ , we also have

$$|xya^{-1}| \neq 1 \quad \text{and} \quad |ya^{-1}x| \neq 1 \quad (\text{A.1})$$

Using the convergence of the geometric series formula

$$1 - xya^{-1} + (xya^{-1})^2 - \dots = \frac{1}{1 + xya^{-1}} \quad |xya^{-1}| \neq 1 \quad (\text{A.2})$$

we obtain

$$\begin{aligned} \frac{1}{a + xy} &= \frac{a^{-1}}{1 + xya^{-1}} \\ &= a^{-1}[1 - xya^{-1} + (xya^{-1})^2 - \dots] \\ &= a^{-1} - a^{-1}xya^{-1} + a^{-1}x(ya^{-1}x)ya^{-1} - a^{-1}x(ya^{-1}x)^2ya^{-1} + \dots \\ &= a^{-1} - a^{-1}xya^{-1}[1 - ya^{-1}x + (ya^{-1}x)^2 - \dots] \\ &= a^{-1} - \frac{a^{-1}xya^{-1}}{1 + ya^{-1}x} \quad |ya^{-1}x| \neq 1 \end{aligned} \quad (\text{A.3})$$

which is the desired result. We begin with a special case of the lemma in which  $a$  is a matrix and  $x$  and  $y$  are vectors. This result then can be generalized to the case in which  $x$  and  $y$  are also matrices.

**LEMMA A.1 (SHERMAN-MORRISON'S FORMULA).** Let  $\mathbf{A}$  be an  $N \times N$  invertible matrix and let  $\mathbf{x}$  and  $\mathbf{y}$  be two  $N \times 1$  vectors such that  $(\mathbf{A} + \mathbf{xy}^H)$  is invertible. Then we have

$$(\mathbf{A} + \mathbf{xy}^H)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{xy}^H\mathbf{A}^{-1}}{1 + \mathbf{y}^H\mathbf{A}^{-1}\mathbf{x}} \quad (\text{A.4})$$

**Proof.** Consider

$$\mathbf{A} + \mathbf{xy}^H = \mathbf{A}(\mathbf{I} + \mathbf{A}^{-1}\mathbf{xy}^H)$$

$$\text{Hence } (\mathbf{A} + \mathbf{xy}^H)^{-1} = (\mathbf{I} + \mathbf{A}^{-1}\mathbf{xy}^H)^{-1}\mathbf{A}^{-1} \quad (\text{A.5})$$

Using the result that if the matrix  $(\mathbf{I} + \mathbf{A})$  is invertible, then  $(\mathbf{I} + \mathbf{A})^{-1} = \mathbf{I} - \mathbf{A} + \mathbf{A}^2 - \mathbf{A}^3 + \dots$ , we obtain

$$\begin{aligned} (\mathbf{I} + \mathbf{A}^{-1}\mathbf{x}\mathbf{y}^H)^{-1} &= \mathbf{I} - \mathbf{A}^{-1}\mathbf{x}\mathbf{y}^H + (\mathbf{A}^{-1}\mathbf{x}\mathbf{y}^H)^2 - (\mathbf{A}^{-1}\mathbf{x}\mathbf{y}^H)^3 + \dots \\ &= \mathbf{I} - \mathbf{A}^{-1}\mathbf{x}\mathbf{y}^H + \mathbf{A}^{-1}\mathbf{x}\mathbf{y}^H\mathbf{A}^{-1}\mathbf{x}\mathbf{y}^H - \dots \end{aligned} \quad (\text{A.6})$$

since from (A.5)  $(\mathbf{I} + \mathbf{A}^{-1}\mathbf{x}\mathbf{y}^H)$  is invertible. Substituting (A.6) into (A.5), we obtain

$$\begin{aligned} (\mathbf{A} + \mathbf{x}\mathbf{y}^H)^{-1} &= \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{x}\mathbf{y}^H\mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{x} \underbrace{(\mathbf{y}^H\mathbf{A}^{-1}\mathbf{x})}_{\text{scalar}} \mathbf{y}^H\mathbf{A}^{-1} - \dots \\ &= \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{x}\mathbf{y}^H\mathbf{A}^{-1}[1 - \mathbf{y}^H\mathbf{A}^{-1}\mathbf{x} + (\mathbf{y}^H\mathbf{A}^{-1}\mathbf{x})^2 - \dots] \\ &= \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{x}\mathbf{y}^H\mathbf{A}^{-1}}{1 + \mathbf{y}^H\mathbf{A}^{-1}\mathbf{x}} \end{aligned}$$

since the scalar  $\mathbf{y}^H\mathbf{A}^{-1}\mathbf{x} \neq 1$  due to the invertibility of  $(\mathbf{I} + \mathbf{A}^{-1}\mathbf{x}\mathbf{y}^H)$  [see also (A.1)]. This completes the proof.

The generalization of (A.4), known as Woodbury's formula, is given by

$$(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1} \quad (\text{A.7})$$

If matrix  $\mathbf{A}$  is partitioned as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \quad (\text{A.8})$$

then (A.7) can be used in determining inverses of submatrices contained in

$$\mathbf{A}^{-1} = \begin{bmatrix} (\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21})^{-1} & -(\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21})^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} \\ -(\mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12})^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & (\mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12})^{-1} \end{bmatrix} \quad (\text{A.9})$$

where inverses  $\mathbf{A}_{11}^{-1}$  and  $\mathbf{A}_{22}^{-1}$  are assumed to exist.

---

# Gradients and Optimization in Complex Space

In the development of many signal processing algorithms, it is necessary to compute the gradient of a real or complex function with respect to a complex vector  $\mathbf{w}$ . The concepts involved in this gradient operation and the application of the gradient in optimization are described in this section. For more details see Gill et al. (1981), Kay (1993), and Luenberger (1984).

## B.1 GRADIENT

We begin with a simplest case. Let  $g(\mathbf{x})$  be a real scalar function of real parameter vector  $\mathbf{x}$ . Then we define the gradient of  $g(\mathbf{x})$  with respect to vector  $\mathbf{x}$  as a column vector

$$\nabla_{\mathbf{x}}(g) \triangleq \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} = \left[ \frac{\partial g(\mathbf{x})}{\partial x_1} \quad \frac{\partial g(\mathbf{x})}{\partial x_2} \quad \dots \quad \frac{\partial g(\mathbf{x})}{\partial x_N} \right]^T \quad (\text{B.1})$$

This definition extends to a vector function  $\mathbf{g}(\mathbf{x})$  of parameter vector  $\mathbf{x}$  as

$$\nabla_{\mathbf{x}}(\mathbf{g}) \triangleq \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial g_1(\mathbf{x})}{\partial \mathbf{x}} \\ \frac{\partial g_2(\mathbf{x})}{\partial \mathbf{x}} \\ \vdots \\ \frac{\partial g_M(\mathbf{x})}{\partial \mathbf{x}} \end{bmatrix}^T = \begin{bmatrix} \frac{\partial g_1(\mathbf{x})}{\partial x_1} & \frac{\partial g_2(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial g_M(\mathbf{x})}{\partial x_1} \\ \frac{\partial g_1(\mathbf{x})}{\partial x_2} & \frac{\partial g_2(\mathbf{x})}{\partial x_2} & \dots & \frac{\partial g_M(\mathbf{x})}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_1(\mathbf{x})}{\partial x_N} & \frac{\partial g_2(\mathbf{x})}{\partial x_N} & \dots & \frac{\partial g_M(\mathbf{x})}{\partial x_N} \end{bmatrix} \quad (\text{B.2})$$

Thus  $\nabla_{\mathbf{x}}(\mathbf{g})$  is an  $N \times M$  matrix. Finally, consider a scalar function  $g(\mathbf{A})$  of an  $M \times N$  matrix  $\mathbf{A}$ . We define the gradient of  $g(\mathbf{A})$  with respect to  $\mathbf{A}$  as a matrix

$$\nabla_{\mathbf{A}}(g) \triangleq \frac{\partial g(\mathbf{A})}{\partial \mathbf{A}} = \begin{bmatrix} \frac{\partial g(\mathbf{A})}{\partial a_{11}} & \frac{\partial g(\mathbf{A})}{\partial a_{12}} & \dots & \frac{\partial g(\mathbf{A})}{\partial a_{1N}} \\ \frac{\partial g(\mathbf{A})}{\partial a_{21}} & \frac{\partial g(\mathbf{A})}{\partial a_{22}} & \dots & \frac{\partial g(\mathbf{A})}{\partial a_{2N}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g(\mathbf{A})}{\partial a_{M1}} & \frac{\partial g(\mathbf{A})}{\partial a_{M2}} & \dots & \frac{\partial g(\mathbf{A})}{\partial a_{MN}} \end{bmatrix} \quad (\text{B.3})$$

Using these definitions, we see it is easy to prove the following results:

$$\nabla_{\mathbf{x}}(\mathbf{y}^T \mathbf{A} \mathbf{x}) = \mathbf{A}^T \mathbf{y} \quad (\text{B.4})$$

$$\nabla_{\mathbf{x}}(\mathbf{x}^T \mathbf{A} \mathbf{y}) = \mathbf{A} \mathbf{y} \quad (\text{B.5})$$

$$\nabla_{\mathbf{x}}(\mathbf{x}^T \mathbf{A} \mathbf{x}) = (\mathbf{A} + \mathbf{A}^T) \mathbf{x} \quad (\text{B.6})$$

$$\nabla_{\mathbf{A}}(\mathbf{x}^T \mathbf{A} \mathbf{y}) = \mathbf{x} \mathbf{y}^T \quad (\text{B.7})$$

$$\nabla_{\mathbf{A}}(\mathbf{x}^T \mathbf{A} \mathbf{x}) = \mathbf{x} \mathbf{x}^T \quad (\text{B.8})$$

Now we consider the case of a complex-valued scalar function  $g(z, z^*)$  of a complex variable  $z$  and its complex conjugate  $z^*$ . We assume that the function is analytic with respect to  $z$  and  $z^*$  independently<sup>†</sup> (in the sense of partial differentiation). An example of such a function is

$$g(z, z^*) = a|z|^2 + bz^* + c = azz^* + bz^* + c \quad (\text{B.9})$$

Let  $f(x, y)$  be the complex function of the real and imaginary parts  $x$  and  $y$  of the variable  $z = x + jy$ , such that  $g(z, z^*) = f(x, y)$ . Again consider the function in (B.9), then

$$f(x, y) = a(x^2 + y^2) + b(x - jy) + c \quad (\text{B.10})$$

$$= a|z|^2 + bz^* + c = g(z, z^*) \quad (\text{B.11})$$

The partial derivative of  $g(z, z^*)$  with respect to  $z$  (keeping  $z^*$  as a constant) is given by

$$\frac{\partial}{\partial z} g(z, z^*) = \frac{1}{2} \left[ \frac{\partial}{\partial x} f(x, y) - j \frac{\partial}{\partial y} f(x, y) \right] \quad (\text{B.12})$$

Similarly, the partial derivative of  $g(z, z^*)$  with respect to  $z^*$  (keeping  $z$  as a constant) is given by

$$\frac{\partial}{\partial z^*} g(z, z^*) = \frac{1}{2} \left[ \frac{\partial}{\partial x} f(x, y) + j \frac{\partial}{\partial y} f(x, y) \right] \quad (\text{B.13})$$

These results can be easily verified for  $g(z, z^*)$  in (B.9):

$$\frac{\partial}{\partial z} a|z|^2 + bz^* + c = \frac{\partial}{\partial z} [azz^* + bz^* + c] = az^* = a(x - jy)$$

$$\begin{aligned} \text{and } \frac{1}{2} \left[ \frac{\partial}{\partial x} f(x, y) - j \frac{\partial}{\partial y} f(x, y) \right] &= \frac{1}{2} \left\{ \frac{\partial}{\partial x} [a(x^2 + y^2) + b(x - jy) + c] \right. \\ &\quad \left. - j \frac{\partial}{\partial y} [a(x^2 + y^2) + b(x - jy) + c] \right\} \\ &= ax + \frac{b}{2} - jy - \frac{b}{2} = a(x - jy) = az^* \end{aligned}$$

Let  $f(\mathbf{x})$  be a real-valued scalar function of the complex vector  $\mathbf{x}$  expressed as

$$f(\mathbf{x}) = g(\mathbf{x}, \mathbf{x}^*) \quad (\text{B.14})$$

where  $g(\cdot)$  is a real-valued function of  $\mathbf{x}$  and  $\mathbf{x}^*$ , analytic with respect to  $\mathbf{x}$  and  $\mathbf{x}^*$  independently (in the sense of partial differentiation). The necessary and sufficient condition to obtain an equilibrium (optimum) point of  $f(\mathbf{x})$  is that

$$\nabla_{\mathbf{x}}(g) = \nabla_{\mathbf{x}^*}(g) = \mathbf{0} \quad (\text{B.15})$$

The necessary gradient  $\nabla_{\mathbf{x}}(g)$  can be computed by using (B.13). In particular, for any complex vector  $\mathbf{y}$ ,  $\mathbf{x}$ , and matrix  $\mathbf{A}$ , we have

$$\nabla_{\mathbf{x}^*}(\mathbf{x}^H \mathbf{y}) = \mathbf{y} \quad (\text{B.16})$$

$$\nabla_{\mathbf{x}^*}(\mathbf{y}^H \mathbf{x}) = \mathbf{0} \quad (\text{B.17})$$

$$\nabla_{\mathbf{x}^*}(\mathbf{x}^H \mathbf{A} \mathbf{y}) = \mathbf{A} \mathbf{y} \quad (\text{B.18})$$

$$\nabla_{\mathbf{x}^*}(\mathbf{x}^H \mathbf{A} \mathbf{x}) = \mathbf{A} \mathbf{x} \quad (\text{B.19})$$

---

<sup>†</sup>In this approach, the quantities  $z$  and  $z^*$  are considered to be independent of each other. Clearly they are not, since  $z$  is uniquely determined by its conjugate. Nevertheless, this technique works.

$$\nabla_{\mathbf{x}}(\mathbf{x}^H \mathbf{A} \mathbf{x}) = \mathbf{x}^H \mathbf{A} \quad (B.20)$$

$$\nabla_{\mathbf{A}}(\mathbf{x}^H \mathbf{A} \mathbf{y}) = \mathbf{x}^* \mathbf{y}^T \quad (B.21)$$

$$\nabla_{\mathbf{A}}(\mathbf{x}^H \mathbf{A} \mathbf{x}) = \mathbf{x}^* \mathbf{x}^T \quad (B.22)$$

## B.2 LAGRANGE MULTIPLIERS

The procedure of using *Lagrange multipliers* is an elegant technique of obtaining optimum values of a function of several variables subject to one or more constraints. Suppose we want to determine the minimum of a function  $f(\mathbf{x})$  of  $N$  variables  $\mathbf{x} = [x_1, \dots, x_N]$ , subject to a constraint relating  $x_1$  through  $x_N$  given in the form

$$g(\mathbf{x}) = 0 \quad (B.23)$$

One straightforward approach would be to solve (B.23) for one of the variables, say  $x_i$ , in terms of the remaining ones and then eliminate  $x_i$  from  $f(\mathbf{x})$ . The minimization of  $f(\mathbf{x})$  can then be carried out in a usual way to determine the minimum point in the  $N$ -dimensional space. In practice, this approach is all but impossible to carry out, especially if  $f(\mathbf{x})$  is highly nonlinear.

A simpler yet elegant approach is to introduce an additional parameter  $\lambda$ , called a *Lagrange multiplier*.<sup>†</sup> To motivate this technique through a geometric viewpoint, consider a two-dimensional function

$$f(x_1, x_2) = x_1^2 + x_2^2 \quad (B.24)$$

which is a bowl-shaped surface whose minimum is at the origin  $x_1 = x_2 = 0$ . Thus minimizing  $f(\mathbf{x})$  is the same as minimizing the length of vector  $\mathbf{x}$ . If there is no constraint, the zero vector is the best  $\mathbf{x}$ . Now let the constraint be a line

$$x_2 = -\frac{1}{2}x_1 + \frac{5}{2} \quad (B.25)$$

in the  $(x_1, x_2)$  plane. Thus

$$g(\mathbf{x}) = x_1 + 2x_2 - 5 = 0 \quad (B.26)$$

This constraint and the bowl-shaped surface are shown in Figure B.1. The constraint plane cuts through the bowl, creating a parabolic edge, as shown in the figure. Since the point  $\mathbf{x}$  is restricted to the constraint line (B.26), the minimization function  $f(\mathbf{x})$  is constrained to the parabolic edge. Thus the minimization of (B.24) becomes a problem of finding the point on the parabolic curve that is nearest to the origin. This is also the point on the constraint line that is nearest to the origin and is obtained by drawing a perpendicular ray, as shown in Figure B.1. This point is  $x_1 = 1$  and  $x_2 = 2$ . At this point the parabolic edge achieves its minimum.

How is all this related to the Lagrange multiplier? Referring to Figure B.1, we see at any point  $P$  on the constraint surface, the gradient of  $f(\mathbf{x})$  is given by vector  $\nabla f$ . To find the minimum point of  $f(\mathbf{x})$  within the constraint surface, we have to find the component  $\nabla_{\parallel} f$  of  $\nabla f$  that lies in the surface and to set it equal to zero, that is,

$$\nabla_{\parallel} f = 0 \quad (B.27)$$

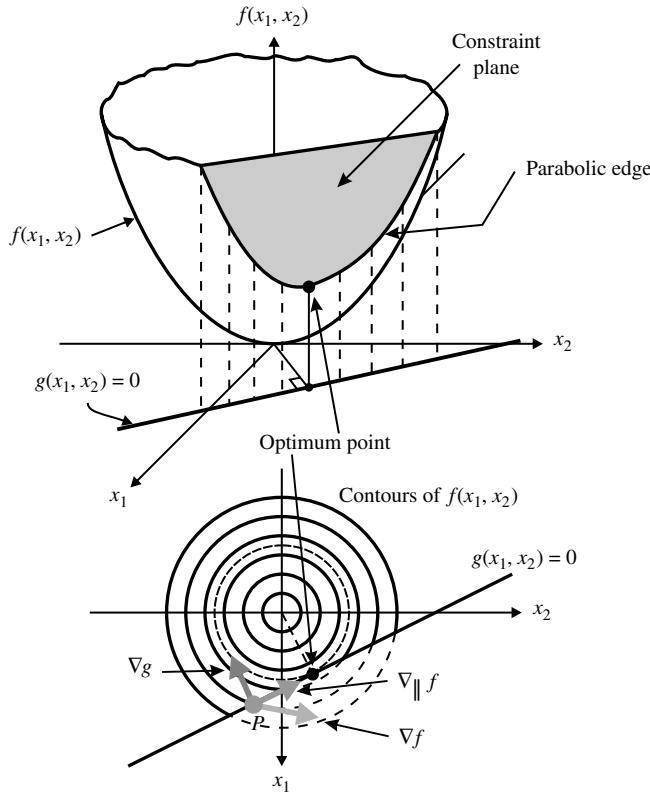
Consider the constraint function  $g(\mathbf{x})$  and perturb  $\mathbf{x}$  to  $\mathbf{x} + \delta\mathbf{x}$  within the surface. Then using the Taylor expansion, we can write

$$g(\mathbf{x} + \delta\mathbf{x}) = g(\mathbf{x}) + \delta\mathbf{x}^T \nabla g(\mathbf{x}) = g(\mathbf{x}) \quad (B.28)$$

since  $\mathbf{x} + \delta\mathbf{x}$  is chosen to lie within the surface  $g(\mathbf{x}) = 0$ . This implies that  $\nabla g(\mathbf{x}) = 0$ , which means that the gradient  $\nabla g(\mathbf{x})$  is normal to the constraint surface. As shown in

---

<sup>†</sup>Although we have reserved  $\lambda$  for eigenvalues, we will follow the tradition and use  $\lambda$  also as a Lagrange multiplier.



**FIGURE B.1**  
Geometric interpretation of Lagrange multiplier.

Figure B.1, we can now obtain the component  $\nabla_{\parallel} f$  by adding a suitable scaled vector  $\nabla g(\mathbf{x})$  to the gradient in the form

$$\nabla_{\parallel} f = \nabla f + \lambda \nabla g(\mathbf{x}) \quad (\text{B.29})$$

where  $\lambda$  is a Lagrange multiplier. Using linearity of the gradient operator, we introduce the *Lagrangian function*

$$\mathcal{L}(\mathbf{x}, \lambda) \triangleq f(\mathbf{x}) + \lambda g(\mathbf{x}) \quad (\text{B.30})$$

so that the gradient  $\nabla \mathcal{L}$  is given by (B.29).

Therefore, to find the minimum of  $f(\mathbf{x})$  subject to  $g(\mathbf{x}) = 0$ , we first define the Lagrangian (B.30) and then find the minimum point of  $\mathcal{L}(\mathbf{x}, \lambda)$  by differentiating it with respect to both  $\mathbf{x}$  and  $\lambda$ . This results in  $N + 1$  equations that can be solved to determine the optimum  $\mathbf{x}_o$  and  $\lambda_o$  from which the minimum  $f(\mathbf{x}_o)$  can be found. Note that  $\partial \mathcal{L} / \partial \lambda = 0$  leads to the constraint  $g(\mathbf{x}) = 0$ . Thus Lagrange multiplier technique leads to the equations for a constrained minimum, and it does not require us to solve for  $g(\mathbf{x}) = 0$ .

This technique can be extended to more than one, say  $K$ , constraints simply by using one Lagrange multiplier  $\lambda_k$  for each of the constraints  $g_k(\mathbf{x}) = 0$ ,  $k = 1, \dots, K$ , and constructing a Lagrangian function of the form

$$\mathcal{L}(\mathbf{x}, \lambda_1, \dots, \lambda_K) = f(\mathbf{x}) + \sum_{k=1}^K \lambda_k g_k(\mathbf{x}) \quad (\text{B.31})$$

This Lagrangian is then minimized with respect to  $\mathbf{x}$  and  $\{\lambda_k\}_1^K$ .

**EXAMPLE B.1.** Consider the problem of fitting the largest (areawise) rectangle inside an ellipse given by

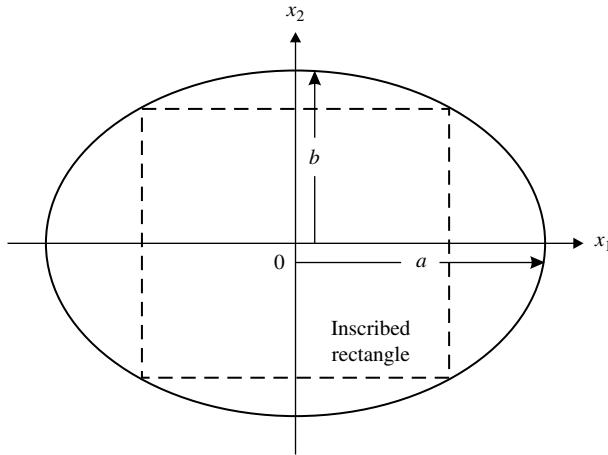
$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = 1 \quad (\text{B.32})$$

The ellipse and an inscribed rectangle are shown in Figure B.2. Thus the objective function that we want to maximize is

$$f(x_1, x_2) = (2x_1)(2x_2) = 4x_1x_2 \quad (\text{B.33})$$

subject to the constraint

$$g(x_1, x_2) = \frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} - 1 \quad (\text{B.34})$$



**FIGURE B.2**  
Ellipse and the inscribed rectangle in Example B.1.

**Method 1.** Solving (B.34) for  $x_2$ , we obtain

$$x_2 = \pm \frac{b}{a} \sqrt{a^2 - x_1^2} \quad (\text{B.35})$$

Since the area is positive, choosing the plus sign and substituting in (B.33), we have

$$f(x_1, x_2) = 4 \frac{b}{a} x_1 \sqrt{a^2 - x_1^2} \quad (\text{B.36})$$

which is a function of  $x_1$  alone. Now to obtain the maximum value of  $f(x_1, x_2)$ , we set

$$\frac{df}{dx_1} = 0 = 4 \frac{b}{a} \left( \sqrt{a^2 - x_1^2} - \frac{x_1^2}{\sqrt{a^2 - x_1^2}} \right) \quad (\text{B.37})$$

Thus from (B.37) we get the optimum value of  $x_1$  and subsequently from (B.35) the optimum value for  $x_2$

$$x_{1,o} = \frac{a}{\sqrt{2}} \quad \text{and} \quad x_{2,o} = \frac{b}{\sqrt{2}} \quad (\text{B.38})$$

**Method 2.** Let us form the Lagrangian

$$\mathcal{L}(x_1, x_2, \lambda) = f(x_1, x_2) + \lambda g(x_1, x_2) = 4x_1x_2 + \lambda \left( \frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} - 1 \right) \quad (\text{B.39})$$

Now to find the optimum point, we set

$$\frac{\partial \mathcal{L}}{\partial x_1} = 0 = 4x_2 + \lambda \frac{2x_1}{a^2} \quad (\text{B.40})$$

$$\frac{\partial \mathcal{L}}{\partial x_2} = 0 = 4x_1 + \lambda \frac{2x_2}{b^2} \quad (\text{B.41})$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} - 1 \quad (\text{B.42})$$

Solving (B.40) through (B.42), we obtain the optimum values

$$x_{1,o} = \frac{a}{\sqrt{2}} \quad x_{2,o} = \frac{b}{\sqrt{2}} \quad \lambda_o = -2ab \quad (\text{B.43})$$

Clearly, the second method is more convenient.

**EXAMPLE B.2.** Let a real-valued random vector  $\mathbf{y}$  be given by

$$\mathbf{y} = \alpha \mathbf{x} + \mathbf{v} \quad (\text{B.44})$$

where  $\mathbf{x}$  is a deterministic vector,  $\alpha$  is a constant, and  $\mathbf{v}$  is a zero-mean random vector with covariance matrix  $\mathbf{R}_v$ . We want to determine a *best linear unbiased estimator (BLUE)* of  $\alpha$ , given  $\mathbf{y}$ . Let

$$\hat{\alpha} = \mathbf{h}^T \mathbf{y} \quad (\text{B.45})$$

Since the estimator must be unbiased, we have

$$\alpha = E\{\hat{\alpha}\} = E\{\mathbf{h}^T \mathbf{y}\} = E\{\mathbf{h}^T(\alpha \mathbf{x} + \mathbf{v})\} = \alpha E\{\mathbf{h}^T \mathbf{x}\} = \alpha \mathbf{h}^T \mathbf{x} \quad (\text{B.46})$$

which implies that  $\mathbf{h}^T \mathbf{x} = 1$ . Hence the constraint  $g(\mathbf{h})$  is

$$g(\mathbf{h}) = \mathbf{h}^T \mathbf{x} - 1 \quad (\text{B.47})$$

Next we want to minimize the variance in the estimation

$$\text{var}(\hat{\alpha}) = \text{var}(\mathbf{h}^T \mathbf{y}) = \text{var}(\mathbf{h}^T \mathbf{v}) = \mathbf{h}^T \mathbf{R}_v \mathbf{h} \quad (\text{B.48})$$

Now to obtain the BLUE of  $\alpha$ , consider the Lagrangian

$$\mathcal{L}(\mathbf{h}, \lambda) = \mathbf{h}^T \mathbf{R}_v \mathbf{h} + \lambda(\mathbf{h}^T \mathbf{x} - 1) \quad (\text{B.49})$$

Using (B.5) and (B.6), we obtain

$$\nabla_{\mathbf{h}}(\mathcal{L}) = 2\mathbf{h}_o^T \mathbf{R}_v + \lambda \mathbf{x}^T = \mathbf{0}^T \quad (\text{B.50})$$

$$\text{or} \quad \mathbf{h}_o = \frac{-\lambda}{2} \mathbf{R}_v^{-1} \mathbf{x} \quad (\text{B.51})$$

Substituting (B.51) into (B.47) and solving for  $\lambda$ , we obtain

$$\lambda = -\frac{2}{\mathbf{x}^T \mathbf{R}_v \mathbf{x}}$$

Finally, the optimum estimator becomes

$$\mathbf{h}_o = \frac{\mathbf{R}_v^{-1} \mathbf{x}}{\mathbf{x}^T \mathbf{R}_v \mathbf{x}} \quad (\text{B.52})$$

which can be recognized as a whitening filter and a matched filter.

**EXAMPLE B.3.** Consider a complex-valued case of the above example. We want to minimize

$$f(\mathbf{h}) = \mathbf{h}^H \mathbf{R}_v \mathbf{h} \quad (\text{B.53})$$

where  $\mathbf{R}_v$  is a real-valued symmetric matrix so that  $f(\mathbf{h})$  is real, subject to

$$\text{Re}\{\mathbf{h}^H \mathbf{x}\} = b \quad (\text{B.54})$$

Consider  $f(\mathbf{h})$  and the constraint function  $g(\mathbf{h})$  as

$$\begin{aligned} f(\mathbf{h}, \mathbf{h}^H) &= \mathbf{h}^H \mathbf{R}_v \mathbf{h} \\ g(\mathbf{h}, \mathbf{h}^H) &= \mathbf{h}^H \mathbf{x} + \mathbf{x}^H \mathbf{h} - 2b \end{aligned} \quad (\text{B.55})$$

Thus the Lagrangian is

$$\mathcal{L}(\mathbf{h}, \mathbf{h}^H, \lambda) = \mathbf{h}^H \mathbf{R}_v \mathbf{h} - \lambda(\mathbf{h}^H \mathbf{x} + \mathbf{x}^H \mathbf{h} - 2b) \quad (\text{B.56})$$

Now using (B.20), we get

$$\nabla_{\mathbf{h}}(\mathcal{L}) = \mathbf{h}_o^H \mathbf{R}_v - \lambda \mathbf{x}^H = \mathbf{0}^T \Rightarrow \mathbf{h}_o = \lambda \mathbf{R}_v^{-1} \mathbf{x} \quad (\text{B.57})$$

From the constraint (B.55)

$$\lambda(\mathbf{x}^H \mathbf{R}_v^{-1} \mathbf{x}) = b$$

$$\text{which gives} \quad \mathbf{h}_o = \frac{b \mathbf{R}_v^{-1} \mathbf{x}}{\mathbf{x}^H \mathbf{R}_v^{-1} \mathbf{x}} \quad (\text{B.58})$$

---

## MATLAB Functions

In this appendix, we provide a brief one-line description of MATLAB functions that were referred to in this book. The source of each function is given in parentheses where detailed information can be found. Page numbers for functions explicitly discussed in the text are also given.

**TABLE C.1**  
**MATLAB functions.**

| Function   | Description                                                                             | Page |
|------------|-----------------------------------------------------------------------------------------|------|
| a2r        | Direct parameters to autocorrelation conversion                                         | 367  |
| aplatest   | Estimation of all-pole lattice parameters (Book toolbox)                                | 460  |
| arls       | AR model estimation using the LA criterion without windowing (Book toolbox)             | 451  |
| armals     | ARMA model estimation using the LA criterion without windowing (Book toolbox)           | 466  |
| arwin      | AR model estimation using the LA criterion without windowing (Book toolbox)             | 451  |
| autoc      | Computation of autocovariance sequence (Book toolbox)                                   | 210  |
| autocfft   | Computation of autocovariance sequence using the FFT (Book toolbox)                     | 210  |
| bartlett   | Computation of Bartlett window coefficients (MATLAB)                                    | 230  |
| boxcar     | Computation of rectangular window coefficients (MATLAB)                                 | 206  |
| bt_psd     | Blackman-Tukey power spectral density computation (Book toolbox)                        | 227  |
| chebwin    | Computation of Chebyshev window coefficients (MATLAB)                                   | 206  |
| chol       | Computation of Cholesky decomposition (MATLAB)                                          | 278  |
| cohere     | Coherence function estimation (MATLAB SP toolbox)                                       | 241  |
| conv       | Convolution sum computation (MATLAB)                                                    | 48   |
| corr       | Computation of cross-correlation sequence (MATLAB)                                      | 240  |
| csd        | Cross-spectral density computation (MATLAB SP toolbox)                                  |      |
| cumsum     | Cumulative-sum computation (MATLAB)                                                     |      |
| df2latcf   | Direct-form to lattice-form conversion (Book toolbox)                                   | 67   |
| df2ldrf    | Direct-form to lattice/ladder-form conversion (Book toolbox)                            |      |
| dpss       | Discrete prolate spheroidal sequence window coefficient computation (MATLAB SP toolbox) | 248  |
| dftgn      | Generation of discrete fractional Gaussian noise (Book toolbox)                         | 721  |
| durbin     | Implementation of Durbin algorithm (Book toolbox)                                       | 358  |
| eig        | Computes eigenvalues and eigenvectors of a matrix (MATLAB)                              |      |
| esprit_ls  | Least-squares ESPRIT for frequency estimation (Book toolbox)                            | 493  |
| esprit_tls | Total least-squares ESPRIT for frequency estimation (Book toolbox)                      | 493  |
| ev_method  | Eigenvector method for frequency estimation (Book toolbox)                              | 488  |
| faest      | FAEST RLS algorithm (Book toolbox)                                                      | 576  |
| filter     | Direct-form-II filter implementation (MATLAB)                                           | 50   |
| filtic     | Computation of direct-form-II filter initial conditions (MATLAB SP toolbox)             | 50   |
| firlms     | FIR LMS adaptive filtering algorithm (Book toolbox)                                     | 526  |
| hamming    | Computation of Hamming window coefficients (MATLAB)                                     | 206  |
| hanning    | Computation of Hann window coefficients (MATLAB)                                        | 206  |

**TABLE C.1**  
**MATLAB functions. (Con't)**

| Function  | Description                                                                          | Page     |
|-----------|--------------------------------------------------------------------------------------|----------|
| invschur  | Implementation of inverse Schür algorithm (Book toolbox)                             | 375      |
| invtoepl  | Computation of $\mathbf{R}^{-1}$ when $\mathbf{R}$ is Toeplitz (Book toolbox)        | 378      |
| k2r       | Lattice parameters to autocorrelation sequence conversion (Book toolbox)             | 367      |
| kaiser    | Computation of Kaiser window coefficients (MATLAB)                                   | 208      |
| ladrfilt  | Lattice/ladder filter implementation (Book toolbox)                                  |          |
| latcf2df  | Lattice to direct-form conversion (Book toolbox)                                     | 67       |
| latcfilt  | Lattice filter implementation (Book toolbox)                                         | 68       |
| ldlt      | Computes the LDU decomposition (Book toolbox)                                        | 277      |
| ldlchol   | Computes $\mathbf{LDL}^T$ using chol                                                 | 278      |
| ldrf2df   | Lattice/ladder to direct-form conversion (Book toolbox)                              |          |
| lduneqs   | Solution of normal equations using LDU decomposition (Book toolbox)                  | 277      |
| levins    | Implementation of Levinson's algorithm (Book toolbox)                                | 359      |
| lsigest   | Computation of LS signal estimators (Book toolbox)                                   | 288      |
| lsmatvec  | Computation of $\mathbf{R}$ and $\mathbf{d}$ for FIR LS filtering (Book toolbox)     | 408      |
| lu        | LU decomposition (MATLAB)                                                            |          |
| mgs       | Implementation of modified GL algorithm (Book toolbox)                               | 430      |
| minnorm   | Minimum-norm method for frequency estimation (Book toolbox)                          | 488      |
| music     | MUSIC frequency estimation (Book toolbox)                                            | 485      |
| phd       | Pisarenko harmonic decomposition (Book toolbox)                                      | 484      |
| pmtm      | Power spectrum estimation via Thomson multitaper method (MATLAB SP toolbox)          | 248      |
| psd       | Power spectrum estimation via Welch's method (MATLAB SP toolbox)                     | 213, 232 |
| pzls      | Pole-zero coefficient estimation using the LS criterion (Book toolbox)               | 463      |
| qr        | Computation of QR decomposition (MATLAB)                                             | 424      |
| rand      | Generates pseudorandom numbers that are uniformly distributed over $(0, 1)$ (MATLAB) | 83       |
| randn     | Generates $\mathcal{N}(0, 1)$ pseudorandom numbers (MATLAB)                          | 83       |
| rls       | Implementation of conventional RLS algorithm (Book toolbox)                          |          |
| rootmusic | Root-MUSIC frequency estimation (Book toolbox)                                       | 485      |
| schurlg   | Schür algorithm (Book toolbox)                                                       | 370      |
| stablepdf | Computes pdf plots of stable distributions numerically (Book toolbox)                | 95       |
| stepdown  | Lattice-form to direct-form conversion in Levinson algorithm (Book toolbox)          | 366      |
| stepup    | Direct-form to lattice-form conversion in Levinson algorithm (Book toolbox)          | 366      |
| svd       | Computation of SVD (MATLAB)                                                          | 436      |
| tfe       | Transfer function estimation (MATLAB SP toolbox)                                     | 243      |
| toeplitz  | Toeplitz matrix from first row and column (MATLAB)                                   | 48       |
| triang    | Computation of triangular window coefficients (MATLAB)                               |          |
| udut      | Computation of $\mathbf{UDU}^H$ decomposition (Book toolbox)                         |          |

---

# Useful Results from Matrix Algebra

In this appendix, we review the fundamental concepts of linear algebra in complex-valued space. The aim is to present as many possible concepts as are necessary to understand the book. For a complete treatment, refer to many excellent references in literature including Leon (1998), Strang (1980), and Gill et al. (1981).

## D.1 COMPLEX-VALUED VECTOR SPACE

The *unitary complex space*  $\mathbb{C}^N$  is defined as the space of all the  $N$ -dimensional complex-valued vectors, which are denoted by a boldface letter, or by the  $N$ -tuple of its component, for example,

$$\mathbf{x} = [x_1 \ x_2 \ \cdots \ x_N]^T = [x_1^* \ x_2^* \ \cdots \ x_N^*]^H \quad (\text{D.1.1})$$

where we use the following notation for the superscripts:  $T$  means transpose,  $*$  means conjugate, and  $H$  means conjugate (of the) transpose, or adjoint. In the case of real-valued vectors, the real space is denoted by  $\mathbb{R}^N$  and is also known as the *Euclidean space*.

### Some Definitions

1. The *inner product* between two vectors  $\mathbf{x}$  and  $\mathbf{y}$  is defined by

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^H \mathbf{y} = \sum_{i=1}^N x_i^* y_i \quad (\text{D.1.2})$$

2. Two vectors  $\mathbf{x}$  and  $\mathbf{y}$  are *orthogonal* if their inner product is zero, that is,

$$\mathbf{x}^H \mathbf{y} = 0 \quad (\text{D.1.3})$$

The zero vector  $\mathbf{0}$  is orthogonal to any vector in the same space.

3. The norm of a vector provides a measure of the “size” of a vector. It is a nonnegative number  $\|\mathbf{x}\|$  that satisfies the following properties:

- a.  $\|\mathbf{x}\| > 0$  for  $\mathbf{x} \neq \mathbf{0}$  and  $\|\mathbf{0}\| = 0$ .
- b.  $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$  for any complex number  $\alpha$ .
- c.  $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$  (triangle inequality).

The  $p$  norm of  $\mathbf{x}$  is defined as

$$\|\mathbf{x}\|_p = \left( \sum_{i=1}^N |x_i|^p \right)^{1/p} \quad (\text{D.1.4})$$

which satisfies all three properties given above. For  $p = 2$ , we obtain the *Euclidean norm*  $\|\mathbf{x}\|_2$  which, for simplicity, is denoted by  $\|\mathbf{x}\|$ . It is defined as

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^H \mathbf{x}} = \sqrt{\sum_{i=1}^N |x_i|^2} \quad (\text{D.1.5})$$

4. An *orthonormalized set* is a set of  $L$  vectors  $\mathbf{x}_l$ ,  $l = 1, 2, \dots, L$ , such that

$$\mathbf{x}_l^H \mathbf{x}_k = \begin{cases} 1 & l = k \\ 0 & l \neq k \end{cases} \quad (\text{D.1.6})$$

5. *Cauchy–Schwartz inequality*: Two vectors  $\mathbf{x}$  and  $\mathbf{y}$  belonging to the same space satisfy

$$|\mathbf{x}^H \mathbf{y}| \leq \|\mathbf{x}\| \cdot \|\mathbf{y}\| \quad (\text{D.1.7})$$

where the equality applies when  $\mathbf{x} = a\mathbf{y}$ , with  $a$  being a (real- or complex-valued) scalar.

6. The *angle*  $\theta$  between two vectors is defined as

$$\cos \theta = \frac{\mathbf{x}^H \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} \quad (\text{D.1.8})$$

## D.2 MATRICES

A rectangular array of  $N \times M$  complex numbers ordered in  $N$  rows and  $M$  columns is called a *matrix* and is denoted by capital boldface letters, for example,

$$\mathbf{A} = [a_{i,k}] \quad 1 \leq i \leq N, 1 \leq k \leq M \quad (\text{D.2.1})$$

Any linear transformation from space  $\mathbb{C}^N$  into space  $\mathbb{C}^M$  can be represented by a suitable  $N \times M$  matrix, if two bases in  $\mathbb{C}^N$  and  $\mathbb{C}^M$  are already defined. Linear transformations from space  $\mathbb{C}^N$  into space  $\mathbb{C}^N$  are given by square  $N \times N$  non-singular matrices, in which case the transformation can be considered as a change of basis. We consider square matrices for the following development.

### D.2.1 Some Definitions

1. A system of *linearly independent* vectors  $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_N$  in a complex space  $\mathbb{C}^N$  is called a *basis* for  $\mathbb{C}^N$  if it is possible to express any vector  $\mathbf{x} \in \mathbb{C}^N$  by means of  $N$  coefficients  $a_1, a_2, \dots, a_N$  as

$$\mathbf{x} = a_1 \mathbf{e}_1 + a_2 \mathbf{e}_2 + \cdots + a_N \mathbf{e}_N = \sum_{i=1}^N a_i \mathbf{x}_i \quad (\text{D.2.2})$$

If a vector has the components  $x_1, x_2, \dots, x_N$  in a given basis, then the linearly transformed vector  $\mathbf{y}$  has components

$$\begin{aligned} y_1 &= a_{11}x_1 + \cdots + a_{1N}x_N \\ &\vdots \\ y_N &= a_{N1}x_1 + \cdots + a_{NN}x_N \end{aligned} \quad (\text{D.2.3})$$

in the basis defined by the transformation

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ a_{21} & \cdots & a_{2N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{bmatrix} \quad (\text{D.2.4})$$

This transformation can be expressed, using the well-known row-by-column product between matrices and vectors, as

$$\mathbf{y} = \mathbf{Ax} \quad (\text{D.2.5})$$

2. The transformation from  $\mathbf{y}$  to  $\mathbf{x}$  is called an *inverse transformation*, which is again *linear*. It is written as

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{y} \quad (\text{D.2.6})$$

where  $\mathbf{A}^{-1}$  (if it exists) is a matrix and is known as an inverse of  $\mathbf{A}$ , defined in (D.3.5).

3. The transformation that leaves unchanged the vector basis is said to be the *identity transformation*, and the related matrix is indicated generally by  $\mathbf{I}$ , which is given by

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \quad (\text{D.2.7})$$

4. Two linear transformations of  $C^N$  into itself can be applied to a vector, obtaining a third transformation, called the *product transformation*

$$\mathbf{y} = \mathbf{Ax} \quad \mathbf{z} = \mathbf{By} = \mathbf{B}(\mathbf{Ax}) = (\mathbf{BA})\mathbf{x} \quad \Rightarrow z = \mathbf{Cx} \quad (\text{D.2.8})$$

where the matrix  $\mathbf{C}$  is the product of  $\mathbf{B}$  and  $\mathbf{A}$ . In general, the matrix product is not *commutative*, that is,  $\mathbf{AB} \neq \mathbf{BA}$ .

5. The operation of *transposition* of a matrix inverts the orders of rows and columns; that is, element  $a_{ij}$  takes the place of  $a_{ji}$  in the new matrix. Similarly, the conjugate transpose of a matrix  $\mathbf{A}$  is a matrix in which element  $a_{ij}^*$  takes the place of  $a_{ji}$ . The operations of conjugation and transposition are commutative, that is,

$$\mathbf{A}^H = (\mathbf{A}^*)^T = (\mathbf{A}^T)^* \quad (\text{D.2.9})$$

6. A *matrix norm*  $\|\mathbf{A}\|$  satisfies the following properties:

- a.  $\|\mathbf{A}\| > 0$  for  $\mathbf{A} \neq \mathbf{0}$  and  $\|\mathbf{0}\| = 0$ .
- b.  $\|\alpha\mathbf{A}\| = |\alpha|\|\mathbf{A}\|$  for any complex number  $\alpha$ .
- c.  $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$  (triangle inequality).
- d.  $\|\mathbf{AB}\| \leq \|\mathbf{A}\| \|\mathbf{B}\|$ , which is needed because the matrix multiplication operation creates new matrices.

An important matrix norm is the *Frobenius norm*, defined as

$$\|\mathbf{A}\|_F \triangleq \sqrt{\sum_{i=1}^N \sum_{k=1}^N |a_{ik}|^2} \quad (\text{D.2.10})$$

which treats the matrix as a “long vector.” Using any vector  $p$  norm, we can obtain the matrix norm

$$\|\mathbf{A}\|_p \triangleq \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Ax}\|_p}{\|\mathbf{x}\|_p} \quad (\text{D.2.11})$$

which measures the *amplification power* of matrix  $\mathbf{A}$ . The matrix norm for  $p = 2$  is known as the *spectral norm* and is of great theoretical significance, and it is simply denoted by  $\|\mathbf{A}\|$ . When a matrix acts upon a vector  $\mathbf{x}$  of length  $\|\mathbf{x}\|_p$ , it transforms  $\mathbf{x}$  into vector  $\mathbf{Ax}$  of length  $\|\mathbf{Ax}\|_p$ . The ratio  $\|\mathbf{Ax}\|_p / \|\mathbf{x}\|_p$  provides the magnification factor of the linear transformation  $\mathbf{Ax}$ . The number  $\|\mathbf{A}\|_p$  is the maximum magnification caused by  $\mathbf{A}$ . Similarly, the minimum magnification due to  $\mathbf{A}$  is given by

$$\min |\mathbf{A}|_p \triangleq \min_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Ax}\|_p}{\|\mathbf{x}\|_p} \quad (\text{D.2.12})$$

and the ratio  $\|\mathbf{A}\|_p / \min |\mathbf{A}|_p$  characterizes the *dynamic range* of the linear transformation performed by matrix  $\mathbf{A}$ . This interpretation provides a nice geometric picture for the concept of condition number (see Section D.3.2).

7. A matrix  $\mathbf{A}$  is called *Hermitian* if

$$\mathbf{A}^H = \mathbf{A} \quad (\text{D.2.13})$$

and a *Hermitian form*  $H(\mathbf{x}, \mathbf{x})$  is the second-order real homogeneous polynomial

$$H(\mathbf{x}, \mathbf{x}) = \sum_{i=1}^N \sum_{k=1}^N h_{ik} x_i^* x_k \quad h_{ik} = h_{ki}^* \quad (\text{D.2.14})$$

8. A real-valued matrix  $\mathbf{A}$  is called *symmetric* if  $\mathbf{A}^T = \mathbf{A}$  and a *quadratic form*  $Q(\mathbf{x}, \mathbf{x})$  is the second-order real homogeneous polynomial

$$Q(\mathbf{x}, \mathbf{x}) = \sum_{i=1}^N \sum_{k=1}^N h_{ik} x_i x_k \quad h_{ik} = h_{ki} \quad (\text{D.2.15})$$

9. Matrix  $\mathbf{L}$  is called a *lower triangular* matrix if all elements above the principal diagonal are zero. Similarly, matrix  $\mathbf{U}$  is called an *upper diagonal* matrix if all elements below the principal diagonal are zero.

10. The *trace* of a matrix is the sum of the elements of its principal diagonal, that is,

$$\text{tr}(\mathbf{A}) = \sum_{i=1}^N a_{ii} \quad (\text{D.2.16})$$

$$\text{with the property} \quad \text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA}) = \text{tr}(\mathbf{A}^H \mathbf{B}^H) \quad (\text{D.2.17})$$

for any square matrices  $\mathbf{A}$  and  $\mathbf{B}$ .

11. A *diagonal* matrix is a square  $N \times N$  matrix with  $a_{ij} = 0$  for  $i \neq j$ ; that is, all elements off the principal diagonal are zero. It appears as

$$\mathbf{A} = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{NN} \end{bmatrix} \quad (\text{D.2.18})$$

12. A *Toeplitz* matrix is defined as

$$\mathbf{A} = [a_{i,k}] = [a_{i-k}] \quad 1 \leq i \leq N, 1 \leq k \leq M \quad (\text{D.2.19})$$

A square Toeplitz matrix appears as

$$\mathbf{A} = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \cdots & a_{1-N} \\ a_1 & a_0 & a_{-1} & \cdots & a_{2-N} \\ a_1 & a_1 & a_0 & \cdots & a_{3-N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N-1} & a_{N-2} & a_{N-3} & \cdots & a_0 \end{bmatrix} \quad (\text{D.2.20})$$

13. A matrix is called *persymmetric* if it is symmetric about the cross-diagonal, that is,  $a_{ij} = a_{N-j+1, N-i+1}$ ,  $1 \leq i \leq N, 1 \leq j \leq N$ .

14. The *exchange* matrix  $\mathbf{J}$  is defined by

$$\mathbf{J} \triangleq \begin{bmatrix} 0 & \dots & 0 & 0 & 1 \\ 0 & \dots & 0 & 1 & 0 \\ 0 & \dots & 1 & 0 & 0 \\ \vdots & & \vdots & \vdots & \vdots \\ 1 & \dots & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{aligned}\mathbf{J}^2 &= \mathbf{J} \\ \mathbf{J}^T &= \mathbf{J} \\ \mathbf{JA} &= \text{flipud}(\mathbf{A}) \\ \mathbf{AJ} &= \text{fliplr}(\mathbf{A})\end{aligned}$$

where the MATLAB functions `flipud(A)` and `fliplr(A)` reverse the order of rows and columns of a matrix  $\mathbf{A}$ , respectively.

15. A matrix is called *centrosymmetric* if it is both symmetric and persymmetric. It can be easily seen that a centrosymmetric matrix has the property  $\mathbf{J}^T \mathbf{AJ} = \mathbf{A}$  when  $\mathbf{A}$  is real or  $\mathbf{J}^T \mathbf{AJ} = \mathbf{A}^*$  when  $\mathbf{A}$  complex.
16. A matrix is called *Hankel* if the elements along the secondary diagonals, that is, the diagonals that are perpendicular to the main diagonal, are equal. If  $\mathbf{A}$  is Hankel, then  $\mathbf{JA}$  is Toeplitz.
17. The inverse of a triangular, symmetric, Hermitian, persymmetric and centrosymmetric matrix has the same structure. The inverse of a Toeplitz matrix is persymmetric and the inverse of a Hankel matrix is symmetric.
18. A *partition* of an  $N \times M$  matrix  $\mathbf{A}$  is a notational rearrangement in terms of its submatrices. For example, a  $2 \times 2$  partitioning of  $\mathbf{A}$  is

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \quad (\text{D.2.21})$$

where each “element”  $\mathbf{A}_{ik}$  is a submatrix of  $\mathbf{A}$ .

### D.2.2 Properties of Square Matrices

1. The operations of transposition  $T$ , conjugation  $*$ , or both  $H$  are distributive, that is,

$$\begin{aligned}(\mathbf{A} + \mathbf{B})^T &= \mathbf{A}^T + \mathbf{B}^T \\ (\mathbf{A} + \mathbf{B})^* &= \mathbf{A}^* + \mathbf{B}^* \\ (\mathbf{A} + \mathbf{B})^H &= \mathbf{A}^H + \mathbf{B}^H\end{aligned} \quad (\text{D.2.22})$$

2. For the operators  $T$ ,  $H$ , or  $-1$  (inversion), we have

$$\begin{aligned}(\mathbf{AB})^T &= \mathbf{B}^T \mathbf{A}^T \\ (\mathbf{AB})^H &= \mathbf{B}^H \mathbf{A}^H \\ (\mathbf{AB})^{-1} &= \mathbf{B}^{-1} \mathbf{A}^{-1}\end{aligned} \quad (\text{D.2.23})$$

3. The operators  $*$ ,  $T$ ,  $H$ , and  $-1$  are commutative, for example,

$$(\mathbf{A}^H)^{-1} = (\mathbf{A}^{-1})^H \quad (\text{D.2.24})$$

Thus we can use the compact notation  $\mathbf{A}^{-T}$ , or  $\mathbf{A}^{-*}$ , etc.

4. Given any matrix  $\mathbf{A}$ , matrix  $\mathbf{B} = \mathbf{A}^H \mathbf{A}$  is Hermitian [see (D.2.13)] and if  $\mathbf{A}$  is invertible, then for such a  $\mathbf{B}$ , we have

$$\mathbf{A}^{-H} \mathbf{BA}^{-1} = \mathbf{A}^{-H} \mathbf{A}^H \mathbf{AA}^{-1} = \mathbf{I} \quad (\text{D.2.25})$$

5. If  $\mathbf{H}$  is the matrix of the coefficients  $h_{ik}$ , the Hermitian form (D.2.14) can be written as

$$H(\mathbf{x}, \mathbf{x}) = \mathbf{x}^H \mathbf{Hx} = \langle \mathbf{x}, \mathbf{Hx} \rangle \quad (\text{D.2.26})$$

Similarly, If  $\mathbf{H}$  is the real-valued matrix of the coefficients  $h_{ik}$ , the quadratic form (D.2.15) can be written as

$$H(\mathbf{x}, \mathbf{x}) = \mathbf{x}^T \mathbf{Hx} = \langle \mathbf{x}, \mathbf{Hx} \rangle \quad (\text{D.2.27})$$

6. A Hermitian matrix  $\mathbf{A}$  is called

|                          |                                             |                             |
|--------------------------|---------------------------------------------|-----------------------------|
| Positive definite if     | $\mathbf{x}^H \mathbf{A} \mathbf{x} > 0$    |                             |
| Positive semidefinite if | $\mathbf{x}^H \mathbf{A} \mathbf{x} \geq 0$ | (also nonnegative definite) |
| Negative definite if     | $\mathbf{x}^H \mathbf{A} \mathbf{x} < 0$    |                             |
| Negative semidefinite if | $\mathbf{x}^H \mathbf{A} \mathbf{x} \leq 0$ | (also nonpositive definite) |

for all  $\mathbf{x} \neq \mathbf{0}$ .

7. The operation of the trace of a matrix satisfies

$$\text{tr}(\mathbf{A} \pm \mathbf{B}) = \text{tr}(\mathbf{A}) \pm \text{tr}(\mathbf{B}) \quad (\text{D.2.29})$$

$$\text{tr}(k\mathbf{A}) = k \text{tr}(\mathbf{A}) \quad (\text{D.2.30})$$

$$\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA}) \quad (\text{D.2.31})$$

$$\text{tr}(\mathbf{B}^{-1}\mathbf{AB}) = \text{tr}(\mathbf{A}) \quad (\text{D.2.32})$$

$$\text{tr}(\mathbf{AA}^H) = \sum_{i=1}^N \sum_{j=1}^N |a_{ij}|^2 \quad (\text{D.2.33})$$

### D.3 DETERMINANT OF A SQUARE MATRIX

The determinant of a square matrix  $\mathbf{A}$  is denoted by

$$\det(\mathbf{A}) \triangleq \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{vmatrix} \quad (\text{D.3.1})$$

and is equal to the sum of the products of the elements of any row or column and their respective cofactors, that is,

$$\det(\mathbf{A}) = a_{i1}C_{i1} + a_{i2}C_{i2} + \cdots + a_{iN}C_{iN} \quad (\text{D.3.2})$$

$$\text{or} \quad \det(\mathbf{A}) = a_{1k}C_{1k} + a_{2k}C_{2k} + \cdots + a_{Nk}C_{Nk} \quad (\text{D.3.3})$$

where the  $C_{ik}$  are called *cofactors*, given by

$$C_{ik} = (-1)^{i+k} \det(\mathbf{A}_{ik}) \quad (\text{D.3.4})$$

where  $\mathbf{A}_{ik}$  is an  $(N - 1)$ st-order square matrix obtained by deleting the  $i$ th row and  $k$ th column. Thus the determinant needs to be computed recursively; that is, the  $N$ th-order determinant is computed from the  $(N - 1)$ st-order determinant, which in turn is computed from the  $(N - 2)$ nd-order, and so on. If  $\det(\mathbf{A}) \neq 0$ , then the inverse  $\mathbf{A}^{-1}$  of  $\mathbf{A}$  exists and is unique. The  $\mathbf{A}^{-1}$  matrix is given by

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{vmatrix} C_{11} & C_{21} & \cdots & C_{N1} \\ C_{12} & C_{22} & \cdots & C_{N2} \\ \vdots & \vdots & \ddots & \vdots \\ C_{1N} & C_{2N} & \cdots & C_{NN} \end{vmatrix} \quad (\text{D.3.5})$$

#### D.3.1 Properties of the Determinant

Below we provide some useful properties of the determinant.

- If a row (or column) of a matrix is a linear combination of other rows (or columns), then  $\det(\mathbf{A}) = 0$ . In particular, if (a) a row (or column) is proportional or equal to another row (or column) or (b) a row (or column) is identically zero, then  $\det(\mathbf{A}) = 0$ .

2. If two rows (or columns) are exchanged with each other, then the determinant changes its sign.
3. For a triangular matrix (upper or lower)  $\mathbf{A}$ , the determinant is obtained by multiplying all the elements of its principal diagonal, that is,

$$\det(\mathbf{A}) = \prod_{n=1}^N a_{nn} \quad (\text{D.3.6})$$

4. The  $\det(\mathbf{A})$  is unchanged if  $\mathbf{A}$  is replaced by its transpose  $\mathbf{A}^T$ ; that is,

$$\det(\mathbf{A}) = \det(\mathbf{A}^T) \quad (\text{D.3.7})$$

5. Using the above property, we also claim that the determinant of a Hermitian matrix is real, since

$$\det(\mathbf{A}) = \det(\mathbf{A}^H) = \det(\mathbf{A}^T) \Rightarrow \det(\mathbf{A}) = \det(\mathbf{A}^*) = \det(\mathbf{A})^* \quad (\text{D.3.8})$$

6. The determinant of a product of matrices is the product of their determinants; that is,

$$\det(\mathbf{AB}) = \det(\mathbf{A}) \det(\mathbf{B}) \quad (\text{D.3.9})$$

7. If matrix  $\mathbf{A}$  is nonsingular, that is, its inverse  $\mathbf{A}^{-1}$  exists, then

$$\det(\mathbf{A}^{-1}) = [\det(\mathbf{A})]^{-1} = \frac{1}{\det(\mathbf{A})} \quad (\text{D.3.10})$$

8. Given an arbitrary constant  $c$  (possibly complex-valued), we have

$$\det(c\mathbf{A}) = c^N \det(\mathbf{A}) \quad (\text{D.3.11})$$

### D.3.2 Condition Number

One of the important equations in signal processing is the linear equation  $\mathbf{R}\mathbf{c} = \mathbf{d}$ , where  $\mathbf{R}$  is a matrix of known values,  $\mathbf{d}$  is a vector of known quantities, and  $\mathbf{c}$  is a vector of unknown coefficients. The investigation of how the solution of  $\mathbf{R}\mathbf{c} = \mathbf{d}$  is affected by small changes (perturbations) in the elements of  $\mathbf{R}$  and  $\mathbf{d}$  leads to an important characteristic number of matrix  $\mathbf{R}$ , called the *condition number*.

If vector  $\mathbf{d}$  is perturbed to  $\mathbf{d} + \delta\mathbf{d}$ , the exact solution  $\mathbf{c}$  is perturbed to  $\mathbf{c} + \delta\mathbf{c}$ . Therefore,

$$\mathbf{R}(\mathbf{c} + \delta\mathbf{c}) = \mathbf{d} + \delta\mathbf{d}$$

which implies that  $\delta\mathbf{c} = \mathbf{R}^{-1}\delta\mathbf{d}$  since  $\mathbf{R}\mathbf{c} = \mathbf{d}$  (D.3.12)

or using property 4 of matrix norm

$$\|\delta\mathbf{c}\| \leq \|\mathbf{R}^{-1}\| \|\delta\mathbf{d}\| \quad (\text{D.3.13})$$

From the same norm property and  $\mathbf{d} = \mathbf{R}\mathbf{c}$ , we obtain

$$\|\mathbf{d}\| \leq \|\mathbf{R}\| \|\mathbf{c}\| \quad (\text{D.3.14})$$

Multiplying (D.3.13) by (D.3.14) and solving, we obtain

$$\frac{\|\delta\mathbf{c}\|}{\|\mathbf{c}\|} \leq \|\mathbf{R}\| \|\mathbf{R}^{-1}\| \frac{\|\delta\mathbf{d}\|}{\|\mathbf{d}\|} \quad (\text{D.3.15})$$

Similarly, keeping  $d$  constant and perturbing  $\mathbf{R}$  to  $\mathbf{R} + \delta\mathbf{R}$ , we have

$$(\mathbf{R} + \delta\mathbf{R})(\mathbf{c} + \delta\mathbf{c}) = \mathbf{d}$$

from which, after ignoring the second-order product term  $\delta\mathbf{R}\delta\mathbf{c}$ , we obtain

$$\frac{\|\delta\mathbf{c}\|}{\|\mathbf{c}\|} \leq \|\mathbf{R}\| \|\mathbf{R}^{-1}\| \frac{\|\delta\mathbf{R}\|}{\|\mathbf{R}\|} \quad (\text{D.3.16})$$

A careful inspection of (D.3.15) and (D.3.16) shows that the *relative error* in the exact solution is bounded by the number

$$\text{cond}(\mathbf{R}) \triangleq \|\mathbf{R}\| \|\mathbf{R}^{-1}\| \quad (\text{D.3.17})$$

which is known as the *condition number* of matrix  $\mathbf{R}$ , multiplied by the relative perturbation in the data ( $\mathbf{R}$  or  $\mathbf{d}$ ). When relatively small perturbations in  $\mathbf{R}$  cause relatively small (large) perturbations in the solution of  $\mathbf{R}\mathbf{c} = \mathbf{d}$ , matrix  $\mathbf{R}$  is said to be *well (ill) conditioned*. Clearly, ill-conditioned matrices have large condition numbers, and therefore their large magnification power amplifies small perturbations to the extent that makes the obtained solution totally inaccurate.

Since the norm of the identity matrix  $\|\mathbf{I}\| = 1$ , we have

$$\|\mathbf{I}\| = \|\mathbf{R}\mathbf{R}^{-1}\| \leq \|\mathbf{R}\| \|\mathbf{R}^{-1}\| = \text{cond}(\mathbf{R})$$

that is,  $\text{cond}(\mathbf{R}) \geq 1$  (D.3.18)

The best possible condition number is 1.

## D.4 UNITARY MATRICES

A matrix  $\mathbf{A}$  is called a *unitary* matrix if its inverse is equal to its conjugate transpose, that is,

$$\mathbf{A}^{-1} = \mathbf{A}^H \Rightarrow \mathbf{A}^H \mathbf{A} = \mathbf{I} \quad (\text{D.4.1})$$

For a real-valued matrix,  $\mathbf{A}$  is called an *orthogonal* matrix if its inverse is equal to its transpose, that is,

$$\mathbf{A}^{-1} = \mathbf{A}^T \Rightarrow \mathbf{A}^T \mathbf{A} = \mathbf{I} \quad (\text{D.4.2})$$

If we write the unitary matrix  $\mathbf{A}$  as a set of  $N$  column vectors, that is,

$$\mathbf{A} = [\mathbf{a}_1 \mathbf{a}_2 \cdots \mathbf{a}_N] \quad (\text{D.4.3})$$

then we can show that

$$\mathbf{a}_i^H \mathbf{a}_k = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases} \triangleq \delta_{ik} \quad (\text{D.4.4})$$

that is, the column vectors of a unitary matrix are orthonormal.

A transformation is called a *unitary transformation* if the transformation matrix is unitary. Vector inner products, vector norms, and angles between two vectors are *invariant* (i.e., they are preserved) under unitary transformation. Thus given two vectors  $\mathbf{x}$  and  $\mathbf{y}$  and a unitary matrix  $\mathbf{A}$ , we have

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{Ax}, \mathbf{Ay} \rangle \quad (\text{D.4.5})$$

and  $\|\mathbf{x}\|^2 = \|\mathbf{Ax}\|^2$  (D.4.6)

This implies that the absolute value of the determinant of a unitary matrix is unity, or

$$|\det(\mathbf{A})| = 1 \quad \mathbf{A} \text{ unitary} \quad (\text{D.4.7})$$

since from (D.4.1), (D.3.9), and (D.3.8), we have

$$\det(\mathbf{I}) = \det(\mathbf{A}^H \mathbf{A}) = \det(\mathbf{A}^H) \det(\mathbf{A}) = \det(\mathbf{A})^* \det(\mathbf{A}) = |\det(\mathbf{A})|^2 = 1 \quad (\text{D.4.8})$$

### D.4.1 Hermitian Forms after Unitary Transformations

Let  $H(\mathbf{y}, \mathbf{y}) = \langle \mathbf{y}, \mathbf{R}\mathbf{y} \rangle = \mathbf{y}^H \mathbf{R}\mathbf{y}$  be an arbitrary Hermitian form for any matrix  $\mathbf{R}$ . Define a transformation  $\mathbf{y} = \mathbf{Ax}$  for any unitary matrix  $\mathbf{A}$ . Then we can write  $H(\mathbf{y}, \mathbf{y})$  as

$$H(\mathbf{y}, \mathbf{y}) = \mathbf{x}^H \mathbf{A}^H \mathbf{R} \mathbf{A} \mathbf{x} = \mathbf{x}^H \mathbf{P} \mathbf{x} \quad (\text{D.4.9})$$

where

$$\mathbf{P} = \mathbf{A}^H \mathbf{R} \mathbf{A} = \mathbf{A}^{-1} \mathbf{R} \mathbf{A} \quad (\text{D.4.10})$$

Matrix  $\mathbf{R}$  can be reduced to a diagonal form by unitary transformation

$$\mathbf{U}^H \mathbf{R} \mathbf{U} = \mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_N) \quad (\text{D.4.11})$$

Hence the Hermitian form  $H(\mathbf{y}, \mathbf{y})$  can be written as

$$H(\mathbf{y}, \mathbf{y}) = \mathbf{y}^H \mathbf{R} \mathbf{y} = \mathbf{y}^H \mathbf{U} \mathbf{\Lambda} \mathbf{U}^H \mathbf{y} = \mathbf{x}^H \mathbf{\Lambda} \mathbf{x} = \langle \mathbf{x}, \mathbf{\Lambda} \mathbf{x} \rangle \quad (\text{D.4.12})$$

where  $\mathbf{x} \triangleq \mathbf{A} \mathbf{y} \triangleq \mathbf{U}^H \mathbf{y}$ . Therefore, we can write

$$H(\mathbf{y}, \mathbf{y}) = \sum_{i=1}^N \sum_{k=1}^N r_{ik} y_i^* y_k = \sum_{i=1}^N \lambda_i |x_i|^2 \quad (\text{D.4.13})$$

## D.4.2 Significant Integral of Quadratic and Hermitian Forms

Consider a quadratic form  $Q(\mathbf{x}, \mathbf{x}) = \langle \mathbf{x}, \mathbf{A} \mathbf{x} \rangle$ . The indefinite integral of the exponential of  $Q(\mathbf{x}, \mathbf{x})$  is given by

$$I_N \triangleq \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \exp(-\mathbf{x}^T \mathbf{A} \mathbf{x}) d\mathbf{x} \quad (\text{D.4.14})$$

where  $d\mathbf{x} = dx_1 dx_2 \cdots dx_N$ , and it has many applications. Using (D.4.12) and (D.4.13) (specialized to the real case), we obtain

$$\langle \mathbf{x}, \mathbf{A} \mathbf{x} \rangle = \langle \mathbf{y}, \mathbf{\Lambda} \mathbf{y} \rangle = \sum_{i=1}^N \lambda_i y_i^2 \quad (\text{D.4.15})$$

where  $\lambda_i, i = 1, 2, \dots, N$ , are eigenvalues of  $\mathbf{A}$ . Thus (D.4.14) becomes

$$I_N = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \exp\left(-\sum_{i=1}^N \lambda_i y_i^2\right) d\mathbf{y} = \prod_{i=1}^N \int_{-\infty}^{\infty} \exp(-\lambda_i y_i^2) dy_i \quad (\text{D.4.16})$$

Now by using the result

$$\int_{-\infty}^{\infty} \exp(-\alpha x^2) dx = \sqrt{\frac{\pi}{\alpha}} \quad (\text{D.4.17})$$

Equation (D.4.14) becomes

$$I_N = \prod_{i=1}^N \int_{-\infty}^{\infty} \sqrt{\frac{\pi}{\lambda_i}} = \sqrt{\frac{\pi^N}{\lambda_1 \lambda_2 \cdots \lambda_N}} \quad (\text{D.4.18})$$

Finally, using the fact that  $\det(\mathbf{A}) = \prod_{i=1}^N \lambda_i$ , we obtain

$$I_N = \sqrt{\frac{\pi^N}{\det(\mathbf{A})}} \quad (\text{D.4.19})$$

The result in (D.4.19) can be extended to the complex case. Let  $H(\mathbf{z}, \mathbf{z})$  be the Hermitian form of a complex-valued vector  $\mathbf{z} = \mathbf{x} + j\mathbf{y}$ . Then the indefinite integral of the exponential of  $H(\mathbf{z}, \mathbf{z})$  is given by

$$J_N \triangleq \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \exp(-\mathbf{z}^T \mathbf{A} \mathbf{z}) d\mathbf{z} \quad (\text{D.4.20})$$

$$= \frac{\pi^N}{\det(\mathbf{A})} \quad (\text{D.4.21})$$

where  $d\mathbf{z} = dx_1 dx_2 \cdots dx_N dy_1 dy_2 \cdots dy_N$ . Thus sometimes we get slightly different results for the complex case.

TABLE D.1  
Summary of properties of vectors and matrices in real and complex spaces.

| Real versus Complex                                                                                      |                   |
|----------------------------------------------------------------------------------------------------------|-------------------|
| $\mathbb{R}^N$ : $N$ -dimensional Euclidean space                                                        | $\leftrightarrow$ |
| Norm: $\ \mathbf{x}\ ^2 = x_1^2 + \dots + x_N^2$                                                         | $\leftrightarrow$ |
| Transpose: $\mathbf{A}^T = [a_{ji}]$                                                                     | $\leftrightarrow$ |
| $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$                                                            | $\leftrightarrow$ |
| Inner product: $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$                        | $\leftrightarrow$ |
| Orthogonality: $\mathbf{x}^T \mathbf{y} = 0$                                                             | $\leftrightarrow$ |
| Symmetric matrices: $\mathbf{A} = \mathbf{A}^T$                                                          | $\leftrightarrow$ |
| Orthogonal matrices: $\mathbf{Q}^T = \mathbf{Q}^{-1}$                                                    | $\leftrightarrow$ |
| $\mathbf{A} = \mathbf{Q} \Lambda \mathbf{Q}^{-1} = \mathbf{Q} \Lambda \mathbf{Q}^{-T}$ (real $\Lambda$ ) | $\leftrightarrow$ |
| Norm invariance: $\ \mathbf{Qx}\  = \ \mathbf{x}\ $                                                      | $\leftrightarrow$ |
| $(\mathbf{Qx})^T (\mathbf{Qy}) = \mathbf{x}^T \mathbf{y}$                                                | $\leftrightarrow$ |
| $\mathbb{C}^N$ : $N$ -dimensional complex space                                                          |                   |
| Norm: $\ \mathbf{x}\ ^2 =  x_1 ^2 + \dots +  x_N ^2$                                                     |                   |
| Hermitian: $\mathbf{A}^H = [a_{ji}^*]$                                                                   |                   |
| $(\mathbf{AB})^H = \mathbf{B}^H \mathbf{A}^H$                                                            |                   |
| Inner product: $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^H \mathbf{y}$                        |                   |
| Orthogonality: $\mathbf{x}^H \mathbf{y} = 0$                                                             |                   |
| Symmetric matrices: $\mathbf{A} = \mathbf{A}^H$                                                          |                   |
| Unitary matrices: $\mathbf{U}^T = \mathbf{U}^{-1}$                                                       |                   |
| $\mathbf{A} = \mathbf{U} \Lambda \mathbf{U}^{-1} = \mathbf{U} \Lambda \mathbf{U}^{-H}$ (real $\Lambda$ ) |                   |
| Norm invariance: $\ \mathbf{Ux}\  = \ \mathbf{x}\ $                                                      |                   |
| $(\mathbf{Ux})^H (\mathbf{Uy}) = \mathbf{x}^H \mathbf{y}$                                                |                   |

Table D.1 summarizes various properties described above as they relate to both complex-valued and real-valued matrices.

## D.5 POSITIVE DEFINITE MATRICES

Positive definite matrices play an important role in signal processing in general and least-squares (LS) estimation in particular, and they deserve some attention. A conjugate symmetric  $M \times M$  matrix  $\mathbf{R}$  is called *positive definite* if and only if the Hermitian form

$$\mathbf{x}^H \mathbf{R} \mathbf{x} = \sum_{i,j}^M r_{ij} x_i^* x_j > 0 \quad (\text{D.5.1})$$

for every  $\mathbf{x} \neq \mathbf{0}$ . For example, the symmetric matrix

$$\mathbf{R} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \quad (\text{D.5.2})$$

is positive definite because the quadratic form

$$\mathbf{x}^T \mathbf{R} \mathbf{x} = x_1^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + x_3^2 > 0 \quad (\text{D.5.3})$$

can be expressed as a sum of squares that is positive unless  $x_1 = x_2 = x_3 = 0$ .

From this simple example it is obvious that using the definition to find out whether a given matrix is positive definite is very tedious. Fortunately, use of this approach is not necessary because other criteria can be used to make a faster decision (Strang 1980; Horn and Johnson 1985; Nobel and Daniel 1988). We next summarize some positive definiteness tests that are useful in LS estimation.

### Positive definiteness criterion

An  $M \times M$  matrix  $\mathbf{R}$  is positive definite if and only if it satisfies any one of the following criteria:

1.  $\mathbf{x}^H \mathbf{R} \mathbf{x} > 0$  for all nonzero vectors  $\mathbf{x}$ .
2. All eigenvalues of  $\mathbf{R}$  are positive.
3. All principal submatrices  $\mathbf{R}_m$ ,  $1 \leq m \leq M$ , have positive determinants. The principal

submatrices of  $\mathbf{R}$  are determined as follows:

$$\mathbf{R}_1 = [r_{11}] \quad \mathbf{R}_2 = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \quad \mathbf{R}_3 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad \dots \quad \mathbf{R}_M = \mathbf{R} \quad (\text{D.5.4})$$

It is important to stress that this criterion applies also to the lower right submatrices or any chain of submatrices that starts with a diagonal element  $r_{ii}$  as the first submatrix and then expands it by adding a new row and column at each step.

4. There exists an  $L \times M$ ,  $M > L$ , matrix  $\mathbf{S}$  with linearly independent columns such that  $\mathbf{R} = \mathbf{S}^H \mathbf{S}$ . This requirement for the columns of  $\mathbf{S}$  to be linearly independent implies that  $\mathbf{S}$  has rank  $M$ .
5. There exists a nonsingular  $M \times M$  matrix  $\mathbf{W}$  such that  $\mathbf{R} = \mathbf{W}^H \mathbf{W}$ . The choices for the matrix  $\mathbf{W}$  are a triangular matrix obtained by Cholesky's decomposition (see Section 6.3) or an orthonormal matrix obtained from the eigenvectors of  $\mathbf{R}$  (see Section 3.5).
6. There exists a nonsingular  $M \times M$  matrix  $\mathbf{P}$  such that the matrix  $\mathbf{P}^H \mathbf{R} \mathbf{P}$  is positive definite.

**Properties of positive definite matrices.** A positive definite matrix  $\mathbf{R}$  has the following properties:

1. The diagonal elements of  $\mathbf{R}$  are positive.
2.  $r_{ii}r_{jj} > |r_{ij}|^2$  ( $i \neq j$ )
3. The element of  $\mathbf{R}$  with the largest absolute value lies on the diagonal.
4. The  $\det \mathbf{R} > 0$ . Hence  $\mathbf{R}$  is nonsingular.
5. The inverse matrix  $\mathbf{R}^{-1}$  is positive definite.
6. The matrix obtained by deleting a row and the corresponding column from  $\mathbf{R}$  is positive definite.

## APPENDIX E

---

# Minimum Phase Test for Polynomials

In this appendix we prove a theorem that provides a test for checking if the zeros of a polynomial are inside the unit circle (minimum phase condition) using the lattice parameters. The required lattice parameters can be obtained from the coefficients of the polynomial using the algorithm (2.5.28) in Section 2.5.

**THEOREM E.1.** The polynomial

$$A_P(z) = 1 + a_1^{(P)}z^{-1} + \cdots + a_P^{(P)}z^{-P} \quad (\text{E.1})$$

is minimum-phase, that is, has all its zeros inside the unit circle if and only if

$$|k_m| < 1 \quad 1 \leq m \leq P \quad (\text{E.2})$$

**Proof.** We will prove the sufficiency part first, followed by the necessary part. Also we will make use of property (2.4.16)–(2.4.17) of the all-pass systems.

**Sufficiency.** We will prove by induction that if  $|k_m| < 1, 1 \leq m \leq P$ , then  $A_P(z)$  is minimum-phase. For  $P = 1$  we have

$$A_1(z) = 1 + a_1^{(1)}z^{-1} = 1 + k_1z^{-1}$$

Clearly if  $|k_1| < 1$ , then  $A_1(z)$  is minimum-phase. Assume now that  $A_{m-1}(z)$  is minimum-phase. It can be then expressed as

$$A_{m-1}(z) = \prod_{i=1}^{m-1} (1 - z_i^{(m-1)}z^{-1}) \quad (\text{E.3})$$

$$\text{where } |z_i^{(m-1)}| < 1 \quad 1 \leq i \leq m-1 \quad (\text{E.4})$$

However, from the recursion (2.5.9),

$$A_m(z) = A_{m-1}(z) + k_m z^{-1} B_{m-1}(z) \quad (\text{E.5})$$

Hence

$$A_m(z_i^{(m)}) = A_{m-1}(z_i^{(m)}) + k_m \left( \frac{1}{z_i^{(m)}} \right) B_{m-1}(z_i^{(m)}) = 0 \quad 1 \leq i \leq m$$

$$\text{or } k_m = \frac{A_{m-1}(z_i^{(m)})}{(1/z_i^{(m)}) B_{m-1}(z_i^{(m)})} \quad 1 \leq i \leq m \quad (\text{E.6})$$

But

$$\begin{aligned} B_{m-1}(z) &= z^{-(m-1)} A_{m-1}(z^{-1}) = z^{-(m-1)} \prod_{i=1}^{m-1} (1 - z_i^{(m-1)} z) \\ &= \prod_{i=1}^{m-1} (z^{-1} - z_i^{(m-1)}) \end{aligned} \quad (\text{E.7})$$

Since  $A^{(m-1)}(z)$  has real coefficients, either its zeros are real or they appear in complex conjugate pairs. Thus, a zero and its complex conjugate can always be grouped in the numerator and denominator of (E.6) as

$$|k_m|^2 = \prod_{i=1}^{m-1} \left| \frac{1 - z_i^{(m-1)} / z_i^{(m)}}{1/z_i^{(m)} - (z_i^{(m-1)})^*} \right|^2 |z_i^{(m)}|^2 \quad 1 \leq i \leq m \quad (\text{E.8})$$

Applying property (2.4.17) to every factor of (E.8), with  $a = z_i^{(m-1)}$ , gives

$$|k_m| \begin{cases} < 1 & |z_i^{(m)}| < 1 \\ = 1 & |z_i^{(m)}| = 1 \\ > 1 & |z_i^{(m)}| > 1 \end{cases} \quad (\text{E.9})$$

Thus, if  $|z_i^{(m)}| < 1$ ,  $1 \leq i \leq m$ , then  $|k_M| < 1$ .

**Necessity.** We will prove that if  $A_P(z)$  is minimum-phase, then  $|k_m| < 1$ ,  $1 \leq m \leq P$ . To this end we will show that if  $A_m(z)$  is minimum-phase, then  $|k_m| < 1$  and  $A_{m-1}(z)$  is minimum-phase. From

$$A_m(z) = \prod_{i=1}^m (1 - z_i^{(m)} z^{-1})$$

we see by inspection that the coefficient of the highest power  $z^{-m}$  is

$$k_m = \alpha_m^{(m)} = \prod_{i=1}^m (-z_i^{(m)}) \quad (\text{E.10})$$

$$\text{Thus, } |k_m| \leq \prod_{i=1}^m |z_i^{(m)}| < 1 \quad (\text{E.11})$$

To show that  $A^{(m-1)}(z)$  is minimum-phase, we recall that

$$A_{m-1}(z) = \frac{A_m(z) - k_m B_m(z)}{1 - |k_m|^2} \quad (\text{E.12})$$

If  $z_i^{(m-1)}$  is a zero of  $A^{(m-1)}(z)$ , we have

$$A_{m-1}(z_i^{(m-1)}) = \frac{A_m(z_i^{(m-1)}) - k_m B_m(z_i^{(m-1)})}{1 - |k_m|^2} = 0 \quad (\text{E.13})$$

If  $|k_m| \neq 1$ , then (E.13) implies that

$$k_m = \frac{A_m(z_i^{(m-1)})}{B_m(z_i^{(m-1)})} \quad 1 \leq i \leq m-1 \quad (\text{E.14})$$

Applying again the property (2.4.17) to  $|k_m|^2$  in (E.14) shows that since  $|k_m| < 1$ , then  $|z_i^{(m-1)}| < 1$  for  $1 \leq i \leq m-1$ . Hence,  $A^{(m-1)}(z)$  is minimum-phase.



---

# Bibliography

- Abed-Meraim, K., W. Qiu, and Y. Hua. 1997. Blind system identification. *Proc. IEEE*, 85(8):1310–1322, August.
- Abramowitz, M., and I. Stegun, Eds. 1970. *Handbook of Mathematical Functions*. Dover Publications, New York.
- Adler, R., R. Feldman, and M. Taqqu, Eds. 1998. *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*. Birkhäuser, Boston.
- Akaike, H. 1969. Fitting autoregressive models for prediction. *Annals Inst. Statistical Mathematics*, 21:243–247.
- \_\_\_\_\_. 1970. Statistical predictor identification. *Annals Inst. Statistical Mathematics*, 22:203–217.
- \_\_\_\_\_. 1974. A new look at the statistical model identification. *IEEE Trans. Automatic Control*, 19:716–722.
- Alexander, S. T. 1987. Transient weight misadjustment properties for the precision LMS algorithm. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP(35):1250–1258.
- \_\_\_\_\_. 1993. A method for recursive least-squares filtering based upon an inverse QR decomposition. *IEEE Trans. Signal Processing*, 41(1):20–30.
- Anderson, T. 1971. *The Statistical Analysis of Time Series*. Wiley, New York.
- Applebaum, S. P., and D. J. Chapman. 1976. Antenna arrays with mainbeam constraints. *IEEE Trans. Antennas and Propagation*, 24(9):650–662.
- Atal, B. S., and M. Schroeder. 1970. Adaptive predictive coding of speech signals. *Bell System Tech. J.*, pp. 1973–1987, October.
- \_\_\_\_\_. and \_\_\_\_\_. 1978. Linear prediction analysis of speech based on a pole-zero representation. *J. Acoust. Soc. Am.*, 64(5):1310–1318, November.
- \_\_\_\_\_. and \_\_\_\_\_. 1979. Predictive coding of speech signals and subjective error criteria. *IEEE Trans. Acoustics, Speech and Signal Processing*, 27(3):247–254.
- Baggeroer, A. B., W. A. Kuperman, and P. N. Mikhalevsky. 1993. An overview of matched field methods in ocean acoustics. *IEEE J. Ocean Engineering*, 18(4):401–424.
- Barabell, A. J. 1983. Improving the resolution performance of eigenstructure-based direction-finding algorithms. *Proc. International Conference on Acoustics, Speech and Signal Processing*, pp. 336–339.
- Baranowski, E. J. 1995. Improved pre-Doppler STAP algorithm for adaptive nulling in airborne radars. *Proc. Asilomar Conf. Signals, Systems, and Computers*, pp. 1173–1177.
- \_\_\_\_\_. and J. Ward. 1997. Source localization using adaptive subspace beamformer outputs. *Proc. International Conference on Acoustics, Speech and Signal Processing*, pp. 3773–3776.
- Barnsley, M. F., R. L. Devaney, B. B. Mandelbrot, H. O. Peitgen, D. Saupe, and R. F. Voss. 1988. The science of fractal images. In H. Peitgen and D. Saupe, Eds., *Fractals*. Springer-Verlag, New York.
- Bartlett, M. S. 1948. Smoothing periodograms for time series with continuous spectra. *Nature*, 161:686–687.
- \_\_\_\_\_. 1950. Periodogram analysis and continuous spectra. *Biometrika*, 31:1–16.
- Bell, A. J., and T. J. Sejnowski. 1995. An information maximization approach to blind separation and blind deconvolution. *Neural Computation*, 6:1129–1159.
- Bellini, S. 1986. Bussgang techniques for blind equalization. In *GLOBECOM*, pp. 1634–1640.

- \_\_\_\_\_. 1994. Bussgang techniques for blind deconvolution and equalization. In S. Haykin, Ed., *Blind Deconvolution*. Prentice Hall, Englewood Cliffs, NJ.
- Bendat, J. S., and A. G. Piersol. 1980. *Engineering Applications of Correlation and Spectral Analysis*. Wiley-Interscience, New York.
- \_\_\_\_\_, \_\_\_\_\_, and \_\_\_\_\_. 1986. *Random Data: Analysis and Measurement Procedures*, 2nd ed., Wiley, New York.
- Benveniste, A., and M. Goursat. 1984. Blind equalizers. *IEEE Trans. Communications*, 32:871–883.
- \_\_\_\_\_, \_\_\_\_\_, and G. Ruget. 1980. Robust identification of a non-minimum phase system: Blind adjustment of a linear equalizer in data communications. *IEEE Trans. Automatic Control*, 25:385–399.
- \_\_\_\_\_, M. Metivier, and P. Priouret. 1987. *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag, New York.
- Beran, J. 1994. *Statistics for Long-Memory Processes*. Chapman and Hall, New York.
- \_\_\_\_\_, R. Sherman, M. S. Taqqu, and W. Willinger. 1995. Long-range dependence in variable-bit-rate video traffic. *IEEE Trans. Communications*, 43(2):1566–1579, February.
- Bienvenu, G., and L. Kopp. 1983. Optimality of high resolution array processing using the eigensystem approach. *IEEE Trans. Acoustics, Speech and Signal Processing*, 31(10):1235–1248.
- Bierman, G. J. 1977a. *Factorization Methods for Discrete Sequential Estimation*. Academic Press, New York.
- \_\_\_\_\_. 1977b. Numerical comparison of Kalman filter algorithmic determination case study. *Automatica*, 13:23–35.
- \_\_\_\_\_, and C. L. Thornton. 1977. Numerical comparison of Kalman filter algorithms: Orbit determination case study. *Automatica*, 13:23–35.
- Bingham, J. A. C. 1988. *The Theory and Practice of Modem Design*. Wiley, New York.
- Björck, A. 1967. Solving linear least-squares problems by Gram-Schmidt orthogonalization. *BIT*, 7:1–21.
- Blackman, R., and J. Tukey. 1958. *The Measurement of Power Spectra*. Dover Publications, New York.
- Bode, H., and C. Shannon. 1950. A simplified derivation of linear least squares smoothing and prediction theory. *Proc. IRE*, 38:417–425.
- Böhme, J. F., and B. Yang. 1992. Rotation-based RLS algorithms. *IEEE Trans. Signal Processing*, 40(5):1151–1167.
- Bolt, B. 1993. *Earthquakes and Geological Discovery*. Freeman and Company, New York.
- Borkar, V. S. 1995. *Probability Theory: An Advanced Course*. Springer-Verlag, New York.
- Boroson, D. M. 1980. Sample size considerations in adaptive arrays. *IEEE Trans. Aerospace and Electronic Systems*, 16(4):446–451.
- Borsari, G. K., and A. O. Steinhardt. 1995. Cost-efficient training strategies for space-time adaptive processing algorithms. *Proc. Asilomar Conf. on Signals, Systems, and Computers*, pp. 650–654.
- Bose, S., and A. O. Steinhardt. 1995. A maximal invariant framework for adaptive detection with structured and unstructured covariance matrices. *IEEE Trans. Signal Processing*, 43(9):2164–2175.
- Box, G. E. P., and G. M. Jenkins. 1976. *Time Series Analysis: Forecasting and Control*, rev. ed. Holden-Day, San Francisco, CA.
- \_\_\_\_\_, \_\_\_\_\_, and G. C. Reinsel. 1994. *Time Series Analysis: Forecasting and Control*, 3rd ed. Prentice Hall, Englewood Cliffs, NJ.
- Bozic, S. M. 1994. *Digital and Kalman Filtering*, 2nd ed. Halsted Press, New York.
- Brennan, L. E., and I. S. Reed. 1973. Theory of adaptive radar. *IEEE Trans. Aerospace and Electronic Systems*, 9(3):237–252.
- Brillinger, D. R. 1965. An introduction to polyspectra. *Ann. Math. Statist.*, 36:1351–1374.
- \_\_\_\_\_. 1980. *Time Series: Data Analysis and Theory*. Holden-Day, San Francisco, CA.
- Brockwell, P. J., and R. A. Davis. 1991. *Time Series: Theory and Methods*, 2nd ed. Springer-Verlag, New York.
- \_\_\_\_\_, and \_\_\_\_\_. 1996. *Introduction to Time Series and Forecasting*. Springer-Verlag, New York.
- Brown, R. G., and P. Y. C. Hwang. 1997. *Introduction to Random Signals and Applied Kalman Filtering*, 3rd ed. Wiley, New York.
- Bryn, F. 1962. Optimum signal processing of three-dimensional arrays operating on Gaussian signals and noise. *J. Acoustical Soc. Am.*, 34:289–297.

- Bucklew, J. A., T. Kurtz, and W. A. Sethares. 1993. Weak convergence and local stability properties of fixed step size recursive algorithms. *IEEE Trans. Information Theory*, 39:966–978.
- Buckley, K. M. 1987. Spatial/spectral filtering with linearly constrained minimum variance beamformers. *IEEE Trans. Acoustics, Speech and Signal Processing*, 35(3):249–266.
- Burg, J. 1975. *Maximum Entropy Spectral Analysis*. Ph.D. thesis, Stanford University, Stanford, CA.
- Burg, J. P. 1978. Maximum entropy spectral estimation. In D. G. Childers, Ed., *Modern Spectral Analysis*. IEEE Press, New York. Originally appeared in *Proceedings of the 37th Annual SEG Meeting*, Dec. 31, 1967.
- Bussgang, J. 1952. Cross correlation functions of amplitude-distorted Gaussian signals. *Tech. Report 216*, MIT Research Lab. of Electronics, Cambridge, MA.
- Butterweck, H. J. 1995. A steady-state analysis of the LMS adaptive algorithm without use of the independence assumption. *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, pp. 1404–1407.
- Cadzow, J. A. 1996. Blind deconvolution via cumulant extrema. *IEEE Signal Processing Magazine*, pp. 24–42, May.
- \_\_\_\_\_, and X. Li. 1995. Blind deconvolution. *Digital Signal Processing J.*, 5(1):3–20.
- Capon, J. 1969. High-resolution frequency-wavenumber spectrum analysis. *Proc. IEEE*, 57:1408–1418, August.
- \_\_\_\_\_, R. J. Greenfield, and R. J. Kolker. 1967. Multidimensional maximum-likelihood processing of large aperture seismic arrays. *J. Acoustical Soc. Am.*, 55(2):192–211.
- Caraicos, C., and B. Liu. 1984. A roundoff error analysis of the LMS adaptive algorithm. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP(32):34–41.
- Carayannis, G., N. Kalouptsidis, and D. G. Manolakis. 1982. Fast recursive algorithms for a class of linear equations. *IEEE Trans. Acoustics, Speech and Signal Processing*, 30(2):227–239, April.
- \_\_\_\_\_, E. Koukoutsis, and C. C. Halkias. 1991. Hardware implementation of partitioned parallel algorithms in linear prediction. *Signal Processing*, pp. 257–259, September.
- \_\_\_\_\_, \_\_\_\_, D. G. Manolakis, and C. C. Halkias. 1985. A new look on the parallel implementation of the Schür algorithm for the solution of Toeplitz equations. *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, pp. 1858–1861.
- \_\_\_\_\_, D. Manolakis, and N. Kalouptsidis. 1986. A unified view of parametric processing algorithms for prewindowed signals. *Signal Processing*, 10(4):335–368, June.
- \_\_\_\_\_, D. G. Manolakis, and \_\_\_\_\_. 1983. A fast sequential algorithm for least-squares filtering and prediction. *IEEE Trans. Acoustics, Speech and Signal Processing*, 31(6):1394–1402.
- Carlson, B. D. 1988. Covariance matrix estimation errors and diagonal loading in adaptive arrays. *IEEE Trans. Aerospace and Electronic Systems*, 24(4):397–401.
- Carter, C. G., and A. H. Nuttall. 1980a. A brief summary of a generalized framework for spectral estimation. *Signal Processing*, 2(4):387–390, October.
- \_\_\_\_\_, and \_\_\_\_\_. 1980b. On the weighted overlapped segment-averaging method for power spectral estimation. *Proc. IEEE*, 68(10):1352–1354, October.
- Chambers, J. M., C. L. Mallows, and B. W. Stuck. 1976. A method for simulating stable random variables. *J. Am. Stat. Assn.*, 71:340–344, June.
- Chan, T. F. 1982. An improved algorithm for computing the SVD. *ACM Trans. Mathematical Software*, 8:72–88.
- Childers, D. G. 1978. *Modern Spectral Analysis*. IEEE Press, New York.
- Chiuppesi, F., G. Galati, and P. Lombardi. 1980. Optimization of rejection filters. *IEE Proc. Part F: Communications, Radar and Signal Processing*, 127(5):354–360, October.
- Cioffi, J. M. 1987. Limited-precision effects in adaptive filtering. *IEEE Trans. Circuits and Systems*, 34(7):821–833.
- \_\_\_\_\_, and T. Kailath. 1984. Fast, recursive-least-squares transversal filters for adaptive filtering. *IEEE Trans. Acoustics, Speech and Signal Processing*, 32(2):304–337.
- Claasen, T., and W. Mecklenbrauker. 1985. Adaptive techniques for signal processing in communications. *IEEE Communications Magazine*, 23(11):8–19, November.
- Claasen, T. A. C. M., and W. F. G. Mecklenbrauker. 1981. Comparison of the convergence of two algorithms for adaptive FIR digital filters. *IEEE Trans. Circuits and Systems*, 28(6):510–518, June.
- Claerbout, J. F., and E. A. Robinson. 1963. The error in least-squares inverse filtering. *Geophysics*, 29(1):118–120, January.

- Colwell, R., Ed. 1983. *Manual of Remote Sensing*. American Society for Photogrammetry and Remote Sensing, Falls Church, VA.
- Compton, R. T. 1988. *Adaptive Antennas*. Prentice Hall, Englewood Cliffs, NJ.
- Conte, E., M. Lops, and G. Ricci. 1996. Adaptive matched filter detection in spherically invariant noise. *IEEE Signal Processing Letters*, 3(8):248–250.
- Cox, H. 1973. Resolving power and sensitivity to mismatch of optimum array processors. *J. Acoustical Soc. Am.*, 54:771–785.
- \_\_\_\_\_, R. M. Zeskind, and M. M. Owen. 1987. Robust adaptive beamforming. *IEEE Trans. Acoustics, Speech and Signal Processing*, 35(10):1365–1377.
- Crawford, D., R. Stewart, and E. Toma. 1997. Digital signal processing strategies for active noise control. *Electronics and Communications Engineering J.*, pp. 81–89, April.
- Dahlquist, G., and A. Bjorck. 1974. *Numerical Methods*. Prentice Hall, Englewood Cliffs, NJ. Translated by N. Anderson.
- Daniell, P. J. 1946. Discussion of “On the theoretical specification and sampling properties of auto-correlated time series.” *J. Royal Stat. Soc.*, 8:88–90.
- Davenport, W. B., Jr. 1970. *Probability and Random Processes*. McGraw-Hill, New York.
- Davis, R. C., L. E. Brennan, and I. S. Reed. 1976. Angle estimation with adaptive arrays in external noise fields. *IEEE Trans. Aerospace and Electronic Systems*, 12(2):179–186.
- Delsarte, P., and Y. Genin. 1986. The split Levinson algorithm. *IEEE Trans. Acoustics, Speech and Signal Processing*, 34(3):470–478, June.
- Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Ann. Royal Stat. Soc.*, pp. 1–38.
- Dennis, J. E., and R. B. Schnabel. 1983. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice Hall, Englewood Cliffs, NJ.
- Deriche, M., and A. H. Tewfik. 1993. Signal modeling with filtered discrete fractional noise processes. *IEEE Trans. Signal Processing*, 41(9):2839–2849, September.
- Ding, Z. 1994. Blind channel identification and equalization using spectral correlation measurements, part 1: Frequency-domain approach. In W. A. Gardner, Ed., *Cyclostationarity in Communications and Signal Processing*, pp. 417–437. IEEE Press, New York.
- \_\_\_\_\_. 1998. Adaptive filters for blind equalization. In V. Madisetti and D. Williams, Eds., *The Digital Signal Processing Handbook*. CRC Press, New York.
- \_\_\_\_\_, 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.
- \_\_\_\_\_, R. Kennedy, B. Anderson, and C. Johnson. 1991. Ill-convergence of Godard blind equalizers in data communication systems. *IEEE Trans. Communications*, 39:1313–1327, September.
- Diniz, P. S. 1997. *Adaptive Filtering*. Kluwer Academic Publishers, Boston, MA.
- Dobrin, M. 1988. *Introduction to Geophysical Prospecting*, 4th ed. McGraw-Hill, New York.
- Dongara, J. J., et al. 1979. *LINPACK User's Guide*. SIAM, Philadelphia, PA.
- Donoho, D. L. 1981. On minimum entropy deconvolution. In B. F. Findlay, Ed., *Applied Time Series Analysis II*. Academic Press, New York.
- Douglas, S., and M. Rupp. 1998. Convergence issues in the LMS adaptive filter. In V. Madisetti and D. Williams, Eds., *The Digital Signal Processing Handbook*. CRC Press, New York.
- Duffy, F., V. Iyer, and W. Surwillo. 1989. *Clinical Electroencephalography and Topographic Brain Mapping*. Springer-Verlag, New York.
- Durbin, J. 1960. The fitting of time-series models. *Rev. Int. Stat. Inst.*, 28:233–244.
- Duttweiler, D. L. 1982. Adaptive filter performance with nonlinearities in the correlation multiplier. *IEEE Trans. Acoustics, Speech and Signal Processing*, 30(4):578–586, August.
- Eleftheriou, E., and D. D. Falconer. 1986. Tracking properties and steady state performance of RLS adaptive filter algorithms. *IEEE Trans. Acoustics, Speech and Signal Processing*, 34:1097–1110.
- El-Jaroudi, A., and J. Makhoul. 1989. Discrete pole-zero modeling and applications. *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, pp. 2162–2165.
- \_\_\_\_\_, and \_\_\_\_\_. 1991. Discrete all-pole modeling. *IEEE Trans. Signal Processing*, 39(2):411–423, February.
- Elliot, S., and P. Nelson. 1993. Active noise control. *IEEE Signal Processing Magazine*, 10(4):12–35, October.
- Embree, P. M., and B. Kimble. 1991. *C Language Algorithms for Digital Signal Processing*. Prentice Hall, Englewood Cliffs, NJ.

- Er, M. H., and A. Cantoni. 1983. Derivative constraints for broad-band element space antenna array processors. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 31(12):1378–1393.
- Falconer, D. D., and L. Ljung. 1978. Application of fast Kalman estimation to adaptive equalization. *IEEE Trans. Communications*, 26(10):1439–1446.
- Falconer, K. 1990. *Fractal Geometry: Mathematical Foundations and Applications*. Wiley, New York.
- Feder, J. 1988. *Fractals*. Plenum Press, New York.
- Feller, W. 1957. *An Introduction to Probability Theory and Its Applications*, 2nd ed., vol. 1. Wiley, New York.
- \_\_\_\_\_. 1971. *An Introduction to Probability Theory and Its Applications*, 2nd ed., vol. 2. Wiley, New York.
- Feuer, A., and E. Weinstein. 1985. Convergence analysis of LMS filters with uncorrelated Gaussian data. *IEEE Trans. Acoustics, Speech and Signal Processing*, 33(1):222–229, February.
- Figueiras-Vidal, A. R., Ed. 1996. *Digital Signal Processing in Telecommunications*. Springer-Verlag, London.
- Fijalkow, I., C. Manlove, and R. Johnson. 1998. Adaptive fractionally spaced blind CMA equalization: Excess MSE. *IEEE Trans. Signal Processing*, 46(1):227–231, January.
- Flanagan, J. L. 1972. *Speech Analysis, Synthesis and Perception*. Springer-Verlag, New York.
- Flandrin, P. 1989. On the spectrum of fractional Brownian motions. *IEEE Trans. Information Theory*, 35(1):197–199, January.
- Foschini, G. J. 1985. Equalizing without altering or detecting data. *AT&T Tech. J.*, 64:1885–1911.
- Friedlander, B. 1982. Lattice filters for adaptive processing. *Proc. IEEE*, 70(8):829–867, August.
- \_\_\_\_\_. and M. Morf. 1982. Least-squares algorithms for adaptive linear-phase filtering. *IEEE Trans. Acoustics, Speech and Signal Processing*, 30(3):381–390, June.
- Frost, O. L. 1972. An algorithm for linearly constrained adaptive array processing. *Proc. IEEE*, 60(8):962–935.
- Fukunaga, K. 1990. *Statistical Pattern Recognition*, 2nd ed. Academic Press, New York.
- Furui, S. 1989. *Digital Speech Processing, Synthesis, and Recognition*. Marcel Dekker, New York.
- \_\_\_\_\_. and M. M. Sondhi, Eds. 1992. *Advances in Speech Signal Processing*. Marcel Dekker, New York.
- Gardner, W. 1991. Exploitation of spectral redundancy in cyclostationary signals. *IEEE Signal Processing Magazine*, 8(2):14–36, April.
- Gardner, W. A. 1984. Learning characteristics of stochastic-gradient-descent algorithm: A general study, analysis, and critique. *Signal Processing*, 6:113–133.
- \_\_\_\_\_. 1994. *Cyclostationarity in Communications and Signal Processing*. IEEE Press, New York.
- Garrett, M. W., and W. Willinger. 1994. Analysis, modeling and generation of self-similar VBR video traffic. *Proc. ACM Sigcomm '94*, pp. 269–280.
- Gelb, A. 1974. *Applied Optimal Estimation*. MIT Press, Cambridge, MA.
- Gentleman, W. M., and H. T. Kung. 1981. Matrix triangularization by systolic arrays. *Proc. SPIE*, (Real time signal processing IV), vol. 298, pp. 19–26.
- Giannakis, G. 1998. Channel estimation and equalization. *IEEE Signal Processing Magazine*, 15(5): 37–40, September.
- Gill, P. E., G. H. Golub, W. Murray, and M. A. Saunders. 1974. Methods of modifying matrix factorizations. *Mathematics of Computation*, 28:505–535.
- \_\_\_\_\_. W. Murray, and M. H. Wright. 1981. *Practical Optimization*. Academic Press, London.
- Gilloire, A., E. Moulines, D. Slock, and P. Duhamel. 1996. State of the art in acoustic echo cancellation. In A. R. Figueiras-Vidal, Ed., *Digital Signal Processing in Telecommunications*. Springer-Verlag, London.
- \_\_\_\_\_. and M. Vetterli. 1992. Adaptive filtering in subbands with critical sampling: Analysis, experiments and application to acoustic echo control. *IEEE Trans. Signal Processing*, 40:1862–1875.
- Gitlin, R., J. Hayes, and S. Weinstein. 1992. *Data Communications*. Plenum Press, New York.
- 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*, 20(2):125–136, March.
- \_\_\_\_\_. H. C. Meadors, and S. B. Weinstein. 1982. The tap leakage algorithm: An algorithm for the stable operation of a digitally implemented fractionally spaced adaptive equalizer. *Bell System Tech. J.*, 61(8):1817–1839, October.
- \_\_\_\_\_. and S. B. Weinstein. 1979. On the required tap-weight precision for digitally implemented, adaptive, mean-squared equalizers. *Bell System Tech. J.*, 58(2):301–321, February.

- Givens, W. 1958. Computation of plane unitary rotations transforming a general matrix to triangular form. *SIAM J. Applied Math.*, 6:26–50.
- Godard, D. N. 1980. Self-recovering equalization and carrier tracking in a two-dimensional data communication system. *IEEE Trans. Communications*, 28(11):1867–1875.
- Godfrey, R., and F. Rocca. 1981. Zero memory non-linear deconvolution. *Geophysics Prospect*, 29:189–228.
- Golub, G. H. 1965. Numerical methods for solving linear least-squares problems. *Numerical Methods*, 7:206–216.
- \_\_\_\_\_, and C. Reinsch. 1970. Singular value decomposition and least-squares problems. *Numer. Math.*, 14:403–420.
- \_\_\_\_\_, and C. F. Van Loan. 1996. *Matrix Computations*, 3rd ed. The Johns Hopkins University Press, Baltimore, MD.
- Goodman, N. R. 1957. On the joint estimation of spectra, cospectrum of a two-dimensional stationary Gaussian process. Scientific paper no. 10, College of Engineering, New York University, New York. ASTIA Doc. AD-134919.
- \_\_\_\_\_. 1963. Statistical analysis based on a certain multivariate complex Gaussian distribution. *Ann. Math. Statist.* 34(1):152–177.
- Goodwin, G. C., and R. L. Payne. 1977. *Dynamic System Identification: Experiment Design and Data Analysis*. Academic Press, New York.
- Gradshteyn, I. S., and I. M. Ryzhik. 1994. *Tables of Integrals, Series and Products*, 5th ed. Academic Press, London.
- Granger, C. W. J., and R. Joyeux. 1980. An introduction to long-memory time series models and fractional differencing. *J. Time Series Analysis*, 1:15–29.
- Grant, P., and B. Mulgrew. 1995. Nonlinear adaptive filter: Design and application. *Proc. IFAC (Adaptive Systems in Control and Signal Processing)*, pp. 31–42.
- Gray, A. H., and J. D. Markel. 1973. Digital lattice and ladder filter synthesis. *IEEE Trans. Audio Electroacoustics*, 21(6):491–500, December.
- Gray, H. L., N. Zhang, and W. A. Woodward. 1989. On generalized fractional processes. *J. Time Series Analysis*, 10:233–257.
- Gray, R. M. 1972. On the asymptotic eigenvalue distribution of Toeplitz matrices. *IEEE Trans. Information Theory*, 18:725–730.
- \_\_\_\_\_, and L. D. Davisson. 1986. *Random Processes: A Mathematical Approach for Engineers*. Prentice Hall, Englewood Cliffs, NJ.
- Green, W. H. 1993. *Econometric Analysis*, 2nd ed. Macmillan, New York.
- Grenander, U., and G. Szegö. 1984. *Toeplitz Forms and Their Applications*, 2nd ed. Chelsea Publishing Company, New York.
- Griffiths, L. J. 1977. A continuously adaptive filter implemented as a lattice filter. *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, pp. 683–686.
- \_\_\_\_\_, and C. W. Jim. 1982. An alternative approach to linearly constrained adaptive beamforming. *IEEE Trans. Antennas and Propagation*, 30(1):27–34.
- Grossman, P. Personal communication.
- \_\_\_\_\_, L. Watkins, F. Wilhelm, D. Manolakis, and B. Lown. 1996. Cardiac vagal control and dynamic responses to psychological stress among patients with coronary artery disease. *Am. J. Cardiology*, 78:1424–1427, December.
- Hager, W. W. 1988. *Applied Numerical Linear Algebra*. Prentice Hall, Englewood Cliffs, NJ.
- Hamilton, J. D. 1994. *Time Series Analysis*. Princeton University Press, Princeton, NJ.
- Harris, F. 1978. On the use of windows for harmonic analysis with a discrete Fourier transform. *Proc. IEEE*, 66(1):51–83, January.
- Hassibi, B., A. H. Sayed, and T. Kailath. 1996. LMS is  $H^\infty$  optimal. *IEEE Trans. Signal Processing*, 44(2):267–280, February.
- Hastings, H. M., and G. Sugihara. 1993. *Fractals: A User's Guide for the Natural Sciences*. Oxford University Press, New York.
- Hatzinakos, D., and C. L. Nikias. 1991. Blind equalization using a tricepstrum based algorithm. *IEEE Trans. Communications*, 39:669–682.
- \_\_\_\_\_, and \_\_\_\_\_. 1994. Blind equalization based on higher-order statistics (HOS). In S. Haykin, Ed., *Blind Deconvolution*. Prentice Hall, Englewood Cliffs, NJ.
- Hayes, M. H. 1996. *Statistical Digital Signal Processing and Modeling*. Wiley, New York.
- Haykin, S. 1989. *Modern Filters*. Macmillan, New York.

- \_\_\_\_\_, Ed. 1991. *Advances in Spectrum Analysis and Array Processing*, vol. 1. Prentice Hall, Englewood Cliffs, NJ.
- \_\_\_\_\_, Ed. 1994. *Blind Deconvolution*. Prentice Hall, Englewood Cliffs, NJ.
- \_\_\_\_\_. 1996. *Adaptive Filter Theory*, 3rd ed. Prentice Hall, Englewood Cliffs, NJ.
- Helstrom, C. W. 1991. *Probability and Stochastic Processes for Engineers*, 2nd ed. Macmillan, New York.
- \_\_\_\_\_. 1995. *Elements of Signal Detection and Estimation*. Prentice Hall, Englewood Cliffs, NJ.
- Hewer, G. A., R. D. Martin, and J. Zeh. 1987. Robust preprocessing for Kalman filtering of glint noise. *IEEE Trans. Aerospace and Electronic Systems*, 23(1):120–128, January.
- Hinich, M. J. 1982. Testing for Gaussianity and linearity of a stationary time series. *J. Time Series Analysis*, 3(3):169–176.
- Horowitz, L. L., and K. D. Senne. 1981. Performance advantage of complex LMS for controlling narrow-band adaptive arrays. *IEEE Trans. Circuits and Systems*, 28(6):562–576.
- Hosking, J. R. M. 1981. Fractional differencing. *Biometrika*, 68:165–176.
- \_\_\_\_\_. 1984. Modeling persistence in hydrological time series using fractional differencing. *Water Resources Research*, 20(10):1898–1908.
- Householder, J. 1958. Unitary triangularization of a nonsymmetric matrix. *J. ACM*, 5:339–342.
- Howells, P. W. Intermediate frequency sidelobe canceler. U.S. Patent 3202990.
- Hsiao, J. 1974. On the optimization of MTI clutter rejection. *IEEE Trans. Aerospace and Electronic Systems*, 10(5):622–629, September.
- Hsieh, S., K. Liu, and K. Yao. 1993. A unified square-root-free approach for QRD-based recursive least-squares estimation. *IEEE Trans. Signal Processing*, 41(3):1405–1409, March.
- Hubing, N. E., and S. T. Alexander. 1991. Statistical analysis of initialization methods for RLS adaptive filters. *IEEE Trans. Signal Processing*, 39(8):1793–1804, August.
- Hudson, J. E. 1981. *Adaptive Array Principles*. Peter Peregrinus, New York.
- Iltis, R. A. 1991. Interference rejection and channel estimation for spread-spectrum communications. In N. Kalouptsidis and S. Theodoridis, Eds., *Adaptive System Identification and Signal Processing Algorithms*, pp. 466–511. Prentice Hall, Englewood Cliffs, NJ.
- Ingle, V. K., and J. G. Proakis. 1996. *Digital Signal Processing Using MATLAB*. PWS Publishing Company, Boston, MA.
- Ishimaru, A. 1990. *Electromagnetic Wave Propagation*. Prentice Hall, Englewood Cliffs, NJ.
- Itakura, F., and S. Saito. 1971a. Digital filtering techniques for speech analysis and synthesis. *Proc. 7th Int. Congress on Acoustics*, 25-C-1:261–264, Budapest.
- \_\_\_\_\_, and \_\_\_\_\_. 1971b. A statistical method for estimation of speech spectral density and formant frequencies. *Electr. Commun. Japan*, 53-A(1):36–43.
- Jain, A. K. 1989. *Digital Image Processing*. Prentice Hall, Englewood Cliffs, NJ.
- Janicki, A., and A. Weron. 1994. *Simulation and Chaotic Behavior of  $\alpha$ -Stable Stochastic Processes*. Marcel Dekker, New York.
- Jayant, N. S., and P. Noll. 1984. *Digital Coding of Waveforms*. Prentice Hall, Englewood Cliffs, NJ.
- Jenkins, G. M., and D. G. Watts. 1968. *Spectral Analysis and Its Applications*. Holden-Day, San Francisco, CA.
- Jenkins, W., and D. Marshall. 1998. Transform domain adaptive filtering. In V. Madisetti and D. Williams, Eds., *The Digital Signal Processing Handbook*. CRC Press, New York.
- Jensen, F. B., W. A. Kuperman, M. B. Porter, and H. Schmidt. 1994. *Computational Ocean Acoustics*. Springer-Verlag, New York.
- Johnson, C., and M. Larimore. 1977. Comments on and additions to “An adaptive recursive LMS filter.” *Proc. IEEE*, 65(9):1399–1402, September.
- Johnson, C. R., Jr. 1984. Adaptive IIR filtering: Current results and open issues. *IEEE Trans. Information Theory*, 30(2), part 1:237–250.
- Johnson, D. H., and S. R. DeGraaf. 1982. Improving the resolution of bearing in passive sonar arrays by eigenvalue analysis. *IEEE Trans. Acoustics, Speech and Signal Processing*, 30(4):638–647, August.
- \_\_\_\_\_, and D. E. Dudgeon. 1993. *Array Signal Processing: Concepts and Techniques*. Prentice Hall, Englewood Cliffs, NJ.
- Johnson, R., et al. 1998. Blind equalization using the constant modulus criterion: A review. *Proc. IEEE*, 86(10):1927–1949, October.
- Kagan, A., Y. V. Linnik, and C. R. Rao. 1973. *Characterization Problems in Mathematical Statistics*. Wiley-Interscience, New York.

- Kailath, T. 1974. A view of three decades of linear filtering theory. *IEEE Trans. Information Theory*, 20:146–181.
- \_\_\_\_\_. 1981. *Lectures on Linear Least-Squares Estimation*. Springer-Verlag, New York.
- Kalouptsidis, N., G. Carayannis, and D. Manolakis. 1984. Efficient recursive-in-order least-squares FIR filtering and prediction. *IEEE Trans. Acoustics, Speech and Signal Processing*, 33:1175–1187, October.
- \_\_\_\_\_. and S. Theodoridis, Eds. 1993. *Adaptive System Identification and Signal Processing Algorithms*. Prentice Hall, Englewood Cliffs, NJ.
- Kamen, E. W., and B. S. Heck. 1997. *Fundamentals of Signals and Systems Using Matlab*. Prentice Hall International, Upper Saddle River, NJ.
- Kasdin, N. J. 1995. Discrete simulation of colored noise and stochastic processes and  $1/f^\alpha$  power law noise generation. *Proc. IEEE*, 83(5):802–827, May.
- Kassam, S. A., and H. V. Poor. 1985. Robust techniques for signal processing: A survey. *Proc. IEEE*, 73(3):433–481, March.
- Kaveh, M., and A. Barabell. 1986. The statistical performance of the MUSIC and minimum-norm algorithms in resolving plane waves in noise. *IEEE Trans. Acoustics, Speech and Signal Processing*, 34(2):331–341.
- Kay, S. M. 1988. *Modern Spectral Estimation*. Prentice Hall, Englewood Cliffs, NJ.
- \_\_\_\_\_. 1993. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall, Englewood Cliffs, NJ.
- \_\_\_\_\_. 1998. *Fundamentals of Statistical Signal Processing Detection Theory*. Prentice Hall, Upper Saddle River, NJ.
- Kelly, E. J. 1986. An adaptive detection algorithm. *IEEE Trans. Aerospace and Electronic Systems*, 22(1):115–127.
- \_\_\_\_\_. 1989. Performance of an adaptive detection algorithm: Rejection of unwanted signals. *IEEE Trans. Aerospace and Electronic systems*, 25(2):122–133.
- Kendall, M. G., and A. Stuart. 1983. *Advanced Theory of Statistics*, 4th ed. Macmillan Publishing Company, New York.
- Kino, G. S. 1987. *Acoustic Waves*. Prentice Hall, Englewood Cliffs, NJ.
- Klema, V. C., and A. J. Laub. 1980. The singular value decomposition: Its computation and some applications. *IEEE Trans. Automatic Control*, 25:164–176.
- Klemm, R. 1999. *Space-Time Adaptive Processing*. IEE, London.
- Kogon, S. M., and D. G. Manolakis. 1994. Fractal-based modeling and interpolation of non-Gaussian images. *Proc. SPIE: Visual Communications and Image Processing '94*, vol. 2308, part 1:467–477.
- \_\_\_\_\_. and \_\_\_\_\_. 1996. Signal modeling with self-similar  $\alpha$ -stable processes: The fractional Lévy stable motion model. *IEEE Trans. Signal Processing*, 44(4):1006–1010, April.
- Kok, A., D. G. Manolakis, and V. K. Ingle. 1993. Symmetric noncausal spatial model for 2-D signals with applications in stochastic texture modeling. *Multidimensional Systems and Signal Processing*, 4:125–147.
- Kolmogorov, A. 1939. Sur l'interpolation et extrapolation des suites stationnaires. *C. R. Acad. Sci.*, 208:2043–2045.
- Kondoz, A. M. 1994. *Digital Speech: Coding for Low Bit Rate Communication Systems*. Wiley, New York.
- Koopmans, L. H. 1974. *The Spectral Analysis of Time Series*. Academic Press, New York.
- Koukoutsis, K., G. Carayannis, and C. C. Halkias. 1991. Superlattice/superladder computational organization for linear prediction and optimal FIR filtering. *IEEE Trans. Signal Processing*, 39(10):2199–2215, October.
- Koutrouvelis, I. A. 1980. Regression-type estimation of the parameters of stable laws. *J. Am. Stat. Assn.*, 75:918–928.
- \_\_\_\_\_. 1981. An iterative procedure for the estimation of the parameters of the stable law. *Communications in Statistics—Computation and Simulation*, 10:17–28.
- Kreithen, D. E., and A. O. Steinhardt. 1995. Target detection in post-STAP undernull clutter. *Proc. Asilomar Conf. Signals, Systems, and Computers*, pp. 1203–1207.
- Kullback, S. 1959. *Information Theory and Statistics*. Wiley, New York.
- Kumaresan, R., and D. W. Tufts. 1983. Estimating the angles of arrival of multiple plane waves. *IEEE Trans. Aerospace and Electronic Systems*, 19:134–139, January.

- Kung, S., and Y. Hu. 1983. A highly concurrent algorithm and pipelined architecture for solving Toeplitz systems. *IEEE Trans. Acoustics, Speech and Signal Processing*, 31(1):66–76, February.
- Kuo, S., and D. Morgan. 1996. *Active Noise Control Systems*. Wiley, New York.
- Kushner, H. J. 1984. *Approximation and Weak Convergence Methods for Random Processes with Applications to Stochastic System Theory*. MIT Press, Cambridge, MA.
- Lacoss, R. T. 1971. Data adaptive spectral analysis methods. *Geophysics*, 36:661–675, August.
- Lamperti, J. W. 1996. *Probability: A Survey of Mathematical Theory*, 2nd ed. Wiley, New York.
- Lapsley, P., J. Bier, A. Shoham, and E. A. Lee. 1997. *DSP Processor Fundamentals: Architectures and Features*. IEEE Press, New York.
- Lawson, C. L., and R. D. Hanson. 1974. *Solving Least-Squares Problems*. Prentice Hall, Englewood Cliffs, NJ.
- Lawrance, A. 1991. Directionality and reversibility in time series. *Int. Stat. Rev.*, 59(1):67–79.
- Lawrence, V. B., and S. K. Tewksbury. 1983. Multiprocessor implementation of adaptive digital filters. *IEEE Trans. Communications*, 31(6):826–835, June.
- Le Roux, J., and C. Gueguen. 1977. A fixed-point computation of partial correlation coefficients. *IEEE Trans. Acoustics, Speech and Signal Processing*, pp. 257–259, June.
- Lee, D. T. L., M. Morf, and B. Friedlander. 1981. Recursive least-squares ladder estimation algorithms. *IEEE Trans. Circuits and Systems*, 28(6):467–481.
- Lee, E. A., and D. G. Messerschmitt. 1994. *Digital Communication*, 2nd ed. Kluwer Academic Publishers, Boston, MA.
- Leon, S. J. 1995. *Linear Algebra with Applications*, 5th ed. Macmillan Publishing Company, New York.
- Levanon, N. 1988. *Radar Principles*. Wiley, New York.
- Levinson, N. 1947. The Wiener RMS (root-mean-square) error criterion in filter design and prediction. *J. Math. Physics*, 25:261–278.
- Lii, K. S., and M. Rosenblatt. 1982. Deconvolution and estimation of transfer function phase and coefficients for non-Gaussian linear processes. *Ann. Stat.*, 10:1195–1208.
- Lin, D. W. 1984. On the digital implementation of the fast Kalman algorithm. *IEEE Trans. Acoustics, Speech and Signal Processing*, 32:998–1005.
- Ling, F. 1991. Givens rotation based least-squares lattice and related algorithms. *IEEE Trans. Signal Processing*, 39:1541–1551.
- \_\_\_\_\_. 1993a. Echo cancellation. In N. Kalouptsidis and S. Theodoridis, Eds., *Adaptive System Identification and Signal Processing Algorithms*, pp. 407–465. Prentice Hall, Englewood Cliffs, NJ.
- \_\_\_\_\_. 1993b. Lattice algorithms. In N. Kalouptsidis and S. Theodoridis, Eds., *Adaptive System Identification and Signal Processing Algorithms*, pp. 191–259. Prentice Hall, Englewood Cliffs, NJ.
- \_\_\_\_\_, D. Manolakis, and J. G. Proakis. 1986. Numerically robust least-squares lattice-ladder algorithm with direct updating of the reflection coefficients. *IEEE Trans. Acoustics, Speech and Signal Processing*, 34(4):837–845.
- \_\_\_\_\_, and J. G. Proakis. 1986. A recursive modified Gram-Schmidt algorithm with applications to least-squares and adaptive filtering. *IEEE Trans. Acoustics, Speech and Signal Processing*, 34(4):829–836.
- Litva, J., and T. Lo. 1996. *Digital Beamforming for Wireless Communications*. Artech House, Boston, MA.
- 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 Processing*, 40:946–958.
- Ljung, L. 1987. *System Identification Theory for the User*. Prentice Hall, Englewood Cliffs, NJ.
- \_\_\_\_\_, and T. Soderstrom. 1983. *Theory and Practice of Recursive Identification*. MIT Press, Cambridge, MA.
- Ljung, S., and L. Ljung. 1985. Error propagation properties of recursive least-squares adaptation algorithms. *Automatica*, 21:157–167.
- Lucky, R. W. 1965. Automatic equalization for digital communications. *Bell System Tech. J.*, 44:547–588, April.
- \_\_\_\_\_. 1966. Techniques for adaptive equalization of digital communication systems. *Bell System Tech. J.*, 45:255–286, February.
- \_\_\_\_\_, J. Salz, and E. J. Weldon. 1968. *Principles of Data Communications*. McGraw-Hill, New York.

- Luenberger, D. G. 1984. *Linear and Nonlinear Programming*, 2nd ed. Addison-Wesley, Reading, MA.
- Lundahl, T., W. J. Ohley, S. M. Kay, and R. Siffert. 1986. Fractional Brownian motion: A maximum likelihood estimator and its application to image texture. *IEEE Trans. Medical Imaging*, 5(3):152–161, September.
- Macchi, O. 1995. *Adaptive Processing: The LMS Approach with Applications in Transmission*. Wiley, New York.
- \_\_\_\_\_. 1996. The theory of adaptive filtering in a random time-varying environment. In A. R. Figueiras-Vidal, Ed., *Digital Signal Processing in Telecommunications*. Springer-Verlag, London.
- Maeder, R. E. 1995. Fractional Brownian motion. *The Mathematica Journal*, 6(1):38–48.
- Makhoul, J. 1975a. A class of all-zero lattice digital filters: Properties and applications. *IEEE Trans. Acoustics, Speech and Signal Processing*, 26(4):304–314, August.
- \_\_\_\_\_. 1975b. Linear prediction: A tutorial review. *Proc. IEEE*, 63(4):561–580.
- \_\_\_\_\_. 1976. New lattice methods for linear prediction. *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, pp. 462–465.
- \_\_\_\_\_. 1977. Stable and efficient lattice methods for linear prediction. *IEEE Trans. Acoustics, Speech and Signal Processing*, 25:423–428, October.
- \_\_\_\_\_. 1978. A class of all-zero lattice digital filters: Properties and applications. *IEEE Trans. Acoustics, Speech and Signal Processing*, 26:304–314, August.
- \_\_\_\_\_. 1981. On the eigenvectors of symmetric Toeplitz matrices. *IEEE Trans. Acoustics, Speech and Signal Processing*, 29:868–872.
- \_\_\_\_\_. 1986. Maximum confusion spectral analysis. *IEEE Workshop on Spectral Estimation and Modeling*, pp. 6–9, November.
- MakSYM, J. N. 1979. A robust formulation of an optimum cross-spectral beamformer for line arrays. *J. Acoust. Soc. Am.*, 65(4):971–975.
- Malik, M., and A. J. Camm, Eds. 1995. *Heart Rate Variability*. Futura Publishing Co.
- Mallat, S. 1998. *A Wavelet Tour of Signal Processing*. Academic Press, Boston, MA.
- Mammone, R. J., Zhang X, and R. P. Ramachandran. 1996. Robust speaker recognition: A feature based approach. *IEEE Signal Processing Magazine*, pp. 58–71, September.
- Mandelbrot, B. B. 1982. *The Fractal Geometry of Nature*. W. H. Freeman and Company, New York.
- \_\_\_\_\_. 1968. Fractional Brownian motion, fractional Gaussian noises, and applications. *SIAM Review*, 10(4):422–438.
- Manolakis, D., G. Carayannis, and N. Kalouptsidis. 1983. Fast algorithms for discrete-time Wiener filters with optimum lag. *IEEE Trans. Acoustics, Speech and Signal Processing*, 21:168–179, February.
- \_\_\_\_\_, \_\_\_, and \_\_\_\_\_. 1984. Fast design of direct and ladder Wiener filters with linear phase. *IEEE Trans. Circuits and Systems*, 33:1175–1187, October.
- \_\_\_\_\_, T. Conley, et al. 1994. Comparison of visual and IR imagery. *1994 Meeting of IRIS Specialty Group on Targets, Background, and Discrimination*, Monterey, CA.
- \_\_\_\_\_, F. Ling, and J. G. Proakis. 1987. Efficient time-recursive least-squares algorithms for finite-memory adaptive filtering. *IEEE Trans. Circuits and Systems*, 34(4):400–408.
- \_\_\_\_\_, and M. Patel. 1992. Implementation of least squares adaptive filters using multiple processors. *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 2172–2175.
- Markel, J. D., and A. H. Gray, Jr. 1980. *Linear Prediction of Speech*. Springer-Verlag, New York.
- Marple, S. L., Jr. 1987. *Digital Spectral Analysis with Applications*. Prentice Hall, Englewood Cliffs, NJ.
- Martin, R. D., and D. J. Thomson. 1982. Robust-resistant spectral estimation. *Proc. IEEE*, 70(9):1097–1114, September.
- Mathews, V. J. 1991. Adaptive polynomial filters. *IEEE Signal Processing Magazine*, 8(3):10–26, July.
- Matsuoka, T., and T. J. Ulrych. 1984. Phase estimation using the bispectrum. *Proc. IEEE*, 72:1403–1411.
- Mazo, J. E. 1979. On the independence theory of equalizer convergence. *Bell System Tech. J.*, 58:963–993.
- \_\_\_\_\_. 1980. Analysis of decision directed equalizer convergence. *Bell System Tech. J.*, 59(10):1857–1876, December.
- McCoy, E. J., A. T. Walden, and D. B. Percival. 1998. Multitaper spectral estimation of power law processes. *IEEE Trans. Signal Processing*, 46(3):655–668, March.

- McCulloch, J. H. 1986. Simple consistent estimators of stable distribution parameters. *Communications in Statistics—Computation and Simulation*, 15:1109–1136.
- McDonough, R. N., and A. D. Whelen. 1995. *Detection of Signals in Noise*, 2nd ed. Academic Press, San Diego, CA.
- McGarty, T. P. 1974. The effect of interfering signals on the performance of angle of arrival estimators. *IEEE Trans. Aerospace and Electronic Systems*, 10(1):70–77.
- McWhirter, J. G. 1983. Recursive least-squares minimization using a systolic array. In *Real-Time Signal Processing VI*, vol. 431, pp. 105–112.
- \_\_\_\_\_, and I. K. Proudler. 1993. The QR family. In N. Kalouptsidis and S. Theodoridis, Eds., *Adaptive System Identification and Signal Processing Algorithms*, pp. 260–321. Prentice Hall, Englewood Cliffs, NJ.
- \_\_\_\_\_, and T. J. Shepherd. 1989. Systolic array processor for MVDR beamforming. *IEE Proc. Part F: Radar and Signal Processing*, 136(2):75–80.
- Mendel, J. M. 1991. Tutorial on higher-order statistics (spectra) in signal processing and system theory: Theoretical results and some applications. *Proc. IEEE*, 79:278–305.
- Messerschmitt, D. G. 1984. Echo cancellation in speech and data transmission. *IEEE J. Selected Areas in Communications*, 2(2):283–297, March.
- Michiel, H., and K. Laevens. 1997. Teletraffic engineering in a broadband era. *Proc. IEEE*, 85(12): 2007–2033, December.
- Miller, K. S. 1974. *Complex Stochastic Processes: An Introduction to Theory and Application*. Addison-Wesley, Reading, MA.
- Miller, T., L. Potter, and J. McCorkle. 1997. RFI suppression for ultra wideband radar. *IEEE Trans. Aerospace and Electronic Systems*, 33(4):1142–1156, October.
- Mitra, S. K. 1998. *Digital Signal Processing*. McGraw-Hill, New York.
- \_\_\_\_\_, and J. F. Kaiser, Eds. 1993. *Handbook for Digital Signal Processing*. Wiley, New York.
- Montgomery, D. C., and E. A. Peck. 1982. *Introduction to Linear Regression Analysis*. Wiley Series in Probability and Mathematical Statistics, Wiley, New York.
- Monzingo, R. A., and T. W. Miller. 1980. *Introduction to Adaptive Arrays*. Wiley, New York.
- Morf, M. 1974. Fast Algorithms for Multivariable Systems. Ph.D. dissertation, Stanford University, Stanford, CA.
- \_\_\_\_\_, T. Kailath, A. Vieira, and B. Dickinson. 1977. Efficient solution of covariance equations for linear prediction. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-25:423–433, October.
- Morgan, D. R. 1978. Partially adaptive array techniques. *IEEE Trans. Antennas and Propagation*, 26(6):823–833.
- \_\_\_\_\_, and S. G. Kratzer. 1996. On a class of computationally efficient, rapidly converging, generalized NLMS algorithms. *IEEE Signal Processing Letters*, 3(8):245–247, August.
- Moulines, E., P. Duhamel, J. F. Cardoso, and S. Mayrargue. 1995. Subspace methods for the blind identification of multichannel FIR filters. *IEEE Trans. Signal Processing*, 43:516–525, February.
- Muirhead, R. J. 1982. *Aspects of Multivariate Statistical Theory*. Wiley-Interscience, New York.
- Murano, K., et al. 1990. Echo cancellation and applications. *IEEE Communications Magazine*, 28: 49–55.
- Nathanson, F. 1991. *Radar Design Principles*, 2nd ed. McGraw-Hill, New York.
- Netto, S., P. S. R. Diniz, and P. Agathaklis. 1995. Adaptive IIR filtering algorithms for system identification: A general framework. *IEEE Trans. Education*, 38(1):54–66, February.
- Newman, T. G., and P. L. Odell. 1971. *The Generation of Random Variates*. Hafner Publishing Company, New York.
- Ng, S., et al. 1996. The genetic search approach. *IEEE Signal Processing Magazine*, 13(6):38–46, November.
- Nickel, U. 1993. Monopulse estimation with adaptive arrays. *IEE Proc. Part F: Radar, Sonar, and Navigation*, 140(5):303–308.
- Niedermeyer, E., and F. H. Lopes Da Silva, Eds. 1998. *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*, 4th ed. Lippincott, Williams, and Wilkins, Philadelphia, PA.
- Nikias, C. L., and J. M. Mendel. 1993. Signal processing with higher-order spectra. *IEEE Signal Processing Magazine*, 10:10–37.
- \_\_\_\_\_, and A. P. Petropulu. 1993. *Higher-Order Spectra Analysis*. Prentice Hall, Englewood Cliffs, NJ.

- \_\_\_\_\_, and M. R. Raghuveer. 1987. Bispectrum estimation: A digital signal processing framework. *Proc. IEEE*, 75(7):869–891.
- Noble, B., and J. W. Daniel. 1988. *Applied Linear Algebra*, 3rd ed. Prentice Hall, Englewood Cliffs, NJ.
- Nuttall, A. H. 1976. Multivariate linear predictive spectral analysis employing weighted forward and backward averaging: A generalization of Burg's algorithm. *Tech. Rep.*, Naval Underwater Systems Center, New London, CT, October.
- \_\_\_\_\_, and G. C. Carter. 1982. Spectral estimation using combined time and lag weighting. *Proc. IEEE*, 70(9):1115–1125, September.
- Oldham, K. B., and J. Spanier. 1974. *The Fractional Calculus*. Academic Press, New York.
- Oppenheim, A. V., and R. W. Schafer. 1975. *Digital Signal Processing*. Prentice Hall, London.
- \_\_\_\_\_, and \_\_\_\_\_. 1989. *Discrete-Time Signal Processing*, 2nd ed. Prentice Hall, Englewood Cliffs, NJ.
- \_\_\_\_\_, A. S. Willsky, and S. H. Nawab. 1997. *Signals and Systems*. Prentice Hall, Upper Saddle River, NJ.
- Ottersten, B., M. Viberg, and T. Kailath. 1991. Performance analysis of the total least squares ESPRIT algorithm. *IEEE Trans. Signal Processing*, 39(5):1122–1135.
- Ozeki, K., and T. Umeda. 1984. An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties. *Electr. Commun. Japan*, 67-A:19–27.
- Painter, S. 1996. Evidence of non-Gaussian scaling behavior in heterogeneous sedimentary formations. *Water Resources Research*, 32(5):1183–1195.
- Pan, C. T., and R. J. Plemmons. 1989. Least-squares modifications with inverse factorizations: Parallel implications. *Comput. Appl. Math.*, 27:109–127.
- Papoulis, A. 1985. Levinson algorithm, Wold's decomposition and spectrum estimation. *SIAM Rev.*, 27(3):405–441, September.
- \_\_\_\_\_. 1991. *Probability, Random Variables, and Stochastic Processes*, 3rd ed. McGraw-Hill, New York.
- Parzen, E. 1960. *Modern Probability Theory and Its Applications*. Wiley, New York.
- \_\_\_\_\_. 1977. Multiple time series modeling: Determining the order of approximating autoregressive schemes. In P. R. Krishnaiah, Ed., *Multivariate Analysis*, vol. 4, pp. 283–295. North-Holland Publishing Company, New York. Originally published by Academic Press, New York, 1969.
- Paulraj, A., R. Roy, and T. Kailath. 1986. A subspace rotation approach to signal parameter estimation. *Proc. IEEE*, 74:1044–1045.
- Paulraj, A. J., and C. B. Papadias. 1997. Space-time adaptive processing for wireless communications. *IEEE Signal Processing Magazine*, 14(6):49–83.
- Peebles, P. Z., Jr. 1987. *Probability, Random Variables, and Random Signal Principles*, 2nd ed. McGraw-Hill, New York.
- Peng, C.-K., et al. 1993. Long-range anticorrelations and non-Gaussian behavior of the heartbeat. *Physical Rev. Letters*, 70(9):1343–1346.
- Pentland, A. P. 1984. Fractal-based description of natural scenes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:661–674, November.
- Percival, D. B., and A. T. Walden. 1993. *Spectral Analysis for Physical Applications*. Cambridge University Press, New York.
- Pflug, A. L., G. E. Ioup, and R. L. Field. 1992. Properties of higher-order correlation and spectra for bandlimited, deterministic transients. *J. Acoustical Society of America*, 91(2):975–988.
- Pham, D. T., and P. Garrat. 1997. Blind separation of mixture of independent sources through a quasi-maximum likelihood approach. *IEEE Trans. Signal Processing*, 45:1712–1725.
- Picchi, G., and G. Prati. 1987. Blind equalization and carrier recovery using a “stop-and-go” decision-directed algorithm. *IEEE Trans. Communications*, 35:877–887, September.
- Pindyck, R., and D. Rubinfeld. 1998. *Econometric Models and Economic Forecasts*. McGraw-Hill, New York.
- Pisarenko, V. F. 1973. The retrieval of harmonics from a covariance function. *Geophysical J. Royal Astro. Soc.*, 33:347–366.
- Poor, H. V. 1994. *An Introduction to Signal Detection and Estimation*. Springer-Verlag, New York.
- \_\_\_\_\_, and G. W. Wornell, Eds. 1998. *Wireless Communications*. Prentice Hall, Upper Saddle River, NJ.
- Porat, B. 1994. *Digital Processing of Random Signals*. Prentice Hall, Englewood Cliffs, NJ.

- Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. 1992. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge.
- Priestley, M. B. 1981. *Spectral Analysis and Time Series*. Academic Press, London.
- Proakis, J. G. 1995. *Digital Communications*, 3rd ed. McGraw-Hill, New York.
- \_\_\_\_\_, and D. G. Manolakis. 1996. *Digital Signal Processing*, 3rd ed. Prentice Hall, Englewood Cliffs, NJ.
- \_\_\_\_\_, and M. Salehi. 1998. *Contemporary Communication Systems*. PWS Publishing Company, Boston, MA.
- Proudler, I. K., J. G. McWhirter, and T. J. Shepherd. 1989. QRD-based lattice filter algorithms. *Proc. SPIE*, vol. 1152, pp. 56–67.
- Qureshi, S. 1985. Adaptive equalization. *Proc. IEEE*, 73(9):1349–1387, September.
- Rabideau, D. J., and A. O. Steinhardt. 1999. Improved adaptive clutter cancellation through data-adaptive training. *IEEE Trans. Aerospace and Electronic Systems*, 35(3):879–891.
- Rabiner, L., and B. Juang. 1993. *Fundamentals of Speech Recognition*. 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. 1984. A simple method for sampling in-phase and quadrature components. *IEEE Trans. Aerospace and Electronic Systems*, 20(6):821–824.
- \_\_\_\_\_, 1996. VLSI systolic arrays for adaptive nulling. *IEEE Signal Processing Magazine*, 13(4):29–49, July.
- \_\_\_\_\_, and A. O. Steinhardt. 1986. Hyperbolic Householder transformations. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP(34):1589–1602.
- Ramsey, F. L. 1974. Characterization of the partial autocorrelation function. *Ann. Stat.*, 2:1296–1301.
- Rao, B. D., and K. V. S. Hari. 1989. Performance analysis of root-MUSIC. *IEEE Trans. Acoustics, Speech and Signal Processing*, 37(12):1939–1949.
- Rao, T. S., and M. M. Gabr. 1984. *An Introduction to Bispectral Analysis and Bilinear Time Series Models*. Lecture Notes in Statistics, 24, Springer-Verlag, New York.
- Rechtschaffen, A., and A. Kales. 1968. *A Manual of Standardized Technology, Techniques and Scoring System for Sleep Stages of Human Subjects*. Public Health Service, US. Dept. of Health, Education and Welfare, Bethesda, MD.
- Reed, I. S., et al. 1974. Rapid convergence in adaptive arrays. *IEEE Trans. Aerospace and Electronic Systems*, 10(6):853–863.
- Regalia, P. A. 1995. *Adaptive IIR Filtering in Signal Processing and Control*. Marcel Dekker, New York.
- Reidle, K., and A. Sidorenko. 1995. Minimum bias multiple taper spectral estimation. *IEEE Trans. Signal Processing*, 43(1):188–195, January.
- Richmond, C. 1999. Statistics of adaptive nulling and use of the generalized eigen-relation (GER) for modeling inhomogeneities in adaptive processing. *IEEE Trans. Signal Processing*, to appear.
- Richmond, C. D. 1997. Statistical performance analysis of the adaptive sidelobe blanker detection algorithm. *Proc. Asilomar Conf. Signals, Systems, and Computers*, pp. 562–565.
- Rife, D. C., and R. R. Boorstyn. 1974. Single-tone parameter estimation from discrete-time observations. *IEEE Trans. Information Theory*, 20:591–598.
- Rissanen, J. 1978. Modeling by shortest data description. *Automatica*, 14:465–471.
- Robey, F. C., et al. 1992. A CFAR adaptive matched filter. *IEEE Trans. Aerospace and Electronic Systems*, 28(1):208–216.
- Robinson, E. 1984. Statistical pulse compression. *Proc. IEEE*, 72(10):1276–1289, October.
- \_\_\_\_\_, and S. Treitel. 1980. Maximum entropy and the relationship of the partial autocorrelation to the reflection coefficients of a layered system. *IEEE Trans. Acoustics, Speech and Signal Processing*, 28(2):224–235, April.
- 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, Germany.
- Ross, S. 1998. *A First Course in Probability*, 5th ed. Prentice Hall, Upper Saddle River, NJ.
- Roy, R., and T. Kailath. 1989. ESPRIT—estimation of signal parameters via rotational invariance techniques. *IEEE Trans. Acoustics, Speech and Signal Processing*, 37(7):984–995, July.
- \_\_\_\_\_, A. Paulraj, and T. Kailath. 1986. ESPRIT—a subspace rotation approach to estimation of parameters of cisoids in noise. *IEEE Trans. Acoustics, Speech and Signal Processing*, 34(4):1340–1342.

- Roy, S., and J. J. Shynk. 1990. Analysis of the momentum LMS algorithm. *IEEE Trans. Acoustics, Speech and Signal Processing*, 38(12):2088–2098, December.
- Rupp, M. 1993. The behavior of LMS and NLMS algorithms in the presence of spherically invariant processes. *IEEE Trans. Signal Processing*, 41(3):1149–1160, March.
- \_\_\_\_\_. 1995. Bursting in the LMS algorithm. *IEEE Trans. Signal Processing*, 43(10):2414–2417, October.
- Sabins, F. 1987. *Remote Sensing*. W. H. Freeman, New York.
- Saltzberg, B. R. 1968. Intersymbol interference error bounds with application to ideal bandlimited signaling. *IEEE Trans. Information Theory*, IT-14:263–268, July.
- Samorodnitsky, G., and M.S. Taqqu. 1994. *Stable Non-Gaussian Random Processes*. Chapman and Hall, New York.
- Sato, Y. 1975. Two extensional applications of zero-forcing equalization method. *IEEE Trans. Communications*, 23(6):684–687.
- Saul, J. P. 1990. Beat-to-beat variations of heart rate reflect modulation of cardiac autonomic outflow. *Int. Union Physiol. Sci.*, 5:32–37.
- Sayed, A. H., and T. Kailath. 1994. A state-space approach to adaptive RLS filtering. *IEEE Signal Processing Magazine*, 11:18–60.
- \_\_\_\_\_, and \_\_\_\_\_. 1998. Recursive least-squares adaptive filters. In V. Madisetti and D. Williams, Eds., *The Digital Signal Processing Handbook*. CRC Press, New York.
- \_\_\_\_\_, and M. Rupp. 1996. Error-energy bounds for adaptive gradient algorithms. *IEEE Trans. Signal Processing*, 44(8):1982–1989, August.
- \_\_\_\_\_, and \_\_\_\_\_. 1998. Robustness issues in adaptive filtering. In V. Madisetti and D. Williams, Eds., *The Digital Signal Processing Handbook*. CRC Press, New York.
- Scales, L. E. 1985. *Introduction to Nonlinear Optimization*. Springer-Verlag, New York.
- Scharf, L. L. 1991. *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*. Addison-Wesley, Reading, MA.
- \_\_\_\_\_, and L. T. McWhorter. 1997. Adaptive matched substance detectors and adaptive coherence estimators. *Proc. Asilomar Conf. Signals, Systems, and Computers*, pp. 1114–1117.
- Schetzen, M. 1989. *The Volterra and Wiener Theories on Nonlinear Systems*, 2nd ed. Krieger Publishing Company, Malabar, FL.
- Schmidt, R. 1986. Multiple emitter location and signal parameter estimation. *IEEE Trans. Antennas and Propagation*, 34:276–290. Originally appeared in *Proc. RADC, Spectral Estimation Workshop*, pp. 243–258, Rome, NY, 1979.
- Schniter, P. 1998. The BERGULATOR. <http://backhoe.ee.cornell.edu/BURG/>.
- Schreiber, R. J. 1986. Implementation of adaptive array algorithms. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 34(10):1038–1045.
- Schür, I. 1917. On power series which are bounded in the interior of the unit circle. *Journal für die Reine und Angewandte Mathematik*, 147:205–232.
- Sethares, W. A. 1993. The least mean square family. In N. Kalouptsidis and S. Theodoridis, Eds., *Adaptive System Identification and Signal Processing Algorithms*, pp. 84–122. Prentice Hall, Englewood Cliffs, NJ.
- Shalvi, O., and E. Weinstein. 1990. New criteria for blind equalization of non-minimum phase systems (channels). *IEEE Trans. Information Theory*, 36:312–321.
- \_\_\_\_\_, and \_\_\_\_\_. 1994. Universal methods for blind deconvolution. In S. Haykin, Ed., *Blind Deconvolution*. Prentice Hall, Englewood Cliffs, NJ.
- Shao, M., and C. L. Nikias. 1993. Signal processing with fractional lower order moments: Stable processes and their applications. *Proc. IEEE*, 81(7):986–1010, July.
- Shaughnessy, D. O. 1987. *Speech Communication*. Addison-Wesley, Reading, MA.
- Shepherd, T. J., and J. G. McWhirter. 1993. Systolic adaptive beamforming. In S. Haykin and T. J. Shepherd, Eds., *Radar Array Processing*, pp. 153–243. Springer-Verlag, New York.
- Sheriff, R. 1994. *Exploration Seismology*, 2nd ed. Cambridge University Press, Cambridge.
- Sherman, S. M. 1984. *Monopulse Principles and Techniques*. Artech House, Dedham, MA.
- Shiavi, R. 1991. *Introduction to Applied Statistical Signal Analysis*. Irwin, Burr Ridge, IL.
- Shynk, J. J. 1989. Adaptive IIR filtering. *IEEE ASSP Magazine*, 6:4–21.
- \_\_\_\_\_. 1992. Frequency-domain and multirate adaptive filtering. *IEEE Signal Processing Magazine*, 9(1):14–37.
- Siller, C. A., Jr. 1984. Multipath propagation. *IEEE Communications Magazine*, 22(2):6–15.
- Skolnik, M. 1980. *Introduction to Radar Systems*, 2nd ed. McGraw-Hill, New York.

- \_\_\_\_\_, Ed. 1990. *Radar Handbook*, 2nd ed. McGraw-Hill, New York.
- Slock, D. T. M. 1993. On the convergence behavior of the LMS and the normalized LMS algorithms. *IEEE Trans. Signal Processing*, 45(12):2811–2825, September.
- \_\_\_\_\_, and T. Kailath. 1991. Numerically stable fast transversal filters for recursive least-squares adaptive filtering. *IEEE Trans. Signal Processing*, 39(1):92–114.
- \_\_\_\_\_, and \_\_\_\_\_. 1993. Fast transversal RLS algorithms. In N. Kalouptsidis and S. Theodoridis, Eds., *Adaptive System Identification and Signal Processing Algorithms*, pp. 123–190. Prentice Hall, Englewood Cliffs, NJ.
- Solo, V. 1997. The stability of LMS. *IEEE Trans. Signal Processing*, 45(12):3017–3026, December.
- \_\_\_\_\_, 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*, 68:948–963.
- Sorenson, H. 1970. Least-squares estimation: From Gauss to Kalman. *IEEE Spectrum*, 7:63–68.
- Stark, H., and J. W. Woods. 1994. *Probability, Random Processes, and Estimation Theory for Engineers*, 2nd ed. Prentice Hall, Englewood Cliffs, NJ.
- Steele, A. K. 1983. Comparison of directional and derivative constraints for beamformers subject to multiple linear constraints. *IEE Proc. Parts H*, 130(1):41–45.
- Steinhardt, A. O. 1988. Householder transforms in signal processing. *IEEE ASSP Magazine*, 5:4–12.
- \_\_\_\_\_. 1992. Adaptive multisensor detection and estimation. In S. Haykin and A. O. Steinhardt, Eds., *Adaptive Radar Detection and Estimation*, pp. 91–160. Wiley, New York.
- Stewart, G. W. 1973. *Introduction to Matrix Computations*. Academic Press, New York.
- Stewart, R. W., and R. Chapman. 1990. Fast stable Kalman filter algorithms utilizing the square root. *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, pp. 1815–1818.
- Stockham, T., T. Cannon, and R. Ingerbretsen. 1975. Blind deconvolution through digital signal processing. *Proc. IEEE*, 63:678–692.
- Stoer, J., and R. Bulirsch. 1980. *Introduction to Numerical Analysis*. Springer-Verlag, New York.
- Stoica, P., and R. L. Moses. 1997. *Introduction to Spectral Analysis*. Prentice Hall, Upper Saddle River, NJ.
- \_\_\_\_\_, and A. Nehorai. 1989. MUSIC, maximum likelihood, and Cramer-Rao bound. *IEEE Trans. Acoustics, Speech and Signal Processing*, 37(5):720–741.
- Strang, G. 1980. *Linear Algebra and Its Applications*. Academic Press, New York.
- \_\_\_\_\_. 1998. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, Wellesley, MA.
- Stuck, B. W. 1978. Minimum error dispersion linear filtering of scalar symmetric stable processes. *IEEE Trans. Automatic Control*, 23(3):507–509, June.
- \_\_\_\_\_, and B. Kleiner. 1974. A statistical analysis of telephone noise. *Bell Systems Tech. J.*, 53:1262–1320.
- Swami, A. 1998. Non-Gaussian processes. *IEEE Signal Processing Magazine*, 15(5):40–42, September.
- \_\_\_\_\_, G. Giannakis, and G. Zhou. 1997. Bibliography on higher-order statistics. *Signal Processing*, 60(1):65–126, July.
- Takao, K., et al. 1976. An adaptive array under directional constraint. *IEEE Trans. Antennas and Propagation*, 24(9):662–669.
- Tang, K., K. Man, S. Kwong, and Q. He. 1996. Genetic algorithms and their applications. *IEEE Signal Processing Magazine*, 13(6):22–37, November.
- Taqqu, M. S., V. Teverovsky, and W. Willinger. 1995. Estimators for long-range dependence: An empirical study. *Fractals*, 3(4):795–798.
- Theodoridis, S., and N. Kalouptsidis. 1993. Spectral analysis. In N. Kalouptsidis and S. Theodoridis, Eds., *Adaptive System Identification and Signal Processing Algorithms*, pp. 322–387. Prentice Hall, Englewood Cliffs, NJ.
- Therrien, C. W. 1992. *Discrete Random Signals and Statistical Signal Processing*. Prentice Hall, Englewood Cliffs, NJ.
- Thomson, D. J. 1982. Spectrum estimation and harmonic analysis. *Proc. IEEE*, 72(9):1055–1096.
- Tong, L., G. Xu, and T. Kailath. 1994a. 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*, pp. 437–454. IEEE Press, New York.
- \_\_\_\_\_, \_\_\_\_, and \_\_\_\_\_. 1994b. Blind identification and equalization based on second-order statistics: A time-domain approach. *IEEE Trans. Information Theory*, 40(2):340–349, March.

- Treichler, J., and B. Agee. 1983. A new approach to multipath correction of constant modulus signals. *IEEE Trans. Acoustics, Speech and Signal Processing*, 31(2):459–472, April.
- \_\_\_\_\_, I. Fijalkow, and R. Johnson. 1996. Fractionally spaced equalizers: How long should they really be? *IEEE Signal Processing Magazine*, 13(5):65–81, May.
- \_\_\_\_\_, C. R. Johnson, and M. G. Larimore. 1987. *Theory and Design of Adaptive Filters*. Wiley-Interscience, New York.
- \_\_\_\_\_, M. Larimore, and J. Harp. 1998. Practical blind demodulators for high-order QAM signals. *Proc. IEEE*, 86(10):1907–1925, October.
- Trench, W. F. 1964. An algorithm for the inversion of finite Toeplitz matrices. *SIAM J. Appl. Math.*, 12:515–521.
- Tsyplkin, Ja. Z., 1971. *Adaptation and Learning in Automatic Systems*, vol. 73 of *Mathematics in Science and Engineering*. Academic Press, New York.
- \_\_\_\_\_. 1973. *Foundations of the Theory of Learning Systems*, vol. 101 of *Mathematics in Science and Engineering*. Academic Press, New York.
- Tugnait, J. K. 1992. Comments on “New criteria for blind deconvolution of non-minimum phase systems (channels).” *IEEE Trans. Information Theory*, 38(1):210–213, January.
- \_\_\_\_\_. 1998. System identification and tests for non-Gaussianity and linearity. *IEEE Signal Processing Magazine*, 15(5):42–43, September.
- Ulrych, T. J., and R. W. Clayton. 1976. Time series modeling and maximum entropy. *Phys. Earth Planet. Inter.*, 12:188–200, August.
- Vaidyanathan, P., J. Tugan, and A. Kirac. 1997. On the minimum phase property of prediction-error polynomials. *IEEE Signal Processing Letters*, 4(5):126–127, May.
- Van Loan, C. 1995. *Introduction to Scientific Computing*. Prentice Hall, Upper Saddle River, NJ.
- Van Trees, H. L. 1968. *Detection, Estimation, and Modulation Theory*, vol. 1. McGraw-Hill, New York.
- Van Veen, B. 1991. Minimum variance beamforming. In S. Haykin and A. Steinhardt, Eds., *Adaptive Radar Detection and Estimation*, pp. 161–236. Wiley, New York.
- \_\_\_\_\_, and K. M. Buckley. 1988. Beamforming: A versatile approach to spatial filtering. *IEEE Acoustic, Speech, and Signal Processing Magazine*, pp. 4–24, April.
- \_\_\_\_\_, and \_\_\_\_\_. 1998. Beamforming techniques for spatial filtering. In V. Madisetti and D. Williams, Eds., *The Digital Signal Processing Handbook*, CRC Press, New York.
- Verhaegen, M. H. 1989. Round-off error propagation in four generally-applicable, recursive, least-square estimation schemes. *Automatica*, 25:437–444.
- Verhoeckx, N. A. M., H. C. Van Den Elzen, F. A. M. Snijders, and P. J. Van Gerwen. 1979. Digital echo cancellation for baseband data transmission. *IEEE Trans. Acoustics, Speech and Signal Processing*, 27(6), part 2:768–781, December.
- Vervaat, W. 1987. Properties of general self-similar processes. *Bull. Int. Statist. Inst.*, 52(4):199–216.
- Viswanathan, R., and J. Makhoul. 1975. Quantization properties of transmission parameters in linear predictive systems. *IEEE Trans. Acoustics, Speech and Signal Processing*, 23(3):309–321, June.
- Walpole, R. E., R. H. Myers, and S. L. Myers. 1998. *Probability and Statistics for Engineers and Scientists*, 6th ed. Prentice Hall, Upper Saddle River, NJ.
- Ward, J. 1994. Space-time adaptive processing for airborne radar. *Tech. Rep. TR-1015*, MIT Lincoln Laboratory, Lexington, MA.
- \_\_\_\_\_. 1995. Space-time adaptive processing for airborne radar. *Proc. Int. Con. Acoustics, Speech and Signal Processing*, pp. 2809–2812.
- \_\_\_\_\_. 1996. Cramér-Rao bounds for target angle and Doppler estimation with space-time adaptive processing radar. *Proc. Asilomar Conf. Signals, Systems, and Computers*, pp. 1198–1202.
- \_\_\_\_\_, and A. O. Steinhardt. 1994. Multiwindow post-Doppler space-time adaptive processing. *Proc. Seventh Workshop on Statistical Signal and Array Processing*, pp. 461–464.
- Watkins, D. S. 1991. *Fundamentals of Matrix Computations*. Wiley, New York.
- Weinstein, S. B. 1977. Echo cancellation in the telephone network. *IEEE Communications Magazine*, 15(1):9–15, January.
- Weiss, G. 1975. Time-reversibility of linear stochastic processes. *J. Appl. Probability*, 12:831–836.
- Welch, P. W. 1967. The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Trans. Audio Electroacoustics*, 15(2):70–76.
- Widrow, B., and M. E. Hoff, Jr. 1960. Adaptive switching circuits. *IRE WESCON Conv. Rec.*, part 4:96–104.

- \_\_\_\_ and S. D. Sterns. 1985. *Adaptive Signal Processing*. Prentice Hall, Englewood Cliffs, NJ.
- \_\_\_\_ and E. Walach. 1994. *Adaptive Inverse Control*. Prentice Hall, Englewood Cliffs, NJ.
- \_\_\_\_ et al. 1975. Adaptive noise cancelling: Principles and applications. *Proc. IEEE*, 63:1692–1716.
- \_\_\_\_ et al. 1976. Stationary and nonstationary learning characteristics of the LMS adaptive filter. *Proc. IEEE*, 64:1151–1162.
- Wiener, N. 1949. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series, with Engineering Applications*. MIT Press, Cambridge, MA.
- Wiggins, R. A. 1978. On minimum entropy deconvolution. *Geoexploration*, 16:21–35.
- Williamson, G. 1998. Adaptive IIR filters. In V. Madisetti and D. Williams, Eds., *The Digital Signal Processing Handbook*. CRC Press, New York.
- Wong, E. 1971. *Stochastic Processes in Information and Dynamical Systems*. McGraw-Hill, New York.
- Wornell, G. W. 1993. Wavelet-based representation for the  $1/f$  family of fractal processes. *Proc. IEEE*, 81(10):1428–1450, October.
- \_\_\_\_\_. 1996. *Signal Processing with Fractals: A Wavelet-Based Approach*. Prentice Hall, Englewood Cliffs, NJ.
- Yang, B., and J. F. Böhme. 1992. Rotation-based RLS algorithms: Unified derivations, numerical properties and parallel implementations. *IEEE Trans. Signal Processing*, 40:1151–1167.
- Yokoya, N., K. Yamamoto, and N. Funakubo. 1989. Fractal based analysis and interpolation of 3D natural surface shapes and their application to terrain modeling. *Computer Vision, Graphics and Image Processing*, 46:284–302.
- Zadeh, L. A., and J. R. Ragazzini. 1950. An extension of Wiener's theory of prediction. *J. Appl. Phys.*, 21:645–655, July.
- Zatman, M. 1998. How narrow is narrowband? *IEE Proc. Part F: Radar, Sonar and Navigation*, 145(2):85–91.
- Zoltowski, M. 1992. Beamspace ML bearing estimation for adaptive phased array radar. In S. Haykin and A. Steinhardt, Eds., *Adaptive Radar Detection and Estimation*, pp. 237–332. Wiley, New York.



---

# Index

a posteriori RLS algorithms, 550  
a posteriori type adaptive algorithms, 512  
a priori adaptive filter, 594  
a priori RLS algorithms, 549  
a priori type adaptive algorithms, 511  
acoustic echo cancelation, 17  
acquisition, 514  
acquisition mode, 507  
active noise control (ANC), 22  
actual estimates, 511  
adaptation algorithm, 24  
adaptation gain vector, 550  
adaptive algorithm, 511  
adaptive algorithm tracking, 590  
adaptive array, 27, 641  
adaptive beamforming, 27, 641, 659  
  weight vector norm constraint, 673  
adaptive channel matching, 628  
adaptive degrees of freedom, 673  
adaptive detection, 660, 686  
adaptive equalization, 541  
adaptive equalizers  
  blind or self-recovery mode, 503  
  decision-directed mode, 503  
  training mode, 503  
adaptive filter, 16, 510  
  adaptation algorithm, 508  
  criterion of performance, 507  
  features, 507  
  filtering structure, 507  
  goal, 499  
  IIR, 608  
  key feature, 499, 500  
  performance, 499  
  unsupervised, 703  
adaptive filtering, 16  
adaptive matched filter (AMF), 660  
adaptive matched filter (AMF) normalization, 644, 679  
adaptive signal processing, 1  
affine projection algorithms, 547  
Akaike information criterion (AIC), 458  
algorithm of Schür, 368

algorithms  
  order-recursive, 334  
  order-updating, 334  
  time-recursive, 334  
  time-updating, 334  
  fixed-order, 334  
aliasing, 41  
all-pass systems, 56  
  decomposition, 57  
  properties, 57  
all-pole (AP) model estimation, 449–462  
  Burg’s lattice method, 459–460  
  direct structures, 449–458  
  frequency domain interpretation, 455  
  Itakura-Saito lattice method, 460  
  lattice structures, 458–460  
  least squares, 451  
  maximum entropy method, 460–461  
  modified covariance method, 454  
all-pole (AP) signal models, 154  
all-pole models  
  autocorrelation, 158  
  cepstrum, 185  
  correlation matching, 161  
  equivalent representations, 162  
  first-order, 165  
  impulse response, 157  
  linear prediction interpretation, 163  
  minimum-phase conditions, 163  
  PACS, 162  
  partial autocorrelation, 162  
  pole locations, 163  
  second-order, 167  
  spectrum, 162  
all-zero (AZ) signal models, 154  
  autocorrelation, 173  
  cepstrum, 188  
  first-order, 174  
  impulse response, 172  
  impulse train excitations, 173  
  partial autocorrelation, 173  
  second-order, 176  
  spectrum, 173

- alternative adaptation gain vector, 550  
 amplitude distribution, 8  
 amplitude-domain LS solutions, 439  
 analysis filter, 152  
 angle estimation, 678–683  
     maximum likelihood, 679–680  
 angle normalized errors, 585  
 angle of arrival, 625  
 antialiasing filter, 197  
 aperture, 622  
 ARMA (P,Q) models, 445  
 array element spacing, 635  
 array gain, 635  
 array output signal-to-noise ratio, 633  
 array pre-steering, 651  
 array processing, 25  
 array response vector, 629, 630  
     for ULA, 629  
 array signal model, 627  
 array signal-to-noise ratio, 633  
 array snapshot, 627  
 autocorrelation, estimation, 209  
 autocorrelation matrix, 85  
 autocorrelation method, 408  
 autocorrelation sequence, 53, 100  
 autocovariance matrix, 86  
 autocovariance sequence, 100  
 autoregressive (AR) signal models, 154  
 autoregressive fractionally integrated  
     moving-average (ARFIMA) models, 722  
 autoregressive integrated moving-average  
     (ARIMA) models, 184  
 autoregressive models, 164  
 autoregressive moving-average models, 179  
 autoregressive moving-average (ARMA) signal  
     models, 154  
 azimuth angle, 624
- backward linear prediction, 289  
 backward prediction, Levinson recursion, 348  
 bandpass signal, 626  
 baud interval, 311  
 baud rate, 311  
 beam response, 632  
 beamforming, 25, 631  
 beamforming gain, 633  
     of spatial matched filter, 635  
 beamforming resolution, 636  
 beampattern, 25, 27, 632  
 beamspace, 651  
 beamsplitting, 678, 681  
 beamwidth, 28, 635  
 best linear unbiased estimator (BLUE), 405, 752  
 bispectrum, 693  
 blind deconvolution, 306, 697  
 blind equalization  
     cyclostationary methods, 704  
     HOS-based methods, 704  
 blind equalizers, 702  
     Bussgang algorithms, 706  
     constant-modulus algorithm, 707  
     fractionally spaced (FSE), 713
- Godard algorithms, 707  
 Sato algorithms, 706  
 symbol rate, 705  
 blind interval, 669  
 block adaptive filtering, 511  
 block adaptive methods, 659  
 block LMS, 546  
 Brownian motion, 727  
 Bussgang processes, 706
- Capon's method, 472  
 carrier frequency, 624  
 Cauchy-Schwarz inequality, 135  
 central limit theorem (CLT), 90, 95  
 centrosymmetric matrix, 759  
 cepstral distance, 188  
 cepstrum, 63, 152  
     all-pole models, 185  
     all-zero models, 188  
     pole-zero models, 184  
 channel equalization, 20  
 characteristic exponent, 94  
 characteristic function, 79  
 Chebyshev's inequality, 79  
 chi-squared distribution, 140  
 Cholesky decomposition, 278, 560  
 close to Toeplitz, 408  
 clutter, 7, 27  
 clutter cancellation, 683  
 coefficient vector, 265, 279  
 coherence, 113  
 coherent output PSD, 242  
 coloring filter, 152  
 complex coherence function, 238  
 complex cross-spectral density, 113  
 complex envelope, 45  
 complex spectral density, 54, 113  
 condition number, 762  
 conditional covariance, 405  
 conditional density, 274  
 conditional mean, 405  
 cone angle, 625  
 confidence interval, 136  
 confidence level, 136  
 constrained optimization, 644, 650  
 conventional beamforming, 27, 634  
 conventional RLS, 548  
 conventional RLS algorithm, 552  
     initialization, 554  
 convergence everywhere, 513  
 convergence in MS sense, 513  
 convergence mode, 507  
 convergence with probability, 1, 513  
 conversion factor, 550  
 correlation, 86  
 correlation, properties, 114  
 correlation coefficient, 86  
 correlation matrix  
     stationary processes, 123  
     random processes, 123  
 correlation matrix properties, 120

- correlation sequence, 53  
 cospectrum, 238  
     estimation, 240  
 covariance filtering-type algorithms, 572  
 covariance method, 408  
 Cramer-Rao bound (CRB) on angle accuracy, 680  
 Cramer-Rao lower bound, 135  
 criterion autoregressive transfer (CAT) function,  
     458  
 criterion of performance (COP), 24  
 cross-amplitude spectrum, 238  
     estimation, 240  
 cross-correlation matrix, 87  
 cross-correlation sequence, 100  
 cross-covariance matrix, 87  
 cross-covariance sequence, 100  
 cross-periodogram, 239  
 cross-power spectral density, 113  
 cross-validation, 449  
 cumulant generating functions, 80  
 cumulant spectra, 692  
 cumulants, 80, 692  
 cumulative distribution function (cdf), 76  
 cumulative-sum method, 735
- data, 261  
 data matrix  
     full-windowing, 450  
     no-windowing, 450  
 data window, 203  
 data-adaptive spectrum estimation, 472  
 decomposition of the covariance rule, 406  
 deconvolution, 306  
 degree of nonstationarity, 595  
 desired response, 261  
 deterministic signals, 2  
 DFT sampling theorem, 201  
 DFT; *see* Discrete Fourier transform  
 diagonally loaded sample correlation matrix, 666,  
     668  
 difference beamformer, 680  
 difference equations, 49  
 digital in-phase/quadrature (DIQ), 627  
 direct error extraction, 424  
 directivity, 622  
 Dirichlet conditions, 38  
 discrete cosine transform, 547  
 discrete Fourier transform, 42  
 discrete fractional Gaussian noise (DFGN), 733  
     generation, 735  
     memory, 734  
     self-similarity, 734  
 discrete Karhunen-Loeve transform (DKLT), 130  
 discrete prolate spheroidal sequences (DPSSs), 247  
 discrete spectrum, 37  
 discrete wavelet transform, 547  
 discrete-time fractional Gaussian noise, 719  
 discrete-time fractional pole noise, 719  
 discrete-time signal, 1  
 discrete-time stochastic processes, 97  
 discrete-time systems, 47  
 dispersion, 656, 707
- dispersion matrix, 657  
 dispertion, 502  
 displacement rank, 389  
 Dolph-Chebyshev taper, 639, 649, 650  
 doppler effect, 7  
 Durbin algorithm; *see* Levinson-Durbin algorithm
- echo, 500  
 echo cancelation, 538  
 echo canceler, 501, 539  
     adaptive, 502  
     fixed, 501  
 echo cancelation, communications, 500  
 echo path, 17, 501, 539  
 echo return loss enhancement, 502  
 echo suppressor, 501  
 echoes, 17  
     acoustic, 500  
     electrical, 500  
     line, 500  
 eigenbeam, 647  
 eigenfilters, 319  
 eigenmatrix, 122  
 eigenvalue spread, 124  
 eigenvalues, 120  
 eigenvector method, 484–485  
 eigenvectors, 120  
 electrophysiological signals, 4  
 elevation angle, 624  
 empirical autocorrelation, 10  
 energy spectrum, 38, 39  
 equalization, 310  
     data communications, 502  
 equalizers  
     fractionally spaced (FSE), 709  
     MMSE FSE, 713  
 equation-error method, 463  
 error performance surface, 266  
 error signal, 262  
 ESPRIT algorithm, 488–493  
     least squares method, 491–492  
     total least squares method, 492–493  
 estimation error, 515  
 estimation misadjustment, 596  
 estimation noise, 596  
 estimator, 133  
     bias, 134  
     consistency, 136  
     MSE, 134  
     variance, 134  
 Euclidean norm, 756  
 Euclidean space, 755  
 evolutionary model, 592  
 exchange matrix, 758  
 excess MSE, 269, 596  
 expected value, 77  
 exponential convergence, 518  
 exponential memory, 593  
 exponential sequence, 35  
 exponentially growing window, 591  
 extended QR-RLS algorithm, 565

- extended Schür algorithm, 372  
 extrapolation, 198  
 eye pattern, 312
- FEST algorithm, 576, 578  
 far field assumption, 624  
 far-end echo, 538  
 fast fixed-order RLS, 574  
 fast Fourier transform (FFT), 43  
 fast Kalman algorithm, 575, 576  
 fast order-recursive RLS, 574  
 fast RLS algorithms, 573  
 features of adaptive filters, 23  
 FFT; *see* fast Fourier transform  
 filtering structure, 24  
 final prediction error (FPE) criterion, 458  
 finite impulse response (FIR) filter, 50  
 FIR; *see* finite impulse response  
 fixed-length sliding window, 591  
 forgetting factor, 548  
 formant frequencies, 4  
 forward linear prediction, 288  
 forward prediction, Levinson recursion, 349  
 forward-backward linear prediction, 413  
 forward/backward LS all-pole modeling, 467  
 forward-backward predictors, 454  
 Fourier series, 37  
 Fourier transform, 37, 38  
 fractal models, 14  
 fractals, 15  
 fractional autoregressive integrated moving-average models, 14  
 fractional bandwidth, 628, 656  
 fractional Brownian motion, 15, 730  
 fractional differentiator, 719  
 fractional Gaussian noise, 15  
 fractional integrator, 14, 719, 732  
 fractional pole processes  
     Gaussian, 723  
     S&S, 723  
 fractional pole systems, continuous-time, 732  
 fractional pole-zero models, 14, 721  
 fractional unit-pole models  
     autocorrelation, 718  
     definition, 716  
     impulse response, 717  
     memory, 719  
     minimum-phase, 718  
     partial autocorrelation, 719  
     spectrum, 718  
 fractionally differenced Gaussian noise, 14  
 fractionally spaced equalizer (FSE), 315  
 frequency analysis, 198  
 frequency estimation, 478–493  
     eigenvector method, 484–485  
     ESPRIT algorithm, 488–493  
     MUSIC algorithm, 484–485  
     Pisarenko harmonic decomposition, 482–484  
     root-MUSIC algorithm, 485  
 frequency response, estimation, 241  
 frequency response function, 49  
 Frobenius norm, 433, 757
- Frost's algorithm, 671  
 FTF algorithm, 577  
 full-duplex data transmission, 538  
 fundamental frequency, 37
- Gaussian moment factorization property, 529  
 general linear process model, 151  
 generalized sidelobe canceler (GSC), 650, 670  
 genetic optimization algorithms, 608  
 geophysical signals, 5  
 Gershgorin circles theorem, 530  
 Givens inverse QR-RLS algorithm, 569, 571  
 Givens QR-RLS algorithm, 566, 568  
 Givens rotation, 427  
 gradient, 747  
 Gram-Schmidt, classical algorithm, 346  
 Gram-Schmidt orthogonalization, 345  
     classical, 429  
     modified, 430  
 grating lobes, 636  
 growing memory, 548
- Hankel matrix, 759  
 harmonic fractional pole models, 723  
 harmonic fractional pole-zero models, 723  
 harmonic model(s), 184, 478–482  
 harmonic processes, 110  
 harmonic spectra, 39  
 harmonizable representation, 733  
 Haussdorff dimension, 732  
 Hermitian matrix  
     negative definite, 760  
     negative semidefinite, 760  
     positive definite, 760  
     positive semidefinite, 760  
 Hessian matrix, 524  
 high resolution spectral estimator, 472  
 higher-order moments  
     definitions, 691  
     linear signal models, 695  
     linear system response, 693  
 higher-order statistics, 691  
 Householder reflections, 425  
 hybrid couplers, 538  
 hybrids, 500
- idempotent matrix, 402  
 IIR; *see* infinite impulse response  
 IIR adaptive filters, 608  
 implementation complexity, 515  
 impulse response, 47  
 incremental filter, 595  
 independence assumption, 528  
 index of stability; *see* characteristic exponent  
 infinite impulse response (IIR) filter, 51  
 infinitely divisible distributions, 95  
 information filtering-type algorithms, 572  
 initialization, CRLS algorithm, 554  
 inner product, 755  
 innovations, 125

- innovations representation, 151
  - eigendecomposition approach, 129
  - LDU triangularization approach, 129
  - UDL triangularization approach, 129
- in-phase component, 45
- input data vector, 265, 279
- interference, 7, 638
  - interference mitigation, 27
  - interference signal, 642
  - interference subspace, 647
  - interference-plus-noise correlation matrix, 642, 647
- intersymbol interference (ISI), 20, 310
- inverse filtering, 306
- inverse QR-RLS algorithm, 566
- inverse Schur algorithm, 374
- inverse system, 54
- invertibility, 54
- isotropic transformation, 126
- Itakura-Saito (IS) distortion measure, 457
- Itakura-Saito distance measure, 462
  
- Jacobian, 87
- jammer, 641, 645
- jamming, 27
- joint cumulative distribution function, 83
- joint ergodicity, 107
- joint signal analysis, 11
  
- Kalman filter, 378, 592
  - algorithm, 384
  - gain matrix, 382
  - measurement model, 381
  - observation error, 381
  - observation model, 381
  - signal model, 381
  - state transition matrix, 381
  - state vector, 381
- Kalman gain matrix, 382
- Karhunen-Loeve transform, 129
- Kolmogorov-Szegö formula, 305
- Kullback-Leibler distance, 458
- kurtosis, 79
  
- lag error, 515
- lag misadjustment, 596
- lag noise, 596
- Lagrange multipliers, 749
- Lagrangian function, 750
- lattice filters, 64
  - all-pass, 70
  - all-pole, 68
  - all-zero, 65
- lattice parameter conversion
  - direct-to-autocorrelation, 367
  - direct-to-lattice, 366
  - lattice-to-autocorrelation, 367
  - lattice-to-direct, 366
- lattice parameter estimation
  - Burg's method, 459–460
  - Itakura-Saito method, 460
  
- lattice-ladder optimization, 365
- lattice-ladder structure, 351
- law of iterated expectations, 406
- $LDL^H$  decomposition, 274
- leading principal submatrix, 335
- leakage, 204
- leaky LMS, 546
- learning curve, 514, 519
- least-squares
  - amplitude-domain techniques, 424
  - comparison with MSE estimation, 419
  - data adaptive estimators, 438
  - FIR filters, 420
  - linear prediction, 411, 420
  - minimum-norm solution, 435
  - normal equations solution, 416
  - orthogonalization techniques, 422
  - power-domain techniques, 424
  - rank-deficient, 437
  - regularization, 438
  - regularized solution, 438
  - signal estimation, 411
  - square root methods, 424
  - SVD solution, 434
- least-mean-square (LMS) algorithm, 524
- least-squares error (LSE) estimation, 395
- least-squares FIR filters, 406
- least-squares inverse filters, 409
- least-squares principle, 395
- left singular vectors, 432
- Levenberg-Marquard regularization, 465
- Levinson, 278
- Levinson algorithm, 353, 358
- Levinson recursion, 338
- Levinson-Durbin algorithm, 356
- Levy distribution, 10, 95
- Levy stable motion, 739
- likelihood variable, 551
- line spectrum, 37, 461, 479
- linear equalizers, 314
- linear LSE estimation, 396
  - data records, 397
  - estimation space, 399
  - normal equations, 399
  - snapshots, 397
  - statistical properties, 403
  - uniqueness, 401
  - weighted, 403
- linear mean square error estimation, 264
- linear MMSE estimator, 265
  - derivation, 268
- linear prediction, 21, 286
  - backward, 289
  - forward, 288
- linear prediction coding (LPC), 21, 470, 503
- linear random signal model, 12, 151
- linear signal estimation, 286
- linear systems
  - frequency-domain analysis, 117
  - input-output cross-correlation, 116
  - output correlation
  - output mean value, 116

- output power, 117, 118
- random inputs, 115
- scale-invariant, 732
- time-domain analysis, 115
- linearly constrained minimum variance beamformer, 672
- LMS
  - adaptation in stationary SOE, 526
  - digital residual error, 546
  - disturbances, 545
  - finite precision effects, 546
  - leakage, 546
  - method of ordinary differential equations, 536
  - misadjustment, 597
  - MSD, 598
  - rate of convergence, 534
  - robustness, 543
  - speed versus quality of adaptation, 534
  - stability, 534
  - steady-state excess MSE, 532, 534
  - stochastic approximation approach, 536
  - tap input power, 535
  - transient MSE, 532
- LMS algorithm, 526, 533
- log likelihood function, 135
- long memory, estimation, 739
- long memory processes, 119
- long-tailed distributions, 107
- long-term persistence, 722
- look direction, 634
  
- magnitude square coherence, 113
- magnitude-squared coherence, 238
  - estimation, 240
- Mahalanobis distance, 126
- mainbeam, 635
- marginal density function, 84
- Markov estimator, 405
- matched filters, 319
- mathematical expectation, 77
- matrix, 756
  - amplification power, 757
  - centrosymmetric matrix, 759
  - column space of, 433
  - condition number, 436, 762
  - dynamic range, 758
  - exchange matrix, 758
  - Hankel matrix, 759
  - Hermitian matrix, 760
  - Hermitian, 758
  - lower triangular, 758
  - null space of, 433
  - numerical rank of, 437
  - orthogonal matrix, 762
  - partition of a matrix, 759
  - persymmetric matrix, 758
  - positive definite matrix, 764–765
  - range space of, 433
  - relative error, 762
  - row space of, 433
  - square matrix, 760
  - symmetric, 758
  - Toeplitz, 758
  - unitary transformation, 762
- unitary matrix, 762
- upper triangular, 758
- well (ill) conditioned matrix, 762
- matrix factorization lemma, 563
- matrix inversion by partitioning lemma, 336
- matrix inversion lemma, 745
- matrix norm, 757
- maximum entropy method, 460–461
- maximum likelihood estimate (MLE), 136
- maximum-phase, 293
- maximum-phase system, 56
- mean square deviation (MSD), 513, 596
- mean square error (MSE) criterion, 264
- mean value, 77
- mean vector, 85
- Mercer’s theorem, 122
- minimax criterion, 545
- minimum description length (MDL) criterion, 458
- minimum mean-square error (MMSE), 678
- minimum MSE equalizer, 316
- minimum-variance estimator, 405
- minimum-norm method, 485–488
- minimum-phase, 293
  - test, 69
- minimum-phase system(s), 55
  - properties, 61
- minimum-variance spectrum estimation, 471–478
  - implementation, 474–477
  - relationship to all-pole spectrum estimation 477–478
  - theory, 472–474
- minimum-variance distortionless response (MVDR), 644
- misadjustment, 514, 596
- mixed processes, 156
- mixed spectra, 39
- mixed-phase system, 56
- MMSE filtering, 652
- model, 11
- model fitting, 447
- model order selection criteria, 457–458
  - Akaike information criterion, 458
  - criterion autoregressive transfer function, 458
  - final prediction order criterion, 458
  - minimum description length criterion, 458
- modified covariance method, 414
- modulation, 625
- moment generating function, 80
- moments, 78
  - central, 78
- monopulse radar, 680
- Moore-Penrose conditions, 435
- Moore-Penrose generalized inverse, 402
- moving-average models, 173
- moving-average (MA) signal models, 154
- multichannel adaptive filters, 608
- multiple linear constraints, 672
- MUSIC algorithm, 484–485
- MVDR beamformer, 644, 650
  
- narrowband assumption, 628, 656
- narrowband interference cancelation, 414
- narrowband steering vector, 656

- natural mode, 518  
near to Toeplitz, 408  
near-end echo, 538  
Newton's type algorithms, 523  
noise cancelation, 505  
noise subspace, 480–481, 647  
nonharmonic spectra, 39  
nonlinear adaptive filters, 608  
nonparametric models, 12  
nonrecursive system representation, 150  
normal equations, 269  
  solution, 274  
normalized cross-correlation, 100  
normalized frequency, 40  
normalized LMS, 535  
normalized LMS algorithm, 526  
normalized MSE, 269  
nulls, 27  
numerical accuracy, 516  
numerical inconsistencies, 555  
numerical stability, 516  
Nyquist rate, 42  
Nyquist's criterion, 311
- observations, 261  
optimal reduced-basis representation, 131  
optimum a priori error, 594  
optimum array processing, 641  
optimum beamformer, 642, 643, 644  
  effect of bandwidth, 656  
  eigenanalysis, 646  
  interference cancelation performance, 648  
  low sidelobe, 650  
  signal mismatch loss (desired signal not in correlation matrix), 654  
  signal mismatch loss from desired signal in correlation matrix, 655  
  spatial null depth, 648  
  tapering, 649  
optimum beamforming weight vector, 643  
optimum estimate  
  order decomposition, 344  
  order-recursive computation, 340  
  order-recursive structure, 342  
  orthogonal structure, 345  
optimum estimator, 262  
optimum filters, 509  
  design, 387  
  frequency-domain interpretation, 285  
  implementation, 388  
optimum FIR filters, 281, 295  
  ladder structure, 362  
  lattice structures, 361  
  order-recursive algorithms, 347  
optimum IIR filters  
  causal, 297  
  factorization, 298  
  irreducible MMSE, 299  
  noise filtering, 300  
  noncausal, 296  
  regular input processes, 297  
  white input processes, 297
- optimum learning, 514  
optimum nesting, 335  
optimum signal processing, 1  
optimum signal processor, 262  
optimum space-time weight vector, 685  
orthogonal matrix, 762  
orthogonal transformation, 125  
orthogonal vectors, 755  
orthogonality principle, 273  
overdetermined LS problem, 402
- p* norm, 755  
Paley-Wiener theorem, 63  
parameter vector, 265  
parametric models, 12  
parametric signal model, 150  
parametric spectrum estimation, 467–470  
Parseval's relation, 39  
partial correlation, 344  
partial correlation coefficients, 364  
partially adaptive arrays, 673–676  
  beamspace, 675  
  subarrays, 675  
partition of a matrix, 759  
peak distortion, 316  
periodic extension, 198  
periodic random sequences, 132  
periodogram  
  definition, 212  
  filter bank interpretation, 213  
  modified, 212  
persymmetric matrix, 758  
persistence, 731  
phase spectrum, 238  
  estimation, 240  
Pisarenko harmonic decomposition, 482–484  
plane wave, 624  
point estimate, 133  
poles, 44  
pole-zero (PZ) model estimation, 447, 462–467  
  equation error method, 463  
  known excitation, 463  
  nonlinear least squares, 464  
  unknown excitation, 463  
pole-zero (PZ) model selection, 446  
pole-zero (PZ) model validation, 447  
  autocorrelation test, 448  
  power spectrum test, 448  
pole-zero (PZ) modeling  
  applications, 467–471  
  speech modeling, 470–471  
pole-zero models  
  autocorrelation, 177  
  cepstrum, 184  
  first-order, 180  
  impulse response, 177  
  mixed representations, 189  
  partial autocorrelation, 179  
  poles on unit circle, 182  
  spectrum, 179  
  summary and dualities, 181  
pole-zero (PZ) signal modeling, 445  
pole-zero (PZ) signal models, 153, 154, 445

- pole-zero (PZ) spectrum estimation, 467–470
- positive definite matrix, 764
  - properties of, 765
- power cross-spectrum estimation, 252
- power spectral density (PSD), 110
  - properties, 109, 114
- power spectrum, 37, 38
- power spectrum estimation, 195
  - Blackman-Tukey method, 223
  - multitaper method, 246
  - nonparametric techniques, 195
  - parametric techniques, 195
  - practical considerations, 232
  - Welch-Bartlett method, 227
- power transfer factor, 153
- power-domain LS solutions, 439
- predictable processes, 306
- predicted estimates, 511
- prewhitening, 468
- prewindowed RLS FIR filters, 573
- prewindowing, 408
- primary input, 505
- primary signal, 22
- principal component analysis, 270
- principal coordinate system, 271
- principle of orthogonality, 273
- probability density function(pdf), 76
- probability mass function (pmf), 77
- projection matrix, 402, 480–481
- propagating wave, 623
- property restoral approach, 507
- PSD; *see* power spectral density
- pseudo random numbers, 83
- pseudo-inverse, 402
- pseudospectrum
  - eigenvector method, 485
  - minimum-norm method, 487
  - MUSIC, 484
  - Pisarenko harmonic decomposition, 482
- pulse repetition frequency, 7
  
- QR decomposition, 423, 474, 560, 667
  - thin, 423
- QR-decomposition RLS, 574
- QR-RLS algorithm, 564
- quadratic constraints, 673
- quadrature component, 45
- quadrature spectrum, 238
  - estimation, 240
- quality of adaptation, 515
- quantization, 503
- quantization error, 504
- quiescent response, 647
  
- radar signals, 7
- raised cosine filters, 312
- random fractals, 15, 725
- random midpoint replacement method, 737
- random process(es)
  - ensemble, 98
  - ergodic, 105
  - ergodic in correlation, 106
  
- ergodic in the mean, 106
- Gaussian, 101
- independent, 101
- independent increment, 101
- innovations representation, 151
- jointly wide-sense stationary, 103
- locally stationary, 105
- Markov, 104
- orthogonal, 101
- predictable, 99
- realization, 98
- stationary, 102
- uncorrelated, 101
- wide sense cyclostationary, 101
- wide-sense periodic, 101
- wide-sense stationary, 102
- random sequences, 98
- random signal memory, 118
  - correlation length, 119
- random signal variability, 107
- random signals, 1, 3, 75
  - generation, 155
- random variable(s), 75
  - Cauchy, 82
  - complex, 84
  - continuous, 76
  - discrete, 76
  - independent, 84
  - normal or Gaussian, 82
  - orthogonal, 86
  - sums, 90
  - uniformly distributed, 81
- random vectors, 83
  - complex, 84
  - decorrelation, 343
  - innovations representation, 125
  - linear transformations, 87
  - linearly equivalent, 343
  - normal, 88
- range, 7
- rate of convergence, 515, 519
- rational models, 13
- Rayleigh's quotient, 121, 322
- receiver, 626
- rectangularly growing memory, 593
- recursive least-squares (RLS), 548
  - methods for beamforming, 670
- recursive representation, 151
- reference input, 505
- reference signal, 22
- region of convergence (ROC), 43
- reflection coefficients, 362, 364
- regression function, 396
- regression vector, 396
- relationship between minimum-variance and all-pole spectrum estimation, 477–478
- relative error, 762
- reverberations, 17
- right singular vectors, 432
- RLS algorithm classification, 589
- RLS lattice-ladder
  - a posteriori, 582
  - a priori, 583

- a priori with error feedback, 584
- Givens rotation-based, 585
- square-root free Givens, 588
- square-root Givens, 588
- RLS lattice-ladder algorithms, 580
- RLS misadjustment, 599
- root method, 62
- root-MUSIC, 485
- rotational invariance, 489–490
  
- sample autocorrelation sequence, 210
- sample correlation matrix, 474, 481, 660
- sample matrix inversion (SMI) adaptive beamformer, 660
  - beam response, 666
  - desired signal present, 665
  - diagonal loading, 665, 666
  - implementation, 667
  - sidelobe levels, 661–665
  - training issues, 665
- sample matrix inversion (SMI) loss, 660, 661
- sample mean, 136
- sample support, 660
- sample variance, 139
- sample-by-sample adaptive methods, 669
- sampling, 503
  - sampling distribution, 134, 137
  - sampling frequency, 40
  - sampling period, 40
  - sampling rate, 40
  - sampling theorem, 42
    - bandpass, 45
    - DFT, 201
- scale-invariance, 725
- scale-invariant, 15
- scatter plot, 1
- seasonal time series, 184
- second characteristic function, 80
- self-similar, 15, 725
  - with stationary increments
  - strict-sense, 728
  - wide-sense, 728
- self-similarity index, 726
- semivariogram, 728
- sensor thermal noise, 627
- Shannon number, 248
- Sherman-Morrison's formula, 745
- shift-invariance, 347
- short memory processes, 119
- short-memory behavior, 155
- sidelobe canceler, 28, 676–678
- sidelobe target, 649
- sidelobes, 635
- signal analysis, 3
- signal filtering, 3
- signal mismatch, 652
- signal model, 34
- signal modeling, 11, 150
- signal operating environment (SOE), 24, 507
- signal prediction, 21
- signal subspace, 480–481
- signal(s), 33
  - causal, 36
  - classification, 35
- complex-valued, 34
- continuous-time, 34
- deterministic, 34
- digital, 34
- discrete-time, 34
- duration, 36
- energy, 35
- narrowband, 44
- one-dimensional, 34
- periodic, 36
- power, 35
- random, 36
- real-valued, 34
- signal-to-interference-plus-noise ratio (SINR), 643
- signal-to-noise ratio
  - array, 633
  - element, 633
- similarity transformation, 270
- singular value decomposition (SVD), 431, 491
- singular values, 432
- SINR maximization, 643
- sinusoidal model, 478–482
- skewness, 79
- Slepian tapers, 247
- space time-bandwidth product, 656
- space-time adaptive processing (STAP), 683–685
- space-time filtering, 683
- spatial ambiguities, 630, 635
- spatial filter, 631
- spatial filtering, 25
- spatial frequency, 630
- spatial matched filter, 634
- spatial power spectrum, 632
- spatial sampling frequency, 630
- spatial sampling period, 630
- spectral dynamic range, 124
- spectral estimation, 8
- spectral factorization, 61, 152
- spectral flatness measure, 153
- spectral norm, 757
- spectral synthesis method, 736
- spectral theorem, 122
- spectrum estimation
  - Capon's method, 472
  - data-adaptive, 472
  - deterministic signals, 196
  - maximum entropy method, 460–461
  - minimum variance, 471–478
  - parametric, 467–470
  - pole-zero models, 467–470
  - relationship between minimum-variance and all-pole methods, 477–478
- spectrum sampling, 199
- spectrum splitting, 457
- speech modeling, 470–471
- speech signals, 4
- speed of adaptation, 515
- spherically invariant random processes (SIRP), 528
- square matrix
  - cofactors, 760
  - determinant, 760
- stability, 518
  - bounded-input bounded-output (BIBO), 48
  - test, 69

- stable distribution(s), 10, 93  
 standard deviation, 79  
 standardized cumulative periodogram, 448  
 statistical signal processing, 1  
 statistically self-affine, 726  
 statistically self-similar, 15, 725  
 steepest descent algorithm (SDA), 517  
 steepest descent methods for beamforming, 670  
 steered response, 632  
 steering vector, 634  
 step-size parameter, 517  
 stochastic convergence, 513  
 stochastic process, 98  
 stochastic processes, self-similar, 725  
 stochastic signals, 3  
 strict white noise, 110  
 Student's  $t$  distribution, 138  
 subarrays, 675  
 subband adaptive filtering, 548  
 subspace techniques, 478–493  
 sum beamformer, 680  
 sum of squared errors (SSE) criterion, 264  
 superladder, 373  
 superlattice, 370  
 superresolution, 478, 682  
 superrsolution, 28  
 supervised adaptation, 507  
 symbol equalizer (SE), 315  
 symbol interval, 20, 311  
 symmetric  $\alpha$ -stable, 94  
 symmetric linear smoother, 288  
 synchronous equalizer, 315  
 synthesis filter, 152  
 system function, 49  
 system identification, 11, 17  
 system inversion, 19  
 system modeling, 11  
 system-based signal model, 151
- tapered conventional beamforming, 638  
 tapering, 197  
 tapering loss, 639, 650  
 target signal, 7  
 thinned arrays, 636  
 time average, 106  
 time dispersion, 502  
 time series, 1, 98  
 time-bandwidth product, 628  
 time-delay steering, 657  
 Toeplitz matrix, 48, 123
  - inversion, 377
  - triangularization, 374
 tracking mode, 507  
 training sequence, 21
- training set, 396  
 transform-domain LMS, 547  
 trispectrum, 693
- uniform linear array (ULA), 25, 624  
 unit gain on noise normalization, 644  
 unit impulse, 35  
 unit sample response, 35, 47  
 unit step sequence, 35  
 unitary complex space, 755  
 unitary matrix, 762  
 unitary transformation, 762  
 unsupervised adaptation, 507  
 unsupervised adaptive filters, 703
- Vandermode matrix, 121  
 variance, 79, 100  
 vectors
  - angle between, 756
  - linearly independent, 756
  - orthonormalized, 756
 vocal tract, 13
- wavelength, 624  
 well (ill) conditioned matrix, 762  
 white noise, 110  
 whitening, 304  
 whitening filter, 152  
 whitening transformation, 126  
 wideband interference, 656  
 wideband steering vector, 656  
 Wiener filters, 278  
 Wiener-Hopf equations, 282  
 windowing, 197, 198, 408  
 windows
  - Dolph-Chebyshev, 208
  - Hamming, 206
  - Kaiser, 207
  - rectangular, 206
 Wishart distribution, 555  
 Wold decomposition, 156  
 Woodbury's formula, 746
- Yule-Walker equations, 160, 164
- zero padding, 199, 201  
 zero-forcing equalizer, 316  
 zeros, 44  
 $z$ -transform, 43