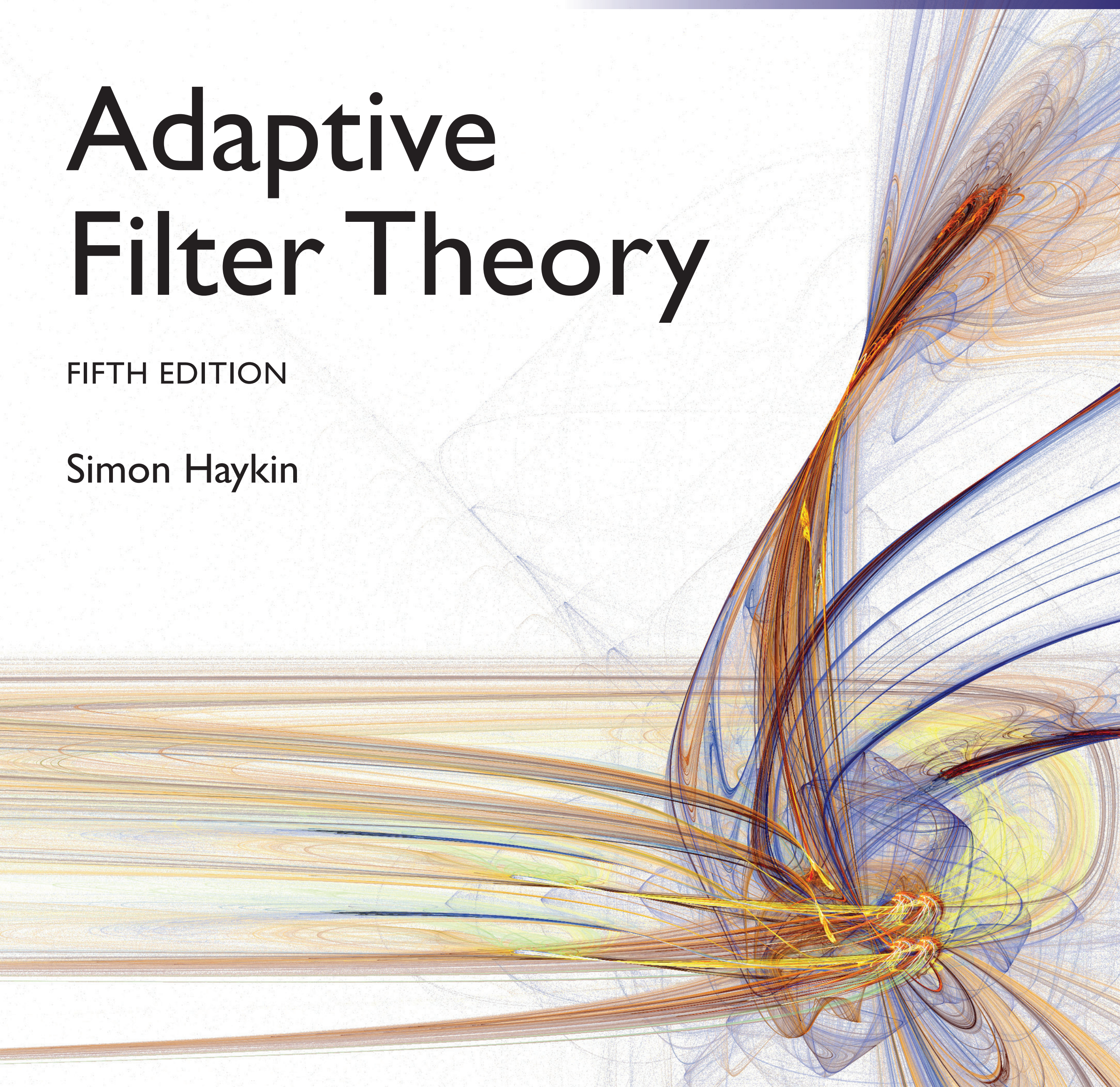
INTERNATIONAL EDITION

Adaptive Filter Theory

FIFTH EDITION

Simon Haykin



ALWAYS LEARNING

PEARSON

ADAPTIVE FILTER THEORY

Fifth Edition

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

International Edition contributions by

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

PEARSON

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

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

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

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

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

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

PEARSON

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

This book is dedicated to

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

Contents

Preface

Comparisons of the New to the Previous Edition of the Book

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

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

Introductory Remarks on the New Edition

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

Aims of the Book

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

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

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

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

Organization of the Book

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

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

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

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

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

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

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

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

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

The book also includes appendices on the following topics:

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

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

Ancillary Material

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

Examples, Computer Experiments, and Problems

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

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

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

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

Solutions Manual

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

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

Two Noteworthy Symbols

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

Use of the Book

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

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

Simon Haykin
Ancaster, Ontario,
Canada

Acknowledgments

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

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

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

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

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

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

Simon Haykin
Ancaster, Ontario,
Canada

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

Background and Preview

1. THE FILTERING PROBLEM

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

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

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

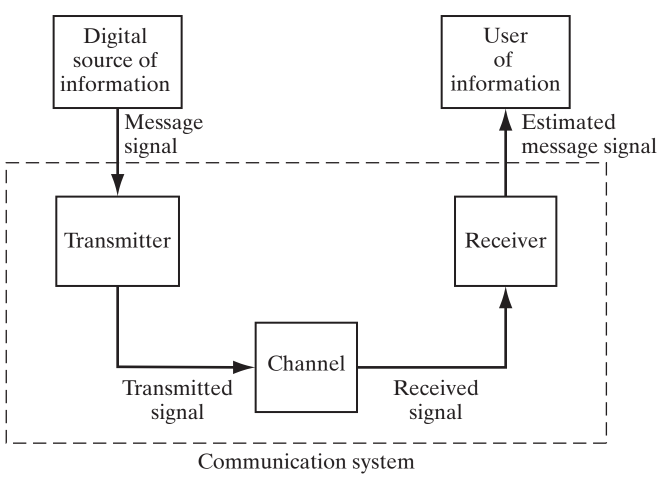


FIGURE 1 Block diagram of a communication system.

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

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

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

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

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

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

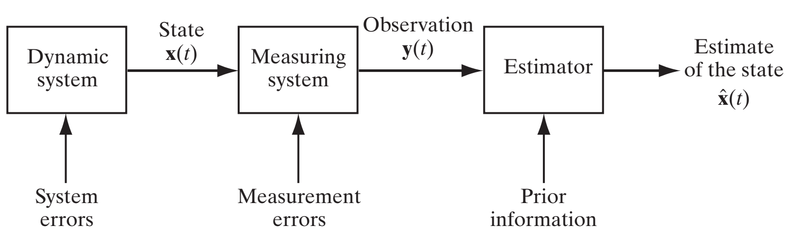


FIGURE 2 Block diagram depicting the components involved in state estimation, namely $\hat{\mathbf{x}}(t)$.

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

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

Three Basic Kinds of Estimation

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

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

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

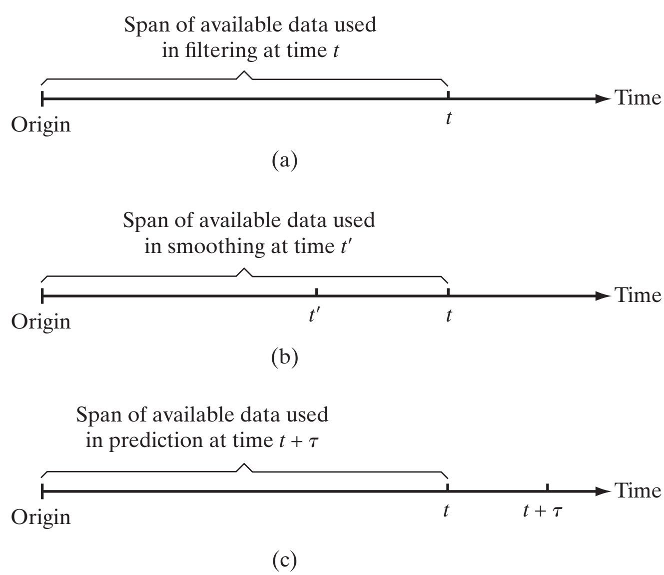


FIGURE 3 Illustrating the three basic forms of estimation: (a) filtering; (b) smoothing; (c) prediction.

2. LINEAR OPTIMUM FILTERS

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

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

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

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

3. ADAPTIVE FILTERS

The design of a Wiener filter requires a priori information about the statistics of the data to be processed. The filter is optimum only when the statistical characteristics of the input data match the a priori information on which the design of the filter is based. When this information is not known completely, however, it may not be possible to design the Wiener filter or else the design may no longer be optimum. A straightforward approach that we may use in such situations is the "estimate and plug" procedure. This is a two-stage process whereby the filter first "estimates" the statistical parameters of the relevant signals and then "plugs" the results so obtained into a *nonrecursive* formula for computing the filter parameters. For real-time operation, this procedure has the disadvantage of requiring excessively elaborate and costly hardware. To mitigate this limitation, we may use an *adaptive filter*. By such a system we mean one that is *self-designing* in that the adaptive filter relies for its operation on a *recursive algorithm*, which makes it possible for the filter to perform satisfactorily in an environment where complete knowledge of the relevant signal characteristics is not available. The algorithm starts from some predetermined set of *initial conditions*, representing whatever we know about the environment. Yet, in a stationary environment, we find that after successive adaptation cycles of the algorithm it *converges* to the optimum Wiener solution in some statistical sense. In a nonstationary environment, the algorithm offers a *tracking* capability, in that it can track time variations in the statistics of the input data, provided that the variations are sufficiently slow.

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

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

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

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

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

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

4. LINEAR FILTER STRUCTURES

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

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

Linear Filters with Finite Memory

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

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

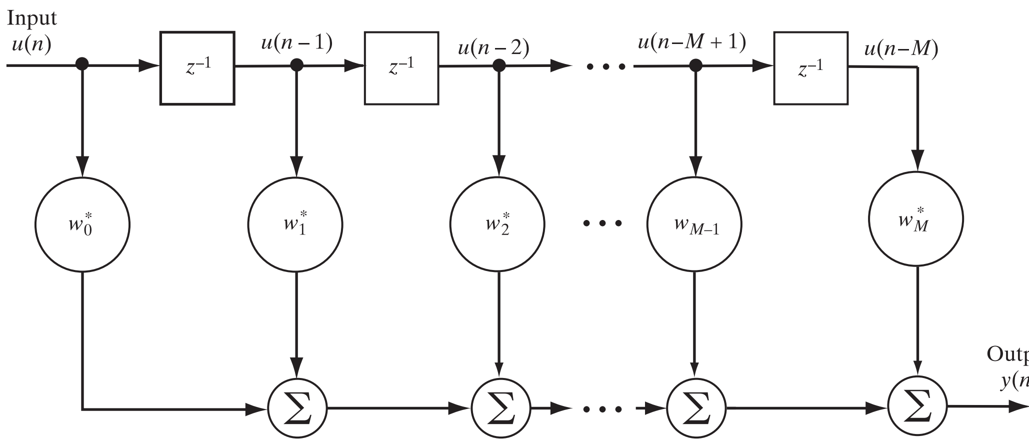


FIGURE 4 FIR filter.

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

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

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

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

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

and

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

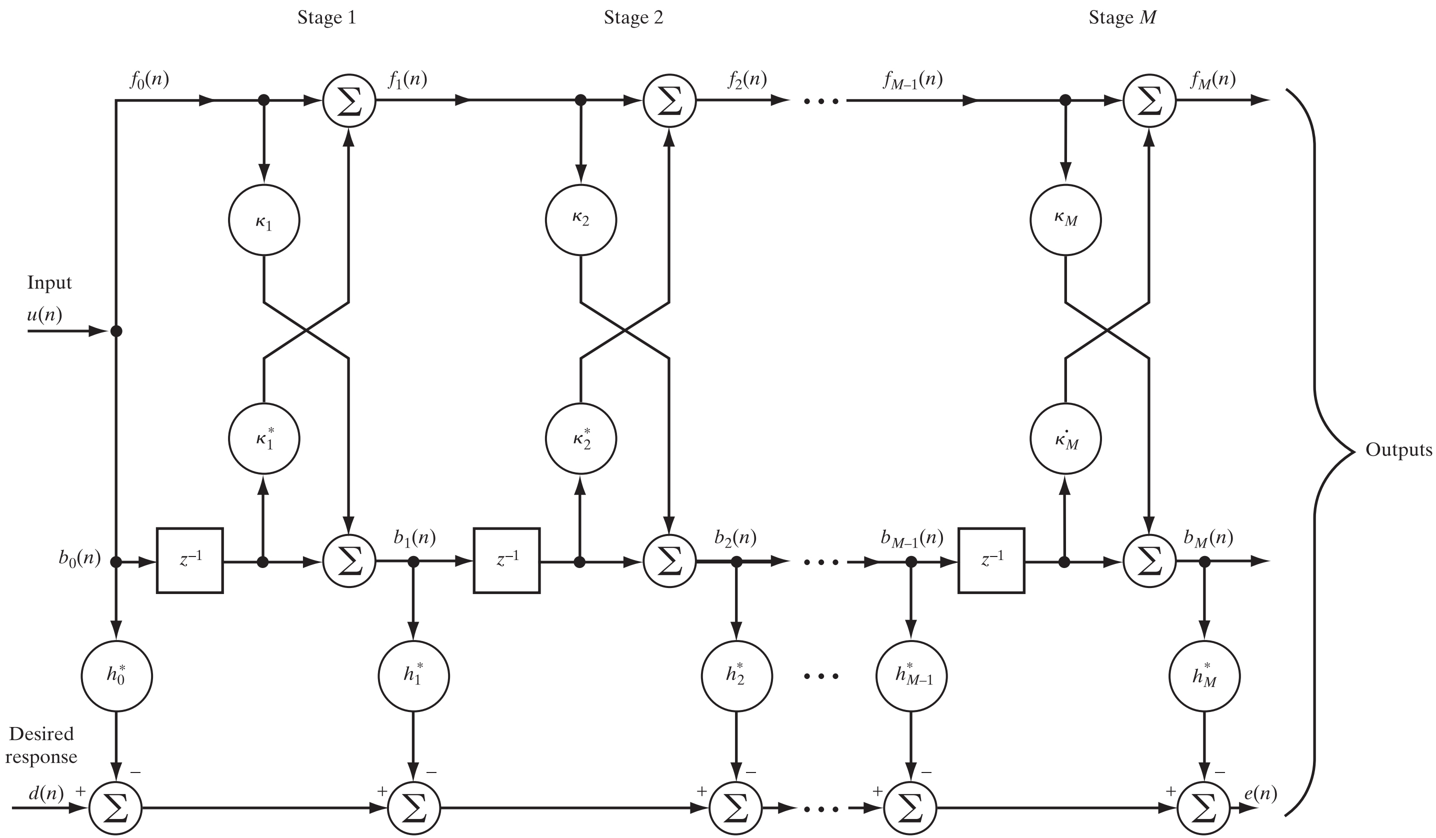


FIGURE 5 Multistage lattice filter.

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

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

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

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

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

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

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

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

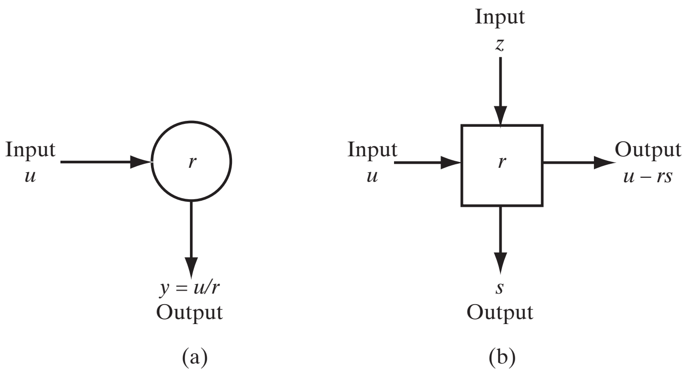


FIGURE 6 Two basic cells of a systolic array: (a) boundary cell; (b) internal cell.

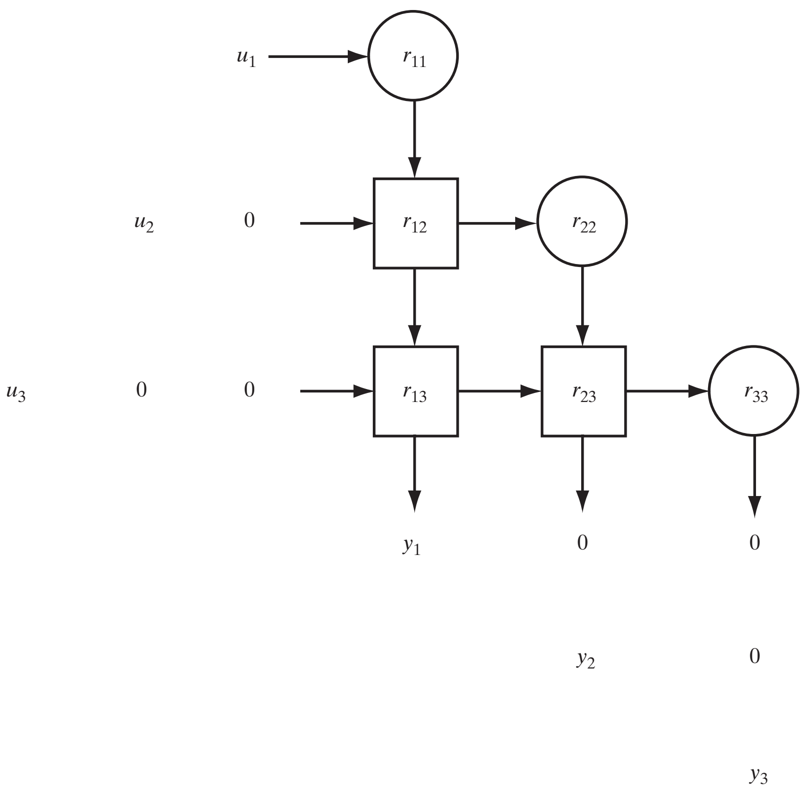


FIGURE 7 Triangular systolic array as example.

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

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

Linear Filters with Infinite Memory

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

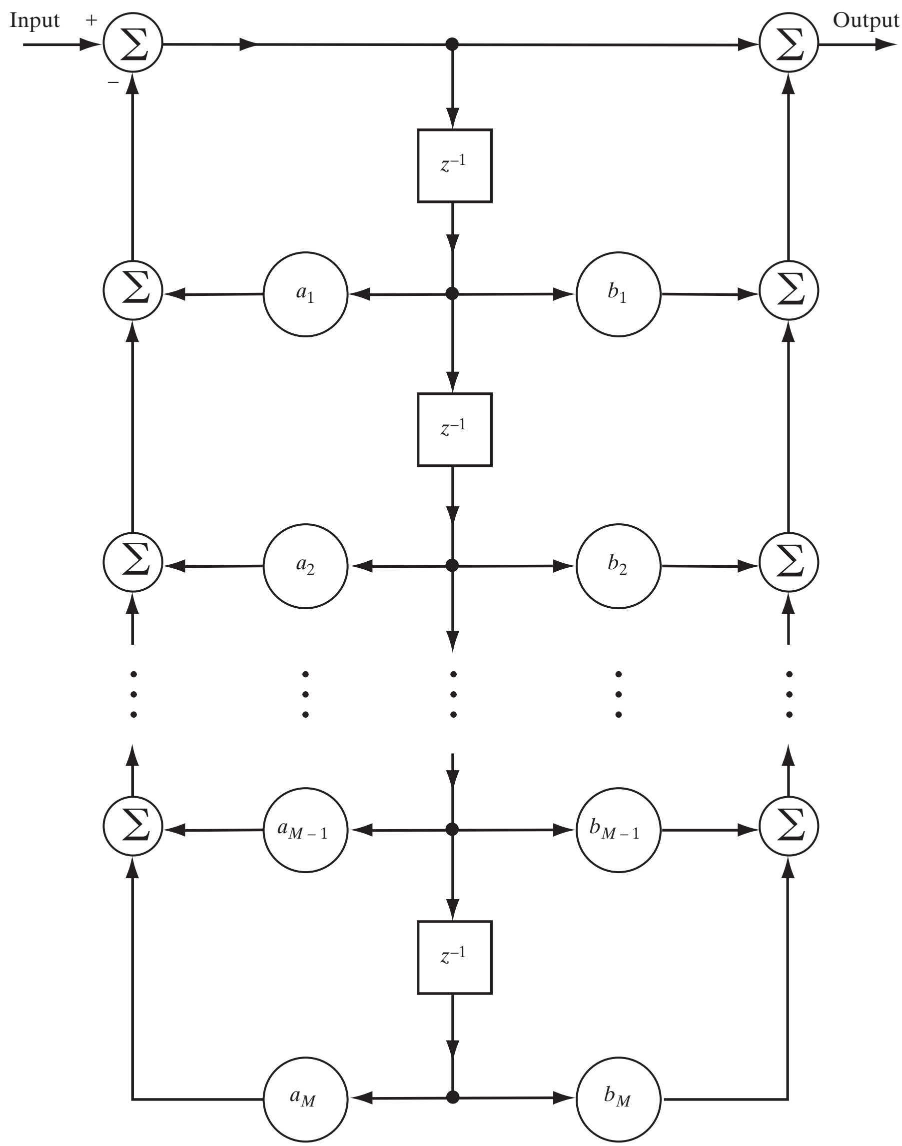


FIGURE 8 IIR filter, assuming real-valued data.

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

5. APPROACHES TO THE DEVELOPMENT OF LINEAR ADAPTIVE FILTERS

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

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

Method of Stochastic Gradient Descent

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

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

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