ADAPTIVE SIGNAL PROCESSING

Bernard Widrow
Samuel D. Stearns

PRENTICE-HALL SIGNAL PROCESSING SERIES
ALAN V. OPPENHEIM, SERIES EDITOR

Sony v. Jawbone
U.S. Patent No. 10,779,080
Sony Ex. 1020

ADAPTIVE SIGNAL PROCESSING

Bernard Widrow
Stanford University

Samuel D. Stearns
Sandia National Laboratories

Prentice-Hall, Inc.
Englewood Cliffs, N.J. 07632

PRENTICE-HALL SIGNAL PROCESSING SERIES

Alan V. Oppenheim, Editor

ANDREWS AND HUNT *Digital Image Restoration*
BRIGHAM *The Fast Fourier Transform*
BURDIC *Underwater Acoustic System Analysis*
CASTLEMAN *Digital Image Processing*
COWAN AND GRANT *Adaptive Filters*
CROCHIERE AND RABINER *Multirate Digital Signal Processing*
DUDGEON AND MERSEREAU *Multi-Dimensional Digital Signal Processing*
HAMMING *Digital Filters, 2e*
HAYKIN, ET AL. *Array Signal Processing*
JAYANT AND NOLL *Digital Coding of Waveforms*
LEA, ED. *Trends in Speech Recognition*
LIM *Speech Enhancement*
MCCLELLAN AND RADER *Number Theory in Digital Signal Processing*
OPPENHEIM, ED. *Applications of Digital Signal Processing*
OPPENHEIM AND SCHAFER *Digital Signal Processing*
OPPENHEIM, WILLSKY, WITH YOUNG *Signals and Systems*
RABINER AND GOLD *Theory and Application of Digital Signal Processing*
RABINER AND SCHAFER *Digital Processing of Speech Signals*
ROBINSON AND TREITEL *Geophysical Signal Analysis*
TRIBOLET *Seismic Applications of Homomorphic Signal Processing*
WIDROW AND STEARNS *Adaptive Signal Processing*

Library of Congress Cataloging in Publication Data

Widrow, Bernard, (date)
Adaptive signal processing.

(Prentice-Hall signal processing series)
Includes index.
1. Adaptive signal processing. I. Stearns, Samuel D. II. Title. III. Series.
TK5102.5.W537 1985 621.38'043 84-18057
ISBN 0-13-004029-0

Editorial/production supervision and interior design: *Tracey L. Orbine*
Cover design: *Sue Behnke*
Manufacturing buyer: *Anthony Caruso*

ENGINEERING SOCIETIES LIBRARY

JUN 24 1985

REVIEW

© 1985 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

621.38
Z85e

ISBN 0-13-004029 01

Prentice-Hall International, Inc., *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S. A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Whitehall Books Limited, *Wellington*, *New Zealand*

621.3807 -

This book is dedicated to the memory of Moses Widrow, William K. Linvill, and Thomas J. Flanagan. It is also dedicated to the cause of peace on earth. We hope and trust that its contents will be used to improve the lot of mankind everywhere.

R. 5-23-85 $41.95

LINDA HALL LIBRARY
Kansas City, Mo.

This material may be protected by Copyright law (Title 17 U.S. Code)

part I

GENERAL INTRODUCTION

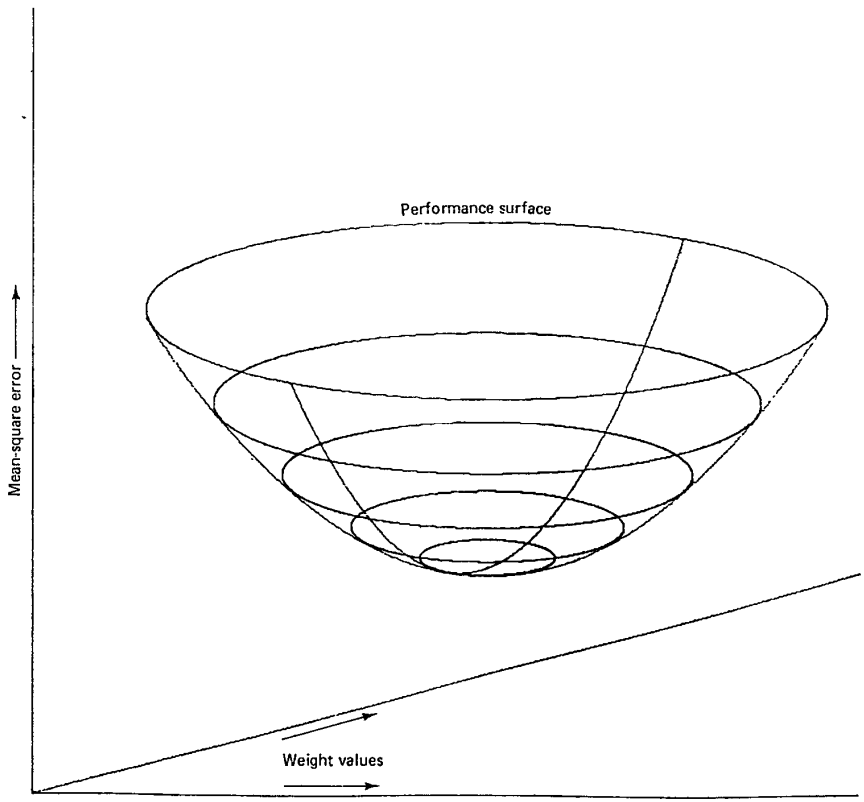
(Chapters 1 and 2)

OBJECTIVES OF PART I

In the first two chapters of this book we have three major objectives. The first is to introduce the basic meaning of "adaptation" (or "adaption") in the engineering sense, and to set adaptive signal processing into the general signal processing context.

The second objective is to describe the adaptive linear combiner, which is the simplest and most widely applicable adaptive processor. It is the basic adaptive device that will be used exclusively through Chapter 6, as well as in much of the rest of the text.

The third objective is to persuade the reader to think of the overall process of adaptation in geometrical terms. We wish to think of adaptation as a procedure for moving generally downhill on a "performance surface" like the one shown on page 2, which is the L-dimensional surface in $(L+1)$-dimensional space formed by plotting the mean-square error versus the two adaptive parameters. These geometrical concepts and terms are described in Chapter 2.


Performance surface
Mean-square error
Weight values

chapter 1

Adaptive Systems

DEFINITION AND CHARACTERISTICS

> adapt, v.t., 1. to make suitable to requirements or conditions; adjust or modify fittingly; ... v.i., 2. to adjust oneself to different conditions, environments, etc. (*Random House Dictionary*, 1971)

In recent years, a growing field of research in "adaptive systems" has resulted in a variety of adaptive automatons whose characteristics in limited ways resemble certain characteristics of living systems and biological adaptive processes. According to the *Random House Dictionary*, some of the meanings of "adaptation" are:

> 1. the act of adapting. 2. the state of being adapted; adjustment. 3. *Biol.* a. any alteration in the structure or function of an organism or any of its parts that results from natural selection and by which the organism becomes better fitted to survive and multiply in its environment. b. a form or structure modified to fit changed environment. 5. *Physiol.* the decrease in response of sensory receptor organs, as those of vision, touch, temperature, olfaction, audition, and pain, to changed, constantly applied, environmental conditions. 6. *Ophthalm.* the regulating by the pupil of the quantity of light entering the eye. 7. *Sociol.* a slow, usually unconscious modification of individual and social activity in adjustment to cultural surroundings.

It will be noted that the definition above is expressed primarily in terms of biological adaptation to environment. The same definitions serve at least to some extent for "artificial" or human-made adaptive systems, which are the central concern of this book.

An adaptive automaton is a system whose structure is alterable or adjustable in such a way that its behavior or performance (according to some desired criterion) improves through contact with its environment. A simple example of an automaton or automatic adaptive system is the automatic gain control (AGC) used in radio and television receivers. The function of this circuit is to adjust the sensitivity of the

receiver inversely as the average incoming signal strength. The receiver is thus able to adapt to a wide range of input levels and to produce a much narrower range of output intensities.

The purpose of this book is to present certain basic principles of adaptation; to explain the design, operating characteristics, and applications of the simpler forms of adaptive systems; and to describe means for their physical realization. The types of systems discussed include those designed primarily for the purposes of adaptive control and adaptive signal processing. Such systems usually have some or all of the following characteristics:

1. They can automatically adapt (self-optimize) in the face of changing (nonstationary) environments and changing system requirements.
2. They can be trained to perform specific filtering and decision-making tasks. Synthesis of systems having these capabilities can be accomplished automatically through training. In a sense, adaptive systems can be "programmed" by a training process.
3. Because of the above, adaptive systems do not require the elaborate synthesis procedures usually needed for nonadaptive systems. Instead, they tend to be "self-designing."
4. They can extrapolate a model of behavior to deal with new situations after having been trained on a finite and often small number of training signals or patterns.
5. To a limited extent, they can repair themselves; that is, they can adapt around certain kinds of internal defects.
6. They can usually be described as nonlinear systems with time-varying parameters.
7. Usually, they are more complex and difficult to analyze than nonadaptive systems, but they offer the possibility of substantially increased system performance when input signal characteristics are unknown or time varying.

AREAS OF APPLICATION

Recent progress in microcircuit design and production has resulted in very compact, economical, and reliable signal processors that rival biological nervous systems in size and are clearly superior to biological systems in speed. The result has been a very fast-growing field of applications for all types of digital signal processing, including adaptive processing. Current applications for adaptive systems are in such fields as communications, radar, sonar, seismology, mechanical design, navigation systems, and biomedical electronics.

Part IV of this book concerns certain classes of applications, and the chapter headings provide a rough picture of the application areas. Chapter 9 covers adaptive modeling and system identification, in which an adaptive system models an un-

known system or "plant," which may be slowly varying with time. For a given input signal, the adaptive system tries to match the plant output. Adaptive modeling is used in electrical and mechanical design and testing.

Chapter 10 is on inverse modeling, deconvolution, and equalization. All of these terms, as well as the term "inverse filtering," refer to the process of removing the effect of some device or medium on a signal. For example, we may wish to make an audio system respond with equal gain to all speech frequencies, or to remove the effects of a transmission line on a radar pulse.

Chapter 11 is on adaptive control systems, that is, control systems whose characteristics change with time and adapt to the environment. An example is the flight control system whose gain and response times change with air density. Adaptive noise canceling, discussed in Chapter 12, has been applied to areas such as speech communications, electrocardiography, and seismic signal processing. Adaptive noise canceling is in fact applicable to a wide variety of signal-enhancement situations, because noise characteristics are not often stationary in real-world situations. Chapters 13 and 14, on adaptive arrays, describe an area where adaptive signal processing concepts have proved to be especially useful.

GENERAL PROPERTIES

The essential and principal property of the adaptive system is its time-varying, self-adjusting performance. The need for such performance may readily be seen by realizing that if a designer develops a system of *fixed design* which he or she considers optimal, the implications are that the designer has foreseen all possible input conditions, at least statistically, and knows what he or she would like the system to do under each of these conditions. The designer has then chosen a specific criterion whereby performance is to be judged, such as the amount of error between the output of the actual system and that of some selected model or "ideal" system. Finally, the designer has chosen the system that appears best according to the performance criterion selected, generally choosing this system from an a priori restricted class of designs (such as linear systems).

In many instances, however, the complete range of input conditions may not be known exactly, or even statistically; or the conditions may change from time to time. In such circumstances, an adaptive system that continually seeks the optimum within an allowed class of possibilities, using an orderly search process, would give superior performance compared with a system of fixed design.

By their very nature, adaptive systems must be time varying and nonlinear. Their characteristics depend, among other things, on their input signals. If an input signal x_1 is applied, an adaptive system will adapt to it and produce an output—let us call it y_1. If another input signal, x_2, is applied, the system will adapt to this second signal and will again produce an output—let us call it, this time, y_2. Generally, the form or the structure or the adjustments of the adaptive system will be different for the two different inputs. If the sum of the two inputs is applied to

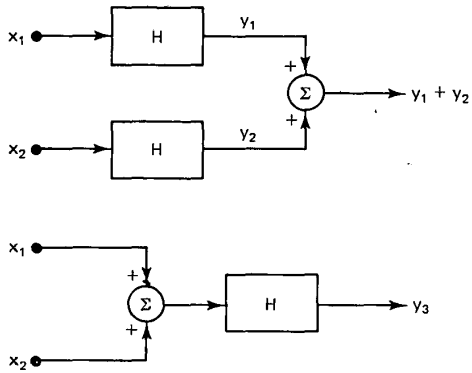


Figure 1.1 The lower output, y_3, is equal to $y_1 + y_2$ if H is a linear system. If H is adaptive, y_3 is generally different from $y_1 + y_2$.

the adaptive system, the latter will adapt to this new input—but it will produce an output that will generally not be the same as $y_1 + y_2$, the sum of the outputs that would have corresponded to inputs x_1 and x_2. In such a case, as illustrated in Figure 1.1, the principle of superposition does not work as it does with linear systems. If a signal is applied to the input of an adaptive system to test its response characteristics, the system adapts to this specific input and thereby changes its own form. Thus the adaptive system is inherently difficult to characterize in conventional terms.

Within the realm of nonlinear systems, adaptive systems cannot be distinguished as belonging to an absolutely clear subset. However, they have two features that generally distinguish them from other forms of nonlinear systems. First, adaptive systems are adjustable, and their adjustments usually depend on finite-time average signal characteristics rather than on instantaneous values of signals or instantaneous values of the internal system states. Second, the adjustments of adaptive systems are changed purposefully in order to optimize specified performance measures.

Certain forms of adaptive systems become linear systems when their adjustments are held constant after adaptation. These may be called "linear adaptive systems." They are very useful; they tend to be mathematically tractable; and they are generally easier to design than other forms of adaptive systems.

OPEN- AND CLOSED-LOOP ADAPTATION

Several ways to classify adaptive schemes have been proposed in the literature. It is most convenient here to begin by thinking in terms of *open-loop* and *closed-loop* adaptation. The open-loop adaptive process involves making measurements of input or environmental characteristics, applying this information to a formula or to a computational algorithm, and using the results to set the adjustments of the adaptive system. Closed-loop adaptation, on the other hand, involves automatic experimen-

tation with these adjustments and knowledge of their outcome in order to optimize a measured system performance. The latter process may be called adaptation by "performance feedback."

The principles of open- and closed-loop adaptation are illustrated in Figures 1.2 and 1.3. In both cases it is helpful to envisage the adaptive process as it might be performed manually by a human operator or "supervisor." In Figures 1.2(a) and 1.3(a) the supervisor is shown adjusting the controls of the processor by reading a display that registers measurements of the preselected performance criterion. In the open-loop system this criterion is a characteristic of the input signal and perhaps other data, and in the closed-loop system it is also a function of the output signal. The adjustments in Figure 1.3 are made even though the operator may have no knowledge of what is inside the processor or of the functions performed by the controls. The operator does not process the input signal; he or she only controls the adjustments of the processor to keep its performance optimized according to the preselected criterion. Thus the operator's function is purely supervisory, and in real automatic adaptive systems the operator is replaced by computational or "adaptive" algorithms, as suggested by Figures 1.2(b) and 1.3(b). The "other data" in these figures may be data about the environment of the adaptive system, or in the closed-loop case, it may be a desired version of the output signal.

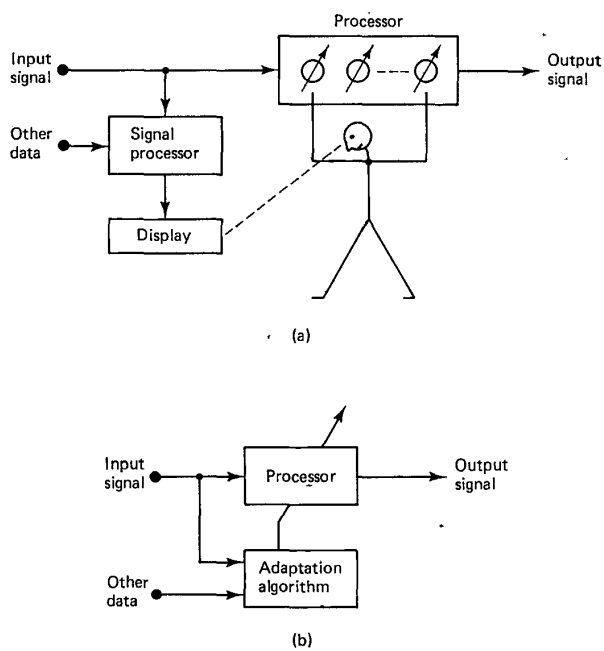
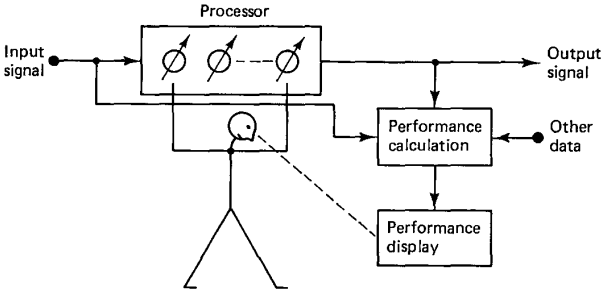
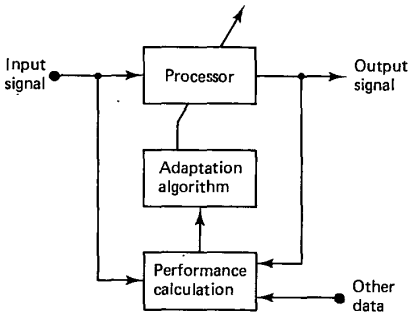


Figure 1.2 Open-loop adaptation: (a) concept; (b) equivalent system.



(a)



(b)

Figure 1.3 Closed-loop adaptation.

When designing an adaptive process, many factors determine the choice of closed-loop versus open-loop adaptation. The availability of input signals and performance-indicating signals is a major consideration. Also, the amount of computing capacity and the type of computer required to implement the open-loop and closed-loop adaptation algorithms will generally differ. Certain algorithms require the use of a general-purpose digital computer, whereas other algorithms could be implemented far more economically with special-purpose chips or other apparatus. Some of these structural considerations are discussed in later chapters. It is difficult to develop general principles to guide all choices, but several advantages and a few disadvantages of closed-loop adaptation, which is the main subject of this book, can be pointed out here.

Closed-loop adaptation has the advantage of being workable in many applications where no analytic synthesis procedure either exists or is known; for example, where error criteria other than mean-square are used, where systems are nonlinear or time variable, where signals are nonstationary, and so on. Closed-loop adaptation

can also be used effectively in situations where physical system component values are variable or inaccurately known. Closed-loop adaptation will find the best choice of component values. In the event of partial system failure, an adaptation mechanism that continually monitors performance will optimize this performance by adjusting and reoptimizing the intact parts. As a result, system reliability can often be improved by the use of performance feedback.

The closed-loop adaptation process is not always free of difficulties, however. In certain situations, performance functions do not have unique optima. Automatic optimization is an uncertain process in such situations. In other situations, the closed-loop adaptation process, like a closed-loop control system, could be unstable. The adaptation process could diverge rather than converge. In spite of these possibilities, performance feedback is a powerful, widely applicable technique for implementing adaptation. Most of the adaptive processes described in this book will be closed-loop processes utilizing performance feedback.

APPLICATIONS OF CLOSED-LOOP ADAPTATION

Let us now consider briefly some applications of the closed-loop, performance feedback concept. These will anticipate the applications chapters in Part IV, beginning with Chapter 9.

We begin by representing the performance feedback process [Figure 1.3(b)] more specifically in Figure 1.4. We call the input signal x and define a "desired response" signal d, which is assumed to represent the desired output of the adaptive system. The signal d is, for our purpose here, the "other data" in Figure 1.3(b).

The error signal, ε, is the difference between the desired output signal and the actual output signal, y, of the adaptive system. Using the error signal, an adaptive algorithm adjusts the structure of the adaptive system, thus altering its response characteristics by minimizing some measure of the error, thereby closing the performance loop.

Different structures are possible for the adaptive system, and these are discussed later, beginning in Chapter 2. Adaptive algorithms for adjusting these structures are also discussed later, beginning in Chapter 4. Here we wish only to show in a general way how the scheme in Figure 1.4 is applied in practical situations.

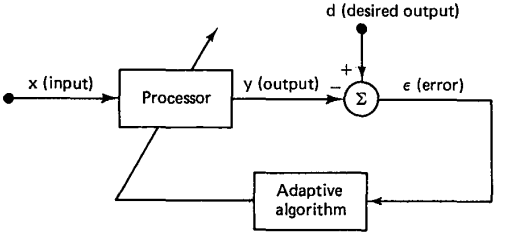


Figure 1.4 Signals in closed-loop adaptation.

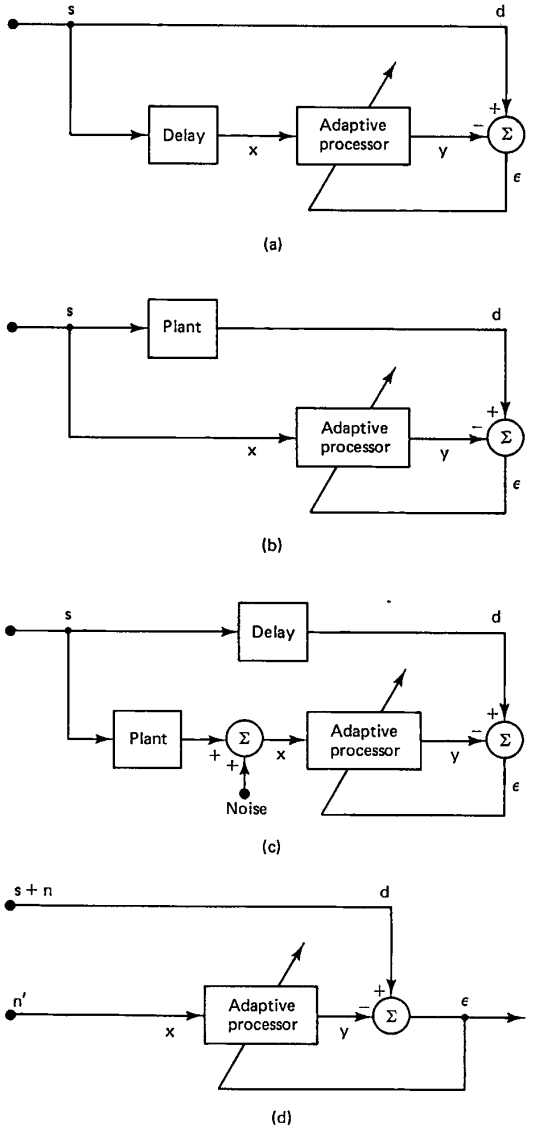


Figure 1.5 Examples showing how the configuration in Figure 1.4 may be applied: (a) prediction; (b) system identification (modeling); (c) equalization (deconvolution, inverse filtering, inverse modeling); (d) interference canceling.

Some examples of applications are given in Figure 1.5. Notice that Figure 1.4, which illustrates the basic closed-loop adaptive process, is simplified slightly and embedded in each part of this figure, and that the application determines how the desired signal, d, is obtained.

The prediction application in Figure 1.5(a) is perhaps the simplest of the four. The desired signal is the input signal, s, and a delayed version of the latter is sent to the adaptive processor, which must therefore try to "predict" the current input signal in order to have y cancel d and drive ε toward zero. Prediction is used in signal encoding and noise reduction, and is discussed in Chapters 8, 9, and 12.

The system identification application in Figure 1.5(b) is also easy to understand. Here a broadband signal, s, is the input to the adaptive processor as well as to an unknown "plant" (a term originating in the control literature). To reduce ε, the adaptive processor tries to emulate the plant's transfer characteristic. After adaptation the plant is "identified" in the sense that its transfer function can be specified as essentially the same as that of the adaptive processor. Adaptive system identification or modeling can be used as such, to model a slowly varying plant whose input and output signals are available, for example, in vibration studies of mechanical systems. It can also be used in many other ways, some of which are described in Chapters 9 and 11.

The inverse modeling application [Figure 1.5(c)] is discussed primarily in Chapter 10, and its uses in control are discussed in Chapter 11. In this application the adaptive processor attempts to recover a delayed version of the signal, s, which is assumed to have been altered by the slowly varying plant and to contain additive noise. The delay in the figure is to allow for the delay, or propagation time, through the plant and the adaptive processor. Adaptive equalization could be used to undo (deconvolve) the effects of a transducer, a communication channel, or some other system, or to produce an inverse model of an unknown plant. It is also applicable in the design of digital filters, as well as in adaptive control problems, and so on.

Finally, Figure 1.5(d) shows the adaptive processor in an interference-canceling configuration. Here the signal, s, is corrupted by additive noise, n, and a distorted but correlated version of the noise, n', is also available. The goal of the adaptive processor in this case is to produce an output, y, that closely resembles n, so that the overall output, ε, will closely resemble s. We will show, under certain very broad conditions, that the optimal adaptive processor is that which minimizes the mean-square value of ε. The subject of adaptive interference canceling is discussed in Chapters 12 through 14.

EXAMPLE OF AN ADAPTIVE SYSTEM

By now we have seen that adaptive signal processing is really a very general and basic term which implies the use of time-varying, self-adjusting signal processors. We think of the performance of these systems as being purposeful, useful, and

sometimes even "intelligent" in a limited sense. Before concluding this introductory chapter we introduce just one specific example of such a system, to provide a more concrete illustration of adaptation in general and closed-loop adaptation in particular. We will use the "prediction" application in Figure 1.5(a), keeping in mind that in doing so we have a very restricted and specific example of a very broad and general class of systems.

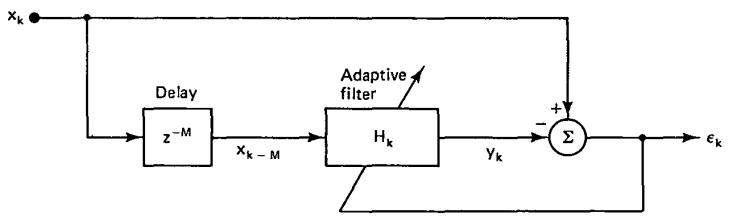


Figure 1.6 Example of an adaptive system: an adaptive predictor.

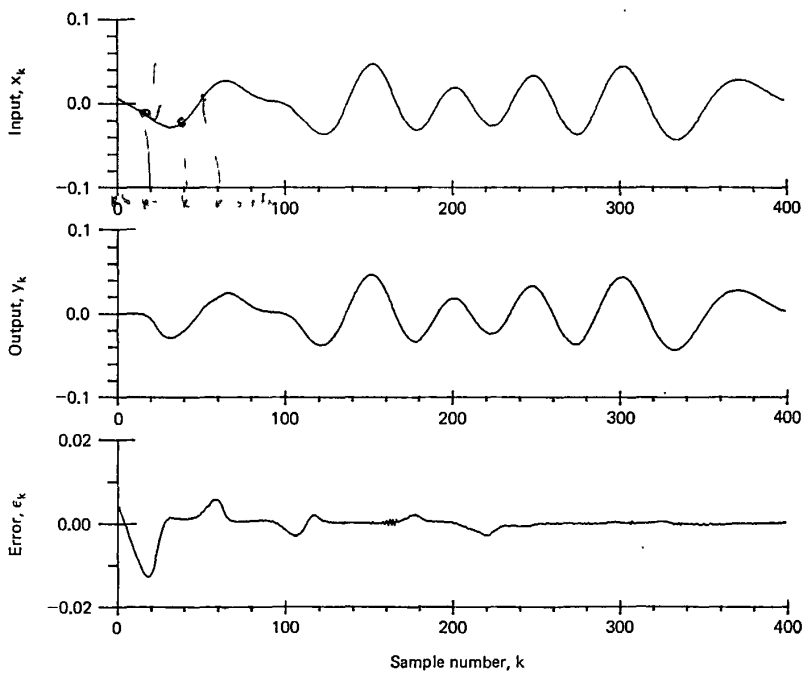


Figure 1.7 Signals in the adaptive predictor in Figure 1.6. The error (ε) becomes nearly zero as the system learns to predict the input (x) and the output (y) approaches x.

The adaptive predictor diagram in Figure 1.5(a) is redrawn in Figure 1.6, with examples of several conventions of notation that are used throughout this text. The symbols x_k, y_k, and ε_k represent the kth elements in the time series represented by x, y, and ε. Usually, the time series may be assumed to be obtained by sampling continuous signals. Thus $x_k = x(kT)$, and so on, where T is the time step or interval between samples. The symbol z^{-M} stands for a fixed delay of M time steps, so that in Figure 1.6 the output of the z^{-M} block is labeled x_{k-M}. The symbol H_k represents the transfer characteristic of the adaptive processor or "adaptive filter." We leave a detailed description of H_k to subsequent chapters, except to note here that the subscript k indicates that the transfer characteristic possibly changes at each sample point.

Thus in Figure 1.6 we see that the input signal, x_k, is delayed and then filtered to produce y_k, and that y_k is subtracted from x_k to obtain the error, ε_k. The transfer function, H_k, is adjusted to keep the average squared value of this error as small as possible. In this way, the processor always uses past values of x to predict the present value of x while using ε to adjust H, and is thus involved in a performance-feedback process.

An example of adaptive predictor performance is shown in Figure 1.7. The waveforms are constructed from digital data (here as well as in examples ahead) by drawing straight lines between the sample points. The input, x, is a frequency-limited random signal, and $M = 1$ in this case. Note that y_k becomes a better prediction of x_k, and that the magnitude of ε_k decreases, as k increases and the adaptive system learns progressively to predict x_k. More detailed examples of adaptive prediction are covered in subsequent chapters.

THE CHAPTERS AHEAD

In Chapters 2 through 8 we present some basic discussion and mathematical description of the adaptive process. The performance of adaptive systems can be analyzed either in the time domain or in the frequency domain; in general, the frequency-domain analysis tends to be more elegant and difficult. Therefore, in Chapters 2 through 6 we stay in the time domain and avoid using transforms, transfer functions, and so on, as much as possible. In Chapter 2 we introduce the adaptive linear combiner, which is the basic nonrecursive form of the adaptive filter. Then in Chapters 3 through 5 we introduce the geometry of the "performance surface" and consider various methods of adaptively seeking the minimum point on this surface.

In Chapter 6 we introduce the well-known least-mean-square (LMS) algorithm, which is the simplest, most important, and most widely used algorithm for adjusting the weights in a linear adaptive system. After reviewing Chapter 2, the more advanced reader or the reader wishing simply to apply the LMS algorithm without covering all of the theory may wish to skip to Chapter 6 and beyond, without covering all of the material in Chapters 3 through 5.

Some areas of adaptive signal processing, including recursive adaptive filters, cannot be analyzed conveniently without reference to the frequency domain, so Chapter 7 is concerned mainly with developing foundations for this type of analysis. Then in Chapter 8 we introduce other types of algorithms that require frequency-domain analysis. In Chapter 8 we also introduce the adaptive lattice, a structure different from the adaptive linear combiner of Chapter 2. Finally, Chapters 9 through 14 deal with some of the important applications of adaptive signal processing, as described above.

This material may be protected by Copyright law (Title 17 U.S. Code)

chapter 2

The Adaptive Linear Combiner

GENERAL DESCRIPTION

The adaptive linear combiner, or nonrecursive adaptive filter, is fundamental to adaptive signal processing. It appears, in one form or another, in most adaptive filters and systems, and it is the single most important element in "learning" systems and adaptive processes in general. Thus a primary purpose of this book is to develop a thorough exposition of the properties, behavior, means of adaptation, rate of adaptation, and useful applications of the adaptive linear combiner.

Because of its nonrecursive structure, the adaptive linear combiner is relatively easy to understand and analyze. In essence it is a time-varying, nonrecursive digital filter, and as such its performance is quite simple. Its behavior and the means of adapting it, as well as its implementation in different forms, are well understood. Furthermore, we know of specific applications where its performance is "best" in some sense.

A diagram of the general form of the adaptive linear combiner is shown in Figure 2.1. There is an input signal vector with elements $x_0, x_1, \ldots, x_L$, a corresponding set of adjustable weights, $w_0, w_1, \ldots, w_L$, a summing unit, and a single output signal, y. A procedure for adjusting or adapting the weights is called a "weight adjustment," "gain adjustment," or "adaptation" procedure. The combiner is called "linear" because for a fixed setting of the weights its output is a linear combination of the input components. However, when the weights are in the process of being adjusted, they, too, are a function of the input components, and the output of the combiner is no longer a linear function of the input. That is, its operation becomes nonlinear in accordance with the general principle set forth in Chapter 1.

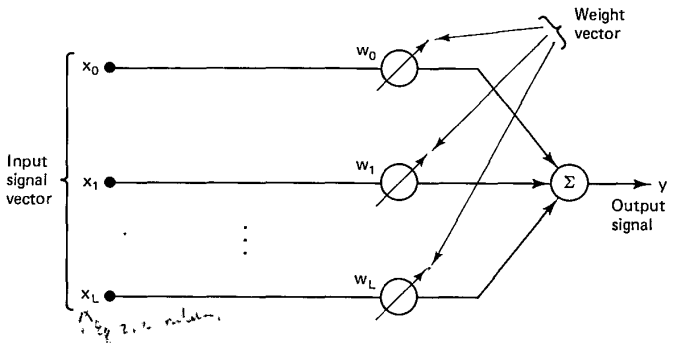


Figure 2.1 General form of adaptive linear combiner.

INPUT SIGNAL AND WEIGHT VECTORS

There are two important ways to interpret physically the elements of the input vector in Figure 2.1. First, they may be considered to be *simultaneous* inputs from $L + 1$ *different* signal sources. An example of this interpretation would be an adaptive antenna or an adaptive acoustic detection system, in which each input line is connected to a separate sensor.

Alternatively, the elements $x_0 - x_L$ may be considered to be $L + 1$ *sequential* samples for the *same* signal source. One example of this interpretation is the adaptive processor described in Chapter 1 and illustrated in Figure 1.6.

We refer to these two interpretations as the *multiple-input* and *single-input* cases, and it is convenient to label the input vector differently for the two cases, as follows:

$$\textit{Multiple inputs}: \quad \mathbf{X}_k = [x_{0k} \quad x_{1k} \quad \cdots \quad x_{Lk}]^{\mathrm{T}} \tag{2.1}$$

$$\textit{Single input}: \quad \mathbf{X}_k = [x_k \quad x_{k-1} \quad \cdots \quad x_{k-L}]^{\mathrm{T}} \tag{2.2}$$

In this notation T stands for transpose, so $\mathbf{X}_k$ is actually a column vector in both cases. The subscript k is used as a time index. Thus in the multiple-input case, all of the elements are taken at the kth sampling time, whereas in the single-input case, the elements are sequential samples taken at points $k, k-1, \ldots,$ going back in time through the sequence of data samples.

In the single-input case, the adaptive processor can be implemented with an adaptive linear combiner and unit delay elements, as shown in Figure 2.2. The structure in Figure 2.2 is called an adaptive transversal filter. Note that a second subscript, k, has been added to the weights, to make them explicitly time varying. The adaptive transversal filter is the temporal (as opposed to spatial) form of the nonrecursive adaptive filter. It has a wide range of applications in the fields of adaptive modeling and adaptive signal processing. Most of the adaptive systems

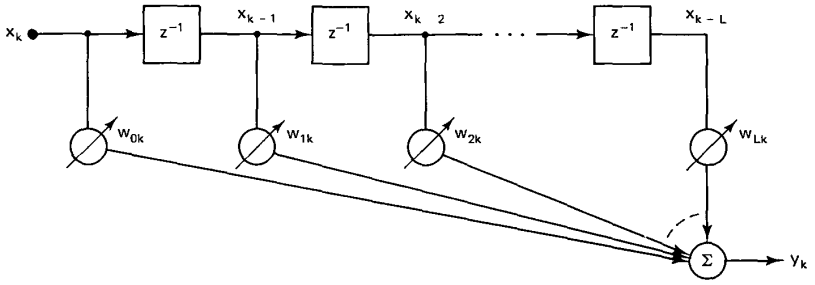


Figure 2.2 Adaptive linear combiner in the form of single-input adaptive transversal filter.

described in later chapters of this book are based on the use of an adaptive transversal filter.

In some multiple-input systems, we need a *bias weight* w_{0k} that simply adds a variable bias into the sum, y_k. We obtain this conveniently where it is needed by setting the first input element, x_{0k} in Figure 2.1 and (2.1), permanently equal to one (or some other constant value) as illustrated in Figure 2.3. The bias weight is not normally required in single-input systems.

From the input signal notation in (2.1) and (2.2), we obtain the input–output relationships for Figures 2.2 and 2.3 as follows:

$$\textit{Single input}: \qquad y_k = \sum_{l=0}^{L} w_{lk} x_{k-l} \tag{2.3}$$

$$\textit{Multiple inputs}: \qquad y_k = \sum_{l=0}^{L} w_{lk} x_{lk} \tag{2.4}$$

When x_{0k} is set identically equal to 1 in (2.4) as just described, w_{0k} becomes a bias weight.

Corresponding with (2.1) and (2.2), we have a weight vector,

$$\mathbf{W}_k = [w_{0k} \quad w_{1k} \quad \cdots \quad w_{Lk}]^{\mathrm{T}} \tag{2.5}$$

With this definition we can express (2.3) and (2.4) in a single relationship using vector notation:†

$$y_k = \mathbf{X}_k^{\mathrm{T}} \mathbf{W}_k = \mathbf{W}_k^{\mathrm{T}} \mathbf{X}_k \tag{2.6}$$

With this description of the operation of the adaptive linear combiner, we can

† We assume that the reader understands vector and matrix multiplication. Any elementary text on matrix algebra should serve as a review.

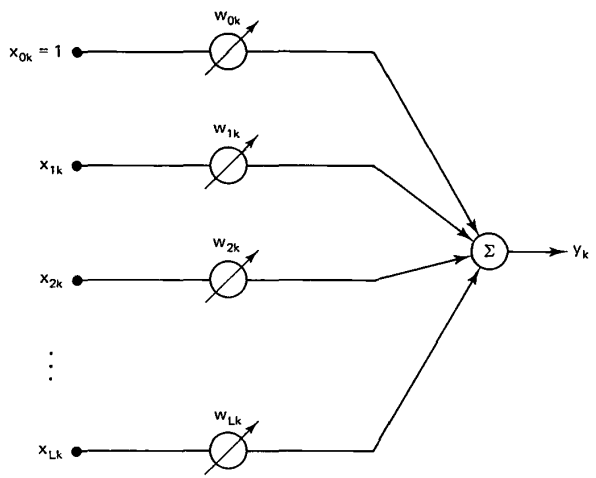


Figure 2.3 Multiple-input adaptive linear combiner with bias weight w_{0k}.

proceed to a discussion of how it adapts, that is, of the effect of changing the vector $\mathbf{W}_k$ as the time index, k, changes.

DESIRED RESPONSE AND ERROR

The adaptive linear combiner can be used in both open- and closed-loop adaptive systems. As discussed in connection with Figure 1.2, the adjustment of the weight vector in open-loop systems does not depend explicitly on properties of the output, but only on the input and on properties of the environment.

With closed-loop systems, however, as in Figure 1.3, the weight vector depends on the output signal as well as other data. Generally, for the adaptive linear combiner the other data include a "desired response" or "training signal." Our discussions in this book will be involved primarily with closed-loop, performance-feedback systems, so we need a thorough understanding of these signals.

In the adaptation process with performance feedback, the weight vector of the linear combiner is adjusted to cause the output, y_k, to agree as closely as possible with the desired response signal. This is accomplished by comparing the output with the desired response to obtain an "error" signal and then adjusting or optimizing the weight vector to minimize this signal. In most practical instances the adaptive process is oriented toward minimizing the mean-square value, or average power of the error signal. Optimization by this criterion, in both adaptive and nonadaptive systems, has been widely practiced in the past and has many advantages [1, 8–16].

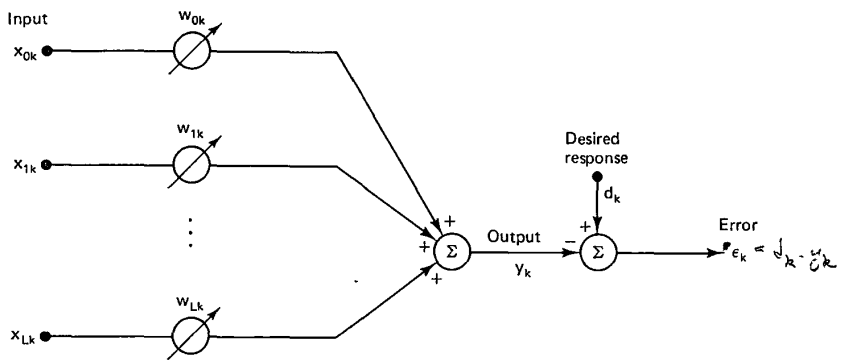


Figure 2.4 Multiple-input adaptive linear combiner with desired response and error signals.

The method of deriving the error signal by means of the desired response input is shown in the multiple-input diagram of Figure 2.4. The output signal, y_k, is simply subtracted from the desired signal, d_k, to produce the error signal, ε_k.

The source of the desired response signal, d_k, depends on the application of the adaptive combiner. For the present we will assume the availability of such a signal. In later chapters we discuss its derivation in more detail. We note, however, that considerable ingenuity is often required to find a suitable signal, since if the actual desired response were available one would generally not need the adaptive system.

We now proceed to a discussion of the performance function, which is a function of the error signal just described.

THE PERFORMANCE FUNCTION

As in Figure 2.4, the error signal with time index k is

$$\varepsilon_k = d_k - y_k \tag{2.7}$$

Substituting (2.6) into this expression yields

$$\varepsilon_k = d_k - \mathbf{X}_k^{\mathrm{T}}\mathbf{W} = d_k - \mathbf{W}^{\mathrm{T}}\mathbf{X}_k \tag{2.8}$$

Here we have dropped the subscript k from the weight vector $\mathbf{W}$ for convenience, because in this discussion we do not wish to adjust the weights. We now square (2.8) to obtain the instantaneous squared error,

$$\varepsilon_k^2 = d_k^2 + \mathbf{W}^{\mathrm{T}}\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}}\mathbf{W} - 2d_k\mathbf{X}_k^{\mathrm{T}}\mathbf{W} \tag{2.9}$$

We assume that ε_k, d_k, and $\mathbf{X}_k$ are statistically stationary and take the expected

value† of (2.9) over k:

$$E\left[\varepsilon_k^2\right] = E\left[d_k^2\right] + \mathbf{W}^{\mathrm{T}}E\left[\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}}\right]\mathbf{W} - 2E\left[d_k\mathbf{X}_k^{\mathrm{T}}\right]\mathbf{W} \tag{2.10}$$

Note that the expected value of any sum is the sum of expected values, but that the expected value of a product is the product of expected values when the variables are statistically independent. The signals x_k and d_k are not generally independent.

The mean-square-error function can be more conveniently expressed as follows. Let $\mathbf{R}$ be defined as the square matrix

$$\mathbf{R} = E\left[\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}}\right] = E\begin{bmatrix} x_{0k}^2 & x_{0k}x_{1k} & x_{0k}x_{2k} & \cdots & x_{0k}x_{Lk} \\ x_{1k}x_{0k} & x_{1k}^2 & x_{1k}x_{2k} & \cdots & x_{1k}x_{Lk} \\ \vdots & \vdots & \vdots & & \vdots \\ x_{Lk}x_{0k} & x_{Lk}x_{1k} & x_{Lk}x_{2k} & \cdots & x_{Lk}^2 \end{bmatrix} \tag{2.11}$$

This matrix is designated the "input correlation matrix." The main diagonal terms are the mean squares of the input components, and the cross terms are the cross correlations among the input components. Let $\mathbf{P}$ be similarly defined as the column vector

$$\mathbf{P} = E\left[d_k\mathbf{X}_k\right] = E\left[d_kx_{0k} \quad d_kx_{1k} \quad \cdots \quad d_kx_{Lk}\right]^{\mathrm{T}} \tag{2.12}$$

This vector is the set of cross correlations between the desired response and the input components. The elements of both $\mathbf{R}$ and $\mathbf{P}$ are all constant second-order statistics when $\mathbf{X}_k$ and d_k are stationary. Note that the multiple-input form of $\mathbf{X}_k$ was used in (2.11) and (2.12), but the single-input form could just as easily have been used.

We now let the mean-square error in (2.10) be designated by ξ and reexpress it in terms of (2.11) and (2.12) as

$$\mathrm{MSE} \triangleq \xi = E\left[\varepsilon_k^2\right] = E\left[d_k^2\right] + \mathbf{W}^{\mathrm{T}}\mathbf{R}\mathbf{W} - 2\mathbf{P}^{\mathrm{T}}\mathbf{W} \tag{2.13}$$

It is clear from this expression that the mean-square error ξ is precisely a quadratic function of the components of the weight vector $\mathbf{W}$ when the input components and desired response input are stationary stochastic variables. That is, when (2.13) is expanded, the elements of $\mathbf{W}$ will appear in first and second degree only.

A portion of a typical two-dimensional mean-square-error function is illustrated in Figure 2.5. The vertical axis represents the mean-square error and the horizontal axes the values of the two weights. The bowl-shaped quadratic error function, or *performance surface*, formed in this manner is a paraboloid (a hyperparaboloid if there are more than two weights). It must be concave upward; otherwise, there would be weight settings that would result in a negative mean-square error, an impossible result with real, physical signals. Contours of constant mean-

† For discussions of statistical concepts such as stationarity, expected value, and so on, we refer to the texts on signal processing and time-series analysis [5, 6].

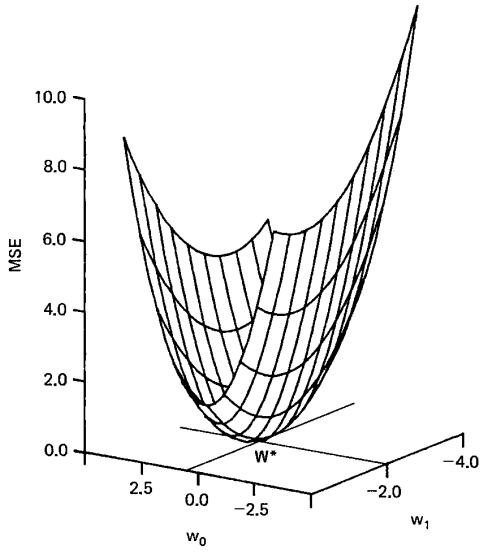


Figure 2.5 Portion of a two-dimensional quadratic performance surface. The mean-square error is plotted vertically, w_0 ranges from -3 to 4, w_1 ranges from -4 to 0, and the optimum weight vector is $\mathbf{W}^* = (0.65, -2.10)$. The minimum MSE is 0.0 in this example.

square error are elliptical, as can be seen by setting ξ constant in (2.13). The point at the "bottom of the bowl" is projected onto the weight-vector plane as $\mathbf{W}^*$, the optimal weight vector or point of minimum mean-square error. With a quadratic performance function there is only a single global optimum; no local minima exist.

GRADIENT AND MINIMUM MEAN-SQUARE ERROR

Many useful adaptive processes that cause the weight vector to seek the minimum of the performance surface do so by gradient methods. The gradient of the mean-square-error performance surface, designated $\nabla(\xi)$ or simply ∇, can be obtained by differentiating (2.13) to obtain the column vector

$$\nabla \triangleq \frac{\partial \xi}{\partial \mathbf{W}} = \begin{bmatrix} \frac{\partial \xi}{\partial w_0} & \frac{\partial \xi}{\partial w_1} & \cdots & \frac{\partial \xi}{\partial w_L} \end{bmatrix}^{\mathrm{T}} \tag{2.14}$$

$$= 2\mathbf{RW} - 2\mathbf{P} \tag{2.15}$$

where $\mathbf{R}$ and $\mathbf{P}$ are given by (2.11) and (2.12), respectively. This expression is obtained by expanding (2.13) and differentiating with respect to each component of the weight vector. Differentiation of the term $\mathbf{W}^{\mathrm{T}}\mathbf{RW}$ can be treated as differentiation of the product $(\mathbf{W}^{\mathrm{T}})(\mathbf{RW})$.

To obtain the minimum mean-square error the weight vector $\mathbf{W}$ is set at its optimal value $\mathbf{W}^*$, where the gradient is zero

$$\nabla = \mathbf{0} = 2\mathbf{RW}^* - 2\mathbf{P} \tag{2.16}$$

Assuming that $\mathbf{R}$ is nonsingular, the optimal weight vector $\mathbf{W}^*$, sometimes called the Wiener weight vector, is found from (2.16) to be

$$\mathbf{W}^* = \mathbf{R}^{-1}\mathbf{P} \tag{2.17}$$

This equation is an expression of the Wiener–Hopf equation [8, 9, 12] in matrix form. The minimum mean-square error is now obtained by substituting $\mathbf{W}^*$ from (2.17) for $\mathbf{W}$ in (2.13):

$$\begin{aligned}\xi_{\min} &= E\left[d_k^2\right] + \mathbf{W}^{*\mathrm{T}}\mathbf{R}\mathbf{W}^* - 2\mathbf{P}^{\mathrm{T}}\mathbf{W}^* \\ &= E\left[d_k^2\right] + [\mathbf{R}^{-1}\mathbf{P}]^{\mathrm{T}}\mathbf{R}\mathbf{R}^{-1}\mathbf{P} - 2\mathbf{P}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{P}\end{aligned} \tag{2.18}$$

We now simplify this result using three rules that are of general utility in discussions of the performance surface:

1. Identity rule for any square matrix: $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$
2. Transpose of a matrix product: $[\mathbf{A}\mathbf{B}]^{\mathrm{T}} = \mathbf{B}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}$
3. Symmetry of the input correlation matrix: $\mathbf{R}^{\mathrm{T}} = \mathbf{R}$; $[\mathbf{R}^{-1}]^{\mathrm{T}} = \mathbf{R}^{-1}$ [see (2.11)].

Using these rules, (2.18) becomes

$$\xi_{\min} = E\left[d_k^2\right] - \mathbf{P}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{P} = E\left[d_k^2\right] - \mathbf{P}^{\mathrm{T}}\mathbf{W}^* \tag{2.19}$$

We now introduce an example to help clarify the concepts of quadratic performance surface, gradient, and mean-square error set forth so far.

EXAMPLE OF A PERFORMANCE SURFACE

A simple example of a single-input adaptive linear combiner with two weights is shown in Figure 2.6. The input and desired signals are sampled sinusoids at the same frequency, with N samples per cycle. We assume that $N > 2$ so that the input

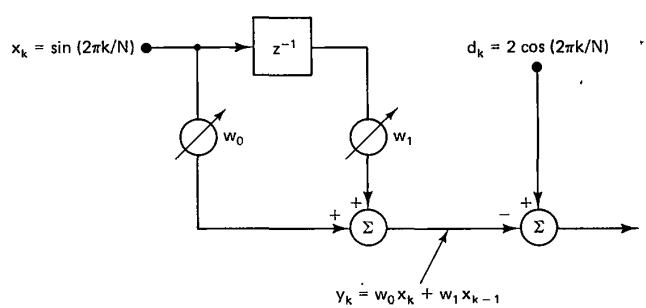


Figure 2.6 Example of an adaptive linear combiner with two weights.

samples are not all zero. We are not concerned here with the origin of these signals, only with the resulting performance surface and its properties.

To obtain the performance function [i.e., ξ in (2.13)], we need the expected signal products in (2.11) and (2.12). Note that we must change the subscripts of x for the single-input case, as in (2.2). The expected products may be found for any product of sinusoidal functions by averaging over one or more periods of the product. Thus,

$$E[x_k x_{k-n}] = \frac{1}{N}\sum_{k=1}^{N} \sin\frac{2\pi k}{N}\sin\frac{2\pi(k-n)}{N}$$

$$= 0.5\cos\frac{2\pi n}{N}; \qquad n = 0, 1 \tag{2.20}$$

$$E[d_k x_{k-n}] = \frac{2}{N}\sum_{k=1}^{N} \cos\frac{2\pi k}{N}\sin\frac{2\pi(k-n)}{N}$$

$$= -\sin\frac{2\pi n}{N}; \qquad n = 0, 1 \tag{2.21}$$

We note further that obviously $E[x_{k-1}^2] = E[x_k^2]$, because the average is over k.

With these results, the input correlation matrix $\mathbf{R}$ and the correlation vector $\mathbf{P}$ can be obtained from (2.11) and (2.12) for this two-dimensional, single-input example:

$$\mathbf{R} = E\begin{bmatrix} x_k^2 & x_k x_{k-1} \\ x_{k-1}x_k & x_{k-1}^2 \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5\cos\frac{2\pi}{N} \\ 0.5\cos\frac{2\pi}{N} & 0.5 \end{bmatrix} \tag{2.22}$$

$$\mathbf{P} = E[d_k x_k \quad d_k x_{k-1}]^{\mathrm{T}} = \left[0 \quad -\sin\frac{2\pi}{N}\right]^{\mathrm{T}} \tag{2.23}$$

As in (2.20) and (2.21) we also obtain $E[d_k^2] = 2$. Using these results in (2.13), we obtain the performance function for this example:

$$\xi = E\left[d_k^2\right] + \mathbf{W}^{\mathrm{T}}\mathbf{R}\mathbf{W} - 2\mathbf{P}^{\mathrm{T}}\mathbf{W}$$

$$= 2 + 0.5[w_0 \quad w_1]\begin{bmatrix} 1 & \cos\frac{2\pi}{N} \\ \cos\frac{2\pi}{N} & 1 \end{bmatrix}\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} - 2\left[0 \quad -\sin\frac{2\pi}{N}\right]\begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

$$= 0.5\left(w_0^2 + w_1^2\right) + w_0 w_1 \cos\frac{2\pi}{N} + 2w_1\sin\frac{2\pi}{N} + 2 \tag{2.24}$$

This performance surface is plotted in Figure 2.5 for $N = 5$ samples per cycle. Note that it is quadratic in w_0 and w_1 and has a single global minimum. The gradient vector at any point w_0, w_1 can be found by substituting (2.22) and (2.23) into (2.15)

and is

$$\nabla = 2\mathbf{RW} - 2\mathbf{P}$$

$$= \begin{bmatrix} 1 & \cos\frac{2\pi}{N} \\ \cos\frac{2\pi}{N} & 1 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} - 2 \begin{bmatrix} 0 \\ -\sin\frac{2\pi}{N} \end{bmatrix}$$

$$= \begin{bmatrix} w_0 + w_1\cos\frac{2\pi}{N} \\ w_0\cos\frac{2\pi}{N} + w_1 + 2\sin\frac{2\pi}{N} \end{bmatrix} \tag{2.25}$$

The Wiener weight vector for this example, $\mathbf{W}^*$, may be found formally from (2.17) by inverting $\mathbf{R}$, or it may be found by setting ∇ equal to zero in (2.25). These are, of course, equivalent operations, and in either case the result is

$$\mathbf{W}^* = \begin{bmatrix} 2\cot\frac{2\pi}{N} & -2\csc\frac{2\pi}{N} \end{bmatrix}^{\mathrm{T}} \tag{2.26}$$

Again, note that $N > 2$ was specified earlier, so that w_0^* and w_1^* in this result are always finite.

Finally, the minimum mean-square error for this example is obtained by substituting (2.23) and (2.26) into (2.19):

$$\xi_{\min} = E\left[d_k^2\right] - \mathbf{P}^{\mathrm{T}}\mathbf{W}^*$$

$$= 2 - \begin{bmatrix} 0 & -\sin\frac{2\pi}{N} \end{bmatrix} \begin{bmatrix} 2\cot\frac{2\pi}{N} \\ -2\csc\frac{2\pi}{N} \end{bmatrix} = 0 \tag{2.27}$$

This result, which says in effect that the weights in Figure 2.6 can be adjusted to reduce ε_k to zero for any value of N, may seem surprising at first. The unit delay by itself can change x_k from a sine into a cosine function only when $N = 4$, that is, only when the unit delay is one-fourth of a cycle. Note that in this case (2.26) gives $w_0^* = 0$ and $w_1^* = -2$. However, with two weights in addition to the delay, the adaptive linear combiner can always shift x_k so that it becomes the proper cosine function, for any N greater than 2 (see also Exercise 3).

ALTERNATIVE EXPRESSION OF THE GRADIENT

Since the mean-square error is a quadratic form in $\mathbf{W}$ which reaches its minimum value when $\mathbf{W}$ equals $\mathbf{W}^*$, one might expect that it could be expressed as

$$\xi = \xi_{\min} + (\mathbf{W} - \mathbf{W}^*)^{\mathrm{T}}\mathbf{R}(\mathbf{W} - \mathbf{W}^*) \tag{2.28}$$

We demonstrate that this expression is valid in the following manner. Noting that in

$E[d_k^2] - P^T W^*$

general $(\mathbf{A} - \mathbf{B})^T$ is just $\mathbf{A}^T - \mathbf{B}^T$, we expand (2.28) to obtain

$$\xi = \xi_{\min} + \mathbf{W}^{*T}\mathbf{R}\mathbf{W}^* + \mathbf{W}^T\mathbf{R}\mathbf{W} - \mathbf{W}^T\mathbf{R}\mathbf{W}^* - \mathbf{W}^{*T}\mathbf{R}\mathbf{W} \tag{2.29}$$

Each term in (2.29) is a scalar and therefore equal to its own transpose. Thus the last two terms are equal. We now combine these and also substitute (2.19) for $\xi_{\min}$, and get

$$\xi = E\left[d_k^2\right] - \mathbf{P}^T\mathbf{W}^* + \mathbf{W}^{*T}\mathbf{R}\mathbf{W}^* + \mathbf{W}^T\mathbf{R}\mathbf{W} - 2\mathbf{W}^T\mathbf{R}\mathbf{W}^* \tag{2.30}$$

Next, substituting (2.17) for $\mathbf{W}^*$ and recalling again that $\mathbf{R}$ is symmetric, we get

$$\begin{aligned}\xi &= E\left[d_k^2\right] - \mathbf{P}^T\mathbf{R}^{-1}\mathbf{P} + \mathbf{P}^T\mathbf{R}^{-1}\mathbf{R}\mathbf{R}^{-1}\mathbf{P} + \mathbf{W}^T\mathbf{R}\mathbf{W} - 2\mathbf{W}^T\mathbf{R}\mathbf{R}^{-1}\mathbf{P} \\ &= E\left[d_k^2\right] + \mathbf{W}^T\mathbf{R}\mathbf{W} - 2\mathbf{W}^T\mathbf{P} \\ &= E\left[d_k^2\right] + \mathbf{W}^T\mathbf{R}\mathbf{W} - 2\mathbf{P}^T\mathbf{W}\end{aligned} \tag{2.31}$$

This result corresponds with (2.13) and thus validates (2.28).

The quadratic form in (2.28) can be expressed more conveniently when we define a weight deviation vector, as follows:

$$\mathbf{V} = \mathbf{W} - \mathbf{W}^* = [v_0 \quad v_1 \quad \cdots \quad v_L]^T \tag{2.32}$$

Accordingly, (2.28) becomes

$$\xi = \xi_{\min} + \mathbf{V}^T\mathbf{R}\mathbf{V} \tag{2.33}$$

The quantity $\mathbf{V}$ is the deviation of the weight vector from the Wiener optimal weight vector. Any departure of $\mathbf{W}$ from $\mathbf{W}^*$ would cause an excess mean-square error according to the quadratic form $\mathbf{V}^T\mathbf{R}\mathbf{V}$.

In order that ξ be nonnegative for all possible $\mathbf{V}$, it is necessary that $\mathbf{V}^T\mathbf{R}\mathbf{V} \geqq 0$ for all $\mathbf{V}$. When $\mathbf{V}^T\mathbf{R}\mathbf{V} > 0$ for all $\mathbf{V} \neq 0$, the matrix $\mathbf{R}$ is said to be "positive definite" [7]. When $\mathbf{V}^T\mathbf{R}\mathbf{V} = 0$ for certain finite values of $\mathbf{V}$ or for all $\mathbf{V}$, the matrix $\mathbf{R}$ is said to be "positive semidefinite." In physical situations, $\mathbf{R}$ will almost always be positive definite, but a positive semidefinite $\mathbf{R}$ could occur. Texts on matrix theory generally discuss conditions of positive definiteness and positive semidefiniteness.

The gradient of the mean-square error with respect to $\mathbf{V}$ is obtained by differentiating (2.33):

$$\frac{\partial \xi}{\partial \mathbf{V}} = \left[\frac{\partial \xi}{\partial v_0} \quad \frac{\partial \xi}{\partial v_1} \quad \cdots \quad \frac{\partial \xi}{\partial v_L}\right] = 2\mathbf{R}\mathbf{V} \tag{2.34}$$

This gradient is the same as that given by (2.15), because $\mathbf{W}$ and $\mathbf{V}$ differ only by a constant. Thus

$$\nabla = \frac{\partial \xi}{\partial \mathbf{W}} = \frac{\partial \xi}{\partial \mathbf{V}} = 2\mathbf{R}\mathbf{V} = 2(\mathbf{R}\mathbf{W} - \mathbf{P}) \tag{2.35}$$

Expression (2.35) will be used in developing and analyzing a variety of adaptive algorithms.

DECORRELATION OF ERROR AND INPUT COMPONENTS

A useful and important statistical condition exists between the error signal and the components of the input signal vector when $\mathbf{W} = \mathbf{W}^*$. Recall from (2.8) that

$$\varepsilon_k = d_k - \mathbf{X}_k^{\mathrm{T}}\mathbf{W} \tag{2.36}$$

We multiply both sides of this equation by $\mathbf{X}_k$. Since each term is a scalar, we can put $\mathbf{X}_k$ on either side of each term. Thus

$$\varepsilon_k \mathbf{X}_k = d_k \mathbf{X}_k - \mathbf{X}_k \mathbf{X}_k^{\mathrm{T}}\mathbf{W} \tag{2.37}$$

Next, we take the expected value of (2.37) and obtain

$$E[\varepsilon_k \mathbf{X}_k] = \mathbf{P} - \mathbf{R}\mathbf{W} \tag{2.38}$$

Finally, we let $\mathbf{W}$ take its optimum value, $\mathbf{W}^* = \mathbf{R}^{-1}\mathbf{P}$ in (2.17), and get

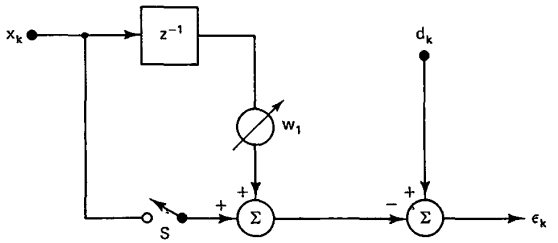
$$\{E[\varepsilon_k \mathbf{X}_k]\}_{\mathbf{W}=\mathbf{W}^*} = \mathbf{P} - \mathbf{P} = \mathbf{0} \tag{2.39}$$

This result is the same as the well-known result of Wiener filter theory: that when the impulse response of a filter is optimized, the error signal is uncorrelated with (orthogonal to) the input signals to the weights.

EXERCISES

1. Much of the discussion in this chapter and in subsequent chapters uses matrix algebra. Prove that the following simple relationships are true for any matrices **A**, **B**, and **C**:
   (a) $\mathbf{AB} \neq \mathbf{BA}$ in general.
   (b) $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$.
   (c) $(\mathbf{AB})^{\mathrm{T}} = \mathbf{B}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}$ and $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$.
   (d) If **A** is symmetric, then $\mathbf{A}^{-1}$ is symmetric.
2. Starting with Equation (2.13), give a detailed derivation of Equation (2.15).
3. In the adaptive linear combiner example shown in Figure 2.6, let $N = 10$. Then:
   (a) Find the optimum weight vector.
   (b) Write an expression for y_k using the answer to part (a).
   (c) Using the result from part (b) and x_k from Figure 2.6, prove that $y_k = d_k$.
4. In Figure 2.6, what weight values will produce a root-mean-square (rms) value of ε_k equal to 2:
   (a) With $N = 5$?
   (b) With $N = 10$?
5. On the performance surface of Figure 2.6, with $N = 8$, what is the gradient vector when $w_1 = 0$ and the mean-squared value of ε_k is:
   (a) 2.0?
   (b) 4.0?
   Why is the gradient steeper in the latter case?

6. Consider the system shown here, which is an adaptive linear combiner with a single weight. Suppose that switch S is open and $E[x_k^2] = 1$, $E[x_k x_{k-1}] = 0.5$, $E[d_k^2] = 4$, $E[d_k x_k] = -1$, and $E[d_k x_{k-1}] = 1$. Derive an expression for the performance function. Make a plot of the performance function.



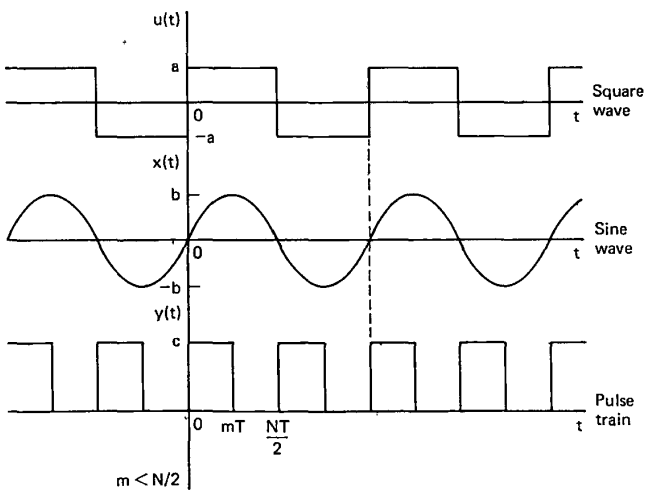
7. Do Exercise 6 with switch S closed.

8. What is the optimum value of w_1 for Exercise 6? What is the corresponding minimum mean-square error?

9. In the adaptive linear combiner diagram of Exercise 6, let x_k and d_k be as given in Figure 2.6, and let $N = 5$. With switch S open:
   (a) Find and plot an expression for ξ.
   (b) Find the optimum value of w_1.
   (c) Find the minimum value of ξ.

10. Do Exercise 9 with switch S closed.

11. The student should have some experience with correlation functions such as those derived in Equations (2.20) and (2.21). Consider the following continuous, periodic waveforms:

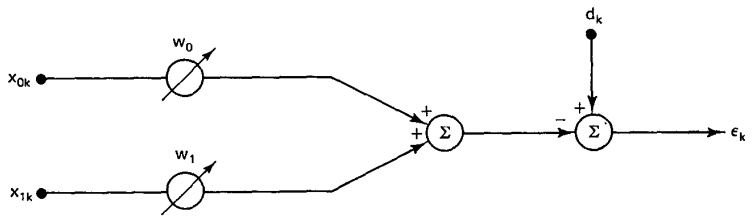


Suppose that each waveform is sampled at $t = 0, T, 2T, \ldots,$ such that there are exactly N samples per cycle in the first two cases, and $N/2$ samples per cycle in the third case. If a sample is taken where $f(t)$ is discontinuous, give it the value just to the right of the discontinuity.

(a) Find $E[u_k u_{k-n}]$.
(b) Find $E[y_k y_{k-n}]$.
(c) Find $E[u_k x_{k-n}]$.
(d) Find $E[u_k y_{k-n}]$.
(e) Find $E[x_k y_{k-n}]$.

12. By using an example from Exercise 11, illustrate that in general, autocorrelation is an even function of the shift variable n, but cross correlation is not.

13. Consider the adaptive linear combiner shown below. Suppose that we wish to minimize $E[\varepsilon_k^4]$ rather than $E[\varepsilon_k^2]$. Assume that the signals x_{0k}, x_{1k}, and d_k are all stationary.
(a) Derive an expression for $E[\varepsilon_k^4]$.
(b) Is $E[\varepsilon_k^4]$ quadratic in w_0 and w_1?
(c) Is $E[\varepsilon_k^4]$ a unimodal function of w_0 and w_1?



ANSWERS TO SELECTED EXERCISES

3. (a) $\mathbf{W}^* = [2.753 \quad -3.403]^{\mathrm{T}}$
(b) $y_k = 2.753 \sin(k\pi/5) - 3.403 \sin[(k-1)\pi/5]$

4. (a) $w_0^2 + w_1^2 + 0.6180 w_0 w_1 + 1.9021 w_1 - 4 = 0$
(b) $w_0^2 + w_1^2 + 1.6180 w_0 w_1 + 1.1756 w_1 - 4 = 0$

5. (a) $\nabla = [0 \quad 1.4142]^{\mathrm{T}}$
(b) $\nabla = [2 \quad 2.8284]^{\mathrm{T}}, [-2 \quad 0]^{\mathrm{T}}$

6. $\xi = w_1^2 - 2w_1 + 4;\ w_1^* = 1$

7. $\xi = w_1^2 - w_1 + 7;\ w_1^* = 0.5$

8. $w_1^* = 1;\ \xi_{\min} = 3$

13. (a) $E[\varepsilon_k^4] = w_0^4 E[x_{0k}^4] + w_1^4 E[x_{1k}^4] - 4w_0^3 E[d_k x_{0k}^3] + \cdots + 12 w_0 w_1 E[d_k x_{0k} x_{1k}] - 4w_0 E[d_k^3 x_{0k}] - 4w_1 E[d_k^3 x_{1k}] + E[d_k^4]$
(b) No
(c) Not in general

REFERENCES AND ADDITIONAL READINGS

1. B. Widrow, "Adaptive Filters," in *Aspects of Network and System Theory*, R. E. Kalman and N. De Claris (Eds.). New York: Holt, Rinehart and Winston, 1970, p. 563.
2. A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Englewood Cliffs, N.J.: Prentice-Hall, 1975, Chap. 8.
3. L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Englewood Cliffs, N.J.: Prentice-Hall, 1975.
4. N. Ahmed and T. Natarajan, *Discrete-Time Signals and Systems*. Reston, Va.: Reston, 1983.
5. L. H. Koopmans, *The Spectral Analysis of Time Series*. New York: Academic Press, 1974.
6. S. D. Stearns, *Digital Signal Analysis*. Rochelle Park, N.J.: Hayden, 1975, Chap. 13.
7. L. G. Kelly, *Handbook of Numerical Methods and Applications*. Reading, Mass.: Addison-Wesley, 1967, Sec. 7.8.
8. N. Wiener, *Extrapolation, Interpolation and Smoothing of Stationary Time Series, with Engineering Applications*. New York: Wiley, 1949.
9. H. W. Bode and C. E. Shannon, "A simplified derivation of linear least squares smoothing and prediction theory," *Proc. IRE*, vol. 38, pp. 417–425, Apr., 1950.
10. R. E. Kalman, "On the general theory of control," in *Proc. First IFAC Congress*. London: Butterworth, 1960.
11. R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Trans. ASME, Ser. D, J. Basic Eng.*, vol. 83, pp. 95–107, Dec. 1961.
12. T. Kailath, "A view of three decades of linear filtering theory," *IEEE Trans. Inf. Theory*, vol. IT-20, pp. 145–181, Mar. 1974.
13. B. Widrow and M. E. Hoff, Jr., "Adaptive switching circuits," *IRE WESCON Conv. Rec.*, pt. 4, pp. 96–104, 1960.
14. J. S. Koford and G. F. Groner, "The use of an adaptive threshold element to design a linear optimal pattern classifier," *IEEE Trans. Inf. Theory*, vol. IT-12, pp. 42–50, Jan. 1966.
15. D. Gabor, W. P. L. Wilby, and R. Woodcock, "A universal nonlinear filter predictor and simulator which optimizes itself by a learning process," *Proc. Inst. Electr. Eng.*, vol. 108B, July 1960.
16. R. W. Lucky, J. Salz, and E. J. Weldon, Jr., *Principles of Data Communication*. New York: McGraw-Hill, 1968.

This material may be protected by Copyright law (Title 17 U.S. Code)

part III

ADAPTIVE ALGORITHMS AND STRUCTURES

(Chapters 6 – 8)

OBJECTIVES OF PART III

From Parts I and II we have most of the concepts necessary to the derivation and analysis of practical adaptive algorithms. These will be algorithms that work in nonstationary as well as stationary signal environments, algorithms that are meant to "track" as well as "seek" the minimum point on the performance surface.

The primary objective of Part III is to help the reader understand the LMS algorithm, which is the simplest scheme for adjusting the weights in a linear adaptive processor. The LMS algorithm, which is covered in Chapter 6, is widely applicable to all of the types of adaptive systems described in Part IV. A central objective of this text is to describe the LMS algorithm and its applications.

A second objective in Part III is to introduce types of adaptive algorithms other than the LMS algorithm, some of which have important advantages. To facilitate this, we introduce certain frequency-domain analysis techniques in Chapter 7. These are standard techniques of digital signal analysis, and the reader familiar with this subject may wish to skim Chapter 7.

The final objective in Part III is to acquaint the reader with lattice structures, which are introduced in the last part of Chapter 8. The lattice and its use in adaptive systems is a large subject, and only some basic ideas are covered in Chapter 8.

chapter 6

The LMS Algorithm

In Chapter 4 we introduced two algorithms, Newton's method and the steepest-descent method, for descending toward the minimum on the performance surface. Both of these algorithms required an estimation of the gradient at each iteration, so in Chapter 5 we discussed general methods for estimating the gradient. The gradient estimation methods in Chapter 5 are general because they are based on taking differences between estimated points on the performance surface, that is, differences between estimates of ξ.

In this chapter we introduce another algorithm for descending on the performance surface, known as the least-mean-square algorithm, or LMS algorithm. The LMS algorithm uses a special estimate of the gradient that is valid for the adaptive linear combiner described in Chapter 2.† Thus the LMS algorithm is more restricted in its use than the algorithms in Chapter 4.

On the other hand, the LMS algorithm is important because of its simplicity and ease of computation, and because it does not require off-line gradient estimations or repetitions of data. If the adaptive system is an adaptive linear combiner, and if the input vector X_k and the desired response d_k are available at each iteration, the LMS algorithm is generally the best choice for many different applications of adaptive signal processing.

DERIVATION OF THE LMS ALGORITHM

We recall that the adaptive linear combiner of Chapter 2 was applied in two basic ways, depending on whether the input is available in parallel (multiple inputs) or serial (single input) form. These two ways are shown together in Figure 6.1. In both

†However, it is possible to extend the LMS algorithm to recursive adaptive filters (see Chapter 8).

cases we have the combiner output, y_k, as a linear combination of the input samples. As in (2.8), we have

$$\varepsilon_k = d_k - \mathbf{X}_k^{\mathrm{T}}\mathbf{W}_k \tag{6.1}$$

where $\mathbf{X}_k$ is the vector of input samples in either of the two configurations in Figure 6.1.

To develop an adaptive algorithm using the previous methods, we would estimate the gradient of $\xi = E[\varepsilon_k^2]$ by taking differences between short-term averages of ε_k^2. Instead, to develop the LMS algorithm, we take ε_k^2 itself as an estimate of ξ_k. Then, at each iteration in the adaptive process, we have a gradient estimate of the form

MSE

$$\frac{\partial \xi}{\partial w}$$

$$\hat{\nabla}_k = \begin{bmatrix} \dfrac{\partial \varepsilon_k^2}{\partial w_0} \\ \vdots \\ \dfrac{\partial \varepsilon_k^2}{\partial w_L} \end{bmatrix} = 2\varepsilon_k \begin{bmatrix} \dfrac{\partial \varepsilon_k}{\partial w_0} \\ \vdots \\ \dfrac{\partial \varepsilon_k}{\partial w_L} \end{bmatrix} = -2\varepsilon_k \mathbf{X}_k \tag{6.2}$$

The derivatives of ε_k with respect to the weights follow directly from (6.1).

With this simple estimate of the gradient, we can now specify a steepest-descent type of adaptive algorithm. From (4.36), we have

$$\begin{aligned} \mathbf{W}_{k+1} &= \mathbf{W}_k - \mu\hat{\nabla}_k \\ &= \mathbf{W}_k + 2\mu\varepsilon_k\mathbf{X}_k \end{aligned} \tag{6.3}$$

This is the LMS algorithm [3, 4]. As before, μ is the gain constant that regulates the speed and stability of adaptation. Since the weight changes at each iteration are based on imperfect gradient estimates, we would expect the adaptive process to be noisy; that is, it would not follow the true line of steepest descent on the performance surface.

From its form in (6.3), we can see that the LMS algorithm can be implemented in a practical system without squaring, averaging, or differentiation and is elegant in its simplicity and efficiency. As noted above, each component of the gradient vector is obtained from a single data sample without perturbing the weight vector. Without averaging, the gradient components do contain a large component of noise, but the noise is attenuated with time by the adaptive process, which acts as a low-pass filter in this respect. We now proceed to discuss some of the properties of the LMS algorithm and to illustrate its use in some examples.

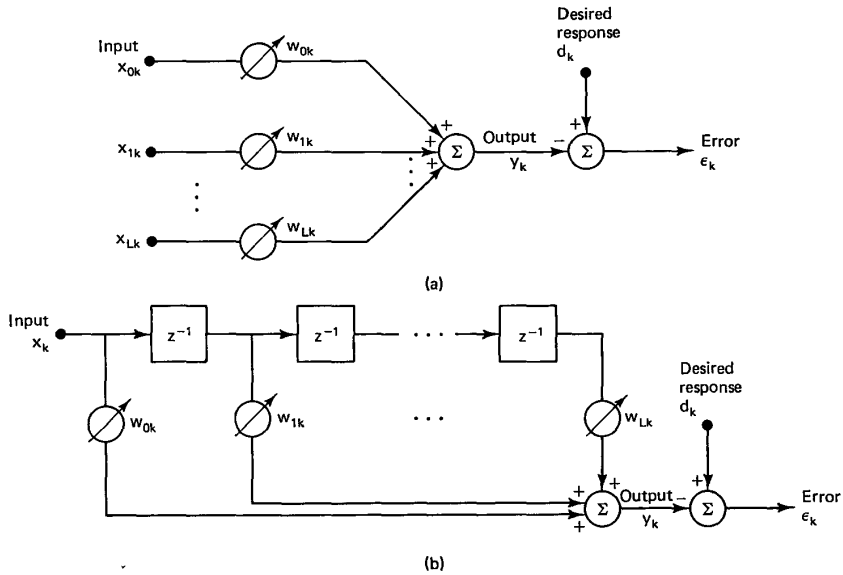


Figure 6.1 The adaptive linear combiner: (a) in general form; (b) as a transversal filter.

CONVERGENCE OF THE WEIGHT VECTOR

As with all adaptive algorithms, a primary concern with the LMS algorithm is its convergence to the optimum weight vector solution, where $E[\varepsilon_k^2]$ is minimized. To examine LMS convergence, we first note that the gradient estimate in (6.2) can readily be shown to be unbiased when the weight vector is held constant. The expected value of (6.2) with $\mathbf{W}_k$ held equal to $\mathbf{W}$ is

$$\begin{aligned} E[\hat{\nabla}_k] &= -2E[\varepsilon_k \mathbf{X}_k] \\ &= -2E[d_k\mathbf{X}_k - \mathbf{X}_k\mathbf{X}_k^{\mathrm{T}}\mathbf{W}] \\ &= 2(\mathbf{RW} - \mathbf{P}) = \nabla \end{aligned} \tag{6.4}$$

The second line of (6.4) follows from (6.1) plus the fact that ε_k is a scalar and can thus be commuted. The last line follows from the definitions of $\mathbf{P}$ and $\mathbf{R}$ in Chapter 2, and from (2.15). Since the mean value of $\hat{\nabla}_k$ is equal to the true gradient ∇, $\hat{\nabla}_k$ must be an unbiased estimate.

Seeing that the gradient estimate is unbiased, we could make the LMS algorithm into a true steepest-descent algorithm, at least in the limiting case, by estimating ∇ at each step as in (6.2) but not adapting the weights until many steps have occurred. In this way $\hat{\nabla}_k$ could be made to approach ∇_k, and (6.3) to approach

(4.36). With the weight vector changing at each iteration, we need to examine the weight vector convergence in a different manner, as follows.

From (6.3) we can see that the weight vector $\mathbf{W}_k$ is a function only of the past input vectors $\mathbf{X}_{k-1}, \mathbf{X}_{k-2}, \ldots, \mathbf{X}_0$. If we assume that successive input vectors are independent over time, $\mathbf{W}_k$ is independent of $\mathbf{X}_k$. For stationary input processes meeting this condition, the expected value of the weight vector $E[\mathbf{W}_k]$ after a sufficient number of iterations can be shown as follows to converge to the Wiener optimal solution given by (2.17), that is, to $\mathbf{W}^* = \mathbf{R}^{-1}\mathbf{P}$.

Taking the expected value of both sides of (6.3) yields the difference equation

$$\begin{aligned} E[\mathbf{W}_{k+1}] &= E[\mathbf{W}_k] + 2\mu E[\varepsilon_k \mathbf{X}_k] \\ &= E[\mathbf{W}_k] + 2\mu\left(E[d_k\mathbf{X}_k] - E[\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}}\mathbf{W}_k]\right) \end{aligned} \tag{6.5}$$

Using the foregoing assumption that $\mathbf{X}_k$ and $\mathbf{W}_k$ are independent, we have the expected products as in (6.4). Also, from (2.17) we have the optimum weight vector given as $\mathbf{W}^* = \mathbf{R}^{-1}\mathbf{P}$. Thus (6.5) becomes

$$\begin{aligned} E[\mathbf{W}_{k+1}] &= E[\mathbf{W}_k] + 2\mu(\mathbf{P} - \mathbf{R}E[\mathbf{W}_k]) \\ &= (\mathbf{I} - 2\mu\mathbf{R})E[\mathbf{W}_k] + 2\mu\mathbf{R}\mathbf{W}^* \end{aligned} \tag{6.6}$$

But now we have just the expected form of (4.38), which was solved (i.e., made nonrecursive) by changing to the principal-axis coordinate system. Using expected values, the solution from (4.42) is

$$E[\mathbf{V}_k'] = (\mathbf{I} - 2\mu\Lambda)^k\mathbf{V}_0' \tag{6.7}$$

where $\mathbf{V}'$ is the weight vector, $\mathbf{W}$, in the principal-axis system, Λ is the diagonal eigenvalue matrix of $\mathbf{R}$, and $\mathbf{V}_0'$ is the initial weight vector in the principal-axis system.

Thus, as k increases without bound, we see that the expected weight vector in (6.7) reaches the optimum solution (i.e., zero in the principal-axis system) only if the right side of the equation converges to zero. We saw in (4.45) that such convergence is guaranteed only if

$$\frac{1}{\lambda_{\max}} > \mu > 0 \tag{6.8}$$

where $\lambda_{\max}$ is the largest eigenvalue, that is, the largest diagonal element in Λ.

So, in (6.8), we have bounds on μ for convergence of the weight vector mean to the optimum weight vector. Within these bounds, the speed of adaptation and also the noise in the weight vector solution are determined by the size of μ. We also note that $\lambda_{\max}$ cannot be greater than the trace of $\mathbf{R}$, which is the sum of the diagonal elements of $\mathbf{R}$, that is,

$$\begin{aligned} \lambda_{\max} \le \mathrm{tr}[\Lambda] &= \sum \text{(diagonal elements of } \Lambda) \\ &= \sum \text{(diagonal elements of } \mathbf{R}) = \mathrm{tr}[\mathbf{R}] \end{aligned} \tag{6.9}$$

Furthermore, with a transversal adaptive filter, (2.11) gives tr[**R**] as just $(L+1)E[x_k^2]$, or $L+1$ times the input signal power. Thus convergence of the weight-vector mean is assured by:

$$\textit{In general}: \quad 0 < \mu < \frac{1}{\mathrm{tr}[\mathbf{R}]}$$

$$\textit{Transversal filter}: \quad 0 < \mu < \frac{1}{(L+1)(\text{signal power})} \tag{6.10}$$

This is a more restrictive bound on μ than (6.8), but is much easier to apply because the elements of **R** and the signal power can generally be estimated more easily than the eigenvalues of **R**.

The assumptions of decorrelation and stationarity of the input vector used to derive the results in this section are not necessary conditions for convergence of the LMS algorithm but have been adopted in this chapter for analytic convenience. Convergence with certain correlated and nonstationary inputs is demonstrated in the literature on the LMS algorithm [2, 6]. Under these conditions the analysis becomes much more complex. We know of no unconditional proof of convergence of the LMS algorithm.

AN EXAMPLE OF CONVERGENCE

To illustrate the convergence of the mean weight vector when employing the LMS algorithm in a specific situation, we again consider the simple two-weight combiner used as an example in Figure 2.6. We modify the input by adding a random signal as in Figure 6.2, in order to introduce a statistical fluctuation into the adaptive process. The input correlation matrix **R**, which was determined in (2.22), must be modified to include the random signal as follows. Let ϕ represent the average

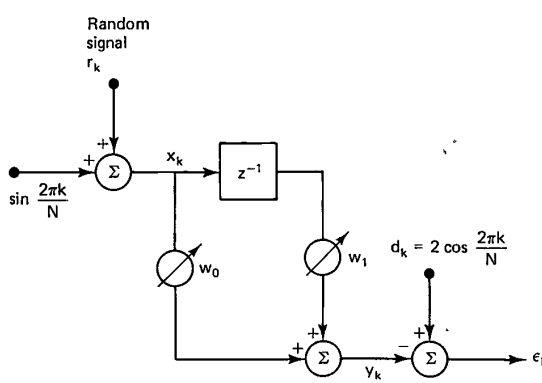


Figure 6.2 The adaptive linear combiner example from Chapter 2 with a random signal added at the input.

random signal power, that is,

$$\phi = E\left[r_k^2\right] \tag{6.11}$$

and suppose the random samples are independent of each other. Then

$$E\left[x_k^2\right] = E\left[\left(\sin\frac{2\pi k}{N} + r_k\right)^2\right] = 0.5 + \phi$$

$$E\left[x_k x_{k-1}\right] = E\left[\left(\sin\frac{2\pi k}{N} + r_k\right)\left(\sin\frac{2\pi(k-1)}{N} + r_{k-1}\right)\right]$$

$$= 0.5\cos\frac{2\pi}{N} \tag{6.12}$$

Thus the modified version of **R** in (2.22) becomes

$$\mathbf{R} = 0.5\begin{bmatrix} 1 + 2\phi & \cos\frac{2\pi}{N} \\ \cos\frac{2\pi}{N} & 1 + 2\phi \end{bmatrix} \tag{6.13}$$

A similar modification in (2.24) gives the performance surface for this example as

$$\xi = (0.5 + \phi)\left(w_0^2 + w_1^2\right) + w_0 w_1 \cos\frac{2\pi}{N} + 2w_1 \sin\frac{2\pi}{N} + 2 \tag{6.14}$$

The optimum weight vector, $\mathbf{W}^*$, can be found by solving $\mathbf{RW}^* = \mathbf{P}$ as described in Chapter 2, with **P** given in (2.23). The result is

$$\begin{bmatrix} 1 + 2\phi & \cos\frac{2\pi}{N} \\ \cos\frac{2\pi}{N} & 1 + 2\phi \end{bmatrix}\begin{bmatrix} w_0^* \\ w_1^* \end{bmatrix} = 2\begin{bmatrix} 0 \\ -\sin\frac{2\pi}{N} \end{bmatrix}$$

$$\begin{bmatrix} w_0^* \\ w_1^* \end{bmatrix} = \begin{bmatrix} \dfrac{2\cos(2\pi/N)\sin(2\pi/N)}{(1 + 2\phi)^2 - \cos^2(2\pi/N)} \\ \dfrac{-2(1 + 2\phi)\sin(2\pi/N)}{(1 + 2\phi)^2 - \cos^2(2\pi/N)} \end{bmatrix} \tag{6.15}$$

The performance of the LMS algorithm in (6.3) is illustrated in Figure 6.3 with $N = 16$ samples per signal cycle, and with a random signal power $\phi = 0.01$. Contours of ξ in (6.14) are shown in this figure, and we note that (6.15) gives $w_0^* = 3.784$ and $w_1^* = -4.178$ as the optimum weight values.

Together with the contours of constant ξ in the figure, there are two weight-value tracks. The two tracks have the following characteristics:

	Starting weight values	Value of μ	Number of iterations
Upper track	0, 0	0.10	250
Lower track	4, −10	0.05	500

The nature of the steepest-descent algorithm can be seen in that both tracks are more or less normal to the ξ contours, similar to Figure 4.7.

Because of the noisy gradient estimate at each iteration, the weight tracks in Figure 6.3 are seen to be erratic; they do not always proceed in the direction of the true gradient. The track for the larger value of μ is more erratic because the weight adjustment is greater at each iteration, but it also arrives at about the same distance from the minimum of ξ in about half the iterations taken for the track with smaller μ. If the LMS algorithm in Figure 6.3 were made to continue, either track would wander around the "bottom of the bowl," in the vicinity of the minimum of ξ,

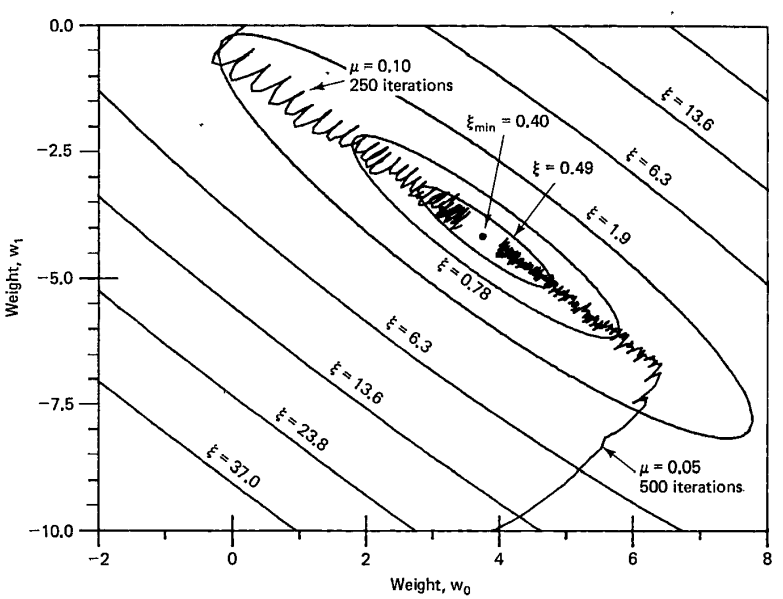


Figure 6.3 Performance surface contours and weight-value tracks for the LMS algorithm operating as in Figure 6.2 with $N = 16$ and $E[r_k^2] = 0.01$.

producing a noisy weight vector solution and an average misadjustment as discussed below.

We can observe the LMS convergence process in still another way, by plotting the error, ε_k, versus the iteration number. An example is given in Figure 6.4, with the error being the same as for the upper weight-value track in Figure 6.3. At first the error is quite sinusoidal, but as the adaptive filter learns to cancel the sinusoidal component, the error becomes increasingly random.

Finally, we note in the particular case of Figure 6.3 that the values of μ (i.e., 0.05 and 0.10) are well below the upper bound given by (6.10). With $\phi = 0.01$ we have from (6.9) and (6.13) the trace of $\mathbf{R}$ as

$$\mathrm{tr}[\mathbf{R}] = 2(0.5 + \phi) = 1.02 \tag{6.16}$$

Therefore, (6.10) gives

$$0.98 > \mu > 0 \tag{6.17}$$

With the LMS algorithm, it is typical to use values of μ on the order of a tenth of the upper bound given in (6.10).

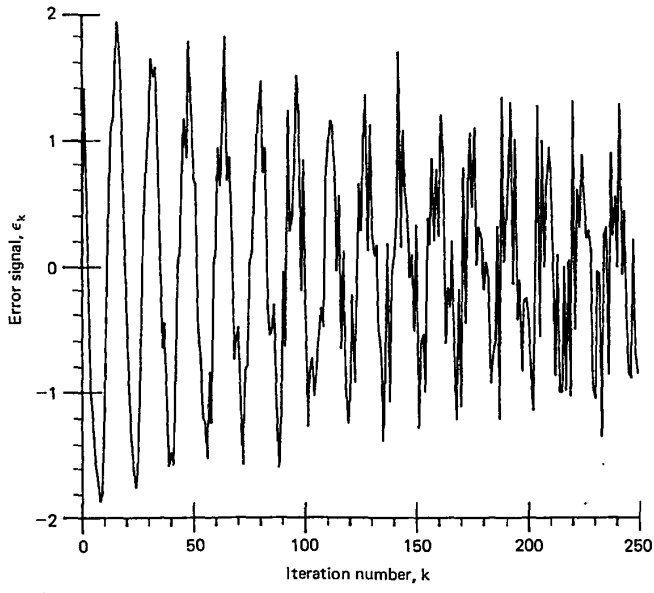


Figure 6.4 Error versus iteration number for the upper track in Figure 6.3.

LEARNING CURVE

In (4.59) we arrived at a formula for the learning curve, that is, the curve of ξ versus the iteration number k, for the steepest-descent method. The curve was theoretical in the sense that an exact knowledge of the gradient at each iteration was assumed. The theoretical learning curve was seen to decay in accordance with $L + 1$ geometric ratios of the form

$$r_n = 1 - 2\mu\lambda_n; \qquad n = 0, 1, \ldots, L \tag{6.18}$$

As a consequence of this result and of (5.86), the time constant for the exponential relaxation of the nth weight vector element to its optimal value is

$$\tau_n \approx \frac{1}{2\mu\lambda_n}; \qquad n = 0, 1, \ldots, L \tag{6.19}$$

From (5.87), the time constant associated with the nth mode of the learning curve is $\tau_n/2$, that is,

$$(\tau_{\text{mse}})_n \approx \frac{1}{4\mu\lambda_n}; \qquad n = 0, 1, \ldots, L \tag{6.20}$$

Furthermore, since each gradient estimate in the LMS algorithm is based on a single data observation, the time constant in terms of the input sample index, T_{mse}, is the same as τ_{mse}, the time constant in terms of algorithm iterations. Thus, for the LMS algorithm,

$$(T_{\text{mse}})_n = (\tau_{\text{mse}})_n \approx \frac{1}{4\mu\lambda_n}; \qquad n = 0, 1, \ldots, L \tag{6.21}$$

In some cases where the LMS algorithm is used, namely those where ε_k^2 at each iteration is a good approximation to $E[\varepsilon_k^2]$ for the current set of weight values, (6.21) gives a good approximation to the learning-curve time constants. But in general with the LMS algorithm, (6.21) will be somewhat optimistic, because ε_k^2 is not generally a good approximation to $E[\varepsilon_k^2]$, and the convergence process is erratic as in Figure 6.3. As an example we take the case illustrated in Figure 6.3, where the eigenvalues of $\mathbf{R}$ are found as follows, using (3.2) and (6.13) with $\phi = 0.01$ and $N = 16$:

$$\det\begin{bmatrix} 0.51 - \lambda & 0.5\cos\frac{\pi}{8} \\ 0.5\cos\frac{\pi}{8} & 0.51 - \lambda \end{bmatrix} = 0 \qquad \lambda_1, \lambda_2 = 0.972, 0.048 \tag{6.22}$$

For the lower weight track in the figure with $\mu = 0.05$, we have from (6.21)

$$(T_{\text{mse}})_1 \approx 5 \text{ iterations} \tag{6.23}$$

$$(T_{\text{mse}})_2 \approx 104 \text{ iterations} \tag{6.24}$$

For comparison with these values, a typical learning curve for this same situation is shown in Figure 6.5. Here ξ, or $E[\varepsilon_k^2]$, is estimated by taking an average of 500 individual runs of ε_k^2 versus k. Each run had a different random sequence and a different starting point on the sinusoidal signals in Figure 6.2. We observe on the logarithmic plot in Figure 6.5 two different slopes of the learning curve, corresponding with the two directions of the lower path in Figure 6.3 and with the two time constants in (6.23) and (6.24). At first, the steeper slope is estimated to be approximately one decade in 13 iterations. Since a decade is a factor of 10 or $e^{2.30}$, a decade also represents a lapse of 2.30 time constants. Thus the initial time constant, $(T_{\text{mse}})_1$, is observed to be on the order of 13/2.30, or 6 iterations. After relaxation of this initial mode, the secondary time constant, $(T_{\text{mse}})_2$, is similarly observed to be approximately 265/2.3, or 115 iterations. In summary:

	Theoretical	Experimental
$(T_{\text{mse}})_1$	5	6
$(T_{\text{mse}})_2$	104	115

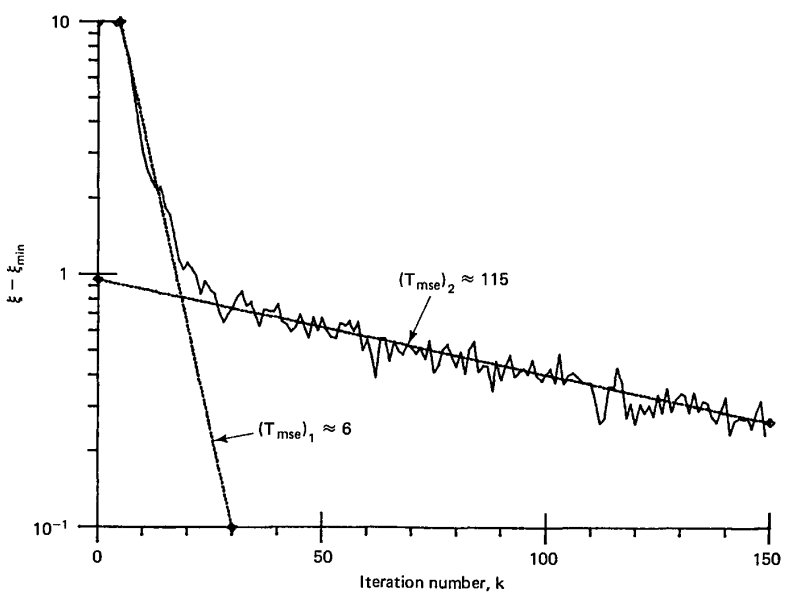


Figure 6.5 Learning curve for the lower track in Figure 6.3, with $\mu = 0.05$. ξ is estimated by averaging 500 runs as described in the text.

Thus, due to the noisy estimate of $E[\varepsilon_k^2]$ discussed previously, the experimental time constants are seen to be somewhat above the theoretical time constants. Such a result is typical with the LMS algorithm.

NOISE IN THE WEIGHT-VECTOR SOLUTION

In Chapter 5 we discussed the variance of the gradient estimate and the noise in the weight-vector solution under the assumption that the gradient estimate was based on weight perturbation. With the LMS algorithm, the gradient estimate as given by (6.2) is not based on weight perturbation, so we must reexamine its variance.

Just as in (5.34), let us define $\mathbf{N}_k$ as a vector of noise in the gradient estimate at the kth iteration. Thus, as in (5.34),

$$\hat{\nabla}_k = \nabla_k + \mathbf{N}_k \tag{6.25}$$

If we assume that the LMS process, using a small value of the adaptive gain constant μ, has converged to a steady-state weight vector solution near $\mathbf{W}^*$, then ∇_k in (6.25) will be close to zero. Then, in accordance with (6.2), the gradient noise is close to

$$\mathbf{N}_k = \hat{\nabla}_k = -2\varepsilon_k \mathbf{X}_k \tag{6.26}$$

The covariance of the noise is thus given by

$$\operatorname{cov}[\mathbf{N}_k] = E\left[\mathbf{N}_k \mathbf{N}_k^{\mathrm{T}}\right] = 4E\left[\varepsilon_k^2 \mathbf{X}_k \mathbf{X}_k^{\mathrm{T}}\right] \tag{6.27}$$

If we assume that the weight vector, $\mathbf{W}_k$, remains near its optimum, $\mathbf{W}^*$, we conclude from (2.39) that ε_k^2 is approximately uncorrelated with the signal vector, so that (6.27) becomes

$$\begin{aligned}\operatorname{cov}[\mathbf{N}_k] &\approx 4E\left[\varepsilon_k^2\right] E\left[\mathbf{X}_k \mathbf{X}_k^{\mathrm{T}}\right] \\ &\approx 4\xi_{\min}\mathbf{R}\end{aligned} \tag{6.28}$$

To apply the results of Chapter 5, we need to transform (6.28) into the principal-axis coordinate system, as follows [see (3.38), (5.40), etc., for similar transformations]:

$$\begin{aligned}\operatorname{cov}[\mathbf{N}_k'] &= \operatorname{cov}[\mathbf{Q}^{-1}\mathbf{N}_k] \\ &= E\left[\mathbf{Q}^{-1}\mathbf{N}_k(\mathbf{Q}^{-1}\mathbf{N}_k)^{\mathrm{T}}\right] \\ &= \mathbf{Q}^{-1}E\left[\mathbf{N}_k\mathbf{N}_k^{\mathrm{T}}\right]\mathbf{Q} \\ &= \mathbf{Q}^{-1}\operatorname{cov}[\mathbf{N}_k]\mathbf{Q} \approx 4\xi_{\min}\Lambda\end{aligned} \tag{6.29}$$

Now we can apply (5.52), which gives the weight vector covariance in the principal-axis coordinate system. The result is

$$\begin{aligned}\operatorname{cov}[\mathbf{V}_k'] &= \frac{\mu}{4}(\Lambda - \mu\Lambda^2)^{-1}\operatorname{cov}[\mathbf{N}_k'] \\ &\approx \mu\xi_{\min}(\Lambda - \mu\Lambda^2)^{-1}\Lambda\end{aligned} \tag{6.30}$$

In practical situations the elements of $\mu\Lambda$ tend to be considerably less than 1, so we simplify the expression in (6.30) by neglecting the term $\mu\Lambda^2$ to obtain

$$\begin{aligned}\operatorname{cov}[\mathbf{V}_k'] &\approx \mu\xi_{\min}\Lambda^{-1}\Lambda \\ &\approx \mu\xi_{\min}\mathbf{I}\end{aligned} \tag{6.31}$$

Thus, transforming back to unprimed coordinates, we have the steady-state noise in the weight vector solution given approximately by

$$\begin{aligned}\operatorname{cov}[\mathbf{V}_k] &= \mathbf{Q}\operatorname{cov}[\mathbf{V}_k']\mathbf{Q}^{-1} \\ &\approx \mu\xi_{\min}\mathbf{Q}\mathbf{I}\mathbf{Q}^{-1} \\ &\approx \mu\xi_{\min}\mathbf{I}\end{aligned} \tag{6.32}$$

A further development of $\operatorname{cov}[\mathbf{V}_k]$ under less restrictive (nonstationary) conditions than those imposed to obtain (6.32) may be found in Farden [6]. In the example of Figure 6.3 the values of μ were 0.1 and 0.05 and $\xi_{\min}$ was about 0.4. Therefore, the diagonal elements of $\operatorname{cov}[\mathbf{V}_k]$ are 0.04 and 0.02 for the upper and lower tracks, respectively. These values correspond to rms weight fluctuations of 0.20 and 0.14, respectively.

MISADJUSTMENT

We recall from (5.96) that the misadjustment in an adaptive process is defined as the ratio of the excess mean-square error to the minimum mean-square error, and is thus a measure of how closely the adaptive process tracks the true Wiener solution, that is, a measure of the "cost of adaptability." The excess mean-square error is illustrated in Figure 5.3, and is given in (5.60) as

$$\text{excess MSE} = E\left[\mathbf{V}_k'^{\mathrm{T}}\Lambda\mathbf{V}_k'\right] \tag{6.33}$$

If we use the $L + 1$ elements of $\mathbf{V}_k'$ and note that Λ is a diagonal matrix, we can express (6.33) in terms of a sum:

$$\text{excess MSE} = \sum_{n=0}^{L} \lambda_n E\left[v_{nk}'^2\right] \tag{6.34}$$

If we assume that the adaptive transient has died out and therefore that the squared error is near the "bottom of the bowl," we may assume that $E[v_{nk}'^2]$ in (6.34) is an element of $\operatorname{cov}[\mathbf{V}_k']$ in (6.31). Using (6.31) in (6.34) then gives the following approximation:

$$\begin{aligned}\text{excess MSE} &\approx \mu\xi_{\min}\sum_{n=0}^{L}\lambda_n \\ &\approx \mu\xi_{\min}\operatorname{tr}[\mathbf{R}]\end{aligned} \tag{6.35}$$

From this result we may compute the misadjustment, defined above and in (5.96), as

$$\boxed{\begin{aligned} M &= \frac{\text{excess MSE}}{\xi_{\min}} \\ &\approx \mu\,\mathrm{tr}[\mathbf{R}] \end{aligned}} \tag{6.36}$$

For example, in Figure 6.3, the values of μ were 0.1 and 0.05 and the trace of $\mathbf{R}$ was found in (6.16) to be 1.02, so we have for Figure 6.3,

$$M \approx \begin{cases} 0.1 & \text{(upper track)} \\ 0.05 & \text{(lower track)} \end{cases} \tag{6.37}$$

We can seen in (6.36) that the misadjustment, M, is directly proportional to the adaptive gain constant, μ. Thus there is a trade-off between the misadjustment and the rate of adaptation. To see this trade-off more clearly, let us recall from (6.20) that the time constant for the nth mode of the learning curve is

$$(\tau_{\text{mse}})_n = \frac{1}{4\mu\lambda_n} \tag{6.38}$$

From this the trace of $\mathbf{R}$ can be written

$$\begin{aligned} \mathrm{tr}[\mathbf{R}] &= \sum_{n=0}^{L} \lambda_n \\ &= \frac{1}{4\mu} \sum_{n=0}^{L} \frac{1}{(\tau_{\text{mse}})_n} = \frac{L+1}{4\mu}\left(\frac{1}{\tau_{\text{mse}}}\right)_{\text{av}} \end{aligned} \tag{6.39}$$

Using this result in (6.36), we have

$$M \approx \frac{L+1}{4}\left(\frac{1}{\tau_{\text{mse}}}\right)_{\text{av}} \tag{6.40}$$

In the special case where all eigenvalues are equal, (6.40) reduces to

$$M \approx \frac{L+1}{4\tau_{\text{mse}}} \tag{6.41}$$

Experience shows this expression to be a good approximation to the relationship between misadjustment, time constant of the learning curve, and number of weights even when the eigenvalues are not equal. Such a relationship is needed in designing an adaptive system when the eigenvalues are unknown.

Since the trace of the R-matrix is the total power of the inputs to the weights, which is generally known, one can use (6.36) in choosing a value of μ that will produce a desired value of M. One can accordingly combine (6.41) and (6.36) to obtain a general expression for the time constant of the learning curve with equal eigenvalues:

$$\tau_{\text{mse}} \approx \frac{L+1}{4\mu\,\mathrm{tr}[\mathbf{R}]} \tag{6.42}$$

This expression is also a good approximation in many cases when the eigenvalues of the R-matrix are unequal.

Since adaptive transients die out or settle in approximately four time constants, (6.41) can be used to derive the following rule of thumb:

> Misadjustment equals number of weights divided by settling time, when the eigenvalues are equal.

Operation with 10 percent misadjustment, satisfactory in many applications, can generally be achieved with an adaptive settling time equal to 10 times the memory time span of the adaptive transversal filter.

PERFORMANCE

We have noted previously that the LMS algorithm differs from the algorithms discussed in Chapters 4 and 5 primarily in the way in which the gradient, ∇_k, is estimated at each time step. In effect, the LMS algorithm takes advantage of additional a priori information, namely that the performance surface is quadratic. We might expect that the LMS algorithm would have a performance advantage over the previous algorithms that used the difference method to estimate ∇_k, and in fact it does.

TABLE 6.1 MISADJUSTMENT AND TIME CONSTANTS OF STEEPEST-DESCENT AND LMS ALGORITHMS

	Steepest-descent algorithm	LMS algorithm
Misadjustment, M	$\frac{\mu(L+1)}{4N\delta^2}\xi_{\min} = \frac{(L+1)^2}{8P}\left(\frac{1}{T_{\text{mse}}}\right)_{\text{av}}$	$\mu \operatorname{tr} \mathbf{R} = \frac{L+1}{4}\left(\frac{1}{T_{\text{mse}}}\right)_{\text{av}}$
Perturbation, P	$\frac{\delta^2\lambda_{\text{av}}}{\xi_{\min}}$	—
Total misadjustment, M_{tot}	$M + P$	M
Time constant of nth mode:		
In number of adaptive iterations, τ_{mse}	$\frac{1}{4\mu\lambda_n}$	$\frac{1}{4\mu\lambda_n}$
In number of data samples, T_{mse}	$\frac{N(L+1)}{2\mu\lambda_n}$	$\frac{1}{4\mu\lambda_n}$

The performance advantage of the LMS algorithm over the steepest-descent algorithm of Chapters 4 and 5 can readily be understood by comparing the expressions for misadjustment and time constant given in Table 6.1. The table entries originate from (5.16), (5.86) through (5.88), (5.99), (5.106), (6.20), and (6.40). It is apparent in both cases that misadjustment is reduced by slow adaptation; that is, by making the value of the time constant large. For a given time constant, however, the misadjustment of the LMS algorithm increases linearly with the number of weights rather than with the square of the number of weights. In typical circumstances much faster adaptation is thus possible.

The relative performance of the two algorithms is indicated in another way in Figure 6.6, which shows the time constant of adaptation T_{mse} as a function of the number of weights. For this comparison the misadjustment of the LMS algorithm was fixed at 10 percent. The *total* misadjustment of the steepest-descent algorithm was also fixed at 10 percent, with the perturbation P optimized according to (5.109). The eigenvalues of the R-matrix were further assumed to be equal. We obtain the curves in Figure 6.6 from Table 6.1 as follows:

$$\begin{aligned} &\textit{Steepest descent}: \quad T_{\text{mse}} = \frac{(L+1)^2}{8MP} = 50(L+1)^2 \\ &\textit{LMS}: \quad T_{\text{mse}} = \frac{L+1}{4M} = 2.5(L+1) \end{aligned} \tag{6.43}$$

The figure illustrates that the LMS algorithm has a much shorter adaptation time, particularly when the number of weights is large.

In Chapter 5 we derived the time constant of adaptation of the method of steepest descent for an adaptive filter with ten weights and a total misadjustment of 10 percent with P optimized. The result obtained is given in (5.110) and is equal to 5000 data samples. A similar calculation for the LMS algorithm yields a time constant of adaptation of only 25 data samples. This value is much more favorable even than that of 625 data samples obtained in (5.111) for off-line adaptation to eliminate the effects of perturbing the weight vector. Since adaptive transients generally die out within approximately four time constants, the settling time would be approximately 100 sampling periods or iterations.

The efficiency of the LMS algorithm has been shown to approach a theoretical limit for adaptive algorithms when the eigenvalues of the R-matrix are equal or nearly equal [5]. With disparate eigenvalues, however, misadjustment is primarily determined by the fastest modes of adaptation, while settling time is limited by the slowest modes. Algorithms similar to the LMS algorithm but based on Newton's method rather than the method of steepest descent have been devised in an attempt to sustain efficiency under these conditions. In these algorithms the gradient estimate is premultiplied at each iteration by an estimate of the inverse of the R-matrix:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu\widehat{\mathbf{R}^{-1}}\hat{\nabla}_k \tag{6.44}$$

or

$$\mathbf{W}_{k+1} = \mathbf{W}_k + 2\mu\widehat{\mathbf{R}^{-1}}\varepsilon_k\mathbf{X}_k \tag{6.45}$$

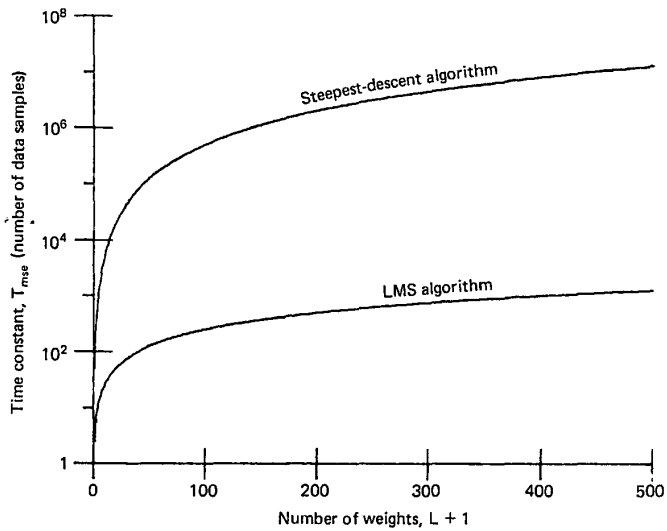


Figure 6.6 Time constant of adaptation versus number of weights for steepest-descent and LMS algorithms. Total misadjustment fixed at 10 percent.

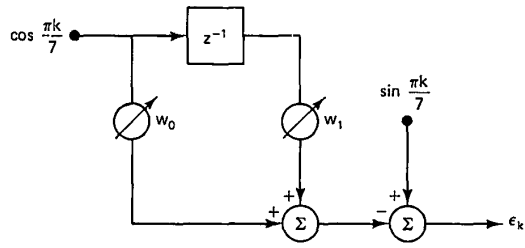
This process causes all adaptive modes to have essentially the same time constant. Algorithms based on this principle are discussed in Chapter 8. They are potentially more efficient than the LMS algorithm but also typically more difficult to implement.

Attempts have also been made to devise algorithms more efficient than the LMS algorithm by using a time-varying value for the constant of adaptation μ. High values are first chosen to achieve rapid convergence; after convergence has been achieved low values are chosen to minimize misadjustment. This method works well as long as the input statistics remain stationary.

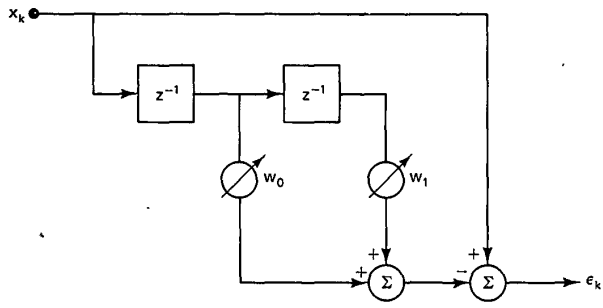
EXERCISES

1. Express the LMS algorithm for an individual weight in a single-input adaptive linear combiner.
2. If an adaptive linear combiner has an input signal power equal to p, what is a reasonable range for the adaptive gain constant, μ?
3. Describe the principal axes of the performance surface in Figure 6.3.
4. Explain why T_{mse} and τ_{mse} are identical in the case of the LMS algorithm.
5. In the example in Figure 6.3, what is the approximate covariance matrix for the gradient noise, assuming that the adaptive transient has died out?

6. What are the eigenvalues in the example in Figure 6.3?
7. What is the effect of adding a third weight to the adaptive linear combiner in Figure 6.2:
   (a) On the minimum mean-square error?
   (b) On the misadjustment?
   (c) On the learning curve?
8. Consider the adaptive signal-canceling situation in the diagram below.



   (a) Write an expression for the performance surface.
   (b) Determine a range for the adaptive gain constant.
   (c) Express the LMS algorithm for this situation.
   (d) Using $\mu = 0.05$ and 0.005 with zero initial conditions, run the LMS algorithm and plot two learning curves similar to Figure 6.5 on the same graph. Estimate the two time constants.
9. Consider the adaptive predictor shown below.



   (a) Given $\phi_{xx}(n) = E[x_k x_{k+n}]$, write an expression for the performance surface.
   (b) Give a specific expression for the performance surface when x_k is equal to $\sin(k\pi/5)$.
   (c) Write the LMS algorithm for the case where $x_k = \sin(k\pi/5)$, with μ at one-fifth of its maximum in (6.10).
   (d) Using the LMS algorithm in part (c), list the first 20 error values, ε_0 through ε_{19}. Assume that $\mathbf{W}_0 = \mathbf{0}$.

10. For the predictor in Exercise 9, generate 5000 samples of the input as follows, using zero initial conditions:

$$x_k = 0.95(r_k - 0.5) + 0.05x_{k-1}; \qquad k = 0, 1, \ldots, 4999$$

where r_k is the kth number in the random sequence in Appendix A. Then:

(a) Using the data, estimate the input correlation matrix, **R**.

(b) Using (6.21), select μ to produce an average learning time constant equal to 100 data samples.

(c) Run the LMS algorithm with μ as above. Plot the first 5000 weight values, w_{0k} and w_{1k}, versus k.

(d) Plot a learning curve similar to that shown in Figure 6.5.

No answers

REFERENCES AND ADDITIONAL READINGS

1. B. Widrow and J. M. McCool, "A comparison of adaptive algorithms based on the methods of steepest descent and random search," *IEEE Trans. Antennas Propag.*, vol. AP-24, no. 5, pp. 615–637, Sept. 1976.
2. B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson, Jr., "Stationary and nonstationary learning characteristics of the LMS adaptive filter," *Proc. IEEE*, vol. 64, no. 8, pp. 1151–1162, Aug. 1976.
3. B. Widrow et al., "Adaptive noise cancelling: principles and applications," *Proc. IEEE*, vol. 63, no. 12, pp. 1692–1716, Dec. 1975.
4. B. Widrow, "Adaptive Filters," in *Aspects of Network and System Theory*, R. E. Kalman and N. De Claris (Eds.). New York: Holt, Rinehart and Winston, 1970, pp. 563–587.
5. B. Widrow and E. Walach, "On the statistical efficiency of the LMS algorithm with nonstationary inputs," *IEEE Trans. Information Theory—Special Issue on Adaptive Filtering*, vol. 30, no. 2, part 1, pp. 211–221, Mar. 1984.
6. D. C. Farden, "Tracking properties of adaptive signal processing algorithms," *IEEE Trans. Acoust. Speech Signal Process.*, vol. ASSP-29, p. 439, June 1981.

This material may be protected by Copyright law (Title 17 U.S. Code)

Contents

PREFACE xiii

LIST OF SYMBOLS xv

part I GENERAL INTRODUCTION 1

Objectives of Part I 1

1 ADAPTIVE SYSTEMS 3

Definition and Characteristics 3
Areas of Application 4
General Properties 5
Open- and Closed-Loop Adaptation 6
Applications of Closed-Loop Adaptation 9
Example of an Adaptive System 11
The Chapters Ahead 13

2 THE ADAPTIVE LINEAR COMBINER 15

General Description 15
Input Signal and Weight Vectors 16
Desired Response and Error 18
The Performance Function 19
Gradient and Minimum Mean-Square Error 21
Example of a Performance Surface 22

Alternative Expression of the Gradient 24
Decorrelation of Error and Input Components 26
Exercises 26

part II THEORY OF ADAPTATION WITH STATIONARY SIGNALS 31

Objectives of Part II 31

3 PROPERTIES OF THE QUADRATIC PERFORMANCE SURFACE 33

Normal Form of the Input Correlation Matrix 34
Eigenvalues and Eigenvectors of the Input Correlation Matrix 34
An Example with Two Weights 36
Geometrical Significance of Eigenvectors and Eigenvalues 38
A Second Example 41
Exercises 43

4 SEARCHING THE PERFORMANCE SURFACE 46

Methods of Searching the Performance Surface 46
Basic Ideas of Gradient Search Methods 47
A Simple Gradient Search Algorithm and Its Solution 48
Stability and Rate of Convergence 49
The Learning Curve 51
Gradient Search by Newton's Method 52
Newton's Method in Multidimensional Space 54
Gradient Search by the Method of Steepest Descent 56
Comparison of Learning Curves 61
Exercises 63

5 GRADIENT ESTIMATION AND ITS EFFECTS ON ADAPTATION 66

Gradient Component Estimation by Derivative Measurement 66

The Performance Penalty 68
Derivative Measurement and Performance Penalties with Multiple Weights 69
Variance of the Gradient Estimate 71
Effects on the Weight-Vector Solution 75
Excess Mean-Square Error and Time Constants 80
Misadjustment 87
Comparative Performance of Newton's and Steepest-Descent Methods 89
Total Misadjustment and Other Practical Considerations 91
Exercises 93

part III ADAPTIVE ALGORITHMS AND STRUCTURES 98

Objectives of Part III 98

6 THE LMS ALGORITHM 99

Derivation of the LMS Algorithm 99
Convergence of the Weight Vector 101
An Example of Convergence 103
Learning Curve 107
Noise in the Weight-Vector Solution 109
Misadjustment 110
Performance 112
Exercises 114

7 THE *z*-TRANSFORM IN ADAPTIVE SIGNAL PROCESSING 117

The *z*-Transform 117
Right- and Left-Handed Sequences 119
Transfer Functions 120
Frequency Response 122
Impulse Response and Stability 124
The Inverse *z*-Transform 126
Correlation Functions and Power Spectra 128
The Performance Function 131
Examples of Performance Surfaces 134
Exercises 137

8 OTHER ADAPTIVE ALGORITHMS AND STRUCTURES 141

An Ideal: The LMS/Newton Algorithm 142
Properties of the LMS/Newton Algorithm 145
The Sequential Regression Algorithm 147
Adaptive Recursive Filters 154
Random-Search Algorithms 161
Lattice Structures 164
The Adaptive Lattice Predictor 173
Adaptive Filters with Orthogonal Signals 182
Exercises 186

part IV APPLICATIONS 193

Objectives of Part IV 193

9 ADAPTIVE MODELING AND SYSTEM IDENTIFICATION 195

General Description 195
Adaptive Modeling of a Multipath Communication Channel 200
Adaptive Modeling in Geophysical Exploration 209
Adaptive Modeling in FIR Digital Filter Synthesis 212
Exercises 225

10 INVERSE ADAPTIVE MODELING, DECONVOLUTION, AND EQUALIZATION 231

General Description of Inverse Modeling 232
Some Theoretical Examples 236
Adaptive Equalization of Telephone Channels 244
Adapting Poles and Zeros for IIR Digital Filter Synthesis 250
Exercises 264

11 ADAPTIVE CONTROL SYSTEMS 270

Adaptive Model Control 271
Adaptive Inverse Control 280

Examples of Adaptive Inverse Control 285
Plant Noise and the Filtered-X LMS Algorithm 288
Inverse Control Using the Filtered-X LMS Algorithm 292
Model Reference Control 294
Exercises 298

12 ADAPTIVE INTERFERENCE CANCELING 302

Early Work in Adaptive Interference Canceling 303
The Concept of Adaptive Noise Canceling 303
Stationary Noise-Canceling Solutions 306
Effects of Signal Components in the Reference Input 311
The Adaptive Interference Canceler as a Notch Filter 316
The Adaptive Interference Canceler as a High-Pass Filter 323
Effects of Finite Length and Causality 324
Multiple-Reference Noise Canceling 327
Canceling 60-Hz Interference in Electrocardiography 329
Canceling Donor-Heart Interference in Heart-Transplant Electrocardiography 330
Canceling the Maternal ECG in Fetal Electrocardiography 334
Canceling Noise in Speech Signals 337
Canceling Echoes in Long-Distance Telephone Circuits 339
Canceling Antenna Sidelobe Interference 347
Canceling Periodic Interference with an Adaptive Predictor 349
The Adaptive Self-Tuning Filter 351
The Adaptive Line Enhancer 354
Conclusion 361
Exercises 361

13 INTRODUCTION TO ADAPTIVE ARRAYS AND ADAPTIVE BEAMFORMING 368

Sidelobe Cancellation 369
Beamforming with a Pilot Signal 383
Spatial Configurations 388
Adaptive Algorithms 391
Narrowband Experiments 394
Broadband Experiments 399
Exercises 404

14 ANALYSIS OF ADAPTIVE BEAMFORMERS 409

Performance Characteristics of Receiving Arrays 409
The Griffiths LMS Beamformer 412
The Frost Adaptive Beamformer 415
An Adaptive Beamformer with Poles and Zeros 420
Signal Cancellation and Distortion 429
Frequency-Hop Spread-Spectrum Techniques 442
Beamformers with Superresolution 445
Exercises 456

APPENDIX A A Portable Random Number Generator 459

INDEX 469