Farhang-Boroujeny
Adaptive Filters

WILEY

Adaptive Filters

Theory and Applications

B. Farhang-Boroujeny

Adaptive Filters
Theory and Applications

B. Farhang-Boroujeny
National University of Singapore

John Wiley & Sons
Chichester • New York • Weinheim • Brisbane • Singapore • Toronto

National 01243 779777
International (+44) 1243 779777

e-mail (for orders and customer service enquiries): cs-books@wiley.co.uk

Visit our Home Page on http://www.wiley.co.uk
or
http://www.wiley.com

Other Wiley Editorial Offices

John Wiley & Sons, Inc., 605 Third Avenue,
New York, NY 10158-0012, USA

Wiley-VCH Verlag GmbH, Pappelallee 3,
D-69469 Weinheim, Germany

Jacaranda Wiley Ltd, 33 Park Road, Milton,
Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop #02-01,
Jin Xing Distripark, Singapore 0512

John Wiley & Sons (Canada) Ltd, 22 Worcester Road,
Rexdale, Ontario M9W 1L1, Canada

Library of Congress Cataloging-in-Publication Data

Farhang-Boroujeny, B.
Adaptive filters : theory and applications / B. Farhang-Boroujeny.
p. cm.
Includes bibliographical references and index.
ISBN 0-471-98337-3
1. Adaptive filters. I. Title.
TK7872.F5F37 1999
621.3815'324—dc21 98-8783
CIP

British Library Cataloguing in Publication Data

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

ISBN 0-471-98337-3

Typeset in part from the author's disks in 10/12pt Times by the Alden Group, Oxford.
Printed and bound in Great Britain by Antony Rowe Ltd, Chippenham

To my family for their support,
understanding and love

Contents

Preface

This book has grown out of the author's research work and teaching experience in the field of adaptive signal processing. It is primarily designed as a text for a first-year graduate level course in adaptive filters. It is also intended to serve as a technical reference for practising engineers.

The book is based on the author's class notes used for teaching a graduate level course at the Department of Electrical Engineering, National University of Singapore. These notes have also been used to conduct short courses for practising engineers from industry.

A typical one-semester course would cover Chapters 1, 3–6, and 12, and the first half of Chapter 11, in depth. Chapter 2, which contains a short review of the basic concepts of the discrete-time signals and systems, may be left as self-study material for students. Selected parts of the rest of the book may also be taught in the same semester, or may be used with supplemental readings for a second semester course on advanced topics and applications.

In the study of adaptive filters, computer simulations constitute an important supplemental component to theoretical analyses and deductions. Often, theoretical developments and analyses involve a number of approximations and/or assumptions. Hence, computer simulations become necessary to confirm the theoretical results. Apart from this, computer simulation turns out to be a necessity in the study of adaptive filters for gaining an in-depth understanding of the behaviour and properties of the various adaptive algorithms. MATLAB, from MathWorks Inc., appears to be the most commonly used software simulation package. Throughout the book we use MATLAB to present a number of simulation results to clarify and/or confirm the theoretical developments. A diskette containing the programs used for generating these results is supplied along with the book so that the reader can run these programs and acquire a more in-depth insight into the concepts of adaptive filtering.

Another integral part of this text is exercise problems at the end of chapters. With the exception of the first few chapters, two kinds of exercise problems are provided in each chapter:

1. *The usual problem exercises.* These problems are designed to sharpen the reader's skill in theoretical development. They are designed to extend results developed in the text, to develop some results that are referred to in the text, and to illustrate applications to practical problems. Solutions to these problems are available to instructors through the publisher (ISBN 0-471-98788-5).

2. *Simulation-oriented problems.* These involve computer simulations and are designed to enhance the reader's understanding of the behaviour of the different adaptive algorithms that are introduced in the text. Most of these problems are based on the MATLAB programs that are provided on the diskette accompanying the book. In addition, there are also other (open-ended) simulation-oriented problems designed to help the reader develop his/her own programs and prepare him/her to experiment with practical problems.

This book assumes that the reader has some basic background of discrete-time signals and systems (including an introduction to linear system theory and random signal analysis), complex variable theory and matrix algebra. However, brief reviews of these topics are provided in Chapters 2 and 4.

The book starts with a general overview of adaptive filters in Chapter 1. Many examples of applications such as system modelling, channel equalization, echo cancellation and antenna arrays are reviewed in this chapter. This is followed by a brief review of discrete-time signals and systems, in Chapter 2, which puts the related concepts in a framework appropriate for the rest of the book.

In Chapter 3 we introduce a class of optimum linear systems collectively known as Wiener filters. Wiener filters are fundamental to the implementation of adaptive filters. We note that the cost function used to formulate the Wiener filters is an elegant choice leading to a mathematically tractable problem. We also discuss the unconstrained Wiener filters with respect to causality and duration of the filter impulse response. This study reveals many interesting aspects of Wiener filters and establishes a good foundation for the study of adaptive filters for the rest of the book. In particular, we find that, in the limit, when the filter length tends to infinity, a Wiener filter treats different frequency components of underlying processes separately. Numerical examples reveal that when the filter length is limited, separation of frequency components may be replaced by separation of frequency bands, within a good approximation. This treatment of adaptive filters that is pursued throughout the book turns out to be an enlightening engineering approach to the study of adaptive filters.

Eigenanalysis is an essential mathematical tool for the study of adaptive filters. A thorough treatment of this topic is covered in the first half of Chapter 4. The second half of this chapter gives an analysis of the performance surface of transversal Wiener filters. This is followed by search methods, which are introduced in Chapter 5. The search methods discussed in this chapter are idealized versions of the statistical search methods that are used in practice for the actual implementation of adaptive filters. They are idealized in the sense that the statistics of the underlying processes are assumed to be known a priori.

The celebrated least-mean-square (LMS) algorithm is introduced in Chapter 6 and studied extensively in Chapters 7–11. The LMS algorithm, which was first proposed by Widrow and Hoff in 1960, is the most widely used adaptive filtering algorithm in practice, owing to its simplicity and robustness to signal statistics.

Chapters 12 and 13 are devoted to the method of least squares. This discussion, although brief, gives the basic concept of the method of least squares and highlights its advantages and disadvantages compared with the LMS-based algorithms. In Chapter 13 the reader is introduced to the fast versions of least-squares algorithms. Overall, these two chapters lay a good foundation for the reader to continue his/her study of this subject with reference to more advanced books and/or papers.

The problem of tracking is discussed in the final chapter of the book. In the context of a system modelling problem, we present a generalized formulation of the LMS algorithm which covers most of the algorithms that are discussed in the various chapters of the book, thus bringing a common platform for the comparison of the different algorithms. We also discuss how the step-size parameter(s) of the LMS algorithm and the forgetting factor of the RLS algorithm may be optimized to achieve good tracking behaviour.

The following notations are adopted in this book. We use non-bold lowercase letters for scalar quantities, bold lowercase for vectors, and bold uppercase for matrices. Non-bold uppercase letters are used for functions of variables, such as $H(z)$, and lengths/dimensions of vectors/matrices. The lowercase letter n is used for the time index. In the case of block processing algorithms, such as those discussed in Chapters 8 and 9, we reserve the lowercase letter k as the block index. The time and block indices are put in brackets, while subscripts are used to refer to elements of vectors and matrices. For example, the ith element of the time-varying tap-weight vector $\mathbf{w}(n)$ is denoted as $w_i(n)$. The superscripts T and H denote vector or matrix transposition and Hermitian transposition, respectively. We keep all vectors in column form. More specific notations are explained in the text as and when found necessary.

B. Farhang-Boroujeny

Acknowledgements

I am deeply indebted to Dr George Mathew of Data Storage Institute, National University of Singapore, for critically reviewing the entire manuscript of this book. Dr Mathew checked through every single line of the manuscript and made numerous invaluable suggestions and improved the book in many ways. I am truly grateful to him for his invaluable help.

I am also grateful to Professor V. U. Reddy, Indian Institute of Science, Bangalore, India, for reviewing Chapters 2–7, and Dr M. Chakraborty, Indian Institute of Technology, Kharagpur, India, for reviewing Chapters 3, 4, and 13 and making many valuable suggestions.

I am indebted to my graduate students, both in Iran and Singapore, for helping me in the development of many results that are presented in this book. In particular, I am grateful to Dr S. Gazor and Mr Y. Lee for helping me to develop some of the results on transform domain adaptive filters that are presented in Chapter 7. I am also grateful to Mr Z. Wang for his enthusiasm towards the development of subband adaptive filters in the form presented in Chapter 9.

I wish to thank my students H. B. Chionh, K. K. Ng (Adrian) and T. P. Ng for their great help and patience in checking the accuracy of all the references in the bibliography.

I also wish to thank my colleagues in the Department of Electrical Engineering, National University of Singapore, for their support and encouragement in the course of the development of this book.

1
Introduction

As we begin our study of 'adaptive filters', it may be worth trying to understand the meaning of the terms 'adaptive' and 'filters' in a very general sense. The adjective 'adaptive' can be understood by considering a system which is trying to adjust itself so as to respond to some phenomenon that is taking place in its surroundings. In other words, the system tries to adjust its parameters with the aim of meeting some well-defined goal or target which depends upon the state of the system as well as its surrounding. This is what 'adaptation' means. Moreover, there is a need to have a set of steps or certain procedure by which this process of 'adaptation' is carried out. And finally, the 'system' that carries out and undergoes the process of 'adaptation' is called by the more technical, yet general enough, name 'filter' – a term that is very familiar to and a favourite of any engineer. Clearly, depending upon the time required to meet the final target of the adaptation process, which we call convergence time, and the complexity/resources that are available to carry out the adaptation, we can have a variety of adaptation algorithms and filter structures. From this point of view, we may summarize the contents/contribution of this book as 'the study of some selected adaptive algorithms and their implementations along with the associated filter structures, from the points of view of their convergence and complexity performance'.

1.1 Linear Filters

The term '*filter*' is commonly used to refer to any device or system that takes a mixture of particles/elements from its input and process them according to some specific rules to generate a corresponding set of particles/elements at its output. In the context of signals and systems, *particles/elements* are the *frequency components* of the underlying signals and, traditionally, filters are used to retain all the frequency components that belong to a particular band of frequencies, while rejecting the rest of them, as much as possible. In a more general sense, the term filter may be used to refer to a system that *reshapes* the frequency components of the input to generate an output signal with some desirable features, and this is how we view the concept of filtering throughout the chapters which follow.

Filters (or systems, in general) may be either *linear* or *non-linear*. In this book, we consider only linear filters and our emphasis will also be on *discrete-time* signals and systems. Thus, all the signals will be represented by sequences, such as $x(n)$. The most

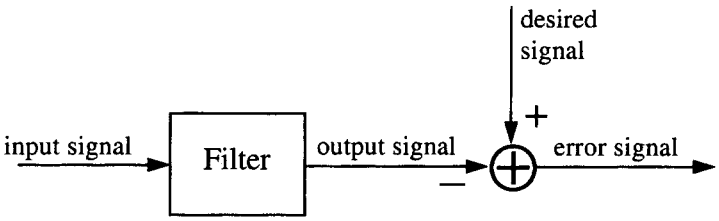


Figure 1.1 Schematic diagram of a filter emphasizing its role in reshaping the input signal to match the desired signal

basic feature of linear systems is that their behaviour is governed by *the principle of superposition*. This means that if the responses of a linear discrete-time system to input sequences $x_1(n)$ and $x_2(n)$ are $y_1(n)$ and $y_2(n)$, respectively, then the response of the same system to the input sequence $x(n) = ax_1(n) + bx_2(n)$, where a and b are arbitrary constants, will be $y(n) = ay_1(n) + by_2(n)$. This property leads to many interesting results in 'linear system theory'. In particular, a linear system is completely characterized by its impulse response or the Fourier transform of its impulse response, known as the *transfer function*. The transfer function of a system at any frequency is equal to its gain at that frequency. In other words, in the context of our discussion above, we may say that the transfer function of a system determines how the various frequency components of its input are reshaped by the system.

Figure 1.1 depicts a general schematic diagram of a filter emphasizing the purpose for which it is used in different problems addressed/discussed in this book. In particular, the filter is used to reshape certain *input signals* in such a way that its output is a good estimate of the given desired signal. The process of selecting the filter parameters (coefficients) so as to achieve the best match between the desired signal and the filter output is often done by optimizing an appropriately defined *performance function*. The performance function can be defined in a *statistical* or *deterministic* framework. In the statistical approach, the most commonly used performance function is the *mean-square value* of the *error signal*, i.e. the difference between the desired signal and the filter output. For stationary input and desired signals, minimizing the mean-square error results in the well-known *Wiener filter*, which is said to be *optimum in the mean-square sense*. The subject of Wiener filters will be covered extensively in Chapter 3. Most of the adaptive algorithms that are studied in this book are practical solutions to Wiener filters. In the deterministic approach, the usual choice of performance function is *a weighted sum of the squared error signal*. Minimizing this function results in a filter which is optimum for the given set of data. However, under some assumptions on certain statistical properties of the data, the deterministic solution will approach the statistical solution, i.e. the Wiener filter, for large data lengths. Chapters 12 and 13 deal with the deterministic approach in detail. We refer the reader to Section 1.4 of this chapter for a brief overview of the adaptive formulations under the stochastic (i.e. statistical) and deterministic frameworks.

1.2 Adaptive Filters

As mentioned in the previous section, the filter required for estimating the given desired signal can be designed using either the stochastic or deterministic formulations. In the

deterministic formulation, the filter design requires the computation of certain average quantities using the given set of data that the filter should process. On the other hand, the design of Wiener filter (i.e. in the stochastic approach) requires a priori knowledge of the statistics of the underlying signals. Strictly speaking, a large number of realizations of the underlying signal sequences are required for reliably estimating these statistics. This procedure is not feasible in practice since we usually have only one realization for each of the signal sequences. To resolve this problem, it is assumed that the underlying signal sequences are *ergodic*, which means that they are stationary and their statistical and time averages are identical. Thus, by using time averages, Wiener filters can be designed, even though there is only one realization for each of the signal sequences.

Although direct measurement of the signal averages to obtain the necessary information for the design of Wiener or other optimum filters is possible, in most of the applications the signal averages (statistics) are used in an indirect manner. All the algorithms covered in this book take the output error of the filter, correlate that with the samples of filter input in some way, and use the result in a recursive equation to adjust the filter coefficients iteratively. The reasons for solving the problem of adaptive filtering in an iterative manner are:

1. Direct computation of the necessary averages and their application for computing the filter coefficients requires the accumulation of a large amount of signal samples. Iterative solutions, on the other hand, do not require accumulation of signal samples, thereby *resulting in a significant amount of saving in memory.*
2. The accumulation of signal samples and their post processing to generate the filter output, as required in non-iterative solutions, introduces a large delay in the filter output. This is unacceptable in many applications. *Iterative solutions,* on the contrary, *do not introduce any significant delay in the filter output.*
3. The use of iterations results in adaptive solutions with some *tracking* capability. That is, if the signal statistics are changing with time, then the solution provided by an iterative adjustment of the filter coefficients will be able to adapt to the new statistics.
4. Iterative solutions, in general, are much *simpler* to code in software or to implement in hardware than their non-iterative counterparts.

1.3 Adaptive Filter Structures

The most commonly used structure in the implementation of adaptive filters is the *transversal structure,* depicted in Figure 1.2. Here, the adaptive filter has a single input, $x(n)$, and an output, $y(n)$. The sequence $d(n)$ is the desired signal. The output, $y(n)$, is generated as a linear combination of the delayed samples of the input sequence, $x(n)$, according to the equation

$$y(n) = \sum_{i=0}^{N-1} w_i(n)x(n-i), \tag{1.1}$$

where the $w_i(n)$s are the filter *tap weights* (coefficients) and N is the filter length. We refer to the input samples, $x(n-i)$, for $i = 0, 1, \ldots, N-1$, as the filter *tap inputs.* The tap weights, the $w_i(n)$s, which may vary in time, are controlled by the adaptation algorithm.

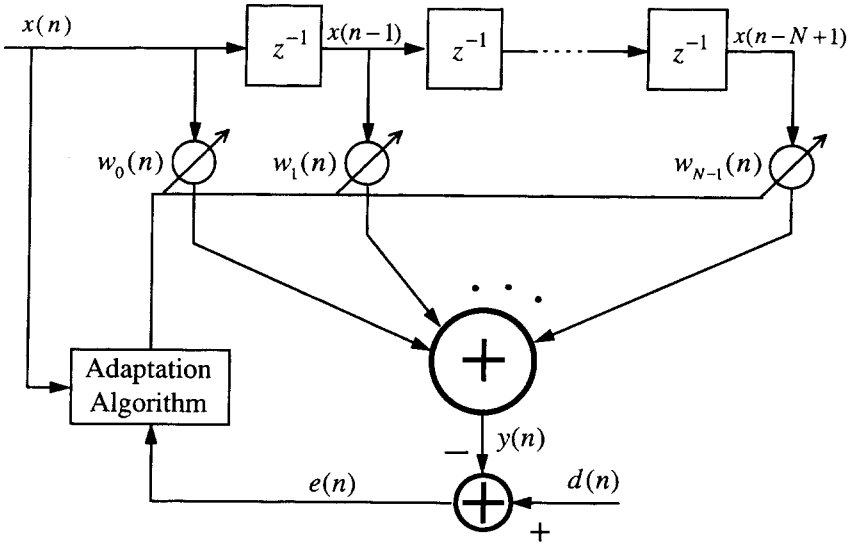


Figure 1.2 Adaptive transversal filter

In some applications, such as beamforming (see Section 1.6.4), the filter tap inputs are not the delayed samples of a single input. In such cases the structure of the adaptive filter assumes the form shown in Figure 1.3. This is called *a linear combiner*, since its output is a linear combination of the different signals received at its tap inputs:

$$y(n) = \sum_{i=0}^{N-1} w_i(n)x_i(n). \tag{1.2}$$

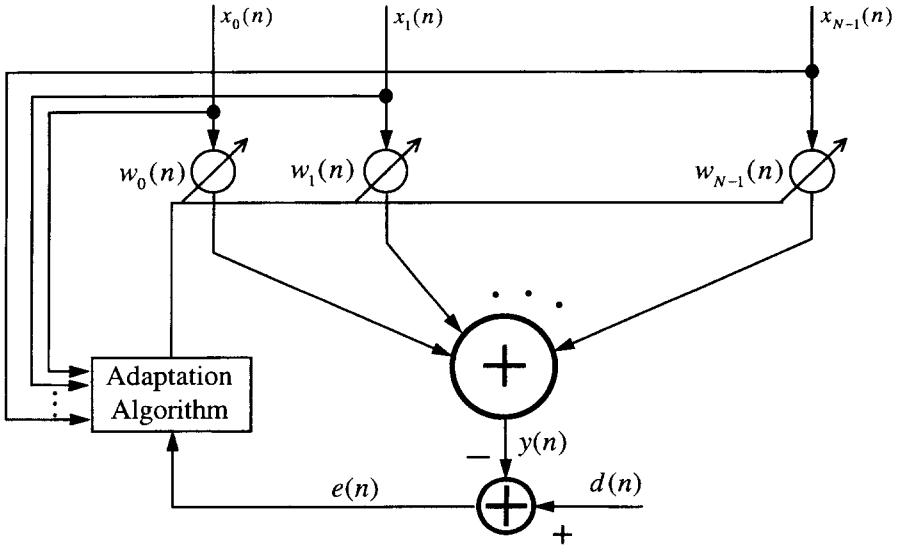


Figure 1.3 Adaptive linear combiner

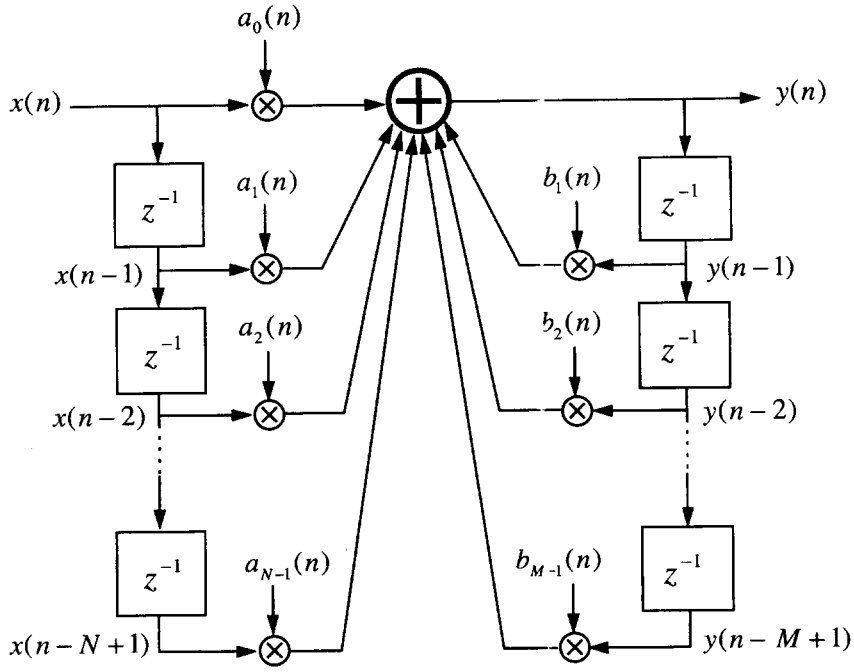


Figure 1.4 The structure of an IIR filter

Note that the linear combiner structure is more general than the transversal. The latter, as a special case of the former, can be obtained by choosing $x_i(n) = x(n-i)$.

The structures of Figures 1.2 and 1.3 are those of the non-recursive filters, i.e. computation of filter output does not involve any feedback mechanism. We also refer to Figure 1.2 as a finite-impulse response (FIR) filter, since its impulse response is of finite duration in time. An infinite-impulse response (IIR) filter is governed by recursive equations such as (see Figure 1.4)

$$y(n) = \sum_{i=0}^{N-1} a_i(n)x(n-i) + \sum_{i=1}^{M-1} b_i(n)y(n-i), \tag{1.3}$$

where $a_i(n)$ and $b_i(n)$ are the forward and feedback tap weights, respectively. IIR filters have been used in many applications. However, as we shall see in the later chapters, because of the many difficulties involved in the adaptation of IIR filters, their application in the area of adaptive filters is rather limited. In particular, they can easily become unstable since their poles may get shifted out of the unit circle (i.e. $|z| = 1$, in the z-plane (see next chapter)) by the adaptation process. Moreover, the performance function (e.g. mean-square error as a function of filter coefficients) of an IIR filter usually has many local minima points. This may result in convergence of the filter to one of the local minima and not to the desired global minimum point of the performance function. On the contrary, the mean-square error functions of the FIR filter and linear combiner are well behaved quadratic functions with a single minimum point which can easily be found

through various adaptive algorithms. Because of these points, the non-recursive filters are the sole candidates in most of the applications of adaptive filters. Hence, most of the discussion in the subsequent chapters is limited to the non-recursive filters. The IIR adaptive filters, with two specific examples of their applications, are discussed in Chapter 10.

The FIR and IIR structures shown in Figures 1.2 and 1.4 are obtained by direct realization of the respective difference equations (1.1) and (1.3). These filters may alternatively be implemented using the *lattice structures*. The lattice structures, in general, are more complicated than the direct implementations. However, in certain applications they have some advantages which make them better candidates than the direct forms. For instance, in the application of linear prediction for speech processing where we need to realize all pole (IIR) filters, the lattice structure can be more easily controlled to prevent possible instability of the filter. The derivation of lattice structures for both FIR and IIR filters is presented in Chapter 11. Also, in the implementation of the least-squares method (see Section 1.4.2), the use of lattice structures leads to a computationally efficient algorithm known as recursive least-squares lattice. A derivation of this algorithm is presented in Chapter 13.

The FIR and IIR filters that were discussed above are classified as linear filters since their outputs are obtained as linear combinations of the present and past samples of input and, in the case of the IIR filter, the past samples of the output also. Although most applications are restricted to the use of linear filters, non-linear adaptive filters become necessary in some applications where the underlying physical phenomena to be modelled are far from being linear. A typical example is magnetic recording where the recording channel becomes non-linear at high densities as a result of the interaction between the magnetization transitions written on the medium. The Volterra series representation of systems is usually used in such applications. The output, $y(n)$, of a Volterra system is related to its input, $x(n)$, according to the equation

$$\begin{aligned} y(n) = w_{0,0}(n) &+ \sum_{i} w_{1,i}(n)x(n-i) \\ &+ \sum_{i,j} w_{2,i,j}(n)x(n-i)x(n-j) \\ &+ \sum_{i,j,k} w_{3,i,j,k}(n)x(n-i)x(n-j)x(n-k) + \cdots, \end{aligned} \tag{1.4}$$

where $w_{0,0}(n)$, the $w_{1,i}(n)$s, the $w_{2,i,j}(n)$s, the $w_{1,i,j,k}(n)$s, ... are filter coefficients. In this book, we do not discuss the Volterra filters any further. However, we note that all the summations in (1.4) may be put together and the Volterra filter may be thought of as a linear combiner whose inputs are determined by the delayed samples of $x(n)$ and their cross-multiplications. Noting this, we find that the extension of most of the adaptive filtering algorithms to the Volterra filters is straightforward.

1.4 Adaptation Approaches

As introduced in Sections 1.1 and 1.2, there are two distinct approaches that have been widely used in the development of various adaptive algorithms; namely, stochastic and

deterministic. Both approaches have many variations in their implementations leading to a rich variety of algorithms, each of which offers desirable features of its own. In this section we present a review of these two approaches and highlight the main features of the related algorithms.

1.4.1 Approach based on Wiener filter theory

According to the Wiener filter theory, which comes from the stochastic framework, the optimum coefficients of a linear filter are obtained by minimization of its mean-square error (MSE). As already noted, strictly speaking, the minimization of MSE requires certain statistics obtained through ensemble averaging, which may not be possible in practical applications. The problem is resolved using ergodicity so as to use time averages instead of ensemble averages. Furthermore, to come up with simple recursive algorithms, very rough estimates of the required statistics are used. In fact, the celebrated *least-mean-square* (LMS) algorithm, which is the most basic and widely used algorithm in various adaptive filtering applications, uses the *instantaneous* value of the square of the error signal as an estimate of the MSE. It turns out that this very rough estimate of the MSE, when used with a *small* step-size parameter in searching for the optimum coefficients of the Wiener filter, leads to a very simple and yet reliable adaptive algorithm.

The main disadvantage of the LMS algorithm is that its convergence behaviour is highly dependent on the power spectral density of the filter input. When the filter input is white, i.e. its power spectrum is flat across the whole range of frequencies, the LMS algorithm converges very fast. However, when certain frequency bands are not well excited (i.e. the signal energy in those bands is relatively low), some slow modes of convergence appear, resulting in very slow convergence compared with the case of white input. In other words, to converge fast, the LMS algorithm requires equal excitation over the whole range of frequencies. Noting this, over the years researchers have developed many algorithms which effectively divide the frequency band of the input signal into a number of subbands and achieve some degree of signal whitening by using some power normalization mechanism, prior to applying the adaptive algorithm. These algorithms, which appear in different forms are presented in Chapters 7, 9 and 11.

In some applications, we need to use adaptive filters whose length exceeds a few hundreds or even a few thousands of taps. Clearly, such filters are computationally expensive to implement. An effective way of implementing such filters at a much lower computational complexity is to use the fast Fourier transform (FFT) algorithm to implement time domain convolutions in the frequency domain, as is commonly done in the implementation of long digital filters (Oppenheim and Schafer, 1975, 1989). Adaptive algorithms that use FFT for reducing computational complexity are presented in Chapter 8.

1.4.2 Method of least squares

The adaptive filtering algorithms whose derivations are based on the Wiener filter theory have their origin in a statistical formulation of the problem. In contrast to this, the *method of least squares* approaches the problem of filter optimization from a deterministic point of view. As already mentioned, in the Wiener filter theory the desired filter is obtained by minimizing the mean-square error (MSE), i.e. a statistical quantity. In the method of least

squares, on the other hand, the performance index is the *sum of weighted error squares* for the given data, i.e. a deterministic quantity. A consequence of this deterministic approach (which will become clear as we go through its derivation in Chapter 12) is that the least-squares-based algorithms, in general, converge much faster than the LMS-based algorithms. They are also insensitive to the power spectral density of the input signal. The price that is paid for achieving this improved convergence performance is higher computational complexity and poorer numerical stability.

Direct formulation of the least-squares problem results in a matrix formulation of its solution which can be applied on a block-by-block basis to the incoming signals. This, which is referred to as the *block estimation of the least-squares method*, has some useful applications in areas such as linear predictive coding of speech signals. However, in the context of adaptive filters, recursive formulations of the least-squares method that update the filter coefficients after the arrival of every sample of input are preferred, for reasons that were given in Section 1.2. There are three major classes of recursive least-squares (RLS) adaptive filtering algorithms:

- The standard RLS algorithm
- The QR-decomposition-based RLS (QRD-RLS) algorithm
- Fast RLS algorithms

The standard RLS algorithm

The derivation of this algorithm involves the use of a well known result from linear algebra known as the *matrix inversion lemma*. Consequently, the implementation of the standard RLS algorithm involves matrix manipulations that result in a computational complexity proportional to the square of the filter length.

The QR-decomposition-based RLS (QRD-RLS) algorithm

This formulation of RLS algorithm also involves matrix manipulations which lead to a computational complexity that grows with the square of the filter length. However, the operations involved here are such that they can be put into some regular structures known as *systolic arrays*. Another important feature of the QRD-RLS algorithm is its robustness to numerical errors as compared with other types of RLS algorithms (Haykin, 1991, 1996).

Fast RLS algorithms

In the case of transversal filters, the tap inputs are successive samples of the input signal, $x(n)$ (see Figure 1.2). The fast RLS algorithms use this property of the filter input and solve the problem of least squares with a computational complexity which is proportional to the length of the filter, thus the name *fast RLS*. Two types of fast RLS algorithms may be recognized:

1. *RLS lattice algorithms:* These lattice algorithms involve the use of order-update as well as the time-update equations. A consequence of this feature is that it results in modular structures which are suitable for hardware implementations using the

pipelining technique. Another desirable feature of these algorithms is that certain variants of them are very robust against numerical errors arising from the use of finite word lengths in computations.

2. *Fast transversal RLS algorithm:* In terms of number of operations per iteration, the fast transversal RLS algorithm is less complex than the lattice RLS algorithms. However, it suffers from numerical instability problems which require careful attention to prevent undesirable behaviour in practice.

In this book we present a complete treatment of the various LMS-based algorithms, in seven chapters. However, our discussion of RLS algorithms is rather limited. We present a comprehensive treatment of the properties of the method of least squares and a derivation of the standard RLS algorithm in Chapter 12. The basic results related to the development of fast RLS algorithms and some examples of such algorithms are presented in Chapter 13. A study of the *tracking behaviour* of selected adaptive filtering algorithms is presented in the final chapter of the book.

1.5 Real and Complex Forms of Adaptive Filters

There are some practical applications in which the filter input and its desired signal are complex-valued. A good example of this situation appears in digital data transmission, where the most widely used signalling techniques are phase shift keying (PSK) and quadrature amplitude modulation (QAM). In this application, the baseband signal

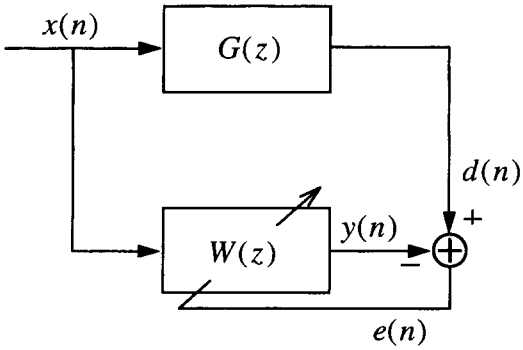


Figure 1.5 Adaptive system modelling

1.6.1 Modelling

Figure 1.5 depicts the problem of modelling in the context of adaptive filters. The aim is to estimate the parameters of the model, $W(z)$, of a plant, $G(z)$. On the basis of some a priori knowledge of the plant, $G(z)$, a transfer function, $W(z)$, with certain number of adjustable parameters is selected first. The parameters of $W(z)$ are then chosen through an adaptive filtering algorithm such that the difference between the plant output, $d(n)$, and the adaptive filter output, $y(n)$, is minimized.

An application of modelling, which may be readily thought of, is *system identification*. In most modern control systems the plant under control is identified on-line and the result is used in a *self-tuning regulator* (STR) loop, as depicted in Figure 1.6 (see Åström and Wittenmark, 1989, for example).

Another application of modelling is *echo cancellation*. In this application an adaptive filter is used to identify the impulse response of the path between the source from which the echo originates and the point where the echo appears. The output of the adaptive filter, which is an estimate of the echo signal, can then be used to cancel the undesirable echo. The subject of echo cancellation is discussed further below in Section 1.6.4.

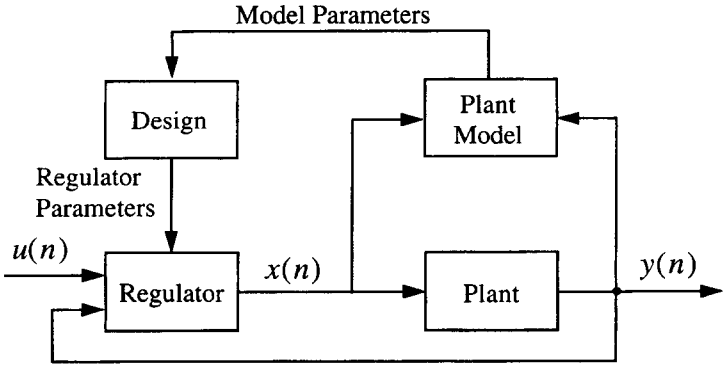


Figure 1.6 Block diagram of a self-tuning regulator

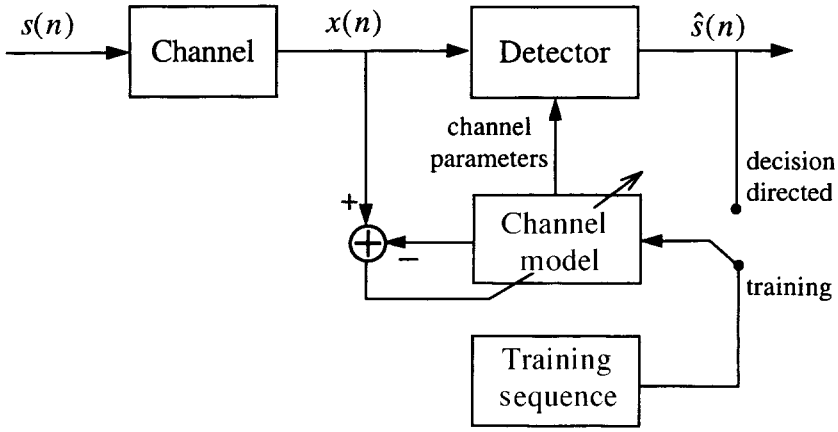


Figure 1.7 An adaptive data receiver using channel identification

Non-ideal characteristics of communication channels often result in some distortion in the received signals. To mitigate such distortion, channel equalizers are usually used. This technique, which is equivalent to implementing the inverse of the channel response, is discussed below in Section 1.6.2. Direct modelling of the channel, however, has also been found useful in some implementations of data receivers. For instance, data receivers equipped with maximum likelihood detectors require an estimate of the channel response (Proakis, 1995). Furthermore, computation of equalizer coefficients from channel response has been proposed by some researchers since this technique has been found to result in better tracking of time-varying channels (Fechtel and Meyr, 1991, and Farhang-Boroujeny, 1996c). In such applications, a *training pattern* is transmitted in the beginning of every connection. The received signal, which acts as the desired signal to an adaptive filter, is used in a set-up to identify the channel, as shown in Figure 1.7. Once the channel is identified and the normal mode of transmission begins, the detected data symbols, $\hat{s}(n)$, are used as input to the channel model and the adaptation process continues for tracking possible variations of the channel. This is known as the *decision directed mode* and is also shown in Figure 1.7.

1.6.2 Inverse modelling

Inverse modelling, also known as *deconvolution*, is another application of adaptive filters which has found extensive use in various engineering disciplines. The most widely used application of inverse modelling is in communications where an inverse model (also

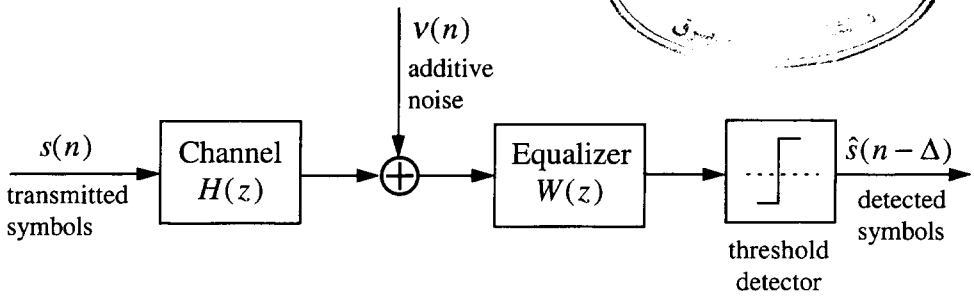


Figure 1.8 A baseband data transmission system with channel equalizer

Channel equalization

Figure 1.8 depicts the block diagram of a baseband transmission system equipped with a channel equalizer. Here, the channel represents the combined response of the transmitter filter, the actual channel, and the receiver front-end filter. The additive noise sequence, $v(n)$, arises from thermal noise in the electronic circuits and possible cross-talks from neighbouring channels. The transmitted data symbols, $s(n)$, that appear in the form of amplitude/phase modulated pulses, are distorted by the channel. The most significant among the different distortions is the *pulse-spreading effect,* which results because the channel impulse response is not equal to an ideal impulse function, but rather a response that is non-zero over many symbol periods. This distortion results in interference of the neighbouring data symbols with one another, thereby making the detection process through a simple threshold detector unreliable. The phenomenon of interference between neighbouring data symbols is known as *intersymbol interference* (ISI). The presence of the additive noise samples, $v(n)$, further deteriorates the performance of data receivers. The role of the equalizer, as a filter, is to resolve the distortion introduced by the channel (i.e. rejection or minimization of ISI) while minimizing the effect of additive noise at the threshold detector input (equalizer output) as much as possible. If the additive noise could be ignored, then the task of equalizer would be rather straightforward. For a channel $H(z)$, an equalizer with transfer function $W(z) = 1/H(z)$ could do the job perfectly, as this results in an overall channel-equalizer transfer function $H(z)W(z) = 1$, which implies that the transmitted data sequence, $s(n)$, will appear at the detector input without any distortion. Unfortunately, this is an ideal situation which cannot be used in most of the practical applications.

We note that the inverse of the channel transfer function, i.e. $1/H(z)$, may be non-causal if $H(z)$ happens to have a zero outside the unit circle, thus making it unrealizable in practice. This problem is solved by selecting the equalizer so that $H(z)W(z) \approx z^{-\Delta}$, where Δ is an appropriate integer delay. This is equivalent to saying that a delayed replica of the transmitted symbols appears at the equalizer output. Example 3.4 of Chapter 3 clarifies the concept of non-causality of $1/H(z)$ and also the way the problem is (approximately) solved by introducing a delay, Δ.

We also note that the choice of $W(z) = 1/H(z)$ (or $W(z) \approx z^{-\Delta}/H(z)$) may lead to a significant enhancement of the additive noise, $v(n)$, in those frequency bands where the magnitude of $H(z)$ is small (i.e. $1/H(z)$ is large). Hence, in choosing an equalizer, $W(z)$, we should keep a balance between residual ISI and noise enhancement at the

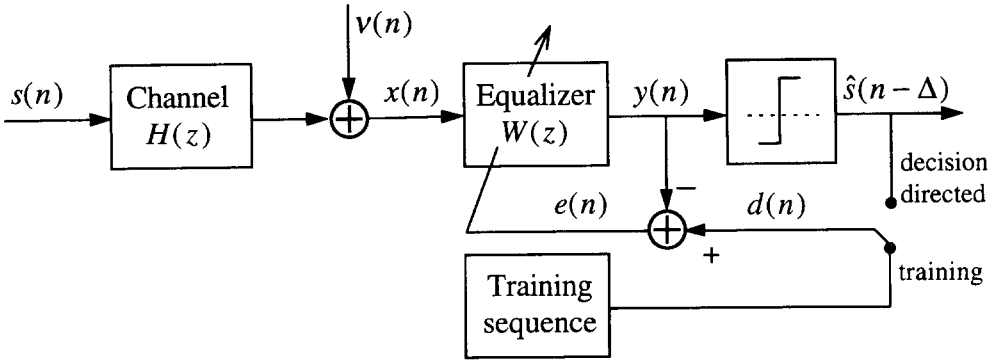


Figure 1.9 Details of a baseband data transmission system equipped with an adaptive channel equalizer

equalizer output. A Wiener filter is a solution with such a balance (see Chapter 3, Section 3.6.4).

Figure 1.9 presents the details of a baseband transmission system, equipped with an adaptive equalizer. The equalizer is usually implemented in the form of a transversal filter. Initial training of the equalizer requires knowledge of the transmitted data symbols (or, to be more accurate, a delayed replica of them) since they should be used as the desired signal samples for adaptation of the equalizer tap weights. This follows from the fact that the equalizer output should ideally be the same as the transmitted data symbols. We thus require an initialization period during which the transmitter sends a sequence of training symbols that are known to the receiver. This is called the *training mode*. Training symbols are usually specified as part of the standards and the manufacturers of data modems should comply with these so that the modems of different manufacturers can communicate with one another. (The term modem, which is an abbreviation for 'modulator and demodulator', is commonly used to refer to data transceivers (transmitter and receiver).)

At the end of the training mode the tap weights of the equalizer would have converged close to their optimal values. The detected symbols would then be similar to the transmitted symbols with probability close to one. Hence, from then onwards, the detected symbols can be treated as the desired signal for further adaptation of the equalizer so that possible variations in the channel can be tracked. This mode of operation of the equalizer is called the *decision directed mode*. The decision directed mode successfully works as long as the channel variation is slow enough so that the adaptation algorithm is able to follow the channel variations satisfactorily. This is necessary for the purpose of ensuring low symbol error rates in detection so that these symbols can still be used as the desired signal.

The inverse modelling discussed above defines the equalizer as an approximation of $z^{-\Delta}/H(z)$, i.e. the target/desired response of the cascade of channel and equalizer is $z^{-\Delta}$, a pure delay. This can be generalized by replacing the target response $z^{-\Delta}$ by a general target response, say $\Gamma(z)$. In fact, to achieve higher efficiency in the usage of the available bandwidth, some special choices of $\Gamma(z) \neq z^{-\Delta}$ are usually considered in communication systems. Systems that incorporate such non-trivial target responses are referred to as *partial-response signalling systems*. The detector in such systems is no more the simple threshold detector, but one which can exploit the information that the overall channel is

now $\Gamma(z)$, instead of the trivial memoryless channel $z^{-\Delta}$. The Viterbi detector (Proakis, 1995) is an example of such a detector. The target response, $\Gamma(z)$, is selected so that its magnitude response approximately matches the channel response, i.e. $|\Gamma(e^{j\omega})| \approx |H(e^{j\omega})|$, over the range of frequencies of interest. The impact of this choice is that the equalizer, which is now $W(z) \approx \Gamma(z)/H(z)$, has a magnitude response that is approximately equal to one, thereby minimizing the noise enhancement. To clarify this further and also to mention another application of inverse modelling, we next discuss the problem of magnetic recording.

Magnetic recording

The process of writing data bits on a magnetic medium (tape or disk) and reading them back later is similar to sending data bits over a communication channel from one end of a transmission line and receiving them at the other end of the line. The data bits, which are converted to signal pulses prior to recording, undergo some distortion due to the non-perfect behaviour of the head and medium, as happens in communication channels because of the non-ideal response of the channel. Additive thermal noise and interference from neighbouring recording tracks (just like neighbouring channels in communications) are also present in the magnetic recording channels (Bergman, 1996).

Magnetic recording channels are usually characterized by their response to an isolated pulse of width one bit interval, T. This is known as the *dibit response*, and in the case of hard-disk channels it is usually modelled by the superposition of positive and negative *Lorentzian* pulses, separated by one bit interval, T. In other words, the Lorentzian pulse models the step response of the channel. The Lorentzian pulse is defined as

$$g_a(t) = \frac{1}{1 + \left(\frac{2t}{t_{50}}\right)^2}, \tag{1.5}$$

where t_{50} is the pulse width measured at 50% of its maximum amplitude. The subscript 'a' in $g_a(t)$ and other functions that appear in the rest of this subsection is to emphasize that they are analog (non-sampled) signals. The ratio $D = t_{50}/T$ is known as the *recording density*. Typical values of D are in the range 1 to 3. A higher density means that more bits are contained in one t_{50} interval, i.e. more ISI. We may also note that t_{50} is a temporal measure of the recording density. When measured spatially, we obtain another parameter, $pw_{50} = t_{50}/v$, where v is the velocity of the medium with respect to the head. Accordingly, for a given speed, v, the value of D specifies the actual number of bits written on a length pw_{50} along the track on the magnetic medium.

Using (1.5), the dibit response of a hard-disk channel is obtained as

$$h_a(t) = g_a(t) - g_a(t - T). \tag{1.6}$$

The response of the channel to a sequence $s(n)$ of data bits is then given by the convolution sum

$$u_a(t) = \sum_n s(n) h_a(t - nT). \tag{1.7}$$

Thus, the dibit response, $h_a(t)$, is nothing but the impulse response of the recording channel.

Figures 1.10(a) and (b) show the dibit (time domain) and magnitude (frequency domain) responses, respectively, of the magnetic channels (based on the Lorentzian model) for densities $D = 1, 2$ and 3. From Figure 1.10(b) we note that most of the energy in the read-back signals is concentrated in a midband range between zero and an upper-limit around $1/2T$. Clearly, the bandwidth increases with increase in density. In the light of our previous discussions, we may thus choose the target response, $\Gamma(z)$, of the equalizer so that it resembles a bandpass filter whose bandwidth and magnitude response are close to that of the Lorentzian dibit responses. In magnetic recording, the most commonly used partial responses (i.e. target responses) are given by the class IV response

$$\Gamma(z) = z^{-\Delta}(1 + z^{-1})^K(1 - z^{-1}), \tag{1.8}$$

where Δ, as before, is an integer delay and K is an integer greater than or equal to one. As the recording density increases, higher values of K will be required to match the channel characteristics. But, as K increases, the channel length also increases, implying higher complexity in the detector. In Chapter 10, we elaborate on these aspects of partial response systems.

1.6.3 Linear prediction

Prediction is a spectral estimation technique that is used for modelling correlated random processes for the purpose of finding a parametric representation of these processes. In general, different parametric representations could be used to model the processes. In the context of linear prediction, the model used is shown in Figure 1.11. Here, the random process, $\tilde{x}(n)$, is assumed to be generated by exciting the filter $G(z)$ with the input $u(n)$. Since $G(z)$ is an all-pole filter, this is known as *autoregressive* (AR) modelling. The choice/type of the excitation signal, $u(n)$, is application dependent and may vary depending on the nature of the process being modelled. However, it is usually chosen to be a white process.

Other models used for parametric representation are *moving average* (MA) models, where $G(z)$ is an all-zero (transversal) filter, and *autoregressive moving average* (ARMA) models, where $G(z)$ has both poles and zeros. However, the use of AR model is more popular than the other two.

The rationale behind the use of AR modelling may be explained as follows. Since the samples of any given non-white random signal, $x(n)$, are correlated with one another, these correlations could be used to make a prediction of the present sample of the process, $x(n)$, in terms of its past samples, $x(n-1)$, $x(n-2), \ldots, x(n-N)$, as in Figure 1.12. Intuitively, such prediction improves as the predictor length increases. However, the improvement obtained may become negligible once the predictor length, N, exceeds a certain value, which depends upon the extent of the correlation in the given process. The prediction error, $e(n)$, will then be approximately white. We now note that the transfer function between the input process, $x(n)$, and the prediction error, $e(n)$, is

$$H(z) = 1 - \sum_{i=1}^{N} a_i z^{-i}, \tag{1.9}$$

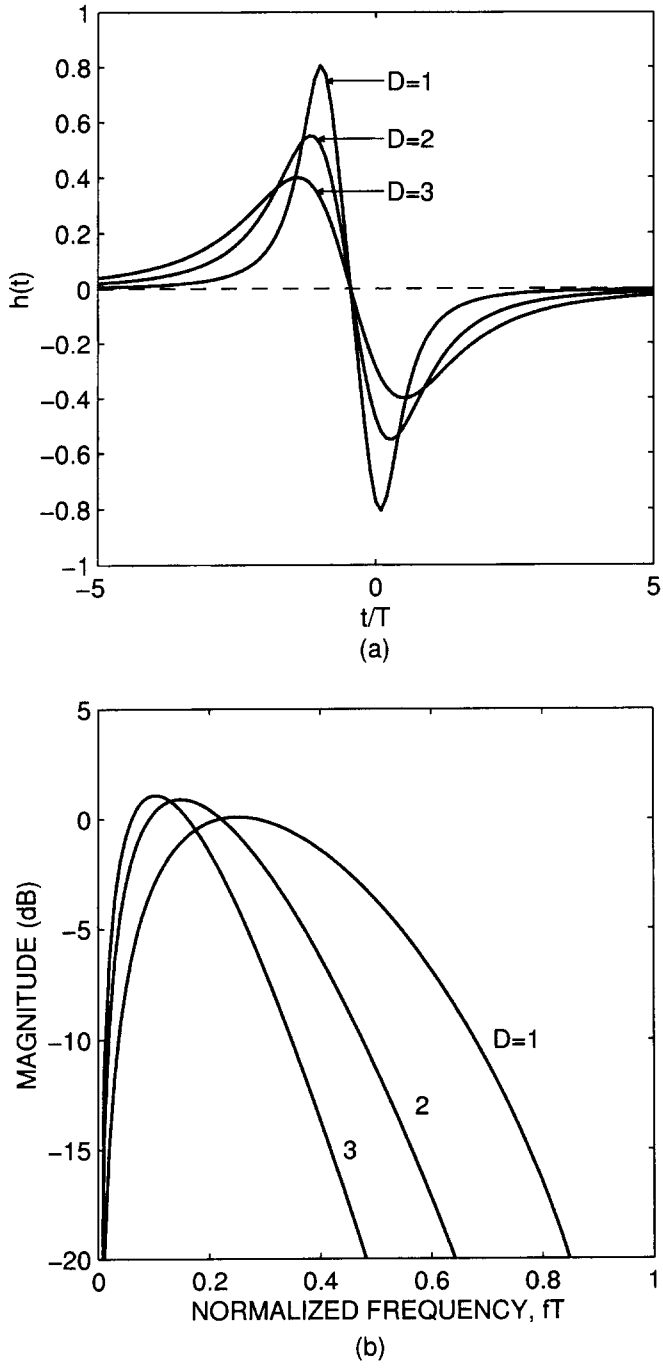


Figure 1.10 Time and frequency domain responses of magnetic recording channels for densities $D = 1$, 2 and 3, modeled using the Lorentzian pulse. (a) Dibit response. (b) Magnitude response of dibit response

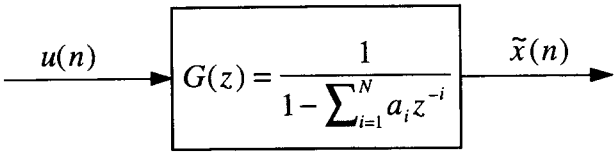


Figure 1.11 Autoregressive modelling of a random process

where the a_is are the predictor coefficients. Now, if a white process, $u(n)$, with similar statistics as $e(n)$ is passed through an all-pole filter with the transfer function,

$$G(z) = \frac{1}{1 - \sum_{i=1}^{N} a_i z^{-i}}, \tag{1.10}$$

as in Figure 1.11, then the generated output, $\tilde{x}(n)$, will clearly be a process with the same statistics as $x(n)$.

With the background developed above, we are now ready to discuss a few applications of adaptive prediction.

Autoregressive spectral analysis

In certain applications we need to estimate the power spectrum of a random process. A trivial way of obtaining such an estimate is to take the Fourier transform (discrete Fourier transform (DFT) in the case of discrete-time processes) and use some averaging (smoothing) technique to improve the estimate. This comes under the class of *non-parametric spectral estimation techniques* (Kay, 1988). When the number of samples of the input are limited, the estimates provided by non-parametric spectral estimation techniques will become unreliable. In such cases the *parametric spectral estimation*, as explained above, may give more reliable estimates.

As mentioned already, parametric spectral estimation could be done by using either AR, MA or ARMA models (Kay, 1988). In the case of AR modelling we proceed as follows. We first choose a proper order, N, for the model. The observed sequence, $x(n)$, is then applied to a predictor structure similar to Figure 1.12 whose coefficients, the a_is, are optimized by minimizing the prediction error, $e(n)$. Once the predictor coefficients have converged, an estimate of the power spectral density of $x(n)$ is obtained according to the following equation:

$$\Phi_{xx}(e^{j\omega}) = N_0 \left| \frac{1}{1 - \sum_{i=1}^{N} a_i e^{-j\omega i}} \right|^2 \tag{1.11}$$

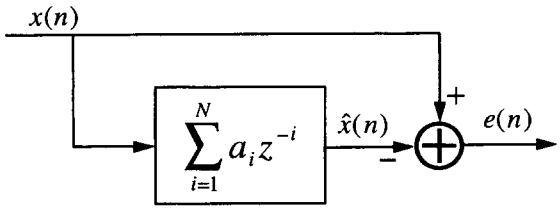


Figure 1.12 Linear predictor

where N_0 is an estimate of the power of the prediction error, $e(n)$. This follows from the model of Figure 1.11 and the fact that after convergence of the predictor, $e(n)$ is approximately white. For further explanation on the derivation of (1.11) from the signal model of Figure 1.11, refer to Chapter 2 (Section 2.4.4).

Adaptive line enhancement

Adaptive line enhancement refers to the situation where a narrow-band signal embedded in a wide-band signal (usually white) needs to be extracted. Depending on the application, the extracted signal may be the signal of interest, or an unwanted interference that should be removed. Examples of the latter case are a spread spectrum signal that has been corrupted by a narrow-band signal and biomedical measurement signals that have been corrupted by the 50/60 Hz power-line interference.

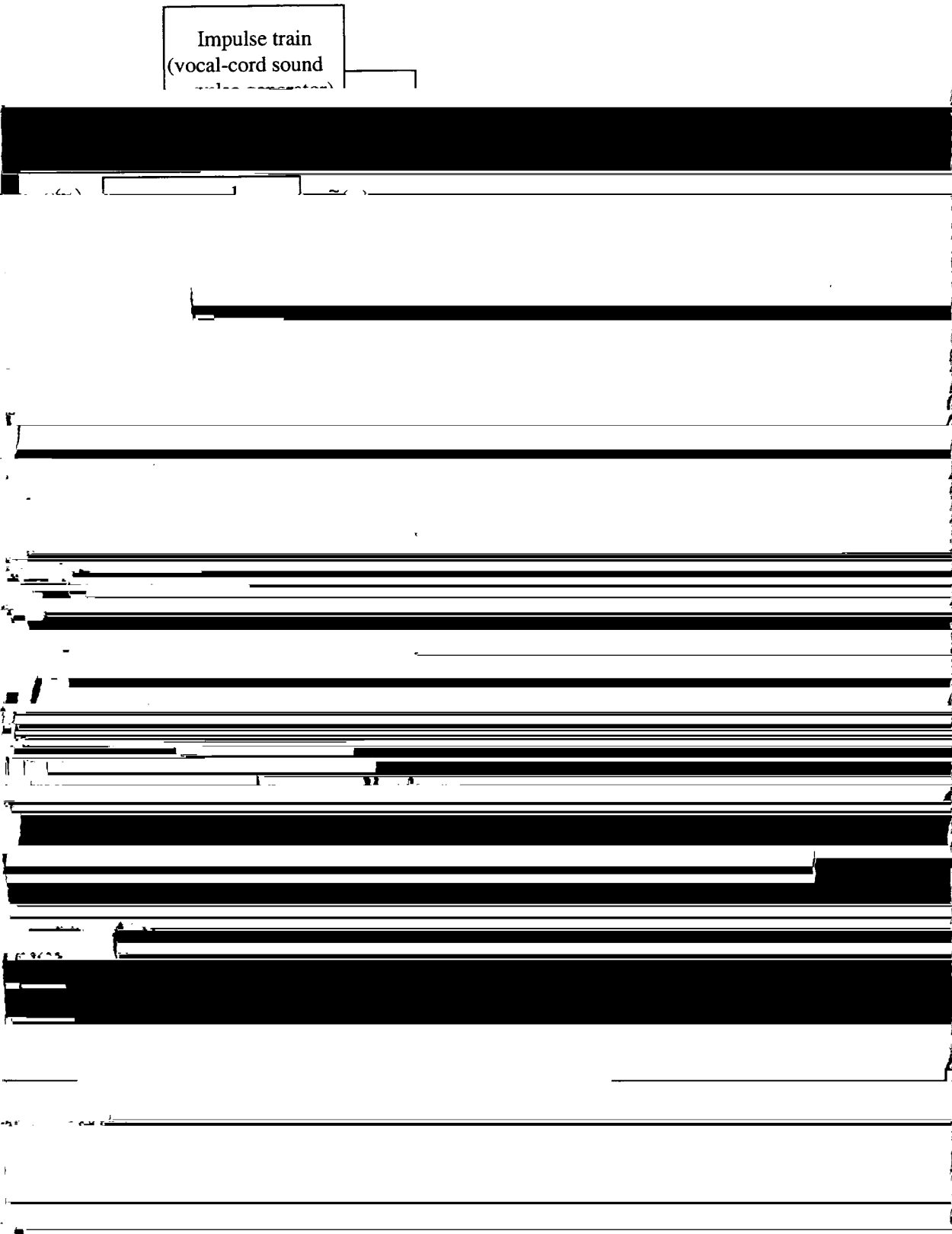
The idea of using prediction to extract a narrow-band signal when mixed with a wide-band signal follows from the following fundamental result of signal analysis: successive samples of a narrow-band signal are highly correlated with one another, whereas there is almost no correlation between successive samples of a wide-band process. Because of this, if a process $x(n)$ consisting of the sum of narrow-band and wide-band processes is applied to a predictor, then the predictor output, $\hat{x}(n)$, will be a good estimate of the narrow-band portion of $x(n)$. In other words, the predictor will act as a narrow-band filter which rejects most of the wide-band portion of $x(n)$ and keeps (enhances) the narrow-band portion, thus the name *line enhancer*. Examples of line enhancers can be found in Chapters 6 and 10. In particular, in Chapter 10 we find that line enhancers can be best implemented using IIR filters.

We also note that in the applications where the narrow-band portion of $x(n)$ has to be rejected (such as the examples mentioned above), the difference between $x(n)$ and $\hat{x}(n)$, i.e. the estimation error, $e(n)$, is taken as the system output. In this case the transfer function between the input, $x(n)$, and the output, $e(n)$, will be that of a notch filter.

Speech coding

Since the advent of digital signal processing, speech processing has always been one of the focused research areas. Among various processing techniques that have been applied to speech signals, linear prediction has been found to be the most promising technique, leading to many useful algorithms. In fact, most of the theory of prediction was developed in the context of speech processing.

There are two major speech coding techniques that involve linear prediction (Jayant and Noll, 1984). Both techniques aim at reducing the number of bits used for every second of speech to achieve saving in storage and/or transmission bandwidth. The first technique, which is categorized under the class of *source coders,* strives to produce digitized voice data at low bit rates in the range 2–10 kb/s. The synthesized speech, however, is not of a high quality. It sounds more synthetic, lacking naturalism. Hence, it becomes difficult to recognize the speaker. The second technique, which comes under the class of *waveform coders,* gives much better quality at the cost of a much higher bit rate (typically, 32 kb/s).


Impulse train
(vocal-cord sound

a prescribed number of bits to generate the information bits associated with the coded speech. Direct quantization of speech samples requires relatively a large number of bits (usually 8 bits per sample) in order to be able to reconstruct the original speech with an acceptable quality.

A modification of the standard PCM, known as *differential pulse code modulation* (DPCM), employs a linear predictor such as Figure 1.12 and uses the bits associated with the quantized samples of the prediction error, $e(n)$, as the coded speech. The rationale here is that the prediction error, $e(n)$, has a much smaller variance than the input, $x(n)$. Thus, for a given quantization level, $e(n)$ may be quantized with fewer bits, as compared with $x(n)$. Moreover, since the number of information bits per every second of the coded speech is directly proportional to the number of bits used per sample, the bit rate of the DPCM will be less compared with the standard PCM.

The prediction filter used in DPCM can be fixed or be made adaptive. A DPCM system with an adaptive predictor is called an *adaptive DPCM* (ADPCM). In the case of speech signals, use of the ADPCM results in superior performance as compared with the case where a non-adaptive DPCM is used. In fact, the ADPCM has been standardized and widely used in practice (International Telecommunication Unit (ITU) Recommendation G.726).

Figure 1.14 depicts a simplified diagram of the ADPCM system, as proposed in ITU Recommendation G.726. Here, the predictor is a six-zero, two-pole adaptive IIR filter. The coefficients of this filter are adjusted adaptively so that the quantized error, $\tilde{e}(n)$, is minimized in the mean-square sense. The predictor input, $\tilde{x}(n)$, is the same as the original input, $x(n)$, except for the quantization error in $\tilde{e}(n)$. To understand the joint operation of the encoder and decoder in Figure 1.14, note that the same signal, $\tilde{e}(n)$, is used as inputs to the predictor structures at the encoder and decoder. Hence, if the stability of the loop consisting of the predictor and adaptation algorithm could be guaranteed, then the steady state value of the reconstructed speech at the decoder, i.e. $\tilde{x}'(n)$, will be equal to that at the encoder, i.e. $\tilde{x}(n)$, since non-equal initial conditions of the encoder and decoder loops will die away after their transient phase.

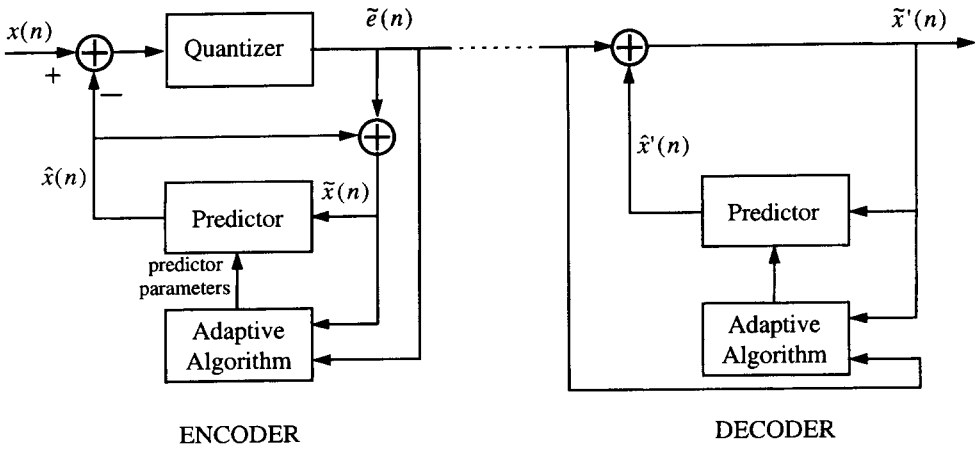


Figure 1.14 ADPCM encoder–decoder

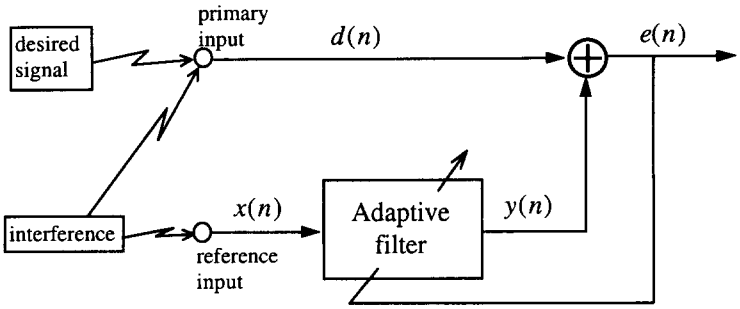


Figure 1.15 Interference cancellation

1.6.4 Interference cancellation

Interference cancellation refers to situations where it is required to cancel an interfering signal/noise from the given signal which is a mixture of the desired signal and the interference. The principle of interference cancellation is to obtain an estimate of the interfering signal and subtract that from the corrupted signal. The feasibility of this idea relies on the availability of a reference source from which the interfering signal originates.

Figure 1.15 depicts the concept of interference cancellation, in its simplest form. There are two inputs to the canceller: *primary* and *reference*. The primary input is the corrupted signal, i.e. the desired signal plus interference. The reference input, on the other hand, originates from the interference source only.[1] The adaptive filter is adjusted so that a replica of the interference signal that is present in the primary signal appears at its output, $y(n)$. Subtracting this from the primary input results in an output that is cleared from interference, thus the name interference cancellation.

We note that the interference cancellation configuration of Figure 1.15 is different from the previous cases of adaptive filters, in the sense that the residual error (which was discarded in other cases) is the cleaned-up signal here. The desired signal in the previous cases has been replaced here by a noisy (corrupted) version of the actual desired signal. Moreover, the use of the term 'reference' to refer to the adaptive filter input is clearly related to the role of this input in the canceller.

In the rest of this section we present some specific applications of interference cancelling.

Echo cancellation in telephone lines

Echoes in telephone lines mostly occur at points where hybrid circuits are used to convert four-wire networks to two-wire networks. Figure 1.16 presents a simplified diagram of a telephone connection network, highlighting the points where echoes occur. The two-wires at the ends are subscriber loops connecting customers' telephones to central offices. It may also include portions of the local network. The four-wires, on the

[1] In some applications of interference cancellation there might also be some leakage of the desired signal to the reference input. Here, we have ignored this situation for simplicity.

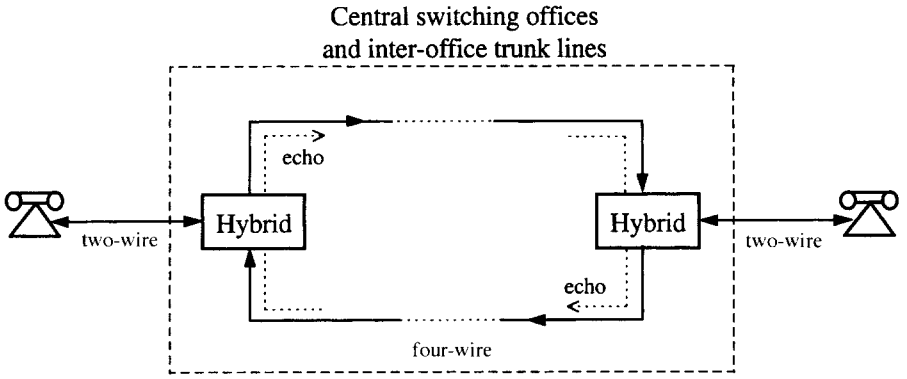


Figure 1.16 Simplified diagram of a telephone network

other hand, are carrier systems (trunk lines) for medium- to long-haul transmission. The distinction is that the two-wire segments carry signals in both directions on the same lines, while in the four-wire segments signals in the two directions are transmitted on two separate lines. Accordingly, the role of the hybrid circuit is to separate the signals in the two directions. Perfect operation of the hybrid circuit requires that the in-coming signal from the trunk lines should be directed to the subscriber line and that there be no leakage (echo) of that to the return line. In practice, however, such ideal behaviour cannot be expected from hybrid circuits. There would always be some echo on the return path. In the case of voice communications (i.e. ordinary conversation on telephone lines), the effect of the echoes becomes more obvious (and annoying to the speaker) in long-distance calls where the delay with which the echo returns to the speaker may be in the range of a few hundred milliseconds. In digital data transmission, both short- and long-delay echoes are serious.

As noted earlier, and also can clearly be seen from Figure 1.17, the problem of echo cancellation may be viewed as one of system modelling. An adaptive filter is put between the in-coming and out-going lines of the hybrid. By adapting the filter to realize an approximation of the echo path, a replica of the echo is obtained at its output. This is then subtracted from the out-going signal to clear that from the undesirable echo.

Echo cancellers are usually implemented in transversal form. The time spread of echoes in a typical hybrid circuit is in the range 20–30 ms. If we assume a sampling rate of 8 kHz for the operation of the echo canceller, then an echo spread of 30 ms requires an

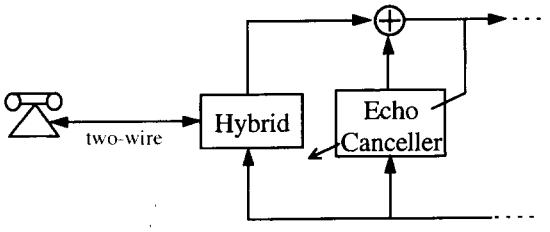


Figure 1.17 Adaptive echo canceller

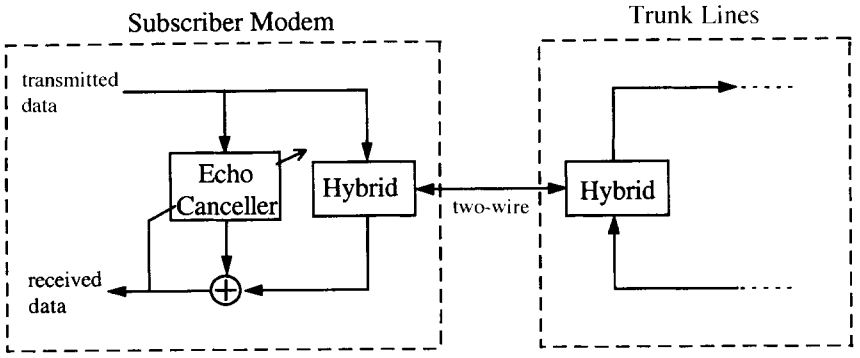


Figure 1.18 Data echo canceller

adaptive filter with at least 240 taps (30 ms × 8 kHz). This is a relatively long filter, requiring a high-speed digital signal processor for its realization. Frequency domain processing is often used to reduce the high computational complexity of long filters. The subject of frequency domain adaptive filters is covered in Chapter 8.

The echo cancellers described above are applicable to both voice and data transmission. However, more stringent conditions need to be satisfied in the case of data transmission. To maximize the usage of the available bandwidth, full-duplex data transmission is often used. This requires the use of a hybrid circuit for connecting the data modem to the two-wire subscriber loop, as shown in Figure 1.18. The leakage of the transmitted data back to the receiver input is thus inevitable and an echo canceller has to be added, as indicated in Figure 1.18. However, we note that the data echo cancellers are different from the voice echo cancellers used in central switching offices in many ways. For instance, since the input to the data echo canceller are data symbols, it can operate at the data symbol rate, which is in the range of 2.4–3 kHz (about three times smaller than the 8 kHz sampling frequency used in voice echo cancellers). For a given echo spread, a lower sampling frequency implies fewer taps for the echo canceller. Clearly, this greatly simplifies the implementation of the echo canceller. On the other hand, the data echo cancellers require a much higher level of echo cancellation to ensure the reliable transmission of data at higher bit rates. In addition, the echoes returned from the other side of the trunk lines should also be taken care of. Detailed discussions on these issues can be found in Lee and Messerschmitt (1994) and Gitlin, Hayes and Weinstein (1992).

Acoustic echo cancellation

The problem of acoustic echo cancellation can be best explained by referring to Figure 1.19 which depicts the scenario that arises in teleconferencing applications. The speech signal from a far-end speaker, received through a communication channel, is broadcast by a loudspeaker in a room and its echo is picked up by a microphone. This echo must be cancelled to prevent its feedback to the far-end speaker. The microphone also picks up the near-end speaker's speech and possible background noise which may exist in the room. An adaptive transversal filter with sufficient length is used to model the

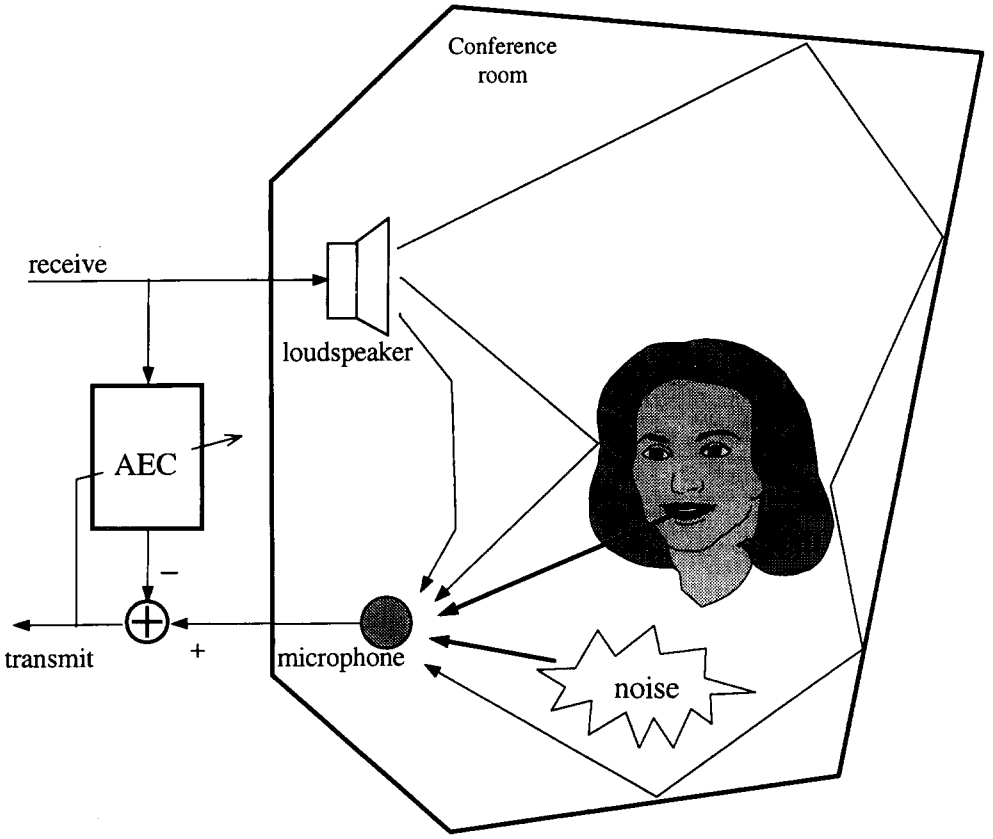


Figure 1.19 Acoustic echo cancellation. Reprinted from Farhang-Boroujeny (1997b)

acoustics of the room. A replica of the loudspeaker echo is then obtained and subtracted from the microphone signal prior to transmission.

Clearly, the problem of acoustic echo cancellation can also be posed as one of system modelling. The main challenge here is that the echo paths spread over a relatively long length in time. For typical office rooms, echoes in the range 100–250 ms spread is quite common. For a sampling rate of 8 kHz, this would mean 800–2000 taps! Thus, the main problem of acoustic echo cancellation is that of realizing very long adaptive filters. In addition, since speech is a low-pass signal, it becomes necessary to use special algorithms to ensure fast adaptation of the echo canceller. The algorithms discussed in Chapters 8 and 9 have been widely used to overcome these difficulties in the implementation of acoustic echo cancellers.

Active noise control

Active noise control (ANC) refers to situations where acoustic *antinoise* waves are generated from electronic circuits (Kuo and Morgan, 1996). The ANC can be best explained by the following example.

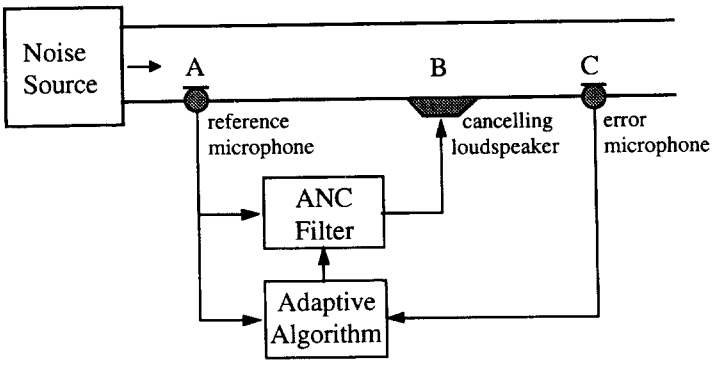


Figure 1.20 Active noise cancellation in a narrow duct

A well-examined application of ANC is cancellation of noise in narrow ducts, such as exhaust pipes and ventilation systems, as illustrated in Figure 1.20. The acoustic noise travelling along the duct is picked up by a microphone at position A. This is used as reference input to an ANC filter whose parameters are adapted so that its output, after conversion to an acoustic wave (through the cancelling loudspeaker), is equal to the negative value of the duct noise at position B, thereby cancelling that. The residual noise, picked up by the error microphone at position C, is the error signal used for adaptation of the ANC filter.

Comparing this ANC set-up with the interference cancellation set-up given in Figure 1.15, we may note the following. The source of interference here is the duct noise, the reference input is the noise picked up by the reference microphone, the desired output (i.e. what we wish to see after cancelling the duct noise) is zero, and the primary input is the duct noise reaching position B. Accordingly, the role of the ANC filter is to model the response of the duct from position A to B.

The above description of ANC assumes that the duct is narrow and the acoustic noise waves are travelling along the duct, which is like a one dimensional model. The acoustic models of wider ducts and large enclosures, such as cars and aircraft, are usually more complicated. Multiple microphones/loudspeakers are needed for successful implementation of ANCs in such enclosures. The adaptive filtering problem is then that of a multiple-input–multiple-output system (Kuo and Morgan, 1996). Nevertheless, the basic principle remains the same, i.e. the generation of antinoise to cancel the actual noise.

Beamforming

In the applications that have been discussed so far the filters/predictors are used to combine samples of the input signal(s) at different time instants to generate the output. Hence, these are classified as *temporal filtering*. Beamforming, however, is different from these in the sense that the inputs to a beamformer are samples of incoming signals at different positions in space. This is called *spatial filtering*. Beamforming finds applications in communications, radar and sonar (Johnson and Dudgeon, 1993), and also imaging in radar and medical engineering (Soumekh, 1994).

In spatial filtering, a number of independent sensors are placed at different points in space to pick up signals coming from various sources (see Figure 1.21). In radar and

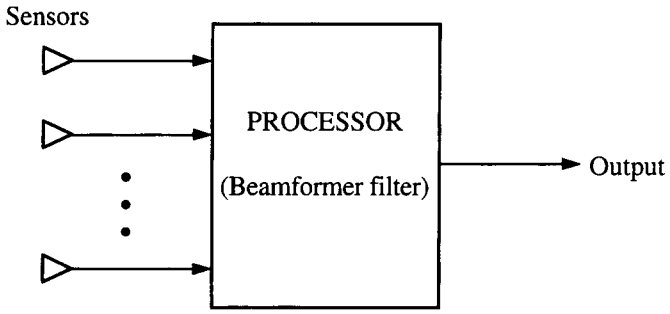


Figure 1.21 Spatial filtering (beamforming)

communications, the signals are usually electromagnetic waves and the sensors are thus antenna elements. Accordingly, the term *antenna arrays* is often used to refer to these applications of beamformers. In sonar applications, the sensors are hydrophones designed to respond to acoustic waves.

In a beamformer, the samples of signals picked up by the sensors at a particular instant of time constitutes a *snapshot*. The samples of snapshot (spatial samples) play the same role as the successive (temporal) samples of input in a transversal filter. The beamformer filter linearly combines the sensors' signals so that signals arriving from some particular directions are amplified, while signals from other directions are attenuated. Thus, in analogy with the frequency response of temporal filters, spatial filters have responses that vary according to the direction of arrival of the in-coming signal(s). This is given in the form of a polar plot (gain vs. angle) and is referred to as the *beam pattern*.

In many applications of beamformers, the signals picked up by sensors are narrow-band having the same carrier (centre) frequency. These signals differ in their directions-of-arrival, which are related to the location of their sources. The operation of beamformers in such applications can be best explained by the following example.

Consider an antenna array consisting of two omni-directional elements A and B, as presented in Figure 1.22. The tone (as an approximation to narrow-band) signals $s(n) = \alpha \cos \omega_0 n$ and $v(n) = \beta \cos \omega_0 n$ arriving at angles 0 and θ_0 (with respect to the line perpendicular to the line connecting A and B), respectively, are the inputs to the array (beamformer) filter which consists of a phase-shifter and a subtracter. The signal $s(n)$

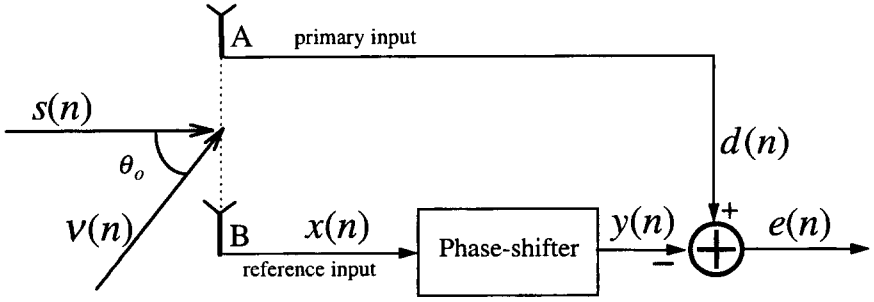


Figure 1.22 A two-element beamformer

arrives at elements A and B at the same time, whereas the arrival times of signal $\nu(n)$ at A and B are different. We may thus write

$$s_A(n) = s_B(n) = \alpha \cos \omega_o n,$$

$$\nu_B(n) = \beta \cos \omega_o n,$$

and

$$\nu_A(n) = \beta \cos(\omega_o n - \varphi),$$

where the subscripts A and B are used to denote the signals picked up by elements A and B, respectively, and φ is the phase-shift arising from the time delay of arrival of $\nu(n)$ at element A with respect to its arrival at element B.

Now, if we assume that $s(n)$ is the desired signal and $\nu(n)$ is an interference, then, by inspection, we can see that if the phase-shifter phase is chosen equal to φ, then the interference, $\nu(n)$, will be completely cancelled by the beamformer. The desired signal, on the other hand, reaches the beamformer output as $\alpha(\cos \omega_o n - \cos(\omega_o n - \varphi))$, which is non-zero (and still holding the information contained in its envelope, α) when $\varphi \neq 0$, i.e. when the interference direction is different from the direction of the desired signal. This shows that we can tune a beamformer to allow the desired signal arriving from a direction to pass through it, while rejecting the unwanted signals (interferences) arriving from other directions.

The idea of using a phase-shifter to adjust the beam pattern of two sensors, is easily extendible to the general case of more than two sensors. In general, by introducing appropriate phase shifts and also gains at the output of the various sensors and summing up these outputs, we can realize any arbitrary beam pattern. This is similar to the selection of tap weights for a transversal filter so that the filter frequency response becomes a good approximation to the desired response. Clearly, by increasing the number of elements in the array, better approximations to the desired beam pattern can be achieved.

The final point that we wish to add here is that in cases where the input signals to the

2

Discrete-Time Signals and Systems

Most adaptive algorithms have been developed for discrete-time (sampled) signals. Discrete-time systems are used for the implementation of adaptive filters. In this chapter we present a short review of discrete-time signals and systems. We assume that the reader is familiar with the basic concepts of discrete-time systems, such as the Nyquist sampling theorem, the z-transform and system function, and also with the theory of random variables and stochastic processes. Our goal, in this chapter, is to review these concepts and put them in a framework appropriate for the rest of the book.

2.1 Sequences and the z-Transform

In discrete-time systems we are concerned with processing signals that are represented by sequences. Such sequences may be samples of a continuous-time analogue signal or may be discrete in nature. As an example, in the channel equalizer structure presented in Figure 1.9, the input sequence to the equalizer, $x(n)$, consists of the samples of the channel output which is an analogue signal, but the original data sequence, $s(n)$, is discrete in nature.

A discrete-time sequence, $x(n)$, may be equivalently represented by its z-transform defined as

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \tag{2.1}$$

where z is a complex variable. The range of values of z for which the above summation converges is called the *region of convergence of* $X(z)$. The following two examples illustrate this.

Example 2.1

Consider the sequence

$$x_1(n) = \begin{cases} a^n, & n \geq 0, \\ 0, & n < 0. \end{cases} \tag{2.2}$$

The z-transform of $x_1(n)$ is

$$X_1(z) = \sum_{n=0}^{\infty} a^n z^{-n}$$
$$= \sum_{n=0}^{\infty} (az^{-1})^n,$$

which converges to

$$X_1(z) = \frac{1}{1 - az^{-1}} \tag{2.3}$$

for $|az^{-1}| < 1$, i.e. $|z| > |a|$. We may also write

$$X_1(z) = \frac{z}{z - a} \tag{2.4}$$

for $|z| > |a|$.

Example 2.2

Consider the sequence

$$x_2(n) = \begin{cases} 0, & n \geq 0, \\ b^n, & n < 0. \end{cases} \tag{2.5}$$

The z-transform of $x_2(n)$ is

$$X_2(z) = \sum_{n=-\infty}^{-1} b^n z^{-n}$$
$$= \sum_{n=1}^{\infty} (b^{-1}z)^n,$$

which converges to

$$X_2(x) = \frac{b^{-1}z}{1 - b^{-1}z} = \frac{z}{b - z} \tag{2.6}$$

for $|z| < |b|$.

The two sequences presented in the above examples are different in many respects. The sequence $x_1(n)$ in Example 2.1 is called *right-sided*, since its non-zero elements start at a finite $n = n_1$ (here, $n_1 = 0$) and extend up to $n = +\infty$. On the other hand, the sequence $x_2(n)$ in Example 2.2 is a *left-sided* one. Its non-zero elements start at a finite $n = n_2$ (here, $n_2 = -1$) and extend up to $n = -\infty$. This definition of right-sided and left-sided sequences also implies that the region of convergence of a right-sided sequence is always the exterior of a circle ($|z| > |a|$ in Example 2.1), while that of a left-sided sequence is always the interior of a circle ($|z| < |b|$ in Example 2.2).

We thus note that the specification of the z-transform, $X(z)$, of a sequence is complete only when its region of convergence is also specified. In other words, the inverse z-transform of $X(z)$ can be uniquely found only if its region of convergence is also specified. For example, one may note that $X_1(z)$ and $X_2(z)$ in the above examples have exactly the same form, except a sign reversal. Hence, if their regions of convergence are not specified, then both may be interpreted as the z-transforms of either left-sided or right-sided sequences.

Two-sided sequences may also exist. A two-sided sequence is one that extends from $n = -\infty$ to $n = +\infty$. The following example shows how to deal with two-sided sequences.

Example 2.3

Consider the sequence

$$x_3(n) = \begin{cases} a^n, & n \geq 0, \\ b^n, & n < 0, \end{cases} \tag{2.7}$$

where $|a| < |b|$. As we shall see, the condition $|a| < |b|$ is necessary to make the convergence of the z-transform of $x_3(n)$ possible. The z-transform of $x_3(n)$ is

$$X_3(z) = \sum_{n=-\infty}^{-1} b^n z^{-n} + \sum_{n=0}^{\infty} a^n z^{-n} \tag{2.8}$$

Clearly, the first sum converges when $|z| < |b|$, and the second sum converges when $|z| > |a|$. Thus, we obtain

$$X_3(z) = \frac{z}{b-z} + \frac{z}{z-a} = \frac{z(a-b)}{(z-a)(z-b)} \tag{2.9}$$

for $|a| < |z| < |b|$.

We may note that the region of convergence of $X_3(z)$ is the area in between two concentric circles. This is true, in general, for all two-sided sequences. For a sequence with a rational z-transform, the radii of the two circles are determined by two of the poles of the z-transform of the sequence. The right-sided part of the sequence is determined by the poles which are surrounded by the region of convergence, and the poles surrounding the region of convergence determine the left-sided part of the sequence. The following example, which also shows one way of calculating the inverse z-transform, clarifies the above points.

Example 2.4

Consider a two-sided sequence, $x(n)$, with the z-transform

$$X(z) = \frac{-0.1z^{-1} + 3.05z^{-2}}{(1 - 0.5z^{-1})(1 + 0.7z^{-1})(1 + 2z^{-1})} \tag{2.10}$$

and the region of convergence $0.7 < |z| < 2$.

To find $x(n)$, i.e. the inverse z-transform of $X(z)$, we use the method of partial fraction and expand $X(z)$ as

$$X(z) = \frac{A}{1 - 0.5z^{-1}} + \frac{B}{1 + 0.7z^{-1}} + \frac{C}{1 + 2z^{-1}},$$

where A, B and C are constants that can be determined as follows:

$$A = (1 - 0.5z^{-1})X(z)|_{z=0.5} = 1$$
$$B = (1 + 0.7z^{-1})X(z)|_{z=-0.7} = -2$$
$$C = (1 + 2z^{-1})X(z)|_{z=-2} = 1.$$

This gives

$$X(z) = \frac{1}{1 - 0.5z^{-1}} - \frac{2}{1 + 0.7z^{-1}} + \frac{1}{1 + 2z^{-1}}. \tag{2.11}$$

We treat each of the terms in the above equation separately. To expand these terms and, from there, extract their corresponding sequences, we use the following identity, which holds for $|a| < 1$:

$$\frac{1}{1 - a} = 1 + a + a^2 + \cdots.$$

We note that within the region of convergence of $X(z)$, both $|0.5z^{-1}|$ and $|0.7z^{-1}|$ are less than one, and thus

$$\frac{1}{1 - 0.5z^{-1}} = 1 + 0.5z^{-1} + 0.5^2z^{-2} + \ldots \tag{2.12}$$

and

$$-\frac{2}{1 + 0.7z^{-1}} = -\frac{2}{1 - (-0.7)z^{-1}}$$
$$= -2\big(1 + (-0.7)z^{-1} + (-0.7)^2z^{-2} + \ldots\big). \tag{2.13}$$

However, for the third term on the right-hand side of (2.11), $|2z^{-1}| > 1$, and, thus, an expansion similar to the last two is not applicable. A similar expansion will be possible if we rearrange this term as

$$\frac{1}{1 + 2z^{-1}} = \frac{0.5z}{1 + 0.5z}.$$

Here, within the region of convergence of $X(z)$, $|0.5z| < 1$, and, thus, we may write

$$\frac{0.5z}{1 + 0.5z} = 0.5z\big(1 + (-0.5)z + (-0.5)^2z^2 + \ldots\big)$$
$$= -(-2)^{-1}z - (-2)^{-2}z^2 - (-2)^{-3}z^3 - \cdots. \tag{2.14}$$

Substituting (2.12), (2.13) and (2.14) into (2.11) and recalling (2.1), we obtain

$$x(n) = \begin{cases} -(-2)^n, & n < 0, \\ 0.5^n - 2(-0.7)^n, & n \geq 0. \end{cases} \tag{2.15}$$

An alternative way of performing an inverse z-transform can be derived by using the *Cauchy integral theorem*, which is stated as follows:

$$\frac{1}{2\pi j}\oint_C z^{k-1}\,\mathrm{d}z = \begin{cases} 1, & k = 0, \\ 0, & k \neq 0, \end{cases} \tag{2.16}$$

where C is a counterclockwise contour that encircles the origin.

The z-transform relation, reproduced here for convenience, is given by

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}. \tag{2.17}$$

Multiplying both sides of (2.17) by z^{k-1} and integrating, we obtain

$$\frac{1}{2\pi j}\oint_C X(z)z^{k-1}\,\mathrm{d}z = \frac{1}{2\pi j}\oint_C \sum_{n=-\infty}^{\infty} x(n)z^{-n+k-1}\,\mathrm{d}z, \tag{2.18}$$

where C is a contour within the region of convergence of $X(z)$ and encircling the origin. Interchanging the order of integration and summation on the right-hand side of (2.18), we obtain

$$\frac{1}{2\pi j}\oint_C X(z)z^{k-1}\,\mathrm{d}z = \sum_{n=-\infty}^{\infty} x(n)\frac{1}{2\pi j}\oint_C z^{-n+k-1}\,\mathrm{d}z. \tag{2.19}$$

Application of the Cauchy integral theorem in (2.19) gives the inverse z-transform relation,

$$x(n) = \frac{1}{2\pi j}\oint_C X(z)z^{n-1}\,\mathrm{d}z, \tag{2.20}$$

where C is a counterclockwise closed contour in the region of convergence of $X(z)$ and encircling the origin of the z-plane.

For rational z-transforms, contour integrals are often conveniently evaluated using the residue theorem, i.e.

$$\begin{aligned} x(n) &= \frac{1}{2\pi j}\oint_C X(z)z^{n-1}\,\mathrm{d}z \\ &= \sum [\text{residues of } X(z)z^{n-1} \text{ at the poles inside } C]. \end{aligned} \tag{2.21}$$

In general, if $X(z)z^{n-1}$ is a rational function of z, and z_p is a pole of $X(z)z^{n-1}$, repeated m times, then

$$\text{residue of } X(z)z^{n-1} \text{ at } z_p = \frac{1}{(m-1)!}\left[\frac{d^{m-1}\psi(z)}{\mathrm{d}z^{m-1}}\right]_{z=z_p}, \tag{2.22}$$

where $\psi(z) = (z - z_p)^m X(z) z^{n-1}$. In particular, if there is a first-order pole at $z = z_p$, i.e. $m = 1$, then

$$\text{residue of } X(z)z^{n-1} \text{ at } z_p = \psi(z_p). \tag{2.23}$$

2.2 Parseval's Relation

Among various important results and properties of the z-transform, in this book we are particularly interested in *Parseval's relation*, which states that for any pair of sequences $x(n)$ and $y(n)$,

$$\sum_{n=-\infty}^{\infty} x(n)y^*(n) = \frac{1}{2\pi j} \oint X(z) Y^*(1/z^*) z^{-1} \, dz, \tag{2.24}$$

where the superscript asterisk denotes complex conjugation and the contour of integration is taken in the overlap of the regions of convergence of $X(z)$ and $Y^*(1/z^*)$. If $X(z)$ and $Y(z)$ converge on the unit circle, then we can choose $z = e^{j\omega}$, and (2.24) becomes

$$\sum_{n=-\infty}^{\infty} x(n)y^*(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) Y^*(e^{j\omega}) \, d\omega. \tag{2.25}$$

Furthermore, if $y(n) = x(n)$, for all n, then (2.25) becomes

$$\sum_{n=-\infty}^{\infty} |x(n)|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X(e^{j\omega})|^2 \, d\omega. \tag{2.26}$$

Equation (2.26) has the following interpretation. *The total energy in a sequence $x(n)$, i.e. $\sum_{n=-\infty}^{\infty} |x(n)|^2$, may be equivalently obtained by averaging $|X(e^{j\omega})|^2$ over one cycle of that.*

2.3 System Function

Consider a discrete-time, linear, time-invariant system with the impulse response $h(n)$. With $x(n)$ and $y(n)$ denoting, respectively, the input and output of the system,

$$y(n) = x(n) * h(n), \tag{2.27}$$

where the asterisk denotes convolution and is defined as

$$x(n) * h(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k). \tag{2.28}$$

Equation (2.27) suggests that any linear, time-invariant system is completely characterized by its impulse response, $h(n)$. Taking the z-transform from both sides of (2.27), we obtain

$$Y(z) = X(z)H(z). \tag{2.29}$$

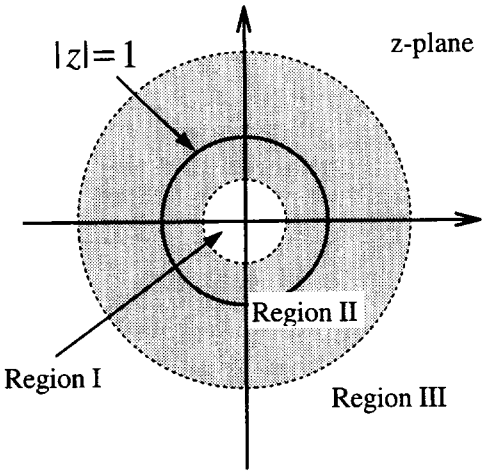


Figure 2.1 Possible regions of convergence for a two-pole z-transform

This shows that the input–output relation for a linear, time-invariant system corresponds to a multiplication of the z-transforms of the input and the impulse response of the system.

The z-transform of the impulse response of a linear, time-invariant system is referred to as its *system function*. The system function evaluated over the unit circle, $|z| = 1$, is the frequency response of the system, $H(e^{j\omega})$. For any particular frequency ω, $H(e^{j\omega})$ is the gain (complex-valued, in general) of the system, when its input is the complex sinusoid $e^{j\omega n}$.

Any stable, linear, time-invariant system has a finite frequency response for all values of ω. This means that the region of convergence of $H(z)$ has to include the unit circle. This fact can be used to determine uniquely the region of convergence of any rational system function, once its poles are known. As an example, if we consider a sequence with the z-transform

$$H(z) = \frac{1}{(1 - 0.5z^{-1})(1 - 2z^{-1})}, \tag{2.30}$$

we find that there are three possible regions of convergence for $H(z)$, as specified in Figure 2.1. These are regions I, II and III, each giving a different time sequence. However, if we assume that $H(z)$ is the system function of a stable, time-invariant system, the only acceptable region of convergence will be region II. Noting this, we obtain (see Problem 2.1)

$$h(n) = \begin{cases} -\frac{4}{3} \times 2^n, & n < 0, \\ -\frac{1}{3} \times 0.5^n, & n \geq 0. \end{cases} \tag{2.31}$$

We note that the impulse response, $h(n)$, obtained above, extends from $n = -\infty$ to $n = +\infty$. This means that, although the input, $\delta(n)$, is applied at time $n = 0$, the system output takes non-zero values even prior to that. Such a system is called *non-causal*. In

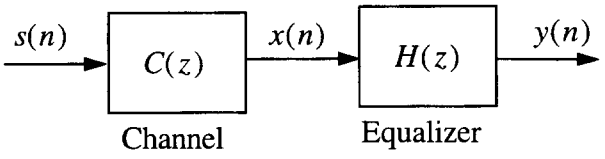


Figure 2.2 A communication system

contrast to this, a system is said to be *causal* if its impulse response is non-zero only for non-negative values of n. Non-causal systems, although not realistic, may be encountered in some theoretical developments. It is important that we find a practical solution for handling such cases. The following example considers such a case and gives a solution to that.

Example 2.5

Figure 2.2 shows a communication system. It consists of a communication channel which is characterized by the system function

$$C(z) = 1 - 2.5z^{-1} + z^{-2} = (1 - 0.5z^{-1})(1 - 2z^{-1}). \tag{2.32}$$

The equalizer, $H(z)$, should be selected so that the original transmitted signal, $s(n)$, can be recovered from the equalizer output without any distortion.

For this, we shall select $H(z)$ so that we get $y(n) = s(n)$. This can be achieved if $H(z)$ is selected so that $C(z)H(z) = 1$. This gives

$$H(z) = \frac{1}{C(z)} = \frac{1}{(1 - 0.5z^{-1})(1 - 2z^{-1})}. \tag{2.33}$$

Noting that this is similar to $H(z)$ in (2.30), we find that the equalizer impulse response is the one given in (2.31). This, of course, is non-causal and, therefore, not realizable. The problem can be easily solved by shifting the non-causal response of the equalizer to the right by sufficient number of samples so that the remaining non-causal samples are sufficiently small and can be ignored. Mathematically, we say

$$H(z) \approx \frac{z^{-\Delta}}{C(z)}, \tag{2.34}$$

where Δ is the number of sample delays introduced to achieve a realizable causal system. We use the approximation sign, $\approx$, in (2.34), since we ignore the non-causal samples of $z^{-\Delta}/C(z)$. The equalizer output is then $s(n - \Delta)$.

2.4 Stochastic Processes

The input signal to an adaptive filter and its desired output are, in general, random, i.e. they are not known a priori. However, they exhibit some statistical characteristics that have to be utilized for optimum adjustment of the filter coefficients. Such random signals

are called *stochastic processes*. Adaptive algorithms are designed to extract these characteristics and use them for adjusting the filter coefficients.

A discrete-time, stochastic process is an indexed set of random variables $\{x(n); n = \ldots, -2, -1, 0, 1, 2, \ldots\}$. As a random signal, the index n is associated with time or possibly some other physical dimension. In this book, for convenience, we frequently refer to n as the time index. So far, we have used the notation $x(n)$ to refer to a particular sequence $x(n)$ that extends from $n = -\infty$ to $n = +\infty$. We use the notation $\{x(n)\}$ for a stochastic process a particular sequence $x(n)$ which may be a single realization of that.

The elements of a stochastic process, $\{x(n)\}$, for different values of n, are in general complex-valued random variables that are characterized by their probability distribution functions. The interrelationships between different elements of $\{x(n)\}$ are determined by their joint distribution functions. Such distribution functions, in general, may change with the time index n. A stochastic process is called *stationary in the strict sense* if all of its (single and joint) distribution functions are independent of a shift in the time origin.

2.4.1 Stochastic averages

It is often useful to characterize stochastic processes by statistical averages of their elements. These averages are called *ensemble averages* and, in general, are time-dependent. For example, the mean of the nth element of a stochastic process $\{x(n)\}$, is defined as

$$m_x(n) = \mathrm{E}[x(n)] \tag{2.35}$$

where $\mathrm{E}[\cdot]$ denotes statistical expectation. It should be noted that since $m_x(n)$ is in general a function of n, it may not be possible to obtain $m_x(n)$ by time averaging of a single realization of the stochastic process $\{x(n)\}$, unless the process possesses certain special properties, indicated at the end of this chapter. Instead, n has to be fixed and averaging has to be done over the nth element of the stochastic process, as a single random variable.

In our later developments we are heavily dependent on the following averages.

1. The *autocorrelation function:* For a stochastic process $\{x(n)\}$, it is defined as

$$\phi_{xx}(n, m) = \mathrm{E}[x(n)x^*(m)] \tag{2.36}$$

   where the superscript asterisk denotes complex conjugation.
2. The *cross-correlation function:* It is defined for two stochastic processes $\{x(n)\}$ and $\{y(n)\}$ as

$$\phi_{xy}(n, m) = \mathrm{E}[x(n)y^*(m)]. \tag{2.37}$$

A stochastic process $\{x(n)\}$ is said to be *stationary in the wide sense* if $m_x(n)$ and $\phi_{xx}(n, m)$ are independent of a shift in the time origin. That is, for any k, m and n,

$$m_x(n) = m_x(n + k)$$

and

$$\phi_{xx}(n, m) = \phi_{xx}(n + k, m + k).$$

These imply that $m_x(n)$ is a constant for all n and $\phi_{xx}(n, m)$ depends on the difference $n - m$ only. Then, it would be more appropriate to define the autocorrelation function of $\{x(n)\}$ as

$$\phi_{xx}(k) = \mathrm{E}[x(n)x^*(n - k)]. \tag{2.38}$$

Similarly, the processes $\{x(n)\}$ and $\{y(n)\}$ are said to be jointly stationary in the wide sense if their means are independent of n, and $\phi_{xy}(n, m)$ depends on $n - m$ only. We may then define the cross-correlation function of $\{x(n)\}$ and $\{y(n)\}$ as

$$\phi_{xy}(k) = \mathrm{E}[x(n)y^*(n - k)]. \tag{2.39}$$

Besides the autocorrelation and cross-correlation functions, the autocovariance and cross-covariance functions are also defined. For stationary processes, these are defined as

$$\gamma_{xx}(k) = \mathrm{E}[(x(n) - m_x)(x(n - k) - m_x)^*] \tag{2.40}$$

and

$$\gamma_{xy}(k) = \mathrm{E}[(x(n) - m_x)(y(n - k) - m_y)^*], \tag{2.41}$$

respectively. By expanding the right-hand sides of (2.40) and (2.41), we obtain

$$\gamma_{xx}(k) = \phi_{xx}(k) - |m_x|^2 \tag{2.42}$$

and

Other important properties of the correlation and covariance functions that should be noted here are their symmetry properties which are summarized below:

$$\phi_{xx}(k) = \phi^*_{xx}(-k), \tag{2.48}$$

$$\gamma_{xx}(k) = \gamma^*_{xx}(-k), \tag{2.49}$$

$$\phi_{xy}(k) = \phi^*_{yx}(-k), \tag{2.50}$$

$$\gamma_{xy}(k) = \gamma^*_{yx}(-k). \tag{2.51}$$

We may also note that

$$\phi_{xx}(0) = \mathrm{E}[|x(n)|^2] = \text{mean-square of } x(n) \tag{2.52}$$

$$\gamma_{xx}(0) = \sigma_x^2 = \text{variance of } x(n). \tag{2.53}$$

2.4.2 z-transform representations

The z-transform of $\phi_{xx}(k)$ is given by

$$\Phi_{xx}(z) = \sum_{k=-\infty}^{\infty} \phi_{xx}(k) z^{-k}. \tag{2.54}$$

We note that a necessary condition for $\Phi_{xx}(z)$ to be convergent is that m_x should be zero (see Problem P2.3). We assume this for the random processes that are considered in the rest of this chapter, and also the following chapters. Exceptional cases will be mentioned explicitly.

From (2.48) we note that

$$\Phi_{xx}(z) = \Phi^*_{xx}(1/z^*). \tag{2.55}$$

Similarly, if $\Phi_{xy}(z)$ denotes the z-transform of $\phi_{xy}(k)$, then

$$\Phi_{xy}(z) = \Phi^*_{yx}(1/z^*). \tag{2.56}$$

Equation (2.55) implies that if $\Phi_{xx}(z)$ is a rational function of z, then its poles and zeros must occur in complex-conjugate reciprocal pairs, as depicted in Figure 2.3. Moreover, (2.55) implies that the points that belong to the region of convergence of $\Phi_{xx}(z)$ also occur in complex-conjugate reciprocal pairs. This in turn suggests that the region of convergence of $\Phi_{xx}(z)$ must be of the form (see Figure 2.3)

$$|a| < |z| < \frac{1}{|a|}. \tag{2.57}$$

It is important that we note this covers the unit circle, $|z| = 1$.

The inverse z-transform relation (2.20) may be used to evaluate $\phi_{xx}(0)$ as

$$\phi_{xx}(0) = \frac{1}{2\pi j} \oint_C \Phi_{xx}(z) z^{-1} \, \mathrm{d}z. \tag{2.58}$$

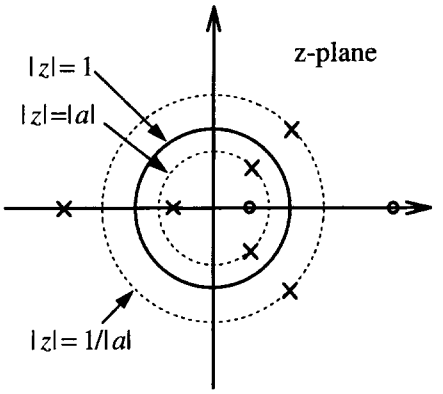


Figure 2.3 Poles and zeros of a typical z-transform of an autocorrelation function

We assume that $\Phi_{xx}(z)$ is convergent on the unit circle and select the unit circle as the contour of integration. For this we substitute z by $e^{j\omega}$. Then, ω changes from $-\pi$ to $+\pi$ as we traverse the unit circle once. Noting that $z^{-1}\,dz = j\,d\omega$, (2.58) becomes

$$\phi_{xx}(0) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_{xx}(e^{j\omega})\,d\omega. \tag{2.59}$$

Since $m_x = 0$, we can combine (2.52) and (2.53) with (2.59) to obtain

$$\sigma_x^2 = E[|x(n)|^2] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_{xx}(e^{j\omega})\,d\omega. \tag{2.60}$$

2.4.3 The power spectral density

The function $\Phi_{xx}(z)$, when evaluated on the unit circle, is the Fourier transform of the autocorrelation sequence $\phi_{xx}(k)$. It is called the *power spectral density* since it reflects the spectral content of the underlying process as a function of frequency. It is also called the *power spectrum* or simply the *spectrum*. The convergence performance of an adaptive filter is directly related to the spectrum of its input process. Next, we present a direct development of the power spectral density of a wide-sense, stationary, discrete-time, stochastic process which reveals its important properties.

Consider a sequence $x(n)$ which represents a single realization of a zero-mean, wide-sense, stationary, stochastic process $\{x(n)\}$. We consider a window of $2N + 1$ elements of $x(n)$ as

$$x_N(n) = \begin{cases} x(n), & -N \le n \le N, \\ 0, & \text{otherwise.} \end{cases} \tag{2.61}$$

By definition, the discrete-time Fourier transform of $x_N(n)$ is

$$X_N(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x_N(n)\,e^{-j\omega n} = \sum_{n=-N}^{N} x(n)\,e^{-j\omega n}. \tag{2.62}$$

Conjugating both sides of (2.62), and replacing n by m, we obtain

$$X_N^*(e^{j\omega}) = \sum_{m=-N}^{N} x^*(m)\, e^{j\omega m}. \tag{2.63}$$

Next, we multiply (2.62) and (2.63) to obtain

$$|X_N(e^{j\omega})|^2 = \sum_{n=-N}^{N} \sum_{m=-N}^{N} x(n)x^*(m)\, e^{-j\omega(n-m)}. \tag{2.64}$$

Taking the expectation on both sides of (2.64), and interchanging the order of expectation and double summation, we get

$$E[|X_N(e^{j\omega})|^2] = \sum_{n=-N}^{N} \sum_{m=-N}^{N} E[x(n)x^*(m)]\, e^{-j\omega(n-m)}. \tag{2.65}$$

Noting that $E[x(n)x^*(m)] = \phi_{xx}(n-m)$, and letting $k = n - m$, we may rearrange the terms in (2.65) to obtain

$$\frac{1}{2N+1} E[|X_N(e^{j\omega})|^2] = \sum_{k=-2N}^{2N} \left(1 - \frac{|k|}{2N+1}\right) \phi_{xx}(k)\, e^{-j\omega k}. \tag{2.66}$$

To simplify (2.66) we assume that for k greater than an arbitrary large constant, but less than infinity, $\phi_{xx}(k)$ is identically equal to zero. This, in general, is a fair assumption, unless the $\{x(n)\}$ contains sinusoidal components, in which case the summation on the right-hand side of (2.66) will not be convergent. With this assumption, we get from (2.66)

$$\lim_{N\to\infty} \frac{1}{2N+1} E[|X_N(e^{j\omega})|^2] = \sum_{k=-\infty}^{\infty} \phi_{xx}(k)\, e^{-j\omega k}, \tag{2.67}$$

which is nothing but the Fourier transform of the autocorrelation function, $\phi_{xx}(k)$. We may thus write

$$\Phi_{xx}(e^{j\omega}) = \lim_{N\to\infty} \frac{1}{2N+1} E[|X_N(e^{j\omega})|^2]. \tag{2.68}$$

The function $\Phi_{xx}(e^{j\omega})$ is called the power spectral density of the stochastic, wide-sense, stationary process $\{x(n)\}$. It is defined as in (2.68) or, more conveniently, as the Fourier transform of the autocorrelation function of $\{x(n)\}$:

$$\Phi_{xx}(e^{j\omega}) = \sum_{k=-\infty}^{\infty} \phi_{xx}(k)\, e^{-j\omega k}. \tag{2.69}$$

The power spectral density possesses certain special properties. These are indicated below for later reference.

Property 1 *When the limit in* (2.68) *exists,* $\Phi_{xx}(e^{j\omega})$ *has the following interpretation:*

$$\frac{1}{2\pi}\Phi_{xx}(e^{j\omega})\,d\omega = \text{average contribution of the frequency components of } \{x(n)\} \text{ located between } \omega \text{ and } \omega + d\omega. \tag{2.70}$$

This interpretation matches (2.60), if both sides of (2.70) are integrated over ω from $-\pi$ to $+\pi$. We will elaborate more on this later, once we introduce response of linear systems to random signals; see Example 2.6.

Property 2 *The power spectral density* $\Phi_{xx}(e^{j\omega})$ *is always real and non-negative.*

This property is obvious from definition (2.68), since $|X_N(e^{j\omega})|^2$ is always real and non-negative.

Property 3 *The power spectral density of a real-valued stationary stochastic process is even, i.e. symmetric with respect to the origin* $\omega = 0$. *In other words,*

$$\Phi_{xx}(e^{j\omega}) = \Phi_{xx}(e^{-j\omega}). \tag{2.71}$$

However, this may not true when the process is complex-valued.

This follows from (2.69) by replacing k with $-k$ and noting that for a real-valued stationary process $\phi_{xx}(k) = \phi_{xx}(-k)$.

2.4.4 Response of linear systems to stochastic processes

We consider a linear, time-invariant, discrete-time system with input $\{x(n)\}$, output $\{y(n)\}$ and impulse response $h(n)$, as depicted in Figure 2.4. The input and output sequences are stochastic processes, but the system impulse response, $h(n)$, is a deterministic sequence. Since $\{x(n)\}$ and $\{y(n)\}$ are stochastic processes, we are interested in finding how they are statistically related. We assume that $\phi_{xx}(k)$ is known, and find the relationships that relate this with $\phi_{xy}(k)$ and $\phi_{yy}(k)$. These relationships can be conveniently established through the z-transforms of the sequences.

We note that

$$\begin{aligned}\Phi_{xy}(z) &= \sum_{k=-\infty}^{\infty} \phi_{xy}(k)z^{-k} \\ &= \sum_{k=-\infty}^{\infty} \mathrm{E}[x(n)y^*(n-k)]z^{-k} \\ &= \sum_{k=-\infty}^{\infty} E\left[x(n)\sum_{l=-\infty}^{\infty} h^*(l)x^*(n-k-l)\right]z^{-k}.\end{aligned} \tag{2.72}$$

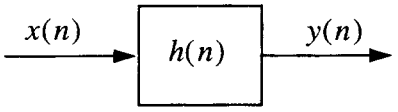


Figure 2.4 A linear time-invariant system

Since both summation and expectation are linear operators, their orders can be interchanged. Using this in (2.72) we get

$$\Phi_{xy}(z) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h^*(l)\mathrm{E}[x(n)x^*(n-k-l)]z^{-k}$$

$$= \sum_{l=-\infty}^{\infty} h^*(l) \sum_{k=-\infty}^{\infty} \phi_{xx}(k+l)z^{-k}. \tag{2.73}$$

If we substitute $k+l$ by m, we get

$$\Phi_{xy}(z) = \sum_{l=-\infty}^{\infty} h^*(l) \sum_{m=-\infty}^{\infty} \phi_{xx}(m)z^{-(m-l)}$$

$$= \sum_{l=-\infty}^{\infty} h^*(l)z^{l} \sum_{m=-\infty}^{\infty} \phi_{xx}(m)z^{-m}. \tag{2.74}$$

This gives

$$\Phi_{xy}(z) = H^*(1/z^*)\Phi_{xx}(z), \tag{2.75}$$

where $H(z)$ follows the conventional definition

$$H(z) = \sum_{n=-\infty}^{\infty} h(n)z^{-n}. \tag{2.76}$$

Furthermore, using (2.55), (2.56) and (2.75) we can also get

$$\Phi_{yx}(z) = H(z)\Phi_{xx}(z). \tag{2.77}$$

The autocorrelation of the output process $\{y(n)\}$ is obtained as follows:

$$\Phi_{yy}(z) = \sum_{k=-\infty}^{\infty} \phi_{yy}(k)z^{-k}$$

$$= \sum_{k=-\infty}^{\infty} \mathrm{E}[y(n)y^*(n-k)]z^{-k}$$

$$= \sum_{k=-\infty}^{\infty} \mathrm{E}\left[\sum_{l=-\infty}^{\infty} h(l)x(n-l) \sum_{m=-\infty}^{\infty} h^*(m)x^*(n-k-m)\right]z^{-k}$$

$$= \sum_{l=-\infty}^{\infty} h(l) \sum_{m=-\infty}^{\infty} h^*(m) \sum_{k=-\infty}^{\infty} \mathrm{E}[x(n-l)x^*(n-k-m)]z^{-k}$$

$$= \sum_{l=-\infty}^{\infty} h(l) \sum_{m=-\infty}^{\infty} h^*(m) \sum_{k=-\infty}^{\infty} \phi_{xx}(k+m-l)z^{-k}. \tag{2.78}$$

Substituting $k + m - l$ by p, we get

$$\Phi_{yy}(z) = \sum_{l=-\infty}^{\infty} h(l)z^{-l} \sum_{m=-\infty}^{\infty} h^*(m)z^m \sum_{p=-\infty}^{\infty} \phi_{xx}(p)z^{-p} \tag{2.79}$$

or

$$\Phi_{yy}(z) = H(z)H^*(1/z^*)\Phi_{xx}(z). \tag{2.80}$$

It would be also convenient if we assume that z varies only over the unit circle, i.e. $|z| = 1$. In that case $1/z^* = z$ and (2.75) and (2.80) simplify to

$$\Phi_{xy}(z) = H^*(z)\Phi_{xx}(z) \tag{2.81}$$

and

$$\Phi_{yy}(z) = H(z)H^*(z)\Phi_{xx}(z) = |H(z)|^2\Phi_{xx}(z), \tag{2.82}$$

respectively. Also, by replacing z with $e^{j\omega}$, we obtain

$$\Phi_{xy}(e^{j\omega}) = H^*(e^{j\omega})\Phi_{xx}(e^{j\omega}), \tag{2.83}$$

$$\Phi_{yx}(e^{j\omega}) = H(e^{j\omega})\Phi_{xx}(e^{j\omega}), \tag{2.84}$$

$$\Phi_{yy}(e^{j\omega}) = |H(e^{j\omega})|^2\Phi_{xx}(e^{j\omega}). \tag{2.85}$$

These equations show how the cross power spectral densities, $\Phi_{xy}(e^{j\omega})$ and $\Phi_{yx}(e^{j\omega})$, and also the output power spectral density, $\Phi_{yy}(e^{j\omega})$, are related to the input power spectral density, $\Phi_{xx}(e^{j\omega})$, and the system transfer function, $H(e^{j\omega})$. As a useful application of the above results, we consider the following example.

Example 2.6

Consider a bandpass filter with a magnitude response as in Figure 2.5. The input process to the filter is a zero-mean, wide-sense, stationary, stochastic process $\{x(n)\}$. We are interested in finding

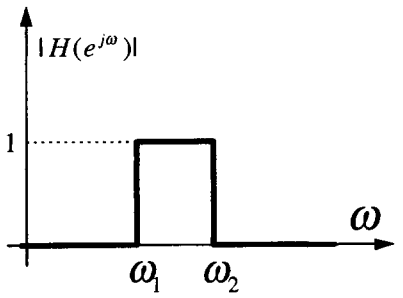


Figure 2.5 Magnitude response of a bandpass filter

the variance of the filter output, $\{y(n)\}$. We note that

$$|H(e^{j\omega})| = \begin{cases} 1, & \omega_1 \le \omega \le \omega_2, \\ 0, & \text{otherwise.} \end{cases} \tag{2.86}$$

Substituting this into (2.85) and using an equation similar to (2.60) for $\{y(n)\}$, we obtain

$$\sigma_y^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} |H(e^{j\omega})|^2 \Phi_{xx}(e^{j\omega})\, d\omega = \frac{1}{2\pi}\int_{\omega_1}^{\omega_2} \Phi_{xx}(e^{j\omega})\, d\omega. \tag{2.87}$$

If ω_2 approaches ω_1, then we may write $\omega_2 - \omega_1 = d\omega$, where $d\omega$ is a variable approaching zero. In that case, we may write

$$\sigma_y^2 = \frac{1}{2\pi}\Phi_{xx}(e^{j\omega_1})\, d\omega.$$

This proves the interpretation of the power spectral density given by Property 1 in Section 2.4.3, i.e. (2.70).

Consider the case where there is a third process, $\{d(n)\}$, whose cross-correlation with the input process, $\{x(n)\}$, of Figure 2.4 is known. We are interested in finding the cross-correlation of $\{d(n)\}$ and $\{y(n)\}$.

In terms of z-transforms, we have

$$\begin{aligned} \Phi_{dy}(z) &= \sum_{k=-\infty}^{\infty} \phi_{dy}(k) z^{-k} \\ &= \sum_{k=-\infty}^{\infty} \mathrm{E}[d(n) y^*(n-k)] z^{-k} \\ &= \sum_{k=-\infty}^{\infty} \mathrm{E}\left[d(n) \sum_{l=-\infty}^{\infty} h^*(l) x^*(n-k-l) \right] z^{-k} \\ &= \sum_{l=-\infty}^{\infty} h^*(l) \sum_{k=-\infty}^{\infty} \phi_{dx}(k+l) z^{-k}. \end{aligned} \tag{2.88}$$

Substituting $k + l$ by m, we obtain

$$\Phi_{dy}(z) = \sum_{l=-\infty}^{\infty} h^*(l) z^{l} \sum_{m=-\infty}^{\infty} \phi_{dx}(m) z^{-m} \tag{2.89}$$

or

$$\Phi_{dy}(z) = H^*(1/z^*)\Phi_{dx}(z). \tag{2.90}$$

Also, using (2.56) we can get, from (2.90),

$$\Phi_{yd}(z) = H(z)\Phi_{xd}(z). \tag{2.91}$$

2.4.5 Ergodicity and time averages

The estimation of stochastic averages, as was suggested above, requires a large number of realizations (sample sequences) of the underlying stochastic processes, even under the condition that the processes are stationary. This is not feasible, in practice, where usually only a single realization of each stochastic process is available. In that case, we have no choice but to use time averages to estimate the desired ensemble averages. Then, a fundamental question that arises is the following. Under what condition(s) do time averages become equal to ensemble averages? As may be intuitively understood, it turns out that under rather mild conditions the only requirement for the time and ensemble averages to be the same is that the corresponding stochastic process be stationary; see Papoulis (1991).

A stationary stochastic process $\{x(n)\}$ is said to be ergodic if its ensemble averages are equal to time averages. Ergodicity is usually defined for specific averages. For example, we may come across terms such as mean-ergodic or correlation-ergodic. In adaptive filters theory it is always assumed that all the underlying processes are *ergodic in the strict sense*. This means that all averages can be obtained by time averages. We make such assumption throughout this book, whenever necessary.

Problems

P2.1 Find the inverse z-transform of (2.30) when

(i) its region of convergence is region I of Figure 2.1;
(ii) its region of convergence is region II of Figure 2.1;
(iii) its region of convergence is region III of Figure 2.1.

P2.2 Use the basic definitions of the correlation and covariance functions to prove the symmetry properties (2.48)–(2.51).

P2.3 Show that for a stationary stochastic process, $\{x(n)\}$, if $m_x \neq 0$, then $\Phi_{xx}(z)$ contains a summation that is not convergent for any value of the complex variable z.

P2.4 Consider a stationary stochastic process,

$$x(n) = \nu(n) + \sin(\omega_0 n + \theta)$$

where $\{\nu(n)\}$ is a stationary white noise, ω_0 is a fixed angular frequency, and θ is a random phase which is uniformly distributed in the interval $-\pi \leq \theta \leq \pi$, but constant for each realization of $\{x(n)\}$. Find the autocorrelation function of $\{x(n)\}$ and show that $\Phi_{xx}(z)$ has no region of convergence in the z-plane.

P2.5 Prove the symmetry equations (2.55) and (2.56).

P2.6 A stationary white noise process, $\{\nu(n)\}$, is passed through a linear time-invariant system with the system function

$$H(z) = \frac{1}{1 - az^{-1}}, \quad |a| < 1.$$

If the system output is referred to as $\{u(n)\}$, find the followings:

(i) $\Phi_{vu}(z)$ and $\Phi_{uu}(z)$.
(ii) The cross-correlation and autocorrelation functions $\phi_{vu}(k)$ and $\phi_{uu}(k)$.

P2.7 Repeat P2.6 when

$$H(z) = \frac{1}{(1 - az^{-1})(1 - bz^{-1})}, \quad |a| \text{ and } |b| < 1.$$

Find the answers for the two cases when $a = b$ and $a \neq b$.

P2.8 Repeat P2.6 when $H(z)$ is a finite-duration impulse response system with

$$H(z) = \sum_{n=0}^{N-1} h(n)z^{-n}.$$

P2.9 Work out the details of derivation of (2.66) from (2.65).

P2.10 By direct derivation show that for a linear time-invariant system with input $\{x(n)\}$, output $\{y(n)\}$ and system function $H(z)$

$$\Phi_{yx}(z) = H(z)\Phi_{xx}(z).$$

Also, if $\{d(n)\}$ is a third process

$$\Phi_{yd}(z) = H(z)\Phi_{xd}(z).$$

P2.11 Write the following z-transform relations in terms of the time series $h(n)$ and the correlation functions:

(i) $\Phi_{yx}(z) = H(z)\Phi_{xx}(z)$.
(ii) $\Phi_{xy}(z) = H^*(1/z^*)\Phi_{xx}(z)$.
(iii) $\Phi_{yd}(z) = H(z)\Phi_{xd}(z)$.
(iv) $\Phi_{dy}(z) = H^*(1/z^*)\Phi_{dx}(z)$.

P2.12 Consider the system shown in Figure P2.12. The input processes, $\{u(n)\}$ and $\{v(n)\}$, are zero-mean and uncorrelated with each other. Derive the relationships that

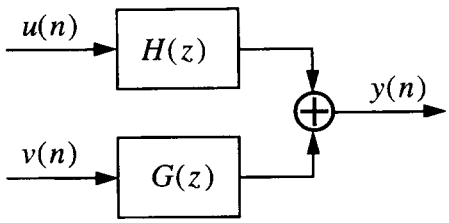


Figure P2.12

relate $\Phi_{uu}(z)$, $\Phi_{vv}(z)$, $H(z)$ and $G(z)$ with the following functions:

(i) $\Phi_{uy}(z)$.
(ii) $\Phi_{vy}(z)$.
(iii) $\Phi_{yy}(z)$.

P2.13 Consider the system shown in Figure P2.13. The input, $\{v(n)\}$, is a stationary zero-mean, unit-variance, white noise process. Show that

(i) $\Phi_{xx}(z) = \dfrac{1}{(1-0.5z^{-1})(1-0.5z)}$.
(ii) $\Phi_{yy}(z) = 4$.
(iii) $\Phi_{vy}(z) = \dfrac{1-2z}{1-0.5z}$.

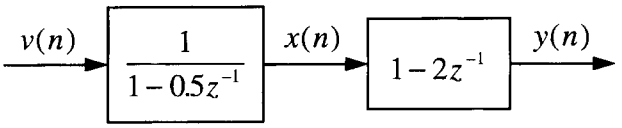


Figure P2.13

P2.14 Consider the system shown in Figure P2.14. The input, $\{v(n)\}$, is a stationary zero-mean, unit-variance, white noise process. Show that

(i) $\phi_{xy}(m) = \sum_l h(l+m)g^*(l)$,
(ii) $\Phi_{xy}(z) = H(z)G^*(1/z^*)$,

where $h(n)$ and $g(n)$ are the impulse responses of the subsystems $H(z)$ and $G(z)$, respectively.

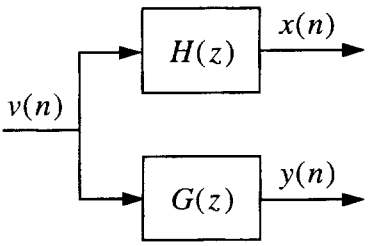


Figure P2.14

3
Wiener Filters

In this chapter we study a class of optimum linear filters known as *Wiener filters*. As we will see in later chapters, the concept of Wiener filters is essential as well as helpful to understand and appreciate adaptive filters. Furthermore, Wiener filtering is general and applicable to any application that involves linear estimation of a desired signal sequence from another related sequence. Applications such as prediction, smoothing, joint process estimation, and channel equalization (deconvolution) are all covered by Wiener filters.

We study Wiener filters by looking at them from different angles. We first develop the theory of causal transversal Wiener filters for the case of discrete-time, real-valued signals. This will then be extended to the case of complex-valued signals. Our discussion follows with a study of *unconstrained Wiener filters*. The term *unconstrained* signifies that the filter impulse response is allowed to be non-causal and infinite in duration. The study of unconstrained Wiener filters is very instructive, as it reveals many important aspects of Wiener filters which otherwise would be difficult to see.

In the theory of Wiener filters the underlying signals are assumed to be random processes and the filter is designed using the statistics obtained by ensemble averaging. We follow this approach while doing the theoretical development and analysis of Wiener filters. However, from the implementation point of view and, in particular, while developing adaptive algorithms in later chapters, we have to consider the use of time averages instead of ensemble averages. The adoption of this approach in the development of Wiener filters is also possible, once we assume all the underlying processes are ergodic; that is, their time and ensemble averages are the same (see Section 2.4.5).

3.1 Mean-Square Error Criterion

Figure 3.1 shows the block schematic of a linear discrete-time filter $W(z)$ in the context of estimating a desired signal $d(n)$ based on an excitation $x(n)$. Here, we assume that both $x(n)$ and $d(n)$ are samples of infinite length, random processes. The filter output is $y(n)$ and $e(n)$ is the estimation error. Clearly, the smaller the estimation error, the better the filter performance. As the error approaches zero, the output of the filter approaches the desired signal, $d(n)$. Hence, the question that arises is the following: What is the most appropriate choice for the parameters of the filter which would result in the smallest possible estimation error? To a certain extent, the statement of this question itself gives

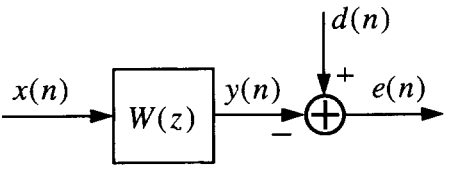


Figure 3.1 Block diagram of a filtering problem

us some hints on the choice of the filter parameters. Since we want the estimation error to be as small as possible, a straightforward approach to the design of the filter parameters appears to be 'to choose an *appropriate* function of this estimation error as a *cost function* and select that set of filter parameters which *optimizes* this cost function *in some sense*'. This is indeed the philosophy that underlies almost all filter design approaches. The various details of this design principle will become clear as we go along. Commonly used synonyms for the cost function are the *performance function* and the *performance surface*.

In choosing a performance function the following points have to be considered:

1. The performance function must be mathematically tractable.
2. The performance function should preferably have a single minimum (or maximum) point, so that the optimum set of filter parameters could be selected unambiguously.

The tractability of the performance function is essential, as it permits analysis of the filter and also greatly simplifies the development of adaptive algorithms for adjustment of the filter parameters. The number of minima (or maxima) points for a performance function is closely related to the filter structure. The *recursive* (infinite-duration impulse response – IIR) filters, in general, result in performance functions that may have many minima (or maxima) points, whereas the *non-recursive* (finite-duration impulse response – FIR) filters are guaranteed to have a single global minimum (or maximum) point if a proper performance function is used. Because of this, application of the IIR filters in adaptive filtering has been very limited. In this book, also, with the exception of a few cases, our discussion is limited to FIR adaptive filters.

In Wiener filters the performance function is chosen to be

$$\xi = \mathrm{E}[|e(n)|^2], \tag{3.1}$$

where $\mathrm{E}[\cdot]$ denotes statistical expectation. In fact, the performance function ξ, which is also called the *mean-square error criterion*, turns out to be the simplest possible function that satisfies the two requirements noted above. It can easily be handled mathematically, and in many cases of interest it has a single global minimum. In particular, in the case of FIR filters the performance function ξ is a hyperparaboloid (bowl shaped) with a single minimum point which can easily be calculated by using the second-order statistics of the underlying random processes.

It is instructive to note that a possible generalization of the mean-square error criterion (3.1) is

$$\xi_p = \mathrm{E}[|e(n)|^p], \tag{3.2}$$

where p takes integer values $1, 2, 3, \ldots$. Clearly, the case of $p = 2$ leads to the Wiener filter performance function defined above. Cases where $p > 2$, with p being even, may result in more than one minimum and/or maximum point. Furthermore, the case of odd p turns out to be difficult to handle mathematically, because of the modulus sign on $e(n)$.

3.2 Wiener Filter – the Transversal, Real-Valued Case

Consider a transversal filter as shown in Figure 3.2. The filter input, $x(n)$, and its desired output, $d(n)$, are assumed to be real-valued stationary processes. The filter tap weights, $w_0, w_1, \ldots, w_{N-1}$, are also assumed to be real-valued. The filter input and tap-weight vectors are defined, respectively, as the column vectors

$$\mathbf{w} = [w_0 \;\; w_1 \;\; \ldots \;\; w_{N-1}]^{\mathrm{T}}, \tag{3.3}$$

and

$$\mathbf{x}(n) = [x(n) \;\; x(n-1) \;\; \ldots \;\; x(n-N+1)]^{\mathrm{T}}, \tag{3.4}$$

where the superscript T stands for transpose.

The filter output is

$$y(n) = \sum_{i=0}^{N-1} w_i x(n-i) = \mathbf{w}^{\mathrm{T}}\mathbf{x}(n), \tag{3.5}$$

which can also be written as

$$y(n) = \mathbf{x}^{\mathrm{T}}(n)\mathbf{w}, \tag{3.6}$$

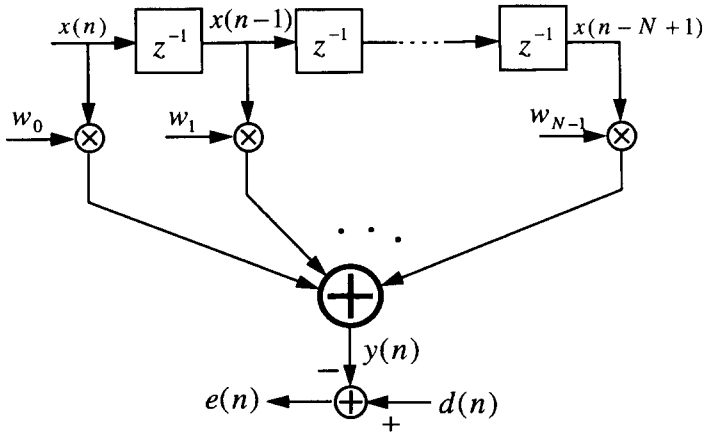


Figure 3.2 A transversal filter

since $\mathbf{w}^{\mathrm{T}}\mathbf{x}(n)$ is a scalar and thus it is equal to its transpose, i.e. $\mathbf{w}^{\mathrm{T}}\mathbf{x}(n) = (\mathbf{w}^{\mathrm{T}}\mathbf{x}(n))^{\mathrm{T}} = \mathbf{x}^{\mathrm{T}}(n)\mathbf{w}$. Thus, we may write

$$\begin{aligned} e(n) &= d(n) - y(n) \\ &= d(n) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(n) \\ &= d(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{w}. \end{aligned} \tag{3.7}$$

Using (3.7) in (3.1) we get

$$\xi = \mathrm{E}[e^2(n)] = \mathrm{E}[(d(n) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(n))(d(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{w})]. \tag{3.8}$$

Expanding the right-hand side of (3.8) and noting that $\mathbf{w}$ can be shifted out of the expectation operator, $\mathrm{E}[\cdot]$, since it is not a statistical variable, we obtain

$$\xi = \mathrm{E}[d^2(n)] - \mathbf{w}^{\mathrm{T}}\mathrm{E}[\mathbf{x}(n)d(n)] - \mathrm{E}[d(n)\mathbf{x}^{\mathrm{T}}(n)]\mathbf{w} + \mathbf{w}^{\mathrm{T}}\mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]\mathbf{w}. \tag{3.9}$$

Next, if we define the $N \times 1$ cross-correlation vector

$$\mathbf{p} = \mathrm{E}[\mathbf{x}(n)d(n)] = [p_0 \;\; p_1 \;\; \cdots \;\; p_{N-1}]^{\mathrm{T}}, \tag{3.10}$$

and the $N \times N$ autocorrelation matrix

$$\mathbf{R} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)] = \begin{bmatrix} r_{00} & r_{01} & r_{02} & \cdots & r_{0,N-1} \\ r_{10} & r_{11} & r_{12} & \cdots & r_{1,N-1} \\ r_{20} & r_{21} & r_{22} & \cdots & r_{2,N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{N-1,0} & r_{N-1,1} & r_{N-1,2} & \cdots & r_{N-1,N-1} \end{bmatrix}, \tag{3.11}$$

and note that $\mathrm{E}[d(n)\mathbf{x}^{\mathrm{T}}(n)] = \mathbf{p}^{\mathrm{T}}$, and also $\mathbf{w}^{\mathrm{T}}\mathbf{p} = \mathbf{p}^{\mathrm{T}}\mathbf{w}$, we obtain

$$\xi = \mathrm{E}[d^2(n)] - 2\mathbf{w}^{\mathrm{T}}\mathbf{p} + \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w}. \tag{3.12}$$

This is a quadratic function of the tap-weight vector $\mathbf{w}$ with a single global minimum.[1] We give the full details of this function in Chapter 4.

To obtain the set of tap weights that minimizes the performance function ξ, we need to solve the system of equations that results from setting the partial derivatives of ξ with respect to every tap weight to zero. That is,

$$\frac{\partial \xi}{\partial w_i} = 0, \quad \text{for } i = 0, 1, \ldots, N-1. \tag{3.13}$$

[1] It may be noted that for (3.12) to correspond to a convex quadratic surface, so that it has a unique minimum point, and not a saddle point, $\mathbf{R}$ has to be a positive definite matrix. This point, which is missed out here, will be examined in detail in Chapter 4.

These equations may collectively be written as

$$\nabla\xi = \mathbf{0}, \tag{3.14}$$

where ∇ is the gradient operator defined as the column vector

$$\nabla = \left[\frac{\partial}{\partial w_0} \quad \frac{\partial}{\partial w_1} \quad \cdots \quad \frac{\partial}{\partial w_{N-1}}\right]^{\mathrm{T}}, \tag{3.15}$$

and $\mathbf{0}$ on the right-hand side of (3.14) denotes the column vector consisting of N zeros.

To find the partial derivatives of ξ with respect to the filter tap weights, we first expand (3.12) as

$$\xi = \mathrm{E}[d^2(n)] - 2\sum_{l=0}^{N-1} p_l w_l + \sum_{l=0}^{N-1}\sum_{m=0}^{N-1} w_l w_m r_{lm}. \tag{3.16}$$

Also, we note that the double summation on the right-hand side of (3.16) may be expanded as

$$\sum_{l=0}^{N-1}\sum_{m=0}^{N-1} w_l w_m r_{lm} = \sum_{\substack{l=0\\ l\neq i}}^{N-1}\sum_{\substack{m=0\\ m\neq i}}^{N-1} w_l w_m r_{lm} + w_i \sum_{\substack{l=0\\ l\neq i}}^{N-1} w_l r_{li} + w_i \sum_{\substack{m=0\\ m\neq i}}^{N-1} w_m r_{im} + w_i^2 r_{ii}. \tag{3.17}$$

Substituting (3.17) into (3.16), taking partial derivative of ξ with respect to w_i, and replacing m by l, we obtain

$$\frac{\partial\xi}{\partial w_i} = -2p_i + \sum_{l=0}^{N-1} w_l(r_{li} + r_{il}), \quad \text{for } i = 0, 1, \ldots, N-1. \tag{3.18}$$

To simplify this, we note that

$$r_{li} = \mathrm{E}[x(n-l)x(n-i)] = \phi_{xx}(i-l), \tag{3.19}$$

where $\phi_{xx}(i-l)$ is the autocorrelation function of $x(n)$ for lag $i-l$. Similarly,

$$r_{il} = \phi_{xx}(l-i). \tag{3.20}$$

Considering the symmetry property of the autocorrelation function, i.e. $\phi_{xx}(k) = \phi_{xx}(-k)$, we get

$$r_{li} = r_{il}. \tag{3.21}$$

Substituting (3.21) in (3.18) we obtain

$$\frac{\partial\xi}{\partial w_i} = 2\sum_{l=0}^{N-1} r_{il} w_l - 2p_i, \quad \text{for } i = 0, 1, \ldots, N-1, \tag{3.22}$$

which can be expressed using matrix notation as

$$\nabla\xi = 2\mathbf{R}\mathbf{w} - 2\mathbf{p}. \tag{3.23}$$

Letting $\nabla\xi = \mathbf{0}$ gives the following equation from which the optimum set of Wiener filter tap weights can be obtained:

$$\mathbf{R}\mathbf{w}_o = \mathbf{p}. \tag{3.24}$$

Note that we have added the subscript 'o' to $\mathbf{w}$ to emphasize that it is the optimum tap-weight vector. Equation (3.24), which is known as the *Wiener–Hopf* equation, has the following solution:

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p}, \tag{3.25}$$

assuming that $\mathbf{R}$ has an inverse.

Replacing $\mathbf{w}$ by $\mathbf{w}_o$ and $\mathbf{R}\mathbf{w}_o$ by $\mathbf{p}$ in (3.12) we obtain

$$\begin{aligned}\xi_{\min} &= \mathrm{E}[d^2(n)] - \mathbf{w}_o^{\mathrm{T}}\mathbf{p} \\ &= \mathrm{E}[d^2(n)] - \mathbf{w}_o^{\mathrm{T}}\mathbf{R}\mathbf{w}_o.\end{aligned} \tag{3.26}$$

This is the *minimum mean-square error* that can be achieved by the transversal Wiener filter $W(z)$ and is obtained when its tap weights are chosen according to the optimum solution given by (3.25).

For our later reference, we may also note that by substituting (3.25) into (3.26) we obtain

$$\xi_{\min} = \mathrm{E}[d^2(n)] - \mathbf{p}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{p}. \tag{3.27}$$

Example 3.1

Consider the modelling problem shown in Figure 3.3. The plant is a two-tap filter with an additive noise, $v(n)$, added to its output. A two-tap Wiener filter with tap weights w_0 and w_1 is used to model the plant parameters. The same input is applied to both the plant and Wiener filter. The

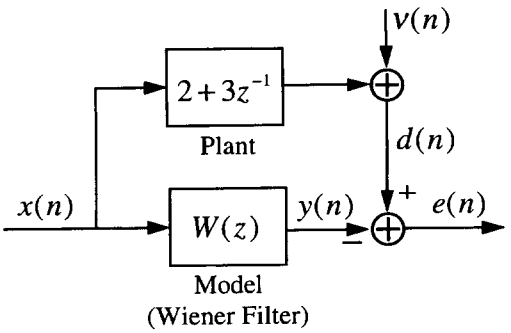


Figure 3.3 A modelling problem

input, $x(n)$, is a stationary white process with a variance of unity. The additive noise, $\nu(n)$, is zero-mean and uncorrelated with $x(n)$, and its variance is $\sigma_v^2 = 0.1$. We want to compute the optimum values of w_0 and w_1 that minimize $\mathrm{E}[e^2(n)]$.

We need to compute $\mathbf{R}$ and $\mathbf{p}$ to obtain the optimum values of w_0 and w_1 that minimize $\mathrm{E}[e^2(n)]$. For this example, we get

$$\mathbf{R} = \begin{bmatrix} \mathrm{E}[x^2(n)] & \mathrm{E}[x(n)x(n-1)] \\ \mathrm{E}[x(n-1)x(n)] & \mathrm{E}[x^2(n-1)] \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{3.28}$$

This follows since $x(n)$ is white, thus $\mathrm{E}[x(n)x(n-1)] = \mathrm{E}[x(n-1)x(n)] = 0$, and also it has a variance of unity. The latter implies that $\mathrm{E}[x^2(n)] = \mathrm{E}[x^2(n-1)] = 1$.

Also, we note that $d(n) = 2x(n) + 3x(n-1) + \nu(n)$, and, thus,

$$\begin{aligned} \mathbf{p} &= \begin{bmatrix} \mathrm{E}[x(n)d(n)] \\ \mathrm{E}[x(n-1)d(n)] \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{E}[x(n)(2x(n) + 3x(n-1) + \nu(n))] \\ \mathrm{E}[x(n-1)(2x(n) + 3x(n-1) + \nu(n))] \end{bmatrix}. \end{aligned} \tag{3.29}$$

Expanding the terms under the expectation operators, and noting that $\mathrm{E}[x^2(n)] = \mathrm{E}[x^2(n-1)] = 1$ and $\mathrm{E}[x(n)x(n-1)] = \mathrm{E}[x(n)\nu(n)] = \mathrm{E}[x(n-1)\nu(n)] = 0$, we get

$$\mathbf{p} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}. \tag{3.30}$$

Similarly, we obtain

$$\begin{aligned} \mathrm{E}[d^2(n)] &= \mathrm{E}[(2x(n) + 3x(n-1) + \nu(n))^2] \\ &= 4\mathrm{E}[x^2(n)] + 9\mathrm{E}[x^2(n-1)] + \sigma_\nu^2 = 13.1. \end{aligned} \tag{3.31}$$

Substituting (3.28), (3.30) and (3.31) in (3.12) we get

$$\xi = 13.1 - 4w_0 - 6w_1 + w_0^2 + w_1^2. \tag{3.32}$$

This is a paraboloid in the three-dimensional space with the axes w_0, w_1 and ξ. Figure 3.4 shows this paraboloid. We may note that the optimum tap weights of the Wiener filter are given by (3.25), which for the present example may be written as

$$\begin{bmatrix} w_{o,0} \\ w_{o,1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}. \tag{3.33}$$

Also, from (3.26),

$$\xi_{\min} = 13.1 - [2 \;\; 3] \begin{bmatrix} 2 \\ 3 \end{bmatrix} = 0.1. \tag{3.34}$$

Clearly, the values of $w_{o,0}$, $w_{o,1}$ and $\xi_{\min}$ coincide with the minimum point in Figure 3.4.

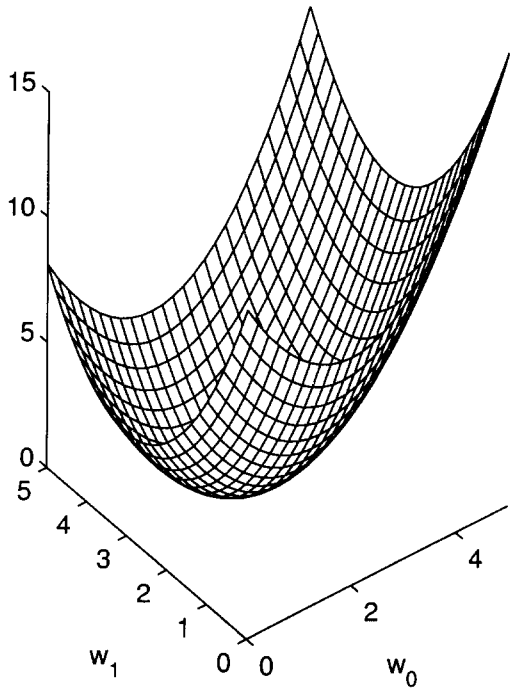


Figure 3.4 The performance surface of the modelling problem of Figure 3.3

The features of interest on the performance surface in Figure 3.4 and the results obtained in (3.33) and (3.34) and also Figure 3.4 may be understood better if we note that the right side of (3.32) may also be expressed as

$$\xi = 0.1 + (w_0 - 2)^2 + (w_1 - 3)^2. \tag{3.35}$$

Clearly, the minimum value of ξ is achieved when the last two terms on the right-hand side of (3.35) are forced to zero. This coincides with the results in (3.33) and (3.34).

3.3 Principle of Orthogonality

In this section we present an alternative approach to the design of Wiener filters. This presentation is a complement to the derivations in the previous section in the sense that the approach presented below can be considered as a simplified/shortened version of the approach in the previous section. More importantly, it leads to more insight into the concept of the Wiener filtering problem.

We start with the cost function equation (3.1), which in the case of real-valued data may be written as

$$\xi = \mathrm{E}[e^2(n)]. \tag{3.36}$$

Taking partial derivatives of ξ with respect to the filter tap weights, $\{w_i; i = 0, 1, \ldots, N-1\}$, and interchanging the derivative and expectation operators (since

these are linear operators), we obtain

$$\frac{\partial \xi}{\partial w_i} = E\left[2e(n)\frac{\partial e(n)}{\partial w_i}\right], \quad \text{for } i = 0, 1, \ldots, N-1, \tag{3.37}$$

where $e(n) = d(n) - y(n)$. Since $d(n)$ is independent of the filter tap weights, we get

$$\frac{\partial e(n)}{\partial w_i} = -\frac{\partial y(n)}{\partial w_i} = -x(n-i), \tag{3.38}$$

where the last result is obtained by replacing for $y(n)$ from (3.5). Using this result in (3.37), we obtain

$$\frac{\partial \xi}{\partial w_i} = -2\mathrm{E}[e(n)x(n-i)], \quad \text{for } i = 0, 1, \ldots, N-1. \tag{3.39}$$

From our discussion in the previous section we know that when the Wiener filter tap weights are set to their optimal values, the partial derivatives of the cost function, ξ, with respect to the filter tap weights are all zero. Hence, if $e_o(n)$ is the estimation error when the filter tap weights are set equal to their optimal values, then (3.39) becomes

$$\mathrm{E}[e_o(n)x(n-i)] = 0, \quad \text{for } i = 0, 1, \ldots, N-1. \tag{3.40}$$

This shows that at the optimal setting of the Wiener filter tap weights, the estimation error is uncorrelated with the filter tap inputs, i.e. the input samples used for estimation. This is known as the *principle of orthogonality*.

The principle of orthogonality is an elegant result of Wiener filtering that is frequently used for simple derivations of results which otherwise would seem far more difficult to derive. We will use the principle of orthogonality throughout this book for many of our derivations.

As a useful corollary to the principle of orthogonality, we note that the filter output is also uncorrelated with the estimation error when its tap weights are set to their optimal values. This may be shown as follows:

$$\begin{aligned}\mathrm{E}[e_o(n)y_o(n)] &= E\left[e_o(n)\sum_{i=0}^{N-1} w_{o,i}x(n-i)\right] \\ &= \sum_{i=0}^{N-1} w_{o,i}\mathrm{E}[e_o(n)x(n-i)],\end{aligned} \tag{3.41}$$

where $y_o(n)$ is the Wiener filter output when its tap weights are set to their optimal values. Then, using (3.40) in (3.41) we obtain

$$\mathrm{E}[e_o(n)y_o(n)] = 0. \tag{3.42}$$

We may also refer to the above result by saying that *the optimized Wiener filter output and the estimation error are orthogonal.*

The words *orthogonality* and *orthogonal* are commonly used for referring to pairs of random variables that are uncorrelated with each other. This originates from the fact that the set of all random variables with finite second moments constitutes a linear space with

an inner product. The inner product in this space is defined to be the correlation between its elements. In particular, if x and y are two elements of the linear space of random variables, then the inner product of x and y is defined as $\mathrm{E}[xy]$, when x and y are real-valued, or $\mathrm{E}[xy^*]$, in the more general case of complex-valued random variables. Then, in analogy with the Euclidian space in which the elements are vectors, the geometrical concepts such as orthogonality, projection and subspaces may also be defined for the space of random variables. The interested reader may refer to Honig and Messerschmitt (1984) for an excellent, yet simple, discussion on this topic.

Next, we use the principle of orthogonality to give an alternative derivation of the Wiener–Hopf equation of (3.24) and also the minimum mean-squared error of (3.26). We note that

$$e_{\mathrm{o}}(n) = d(n) - \sum_{l=0}^{N-1} w_{\mathrm{o},l}\, x(n-l), \tag{3.43}$$

where the $w_{\mathrm{o},l}$s are the optimum values of the Wiener filter tap weights. Substituting (3.43) in (3.40) and rearranging the results, we get

$$\sum_{l=0}^{N-1} \mathrm{E}[x(n-i)x(n-l)]w_{\mathrm{o},l} = \mathrm{E}[d(n)x(n-i)], \quad \text{for } i = 0, 1, \ldots, N-1. \tag{3.44}$$

We also note that $\mathrm{E}[x(n-i)x(n-l)] = r_{il}$ and $\mathrm{E}[d(n)x(n-i)] = p_i$. Using these in (3.44) we obtain

$$\sum_{l=0}^{N-1} r_{il} w_{\mathrm{o},l} = p_i, \quad \text{for } i = 0, 1, \ldots, N-1, \tag{3.45}$$

which is nothing but (3.24) in expanded form.

Also, we note that

$$\begin{aligned} \xi_{\min} &= \mathrm{E}[e_{\mathrm{o}}^2(n)] \\ &= \mathrm{E}[e_{\mathrm{o}}(n)(d(n) - y_{\mathrm{o}}(n))] \\ &= \mathrm{E}[e_{\mathrm{o}}(n)d(n)] - \mathrm{E}[e_{\mathrm{o}}(n)y_{\mathrm{o}}(n)] \\ &= \mathrm{E}[e_{\mathrm{o}}(n)d(n)], \end{aligned} \tag{3.46}$$

where (3.42) has been used to obtain the last equality. Now, substituting (3.43) in (3.46), we obtain

$$\begin{aligned} \xi_{\min} &= E\left[\left(d(n) - \sum_{i=0}^{N-1} w_{\mathrm{o},i}\, x(n-i)\right) d(n)\right] \\ &= \mathrm{E}[d^2(n)] - \sum_{i=0}^{N-1} w_{\mathrm{o},i} \mathrm{E}[d(n)x(n-i)] \\ &= \mathrm{E}[d^2(n)] - \sum_{i=0}^{N-1} w_{\mathrm{o},i} p_i, \end{aligned} \tag{3.47}$$

which is nothing but (3.26) in expanded form.

3.4 Normalized Performance Function

Equation (3.43) can be written as

$$d(n) = e_o(n) + y_o(n). \tag{3.48}$$

Squaring both sides of (3.48) and taking expectation, we get

$$\mathrm{E}[d^2(n)] = \mathrm{E}[e_o^2(n)] + \mathrm{E}[y_o^2(n)] + 2\mathrm{E}[e_o(n)y_o(n)]. \tag{3.49}$$

We may note that $\mathrm{E}[e_o^2(n)] = \xi_{\min}$, and the last term in (3.49) is zero because of (3.42). Thus, we obtain

$$\xi_{\min} = \mathrm{E}[d^2(n)] - \mathrm{E}[y_o^2(n)], \tag{3.50}$$

which suggests that *the minimum mean-square error at the Wiener filter output is the difference between the mean-square error of the desired output and the mean-square error of the best estimate of that at the filter output.*

It is appropriate if we define the ratio

$$\zeta = \frac{\xi}{\mathrm{E}[d^2(n)]} \tag{3.51}$$

as the normalized performance function. We may note that $\zeta = 1$ when $y(n)$ is forced to zero, i.e. when no estimation of $d(n)$ has been made. It reaches its minimum value, $\zeta_{\min}$, when the filter tap weights are chosen to achieve the minimum mean-squared error. This is given by

$$\zeta_{\min} = 1 - \frac{\mathrm{E}[y_o^2(n)]}{\mathrm{E}[d^2(n)]}. \tag{3.52}$$

Noting that $\zeta_{\min}$ cannot be negative, we find that its value remains between 0 and 1. *The value of ζ_{min} is an indication of the ability of the filter to estimate the desired output. A value of ζ_{min} close to zero is an indication of good performance of the filter, and a value of ζ_{min} close to one indicates poor performance of the filter.*

3.5 Extension to the Complex-Valued Case

There are some practical applications in which the underlying random processes are complex-valued. For instance, in data transmission the most frequently used signalling techniques are phase shift keying (PSK) and quadrature amplitude modulation (QAM) in which the baseband signal consists of two separate components which are the real and imaginary parts of a complex-valued signal. Moreover, in the case of frequency domain implementation of adaptive filters (Chapter 8) and subband adaptive filters (Chapter 9), we will be dealing with complex-valued signals, even though the original signals may be real-valued.

In this section we extend the results of the previous two sections to the case of complex-valued signals. We assume a transversal filter as in Figure 3.2. The input, $x(n)$, the desired output, $d(n)$, and the filter tap weights are all assumed to be complex variables. Then, the estimation error, $e(n)$, is also complex and we may write

$$\xi = \mathrm{E}[|e(n)|^2] = \mathrm{E}[e(n)e^*(n)], \tag{3.53}$$

where the asterisk denotes complex conjugation.

As in the real-valued case, the performance function, ξ, in the complex-valued case is also a quadratic function of filter tap weights. Similarly, to find the optimum set of the filter tap weights, we have to solve the system of equations that results from setting the partial derivatives of ξ with respect to every tap weight to zero. However, noting that the filter tap weights are complex variables, the conventional definition of derivative with respect to an independent variable is not applicable to the present case. In fact, we note that each tap weight, in the present case, consists of two independent variables that make the real and imaginary parts of that. Thus, the partial derivatives with respect to these two independent variables have to be performed separately and the results have to be set to zero to obtain the optimum tap weights of the Wiener filter. In particular, to obtain the optimum set of filter tap weights, the following set of equations have to be solved simultaneously.

$$\frac{\partial \xi}{\partial w_{i,\mathrm{R}}} = 0 \quad \text{and} \quad \frac{\partial \xi}{\partial w_{i,\mathrm{I}}} = 0, \quad \text{for } i = 0, 1, \ldots, N-1, \tag{3.54}$$

where $w_{i,\mathrm{R}}$ and $w_{i,\mathrm{I}}$ denote the real and imaginary parts of w_i, respectively. To write (3.54) in a more compact form, we note that ξ, $w_{i,\mathrm{R}}$ and $w_{i,\mathrm{I}}$ are all real. This implies that the partial derivatives in (3.54) are also all real and thus the pairs of equations in (3.54) may be combined to obtain

$$\frac{\partial \xi}{\partial w_{i,\mathrm{R}}} + j\frac{\partial \xi}{\partial w_{i,\mathrm{I}}} = 0, \quad \text{for } i = 0, 1, \ldots, N-1, \tag{3.55}$$

where $j = \sqrt{-1}$. This, in turn, suggests the following definition of the gradient of a function with respect to a complex variable $w = w_{\mathrm{R}} + jw_{\mathrm{I}}$:

$$\nabla_w^{\mathrm{C}} \triangleq \frac{\partial}{\partial w_{\mathrm{R}}} + j\frac{\partial}{\partial w_{\mathrm{I}}}. \tag{3.56}$$

We note that when ξ is a real function of w_{R} and w_{I}, the real and imaginary parts of $\nabla_w^{\mathrm{C}}\xi$ are, respectively, equal to $\partial\xi/\partial w_{\mathrm{R}}$ and $\partial\xi/\partial w_{\mathrm{I}}$, and in that case $\nabla_w^{\mathrm{C}}\xi = 0$ implies that $\partial\xi/\partial w_{\mathrm{R}} = \partial\xi/\partial w_{\mathrm{I}} = 0$. It is in this context that we can say (3.54) and (3.55) are equivalent. This would not be true, in general, if ξ was complex (see Problem 3.5).

With the above background, we may now continue with the derivation of the principle of orthogonality and its subsequent results, for the case of complex-valued signals. From (3.53) we note that

$$\nabla_{w_i}^{\mathrm{C}}\xi = E[e(n)\nabla_{w_i}^{\mathrm{C}}e^*(n) + e^*(n)\nabla_{w_i}^{\mathrm{C}}e(n)]. \tag{3.57}$$

Noting that

$$e(n) = d(n) - \sum_{k=0}^{N-1} w_k x(n-k), \tag{3.58}$$

we obtain

$$\nabla^{\mathrm{C}}_{w_i} e(n) = -x(n-i)\nabla^{\mathrm{C}}_{w_i} w_i \tag{3.59}$$

and

$$\nabla^{\mathrm{C}}_{w_i} e^*(n) = -x^*(n-i)\nabla^{\mathrm{C}}_{w_i} w_i^*. \tag{3.60}$$

Applying the definition (3.56) we obtain

$$\nabla^{\mathrm{C}}_{w_i} w_i = \frac{\partial w_i}{\partial w_{i,\mathrm{R}}} + j\frac{\partial w_i}{\partial w_{i,\mathrm{I}}} = 1 + j(j) = 1 - 1 = 0 \tag{3.61}$$

and

$$\nabla^{\mathrm{C}}_{w_i} w_i^* = \frac{\partial w_i^*}{\partial w_{i,\mathrm{R}}} + j\frac{\partial w_i^*}{\partial w_{i,\mathrm{I}}} = 1 + j(-j) = 1 + 1 = 2. \tag{3.62}$$

Substituting (3.61) and (3.62) into (3.59) and (3.60), respectively, and the results into (3.57), we obtain

$$\nabla^{\mathrm{C}}_{w_i} \xi = -2\mathrm{E}[e(n)x^*(n-i)]. \tag{3.63}$$

When the Wiener filter tap weights are set to their optimal values, $\nabla^{\mathrm{C}}_{w_i} \xi = 0$. This gives

$$\mathrm{E}[e_{\mathrm{o}}(n)x^*(n-i)] = 0, \quad \text{for } i = 0, 1, \ldots, N-1, \tag{3.64}$$

where $e_{\mathrm{o}}(n)$ is the optimum estimation error. The set of equations (3.64) represents *the principle of orthogonality for the case of complex-valued signals.*

To proceed with the derivation of the Wiener–Hopf equation, we define the input and tap-weight vectors of the filter as

$$\mathbf{x}(n) \triangleq [x(n) \;\; x(n-1) \;\; \ldots \;\; x(n-N+1)]^{\mathrm{T}} \tag{3.65}$$

and

$$\mathbf{w} \triangleq [w_0^* \;\; w_1^* \;\; \ldots \;\; w_{N-1}^*]^{\mathrm{T}} \tag{3.66}$$

respectively, where the asterisk and T denote complex conjugation and transpose, respectively. Note that the elements of the column vector $\mathbf{w}$ are complex conjugates of

the actual tap weights of the filter, while conjugation is not applied to samples of the input in $\mathbf{x}(n)$. Also, for future reference we may write

$$\mathbf{x}(n) = [x^*(n) \;\; x^*(n-1) \;\; \ldots \;\; x^*(n-N+1)]^{\mathrm{H}}, \tag{3.67}$$

and

$$\mathbf{w} = [w_0 \;\; w_1 \;\; \ldots \;\; w_{N-1}]^{\mathrm{H}} \tag{3.68}$$

where the superscript H denotes complex-conjugate transpose or Hermitian.

The set of equations (3.64) may also be written as

$$\mathrm{E}[e_{\mathrm{o}}^*(n)x(n-i)] = 0, \quad \text{for } i = 0, 1, \ldots, N-1. \tag{3.69}$$

Using definition (3.65), these may be packed together as

$$\mathrm{E}[e_{\mathrm{o}}^*(n)\mathbf{x}(n)] = \mathbf{0}. \tag{3.70}$$

Also, we note that

$$e_{\mathrm{o}}(n) = d(n) - \mathbf{w}_{\mathrm{o}}^{\mathrm{H}}\mathbf{x}(n), \tag{3.71}$$

where $\mathbf{w}_{\mathrm{o}}$ is the optimum tap-weight vector of the Wiener filter.

Replacing (3.71) in (3.70), we obtain

$$\mathrm{E}[\mathbf{x}(n)(d^*(n) - \mathbf{x}^{\mathrm{H}}(n)\mathbf{w}_{\mathrm{o}})] = \mathbf{0}. \tag{3.72}$$

Rearranging (3.72) we get

$$\mathbf{R}\mathbf{w}_{\mathrm{o}} = \mathbf{p}, \tag{3.73}$$

where $\mathbf{R} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{H}}(n)]$ and $\mathbf{p} = \mathrm{E}[\mathbf{x}(n)d^*(n)]$. This is *the Wiener–Hopf equation for the case of complex-valued signals.*

Also, following the same derivations as (3.26) and (3.47), for the present case we obtain

$$\begin{aligned} \xi_{\min} &= \mathrm{E}[|d(n)|^2] - \mathbf{w}_{\mathrm{o}}^{\mathrm{H}}\mathbf{p} \\ &= \mathrm{E}[|d(n)|^2] - \mathbf{w}_{\mathrm{o}}^{\mathrm{H}}\mathbf{R}\mathbf{w}_{\mathrm{o}}. \end{aligned} \tag{3.74}$$

3.6 Unconstrained Wiener Filters

The developments in the previous three sections put some constraints on the Wiener filter by assuming that it is causal and the duration of its impulse response is limited. In this section we remove such constraints and let the Wiener filter impulse response, w_i, extend from $i = -\infty$ to $i = +\infty$, and derive equations for the filter performance function and its optimal system function. Such developments are very instructive for understanding many

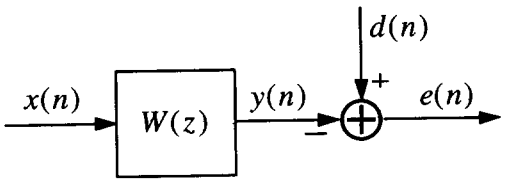


Figure 3.5 Block diagram of a Wiener filter

of the important aspects of the Wiener filter, which otherwise could not be easily understood.

Consider the Wiener filter shown in Figure 3.1, and repeated here in Figure 3.5, for convenience. We assume that the filter $W(z)$ may be non-causal and/or IIR. To keep the derivations in this section as simple as possible and also to concentrate more on the concepts, we consider only the case in which the underlying signals and system parameters are real-valued. Moreover, we assume that the complex variable z remains on the unit circle, i.e. $|z| = 1$. This implies that $z^* = z^{-1}$. Also, for future reference, we note that when the coefficients of a system function, such as $W(z)$, are real-valued, $W^*(1/z^*) = W(z^{-1})$ for all values of z, and $W(z^{-1}) = W^*(z)$, when $|z| = 1$.

The derivations that follow in this section depend highly on the results developed in Section 2.4.4 of the previous chapter. The reader is encouraged to review the latter section before continuing with the rest of this section.

3.6.1 Performance function

Recall that the Wiener filter performance function is defined as

$$\xi = \mathrm{E}[e^2(n)].$$

Substituting $e(n)$ by $d(n) - y(n)$ and expanding, we get

$$\xi = \mathrm{E}[d^2(n)] + \mathrm{E}[y^2(n)] - 2\mathrm{E}[y(n)d(n)]. \tag{3.75}$$

In terms of autocorrelation and cross-correlation functions (see Chapter 2), we may write

$$\xi = \phi_{dd}(0) + \phi_{yy}(0) - 2\phi_{yd}(0). \tag{3.76}$$

Replacing the last two terms on the right-hand side of (3.76) with their corresponding inverse z-transform relations, we obtain

$$\xi = \phi_{dd}(0) + \frac{1}{2\pi j}\oint_C \Phi_{yy}(z)\frac{\mathrm{d}z}{z} - 2 \times \frac{1}{2\pi j}\oint_C \Phi_{yd}(z)\frac{\mathrm{d}z}{z}. \tag{3.77}$$

Also, from our discussion in Chapter 2, Section 2.4.4, we recall that when $x(n)$ and $y(n)$ are related, as in Figure 3.5, for an arbitrary sequence $d(n)$, $\Phi_{yd}(z) = W(z)\Phi_{xd}(z)$. Also, if z is selected to be on the unit circle in the z-plane, then $\Phi_{yy}(z) = |W(z)|^2\Phi_{xx}(z)$,

$|W(z)|^2 = W(z)W^*(z)$ and $W^*(z) = W(z^{-1})$. Using these in (3.77), we obtain

$$\xi = \phi_{dd}(0) + \frac{1}{2\pi j}\oint_C |W(z)|^2 \Phi_{xx}(z)\frac{dz}{z} - 2 \times \frac{1}{2\pi j}\oint_C W(z)\Phi_{xd}(z)\frac{dz}{z}$$

$$= \phi_{dd}(0) + \frac{1}{2\pi j}\oint_C \left[W^*(z)\Phi_{xx}(z) - 2\Phi_{xd}(z)\right] W(z)\frac{dz}{z}, \tag{3.78}$$

where the contour of integration, C, is the unit circle. *This is the performance function for a Wiener filter with the system function $W(z)$, in its most general form. It covers IIR and FIR, as well as causal and non-causal filters.* The following examples show some of the flexibilities of (3.78).

Example 3.2

Consider the case where the Wiener filter is an N-tap FIR filter with the system function

$$W(z) = \sum_{l=0}^{N-1} w_l z^{-l}. \tag{3.79}$$

This is the case that we studied in Section 3.2.

Using (3.79) in the first line of (3.78), we obtain

$$\xi = \phi_{dd}(0) + \frac{1}{2\pi j}\oint_C \left(\sum_{l=0}^{N-1} w_l z^{-l}\right)\left(\sum_{m=0}^{N-1} w_m z^m\right)\Phi_{xx}(z)\frac{dz}{z}$$

$$- 2 \times \frac{1}{2\pi j}\oint_C \left(\sum_{l=0}^{N-1} w_l z^{-l}\right)\Phi_{xd}(z)\frac{dz}{z}. \tag{3.80}$$

Interchanging the order of the integrations and summations, (3.80) is simplified to

$$\xi = \phi_{dd}(0) + \sum_{l=0}^{N-1}\sum_{m=0}^{N-1} w_l w_m \frac{1}{2\pi j}\oint_C \Phi_{xx}(z) z^{m-l-1} dz - 2\sum_{l=0}^{N-1} w_l \frac{1}{2\pi j}\oint_C \Phi_{xd}(z) z^{-l-1} dz. \tag{3.81}$$

Using the inverse z-transform relation, this gives

$$\xi = \phi_{dd}(0) + \sum_{l=0}^{N-1}\sum_{m=0}^{N-1} w_l w_m \phi_{xx}(m-l) - 2\sum_{l=0}^{N-1} w_l \phi_{xd}(-l). \tag{3.82}$$

Now, using the notations $\phi_{dd}(0) = \mathrm{E}[d^2(n)]$, $\phi_{xd}(-l) = p_l$ and $\phi_{xx}(m-l) = \phi_{xx}(l-m) = r_{lm}$, we see that the performance function given by (3.82) is the same as what we derived earlier in (3.16).

Example 3.3

Consider the modelling problem depicted in Figure 3.6, where a plant $G(z)$ is being modelled by a single-pole, single-zero, Wiener filter

$$W(z) = \frac{1 - w_0 z^{-1}}{1 - w_1 z^{-1}}. \tag{3.83}$$

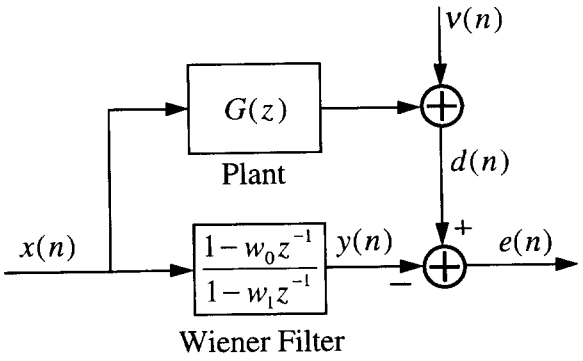


Figure 3.6 A modelling problem with an IIR model

To keep our discussion simple, we assume all the involved signals and system parameters are real-valued. The input sequence, $x(n)$, is assumed to be a white process with zero mean and a variance of unity, and uncorrelated with the additive noise $v(n)$. This implies that

$$\Phi_{xx}(z) = 1 \quad \text{and} \quad \Phi_{vx}(z) = 0. \tag{3.84}$$

We note that $d(n)$ is the noise-corrupted output of the plant, $G(z)$, when it is excited with the input $x(n)$. Then, using the relationship (2.75) of the previous chapter, and noting that all the signals here are real-valued, we get

$$\Phi_{xd}(z) = G(z^{-1})\Phi_{xx}(z). \tag{3.85}$$

Using this in (3.78) we obtain

$$\xi = \phi_{dd}(0) + \frac{1}{2\pi j}\oint_C \frac{1 - w_0 z^{-1}}{1 - w_1 z^{-1}} \cdot \frac{1 - w_0 z}{1 - w_1 z} \cdot \frac{dz}{z} - 2 \times \frac{1}{2\pi j}\oint_C \frac{1 - w_0 z^{-1}}{1 - w_1 z^{-1}} G(z^{-1}) \frac{dz}{z}. \tag{3.86}$$

Using the residue theorem to calculate the above integrals, and assuming that $G(z^{-1})$ has no pole inside the unit circle, we get

$$\xi = \phi_{dd}(0) + \frac{w_1 - w_0}{w_1} \cdot \frac{1 - w_0 w_1}{1 - w_1^2} + \frac{w_0}{w_1} - 2\left[\frac{w_1 - w_0}{w_1} G(w_1^{-1}) + \frac{w_0}{w_1} G(\infty)\right]. \tag{3.87}$$

This is the performance function of the IIR filter shown in Figure 3.6. We note that although we have selected a very simple example, the resulting performance function is a complicated one. It is clear that a performance function such as (3.87) – or more complicated ones that would result for higher order filters are difficult to handle. In particular, we may find that there can be many local minima, and searching for the global minimum of the performance function may not be a trivial task. This, when compared with the nicely shaped quadratic performance function of FIR filters, makes it clear why most of the attention in adaptive filters has been devoted to the transversal structure.

3.6.2 Optimum transfer function

We now derive an equation for the optimum transfer function of unconstrained Wiener filters, i.e. when the filter impulse response is allowed to extend from time $n = -\infty$ to

$n = +\infty$. We use the principle of orthogonality for this purpose. Since the filter impulse response stretches from time $n = -\infty$ to $n = +\infty$, the principle of orthogonality for real-valued signals suggests

$$\mathrm{E}[e_{\mathrm{o}}(n)x(n-i)] = 0, \quad \text{for } i = \ldots, -2, -1, 0, 1, 2, \ldots, \tag{3.88}$$

where $e_{\mathrm{o}}(n)$ is the optimum estimation error and is given by

$$e_{\mathrm{o}}(n) = d(n) - \sum_{l=-\infty}^{\infty} w_{\mathrm{o},l}x(n-l). \tag{3.89}$$

Here, the $w_{\mathrm{o},l}$s are the samples of the optimized Wiener filter impulse response.

Substituting (3.89) in (3.88) and rearranging the result, we obtain

$$\sum_{l=-\infty}^{\infty} w_{\mathrm{o},l}\mathrm{E}[x(n-l)x(n-i)] = \mathrm{E}[d(n)x(n-i)]. \tag{3.90}$$

We may also note that $\mathrm{E}[x(n-l)x(n-i)] = \phi_{xx}(i-l)$ and $\mathrm{E}[d(n)x(n-i)] = \phi_{dx}(i)$. Using these in (3.90) we get

$$\sum_{l=-\infty}^{\infty} w_{\mathrm{o},l}\phi_{xx}(i-l) = \phi_{dx}(i), \quad \text{for } i = \ldots, -2, -1, 0, 1, 2, \ldots. \tag{3.91}$$

Noting that (3.91) holds for all values of i, we may take z-transforms on both sides to obtain

$$\Phi_{xx}(z)W_{\mathrm{o}}(z) = \Phi_{dx}(z). \tag{3.92}$$

This is referred to as *the Wiener–Hopf equation for the unconstrained Wiener filtering problem*. The optimum unconstrained Wiener filter is given by

$$W_{\mathrm{o}}(z) = \frac{\Phi_{dx}(z)}{\Phi_{xx}(z)}. \tag{3.93}$$

Replacing z by $\mathrm{e}^{j\omega}$ in (3.93) we obtain

$$W_{\mathrm{o}}(\mathrm{e}^{j\omega}) = \frac{\Phi_{dx}(\mathrm{e}^{j\omega})}{\Phi_{xx}(\mathrm{e}^{j\omega})}. \tag{3.94}$$

This result has an interesting interpretation. It shows that *the frequency response of the optimal Wiener filter, for a particular frequency, say $\omega = \omega_i$, is determined by the ratio of the cross-power spectral density of $d(n)$ and $x(n)$, to the power spectral density of $x(n)$, at $\omega = \omega_i$*. This, in turn, may be obtained through a sequence of filtering and averaging steps, as depicted in Figure 3.7. The sequences $x(n)$ and $d(n)$ are first filtered by two identical narrow-band filters, centred at $\omega = \omega_i$. To retain the phase information of the underlying signals, these filters are designed to pick up signals from the positive side of

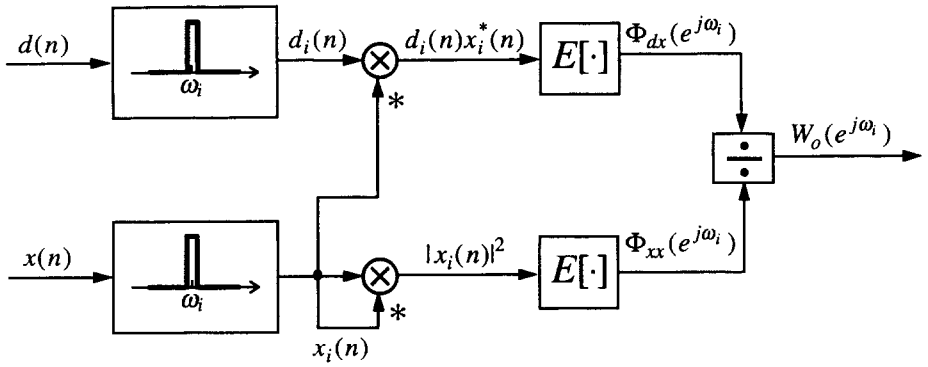


Figure 3.7 Procedure for calculating the transfer function of a Wiener filter through a sequence of filtering and averaging

the frequency axis only. The signal spectra belonging to negative frequencies are completely rejected. As a result, the filtered signals, $d_i(n)$ and $x_i(n)$, are both complex-valued. The cross-correlation of $d_i(n)$ and $x_i(n)$ with zero lag, i.e. $\mathrm{E}[d_i(n)x_i^*(n)]$, gives a quantity proportional to $\Phi_{dx}(\mathrm{e}^{j\omega_i})$, and the average energy of $x_i(n)$ gives a quantity proportional to $\Phi_{xx}(\mathrm{e}^{j\omega_i})$ – see Papoulis (1991). The ratio of these two quantities gives $W_{\mathrm{o}}(\mathrm{e}^{j\omega_i})$. This interpretation becomes more interesting if we note that $W_{\mathrm{o}}(\mathrm{e}^{j\omega_i})$ is also the optimum tap weight of a single-tap Wiener filter whose input and desired output are the complex-valued random processes $x_i(n)$ and $d_i(n)$, respectively; see Problem P3.12.

The minimum mean-squared estimation error for the unconstrained Wiener filtering case can be obtained by substituting (3.93) in (3.78). For this, we first note that when $|z| = 1$,

$$\begin{aligned} |W_{\mathrm{o}}(z)|^2 &= W_{\mathrm{o}}(z)W_{\mathrm{o}}^*(z) \\ &= W_{\mathrm{o}}(z)\frac{\Phi_{dx}^*(z)}{\Phi_{xx}^*(z)} \\ &= W_{\mathrm{o}}(z)\frac{\Phi_{xd}(z)}{\Phi_{xx}(z)} \end{aligned} \tag{3.95}$$

since, on the unit circle, $\Phi_{dx}^*(z) = \Phi_{xd}(z)$ and $\Phi_{xx}^*(z) = \Phi_{xx}(z)$. Using this result in (3.78), we get

$$\xi_{\min} = \phi_{dd}(0) - \frac{1}{2\pi j}\oint_C W_{\mathrm{o}}(z)\Phi_{xd}(z)\frac{\mathrm{d}z}{z}. \tag{3.96}$$

This may be considered as a dual of the previous derivations in (3.26), (3.27) and (3.74); see Problem P3.13.

Replacing z by $\mathrm{e}^{j\omega}$ in (3.96), we obtain

$$\xi_{\min} = \phi_{dd}(0) - \frac{1}{2\pi}\int_{-\pi}^{\pi} W_{\mathrm{o}}(\mathrm{e}^{j\omega})\Phi_{xd}(\mathrm{e}^{j\omega})\,\mathrm{d}\omega. \tag{3.97}$$

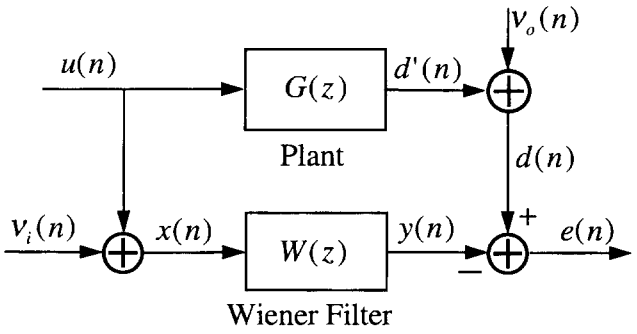


Figure 3.8 Block diagram of a modelling problem

3.6.3 Modelling

In this and the subsequent two subsections, we discuss three specific applications of Wiener filters, namely modelling, inverse modelling, and noise cancellation. These cover most of the cases that we encounter in adaptive filtering. Our aim, in these presentations, is to highlight some of the important features of Wiener filters when applied to various applications of adaptive signal processing.

Consider the modelling problem depicted in Figure 3.8. An estimate of the model of a plant $G(z)$ is to be obtained by the Wiener filter $W(z)$. The plant input $u(n)$, contaminated with an additive noise $\nu_i(n)$, is available as the Wiener filter input. The noise sequence $\nu_i(n)$ may be thought of as introduced by a transducer that is used to get samples of the plant input. There is also an additive noise $\nu_o(n)$ at the plant output. The sequences $u(n)$, $\nu_i(n)$ and $\nu_o(n)$ are assumed to be stationary, zero-mean and uncorrelated with one another.

We note that, for the present problem, the Wiener filter input and its desired output are, respectively,

$$x(n) = u(n) + \nu_i(n) \tag{3.98}$$

and

$$d(n) = g_n * u(n) + \nu_o(n), \tag{3.99}$$

where the g_ns are the samples of the plant impulse response, and the asterisk denotes convolution.

We use (3.93) to obtain the optimum transfer function of the Wiener filter, $W_o(z)$. For this, we should first find $\Phi_{xx}(z)$ and $\Phi_{dx}(z)$. We note that

$$\begin{aligned}\phi_{xx}(k) &= \mathrm{E}[x(n)x(n-k)] \\ &= \mathrm{E}[(u(n) + \nu_i(n))(u(n-k) + \nu_i(n-k))] \\ &= \mathrm{E}[u(n)u(n-k)] + \mathrm{E}[u(n)\nu_i(n-k)] \\ &\quad + \mathrm{E}[\nu_i(n)u(n-k)] + \mathrm{E}[\nu_i(n)\nu_i(n-k)].\end{aligned} \tag{3.100}$$

Since $u(n)$ and $\nu_i(n)$ are uncorrelated with each other, the second and third terms on the right-hand side of (3.100) are zero. Thus, we obtain

$$\phi_{xx}(k) = \phi_{uu}(k) + \phi_{\nu_i\nu_i}(k). \tag{3.101}$$

Taking z-transforms on both sides of (3.101), we get

$$\Phi_{xx}(z) = \Phi_{uu}(z) + \Phi_{\nu_i\nu_i}(z). \tag{3.102}$$

To find $\Phi_{dx}(z)$, we note that only $u(n)$ is common to $x(n)$ and $d(n)$, and the signals $u(n)$, $\nu_i(n)$ and $\nu_o(n)$ are uncorrelated with one another. Considering these, and following a procedure similar to the one used to arrive at (3.102), one can show that

$$\Phi_{dx}(z) = \Phi_{d'u}(z), \tag{3.103}$$

where $d'(n)$ is the plant output when the additive noise $\nu_o(n)$ is excluded from that. Moreover, from our discussions in the previous chapter, we have

$$\Phi_{d'u}(z) = G(z)\Phi_{uu}(z). \tag{3.104}$$

Thus,

$$\Phi_{dx}(z) = G(z)\Phi_{uu}(z). \tag{3.105}$$

Using (3.105) and (3.102) in (3.93), we obtain

$$W_o(z) = \frac{\Phi_{uu}(z)}{\Phi_{uu}(z) + \Phi_{\nu_i\nu_i}(z)} G(z). \tag{3.106}$$

We note that $W_o(z)$ is equal to $G(z)$, only when $\Phi_{\nu_i\nu_i}(z)$ is equal to zero. That is, when $\nu_i(n)$ is zero for all values of n.

It is also instructive to replace z by $e^{j\omega}$ in (3.106). This gives

$$W_o(e^{j\omega}) = \frac{\Phi_{uu}(e^{j\omega})}{\Phi_{uu}(e^{j\omega}) + \Phi_{\nu_i\nu_i}(e^{j\omega})} G(e^{j\omega}). \tag{3.107}$$

This result has the following interpretation. *Matching between the unconstrained Wiener filter and the plant frequency response at any particular frequency, ω, depends on the signal-to-noise power spectral density ratio $\Phi_{uu}(e^{j\omega})/\Phi_{\nu_i\nu_i}(e^{j\omega})$. Perfect matching is achieved when this ratio is infinity (i.e. when $\Phi_{\nu_i\nu_i}(e^{j\omega}) = 0$), and the mismatch between the plant and its model increases as $\Phi_{uu}(e^{j\omega})/\Phi_{\nu_i\nu_i}(e^{j\omega})$ decreases.* Note that $\Phi_{uu}(e^{j\omega})$ and $\Phi_{\nu_i\nu_i}(e^{j\omega})$ are power spectral density functions, and thus are real and non-negative.

We may also define

$$K(e^{j\omega}) \triangleq \frac{\Phi_{uu}(e^{j\omega})}{\Phi_{uu}(e^{j\omega}) + \Phi_{\nu_i\nu_i}(e^{j\omega})} \tag{3.108}$$

and note that $K(e^{j\omega})$ is real and varies in the range 0 to 1, since the power spectral density functions $\Phi_{uu}(e^{j\omega})$ and $\Phi_{\nu_i\nu_i}(e^{j\omega})$ are both real and non-negative. Further, to prevent ambiguity of the above ratio, we assume that for all values of ω, $\Phi_{uu}(e^{j\omega})$ and $\Phi_{\nu_i\nu_i}(e^{j\omega})$ are never simultaneously equal to zero. Using this, we obtain

$$W_o(e^{j\omega}) = K(e^{j\omega})G(e^{j\omega}). \tag{3.109}$$

An expression for the minimum mean-square error of the modelling problem is obtained by replacing (3.105) and (3.109) in (3.97). This gives

$$\xi_{\min} = \phi_{dd}(0) - \frac{1}{2\pi}\int_{-\pi}^{\pi} K(e^{j\omega})\Phi_{uu}(e^{j\omega})|G(e^{j\omega})|^2\,d\omega. \tag{3.110}$$

We may also note that $d'(n)$ and $\nu_o(n)$ are uncorrelated, and thus

$$\phi_{dd}(0) = \phi_{\nu_o\nu_o}(0) + \phi_{d'd'}(0). \tag{3.111}$$

Also,

$$\phi_{d'd'}(0) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_{uu}(e^{j\omega})|G(e^{j\omega})|^2\,d\omega. \tag{3.112}$$

Substituting (3.111) and (3.112) into (3.110) we obtain

$$\xi_{\min} = \phi_{\nu_o\nu_o}(0) + \frac{1}{2\pi}\int_{-\pi}^{\pi}(1 - K(e^{j\omega}))\Phi_{uu}(e^{j\omega})|G(e^{j\omega})|^2\,d\omega. \tag{3.113}$$

We note that the minimum mean-square error of the estimation error consists of two distinct components. The first one comes directly from the additive noise, $\nu_o(n)$, at the plant output. The Wiener filter will not be able to reduce this component since $\nu_o(n)$ is uncorrelated with its input $x(n)$. The second component arises due to the input noise, $\nu_i(n)$, which, in turn, results in some mismatch between $G(z)$ and $W_o(z)$. Thus, the best performance that one can expect from the optimum unconstrained Wiener filter is $\xi_{\min} = \phi_{\nu_o\nu_o}(0)$ and this happens when the input noise $\nu_i(n)$ is absent.

Another very important and useful concept that can be understood based on the above theoretical exercise is the *principle of correlation cancellation*. We remarked above that the Wiener filter cannot do anything to reduce the contribution $\phi_{\nu_o\nu_o}(0)$ from the total mean-square error. This is because the input $x(n)$ of the Wiener filter is uncorrelated with the output noise $\nu_o(n)$ and hence the filter tries to match its output $y(n)$ with the plant output $d'(n)$ without bothering about $\nu_o(n)$. In other words, the Wiener filter attempts to estimate that part of the target signal $d(n)$ that is correlated with its own input $x(n)$ (i.e. $d'(n)$) and leave the remaining part of $d(n)$ (i.e. $\nu_o(n)$) unaffected. This is known as the *principle of correlation cancellation*. However, as noted above, perfect cancellation of the correlated part $d'(n)$ from $d(n)$ will be possible when the input noise $\nu_i(n)$ is absent.

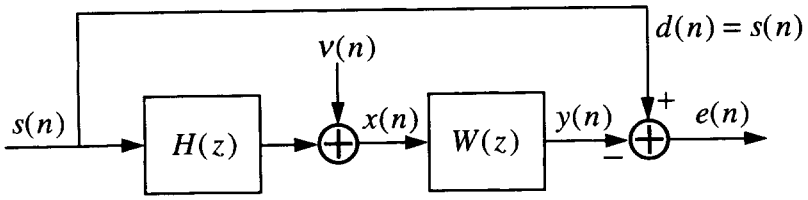


Figure 3.9 Channel equalization

3.6.4 Inverse modelling

Inverse modelling has applications in both communications and control. However, most of the theory of inverse modelling has been developed in the context of channel equalization. We also concentrate on the latter. Figure 3.9 depicts a channel equalization scenario. The data samples, $s(n)$, are transmitted through a communication channel with the system function $H(z)$. The received signal at the channel output is contaminated with an additive noise $\nu(n)$, which is assumed to be uncorrelated with the data samples, $s(n)$. An equalizer, $W(z)$, is used to process the received noisy signal samples, $x(n)$, to recover the original data samples, $s(n)$.

When the additive noise at the channel output is absent, the equalizer has the following trivial solution:

$$W_o(z) = \frac{1}{H(z)}. \tag{3.114}$$

In the absence of channel noise, this results in perfect recovery of the original data samples, as $W_o(z)H(z) = 1$. This implies that $y(n) = s(n)$, and thus $e(n) = 0$, for all n. This, clearly, is the optimum solution, as it results in a zero mean-square error which of course is the minimum, since mean-square error is a non-negative quantity. The following example gives a better view of the problem.

Example 3.4

Consider a channel with

$$H(z) = -0.4 + z^{-1} - 0.4z^{-2}. \tag{3.115}$$

Also, assume that the channel noise, $\nu(n)$, is zero, for all n. The channel output then is obtained by convolving the input data sequence, $s(n)$, with the channel impulse response, h_n, which consists of three non-zero samples $h_0 = -0.4$, $h_1 = 1$ and $h_2 = -0.4$. This gives

$$x(n) = -0.4s(n) + s(n-1) - 0.4s(n-2) \tag{3.116}$$

in the absence of channel noise.

We note that each sample of $x(n)$ is made from a mixture of three successive samples of the original data. This is called intersymbol interference (ISI) and should be compensated or cancelled for correct detection of transmitted data. For this purpose, we may use an equalizer with the

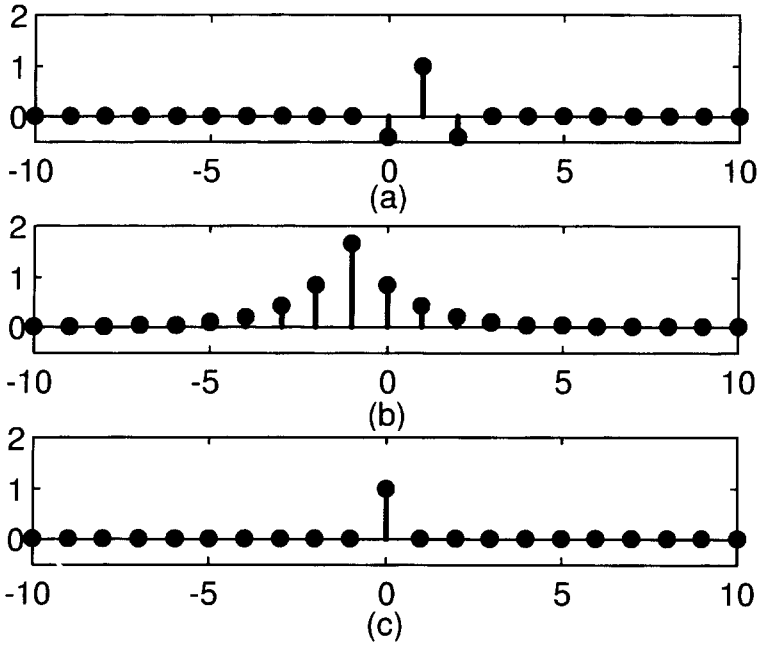


Figure 3.10 Impulse response of (a) channel response, (b) equalizer response, and (c) cascade of channel and equalizer

system function (see (3.114))

$$W_o(z) = \frac{1}{H(z)} = \frac{1}{-0.4 + z^{-1} - 0.4z^{-2}}. \tag{3.117}$$

Factorizing the denominator of $W_o(z)$ and rearranging, we get

$$W_o(z) = \frac{-2.5}{(1 - 0.5z^{-1})(1 - 2z^{-1})}. \tag{3.118}$$

This is a system function with one pole inside and one pole outside the unit circle. With reference to our discussions in the previous chapter, we recall that (3.118) will correspond to a stable time-invariant system, if the region of convergence of $W_o(z)$ includes the unit circle. Considering this and finding the inverse z-transform of $W_o(z)$, we obtain (see Chapter 2, Section 2.3)

$$w_{o,i} = \begin{cases} \frac{10}{3} \times 2^i, & i < 0, \\ \frac{2.5}{3} \times 0.5^i, & i \geq 0. \end{cases} \tag{3.119}$$

To obtain this result, we have noted that $W_o(z)$ of (3.118) is similar to $H(z)$ of (2.30), except for the factor -2.5 in (3.118). Figures 3.10(a), (b) and (c) show the samples of the impulse responses of the channel, equalizer, and their convolution, respectively. Existence of ISI at the channel output, as noted above, is due to more than one (here, three) non-zero samples in the channel impulse response. This is observed in Figure 3.10(a). Figure 3.10(c) shows that the ISI is completely removed after passing the received signal through the equalizer.

When the channel noise, $\nu(n)$, is non-zero, the solution provided by (3.114) may not be optimal. The channel noise also passes through the equalizer and may be greatly enhanced in the frequency bands where $H(e^{j\omega})$ is small. In this situation a compromise has to be made between the cancellation of ISI and noise enhancement. As we show below, the Wiener filter achieves this trade-off in an effective way.

To derive an equation for $W_o(z)$ when the channel noise is non-zero, we use (3.93). We note that

$$x(n) = h_n * s(n) + \nu(n) \tag{3.120}$$

and

$$d(n) = s(n), \tag{3.121}$$

where h_n is the impulse response of the channel, $H(z)$.

Noting that $s(n)$ and $\nu(n)$ are uncorrelated and using the results of Section 2.4.4, we obtain from (3.120)

$$\Phi_{xx}(z) = \Phi_{ss}(z)|H(z)|^2 + \Phi_{\nu\nu}(z). \tag{3.122}$$

Also, from (3.120) and (3.121) we may note that $x(n)$ is the output of a system with input $s(n)$ and impulse response h_n, plus an uncorrelated noise, $\nu(n)$. Noting these and the fact that all the processes and system parameters are real-valued, we obtain

$$\Phi_{dx}(z) = \Phi_{sx}(z) = H(z^{-1})\Phi_{ss}(z). \tag{3.123}$$

Note that the above result is independent of $\nu(n)$. Also, with $|z| = 1$, we may also write

$$\Phi_{dx}(z) = H^*(z)\Phi_{ss}(z). \tag{3.124}$$

Using (3.122) and (3.124) in (3.93), we obtain

$$W_o(z) = \frac{H^*(z)\Phi_{ss}(z)}{\Phi_{ss}(z)|H(z)|^2 + \Phi_{\nu\nu}(z)}. \tag{3.125}$$

This is the general solution to the equalization problem when there is no constraint on the equalizer length and, also, it may be let to be non-causal. Equation (3.125) includes the effects of the autocorrelation function of the data, $s(n)$, and the noise, $\nu(n)$.

To give an interpretation of (3.125), we divide the numerator and denominator by the first term in the denominator to obtain

$$W_o(z) = \frac{1}{1 + \dfrac{\Phi_{\nu\nu}(z)}{\Phi_{ss}(z)|H(z)|^2}} \cdot \frac{1}{H(z)}. \tag{3.126}$$

Next, we replace z by $e^{j\omega}$, and define the parameter

$$\rho(e^{j\omega}) \triangleq \frac{\Phi_{ss}(e^{j\omega})|H(e^{j\omega})|^2}{\Phi_{\nu\nu}(e^{j\omega})}. \tag{3.127}$$

We may note that this is the signal-to-noise power spectral density ratio at the channel output. $\Phi_{ss}(e^{j\omega})|H(e^{j\omega})|^2$ and $\Phi_{\nu\nu}(e^{j\omega})$ are the signal power spectral density and noise power spectral density, respectively, at the channel output. Substituting (3.127) in (3.126) and rearranging, we obtain

$$W_o(e^{j\omega}) = \frac{\rho(e^{j\omega})}{1+\rho(e^{j\omega})} \cdot \frac{1}{H(e^{j\omega})}. \tag{3.128}$$

We note that the frequency response of the optimized equalizer is proportional to the inverse of the channel frequency response, with a proportionality factor that is frequency dependent. Furthermore, $\rho(e^{j\omega})$ is a non-negative real quantity, for power spectra are non-negative real functions. Hence,

$$0 \leq \frac{\rho(e^{j\omega})}{1+\rho(e^{j\omega})} \leq 1. \tag{3.129}$$

This brings us to the following interpretation of (3.128). *The frequency response of the optimum equalizer resembles the channel inverse within a real-valued factor in the range of zero to one. This factor, which is frequency dependent, depends on the signal-to-noise power spectral density ratio, $\rho(e^{j\omega})$, at the equalizer input. It approaches one when $\rho(e^{j\omega})$ is large, and reduces with $\rho(e^{j\omega})$.*

Once again, it is important to note that different frequencies are treated independently of one another by the equalizer. In particular, at a given frequency $\omega = \omega_i$, $W_o(e^{j\omega_i})$ depends only on the values of $H(e^{j\omega})$ and $\rho(e^{j\omega})$ at $\omega = \omega_i$. With this background, we shall now examine (3.128) closely to see how the equalizer is able to make a good trade-off between the cancellation of ISI and noise enhancement. In the frequency regions where the noise is almost absent, the value of $\rho(e^{j\omega})$ is very large and hence the equalizer approximates the inverse of the channel closely, without any significant enhancement of noise. On the other hand, in the frequency regions where the noise level is high (relative to the signal level) the value of $\rho(e^{j\omega})$ is not large and hence the equalizer does not approximate the channel inverse well. This, of course, is to prevent noise enhancement.

Example 3.5

Consider the channel $H(z)$ of Example 3.4. We assume that the data sequence, $s(n)$, is binary (taking values of $+1$ and -1) and white. We also assume that $\nu(n)$ is a white noise process with a variance of 0.04. With these we obtain

$$\Phi_{ss}(z) = 1 \quad \text{and} \quad \Phi_{\nu\nu}(z) = 0.04.$$

Using these in (3.125) we get

$$W_o(z) = \frac{-0.4 + z - 0.4z^2}{(-0.4 + z^{-1} - 0.4z^{-2})(-0.4 + z - 0.4z^2) + 0.04}. \tag{3.130}$$

Figure 3.11 presents the plots of $1/|H(e^{j\omega})|$ and $|W_o(e^{j\omega})|$. We note that at those frequencies where $1/|H(e^{j\omega})|$ is small, a near perfect match between $1/|H(e^{j\omega})|$ and $|W_o(e^{j\omega})|$ is observed. On the

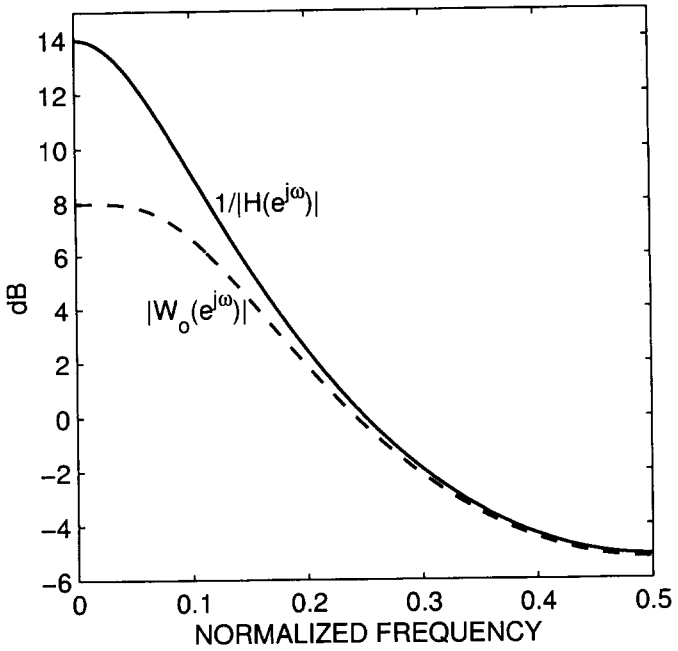


Figure 3.11 Plots of $1/|H(e^{j\omega})|$ and $|W_o(e^{j\omega})|$

other hand, at those frequencies where $1/|H(e^{j\omega})|$ is large, the deviation between the two increases. We may also note that $|W_o(e^{j\omega})|$ remains less than $1/|H(e^{j\omega})|$ for all values of ω. This is consistent with the conclusion drawn above, since a small value of $1/|H(e^{j\omega})|$ implies that $|H(e^{j\omega})|$ is large and thus, according to (3.127), $\rho(e^{j\omega})$ also is large. This, in turn, implies that the ratio $\rho(e^{j\omega})/(1+\rho(e^{j\omega}))$ is close to one, and hence from (3.128) we get

$$W_o(e^{j\omega}) \approx \frac{1}{H(e^{j\omega})}.$$

A similar argument may be used to explain why $W_o(e^{j\omega})$ is significantly smaller than $1/|H(e^{j\omega})|$ when the latter is large. Furthermore, the fact that $|W_o(e^{j\omega})|$ remains less than $1/|H(e^{j\omega})|$, for all values of ω, is predicted by (3.128).

3.6.5 Noise cancellation

Figure 3.12 depicts a typical noise canceller set-up. There are two inputs to this set-up: a signal source, $s(n)$, and a noise source, $\nu(n)$. These two signals, which are assumed to be uncorrelated with each other, are mixed together through the system functions $H(z)$ and $G(z)$ and result in the *primary input*, $d(n)$, and *reference input*, $x(n)$, as shown in Figure 3.12. The reference input is passed through a Wiener filter $W(z)$ which is designed so that the difference between the primary input and the filter output is minimized in the mean-squared sense. The noise canceller output is the error sequence $e(n)$. The aim of a noise-canceller set-up as explained above is to extract the signal $s(n)$ from the primary input $d(n)$.

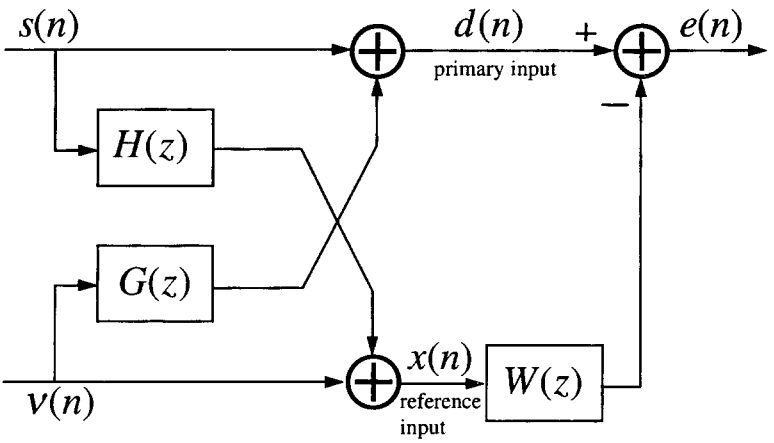


Figure 3.12 Noise canceller set-up

We note that

$$x(n) = \nu(n) + h_n * s(n) \tag{3.131}$$

and

$$d(n) = s(n) + g_n * \nu(n), \tag{3.132}$$

where h_n and g_n are the impulse responses of the filters $H(z)$ and $G(z)$, respectively.

Noting that $s(n)$ and $\nu(n)$ are uncorrelated with each other and recalling the results of Section 2.4.4, from (3.131) we obtain

$$\Phi_{xx}(z) = \Phi_{\nu\nu}(z) + \Phi_{ss}(z)|H(z)|^2. \tag{3.133}$$

To find $\Phi_{dx}(z)$, we note that $d(n)$ and $x(n)$ are related with each other through the signal sequences $s(n)$ and $\nu(n)$ and the filters $H(z)$ and $G(z)$. Since $s(n)$ and $\nu(n)$ are uncorrelated with each other, their contribution in $\Phi_{dx}(z)$ may be considered separately. In particular, we may write

$$\Phi_{dx}(z) = \Phi_{dx}^{s}(z) + \Phi_{dx}^{\nu}(z), \tag{3.134}$$

where $\Phi_{dx}^{s}(z)$ is $\Phi_{dx}(z)$ when $\nu(n) = 0$, for all values of n, and $\Phi_{dx}^{\nu}(z)$ is $\Phi_{dx}(z)$ when $s(n) = 0$, for all values of n. Thus, we obtain

$$\Phi_{dx}^{s}(z) = H^*(z)\Phi_{ss}(z) \tag{3.135}$$

and

$$\Phi_{dx}^{\nu}(z) = G(z)\Phi_{\nu\nu}(z). \tag{3.136}$$

Recall that we assume $|z| = 1$. Substituting (3.135) and (3.136) in (3.134), we get

$$\Phi_{dx}(z) = H^*(z)\Phi_{ss}(z) + G(z)\Phi_{\nu\nu}(z). \tag{3.137}$$

Using (3.133) and (3.137) in (3.93), we obtain

$$W_o(z) = \frac{H^*(z)\Phi_{ss}(z) + G(z)\Phi_{\nu\nu}(z)}{\Phi_{\nu\nu}(z) + \Phi_{ss}(z)|H(z)|^2}. \tag{3.138}$$

A comparison of (3.138) with (3.106) and (3.125) reveals that (3.138) may be thought of as a generalization of the results we obtained in the previous two sections for the modelling and inverse modelling scenarios. In fact, if we refer to Figure 3.12, we can easily find that the modelling and inverse modelling scenarios are embedded in the noise canceller set-up. While trying to minimize the mean-square value of the output error, we must strike a balance between noise cancellation and signal cancellation at the output of the noise canceller. Cancellation of the noise $\nu(n)$ occurs when the Wiener filter $W(z)$ is chosen to be close to $G(z)$, and cancellation of the signal $s(n)$ occurs when $W(z)$ is close to the inverse of $H(z)$. In this sense we may note that the noise canceller treats $s(n)$ and $\nu(n)$ without making any distinction between them and tries to cancel both of them as much as possible so as to achieve the minimum mean-square error in $e(n)$. This seems contrary to the main goal of the noise canceller, which is meant to cancel only the noise. The following discussion aims at revealing some of the peculiar characteristics of the noise canceller set-up and show under which conditions an acceptable cancellation occurs.

To proceed with our discussion, we define $\rho_{\text{pri}}(e^{j\omega})$, $\rho_{\text{ref}}(e^{j\omega})$ and $\rho_{\text{out}}(e^{j\omega})$, as the signal-to-noise power spectral density ratios at the primary input, reference input, and output, respectively. By direct inspection of Figure 3.12 and application of (2.85) we obtain

$$\rho_{\text{pri}}(e^{j\omega}) = \frac{\Phi_{ss}(e^{j\omega})}{|G(e^{j\omega})|^2\Phi_{\nu\nu}(e^{j\omega})} \tag{3.139}$$

and

$$\rho_{\text{ref}}(e^{j\omega}) = \frac{|H(e^{j\omega})|^2\Phi_{ss}(e^{j\omega})}{\Phi_{\nu\nu}(e^{j\omega})}. \tag{3.140}$$

To derive a similar equation for $\rho_{\text{out}}(e^{j\omega})$, we note that $s(n)$ reaches the canceller output through two routes: one direct and one through the cascade of $H(z)$ and $W(z)$. This gives

$$\Phi_{ee}^{s}(e^{j\omega}) = |1 - H(e^{j\omega})W(e^{j\omega})|^2\Phi_{ss}(e^{j\omega}), \tag{3.141}$$

where the superscript s refers to the portion of $\Phi_{ee}(e^{j\omega})$ that comes from $s(n)$. Similarly, $\nu(n)$ reaches the output through the routes $G(z)$ and $W(z)$. Thus,

$$\Phi_{ee}^{\nu}(e^{j\omega}) = |G(e^{j\omega}) - W(e^{j\omega})|^2\Phi_{\nu\nu}(e^{j\omega}). \tag{3.142}$$

Replacing $W(e^{j\omega})$ by $W_o(e^{j\omega})$ and using (3.138) in (3.141) and (3.142), we obtain

$$\Phi_{ee}^{s}(e^{j\omega}) = \frac{|1 - G(e^{j\omega})H(e^{j\omega})|^2 \Phi_{\nu\nu}^2(e^{j\omega})}{[\Phi_{\nu\nu}(e^{j\omega}) + |H(e^{j\omega})|^2 \Phi_{ss}(e^{j\omega})]^2} \Phi_{ss}(e^{j\omega}) \tag{3.143}$$

and

$$\Phi_{ee}^{\nu}(e^{j\omega}) = \frac{|H(e^{j\omega})|^2 |1 - G(e^{j\omega})H(e^{j\omega})|^2 \Phi_{ss}^2(e^{j\omega})}{[\Phi_{\nu\nu}(e^{j\omega}) + |H(e^{j\omega})|^2 \Phi_{ss}(e^{j\omega})]^2} \Phi_{\nu\nu}(e^{j\omega}), \tag{3.144}$$

respectively. Hence, $\rho_{out}(e^{j\omega})$ can now be obtained as

$$\rho_{out}(e^{j\omega}) = \frac{\Phi_{ee}^{s}(e^{j\omega})}{\Phi_{ee}^{\nu}(e^{j\omega})} = \frac{\Phi_{\nu\nu}(e^{j\omega})}{|H(e^{j\omega})|^2 \Phi_{ss}(e^{j\omega})}. \tag{3.145}$$

Comparing (3.145) with (3.140), we find that

$$\rho_{out}(e^{j\omega}) = \frac{1}{\rho_{ref}(e^{j\omega})}. \tag{3.146}$$

This is known as *power inversion* (Widrow et al., 1975). It shows that *the signal-to-noise power spectral density ratio at the noise canceller output is equal to the inverse of the signal-to-noise power spectral density ratio at the reference input.* This means that if the signal-to-noise power spectral density ratio at the reference input is low, then we should expect a good cancellation of the noise at the output. On the other hand, we should expect a poor performance from the canceller when the signal-to-noise power spectral density ratio at the reference input is high. This surprising result suggests that *the noise canceller works better in situations when the noise level is high and signal level is low.* The following example gives a clear picture of this general result.

Example 3.6

To demonstrate how the power inversion property of the noise canceller may be utilized in practice, we consider a receiver with two omni-directional (equally sensitive to all directions) antennas, A and B, as in Figure 3.13.

A desired signal $s(n) = \alpha(n)\cos n\omega_o$ arrives in the direction perpendicular to the line connecting A and B. An interferer (jammer) signal $\nu(n) = \beta(n)\cos n\omega_o$ arrives at an angle θ_o with respect to the direction of $s(n)$. The amplitudes $\alpha(n)$ and $\beta(n)$ are narrow-band baseband signals. This implies that $s(n)$ and $\nu(n)$ are narrow-band signals concentrated around $\omega = \omega_o$. Such signals may be treated as single tones, and thus a filter with two degrees of freedom is sufficient for any linear filtering that may have to be performed on them. This is why only a two-tap linear combiner is considered in Figure 3.13. This is expected to perform almost as good as any other unconstrained linear (Wiener) filter. We also assume that $\alpha(n)$ and $\beta(n)$ are zero-mean and uncorrelated with each other. The two omnis are separated by a distance of l metres. The linear combiner coefficients are adjusted so that the output error, $e(n)$, is minimized in the mean-square sense.

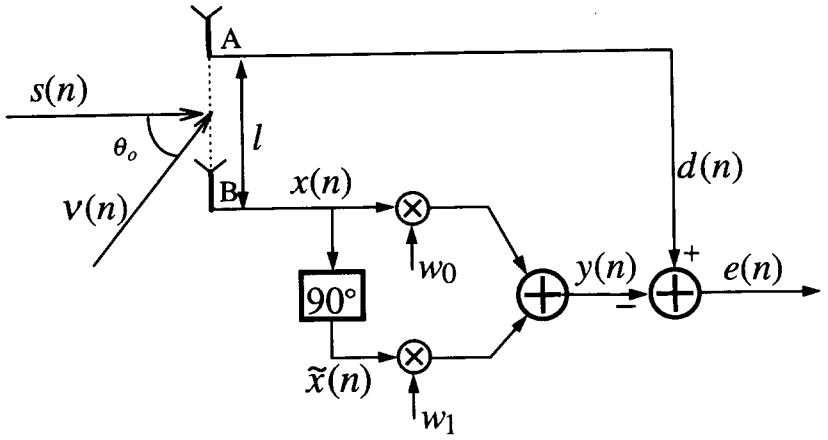


Figure 3.13 A receiver with two omni-directional antennas

The desired signal, $s(n)$, arrives at the same time at both omnis. However, $v(n)$ arrives at B first, and arrives at A with a delay

$$\delta = \frac{l \sin \theta_o}{c}, \tag{3.147}$$

where c is the propagation speed. To add this to the time index n, it has to be normalized by the time step T which corresponds to one increment of n. This gives

$$\delta_o = \frac{l \sin \theta_o}{cT}. \tag{3.148}$$

Noting these, in Figure 3.13, we have

$$d(n) = \alpha(n) \cos n\omega_o + \beta(n) \cos[(n - \delta_o)\omega_o] \tag{3.149}$$

$$x(n) = \alpha(n) \cos n\omega_o + \beta(n) \cos n\omega_o \tag{3.150}$$

$$\tilde{x}(n) = \alpha(n) \sin n\omega_o + \beta(n) \sin n\omega_o \tag{3.151}$$

It may be noted that in (3.149) we have used $\beta(n)$ instead of $\beta(n - \delta_o)$. This, which has been done to simplify the following equations, in practice is valid with a very good approximation because of the narrow bandwidth of $\beta(n)$, which implies that its variation in time is slow, and the small size of δ_o.

To find the optimum coefficients of the linear combiner, we shall derive and solve the Wiener–Hopf equation governing the linear combiner. We note here that

$$\mathbf{R} = \begin{bmatrix} \mathrm{E}[x^2(n)] & \mathrm{E}[x(n)\tilde{x}(n)] \\ \mathrm{E}[x(n)\tilde{x}(n)] & \mathrm{E}[\tilde{x}^2(n)] \end{bmatrix}. \tag{3.152}$$

Also,

$$\mathrm{E}[x^2(n)] = \mathrm{E}[(\alpha(n) \cos n\omega_o + \beta(n) \cos n\omega_o)^2]. \tag{3.153}$$

Expanding (3.153) and recalling that $\alpha(n)$ and $\beta(n)$ are uncorrelated with each other, we obtain

$$\mathrm{E}[x^2(n)] = \frac{\sigma_\alpha^2 + \sigma_\beta^2}{2} + \tfrac{1}{2}(\sigma_\alpha^2 \mathrm{E}[\cos 2n\omega_\mathrm{o}] + \sigma_\beta^2 \mathrm{E}[\cos 2n\omega_\mathrm{o}]), \tag{3.154}$$

where σ_α^2 and σ_β^2 are variances of $\alpha(n)$ and $\beta(n)$, respectively. Also, $\mathrm{E}[\cos 2n\omega_\mathrm{o}]$ is replaced by its time average.[2] This is assumed to be zero. Thus, we obtain

$$\mathrm{E}[x^2(n)] = \frac{\sigma_\alpha^2 + \sigma_\beta^2}{2}. \tag{3.155}$$

Similarly, we can obtain

$$\mathrm{E}[\tilde{x}^2(n)] = \frac{\sigma_\alpha^2 + \sigma_\beta^2}{2} \tag{3.156}$$

and

$$\mathrm{E}[x(n)\tilde{x}(n)] = 0. \tag{3.157}$$

Substituting these in (3.152) we have

$$\mathbf{R} = \frac{\sigma_\alpha^2 + \sigma_\beta^2}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{3.158}$$

It is also straightforward to show that

$$\mathbf{p} = \begin{bmatrix} \mathrm{E}[d(n)x(n)] \\ \mathrm{E}[d(n)\tilde{x}(n)] \end{bmatrix} = \frac{1}{2}\begin{bmatrix} \sigma_\alpha^2 + \sigma_\beta^2 \cos \delta_\mathrm{o}\omega_\mathrm{o} \\ \sigma_\beta^2 \sin \delta_\mathrm{o}\omega_\mathrm{o} \end{bmatrix}. \tag{3.159}$$

Using (3.158) and (3.159) in the Wiener–Hopf equation $\mathbf{R}\mathbf{w}_\mathrm{o} = \mathbf{p}$, we get

$$\mathbf{w}_\mathrm{o} = \begin{bmatrix} \dfrac{\sigma_\alpha^2 + \sigma_\beta^2 \cos \delta_\mathrm{o}\omega_\mathrm{o}}{\sigma_\alpha^2 + \sigma_\beta^2} \\ \dfrac{\sigma_\beta^2 \sin \delta_\mathrm{o}\omega_\mathrm{o}}{\sigma_\alpha^2 + \sigma_\beta^2} \end{bmatrix}. \tag{3.160}$$

The optimized output of the receiver is

$$e_\mathrm{o}(n) = d(n) - \mathbf{w}_\mathrm{o}^\mathrm{T}\mathbf{x}(n), \tag{3.161}$$

---

[2]Strictly speaking the replacement of the time average of the periodic sequence $\cos 2n\omega_\mathrm{o}$ as $E[\cos 2n\omega_\mathrm{o}]$ does not fit into the conventional definitions of stochastic processes. The sequence $\cos 2n\omega_\mathrm{o}$ is deterministic and thus it does not really make sense to talk about its expectation, which conventionally is defined as an ensemble average. On the other hand, this is a reality that in many occasions in adaptive filters (such as our example here) the involved signals are deterministic and the time averages are used to evaluate the performance of the filters and/or calculate their parameters. This is in this context that we replace statistical expectations by time averages. We may note that the problem stated in Example 3.6 could also be put in a more statistical form to prevent the above arguments; see Problem P3.21. Here, we have decided not to do this in order to emphasize the fact that in practice time averages are used instead of statistical expectations.

where $\mathbf{x}(n) = [x(n)\ \tilde{x}(n)]^{\mathrm{T}}$. Using (3.149), (3.150), (3.151) and (3.160) in (3.161), we get, after some manipulations,

$$e_{\mathrm{o}}(n) = \frac{\cos n\omega_{\mathrm{o}} - \cos[(n-\delta_{\mathrm{o}})\omega_{\mathrm{o}}]}{\sigma_\alpha^2 + \sigma_\beta^2}(\sigma_\beta^2\alpha(n) - \sigma_\alpha^2\beta(n)). \tag{3.162}$$

Now, by inspection of (3.150) and (3.162), we find that

$$\text{the signal-to-noise ratio at the reference input} = \frac{\sigma_\alpha^2}{\sigma_\beta^2} \tag{3.163}$$

and

$$\text{the signal-to-noise ratio at the output} = \frac{(\sigma_\beta^2)^2\sigma_\alpha^2}{(\sigma_\alpha^2)^2\sigma_\beta^2} = \frac{\sigma_\beta^2}{\sigma_\alpha^2} \tag{3.164}$$

which match the power inversion equation (3.146).

3.7 Summary and Discussion

In this chapter we reviewed a class of optimum linear systems collectively known as Wiener filters. We noted that the performance function used in formulating the Wiener filters is an elegant choice which leads to a mathematically tractable problem. We discussed the Wiener filters in the context of discrete-time signals and systems, and presented different formulations of the Wiener filtering problem. We started with the

transfer function at any particular frequency depends only on the power spectral density of the filter input and the cross-power spectral density between the filter input and its desired output, at that frequency.

The last property, although it could only be derived in the case of unconstrained Wiener filters, is also approximately valid when the filter length is constrained. The concept of power spectra and their influence on the performance of Wiener filters is fundamental to understanding the behaviour of adaptive filters. We note that the adaptive filters, as commonly implemented, are aimed at implementing Wiener filters. In this chapter we saw that the optimum coefficients of the Wiener filter are a function of the autocorrelation function of the filter input and the cross-correlation function between the filter input and its desired output. Since correlation functions and power spectra are uniquely related, we also saw that the optimum coefficients can be expressed in terms of the corresponding power spectra instead of the correlation functions. In the next few chapters, we will show that the convergence behaviour of adaptive filters is closely related to the power spectrum of their inputs. In the rest of this book we will make frequent references to the results derived in this chapter.

Problems

P3.1 Consider a two-tap Wiener filter with the following statistics:

$$\mathrm{E}[d^2(n)] = 2, \quad \mathbf{R} = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}.$$

(i) Use the above information to obtain the performance function of the filter. By direct evaluation of the performance function obtain the optimum values of the filter tap weights.
(ii) Insert the result obtained in (i) in the performance function expression to obtain the minimum mean-square error of the filter.
(iii) Find the optimum tap weights of the filter and its minimum mean-square error using the equations derived in this chapter to confirm the results obtained in (i) and (ii).

P3.2 Consider a three-tap Wiener filter with the following statistics:

$$\mathrm{E}[d^2(n)] = 10, \quad \mathbf{R} = \begin{bmatrix} 1 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 1 \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix}.$$

Repeat Steps (i), (ii) and (iii) of Problem P3.1.

P3.3 Consider the modelling problem shown in Figure P3.3.

(i) Find the correlation matrix **R** of the filter tap inputs and the cross-correlation vector **p** between the filter tap inputs and its desired output.

(ii) Find the optimum tap weights of the Wiener filter.
(iii) What is the minimum mean-squared error? Obtain this analytically as well as by direct inspection of Figure P3.3.

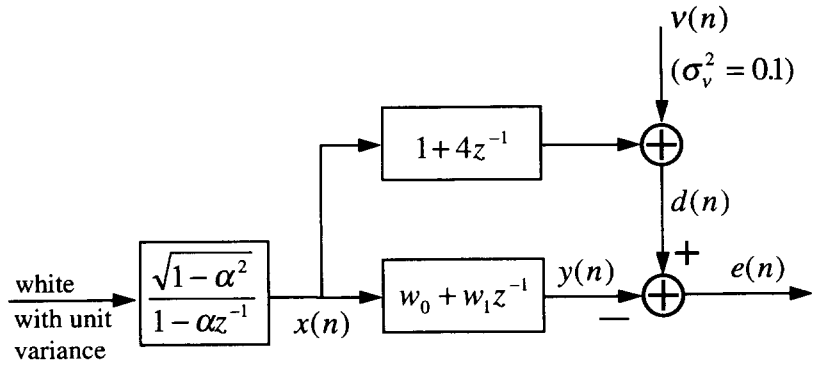


Figure P3.3

P3.4 Consider the channel equalization problem shown in Figure P3.4. The data symbols, $s(n)$, are assumed to be samples of a stationary white process.

(i) Find the correlation matrix **R** of the equalizer tap inputs and the cross-correlation vector **p** between the equalizer tap inputs and the desired output.
(ii) Find the optimum tap weights of the equalizer.
(iii) What is the minimum mean-square error at the equalizer output?
(iv) Could you guess the results obtained in (ii) and (iii) without going through the derivations? How and why?

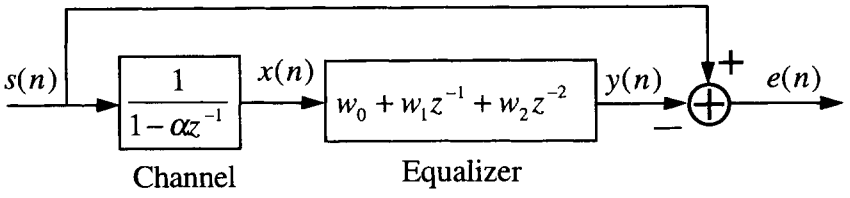


Figure P3.4

P3.5 In Section 3.5 we emphasized that for a complex variable w

$$\nabla_w^{\mathrm{C}} f(w) = 0 \tag{P3.5-1}$$

does not imply that

$$\frac{\partial f(w)}{\partial w_{\mathrm{R}}} = \frac{\partial f(w)}{\partial w_{\mathrm{I}}} = 0, \tag{P3.5-2}$$

in general. In this problem we want to elaborate on this further.

(i) Assume that $f(w) = w^m$ and show that for this function (P3.5-1) is true, but (P3.5-2) is false.
(ii) Can you extend this result to the case when

$$f(w) = \sum_{i=0}^{L} a_i w^i$$

and the a_is are fixed real or complex coefficients?

P3.6 Work out the details of the derivation of (3.74).

P3.7 In Section 3.5, for the complex-valued signals, we used the principle of orthogonality to derive the Wiener–Hopf equation and the minimum mean-square error (3.74). Starting with the definition of the performance function, derive an equation similar to (3.12) for the case of complex-valued signals. Use this equation to give a direct derivation for the Wiener–Hopf equation in the present case. Also, confirm the minimum mean-square error equation (3.74).

P3.8 Show that for a Wiener filter with a complex-valued tap-input vector $\mathbf{x}(n)$ and optimum tap-weight vector $\mathbf{w}_o$

$$\mathbf{w}_o^H \mathbf{p} = \mathrm{E}[|\mathbf{w}_o^H \mathbf{x}(n)|^2],$$

where $\mathbf{p} = \mathrm{E}[d(n)\mathbf{x}^*(n)]$ and $d(n)$ is the desired output of the filter. Use this result to argue that $\mathbf{w}_o^H \mathbf{p}$ is always positive. Also, use the above result to derive an equation similar to (3.50) for the general case of Wiener filters with complex-valued signals.

P3.9 Consider the channel equalization problem depicted in Figure P3.9. Assume that the underlying processes are real-valued with

$$\mathrm{E}[s(n)s(n-k)] = \begin{cases} 1, & k = 0, \\ 0, & k \neq 0, \end{cases}$$

and

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

(i) For $\sigma_\nu^2 = 0$, obtain the equalizer tap weights by direct solution of the Wiener–Hopf equation. To be sure of your results, you may also guess the equalizer tap weights and compare them with the calculated ones.
(ii) Find the equalizer tap weights when $\sigma_\nu^2 = 0.1$ and compare the results with what you obtained in (i).
(iii) Plot the magnitude and phase responses of the two designs obtained above and compare the results.

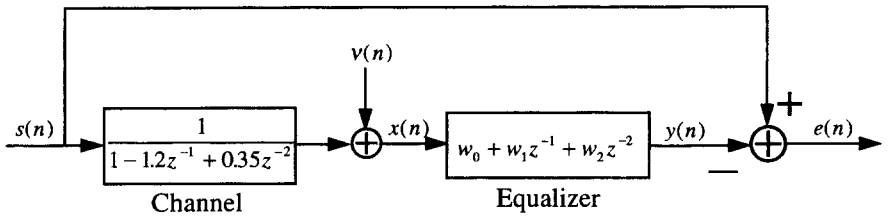


Figure P3.9

P3.10 By following a procedure similar to the one given in Section 3.6.1, show that when the involved processes and system parameters are complex-valued

$$\xi = \phi_{dd}(0) + \phi_{yy}(0) - 2\Re\{\phi_{yd}(0)\},$$

where $\Re\{x\}$ denotes the real part of x. Proceed with the above result to develop the dual of (3.78).

P3.11 Show that (3.93) is a valid result even when the involved processes are complex-valued.

P3.12 Consider Figure P3.12, in which $x_i(n)$ and $d_i(n)$ are the outputs of two similar narrowband filters centred at $\omega = \omega_i$, as in Figure 3.7. Show that if $w_{i,o}$ is the optimum value of w_i that minimizes the mean-square error of the output error, $e_i(n)$, then

$$w_{i,o} \approx \frac{\Phi_{dx}(e^{j\omega_i})}{\Phi_{xx}(e^{j\omega_i})}.$$

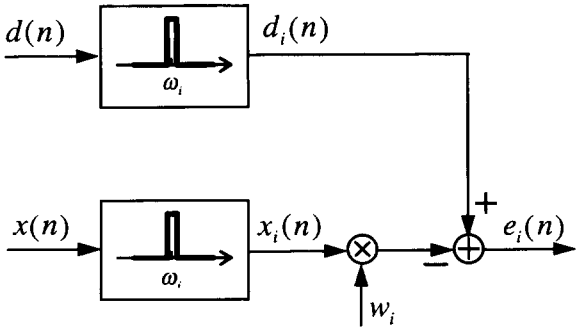


Figure P3.12

P3.13 Assuming that $W_o(z)$ is the optimum system function of a FIR Wiener filter, show that (3.96) can be converted to (3.26), and vice-versa.

P3.14 Give a detailed derivation of (3.122) from (3.120).

P3.15 Give a detailed derivation of (3.123).

P3.16 For the noise canceller set-up of Figure 3.12, consider the case when $\Phi_{ss}(z)|H(z)|^2 \ll \Phi_{\nu\nu}(z)$.

(i) Show that, in this case,

$$W_o(z) \approx G(z) + \frac{\Phi_{ss}(z)}{\Phi_{\nu\nu}(z)} H^*(z).$$

(ii) Show that the power spectral density of the noise reaching the noise canceller output is

$$\Phi_{\text{output noise}}(z) \approx \Phi_{\nu\nu}(z)\rho_{\text{ref}}(z)\rho_{\text{pri}}(z)|G(z)|^2.$$

(iii) Define the signal distortion at the canceller output, $D(z)$, as the ratio of the power spectral density of the signal propagating through $W_o(z)$ to the output to the power spectral density of the signal at the primary input. Show that

$$D(z) \approx |H(z)G(z) + \rho_{\text{ref}}(z)|^2.$$

(iv) Show that the result obtained in (iii) may be written as

$$D(z) \approx \frac{\rho_{\text{ref}}(z)}{\rho_{\text{pri}}(z)}$$

when $\rho_{\text{ref}}(z) \ll |H(z)| \cdot |G(z)|$.

P3.17 Consider the noise canceller set-up shown in Figure P3.17.

(i) Derive an unconstrained Wiener filter $W_o(z)$.
(ii) Show that the power inversion formula (3.146) is also valid for this set-up.

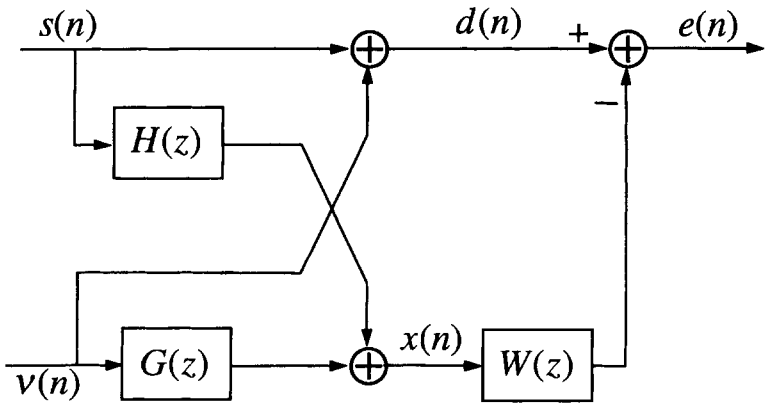


Figure P3.17

P3.18 Consider an array of three omni-directional antennas as in Figure P3.18. The signal, $s(n)$, and jammer, $\nu(n)$, are narrowband processes, as in Example 3.6. To cancel

the jammer, we use a two-tap filter, similar to the one used in Figure 3.13, at either of the points 1 or 2, in Figure P3.18.

(i) To maximize the cancellation of the jammer, where will you place the two-tap filter?
(ii) For your choice in (i), find the optimum values of the filter tap weights.
(iii) Find an expression for the signal and jammer components reaching the canceller output, and confirm the power inversion formula.

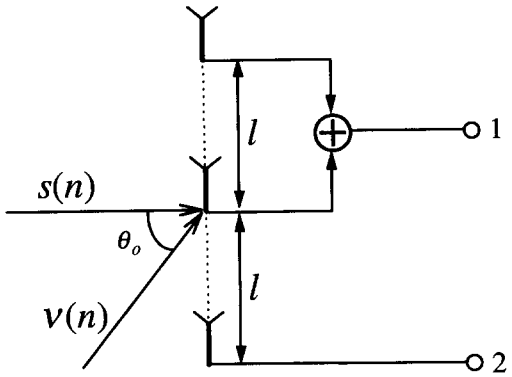


Figure P3.18

P3.19 Consider an array of three omni-directional antennas as in Figure P3.19. The signal, $s(n)$, and jammer, $v(n)$, are narrow-band processes, as in Example 3.6.

(i) Find the optimum values of the filter tap weights that minimize the mean-square error of the output error, $e(n)$.
(ii) Find an expression for the canceller output, and investigate the validity of the power inversion formula in this case.

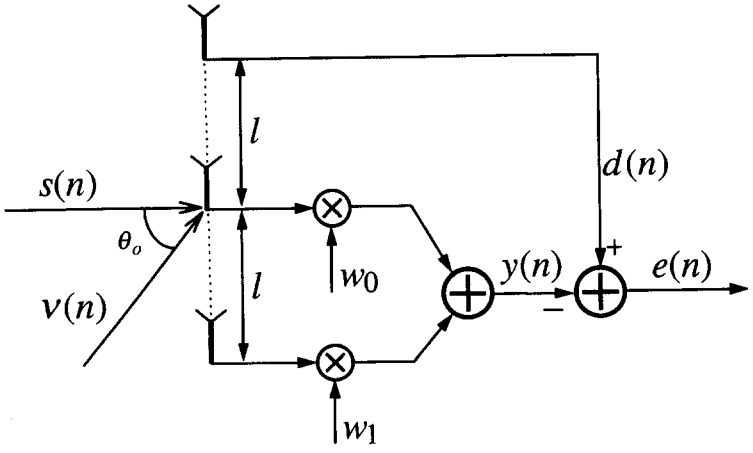


Figure P3.19

P3.20 Repeat P3.19 for the array shown in Figure P3.20, and compare the results obtained with those of P3.19.

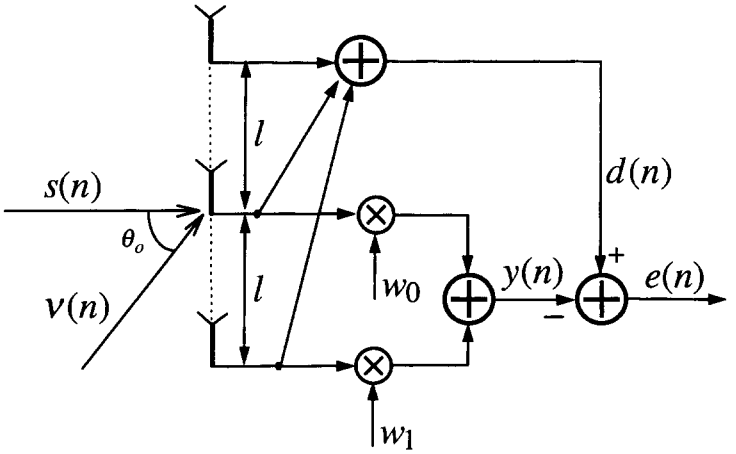


Figure P3.20

P3.21 To prevent time averages and derive the results presented in Example 3.6 through ensemble averages, the desired signal and jammer may be redefined as $s(n) = \alpha(n)\cos(n\omega_o + \varphi_1)$ and $v(n) = \beta(n)\cos(n\omega_o + \varphi_2)$, respectively, where φ_1 and φ_2 are random initial phases of the carrier, and assumed to be uniformly distributed in the interval $-\pi$ to $+\pi$. The amplitudes $\alpha(n)$ and $\beta(n)$, as in Example 3.6, are uncorrelated narrow-band baseband signals. Furthermore, the random phases φ_1 and φ_2 are assumed to be independent among themselves as well as with respect to $\alpha(n)$ and $\beta(n)$.

(i) Using the new definitions of $s(n)$ and $v(n)$, show that the same result as in (3.160) is also obtained through ensemble averages.

(ii) Show that, for the present case,

$$e_o(n) = \frac{\sigma_\beta^2 \alpha(n)}{\sigma_\alpha^2 + \sigma_\beta^2}[\cos(n\omega_o + \varphi_1) - \cos((n - \delta_o)\omega_o + \varphi_1)]$$

$$- \frac{\sigma_\alpha^2 \beta(n)}{\sigma_\alpha^2 + \sigma_\beta^2}[\cos(n\omega_o + \varphi_2) - \cos((n - \delta_o)\omega_o + \varphi_2)].$$

(iii) Use the result in (ii) to verify the power inversion formula in the present case.

4

Eigenanalysis and the Performance Surface

The transversal Wiener filter was introduced in the previous chapter as a powerful signal processing structure with a unique performance function which has many desirable features for adaptive filtering applications. In particular, it was noted that the performance function of the transversal Wiener filter has a unique global minimum point which can be easily obtained using the second-order moments of the underlying processes. This is a consequence of the fact that the performance function of the transversal Wiener filter is a convex quadratic function of its tap weights.

Our goal in this chapter is to analyse in detail the quadratic performance function of the transversal Wiener filter. We get a clear picture of the shape of the performance function when it is visualized as a surface in the $(N+1)$-dimensional space of variables consisting of the filter tap weights, as the first N axes, and the performance function, as the $(N+1)$ th axis. This is called the *performance surface*.

The shape of the performance surface of a transversal Wiener filter is closely related to the eigenvalues of the correlation matrix $\mathbf{R}$ of the filter tap inputs. Hence, we start with a thorough discussion on the eigenvalues and eigenvectors of the correlation matrix $\mathbf{R}$.

4.1 Eigenvalues and Eigenvectors

Let

$$\mathbf{R} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{H}}(n)] \tag{4.1}$$

be the $N \times N$ correlation matrix of a complex-valued wide-sense stationary stochastic process represented by the $N \times 1$ observation vector $\mathbf{x}(n) = [x(n)\ x(n-1)\ \cdots\ x(n-N+1)]^{\mathrm{T}}$, where the superscripts H and T denote Hermitian and transpose, respectively.

A non-zero $N \times 1$ vector $\mathbf{q}$ is said to be an *eigenvector* of $\mathbf{R}$ if it satisfies the equation

$$\mathbf{R}\mathbf{q} = \lambda\mathbf{q} \tag{4.2}$$

for some scalar constant λ. The scalar λ is called the *eigenvalue* of $\mathbf{R}$ associated with the eigenvector $\mathbf{q}$. We note that if $\mathbf{q}$ is an eigenvector of $\mathbf{R}$, then for any non-zero scalar a, $a\mathbf{q}$ is also an eigenvector of $\mathbf{R}$, corresponding to the same eigenvalue, λ. This is easily verified by multiplying (4.2) through by a.

To find the eigenvalues and eigenvectors of $\mathbf{R}$, we note that (4.2) may be rearranged as

$$(\mathbf{R} - \lambda_i \mathbf{I})\mathbf{q} = \mathbf{0}, \tag{4.3}$$

where $\mathbf{I}$ is the $N \times N$ identity matrix, and $\mathbf{0}$ is the $N \times 1$ null vector. To prevent the trivial solution $\mathbf{q} = \mathbf{0}$, the matrix $\mathbf{R} - \lambda\mathbf{I}$ has to be singular. This implies

$$\det(\mathbf{R} - \lambda\mathbf{I}) = 0, \tag{4.4}$$

where $\det(\cdot)$ denotes determinant. Equation (4.4) is called the *characteristic equation* of the matrix $\mathbf{R}$. The characteristic equation (4.4), when expanded, is an N th order equation in the unknown parameter λ. The roots of this equation, which may be called $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$, are the eigenvalues of $\mathbf{R}$. When λ_is are distinct, $\mathbf{R} - \lambda_i\mathbf{I}$, for $i = 0, 1, \ldots, N-1$, will be of rank $N-1$. This leads to N eigenvectors $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{N-1}$, for the matrix $\mathbf{R}$, which are unique up to a scale factor. On the other hand, when the characteristic equation (4.4) has repeated roots, the matrix $\mathbf{R}$ is said to have *degenerate* eigenvalues. In that case, the eigenvectors of $\mathbf{R}$ will not be unique. For example, if λ_m is an eigenvalue of $\mathbf{R}$ repeated p times, then the rank of $\mathbf{R} - \lambda_m\mathbf{I}$ is $N - p$, and thus the solution of the equation $(\mathbf{R} - \lambda_m\mathbf{I})\mathbf{q}_m = \mathbf{0}$ can be any vector in a p-dimensional subspace of the N-dimensional complex vector space. This, in general, creates some confusion in eigenanalysis of matrices which should be handled carefully. To prevent such confusion, in the discussion that follows, wherever necessary, we start with the case that the eigenvalues of $\mathbf{R}$ are distinct. The results will then be extended to the case of repeated eigenvalues.

4.2 Properties of Eigenvalues and Eigenvectors

We discuss the various properties of the eigenvalues and eigenvectors of the correlation matrix $\mathbf{R}$. Some of the properties derived here are directly related to the fact that the correlation matrix $\mathbf{R}$ is Hermitian and non-negative definite. A matrix $\mathbf{A}$, in general, is said to be Hermitian if $\mathbf{A} = \mathbf{A}^{\mathrm{H}}$. This, for the correlation matrix $\mathbf{R}$, is observed by direct inspection of (4.1). The $N \times N$ Hermitian matrix $\mathbf{A}$ is said to be non-negative definite or *positive semidefinite*, if

$$\mathbf{v}^{\mathrm{H}}\mathbf{A}\mathbf{v} \geq 0 \tag{4.5}$$

for any $N \times 1$ vector $\mathbf{v}$. The fact that $\mathbf{A}$ is Hermitian implies that $\mathbf{v}^{\mathrm{H}}\mathbf{A}\mathbf{v}$ is real-valued. This can be seen easily if we note that with the dimensions specified above, $\mathbf{v}^{\mathrm{H}}\mathbf{A}\mathbf{v}$ is a scalar and $(\mathbf{v}^{\mathrm{H}}\mathbf{A}\mathbf{v})^* = (\mathbf{v}^{\mathrm{H}}\mathbf{A}\mathbf{v})^{\mathrm{H}} = \mathbf{v}^{\mathrm{H}}\mathbf{A}\mathbf{v}$. For the correlation matrix $\mathbf{R}$, to show that $\mathbf{v}^{\mathrm{H}}\mathbf{R}\mathbf{v}$ can never be negative, we replace $\mathbf{R}$ from (4.1) to obtain

$$\begin{aligned} \mathbf{v}^{\mathrm{H}}\mathbf{R}\mathbf{v} &= \mathbf{v}^{\mathrm{H}}\mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{H}}(n)]\mathbf{v} \\ &= \mathrm{E}[\mathbf{v}^{\mathrm{H}}\mathbf{x}(n)\mathbf{x}^{\mathrm{H}}(n)\mathbf{v}]. \end{aligned} \tag{4.6}$$

We note that $\mathbf{v}^H\mathbf{x}(n)$ and $\mathbf{x}^H(n)\mathbf{v}$ constitute a pair of complex-conjugate scalars. This, when used in (4.6), gives

$$\mathbf{v}^H\mathbf{R}\mathbf{v} = E[|\mathbf{v}^H\mathbf{x}(n)|^2], \tag{4.7}$$

which is non-negative for any vector $\mathbf{v}$.

From (4.7) we note that when $\mathbf{v}$ is non-zero, the Hermitian form $\mathbf{v}^H\mathbf{R}\mathbf{v}$ may be zero only when there is a consistent dependency between the elements of the observation vector $\mathbf{x}(n)$, so that $\mathbf{v}^H\mathbf{x}(n) = 0$ for all observations of $\mathbf{x}(n)$. For a random process $\{x(n)\}$, this can only happen when $\{x(n)\}$ consists of a sum of L sinusoids with $L < N$. In practice, we find that this situation is very rare and thus for any non-zero $\mathbf{v}$, $\mathbf{v}^H\mathbf{R}\mathbf{v}$ is almost always positive. We thus say that the correlation matrix $\mathbf{R}$ is almost always *positive definite*.

With this background, we are now prepared to discuss the properties of the eigenvalues and eigenvectors of the correlation matrix $\mathbf{R}$.

Property 1 *The eigenvalues of the correlation matrix* $\mathbf{R}$ *are all real and non-negative.*

Consider an eigenvector $\mathbf{q}_i$ of $\mathbf{R}$ and its corresponding eigenvalue λ_i. These two are related according to the equation

$$\mathbf{R}\mathbf{q}_i = \lambda_i\mathbf{q}_i. \tag{4.8}$$

Premultiplying (4.8) by $\mathbf{q}_i^H$ and noting that λ_i is a scalar, we get

$$\mathbf{q}_i^H\mathbf{R}\mathbf{q}_i = \lambda_i\mathbf{q}_i^H\mathbf{q}_i. \tag{4.9}$$

The quantity $\mathbf{q}_i^H\mathbf{q}_i$ on the right-hand side is always real and positive, since it is the squared length of the vector $\mathbf{q}_i$. Furthermore, the Hermitian form $\mathbf{q}_i^H\mathbf{R}\mathbf{q}_i$ on the left-hand side of (4.9) is always real and non-negative, since the correlation matrix $\mathbf{R}$ is non-negative definite. Noting these, it follows from (4.9) that

$$\lambda_i \geq 0, \quad \text{for } i = 0, 1, \ldots, N-1. \tag{4.10}$$

Property 2 *If* $\mathbf{q}_i$ *and* $\mathbf{q}_j$ *are two eigenvectors of the correlation matrix* $\mathbf{R}$ *that correspond to two of its distinct eigenvalues, then*

$$\mathbf{q}_i^H\mathbf{q}_j = 0. \tag{4.11}$$

In other words, eigenvectors associated with the distinct eigenvalues of the correlation matrix $\mathbf{R}$ *are mutually orthogonal.*

Let λ_i and λ_j be the distinct eigenvalues corresponding to the eigenvectors $\mathbf{q}_i$ and $\mathbf{q}_j$, respectively. We have

$$\mathbf{R}\mathbf{q}_i = \lambda_i\mathbf{q}_i \tag{4.12}$$

and

$$\mathbf{R}\mathbf{q}_j = \lambda_j \mathbf{q}_j. \tag{4.13}$$

Applying the conjugate transpose on both sides of (4.12) and noting that λ_i is a scalar and for the Hermitian matrix $\mathbf{R}$, $\mathbf{R}^{\mathrm{H}} = \mathbf{R}$, we obtain

$$\mathbf{q}_i^{\mathrm{H}}\mathbf{R} = \lambda_i \mathbf{q}_i^{\mathrm{H}}. \tag{4.14}$$

Premultiplying (4.13) by $\mathbf{q}_i^{\mathrm{H}}$, post-multiplying (4.14) by $\mathbf{q}_j$, and subtracting the two resulting equations, gives

$$(\lambda_j - \lambda_i)\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_j = 0. \tag{4.15}$$

Noting that λ_i and λ_j are distinct, this gives (4.11).

Property 3 *Let $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{N-1}$ be the eigenvectors associated with the distinct eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$ of the $N \times N$ correlation matrix $\mathbf{R}$, respectively. Assume that the eigenvectors $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{N-1}$ are all normalized to have a length of unity, and define the $N \times N$ matrix*

$$\mathbf{Q} = [\mathbf{q}_0 \ \mathbf{q}_1 \ \cdots \ \mathbf{q}_{N-1}]. \tag{4.16}$$

$\mathbf{Q}$ is then a unitary matrix, i.e.

$$\mathbf{Q}^{\mathrm{H}}\mathbf{Q} = \mathbf{I}. \tag{4.17}$$

This implies that the matrices $\mathbf{Q}$ and $\mathbf{Q}^{\mathrm{H}}$ are the inverse of each other.

To show this property, we note that the ijth element of the $N \times N$ matrix $\mathbf{Q}^{\mathrm{H}}\mathbf{Q}$ is the product of the ith row of $\mathbf{Q}^{\mathrm{H}}$, which is $\mathbf{q}_i^{\mathrm{H}}$, and the jth column of $\mathbf{Q}$, which is $\mathbf{q}_j$. That is,

$$\text{the } ij\text{th element of } \mathbf{Q}^{\mathrm{H}}\mathbf{Q} = \mathbf{q}_i^{\mathrm{H}}\mathbf{q}_j. \tag{4.18}$$

Noting this, (4.17) follows immediately from Property 2.

In cases where the correlation matrix $\mathbf{R}$ has one or more repeated eigenvalues, as was noted above, attached to each of these repeated eigenvalues there is a subspace of the same dimension as the multiplicity of the eigenvalue in which any vector is an eigenvector of $\mathbf{R}$. From Property 2, we can say that the subspaces that belong to distinct eigenvalues are orthogonal. Moreover, within each subspace we can always find a set of orthogonal basis vectors which span the whole subspace. Clearly, such a set is not unique, but can always be chosen. This means that for any repeated eigenvalue with multiplicity p, one can always find a set of p orthogonal eigenvectors. Noting this, we can say, in general, that for any $N \times N$ correlation matrix $\mathbf{R}$, we can always make a unitary matrix $\mathbf{Q}$ whose columns are made-up of a set of eigenvectors of $\mathbf{R}$.

Property 4 *For any $N \times N$ correlation matrix $\mathbf{R}$, we can always find a set of mutually orthogonal eigenvectors. Such a set may be used as a basis to express any vector in the N-dimensional space of complex vectors.*

This property follows from the above discussion.

Property 5 *Unitary Similarity Transformation. The correlation matrix $\mathbf{R}$ can always be decomposed as*

$$\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{H}}, \tag{4.19}$$

where the matrix $\mathbf{Q}$ is made up from a set of unit-length orthogonal eigenvectors of $\mathbf{R}$ as specified in (4.16) *and* (4.17),

$$\mathbf{\Lambda} \triangleq \begin{bmatrix} \lambda_0 & 0 & \dots & 0 \\ 0 & \lambda_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_{N-1} \end{bmatrix}, \tag{4.20}$$

and the order of the eigenvalues $\lambda_0, \lambda_1, \dots, \lambda_{N-1}$ matches that of the corresponding eigenvectors in the columns of $\mathbf{Q}$.

To prove this property we note that the set of equations

$$\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i, \quad \text{for } i = 0, 1, \dots, N-1, \tag{4.21}$$

may be packed together as a single matrix equation

$$\mathbf{R}\mathbf{Q} = \mathbf{Q}\mathbf{\Lambda}. \tag{4.22}$$

Then, post-multiplying (4.22) by $\mathbf{Q}^{\mathrm{H}}$ and noting that $\mathbf{Q}\mathbf{Q}^{\mathrm{H}} = \mathbf{I}$, we can get (4.19).

The right-hand side of (4.19) may be expanded as

$$\mathbf{R} = \sum_{i=0}^{N-1} \lambda_i \mathbf{q}_i \mathbf{q}_i^{\mathrm{H}}. \tag{4.23}$$

Property 6 *Let $\lambda_0, \lambda_1, \dots, \lambda_{N-1}$ be the eigenvalues of the correlation matrix $\mathbf{R}$. Then,*

$$\mathrm{tr}[\mathbf{R}] = \sum_{i=0}^{N-1} \lambda_i, \tag{4.24}$$

where $\mathrm{tr}[\mathbf{R}]$ denotes trace of $\mathbf{R}$ and is defined as the sum of the diagonal elements of $\mathbf{R}$.

Taking the trace on both sides of (4.19), we get

$$\mathrm{tr}[\mathbf{R}] = \mathrm{tr}[\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{H}}]. \tag{4.25}$$

To proceed, we may use the following result from matrix algebra. If $\mathbf{A}$ and $\mathbf{B}$ are $N \times M$ and $M \times N$ matrices, respectively, then,

$$\text{tr}[\mathbf{AB}] = \text{tr}[\mathbf{BA}]. \tag{4.26}$$

Using this result, we may swap $\mathbf{Q}\Lambda$ and $\mathbf{Q}^{\text{H}}$ on the right-hand side of (4.25). Then, noting that $\mathbf{Q}^{\text{H}}\mathbf{Q} = \mathbf{I}$, (4.25) is simplified as

$$\text{tr}[\mathbf{R}] = \text{tr}[\Lambda]. \tag{4.27}$$

Using definition (4.20) in (4.27) completes the proof.

An alternative way of proving the above result is by direct expansion of (4.4); see Problem P4.8. This proof shows that the identity (4.24) is not limited to the Hermitian matrices. It applies to any square matrix.

Property 7: Minimax Theorem[1] *The distinct eigenvalues* $\lambda_0 > \lambda_1 > \cdots > \lambda_{N-1}$ *of the correlation matrix* $\mathbf{R}$ *of an observation vector* $\mathbf{x}(n)$*, and their corresponding eigenvectors,* $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{N-1}$*, may be obtained through the following optimization procedure:*

$$\lambda_{\max} = \lambda_0 = \max_{\|\mathbf{q}_0\|=1} \text{E}[|\mathbf{q}_0^{\text{H}}\mathbf{x}(n)|^2] \tag{4.28}$$

and for $i = 1, 2, \ldots, N-1$

$$\lambda_i = \max_{\|\mathbf{q}_i\|=1} \text{E}[|\mathbf{q}_i^{\text{H}}\mathbf{x}(n)|^2] \tag{4.29}$$

with

$$\mathbf{q}_i^{\text{H}}\mathbf{q}_j = 0, \quad \text{for } 0 \le j < i \tag{4.30}$$

where $\|\mathbf{q}_i\| \triangleq \sqrt{\mathbf{q}_i^{\text{H}}\mathbf{q}_i}$ *denotes the length or norm of the complex vector* $\mathbf{q}_i$.

Alternatively, the following procedure may also be used to obtain the eigenvalues of the correlation matrix $\mathbf{R}$*, in the ascending order:*

$$\lambda_{\min} = \lambda_{N-1} = \min_{\|\mathbf{q}_{N-1}\|=1} \text{E}[|\mathbf{q}_{N-1}^{\text{H}}\mathbf{x}(n)|^2] \tag{4.31}$$

and for $i = N-2, \ldots, 1, 0$

$$\lambda_i = \min_{\|\mathbf{q}_i\|=1} \text{E}[|\mathbf{q}_i^{\text{H}}\mathbf{x}(n)|^2] \tag{4.32}$$

with

$$\mathbf{q}_i^{\text{H}}\mathbf{q}_j = 0, \quad \text{for } i < j \le N-1. \tag{4.33}$$

[1] In the matrix algebra literature, the minimax theorem is usually stated using the Hermitian form $\mathbf{q}_i^{\text{H}}\mathbf{R}\mathbf{q}$ instead of $\text{E}[|\mathbf{q}^{\text{H}}\mathbf{x}(n)|^2]$, see Haykin (1991), for example. The method that we have adopted here is to simplify some of our discussions in the following chapters. This method has been adopted from Farhang-Boroujeny and Gazor (1992).

Let us assume that the set of vectors that satisfies the minimax optimization procedure are the unit-length vectors $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{N-1}$. From Property 4 we recall that the eigenvectors $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{N-1}$ are a set of basis vectors for the N-dimensional complex vector space. This implies that, we may write

$$\mathbf{p}_i = \sum_{j=0}^{N-1} \alpha_{ij}\mathbf{q}_j, \quad \text{for } i = 0, 1, \ldots, N-1, \tag{4.34}$$

where the complex-valued coefficients α_{ij}s are the coordinates of the complex vectors $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{N-1}$ in the N-dimensional space spanned by the basis vectors $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{N-1}$.

Let $\mathbf{p}_0$ be the unit-length complex vector which maximizes $\mathrm{E}[|\mathbf{p}_0^{\mathrm{H}}\mathbf{x}(n)|^2]$. We note that

$$\begin{aligned}\mathrm{E}[|\mathbf{p}_0^{\mathrm{H}}\mathbf{x}(n)|^2] &= \mathrm{E}[\mathbf{p}_0^{\mathrm{H}}\mathbf{x}(n)\mathbf{x}^{\mathrm{H}}(n)\mathbf{p}_0] \\ &= \mathbf{p}_0^{\mathrm{H}}\mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{H}}(n)]\mathbf{p}_0 \\ &= \mathbf{p}_0^{\mathrm{H}}\mathbf{R}\mathbf{p}_0. \end{aligned} \tag{4.35}$$

Substituting (4.23) in (4.35) we obtain

$$\mathrm{E}[|\mathbf{p}_0^{\mathrm{H}}\mathbf{x}(n)|^2] = \sum_{i=0}^{N-1} \lambda_i \mathbf{p}_0^{\mathrm{H}}\mathbf{q}_i\mathbf{q}_i^{\mathrm{H}}\mathbf{p}_0. \tag{4.36}$$

Using Property 3, we get

$$\mathbf{p}_0^{\mathrm{H}}\mathbf{q}_i = \alpha_{0i}^* \tag{4.37}$$

and

$$\mathbf{q}_i^{\mathrm{H}}\mathbf{p}_0 = \alpha_{0i}. \tag{4.38}$$

Substituting these in (4.36) we obtain

$$\mathrm{E}[|\mathbf{p}_0^{\mathrm{H}}\mathbf{x}(n)|^2] = \sum_{i=0}^{N-1} \lambda_i |\alpha_{0i}|^2. \tag{4.39}$$

On the other hand, we may note that, since $\lambda_0 > \lambda_1, \lambda_2, \ldots, \lambda_{N-1}$,

$$\sum_{i=0}^{N-1} \lambda_i |\alpha_{0i}|^2 \leq \lambda_0 \sum_{i=0}^{N-1} |\alpha_{0i}|^2 \tag{4.40}$$

where the equality holds (i.e. $\mathbf{p}_0$ maximizes $\mathrm{E}[|\mathbf{p}_0^{\mathrm{H}}\mathbf{x}(n)|^2]$) only when $\alpha_{0i} = 0$, for $i = 1, 2, \ldots, N-1$. Furthermore, the fact the $\mathbf{p}_0$ is constrained to the length of unity implies that

$$\sum_{i=0}^{N-1} |\alpha_{0i}|^2 = 1. \tag{4.41}$$

Application of (4.39) and (4.41) in (4.40) gives

$$\max_{\|\mathbf{p}_0\|=1} \mathrm{E}[|\mathbf{p}_0^{\mathrm{H}}\mathbf{x}(n)|^2] = \lambda_0 \tag{4.42}$$

and this is achieved when

$$\mathbf{p}_0 = \alpha_{00}\mathbf{q}_0 \quad \text{with } |\alpha_{00}| = 1. \tag{4.43}$$

We may note that the factor α_{00} is arbitrary and has no significance since it does not affect the maximum in (4.42) because of the constraint $|\alpha_{00}| = 1$ in (4.43). Hence, without any loss of generality, we assume $\alpha_{00} = 1$. This gives

$$\mathbf{p}_0 = \mathbf{q}_0 \tag{4.44}$$

as a solution to the maximization problem

$$\max_{\|\mathbf{p}_0\|=1} \mathrm{E}[|\mathbf{p}_0^{\mathrm{H}}\mathbf{x}(n)|^2]. \tag{4.45}$$

The fact that the solution obtained here is not unique follows from the more general fact that the eigenvector corresponding to an eigenvalue is always arbitrary to the extent of a scalar multiplier factor. Here, the scalar multiplier is constrained to have a modulus of unity to satisfy the condition that both the $\mathbf{p}_i$ and $\mathbf{q}_i$ vectors are constrained to the length of unity.

In proceeding to find $\mathbf{p}_1$, we note that the constraint (4.30), for $i = 1$, implies that

$$\mathbf{p}_1^{\mathrm{H}}\mathbf{q}_0 = 0. \tag{4.46}$$

This in turn requires $\mathbf{p}_1$ to be limited to a linear combination of $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{N-1}$, only. That is,

$$\mathbf{p}_1 = \sum_{j=1}^{N-1} \alpha_{1,j}\mathbf{q}_j. \tag{4.47}$$

Noting this and following a procedure similar to the one used to find $\mathbf{p}_0$, we get

$$\max_{\|\mathbf{p}_1\|=1} \mathrm{E}[|\mathbf{p}_1^{\mathrm{H}}\mathbf{x}(n)|^2] = \lambda_1 \tag{4.48}$$

and

$$\mathbf{p}_1 = \mathbf{q}_1. \tag{4.49}$$

Following the same procedure for the rest of the eigenvalues and eigenvectors of $\mathbf{R}$ completes the proof of the first procedure of the minimax theorem.

The alternative procedure of the minimax theorem, suggested by (4.31)–(4.33), can also be proved is a similar way.

Property 8 *The eigenvalues of the correlation matrix* $\mathbf{R}$ *of a discrete-time stationary stochastic process* $\{x(n)\}$ *are bounded by the minimum and maximum values of the power spectral density,* $\Phi_{xx}(e^{j\omega})$, *of the process.*

The minimax theorem, as introduced in Property 7, views the eigenvectors of the correlation matrix of a discrete-time stochastic process as the conjugate of a set of tap-weight vectors corresponding to a set of FIR filters which are optimized in the minimax sense introduced there. Such filters are conveniently called *eigenfilters*. The minimax optimization procedure suggests that the eigenfilters may be obtained through a maximization or a minimization procedure that looks at the output powers of the eigenfilters. In particular, the maximum and minimum eigenvalues of $\mathbf{R}$ may be obtained by solving the following two independent problems, respectively:

$$\lambda_{\max} = \max_{\|\mathbf{q}_0\|=1} \mathrm{E}[|\mathbf{q}_0^{\mathrm{H}}\mathbf{x}(n)|^2] \tag{4.50}$$

and

$$\lambda_{\min} = \min_{\|\mathbf{q}_{N-1}\|=1} \mathrm{E}[|\mathbf{q}_{N-1}^{\mathrm{H}}\mathbf{x}(n)|^2]. \tag{4.51}$$

Let $Q_i(z)$ denote the system function of the ith eigenfilter of the discrete-time stochastic process $\{x(n)\}$. Using the Parseval's relation (equation (2.26) of Chapter 2), we obtain

$$\|\mathbf{q}_i\|^2 = \mathbf{q}_i^{\mathrm{H}}\mathbf{q}_i = \frac{1}{2\pi}\int_{-\pi}^{\pi} |Q_i(e^{j\omega})|^2 \, d\omega. \tag{4.52}$$

With the constraint $\|\mathbf{q}_i\| = 1$, this gives

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} |Q_i(e^{j\omega})|^2 \, d\omega = 1. \tag{4.53}$$

On the other hand, if we define $x_i'(n)$ as the output of the ith eigenfilter of $\mathbf{R}$, i.e.

$$x_i'(n) = \mathbf{q}_i^{\mathrm{H}}\mathbf{x}(n), \tag{4.54}$$

then, using the power spectral density relationships provided in Chapter 2, we obtain

$$\Phi_{x_i'x_i'}(e^{j\omega}) = |Q_i(e^{j\omega})|^2 \Phi_{xx}(e^{j\omega}). \tag{4.55}$$

We may also recall from the results presented in Chapter 2 that

$$\mathrm{E}[|x_i'(n)|^2] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_{x_i'x_i'}(e^{j\omega}) \, d\omega. \tag{4.56}$$

Substituting (4.55) in (4.56) we obtain

$$\mathrm{E}[|x_i'(n)|^2] = \frac{1}{2\pi}\int_{-\pi}^{\pi} |Q_i(e^{j\omega})|^2 \Phi_{xx}(e^{j\omega}) \, d\omega. \tag{4.57}$$

This result has the following interpretation. *The signal power at the output of the i th eigenfilter of the correlation matrix* $\mathbf{R}$ *of a stochastic process* $\{x(n)\}$ *is given by a weighted average of the power spectral density of* $\{x(n)\}$. *The weighting function used for averaging is the squared magnitude response of the corresponding eigenfilter.*

Using the above results, (4.50) may be written as

$$\lambda_{\max} = \max \frac{1}{2\pi} \int_{-\pi}^{\pi} |Q_0(e^{j\omega})|^2 \Phi_{xx}(e^{j\omega})\, d\omega \tag{4.58}$$

subject to the constraint

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} |Q_0(e^{j\omega})|^2\, d\omega = 1. \tag{4.59}$$

We may also note that

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} |Q_0(e^{j\omega})|^2 \Phi_{xx}(e^{j\omega})\, d\omega \leq \Phi_{xx}^{\max} \frac{1}{2\pi} \int_{-\pi}^{\pi} |Q_0(e^{j\omega})|^2\, d\omega, \tag{4.60}$$

where

$$\Phi_{xx}^{\max} \triangleq \max_{-\pi \leq \omega \leq \pi} \Phi_{xx}(e^{j\omega}). \tag{4.61}$$

With the constraint (4.59), (4.60) simplifies to

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} |Q_0(e^{j\omega})|^2 \Phi_{xx}(e^{j\omega})\, d\omega \leq \Phi_{xx}^{\max}. \tag{4.62}$$

Using (4.62) in (4.58) we obtain

$$\lambda_{\max} \leq \Phi_{xx}^{\max}. \tag{4.63}$$

Following a similar procedure, we may also find that

$$\lambda_{\min} \geq \Phi_{xx}^{\min}, \tag{4.64}$$

where

$$\Phi_{xx}^{\min} \triangleq \min_{-\pi \leq \omega \leq \pi} \Phi_{xx}(e^{j\omega}). \tag{4.65}$$

Property 9 *Let* $\mathbf{x}(n)$ *be an observation vector with the correlation matrix* $\mathbf{R}$. *Assume that* $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{N-1}$ *are a set of orthogonal eigenvectors of* $\mathbf{R}$ *and the matrix* $\mathbf{Q}$ *is defined as in* (4.16). *Then, the elements of the vector*

$$\mathbf{x}'(n) = \mathbf{Q}^{\mathrm{H}} \mathbf{x}(n) \tag{4.66}$$

constitute a set of uncorrelated random variables. The transformation defined by (4.66) *is called the Karhunen–Loéve transform.*

Using (4.66) we obtain

$$\mathrm{E}[\mathbf{x}'(n)\mathbf{x}'^{\mathrm{H}}(n)] = \mathbf{Q}^{\mathrm{H}}\mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{H}}(n)]\mathbf{Q} = \mathbf{Q}^{\mathrm{H}}\mathbf{R}\mathbf{Q}. \tag{4.67}$$

Substituting for $\mathbf{R}$ from (4.19) and assuming that the eigenvectors $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{N-1}$ are normalized to the length of unity[2] so that $\mathbf{Q}^{\mathrm{H}}\mathbf{Q} = \mathbf{I}$, we obtain

$$\mathrm{E}[\mathbf{x}'(n)\mathbf{x}'^{\mathrm{H}}(n)] = \mathbf{\Lambda}. \tag{4.68}$$

Noting that $\mathbf{\Lambda}$ is a diagonal matrix, this clearly shows that the elements of $\mathbf{x}'(n)$ are uncorrelated with one another.

It is worth noting that the ith element of $\mathbf{x}'(n)$ is the output of the ith eigenfilter of the correlation matrix of the process $\{x(n)\}$, i.e. the variable $x_i'(n)$ as defined by (5.54). Thus, an alternative way of stating Property 9 is to say that *the eigenfilters associated with a process $x(n)$ may be selected so that their output samples, at any time instant n, constitute a set of mutually orthogonal random variables.*

It may also be noted that by premultiplying (4.66) with $\mathbf{Q}$ and using $\mathbf{Q}\mathbf{Q}^{\mathrm{H}} = \mathbf{I}$, we obtain

$$\mathbf{x}(n) = \mathbf{Q}\mathbf{x}'(n). \tag{4.69}$$

Replacing $\mathbf{x}'(n)$ by the column vector $[x_0'(n)\ x_1'(n)\ \ldots\ x_{N-1}'(n)]^{\mathrm{T}}$, and expanding (4.69) in terms of the elements of $\mathbf{x}'(n)$ and columns of $\mathbf{Q}$, we get

$$\mathbf{x}(n) = \sum_{i=0}^{N-1} x_i'(n)\mathbf{q}_i. \tag{4.70}$$

This is known as the *Karhunen–Loéve expansion.*

Example 4.1

Consider a stationary random process $\{x(n)\}$ that is generated by passing a real-valued stationary zero-mean, unit-variance, white noise process $\{\nu(n)\}$ through a system with the system function

$$H(z) = \frac{\sqrt{1-\alpha^2}}{1-\alpha z^{-1}}, \tag{4.71}$$

where α is a real-valued constant in the range -1 to $+1$. We want to verify some of the results developed above for the process $\{x(n)\}$.

We note that for the unit-variance white noise process $\{\nu(n)\}$

$$\Phi_{\nu\nu}(z) = 1.$$

[2]This is not necessary for the above property to hold. However, it is a useful assumption as it simplifies our discussion.

Also, using (2.80) of Chapter 2 and noting that α is real-valued, we obtain

$$\Phi_{xx}(z) = H(z)H(z^{-1})\Phi_{\nu\nu}(z) = \frac{1-\alpha^2}{(1-\alpha z^{-1})(1-\alpha z)}. \tag{4.72}$$

Taking an inverse z-transform, we get

$$\phi_{xx}(k) = \alpha^{|k|}, \quad \text{for } k = \ldots -2, -1, 0, 1, 2, \ldots. \tag{4.73}$$

Using this result, we find that the correlation matrix of an N-tap transversal filter with input $\{x(n)\}$ is

$$\mathbf{R} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{N-1} \\ \alpha & 1 & \alpha & \cdots & \alpha^{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha^{N-1} & \alpha^{N-2} & \alpha^{N-3} & \cdots & 1 \end{bmatrix}. \tag{4.74}$$

Next, we present some numerical results that demonstrate the relationships between the power spectral density of the process $\{x(n)\}$, $\Phi_{xx}(e^{j\omega})$, and its corresponding correlation matrix.

Figure 4.1 shows a set of the plots of $\Phi_{xx}(e^{j\omega})$ for values of $\alpha = 0$, 0.5, and 0.75. We note that $\alpha = 0$ corresponds to the case where $\{x(n)\}$ is white and, therefore, its power spectral density is flat. As α increases from 0 to 1, $\{x(n)\}$ becomes more coloured and for values of α close to 1, most of its energy is concentrated around $\omega = 0$.

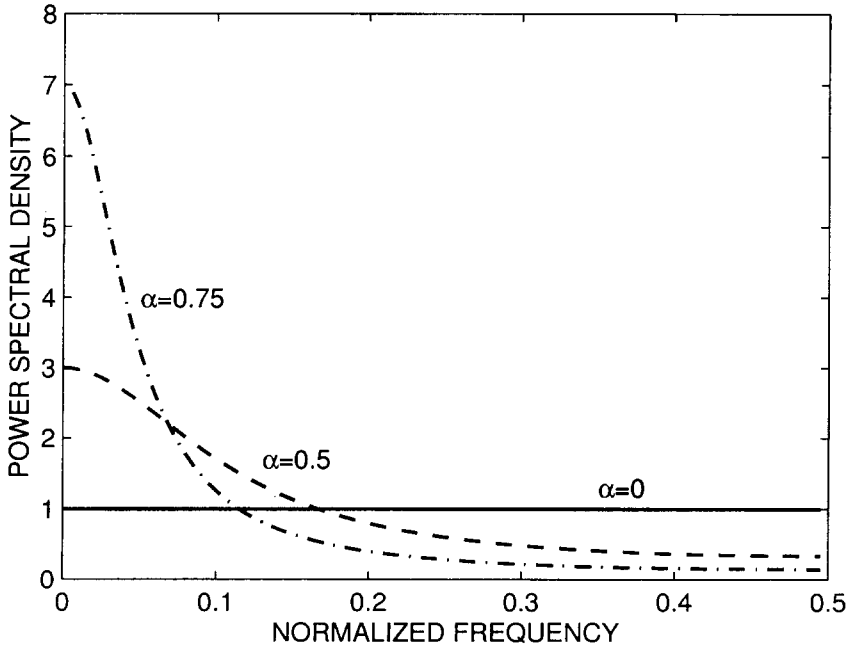


Figure 4.1 Power spectral density of $\{x(n)\}$ for different values of the parameter α

From Property 8 we recall that the eigenvalues of the correlation matrix $\mathbf{R}$ are bounded by the minimum and maximum values of $\Phi_{xx}(e^{j\omega})$. To illustrate this, in Figures 4.2(a), (b) and (c) we have plotted the minimum and maximum eigenvalues of $\mathbf{R}$ for values of $\alpha = 0.5, 0.75$ and 0.9, as N varies from 2 to 20. It may be noted that the limits predicted by the minimum and maximum values of $\Phi_{xx}(e^{j\omega})$ are achieved asymptotically as N increases. However, for values of α close to one, such limits are approached only when N is very large. This may be explained using the concept of eigenfilters. We note that when α is close to one, the peak of the power spectral density function $\Phi_{xx}(e^{j\omega})$ is very narrow; see the case of $\alpha = 0.75$ in Figure 4.1. To pick up this peak accurately, an eigenfilter with a very narrow pass-band (i.e. high selectivity) is required. On the other hand, a narrow-band filter can be realized only if the filter length, N, is selected long enough.

Example 4.2

Consider the case where the input process, $\{x(n)\}$, to an N-tap transversal filter consists of the summation of a zero-mean, white noise process, $\{\nu(n)\}$, and a complex sinusoid, $\{e^{j(\omega_0 n+\theta)}\}$, where θ is an initial random phase which varies for different realizations of the process. The correlation matrix of $\{x(n)\}$ is

$$\mathbf{R} = \sigma_\nu^2 \mathbf{I} + \begin{bmatrix} 1 & e^{j\omega_0} & \cdots & e^{j(N-1)\omega_0} \\ e^{-j\omega_0} & 1 & \cdots & e^{j(N-2)\omega_0} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-j(N-1)\omega_0} & e^{-j(N-2)\omega_0} & \cdots & 1 \end{bmatrix}, \tag{4.75}$$

where the first term on the right-hand side is the correlation matrix of the white noise process and the second term is that of the sinusoidal process. We are interested in finding the eigenvalues and eigenvectors of $\mathbf{R}$. These are conveniently obtained through the minimax theorem and the concept of eigenfilters.

Figure 4.3 shows the power spectral density of the process $\{x(n)\}$. It consists of a flat level which is contributed by $\{\nu(n)\}$ and an impulse at $\omega = \omega_0$ due to the sinusoidal part of $\{x(n)\}$. The eigenfilter that picks up maximum energy of the input is the one that is matched to the sinusoidal part of the input. The coefficients of this filter are the elements of the eigenfilter

$$\mathbf{q}_0 = \frac{1}{\sqrt{N}}[1 \quad e^{-j\omega_0} \quad \cdots \quad e^{-j(N-1)\omega_0}]^{\mathrm{T}}. \tag{4.76}$$

The factor $1/\sqrt{N}$ in (4.76) is to normalize $\mathbf{q}_0$ to the length of unity. The vector $\mathbf{q}_0$ can easily be confirmed to be an eigenvector of $\mathbf{R}$ by evaluating $\mathbf{R}\mathbf{q}_0$ and noting that this gives

$$\mathbf{R}\mathbf{q}_0 = (\sigma_\nu^2 + N)\mathbf{q}_0. \tag{4.77}$$

This also shows that the eigenvalue corresponding to the eigenvector $\mathbf{q}_0$ is

$$\lambda_0 = \sigma_\nu^2 + N. \tag{4.78}$$

Also, from the minimax theorem, we note that the rest of the eigenvectors of $\mathbf{R}$ have to be orthogonal to $\mathbf{q}_0$, i.e.

$$\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_0 = 0, \quad \text{for } i = 1, 2, \ldots, N-1. \tag{4.79}$$

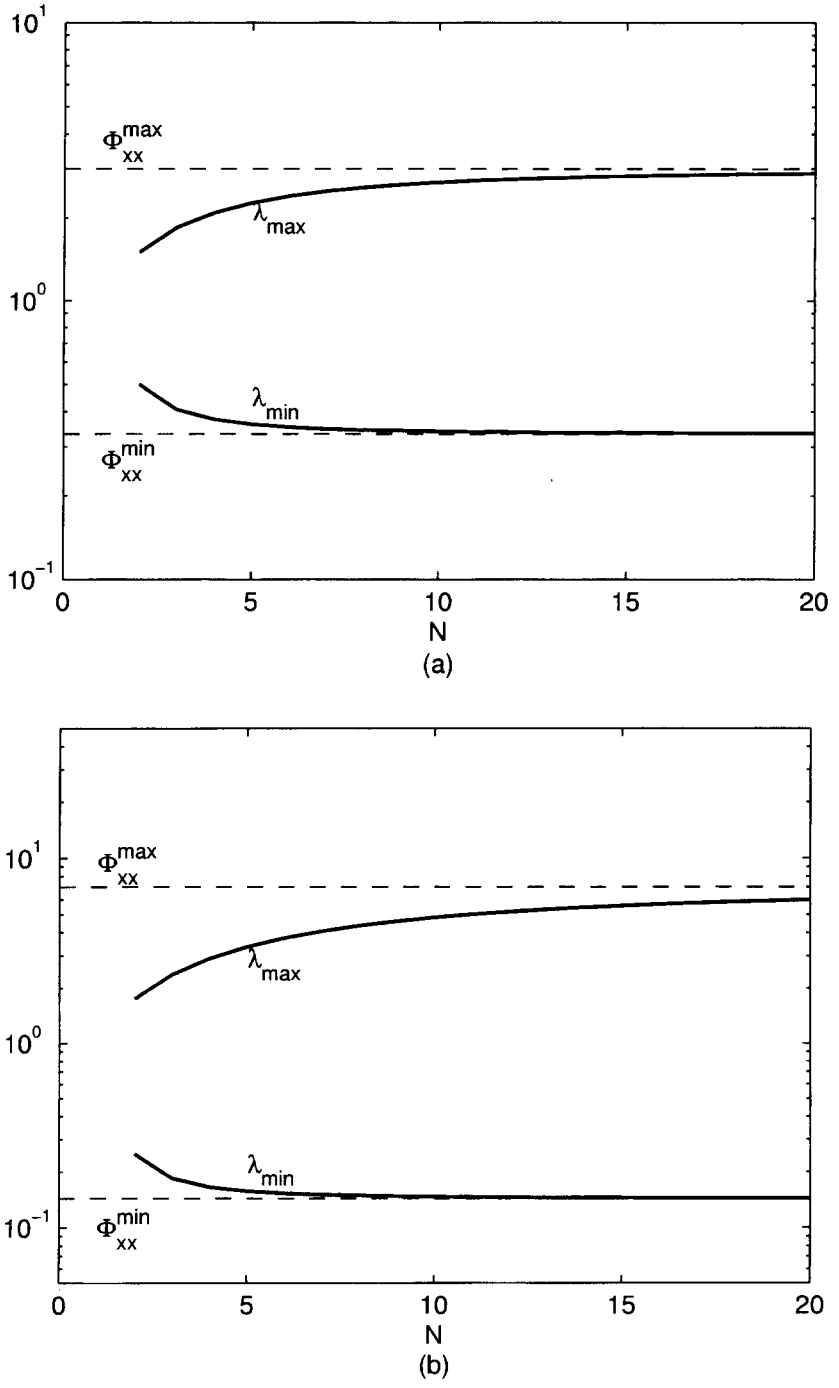


Figure 4.2 Minimum and maximum eigenvalues of the correlation matrix for different values of the parameter α: (a) $\alpha = 0.5$, (b) $\alpha = 0.75$, (c) $\alpha = 0.9$

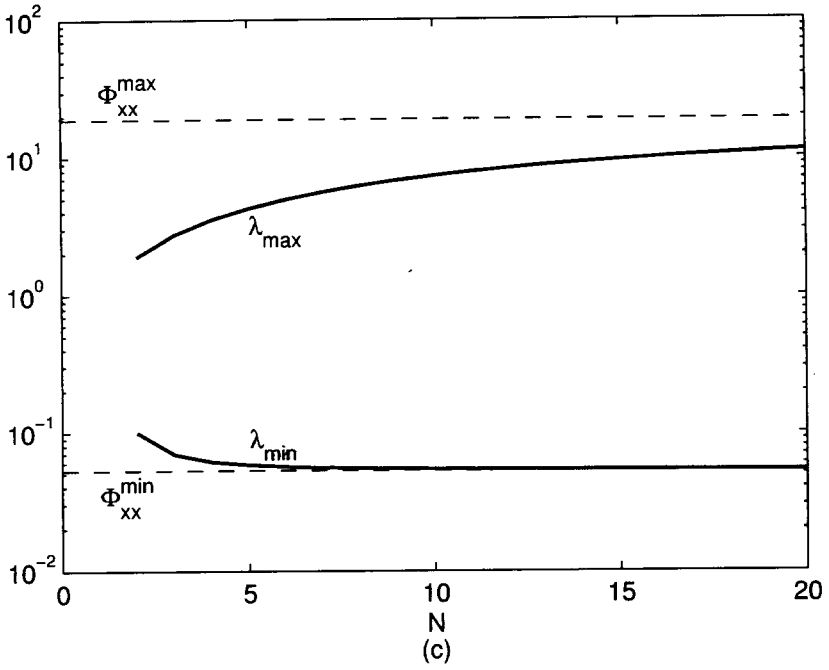


Figure 4.2 Continued

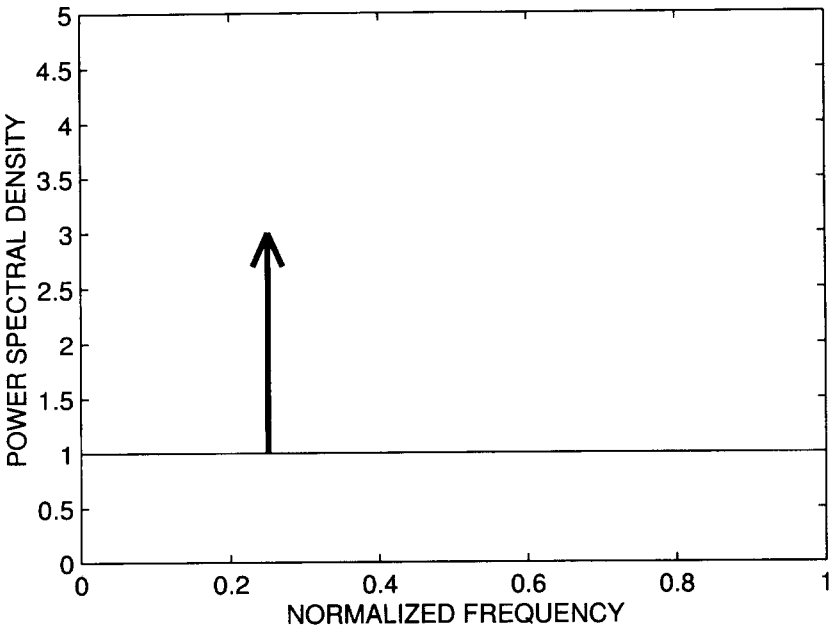


Figure 4.3 Power spectral density of the process $\{x(n)\}$ consisting of a white noise plus a single tone sinusoidal signal

Using this, it is not difficult (see Problem P4.7) to show that

$$\mathbf{R}\mathbf{q}_i = \sigma_v^2 \mathbf{q}_i, \quad \text{for } i = 1, 2, \ldots, N-1. \tag{4.80}$$

This result shows that as long as (4.79) hold, the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{N-1}$ of $\mathbf{R}$ are arbitrary. In other words, any set of vectors which belongs to the subspace orthogonal to the eigenvector $\mathbf{q}_0$ makes an acceptable set for the rest of the eigenvectors of $\mathbf{R}$. Furthermore, the eigenvalues corresponding to these eigenvectors are all equal to σ_v^2.

4.3 The Performance Surface

With the background developed so far, we are now ready to proceed with exploring the performance surface of transversal Wiener filters. We start with the case where the filter coefficients, input and desired output are real-valued. The results will then be extended to the complex-valued case.

We recall from Chapter 3 that the performance function of a transversal Wiener filter with a real-valued input sequence $x(n)$ and a desired output sequence $d(n)$ is

$$\xi = \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - 2\mathbf{p}^{\mathrm{T}}\mathbf{w} + \mathrm{E}[d^2(n)] \tag{4.81}$$

where the superscript T denotes vector or matrix transpose, $\mathbf{w} = [w_0 \;\; w_1 \;\; \cdots \;\; w_{N-1}]^{\mathrm{T}}$ is the filter tap-weight vector, $\mathbf{R} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]$ is the correlation matrix of the filter tap-input vector $\mathbf{x}(n) = [x(n) \;\; x(n-1) \;\; \cdots \;\; x(n-N+1)]^{\mathrm{T}}$, and $\mathbf{p} = \mathrm{E}[d(n)\mathbf{x}(n)]$ is the cross-correlation vector between $d(n)$ and $\mathbf{x}(n)$. We want to study the shape of the performance function ξ when it is viewed as a surface in the $(N+1)$-dimensional Euclidian space constituted by the filter tap weights $w_i, i = 0, 1, \ldots, N-1$, and the performance function, ξ.

Also, we recall that the optimum value of the Wiener filter tap-weight vector is obtained from the Wiener–Hopf equation

$$\mathbf{R}\mathbf{w}_{\mathrm{o}} = \mathbf{p}. \tag{4.82}$$

The performance function ξ may be rearranged as follows:

$$\xi = \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\mathbf{p} - \mathbf{p}^{\mathrm{T}}\mathbf{w} + \mathrm{E}[d^2(n)] \tag{4.83}$$

where we have noted that $\mathbf{w}^{\mathrm{T}}\mathbf{p} = \mathbf{p}^{\mathrm{T}}\mathbf{w}$. Next, we substitute for $\mathbf{p}$ in (4.83) from (4.82) and add and subtract the term $\mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{R}\mathbf{w}_{\mathrm{o}}$ to obtain

$$\xi = \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w}_{\mathrm{o}} - \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{w} + \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{R}\mathbf{w}_{\mathrm{o}} + \mathrm{E}[d^2(n)] - \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{R}\mathbf{w}_{\mathrm{o}}. \tag{4.84}$$

Since $\mathbf{R}^{\mathrm{T}} = \mathbf{R}$, the first four terms on the right-hand side of (4.84) can be combined to obtain

$$\xi = (\mathbf{w} - \mathbf{w}_{\mathrm{o}})^{\mathrm{T}}\mathbf{R}(\mathbf{w} - \mathbf{w}_{\mathrm{o}}) + \mathrm{E}[d^2(n)] - \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{R}\mathbf{w}_{\mathrm{o}}. \tag{4.85}$$

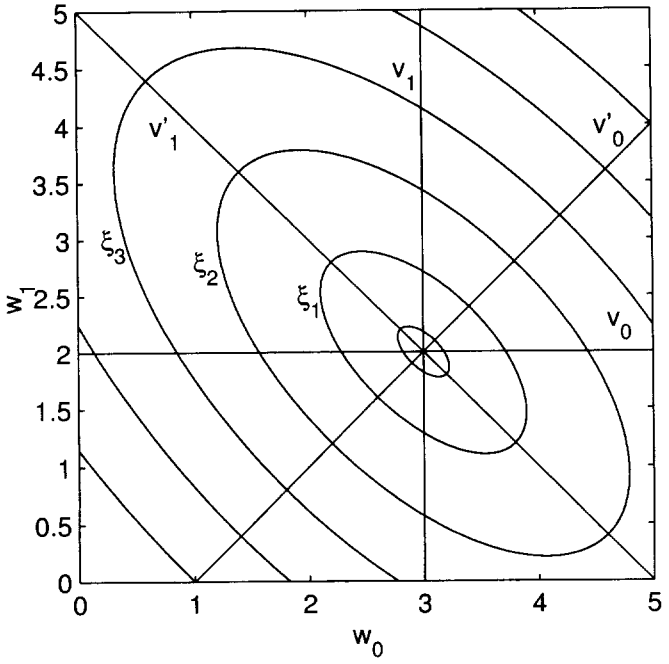


Figure 4.4 A typical performance surface of a two-tap transversal filter

We may also recall from Chapter 3 that

$$\xi_{\min} = \mathrm{E}[d^2(n)] - \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{R}\mathbf{w}_{\mathrm{o}}, \tag{4.86}$$

where $\xi_{\min}$ is the minimum value of ξ which is obtained when $\mathbf{w} = \mathbf{w}_{\mathrm{o}}$. Substituting (4.86) in (4.85), we get

$$\xi = \xi_{\min} + (\mathbf{w} - \mathbf{w}_{\mathrm{o}})^{\mathrm{T}}\mathbf{R}(\mathbf{w} - \mathbf{w}_{\mathrm{o}}). \tag{4.87}$$

This result has the following interpretation. The non-negative definiteness of the correlation matrix $\mathbf{R}$ implies that the second term on the right-hand side of (4.87) is non-negative. When $\mathbf{R}$ is positive definite (a case very likely to happen in practice), the second term on the right-hand side of (4.87) is zero only when $\mathbf{w} = \mathbf{w}_{\mathrm{o}}$, and in that case ξ coincides with its minimum value. This is depicted in Figure 4.4 where a typical performance surface of a two-tap Wiener filter is presented by a set of contours which correspond to different levels of ξ, and

$$\xi_{\min} < \xi_1 < \xi_2 < \cdots.$$

To proceed further, we define the vector

$$\mathbf{v} \triangleq \mathbf{w} - \mathbf{w}_{\mathrm{o}} \tag{4.88}$$

and substitute it in (4.87) to obtain

$$\xi = \xi_{\min} + \mathbf{v}^{\mathrm{T}}\mathbf{R}\mathbf{v}. \tag{4.89}$$

This simpler form of the performance function in effect is equivalent to shifting the origin of the N-dimensional Euclidian space defined by the elements of $\mathbf{w}$ to the point $\mathbf{w} = \mathbf{w}_{\mathrm{o}}$. The new Euclidian space has a new set of axes given by $v_0, v_1, \ldots, v_{N-1}$; see Figure 4.4. These are in parallel with the original axes $w_0, w_1, \ldots, w_{N-1}$. Obviously, the shape of the performance surface is not affected by the shift in the origin.

To simplify (4.89) further, we use the unitary similarity transformation, i.e. (4.19) of the previous section, which for real-valued signals is written as

$$\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{T}}. \tag{4.90}$$

Substituting (4.90) in (4.89) we obtain

$$\xi = \xi_{\min} + \mathbf{v}^{\mathrm{T}}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{T}}\mathbf{v}. \tag{4.91}$$

We define

$$\mathbf{v}' = \mathbf{Q}^{\mathrm{T}}\mathbf{v} \tag{4.92}$$

and note that multiplication of the vector $\mathbf{v}$ by the unitary matrix $\mathbf{Q}^{\mathrm{T}}$ is equivalent to rotating the v-axes to a new set of axes given by $v'_0, v'_1, \ldots, v'_{N-1}$, as depicted in Figure 4.4. The new axes are in the directions specified by the rows of the transformation matrix $\mathbf{Q}^{\mathrm{T}}$. We may further note that the rows of $\mathbf{Q}^{\mathrm{T}}$ are the eigenvectors of the correlation matrix $\mathbf{R}$. This means that the v'-axes, defined by (4.92), are in the directions of the basis vectors specified by the eigenvectors of $\mathbf{R}$.

Substituting (4.92) in (4.91) we obtain

$$\xi = \xi_{\min} + \mathbf{v}'^{\mathrm{T}}\mathbf{\Lambda}\mathbf{v}'. \tag{4.93}$$

This in known as the *canonical form* of the performance function. Expanding (4.93) in terms of the elements of the vector $\mathbf{v}'$ and the diagonal elements of the matrix Λ, we get

$$\xi = \xi_{\min} + \sum_{i=0}^{N-1} \lambda_i v_i'^2. \tag{4.94}$$

This, when compared with the previous forms of the performance function in (4.81) and (4.89), is a much easier function to visualize. In particular, if all the variables $v'_0, v'_1, \ldots, v'_{N-1}$, except v'_k, are set to zero, then

$$\xi = \xi_{\min} + \lambda_k v_k'^2. \tag{4.95}$$

This is a parabola whose minimum occurs at $v'_k = 0$. The parameter λ_k determines the shape of the parabola, in the sense that for smaller values of λ_k the resulting parabolas

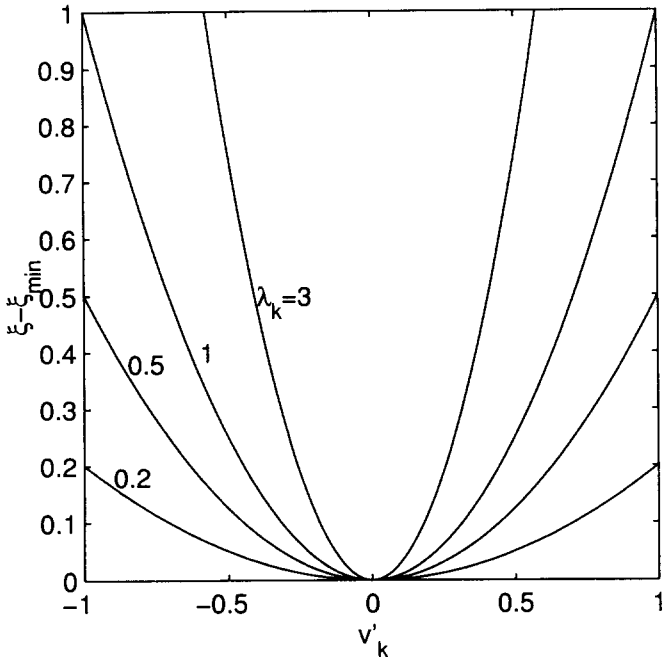


Figure 4.5 The effect of eigenvalues on the shape of the performance function when only one of the filter tap weights is varied

are wider (flatter in shape) when compared with those obtained for larger values of λ_k. This is demonstrated if Figure 4.5 where ξ, as a function of v_k', is plotted for a few values of λ_k.

When all variables $v_0', v_1', \ldots, v_{N-1}'$ are varied simultaneously, the performance function ξ, in the $(N+1)$-dimensional Euclidian space, is a hyperparabola. The path traced by ξ as we move along any of the axes $v_0', v_1', \ldots, v_{N-1}'$ is a parabola whose shape is determined by the corresponding eigenvalue.

The hyperparabola shape of the performance surface can be best understood in the case of a two-tap filter when the performance surface can easily be visualized in the 3-dimensional Euclidian space whose axes are the two independent taps of the filter and the function ξ; see Figure 3.4 as an example. Alternatively, the contour plots, such as those presented in Figure 4.4, may be used to visualize the performance surface in a very convenient way.

For $N = 2$, the canonical form of the performance function is

$$\xi = \xi_{\min} + \lambda_0 v_0'^2 + \lambda_1 v_1'^2. \tag{4.96}$$

This may be rearranged as

$$\left(\frac{v_0'}{a_0}\right)^2 + \left(\frac{v_1'}{a_1}\right)^2 = 1 \tag{4.97}$$

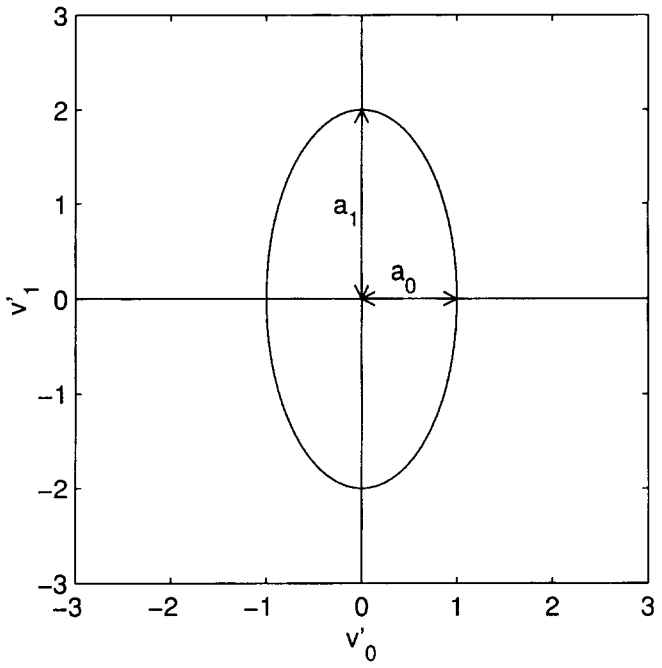


Figure 4.6 A typical plot of the ellipse defined by (4.97)

where

$$a_0 = \sqrt{\frac{\xi - \xi_{\min}}{\lambda_0}} \tag{4.98}$$

and

$$a_1 = \sqrt{\frac{\xi - \xi_{\min}}{\lambda_1}}. \tag{4.99}$$

Equation (4.97) represents an ellipse whose principal axes are along v_0' and v_1', and for $a_1 > a_0$, the lengths of its major and minor principal axes are $2a_1$ and $2a_0$, respectively. These are highlighted in Figure 4.6, where a typical plot of the ellipse defined by (4.97) is presented. We may also note that $a_1/a_0 = \sqrt{\lambda_0/\lambda_1}$. This implies that for a particular performance surface the aspect ratio of the contour ellipses is fixed and is equal to the square root of the ratio of its eigenvalues. In other words, the eccentricity of the contour ellipses of a performance surface is determined by the ratio of the eigenvalues of the corresponding correlation matrix. A larger ratio of the eigenvalues results in more eccentric ellipses and, thus, a narrower bowl-shape performance surface.

Example 4.3

Consider the case where a two-tap transversal Wiener filter is characterized by the following parameters:

$$\mathbf{R} = \begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \text{and } \mathrm{E}[d^2(n)] = 2.$$

We want to explore the performance surface of this filter for values of α ranging from 0 to 1.

The performance function of the filter is obtained by substituting the above parameters in (4.81). This gives

$$\xi = [w_0 \; w_1]\begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix}\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} - 2[1 \;\; 1]\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} + 2. \tag{4.100}$$

Solving the Wiener–Hopf equation to obtain the optimum tap weights of the filter, we obtain

$$\begin{bmatrix} w_{o,0} \\ w_{o,1} \end{bmatrix} = \mathbf{R}^{-1}\mathbf{p} = \begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix}^{-1}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \dfrac{1}{1+\alpha} \\ \dfrac{1}{1+\alpha} \end{bmatrix}. \tag{4.101}$$

Using this result, we get

$$\begin{aligned} \xi_{\min} &= \mathrm{E}[d^2(n)] - \mathbf{w}_o^{\mathrm{T}}\mathbf{p} \\ &= 2 - \begin{bmatrix} \dfrac{1}{1+\alpha} & \dfrac{1}{1+\alpha} \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{2\alpha}{1+\alpha}. \end{aligned} \tag{4.102}$$

Also,

$$\begin{aligned} \xi &= \xi_{\min} + (\mathbf{w} - \mathbf{w}_o)^{\mathrm{T}}\mathbf{R}(\mathbf{w} - \mathbf{w}_o) \\ &= \frac{2\alpha}{1+\alpha} + [v_0 \; v_1]\begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix}\begin{bmatrix} v_0 \\ v_1 \end{bmatrix}. \end{aligned} \tag{4.103}$$

To convert this to its canonical form, we should first find the eigenvalues and eigenvectors of $\mathbf{R}$. To find the eigenvalues of $\mathbf{R}$, we should solve the characteristic equation

$$\det(\lambda\mathbf{I} - \mathbf{R}) = \begin{vmatrix} \lambda - 1 & -\alpha \\ -\alpha & \lambda - 1 \end{vmatrix} = 0. \tag{4.104}$$

Expanding (4.104), we obtain

$$(\lambda - 1)^2 - \alpha^2 = 0,$$

which gives

$$\lambda_0 = 1 + \alpha \tag{4.105}$$

and

$$\lambda_1 = 1 - \alpha. \tag{4.106}$$

The eigenvectors $\mathbf{q}_0 = [q_{00} \;\; q_{01}]^T$ and $\mathbf{q}_1 = [q_{10} \;\; q_{11}]^T$ of $\mathbf{R}$ are obtained by solving the equations

$$\begin{bmatrix} \lambda_0 - 1 & -\alpha \\ -\alpha & \lambda_0 - 1 \end{bmatrix} \begin{bmatrix} q_{00} \\ q_{01} \end{bmatrix} = 0 \tag{4.107}$$

and

$$\begin{bmatrix} \lambda_1 - 1 & -\alpha \\ -\alpha & \lambda_1 - 1 \end{bmatrix} \begin{bmatrix} q_{10} \\ q_{11} \end{bmatrix} = 0. \tag{4.108}$$

Substituting (4.105) and (4.106) in (4.107) and (4.108), respectively, we obtain

$$q_{00} = q_{01} \quad \text{and} \quad q_{10} = -q_{11}.$$

Using these results and normalizing $\mathbf{q}_0$ and $\mathbf{q}_1$ to have lengths of unity, we obtain

$$\mathbf{q}_0 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{and} \quad \mathbf{q}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

It may be noted that the eigenvectors $\mathbf{q}_0$ and $\mathbf{q}_1$ of $\mathbf{R}$ are independent of the parameter α. This is an interesting property of the correlation matrices of two-tap transversal filters which implies that the v'-axes are always obtained by a 45 degree rotation of the v-axes. The eigenvectors associated with the correlation matrices of three-tap transversal filters also have some special form. This is discussed in Problem P4.5.

With the above results, we get

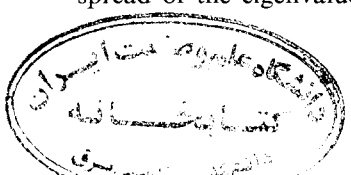
$$\begin{bmatrix} v_0' \\ v_1' \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} v_0 + v_1 \\ v_0 - v_1 \end{bmatrix} \tag{4.109}$$

and

$$\xi = \frac{2\alpha}{1+\alpha} + (1+\alpha)v_0'^2 + (1-\alpha)v_1'^2. \tag{4.110}$$

Figures 4.7(a), (b), and (c) show the contour plots of the performance surface of the two-tap transversal filter for $\alpha = 0.5, 0.8$ and 0.95, which correspond to the eigenvalue ratios of 3, 9 and 39, respectively. These plots clearly show how the eccentricity of the performance surface changes as the eigenvalue ratio of the correlation matrix $\mathbf{R}$ increases.

The above results may be generalized as follows. The performance surface of an N-tap transversal filter with real-valued data is a hyperparaboloid in the $(N+1)$-dimensional Euclidian space whose axes are the N tap-weight variables of the filter and the performance function ξ. The performance function may also be represented by a set of hyperellipses in the N-dimensional Euclidian space of the filter tap-weight variables. Each hyperellipse corresponds to a fixed value of ξ. The directions of the principal axes of the hyperellipses are determined by the eigenvectors of the correlation matrix $\mathbf{R}$. The size of the various principal axes of each hyperellipse are proportional to the square root of the inverse of the corresponding eigenvalues. Thus, the eccentricity of the hyperellipses is determined by the spread of the eigenvalues of the correlation matrix $\mathbf{R}$. This shows that the shape of the performance surface of a Wiener FIR filter is directly related to the spread of the eigenvalues of $\mathbf{R}$. In addition, from Property 8 of the eigenvalues and



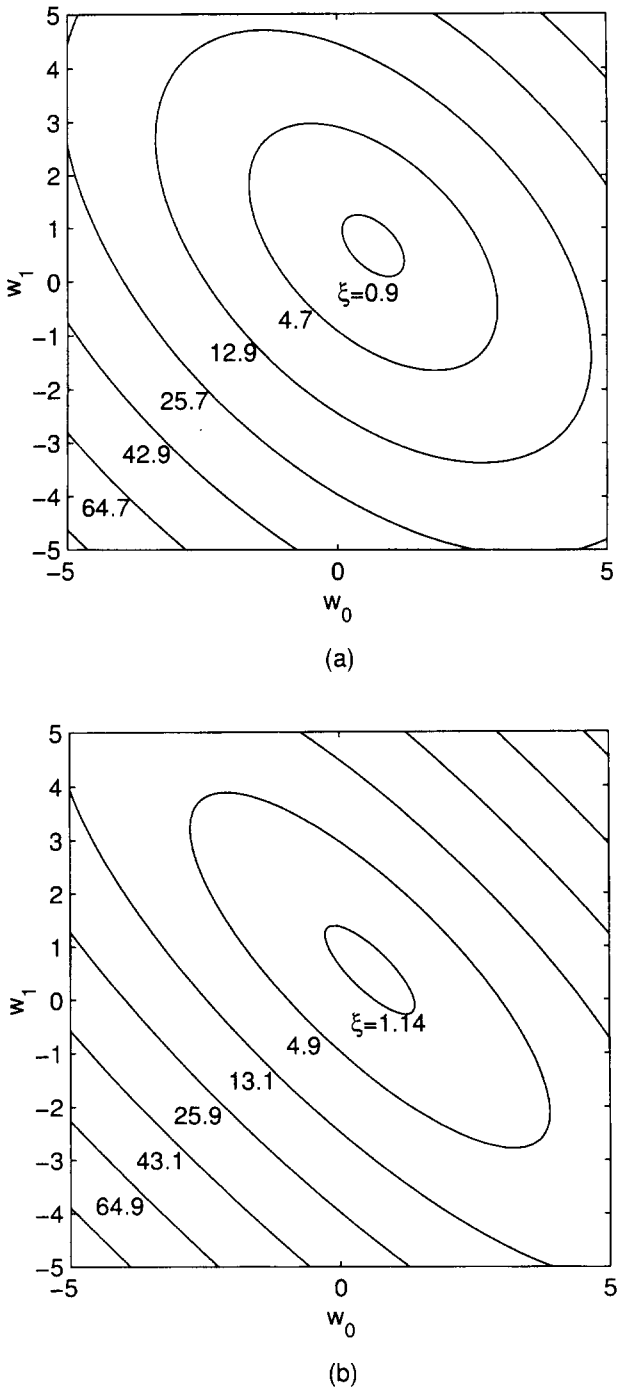
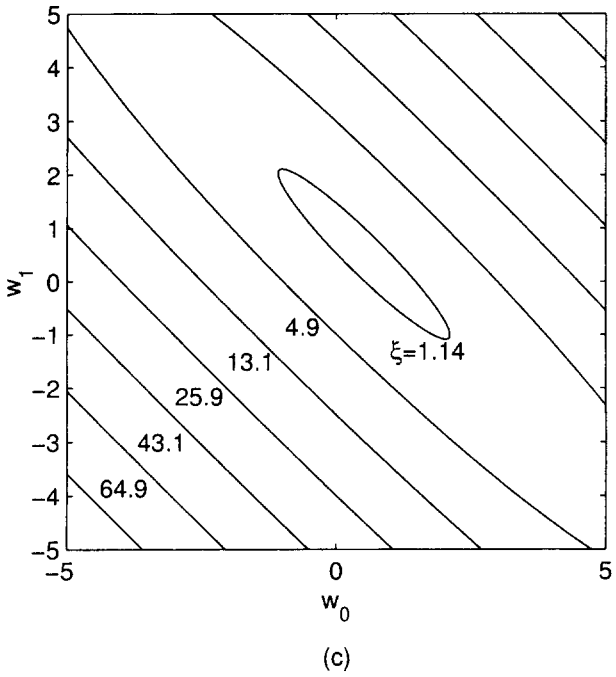


Figure 4.7 Performance surface of a two-tap transversal filter for different eigenvalue spread of $\mathbf{R}$: (a) $\lambda_0/\lambda_1 = 3$, (b) $\lambda_0/\lambda_1 = 9$, (c) $\lambda_0/\lambda_1 = 39$



(c)

Figure 4.7 Continued

eigenvectors, we recall that the spread of the eigenvalues of the correlation matrix of a stochastic process $\{x(n)\}$ is directly linked to the variation in the power spectral density function $\Phi_{xx}(e^{j\omega})$ of the process. This, in turn, means that there is a close relationship between the power spectral density of a random process and the shape of the performance surface of an FIR Wiener filter for which the latter is used as input.

The above results can easily be extended to the case where the filter coefficients, input and desired output are complex-valued. We should remember that the elements of all the involved vectors and matrices are complex-valued and replace all the transpose operators in the developed equations by Hermitian operators. Doing this, (4.93) becomes

$$\xi = \xi_{\min} + \mathbf{v}'^{\mathrm{H}} \mathbf{\Lambda} \mathbf{v}'. \tag{4.111}$$

This can be expanded as

$$\xi = \xi_{\min} + \sum_{i=0}^{N-1} \lambda_i |v_i'|^2. \tag{4.112}$$

The difference between this result and its dual (for the real-valued case) in (4.94) is an additional modulus sign on the v_i's, in (4.112). This, of course, is due to the fact that here the v_i's are complex-valued.

The performance function ξ of (4.112) may be thought of as a hyperparabola in the $(N+1)$-dimensional space whose first N axes are defined by the complex-valued

variables, the v_i's, and its $(N+1)$th axis is the real-valued performance function ξ. To prevent such a mixed domain and have a clearer picture of the performance surface in the case of complex signals, we may expand (4.112) further by replacing v_i' with $v_{i,\mathrm{R}}' + jv_{i,\mathrm{I}}'$, where $v_{i,\mathrm{R}}'$ and $v_{i,\mathrm{I}}'$ are the real and imaginary parts of v_i'. With this, we obtain

$$\xi = \xi_{\min} + \sum_{i=0}^{N-1} \lambda_i (v_{i,\mathrm{R}}'^2 + v_{i,\mathrm{I}}'^2). \tag{4.113}$$

Here, $v_{i,\mathrm{R}}'$ and $v_{i,\mathrm{I}}'$ are both real-valued variables. Equation (4.113) shows that *the performance surface of an N-tap transversal Wiener filter with complex-valued coefficients is a hyperparabola in the* $(2N+1)$*-dimensional Euclidian space of the variables consisting of the real and imaginary parts of the filter coefficients and the performance function.*

Problems

P4.1 Consider the performance function

$$\xi = w_0^2 + w_1^2 + w_0 w_1 - w_0 + w_1 + 1.$$

(i) Convert this to its canonical form.
(ii) Plot the set of contour ellipses of the performance surface of ξ for values of $\xi = 1, 2, 3$ and 4.

P4.2 $\mathbf{R}$ is a correlation matrix.

(i) Using the unitary similarity transformation, show that for any integer n

$$\mathbf{R}^n = \mathbf{Q}\boldsymbol{\Lambda}^n\mathbf{Q}^{\mathrm{H}}.$$

(ii) The matrix $\mathbf{R}^{1/2}$ with the property $\mathbf{R}^{1/2}\mathbf{R}^{1/2} = \mathbf{R}$ is defined as the square-root of $\mathbf{R}$. Show that

$$\mathbf{R}^{1/2} = \mathbf{Q}\boldsymbol{\Lambda}^{1/2}\mathbf{Q}^{\mathrm{H}}.$$

(iii) Show that the identity

$$\mathbf{R}^a = \mathbf{Q}\boldsymbol{\Lambda}^a\mathbf{Q}^{\mathrm{H}}$$

is valid for any rational number a.

P4.3 Consider the correlation matrix $\mathbf{R}$ of an $N \times 1$ observation vector $\mathbf{x}(n)$, and an arbitrary $N \times N$ unitary transformation matrix $\mathbf{U}$. Define the vector

$$\mathbf{x}_{\mathrm{U}}(n) = \mathbf{U}\mathbf{x}(n)$$

and its corresponding correlation matrix $\mathbf{R}_{\mathrm{U}} = \mathrm{E}[\mathbf{x}_{\mathrm{U}}(n)\mathbf{x}_{\mathrm{U}}^{\mathrm{H}}(n)]$.

(i) Show that $\mathbf{R}$ and $\mathbf{R}_U$ share the same set of eigenvalues.
(ii) Find an expression for the eigenvectors of $\mathbf{R}_U$ in terms of the eigenvectors of $\mathbf{R}$ and the transformation matrix $\mathbf{U}$.

P4.4 In Example 4.3 we noted that the eigenvectors of the correlation matrix of any two-tap transversal filter with real-valued input are fixed and are

$$\mathbf{q}_0 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

and

$$\mathbf{q}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Plot the magnitude responses of the eigenfilters defined by $\mathbf{q}_0$ and $\mathbf{q}_1$ and verify that $\mathbf{q}_0$ corresponds to a lowpass filter and $\mathbf{q}_1$ corresponds to a highpass one. How do you relate this observation with the minimax theorem?

P4.5 Consider the correlation matrix $\mathbf{R}$ of a three-tap transversal filter with a real-valued input $x(n)$.

(i) Show that when $\mathrm{E}[x^2(n)] = 1$, $\mathbf{R}$ has the form

$$\mathbf{R} = \begin{bmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 \\ \rho_2 & \rho_1 & 1 \end{bmatrix}.$$

(ii) Show that

$$\mathbf{q}_0 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

is an eigenvector of $\mathbf{R}$ and find its corresponding eigenvalue.
(iii) Show that the other eigenvectors of $\mathbf{R}$ are

$$\mathbf{q}_i = \frac{1}{\sqrt{2+\alpha_i^2}}\begin{bmatrix} 1 \\ \alpha_i \\ 1 \end{bmatrix}, \quad \text{for } i = 1, 2$$

where

$$\alpha_1 = \frac{-\rho_2/\rho_1 + \sqrt{(\rho_2/\rho_1)^2 + 8}}{2} \quad \text{and} \quad \alpha_2 = \frac{-\rho_2/\rho_1 - \sqrt{(\rho_2/\rho_1)^2 + 8}}{2}.$$

Find the eigenvalues that correspond to $\mathbf{q}_1$ and $\mathbf{q}_2$.

(iv) For the following numerical values plot the magnitude responses of the eigenfilters defined by $\mathbf{q}_0, \mathbf{q}_1$ and $\mathbf{q}_2$ and find that in all cases these correspond to bandpass, lowpass and highpass filters, respectively:

ρ_1	ρ_2
0.5	0.25
0.8	0.30
0.9	−0.4

How do you relate this observation to the minimax theorem?

P4.6 Consider the correlation matrix $\mathbf{R}$ of an observation vector $\mathbf{x}(n)$. Define the vector

$$\tilde{\mathbf{x}}(n) = \mathbf{R}^{-1/2}\mathbf{x}(n),$$

where $\mathbf{R}^{-1/2}$ is the inverse of $\mathbf{R}^{1/2}$, and $\mathbf{R}^{1/2}$ is defined as in Problem P4.2. Show that the correlation matrix of $\tilde{\mathbf{x}}(n)$ is the identity matrix.

P4.7 Consider the case discussed in Example 4.2, and the eigenvector $\mathbf{q}_0$ as defined by (4.76). Show that any vector $\mathbf{q}_i$ that is orthogonal to $\mathbf{q}_0$ (i.e. $\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_0 = 0$) is a solution to the equation

$$\mathbf{R}\mathbf{q}_i = \sigma_v^2\mathbf{q}_i.$$

P4.8 The determinant of an $N \times N$ matrix $\mathbf{A}$ can be obtained by iterating the equation

$$\det(\mathbf{A}) = \sum_{j=0}^{N-1} a_{0j}\,\mathrm{cof}_{0j}/(\mathbf{A}),$$

where a_{ij} is the ijth element of $\mathbf{A}$, and $\mathrm{cof}_{ij}(\mathbf{A})$ denotes the ijth cofactor of $\mathbf{A}$ which is defined as

$$\mathrm{cof}_{ij}(\mathbf{A}) = (-1)^{i+j}\det(\mathbf{A}_{ij})$$

where $\mathbf{A}_{ij}$ is the $(N-1) \times (N-1)$ matrix obtained by deleting the ith row and jth column of $\mathbf{A}$. This procedure is general and applicable to all square matrices. Use this procedure to show that

(i) equation (4.24) is a valid result for any arbitrary square matrix $\mathbf{A}$,
(ii) for any square matrix $\mathbf{A}$

$$\det(\mathbf{A}) = \prod_{i=0}^{N-1} \lambda_i,$$

where the λ_is are the eigenvalues of $\mathbf{A}$.

P4.9 Give a proof for the minimax procedure suggested by (4.31)–(4.33).

P4.10 Consider a filter whose input is the vector $\tilde{\mathbf{x}}(n)$, as defined in Problem P4.6, and its output is $y(n) = \tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}(n)$, where $\tilde{\mathbf{w}}$ is $N \times 1$ tap-weight vector of the filter. Discuss the shape of the performance surface of this filter.

P4.11 Work out the details of the derivation of (4.70).

P4.12 Give a detailed derivation of (4.111).

P4.13 The input process to an N-tap transversal filter is

$$x(n) = a_1 e^{j\omega_1 n} + a_2 e^{j\omega_2 n} + \nu(n)$$

where a_1 and a_2 are uncorrelated, complex-valued, zero-mean, random variables with variances σ_1^2 and σ_2^2, respectively, and $\{\nu(n)\}$ is a white noise process with variance unity.

(i) Derive an equation for the correlation matrix, $\mathbf{R}$, of the observation vector at the filter input.

(ii) Following an argument similar to the one in Example 4.2, show that the smallest $N-2$ eigenvalues of $\mathbf{R}$ are all equal to σ_ν^2.

(iii) Let

$$\mathbf{u}_0 = \frac{1}{\sqrt{N}}[1 \quad e^{-j\omega_1} \quad e^{-j2\omega_1} \quad \cdots \quad e^{-j(N-1)\omega_1}]^{\mathrm{T}}$$

and

$$\mathbf{u}_1 = \frac{1}{\sqrt{N}}[1 \quad e^{-j\omega_2} \quad e^{-j2\omega_2} \quad \cdots \quad e^{-j(N-1)\omega_2}]^{\mathrm{T}}.$$

Show that the eigenvectors corresponding to the largest two eigenvalues of $\mathbf{R}$ are

$$\mathbf{q}_0 = \alpha_{00}\mathbf{u}_0 + \alpha_{01}\mathbf{u}_1$$

and

$$\mathbf{q}_1 = \alpha_{10}\mathbf{u}_0 + \alpha_{11}\mathbf{u}_1$$

where $\alpha_{00}, \alpha_{01}, \alpha_{10}$ and α_{11} are a set coefficients to be found. Propose a minimax procedure for finding these coefficients.

(iv) Find the coefficients $\alpha_{00}, \alpha_{01}, \alpha_{10}$ and α_{11} of part (iii) in the case where $\mathbf{u}_0^{\mathrm{H}}\mathbf{u}_1 = 0$. Discuss the uniqueness of the answer in the cases where $\sigma_1^2 \neq \sigma_2^2$ and $\sigma_1^2 = \sigma_2^2$.

P4.14 Equation (4.113) suggests that the performance surface of an N-tap FIR Wiener filter with complex-valued input is equivalent to the performance surface of a $2N$-tap filter with real-valued input. Furthermore, the eigenvalues corresponding to the latter surface appear with multiplicity of at least two. This problem suggests an alternative procedure which also leads to the same results.

(i) Show that the Hermitian form $\mathbf{w}^{\mathrm{H}}\mathbf{R}\mathbf{w}$ may be expanded as

$$\mathbf{w}^{\mathrm{H}}\mathbf{R}\mathbf{w} = [\mathbf{w}_{\mathrm{R}}^{\mathrm{T}} \quad \mathbf{w}_{\mathrm{I}}^{\mathrm{T}}] \begin{bmatrix} \mathbf{R}_{\mathrm{R}} & -\mathbf{R}_{\mathrm{I}} \\ \mathbf{R}_{\mathrm{I}} & \mathbf{R}_{\mathrm{R}} \end{bmatrix} \begin{bmatrix} \mathbf{w}_{\mathrm{R}} \\ \mathbf{w}_{\mathrm{I}} \end{bmatrix}$$

where the subscripts R and I refer to real and imaginary parts.

Hint: Note that $\mathbf{R}_{\mathrm{I}}^{\mathrm{T}} = -\mathbf{R}_{\mathrm{I}}$ and this implies that for any arbitrary vector $\mathbf{v}$, $\mathbf{v}^{\mathrm{T}}\mathbf{R}_{\mathrm{I}}\mathbf{v} = 0$.

(ii) Show that the equation

$$\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i \tag{P4.14-1}$$

implies

$$\begin{bmatrix} \mathbf{R}_{\mathrm{R}} & -\mathbf{R}_{\mathrm{I}} \\ \mathbf{R}_{\mathrm{I}} & \mathbf{R}_{\mathrm{R}} \end{bmatrix} \begin{bmatrix} \mathbf{q}_{i,\mathrm{R}} \\ \mathbf{q}_{i,\mathrm{I}} \end{bmatrix} = \lambda_i \begin{bmatrix} \mathbf{q}_{i,\mathrm{R}} \\ \mathbf{q}_{i,\mathrm{I}} \end{bmatrix}.$$

Also, multiplying (P4.14-1) through by $j = \sqrt{-1}$, we get $\mathbf{R}(j\mathbf{q}_i) = \lambda_i(j\mathbf{q}_i)$. Show that this implies

$$\begin{bmatrix} \mathbf{R}_{\mathrm{R}} & -\mathbf{R}_{\mathrm{I}} \\ \mathbf{R}_{\mathrm{I}} & \mathbf{R}_{\mathrm{R}} \end{bmatrix} \begin{bmatrix} -\mathbf{q}_{i,\mathrm{I}} \\ \mathbf{q}_{i,\mathrm{R}} \end{bmatrix} = \lambda_i \begin{bmatrix} -\mathbf{q}_{i,\mathrm{I}} \\ \mathbf{q}_{i,\mathrm{R}} \end{bmatrix}.$$

Relate these with (4.113).

5
Search Methods

In the previous two chapters we established that the optimum tap weights of a transversal Wiener filter can be obtained by solving the Wiener–Hopf equation, provided the required statistics of the underlying signals are available. We arrived at this solution by minimizing a cost function that is a quadratic function of the filter tap-weight vector. An alternative way of finding the optimum tap weights of a transversal filter is to use an iterative search algorithm that starts at some arbitrary initial point in the tap-weight vector space and progressively moves towards the optimum tap-weight vector in steps. Each step is chosen so that the underlying cost function is reduced. If the cost function is convex (which is so for the transversal filter problem), then such an iterative search procedure is guaranteed to converge to the optimum solution. The principle of finding the optimum tap-weight vector by progressive minimization of the underlying cost function by means of an iterative algorithm is central to the development of adaptive algorithms, which will be discussed extensively in the forthcoming chapters of this book. Using a highly simplified language, we might state at this point that adaptive algorithms are nothing but iterative search algorithms derived for minimizing the underlying cost function with the true statistics replaced by their estimates obtained in some manner. Hence, a very thorough understanding of the iterative algorithms from the point of view of their development and convergence property is an essential prerequisite for the study of adaptive algorithms. This is the subject of this chapter.

In this chapter we discuss two *gradient-based* iterative methods for searching the performance surface of a transversal Wiener filter to find the tap weights that correspond to its minimum point. Th se methods are idealized versions of the class of practical algorithms which will be presented in the next few chapters. We assume that the correlation matrix of the input samples to the filter and the cross-correlation vector between the desired output and filter input are known a priori.

The first method that we discuss is known as the *method of steepest descent*. The basic concept behind this method is simple. Assuming that the cost function to be minimized is convex, we may start with an arbitrary point on the performance surface and take a small step in the direction in which the cost function decreases fastest. This corresponds to a step along the steepest-descent slope of the performance surface at that point. Repeating this successively, convergence towards the bottom of the performance surface, at which point the set of parameters that minimize the cost function assume

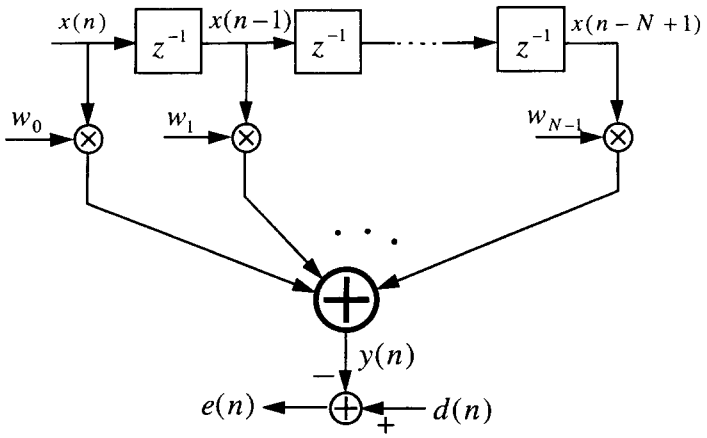


Figure 5.1 A transversal filter

their optimum values, is guaranteed. For the transversal Wiener filters we find that this method may suffer from slow convergence. The second method that we introduce can overcome this problem at the cost of additional complexity. This, which is known as *Newton's method,* takes steps that are in the direction pointing towards the bottom of the performance surface.

Our discussion in this chapter is limited to the case where the filter tap weights, input and desired output are real-valued. The extension of the results to the case of complex-valued signals is straightforward and deferred to the problems at the end of the chapter.

5.1 Method of Steepest Descent

Consider a transversal Wiener filter, as in Figure 5.1. The filter input, $x(n)$, and its desired output, $d(n)$, are assumed to be real-valued sequences. The filter tap weights, $w_0, w_1, \ldots, w_{N-1}$, are also assumed to be real-valued. The filter input and tap-weight vectors are defined, respectively, by the column vectors

$$\mathbf{w} = [w_0 \;\; w_1 \;\; \ldots \;\; w_{N-1}]^{\mathrm{T}} \tag{5.1}$$

and

$$\mathbf{x}(n) = [x(n) \;\; x(n-1) \;\; \ldots \;\; x(n-N+1)]^{\mathrm{T}}, \tag{5.2}$$

where the superscript T stands for transpose. The filter output is

$$y(n) = \mathbf{w}^{\mathrm{T}}\mathbf{x}(n). \tag{5.3}$$

We recall from Chapter 3 that the optimum tap-weight vector $\mathbf{w}_o$ is the one that minimizes the performance function

$$\xi = \mathrm{E}[e^2(n)] \tag{5.4}$$

where $e(n) = d(n) - y(n)$ is the *estimation error* of the Wiener filter. Also, we recall that the performance function ξ can be expanded as

$$\xi = \mathrm{E}[d^2(n)] - 2\mathbf{w}^{\mathrm{T}}\mathbf{p} + \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w}, \tag{5.5}$$

where $\mathbf{R} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]$ is the autocorrelation matrix of the filter input and $\mathbf{p} = \mathrm{E}[\mathbf{x}(n)d(n)]$ is the cross-correlation vector between the filter input and its desired output. The function ξ (whose details were given in the previous chapter) is a quadratic function of the filter tap-weight vector $\mathbf{w}$. It has a single global minimum which can be obtained by solving the Wiener–Hopf equation

$$\mathbf{R}\mathbf{w}_{\mathrm{o}} = \mathbf{p} \tag{5.6}$$

if $\mathbf{R}$ and $\mathbf{p}$ are available. Here, we assume that $\mathbf{R}$ and $\mathbf{p}$ are available, but resort to a different approach to find $\mathbf{w}_{\mathrm{o}}$. Instead of trying to solve equation (5.6) directly, we choose an *iterative search method* in which starting with an initial guess for $\mathbf{w}_{\mathrm{o}}$, say $\mathbf{w}(0)$, a recursive search method that may require many iterations (steps) to converge to $\mathbf{w}_{\mathrm{o}}$ is used. An understanding of this method is basic to the development of the *iterative algorithms* which are commonly used in the implementation of adaptive filters in practice.

The method of steepest descent is a general scheme that uses the following steps to search for the minimum point of any convex function of a set of parameters:

1. Start with an initial guess of the parameters whose optimum values are to be found for minimizing the function.
2. Find the gradient of the function with respect to these parameters at the present point.
3. Update the parameters by taking a step in the opposite direction of the gradient vector obtained in Step 2. This corresponds to a step in the direction of steepest descent in the cost function at the present point. Furthermore, the size of the step taken is chosen proportional to the size of the gradient vector.
4. Repeat Steps 2 and 3 until no further significant change is observed in the parameters.

To implement this procedure in the case of the transversal filter shown in Figure 5.1, we recall from Chapter 3 that

$$\nabla\xi = 2\mathbf{R}\mathbf{w} - 2\mathbf{p}, \tag{5.7}$$

where ∇ is the gradient operator defined as the column vector

$$\nabla = \left[\frac{\partial}{\partial w_0} \quad \frac{\partial}{\partial w_1} \quad \cdots \quad \frac{\partial}{\partial w_{N-1}}\right]^{\mathrm{T}}. \tag{5.8}$$

According to the above procedure, if $\mathbf{w}(k)$ is the tap-weight vector at the kth iteration, then the following recursive equation may be used to update $\mathbf{w}(k)$:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu\nabla_k\xi, \tag{5.9}$$

where μ is a positive scalar called the step-size, and $\nabla_k \xi$ denotes the gradient vector $\nabla \xi$ evaluated at the point $\mathbf{w} = \mathbf{w}(k)$. Substituting (5.7) in (5.9), we get

$$\mathbf{w}(k+1) = \mathbf{w}(k) - 2\mu(\mathbf{R}\mathbf{w}(k) - \mathbf{p}). \tag{5.10}$$

As we shall soon show, the convergence of $\mathbf{w}(k)$ to the optimum solution $\mathbf{w}_o$ and the speed at which this convergence takes place are dependent on the size of the step-size parameter μ. A large step-size may result in divergence of this recursive equation.

To see how the recursive update $\mathbf{w}(k)$ converges towards $\mathbf{w}_o$, we rearrange (5.10) as

$$\mathbf{w}(\mathrm{k}+1) = (\mathbf{I} - 2\mu\mathbf{R})\mathbf{w}(\mathrm{k}) + 2\mu\mathbf{p}, \tag{5.11}$$

where $\mathbf{I}$ is the $N \times N$ identity matrix. Next, we substitute for $\mathbf{p}$ from (5.6). Also, we subtract $\mathbf{w}_o$ from both sides of (5.11) and rearrange the result to obtain

$$\mathbf{w}(k+1) - \mathbf{w}_o = (\mathbf{I} - 2\mu\mathbf{R})(\mathbf{w}(k) - \mathbf{w}_o). \tag{5.12}$$

Defining the vector $\mathbf{v}(k)$ as

$$\mathbf{v}(k) = \mathbf{w}(k) - \mathbf{w}_o, \tag{5.13}$$

and substituting this in (5.12), we obtain

$$\mathbf{v}(k+1) = (\mathbf{I} - 2\mu\mathbf{R})\mathbf{v}(k). \tag{5.14}$$

This is the tap-weight update equation in terms of the v-axes (see Chapter 4 for further discussion on the v-axes). This result can be simplified further if we transform these to the v'-axes (see (4.92) of Chapter 4 for the definition of v'-axes). Recall from Chapter 4 that $\mathbf{R}$ has the following unitary similarity decomposition:

$$\mathbf{R} = \mathbf{Q}\Lambda\mathbf{Q}^{\mathrm{T}}, \tag{5.15}$$

where Λ is a diagonal matrix consisting of the eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$ of $\mathbf{R}$ and the columns of $\mathbf{Q}$ contain the corresponding orthonormal eigenvectors. Substituting (5.15) in (5.14) and replacing $\mathbf{I}$ with $\mathbf{Q}\mathbf{Q}^{\mathrm{T}}$, we get

$$\begin{aligned}\mathbf{v}(k+1) &= (\mathbf{Q}\mathbf{Q}^{\mathrm{T}} - 2\mu\mathbf{Q}\Lambda\mathbf{Q}^{\mathrm{T}})\mathbf{v}(k) \\ &= \mathbf{Q}(\mathbf{I} - 2\mu\Lambda)\mathbf{Q}^{\mathrm{T}}\mathbf{v}(k).\end{aligned} \tag{5.16}$$

Premultiplying (5.16) by $\mathbf{Q}^{\mathrm{T}}$ and recalling the transformation

$$\mathbf{v}'(k) = \mathbf{Q}^{\mathrm{T}}\mathbf{v}(k), \tag{5.17}$$

we obtain the recursive equation in terms of v'-axes as

$$\mathbf{v}'(k+1) = (\mathbf{I} - 2\mu\Lambda)\mathbf{v}'(k). \tag{5.18}$$

The vector recursive equation (5.18) may be separated into the scalar recursive equations

$$v_i'(k+1) = (1 - 2\mu\lambda_i)v_i'(k), \quad \text{for } i = 0, 1, \ldots, N-1, \tag{5.19}$$

where $v_i'(k)$ is the ith element of the vector $\mathbf{v}'(k)$.

Starting with a set of initial values $v_0'(0), v_1'(0), \ldots, v_{N-1}'(0)$ and iterating (5.19) k times, we get

$$v_i'(k) = (1 - 2\mu\lambda_i)^k v_i'(0), \quad \text{for } i = 0, 1, \ldots, N-1. \tag{5.20}$$

From (5.13) and (5.17) we see that $\mathbf{w}(k)$ converges to $\mathbf{w}_o$ if and only if $\mathbf{v}'(k)$ converges to the zero vector. But, (5.20) implies that $\mathbf{v}'(k)$ can converge to zero if and only if the step-size parameter μ is selected so that

$$|1 - 2\mu\lambda_i| < 1, \quad \text{for } i = 0, 1, \ldots, N-1. \tag{5.21}$$

When (5.21) is satisfied, the scalars $v_i'(k)$, for $i = 0, 1, \ldots, N-1$, exponentially decay towards zero as the number of iterations, k, increases. Furthermore, (5.21) provides the condition for the recursive equations (5.20) and, hence, the steepest-descent algorithm to be stable. The inequalities (5.21) may be expanded as

$$-1 < 1 - 2\mu\lambda_i < 1$$

or

$$0 < \mu < \frac{1}{\lambda_i}, \quad \text{for } i = 0, 1, \ldots, N-1. \tag{5.22}$$

Noting that the step-size parameter μ is common for all values of i, convergence (stability) of the steepest-descent algorithm is guaranteed only when

$$0 < \mu < \frac{1}{\lambda_{\max}}, \tag{5.23}$$

where $\lambda_{\max}$ is the maximum of the eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$. The left limit in (5.23) refers to the fact that the tap-weight correction must be in the opposite direction of the gradient vector. The right limit is to ensure that all the scalar tap-weight parameters in the recursive equations (5.19) decay exponentially as k increases.

Figure 5.2 depicts a set of plots that shows how a particular tap-weight parameter $v_i'(k)$ varies as a function of the iteration index k and for different values of the step-size parameter μ. The cases considered here correspond to the typical distinct ranges of μ, referred to as *overdamped* $(0 < \mu < 1/2\lambda_i)$, *underdamped* $(1/2\lambda_i < \mu < 1/\lambda_i)$, and *unstable* $(\mu < 0$ or $\mu > 1/\lambda_i)$.

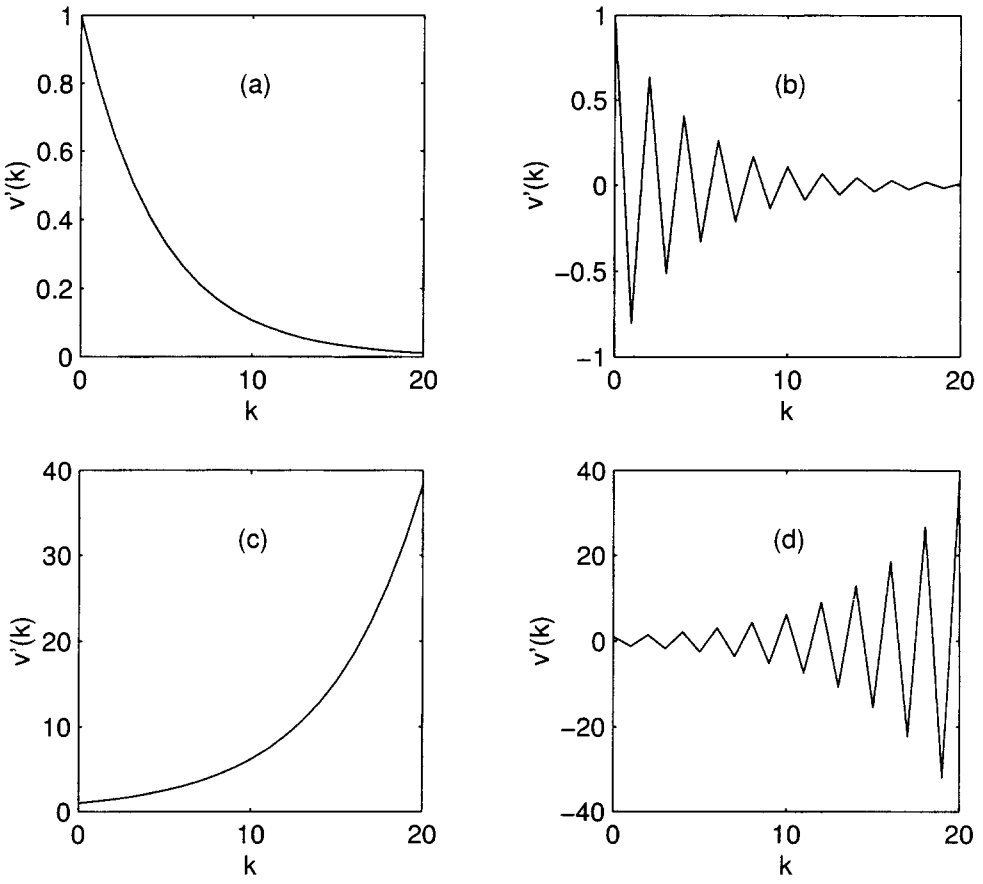


Figure 5.2 Convergence of $v_i'(k)$ as a function of iteration index k, for different values of the step-size parameter μ. (a) Overdamped case: $0 < \mu < 1/2\lambda_i$, (b) underdamped case: $1/2\lambda_i < \mu < 1/\lambda_i$, (c) unstable: $\mu < 0$, (d) unstable: $\mu > 1/\lambda_i$

We may now derive a more explicit formulation for the transient behaviour of the steepest-descent algorithm in terms of the original tap-weight vector $\mathbf{w}(k)$. We note that

$$\begin{aligned}
\mathbf{w}(k) &= \mathbf{w}_o + \mathbf{v}(k) \\
&= \mathbf{w}_o + \mathbf{Q}\mathbf{v}'(k) \\
&= \mathbf{w}_o + [\mathbf{q}_0 \ \mathbf{q}_1 \ \cdots \ \mathbf{q}_{N-1}] \begin{bmatrix} v_0'(k) \\ v_1'(k) \\ \vdots \\ v_{N-1}'(k) \end{bmatrix} \\
&= \mathbf{w}_o + \sum_{i=0}^{N-1} \mathbf{q}_i v_i'(k) \qquad (5.24)
\end{aligned}$$

where $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{N-1}$ are the eigenvectors associated with the eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$ of the correlation matrix $\mathbf{R}$. Substituting (5.20) in (5.24) we obtain

$$\mathbf{w}(k) = \mathbf{w}_o + \sum_{i=0}^{N-1} v_i'(0)(1 - 2\mu\lambda_i)^k \mathbf{q}_i. \tag{5.25}$$

This result shows that the transient behaviour of the steepest-descent algorithm for an N-tap transversal filter is determined by a sum of N exponential terms each of which is controlled by one of the eigenvalues of the correlation matrix $\mathbf{R}$. Each eigenvalue λ_i determines a particular mode of convergence in the direction defined by its associated eigenvector $\mathbf{q}_i$. The various modes work independently of one another. For a selected value of the step-size parameter μ, the geometrical ratio factor $1 - 2\mu\lambda_i$, which determines how fast the ith mode converges, is determined by the value of λ_i.

Example 5.1

Consider the modelling problem depicted in Figure 5.3. The input signal, $x(n)$, is generated by passing a white noise signal, $v(n)$, through a colouring filter with the system function

$$H(z) = \frac{\sqrt{1-\alpha^2}}{1 - \alpha z^{-1}}, \tag{5.26}$$

where α is a real-valued constant in the range -1 to $+1$. The plant is a two-tap FIR system with the system function

$$P(z) = 1 - 4z^{-1}.$$

An adaptive filter with the system function

$$W(z) = w_0 + w_1 z^{-1}$$

is used to identify the plant system function. The steepest-descent algorithm is used to find the optimum values of the tap weights w_0 and w_1. We want to see, as the iteration number increases, how the tap weights w_0 and w_1 converge towards the plant coefficients 1 and -4, respectively. We examine this for different values of the parameter α.

From the results derived in Example 4.1 of Chapter 4, we note that

$$\mathrm{E}[x^2(n)] = 1 \qquad \text{and} \qquad \mathrm{E}[x(n)x(n-1)] = \alpha.$$

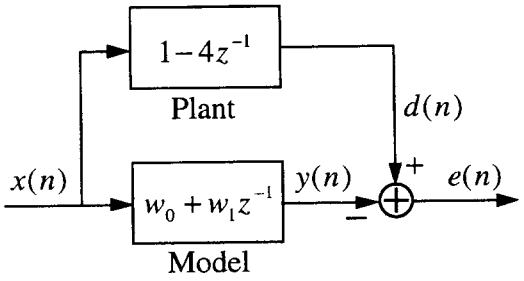


Figure 5.3 A modelling problem

These give

$$\mathbf{R} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)] = \begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix}, \tag{5.27}$$

where $\mathbf{x}(n) = [x(n) \;\; x(n-1)]^{\mathrm{T}}$. Furthermore, the elements of the cross-correlation vector $\mathbf{p} = \mathrm{E}[\mathbf{x}(n)d(n)]$ are obtained as follows:

$$\begin{aligned} p_0 &= \mathrm{E}[x(n)d(n)] = \mathrm{E}[x(n)(x(n) - 4x(n-1))] \\ &= \mathrm{E}[x^2(n)] - 4\mathrm{E}[x(n)x(n-1)] = 1 - 4\alpha \end{aligned}$$

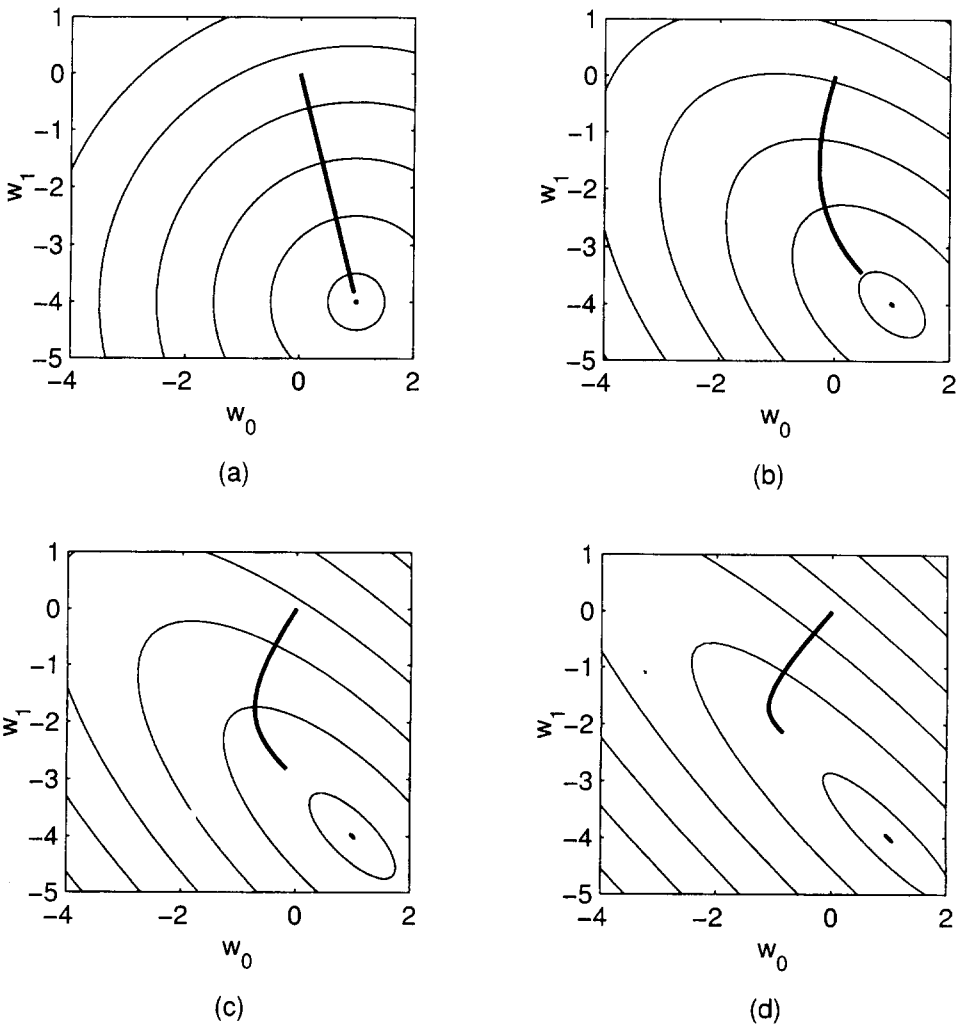


Figure 5.4 Trajectories showing how the filter tap weights vary when the steepest-descent algorithm is used: (a) $\alpha = 0$, (b) $\alpha = 0.5$, (c) $\alpha = 0.75$, (d) $\alpha = 0.9$. Each plot is based on 30 iterations and $\mu = 0.05$

and

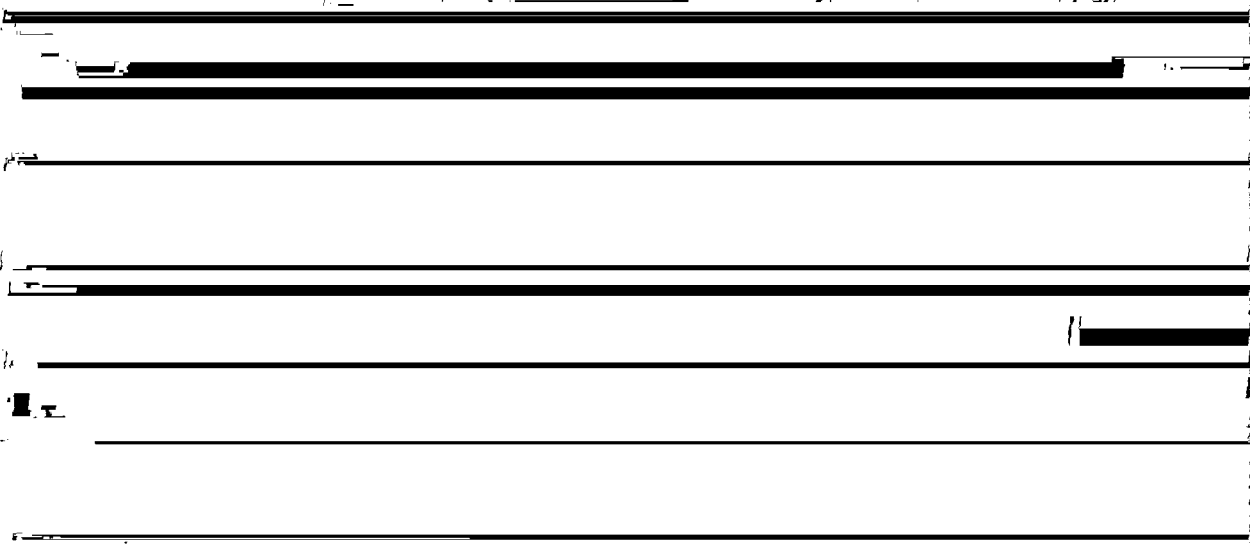
$$p_1 = \mathrm{E}[x(n-1)d(n)] = \mathrm{E}[x(n-1)(x(n) - 4x(n-1))]$$
$$= \mathrm{E}[x(n-1)x(n)] - 4\mathrm{E}[x^2(n-1)] = \alpha - 4.$$

These give

$$\mathbf{p} = \begin{bmatrix} 1 - 4\alpha \\ \alpha - 4 \end{bmatrix}. \tag{5.28}$$

Substituting (5.27) and (5.28) in (5.11), we get

$$\begin{bmatrix} w_0(k+1) \end{bmatrix} \quad \begin{bmatrix} 1 - 2\mu & -2\mu\alpha \end{bmatrix} \begin{bmatrix} w_0(k) \end{bmatrix} + 2\mu \begin{bmatrix} 1 - 4\alpha \end{bmatrix} \tag{5.29}$$



Starting with an initial value $\mathbf{w}(0) = [w_0(0)\ w_1(0)]^{\mathrm{T}}$ and letting the recursive equation (5.29) run, we get two sequences of the tap-weight variables $w_0(k)$ and $w_1(k)$. We may then plot $w_1(k)$ versus $w_0(k)$ to get the *trajectory* (path) that the steepest-descent algorithm follows. Figures 5.4(a), (b), (c) and (d) show four such trajectories that we have obtained for values of $\alpha = 0, 0.5, 0.75$ and 0.9, respectively. Also shown in the figures are the contour plots that highlight the performance surface of the filter. The convergence of the algorithm along the steepest-descent slope of the performance surface can be clearly seen. The results presented are for $\mu = 0.05$ and 30 iterations, for all cases. It is interesting to note that in the case $\alpha = 0$, which corresponds to a white input sequence, $x(n)$, the convergence is almost complete within 30 iterations. However, the other three cases require more iterations before they converge to the minimum point of the performance surface. This can be understood if we note that the eigenvalues of $\mathbf{R}$ are $\lambda_0 = 1 + \alpha$ and $\lambda_1 = 1 - \alpha$, and for α close to one, the geometrical ratio factor $1 - 2\mu\lambda_1$ may be very close to one. This introduces a slow mode of convergence along the v_1'-axis (i.e. in the direction defined by the eigenvector $\mathbf{q}_1$).

5.2 Learning Curve

Although the recursive equations (5.19) and (5.24) provide detailed information about the transient behaviour of the steepest-descent algorithm, the multi-parameter nature of the equations makes it difficult to visualize such behaviour graphically. Instead, it is more convenient to consider the variation of the mean-square error (MSE), i.e. the performance function ξ, versus the number of iterations.

When μ is selected within the bounds defined by (5.23), the terms under the summation in (5.31) converge to zero as k increases. As a result, the minimum MSE is achieved after a sufficient number of iterations.

The curve obtained by plotting $\xi(k)$ as a function of the iteration index, k, is called the *learning curve*. A learning curve of the steepest-descent algorithm, as can be seen from (5.31), consists of the sum of N exponentially decaying terms each of which corresponds to one of the modes of convergence of the algorithm. Each exponential term may be characterized by a time constant which is obtained as follows.

Let

$$(1 - 2\mu\lambda_i)^{2k} = e^{-k/\tau_i} \tag{5.32}$$

and define τ_i as the time constant associated with the exponential term $(1 - 2\mu\lambda_i)^{2k}$. Solving (5.32) for τ_i, we get

$$\tau_i = \frac{-1}{2\ln(1 - 2\mu\lambda_i)}. \tag{5.33}$$

For small values of the step-size parameter μ, when $2\mu\lambda_i \ll 1$, we note that

$$\ln(1 - 2\mu\lambda_i) \approx -2\mu\lambda_i. \tag{5.34}$$

Substituting this in (5.33) we obtain

$$\tau_i \approx \frac{1}{4\mu\lambda_i}. \tag{5.35}$$

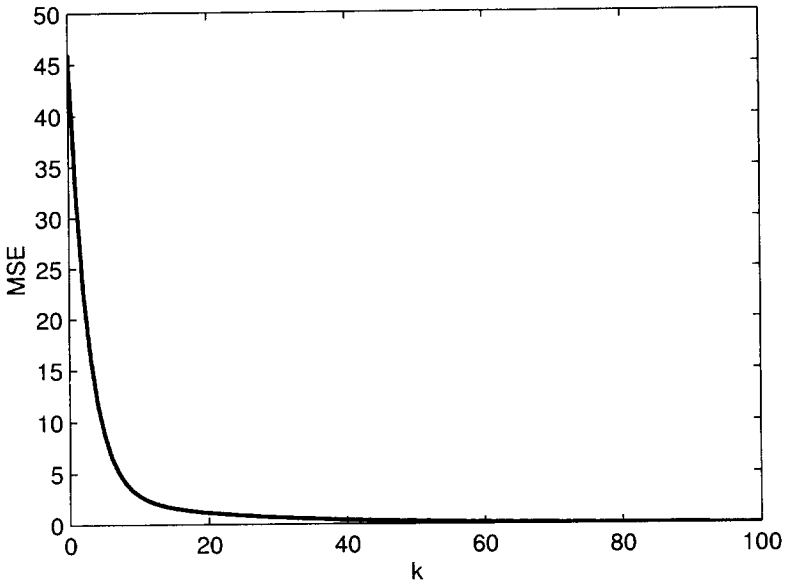


Figure 5.5 A learning curve of the modelling problem. The ξ (MSE) axis is scaled linearly

These are the time constants that characterize the learning curve of the modelling problem. Figure 5.5 shows a learning curve of the modelling problem when $\mathbf{w}(0) = [2\ 2]^{\mathrm{T}}, \alpha = 0.75$ and $\mu = 0.05$. For these values, we obtain

$$\tau_0 \approx 2.85 \qquad \text{and} \qquad \tau_1 \approx 20. \tag{5.38}$$

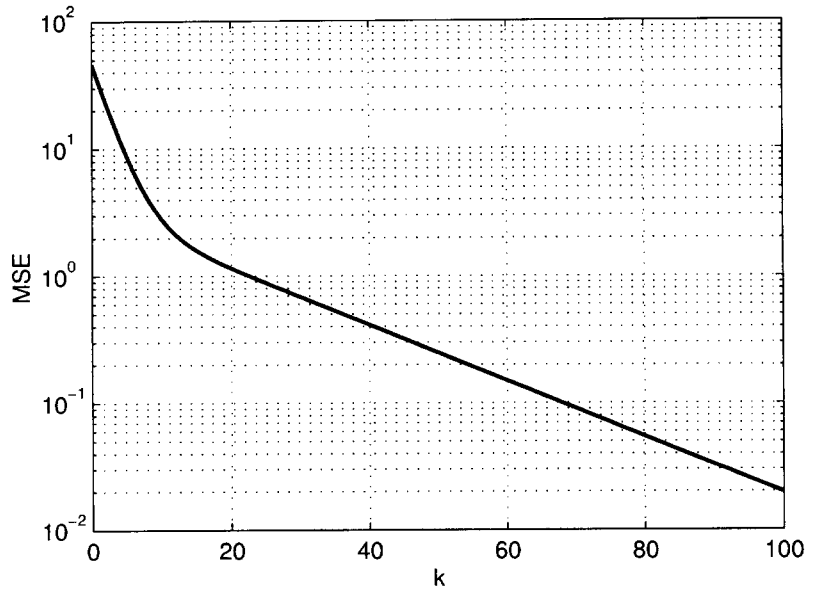


Figure 5.6 A learning curve of the modelling problem. The ξ (MSE) axis is scaled logarithmically

The existence of two distinct time constants on the learning curve in Figure 5.5 is clearly observed.

The two time constants could be observed more clearly if the ξ-axis were scaled logarithmically. To see this, the learning curve of the modelling problem is plotted in Figure 5.6 with the ξ-axis scaled logarithmically. The two exponentials appear as two straight lines on this plot. The first part of the plot, with a steep slope, is dominantly controlled by τ_0. The remaining part of the learning curve shows the contribution of the second exponential which is characterized by τ_1. Estimates of the time constants may be obtained by finding the number of iterations required for ξ to drop 2.73 (i.e. the napier number) times along each of the slopes. This gives

$$\tau_0 \approx 3 \qquad \text{and} \qquad \tau_1 \approx 20.$$

which match well with those in (5.38).

5.3 The Effect of Eigenvalue Spread

Our study in the previous two sections shows that the performance of the steepest-descent algorithm is highly dependent on the eigenvalues of the correlation matrix $\mathbf{R}$. In general, a wider spread of the eigenvalues results in a poorer performance of the steepest-descent algorithm. To gain further insight into this property of the steepest-descent algorithm, we find the optimum value of the step-size parameter μ which results in the fastest possible convergence of the steepest-descent algorithm.

We note that the speeds at which various modes of the steepest-descent algorithm converge are determined by the size (absolute value) of the geometrical ratio factors $1 - 2\mu\lambda_i$, for $i = 0, 1, \ldots, N-1$. For a given value of μ, the transient time of the steepest-descent algorithm is determined by the largest element in the set $\{|1 - 2\mu\lambda_i|, i = 0, 1, \ldots, N-1\}$. The optimum value of μ that minimizes the largest element in the latter set is obtained by looking at the two extreme cases which correspond to $\lambda_{\max}$ and $\lambda_{\min}$, i.e. the maximum and minimum eigenvalues of $\mathbf{R}$. Figure 5.7 shows the plots of $|1 - 2\mu\lambda_{\min}|$ and $|1 - 2\mu\lambda_{\max}|$ as functions of μ. The plots of the other eigenvalues lie in between these two plots. From these plots we can clearly see that the optimum value of the step-size parameter μ corresponds to the point where the two plots meet. This is the point highlighted as μ_{opt} in Figure 5.7. It corresponds to the case where

$$1 - 2\mu_{\text{opt}}\lambda_{\min} = -(1 - 2\mu_{\text{opt}}\lambda_{\max}). \tag{5.39}$$

Solving this for μ_{opt}, we obtain

$$\mu_{\text{opt}} = \frac{1}{\lambda_{\min} + \lambda_{\max}}. \tag{5.40}$$

For this choice of the step-size parameter, $1 - 2\mu_{\text{opt}}\lambda_{\min}$ is positive and $1 - 2\mu_{\text{opt}}\lambda_{\max}$ is negative. These correspond to the overdamped and underdamped cases presented in Figure 5.2, respectively. However, the two modes converge at the same speed. For $\mu = \mu_{\text{opt}}$, the speed of convergence of the steepest-descent algorithm is determined by the geometrical ratio factor

$$\beta = 1 - 2\mu_{\text{opt}}\lambda_{\min}. \tag{5.41}$$

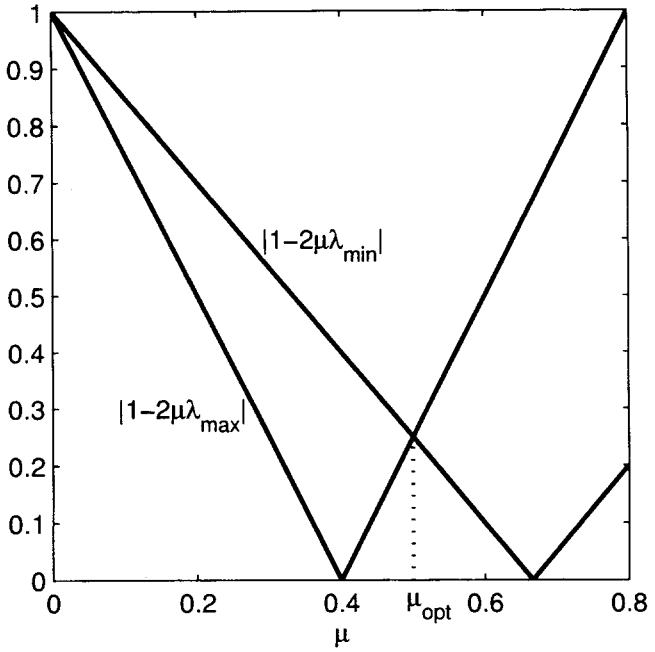


Figure 5.7 The extreme cases showing how $1 - 2\mu\lambda_i$ varies as a function of the step-size parameter μ

Substituting (5.40) in (5.41) we obtain

$$\beta = \frac{\lambda_{\max}/\lambda_{\min} - 1}{\lambda_{\max}/\lambda_{\min} + 1}. \tag{5.42}$$

This has a value that remains between 0 and 1. When $\lambda_{\max} = \lambda_{\min}$, $\beta = 0$ and the steepest-descent algorithm can converge in one step. As the ratio $\lambda_{\max}/\lambda_{\min}$ increases, β also increases and becomes close to one when $\lambda_{\max}/\lambda_{\min}$ is large. Clearly, a value of β close to one corresponds to a slow mode of convergence. Thus, we note that the ratio $\lambda_{\max}/\lambda_{\min}$ plays a fundamental role in limiting the convergence performance of the steepest-descent algorithm. This ratio is called the *eigenvalue spread.*

We may also recall from the previous chapter that the values of $\lambda_{\max}$ and $\lambda_{\min}$ are closely related to the maximum and minimum values of the power spectral density of the underlying process. Noting this, we may say that *the performance of the steepest-descent algorithm is closely related to the shape of the power spectral density of the underlying input process.* A wide distribution of the energy of the underlying process within different frequency bands introduces slow modes of convergence which result in a poor performance of the steepest-descent algorithm. When the underlying process contains very little energy in a band of frequencies, we say the filter is *weakly excited* in that band. *Weak excitation,* as we see, degrades the performance of the steepest-descent algorithm.

5.4 Newton's Method

Our discussions in the previous sections show that the steepest-descent algorithm may suffer from slow modes of convergence which arise as a result of the spread in the eigenvalues of the correlation matrix $\mathbf{R}$. This means that if we can somehow get rid of the eigenvalue spread, we can get much better convergence performance. This is exactly what Newton's method does. To derive Newton's method for the quadratic case, we start from the steepest-descent algorithm given in (5.10). Using $\mathbf{p} = \mathbf{R}\mathbf{w}_o$, (5.10) becomes

$$\mathbf{w}(k+1) = \mathbf{w}(k) - 2\mu\mathbf{R}(\mathbf{w}(k) - \mathbf{w}_o). \tag{5.43}$$

We may note that it is the presence of $\mathbf{R}$ in (5.43) that causes the eigenvalue-spread problem in the steepest-descent algorithm. Newton's method overcomes this problem by replacing the scalar step-size parameter μ with a matrix step-size given by $\mu\mathbf{R}^{-1}$. The resulting algorithm is

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu\mathbf{R}^{-1}\nabla_k\xi. \tag{5.44}$$

Figure 5.8 demonstrates the effect of the addition of $\mathbf{R}^{-1}$ in front of the gradient vector in Newton's update equation (5.44). This has the effect of rotating the gradient vector to the direction pointing towards the minimum point of the performance surface.

Substituting (5.7) in (5.44) we obtain

$$\begin{aligned}\mathbf{w}(k+1) &= \mathbf{w}(k) - 2\mu\mathbf{R}^{-1}(\mathbf{R}w(k) - \mathbf{p}) \\ &= (1 - 2\mu)\mathbf{w}(k) + 2\mu\mathbf{R}^{-1}\mathbf{p}.\end{aligned} \tag{5.45}$$

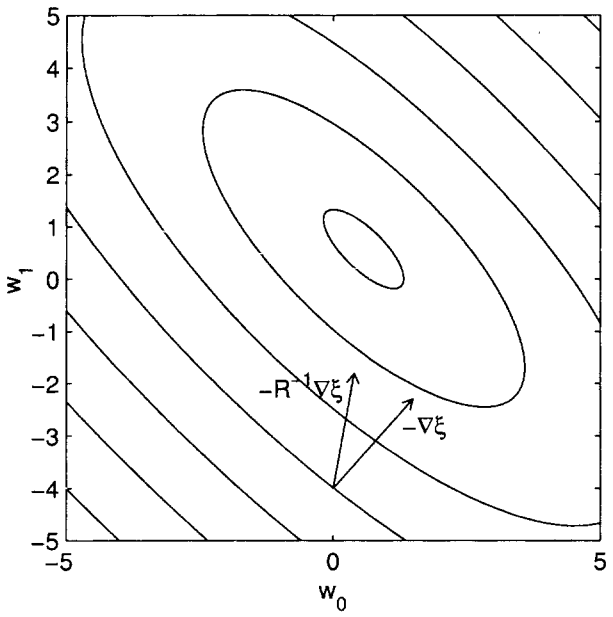


Figure 5.8 The negative gradient vector and its correction by Newton's method

We also note that $\mathbf{R}^{-1}\mathbf{p}$ is equal to the optimum tap-weight vector $\mathbf{w}_o$. Using this in (5.45) we obtain

$$\mathbf{w}(k+1) = (1-2\mu)\mathbf{w}(k) + 2\mu\mathbf{w}_o. \tag{5.46}$$

Subtracting $\mathbf{w}_o$ from both sides of (5.46), we get

$$\mathbf{w}(k+1) - \mathbf{w}_o = (1-2\mu)(\mathbf{w}(k) - \mathbf{w}_o). \tag{5.47}$$

Starting with an initial value $\mathbf{w}(0)$ and iterating (5.47), we obtain

$$\mathbf{w}(k) - \mathbf{w}_o = (1-2\mu)^k(\mathbf{w}(0) - \mathbf{w}_o). \tag{5.48}$$

The original Newton's method selects the step-size parameter μ equal to 0.5. This leads to convergence of $\mathbf{w}(k)$ to its optimum value, $\mathbf{w}_o$, in one iteration. In particular, we note that setting $\mu = 0.5$ and $k = 1$ in (5.48), we obtain $\mathbf{w}(1) = \mathbf{w}_o$. However, in the actual implementation of adaptive filters, where the exact values of $\nabla_k\xi$ and $\mathbf{R}^{-1}$ are not available and have to be estimated, we need to use a step-size parameter much smaller than 0.5. Thus, an evaluation of Newton's recursion (5.44) for values of $\mu \neq 0.5$ is instructive for our further study in later chapters.

Using (5.48) and following the same line of derivations as in the case of the steepest-descent method, it is straightforward to show that (see Problem P5.4)

$$\xi(k) = \xi_{\min} + (1-2\mu)^{2k}(\xi(0) - \xi_{\min}), \tag{5.49}$$

where $\xi(k)$ is the value of the performance function, ξ, when $\mathbf{w} = \mathbf{w}(k)$.

From (5.49) we note that the stability of Newton's algorithm is guaranteed when $|1-2\mu| < 1$ or, equivalently,

$$0 \leq \mu \leq 1. \tag{5.50}$$

With reference to (5.49), we make the following observations. *The transient behaviour of Newton's algorithm is characterized by a single exponential whose corresponding time constant is obtained by solving the equation*

$$(1-2\mu)^{2k} = e^{-k/\tau}. \tag{5.51}$$

When $2\mu \ll 1$, this gives

$$\tau \approx \frac{1}{4\mu}. \tag{5.52}$$

This result shows that Newton's method has only one mode of convergence and that is solely determined by its step-size parameter μ.

5.5 An Alternative Interpretation of Newton's Algorithm

Further insight into the operation of Newton's algorithm is developed by giving an alternative derivation of it. This derivation uses the Karhunen–Loéve transform (KLT) which was introduced in the previous chapter.

For an observation vector $\mathbf{x}(n)$ with real-valued elements, the KLT is defined by the equation

$$\mathbf{x}'(n) = \mathbf{Q}^{\mathrm{T}}\mathbf{x}(n) \tag{5.53}$$

where $\mathbf{Q}$ is the $N \times N$ matrix whose columns are the eigenvectors $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{N-1}$ of the correlation matrix $\mathbf{R} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]$. We recall from Chapter 4 that the elements of the transformed vector $\mathbf{x}'(n)$, denoted by $x'_0(n), x'_1(n), \ldots, x'_{N-1}(n)$, constitute a set of mutually uncorrelated random variables. Furthermore, (4.68) implies that

$$\mathrm{E}[x'^2_i(n)] = \lambda_i, \quad \text{for } i = 0, 1, \ldots, N-1, \tag{5.54}$$

where λ_is are the eigenvalues of the correlation matrix $\mathbf{R}$.

We define the vector $\mathbf{x}'^{\mathcal{N}}(n)$ whose elements are

$$x'^{\mathcal{N}}_i(n) = \lambda_i^{-1/2} x'_i(n), \quad \text{for } i = 0, 1, \ldots, N-1, \tag{5.55}$$

where the superscript $\mathcal{N}$ signifies the fact that $x'^{\mathcal{N}}_i(n)$ is normalized to the power of unity (see (5.57) below). These equations may collectively be written as

$$\mathbf{x}'^{\mathcal{N}}(n) = \mathbf{\Lambda}^{-1/2}\mathbf{x}'(n) \tag{5.56}$$

where $\mathbf{\Lambda}$ is a diagonal matrix consisting of the eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$. It is straightforward to show that

$$\mathbf{R}'^{\mathcal{N}} = \mathrm{E}[\mathbf{x}'^{\mathcal{N}}(n)\mathbf{x}'^{\mathcal{N}T}(n)] = \mathbf{I}, \tag{5.57}$$

where $\mathbf{I}$ is the $N \times N$ identity matrix.

We also define

$$\mathbf{w}'^{\mathcal{N}} = \mathbf{\Lambda}^{1/2}\mathbf{Q}^{\mathrm{T}}\mathbf{w} \tag{5.58}$$

and note that

$$\mathbf{w}'^{\mathcal{N}T}\mathbf{x}'^{\mathcal{N}}(n) = \mathbf{w}^{\mathrm{T}}\mathbf{Q}\mathbf{\Lambda}^{1/2}\mathbf{\Lambda}^{-1/2}\mathbf{Q}^{\mathrm{T}}\mathbf{x}(n) = \mathbf{w}^{\mathrm{T}}\mathbf{x}(n). \tag{5.59}$$

This result shows that a filter with an input vector $\mathbf{x}(n)$ and output $y(n) = \mathbf{w}^{\mathrm{T}}\mathbf{x}(n)$ may alternatively be realized by using $\mathbf{x}'^{\mathcal{N}}(n)$ and $\mathbf{w}'^{\mathcal{N}}$ as the filter input and tap-weight vector, respectively. The steepest-descent algorithm for this realization may be written as (see (5.11))

$$\mathbf{w}'^{\mathcal{N}}(k+1) = (\mathbf{I} - 2\mu\mathbf{R}'^{\mathcal{N}})\mathbf{w}'^{\mathcal{N}}(k) + 2\mu\mathbf{p}'^{\mathcal{N}}, \tag{5.60}$$

where

$$\mathbf{p}'^{\mathcal{N}} = \mathrm{E}[\mathbf{x}'^{\mathcal{N}}(n)d(n)]. \tag{5.61}$$

Since $\mathbf{R}'^{\mathcal{N}} = \mathbf{I}$, (5.60) simplifies to

$$\mathbf{w}'^{\mathcal{N}}(k+1) = (1-2\mu)\mathbf{w}'^{\mathcal{N}}(k) + 2\mu\mathbf{w}_{\mathrm{o}}'^{\mathcal{N}}, \tag{5.62}$$

where $\mathbf{w}_{\mathrm{o}}'^{\mathcal{N}} = (\mathbf{R}'^{\mathcal{N}})^{-1}\mathbf{p}'^{\mathcal{N}} = \mathbf{p}'^{\mathcal{N}}$ is the optimum value of the tap-weight vector $\mathbf{w}'^{\mathcal{N}}$. Comparing this with Newton's algorithm (5.46), we find that the steepest-descent algorithm in this case works just like Newton's algorithm.

Next, we show that the recursive equation (5.60) is nothing but Newton's recursive equation (5.44) written in a slightly different form. For this, we use (5.58) in (5.62) to obtain

$$\mathbf{\Lambda}^{1/2}\mathbf{Q}^{\mathrm{T}}\mathbf{w}(k+1) = (1-2\mu)\mathbf{\Lambda}^{1/2}\mathbf{Q}^{\mathrm{T}}\mathbf{w}(k) + 2\mu\mathbf{\Lambda}^{1/2}\mathbf{Q}^{\mathrm{T}}\mathbf{w}_{\mathrm{o}}. \tag{5.63}$$

Premultiplying both sides of this equation by $(\mathbf{\Lambda}^{1/2}\mathbf{Q}^{\mathrm{T}})^{-1} = \mathbf{Q}\,\mathbf{\Lambda}^{-1/2}$ (since $(\mathbf{Q}^{\mathrm{T}})^{-1} = \mathbf{Q}$), we get (5.46), which can easily be converted to (5.44).

The above development shows that Newton's algorithm may be viewed as *a steepest-descent algorithm for the transformed input signal.* The eigenvalue-spread problem associated with the steepest-descent algorithm is resolved by decorrelating the filter input samples (through their corresponding Karhunen–Loéve transform) followed by a power normalization procedure. *This is a whitening process; namely, the input samples are decorrelated and then normalized to the unit power prior to the filtering process.*

Problems

P5.1 Starting with the canonical form of the performance function, i.e. (4.93), suggest an alternative derivation of (5.25).

P5.2 Show that when the steepest-descent algorithm (5.10) is used, the time constants that control the variation of the tap weights of a transversal filter are

$$\tau_i' = \frac{1}{2\mu\lambda_i}, \quad \text{for } i = 0, 1, \ldots, N-1.$$

P5.3 Give a detailed derivation of (5.25) in the case where the underlying signals are complex-valued.

P5.4 Give a detailed derivation of (5.49).

P5.5 Show that if in the steepest-descent algorithm the tap-weight vector is initialized to zero,

$$\mathbf{w}(k) = [\mathbf{I} - (\mathbf{I} - 2\mu\mathbf{R})^k]\mathbf{w}_{\mathrm{o}},$$

where $\mathbf{w}_{\mathrm{o}}$ is the optimum tap-weight vector.

P5.6 Consider the modelling problem depicted in Figure P5.6. Note that the input to the model is a noisy version of the plant input. The additive noise at the model input, $\nu_i(n)$, is white and its variance is σ_i^2. The sequence $\nu_o(n)$ is the plant noise. It is uncorrelated with $u(n)$ and $\nu_i(n)$. The correlation matrix of the plant input, $u(n)$, is denoted by $\mathbf{R}$. The model has to be selected so that the MSE at the model output is minimized.

(i) Find the correlation matrix of the model input and show that it shares the same set of eigenvectors with $\mathbf{R}$.

(ii) Derive the corresponding Wiener–Hopf equation.

(iii) Show that the difference between the plant tap-weight vector, $\mathbf{w}_o$, and its estimate, $\hat{\mathbf{w}}_o$, which is obtained through the Wiener–Hopf equation derived in (ii), is

$$\mathbf{w}_o - \hat{\mathbf{w}}_o = \sigma_i^2 \sum_{l=0}^{N-1} \frac{\mathbf{q}_l^T \mathbf{p}}{\lambda_l(\lambda_l + \sigma_i^2)} \mathbf{q}_l$$

where the $\mathbf{q}_l$s are the eigenvectors of $\mathbf{R}$ and $\mathbf{p}$ is the cross-correlation between the model input and the desired output.

(iv) Show that the mismatch of the plant and model, i.e. the difference $\mathbf{w}_o - \hat{\mathbf{w}}_o$, results in an excess MSE at the model output which is given by

$$\text{excess MSE} = \sigma_i^4 \sum_{l=0}^{N-1} \frac{(\mathbf{q}_l^T \mathbf{p})^2}{\lambda_l(\lambda_l + \sigma_i^2)^2}.$$

(v) If the steepest-descent algorithm is used to find $\hat{\mathbf{w}}_o$, find the time constants of the resulting learning curve. How do these time constants vary with σ_i^2? Discuss the eigenvalue-spread problem as σ_i^2 varies.

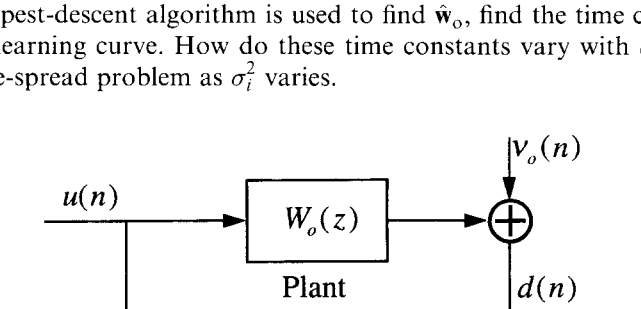


Figure P5.6

Define the vector

$$\breve{\mathbf{x}}(n) = \mathbf{R}^{-1/2}\mathbf{x}(n),$$

where $\mathbf{R} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]$. Let $\breve{\mathbf{x}}(n)$ be the input to a filter whose output is obtained through the equation

$$\breve{y}(n) = \breve{\mathbf{w}}^{\mathrm{T}}\breve{\mathbf{x}}(n),$$

where $\breve{\mathbf{w}}$ is the filter tap-weight vector.

(i) Derive an equation for $\breve{\mathbf{w}}$ so that the two outputs $y(n)$ and $\breve{y}(n)$ are the same.
(ii) Derive a steepest-descent update equation for the tap-weight vector $\breve{\mathbf{w}}$.
(iii) Derive an equation that demonstrates the variation of the tap weights of the filter as the steepest-descent algorithm derived in part (ii) is running.
(iv) Find the time constants of the learning curve of the algorithm.
(v) Show that the update equation derived in (ii) is equivalent to Newton's algorithm.

P5.8 Consider a two-tap Wiener filter which is characterized by the following parameters:

$$\mathbf{R} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{p} = \begin{bmatrix} 2 \\ 1 \end{bmatrix},$$

where $\mathbf{R}$ is the correlation matrix of the filter tap-input vector, $\mathbf{x}(n)$, and $\mathbf{p}$ is the cross-correlation between $\mathbf{x}(n)$ and the desired output, $d(n)$.

(i) Find the range of the step-size parameter μ that ensures convergence of the steepest-descent algorithm. Does this result depend on the cross-correlation vector $\mathbf{p}$?
(ii) Run the steepest-descent algorithm for $\mu = 0.05$, 0.1, 0.5 and 1 and plot the corresponding trajectories in the (w_0, w_1)-plane.
(iii) For $\mu = 0.05$, plot $w_0(k)$ and $w_1(k)$, separately, as functions of the iteration index, k.
(iv) On the plots obtained in (iii), you should find that the variation in each tap weight is signified by two distinct time constants. This implies that the variation of each tap weight may be decomposed into the summation of two distinct exponential series. Explain this observation.

P5.9 For the modelling problem discussed in Example 5.1, plot the trajectories of the steepest-descent and Newton's algorithm on the same plane for $\mu = 0.05$ and $\alpha = 0$, 0.5, 0.75 and 0.9. Comment on your observations.

P5.10 Consider the modelling problem depicted in Figure 5.3. Let $x(n) = 1$, for all values of n.

(i) Derive the steepest-descent algorithm which may be used to find the model parameters.

(ii) Derive an equation for the performance function of the present problem, and plot the contours that show its performance surface.

(iii) Run the algorithm that you have derived in (i) and find the model parameters that it converges to.

(iv) On the performance surface obtained in (ii), plot the trajectory showing the variation of the model parameters. Comment on your observation.

6
The LMS Algorithm

The celebrated least-mean-square (LMS) algorithm is introduced in this chapter. The LMS algorithm, which was first proposed by Widrow and Hoff in 1960, is the most widely used adaptive filtering algorithm, in practice. This wide spectrum of applications of the LMS algorithm can be attributed to its simplicity and robustness to signal statistics. The LMS algorithm has also been cited and worked upon by many researchers, and over the years many modifications to it have been proposed. In this and the subsequent few chapters we introduce and study several of such modifications.

6.1 Derivation of the LMS Algorithm

Figure 6.1 depicts an N-tap transversal adaptive filter. The filter input, $x(n)$, desired output, $d(n)$, and the filter output,

$$y(n) = \sum_{i=0}^{N-1} w_i(n)x(n-i), \tag{6.1}$$

are assumed to be real-valued sequences. The tap weights $w_0(n), w_1(n), \ldots, w_{N-1}(n)$ are selected so that the difference (error)

$$e(n) = d(n) - y(n), \tag{6.2}$$

is minimized in some sense. It may be noted that the filter tap weights are explicitly indicated to be functions of the time index n. This signifies that in an adaptive filter, in general, tap weights are time varying, since they are continuously being adapted so that any variations in the signal's statistics could be tracked. The LMS algorithm changes (adapts) the filter tap weights so that $e(n)$ is minimized in the mean-square sense, thus the name least mean square. When the processes $x(n)$ and $d(n)$ are jointly stationary, this algorithm converges to a set of tap weights which, on average, are equal to the Wiener–Hopf solution discussed in Chapter 3. In other words, the LMS algorithm is a practical scheme for realizing Wiener filters, without explicitly solving the Wiener–Hopf equation. It is a sequential algorithm which can be used to adapt the tap weights of a filter by continuous observation of its input, $x(n)$, and desired output, $d(n)$.

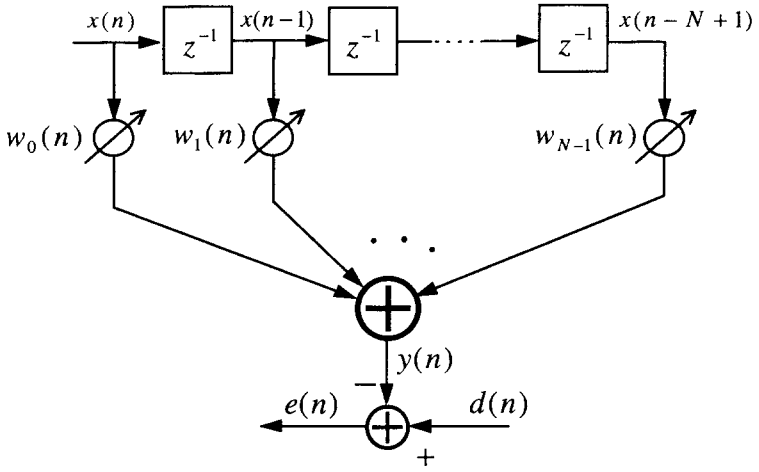


Figure 6.1 An N-tap transversal adaptive filter

The conventional LMS algorithm is a stochastic implementation of the steepest-descent algorithm. It simply replaces the cost function $\xi = \mathrm{E}[e^2(n)]$ by its instantaneous [illegible] $\hat{\xi}(n) = e^2(n)$. Substituting $\hat{\xi}(n) = e^2(n)$ for ξ in the steepest-descent [illegible]

recursion (5.9), of Chapter 5, and replacing the iteration index k by the time index n, we obtain

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu \nabla e^2(n) \tag{6.3}$$

Table 6.1 Summary of the LMS algorithm

Input:	Tap-weight vector, $\mathbf{w}(n)$, Input vector, $\mathbf{x}(n)$, and desired output, $d(n)$.
Output:	Filter output, $y(n)$, Tap-weight vector update, $\mathbf{w}(n+1)$.

1. Filtering:
   $y(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n)$
2. Error estimation:
   $e(n) = d(n) - y(n)$
3. Tap-weight vector adaptation:
   $\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n)$

Using (6.4) and (6.7) we obtain

$$\nabla e^2(n) = -2e(n)\mathbf{x}(n), \tag{6.8}$$

where $\mathbf{x}(n) = [x(n) \;\; x(n-1) \;\; \ldots \;\; x(n-N+1)]^{\mathrm{T}}$. Substituting this result in (6.3) we get

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n). \tag{6.9}$$

This is referred to as the LMS recursion. It suggests a simple procedure for recursive adaptation of the filter coefficients after the arrival of every new input sample, $x(n)$, and its corresponding desired output sample, $d(n)$. Equations (6.1), (6.2) and (6.9), in this order, specify the three steps required to complete each iteration of the LMS algorithm. Equation (6.1) is referred to as filtering. It is performed to obtain the filter output. Equation (6.2) is used to calculate the estimation error. Equation (6.9) is the tap-weight adaptation recursion. Table 6.1 summarizes of the LMS algorithm.

The eminent feature of the LMS algorithm which has made it the most popular adaptive filtering scheme is its simplicity. Its implementation requires $2N + 1$ multiplications (N multiplications for calculating the output $y(n)$, one to obtain $(2\mu) \times e(n)$ and N for the scalar-by-vector multiplication $(2\mu e(n)) \times \mathbf{x}(n)$) and $2N$ additions. Another important feature of the LMS algorithm which is equally important from an implementation point of view is its stable and robust performance against different signal conditions. This aspect of the LMS algorithm will be studied in later chapters when it is compared with other alternative adaptive filtering algorithms. The major problem of the LMS recursion (6.9) is its slow convergence when the underlying input process is highly coloured. This aspect of the LMS algorithm is discussed in the next section and solutions to it will be given in later chapters.

6.2 Average Tap-Weight Behaviour of the LMS Algorithm

Consider the case where the filter input, $x(n)$, and its desired output, $d(n)$, are stationary. In that case the optimum tap-weight vector, $\mathbf{w}_o$, of the transversal Wiener filter is fixed and can be obtained according to the Wiener–Hopf equation (3.24). Subtracting $\mathbf{w}_o$

from both sides of (6.9) we obtain

$$\mathbf{v}(n+1) = \mathbf{v}(n) + 2\mu e(n)\mathbf{x}(n), \tag{6.10}$$

where $\mathbf{v}(n) = \mathbf{w}(n) - \mathbf{w}_o$ is the weight-error vector. We also note that

$$\begin{aligned} e(n) &= d(n) - \mathbf{w}^T(n)\mathbf{x}(n) \\ &= d(n) - \mathbf{x}^T(n)\mathbf{w}(n) \\ &= d(n) - \mathbf{x}^T(n)\mathbf{w}_o - \mathbf{x}^T(n)(\mathbf{w}(n) - \mathbf{w}_o) \\ &= e_o(n) - \mathbf{x}^T(n)\mathbf{v}(n) \end{aligned} \tag{6.11}$$

where

$$e_o(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}_o \tag{6.12}$$

is the estimation error when the filter tap weights are optimum. Substituting (6.11) in (6.10) and rearranging, we obtain

$$\mathbf{v}(n+1) = \big(\mathbf{I} - 2\mu\mathbf{x}(n)\mathbf{x}^T(n)\big)\mathbf{v}(n) + 2\mu e_o(n)\mathbf{x}(n), \tag{6.13}$$

where $\mathbf{I}$ is the identity matrix. Taking expectations on both sides of (6.13), we get

$$\begin{aligned} \mathrm{E}[\mathbf{v}(n+1)] &= \mathrm{E}\big[\big(\mathbf{I} - 2\mu\mathbf{x}(n)\mathbf{x}^T(n)\big)\mathbf{v}(n)\big] + 2\mu\mathrm{E}[e_o(n)\mathbf{x}(n)] \\ &= \mathrm{E}\big[\big(\mathbf{I} - 2\mu\mathbf{x}(n)\mathbf{x}^T(n)\big)\mathbf{v}(n)\big], \end{aligned} \tag{6.14}$$

where the last equality follows from the fact that $\mathrm{E}[e_o(n)\mathbf{x}(n)] = \mathbf{0}$, according to the principle of orthogonality.

The main difficulty with any further analysis of the right-hand side of (6.14) is that it involves evaluation of the third-order moment vector $\mathrm{E}[\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{v}(n)]$, which in general is a difficult mathematical task. Different approaches have been adopted by researchers to overcome this mathematical hurdle. The most widely used analysis assumes that the present observation data samples $(\mathbf{x}(n), d(n))$ are independent of the past observations $(\mathbf{x}(n-1), d(n-1)), (\mathbf{x}(n-2), d(n-2)), \ldots$; see, for example, Widrow et al. (1976) and Feuer and Weinstein (1985). This is referred to as *the independence assumption*. Using the independence assumption, we can argue that since $\mathbf{v}(n)$ depends only on the past observations $(\mathbf{x}(n-1), d(n-1)), (\mathbf{x}(n-2), d(n-2)), \ldots$, it is independent of $\mathbf{x}(n)$, and thus

$$\mathrm{E}[\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{v}(n)] = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^T(n)]\mathrm{E}[\mathbf{v}(n)]. \tag{6.15}$$

We may note that in most practical cases the independence assumption is questionable. For example, in the case of a length N transversal filter the input vectors

$$\mathbf{x}(n) = [x(n) \;\; x(n-1) \;\; \ldots \;\; x(n-N+1)]^T$$

and

$$\mathbf{x}(n-1) = [x(n-1) \;\; x(n-2) \;\; \ldots \;\; x(n-N)]^T$$

have $(N-1)$ terms in common, out of N. Nevertheless, experience with the LMS algorithm has shown that the predictions made by the independence assumption match the computer simulations and the actual performance of the LMS algorithm, in practice. This may be explained as follows. The tap-weight vector $\mathbf{w}(n)$ at any given time has been affected by the whole past history of the observation data samples $(\mathbf{x}(n-1), d(n-1))$, $(\mathbf{x}(n-2), d(n-2)), \ldots$. When the step-size parameter μ is small the share of the last N observations in the present value of $\mathbf{w}(n)$ is small, and thus we may say $\mathbf{x}(n)$ and $\mathbf{w}(n)$ are weakly dependent. This clearly leads to (6.15), with some degree of approximation, if we can assume that the observation samples which are apart from each other at a distance of N or greater are weakly dependent. This reasoning seems to be more appealing than the independence assumption. In any case we use (6.15) and other similar equations (approximations) which will be introduced later to proceed with our analysis in this book.

Substituting (6.15) in (6.14) we obtain

$$\mathrm{E}[\mathbf{v}(n+1)] = (\mathbf{I} - 2\mu\mathbf{R})\mathrm{E}[\mathbf{v}(n)] \tag{6.16}$$

where $\mathbf{R} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]$ is the correlation matrix of the input vector $\mathbf{x}(n)$.

Comparing the recursions (6.16) nad (5.14), we find that they are of exactly the same mathematical form. The deterministic weight-error vector $\mathbf{v}(k)$ in (5.14) of the steepest-descent algorithm is replaced by the averaged weight-error vector $\mathrm{E}[\mathbf{v}(n)]$ of the LMS algorithm. This suggests that, on average, the LMS algorithm behaves just like the steepest-descent algorithm. In particular, similar to the steepest-descent algorithm, the LMS algorithm is controlled by N modes of convergence which are characterized by the eigenvalues of the correlation matrix $\mathbf{R}$. Consequently, the convergence behaviour of the LMS algorithm is directly linked to the eigenvalue spread of the correlation matrix $\mathbf{R}$. Furthermore, recalling the relationship between the eigenvalue spread of $\mathbf{R}$ and the power spectrum of $x(n)$, we can say that the convergence of the LMS algorithm is directly related to the flatness in the spectral content of the underlying input process.

Following a similar procedure as in Chapter 5, by manipulating (6.16) we can show that $\mathrm{E}[\mathbf{v}(n)]$ converges to zero when μ remains within the range

$$0 < \mu < \frac{1}{\lambda_{\max}}, \tag{6.17}$$

where $\lambda_{\max}$ is the maximum eigenvalue of $\mathbf{R}$. However, we should point out here that the above range does not necessarily guarantee the stability of the LMS algorithm. *The convergence of the LMS algorithm requires convergence of the mean of* $\mathbf{w}(n)$ *towards* $\mathbf{w}_o$ *and also convergence of the variance of the elements of* $\mathbf{w}(n)$ *to some limited values.* As we shall show later, to guarantee the stability of the LMS algorithm the latter requirement imposes a stringent condition on the size of μ. Furthermore, we may note that the independence assumption used to obtain (6,16) was based on the assumption that μ was very small. The upper limit of μ in (6.17) may badly violate this assumption. Thus, the validity of (6.17), even for the convergence of $\mathrm{E}[\mathbf{w}(n)]$, is questionable.

Example 6.1

Consider the modelling problem of Example 5.1, which is repeated in Figure 6.2, for convenience. As in Example 5.1, the input signal, $x(n)$, is generated by passing a white noise signal, $v(n)$,

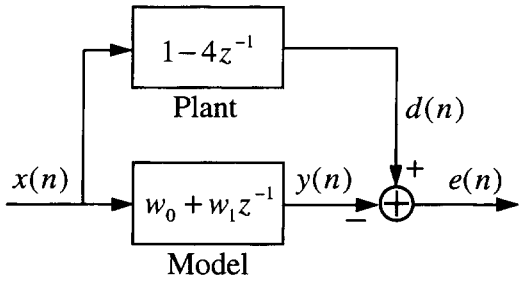


Figure 6.2 A modelling problem

through a colouring filter with the system function

$$H(z) = \frac{\sqrt{1-\alpha^2}}{1-\alpha z^{-1}}, \tag{6.18}$$

where α is a real-valued constant in the range -1 to $+1$. The plant is a two-tap FIR system with the system function $P(z) = 1 - 4z^{-1}$. An adaptive filter with the system function $W(z) = w_0 + w_1 z^{-1}$ is used to identify the plant. Here, the LMS algorithm is used to find the optimum values of the tap weights w_0 and w_1. We want to see, as the iteration number increases, how the tap weights w_0 and w_1 converge toward the plant coefficients, 1 and -4, respectively. We examine this for different values of the parameter α. We recall from Example 5.1 that the parameter α controls the eigenvalue spread of the correlation matrix $\mathbf{R}$ of the input samples to the filter $W(z)$.

Figures 6.3(a), (b), (c) and (d) present four plots showing typical trajectories of the LMS algorithm which have been obtained for the values of $\alpha = 0$, 0.5, 0.75 and 0.9, respectively. Also shown in the figures are the contour plots which highlight the performance surface of the filter. The results presented are for $\mu = 0.01$ and 150 iterations, for all cases. In comparison with the parameters used in Figure 5.4 of Example 5.1, here μ is selected five times smaller, while the number of iterations is chosen five times larger. Comparing the results here with those of Figure 5.4, we can clearly see that, as predicted above, the LMS algorithm, on average, follows the same trajectories as the steepest-descent algorithm. In particular, the convergence of the LMS algorithm along the steepest-descent slope of the performance surface is clearly observed. Also, we note that in the case

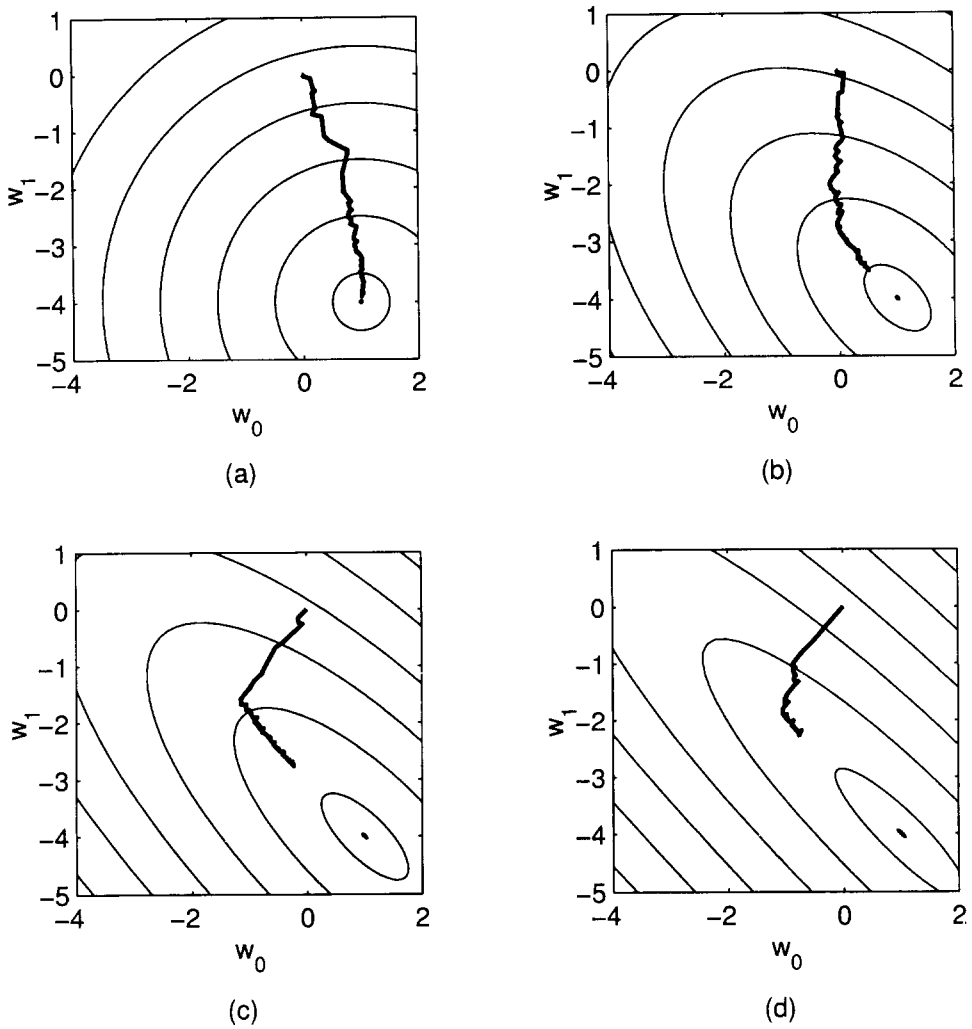


Figure 6.3 Trajectories showing how the filter tap weights vary when the LMS algorithm is used: (a) $\alpha = 0$, (b) $\alpha = 0.5$, (c) $\alpha = 0.75$, and (d) $\alpha = 0.9$. Each plot is based on 150 iterations and $\mu = 0.01$

In the derivations that follow it is assumed that

1. the input, $x(n)$, and desired output, $d(n)$, are zero-mean stationary processes,
2. $x(n)$ and $d(n)$ are jointly Gaussian distributed random variables, for all n, and

results in some simplification in the final results, as the third and higher-order moments that appear in the derivations can be expressed in terms of the second-order moments when the underlying random variables are jointly Gaussian.

6.3.1 Learning curve

We note from (6.11) that the estimation error, $e(n)$, can be expressed as

$$e(n) = e_o(n) - \mathbf{v}^T(n)\mathbf{x}(n). \tag{6.19}$$

Squaring both sides of (6.19) and taking the expectation on both sides, we obtain

$$E[e^2(n)] = E[e_o^2(n)] + E[(\mathbf{v}^T(n)\mathbf{x}(n))^2] - 2E[e_o(n)\mathbf{v}^T(n)\mathbf{x}(n)]. \tag{6.20}$$

Noting that $\mathbf{v}^T(n)\mathbf{x}(n) = \mathbf{x}^T(n)\mathbf{v}(n)$ and using the independence assumption, the second term on the right-hand side of (6.20) can be expanded as[2]

$$\begin{aligned} E[(\mathbf{v}^T(n)\mathbf{x}(n))^2] &= E[\mathbf{v}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{v}(n)] \\ &= E[\mathbf{v}^T(n)E[\mathbf{x}(n)\mathbf{x}^T(n)]\mathbf{v}(n)] \\ &= E[\mathbf{v}^T(n)\mathbf{R}\mathbf{v}(n)]. \end{aligned} \tag{6.21}$$

Noting that $E[(\mathbf{v}^T(n)\mathbf{x}(n))^2]$ is a scalar and using (6.21), we may also write

$$= E\left[\mathrm{tr}[\mathbf{v}^T(n)\mathbf{R}\mathbf{v}(n)]\right], \tag{6.22}$$

Using this identity, we obtain

$$\begin{aligned} \mathrm{E}\left[\mathrm{tr}[\mathbf{v}^{\mathrm{T}}(n)\mathbf{R}\mathbf{v}(n)]\right] &= \mathrm{E}\left[\mathrm{tr}[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)\mathbf{R}]\right] \\ &= \mathrm{tr}\left[\mathrm{E}[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)]\mathbf{R}\right]. \end{aligned} \tag{6.24}$$

Defining the correlation matrix of the weight-error vector $\mathbf{v}(n)$ as

$$\mathbf{K}(n) = \mathrm{E}[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)], \tag{6.25}$$

the above result reduces to

$$\mathrm{E}[(\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n))^2] = \mathrm{tr}[\mathbf{K}(n)\mathbf{R}]. \tag{6.26}$$

Using the independence assumption and noting that $e_{\mathrm{o}}(n)$ is a scalar, the last term on the right-hand side of (6.20) can be written as

$$\begin{aligned} \mathrm{E}[e_{\mathrm{o}}(n)\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)] &= \mathrm{E}[\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)e_{\mathrm{o}}(n)] \\ &= \mathrm{E}[\mathbf{v}^{\mathrm{T}}(n)]\mathrm{E}[\mathbf{x}(n)e_{\mathrm{o}}(n)] \\ &= 0, \end{aligned} \tag{6.27}$$

where the last step follows from the principle of orthogonality which states that the optimal estimation error and the input data samples to a Wiener filter are orthogonal (uncorrelated), i.e. $\mathrm{E}[e_{\mathrm{o}}(n)\mathbf{x}(n)] = 0$.

Using (6.26), and (6.27) in (6.20), we obtain

$$\xi(n) = \mathrm{E}[e^2(n)] = \xi_{\min} + \mathrm{tr}[\mathbf{K}(n)\mathbf{R}] \tag{6.28}$$

where $\xi_{\min} = \mathrm{E}[e_{\mathrm{o}}^2(n)]$, i.e. the minimum mean-square error (MSE) at the filter output.

This result may be written in a more convenient form for future analysis, if we recall from Chapter 4 that the correlation matrix $\mathbf{R}$ may be decomposed as

$$\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{T}}, \tag{6.29}$$

where $\mathbf{Q}$ is the $N \times N$ matrix whose columns are the eigenvectors of $\mathbf{R}$ and $\mathbf{\Lambda}$ is the diagonal matrix consisting of the eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$ of $\mathbf{R}$. Substituting (6.29) in (6.28) and using the identity (6.23), we obtain

$$\xi(n) = \xi_{\min} + \mathrm{tr}[\mathbf{K}'(n)\mathbf{\Lambda}], \tag{6.30}$$

where $\mathbf{K}'(n) = \mathbf{Q}^{\mathrm{T}}\mathbf{K}(n)\mathbf{Q}$. Furthermore, using (6.25), and recalling the definition $\mathbf{v}'(n) = \mathbf{Q}^{\mathrm{T}}\mathbf{v}(n)$ from Chapter 4, we find that

$$\mathbf{K}'(n) = \mathrm{E}[\mathbf{v}'(n)\mathbf{v}'^{\mathrm{T}}(n)]. \tag{6.31}$$

Also, we recall that $\mathbf{v}'(n)$ is the weight-error vector in the coordinates defined by the basis vectors specified by the eigenvectors of $\mathbf{R}$.

Noting that $\mathbf{\Lambda}$ is a diagonal matrix, (6.30) can be expanded as

$$\xi(n) = \xi_{\min} + \sum_{i=0}^{N-1} \lambda_i k'_{ii}(n) \tag{6.32}$$

where $k'_{ij}(n)$ is the ijth element of the matrix $\mathbf{K}'(n)$.

The plot $\xi(n)$ versus the time index n, defined by (6.28) or its alternative forms in (6.30) or (6.32), is called the learning curve of the LMS algorithm. It is very similar to the learning curve of the steepest-descent algorithm, since, according to the derivations in the previous section, the LMS algorithm on average follows the same trajectory as the steepest-descent algorithm. The noisy variations of the filter tap weights in the case of the LMS algorithm introduce some additional error and push up its learning curve compared with that of the steepest-descent algorithm. However, when the step-size parameter, μ, is small (which is usually the case in practice) we find that the difference between the two curves is noticeable only when they have converged and approached their steady state. The following example shows this.

Example 6.2

Figure 6.4 shows the learning curves of the LMS algorithm and the steepest-descent algorithm for the modelling problem discussed in Examples 5.1 and 6.1, when $\alpha = 0.75$ and $\mu = 0.01$. For both cases the filter tap weights have been initialized with $w_0(0) = w_1(0) = 0$. The learning curve of the steepest-descent algorithm has been obtained by inserting the numerical values of the parameters

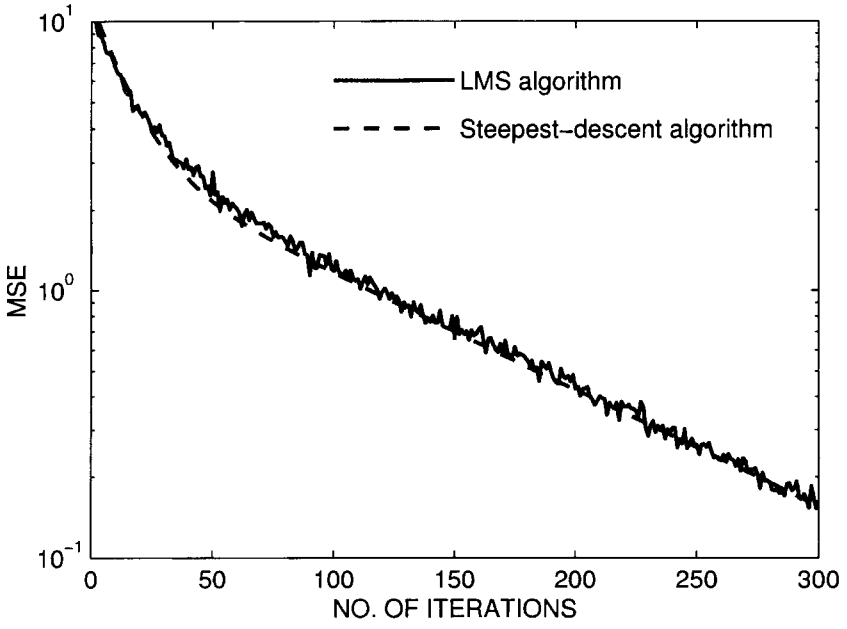


Figure 6.4 Learning curves of the steepest-descent algorithm and the LMS algorithm for the modelling problem of Figure 6.2 and the parameter values of $\alpha = 0.75$ and $\mu = 0.01$

in (5.31). The learning curve of the LMS algorithm is obtained by an ensemble average of the sequence $e^2(n)$ over 1000 independent runs. We note that the two curves match closely. The learning curve of the LMS algorithm remains slightly above the learning curve of the steepest-descent algorithm. This is because of the use of noisy estimates of the gradient vector in the LMS algorithm.

We shall emphasize that, despite the noisy variation of the filter tap weights, the learning curve of the LMS algorithm matches closely the theoretical results of the steepest-descent algorithm. In particular, (5.31) is applicable and the time constant equation

$$\tau_i = \frac{1}{4\mu\lambda_i} \tag{6.33}$$

can be used for predicting the transient behaviour of the LMS algorithm.

6.3.2 The weight-error correlation matrix

The weight-error correlation matrix $\mathbf{K}(n)$ plays an important role in the study of the LMS algorithm. From (6.28) we note that the value of $\xi(n)$ is directly related to $\mathbf{K}(n)$. Equation (6.28) implies that the stability of the LMS algorithm is guaranteed if, as n increases, the elements of $\mathbf{K}(n)$ remain bounded. Also, from (6.30) and (6.32) we note that $\mathbf{K}'(n)$ may equivalently be used in the study of the convergence of the LMS algorithm. Here, we develop a time-update equation for $\mathbf{K}'(n)$.

Multiplying both sides of (6.13) from the left by $\mathbf{Q}^{\mathrm{T}}$, using the definitions $\mathbf{v}'(n) = \mathbf{Q}^{\mathrm{T}}\mathbf{v}(n)$ and $\mathbf{x}'(n) = \mathbf{Q}^{\mathrm{T}}\mathbf{x}(n)$, and rearranging the result, we obtain

$$\mathbf{v}'(n+1) = (\mathbf{I} - 2\mu\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n))\mathbf{v}'(n) + 2\mu e_{\mathrm{o}}(n)\mathbf{x}'(n). \tag{6.34}$$

Next, we multiply both sides of (6.37) from the right by their respective transposes, take statistical expectation of the result and expand to obtain

$$\begin{aligned}
\mathbf{K}'(n+1) = \mathbf{K}'(n) &- 2\mu\mathrm{E}[\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\mathbf{v}'(n)\mathbf{v}'^{\mathrm{T}}(n)] \\
&- 2\mu\mathrm{E}[\mathbf{v}'(n)\mathbf{v}'^{\mathrm{T}}(n)\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)] \\
&+ 4\mu^2\mathrm{E}[\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\mathbf{v}'(n)\mathbf{v}'^{\mathrm{T}}(n)\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)] \\
&+ 2\mu\mathrm{E}[e_{\mathrm{o}}(n)\mathbf{x}'(n)\mathbf{v}'^{\mathrm{T}}(n)] \\
&+ 2\mu\mathrm{E}[e_{\mathrm{o}}(n)\mathbf{v}'(n)\mathbf{x}'^{\mathrm{T}}(n)] \\
&- 4\mu^2\mathrm{E}[e_{\mathrm{o}}(n)\mathbf{x}'(n)\mathbf{v}'^{\mathrm{T}}(n)\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)] \\
&- 4\mu^2\mathrm{E}[e_{\mathrm{o}}(n)\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\mathbf{v}'(n)\mathbf{x}'^{\mathrm{T}}(n)] \\
&+ 4\mu^2\mathrm{E}[e_{\mathrm{o}}^2(n)\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)].
\end{aligned} \tag{6.35}$$

We note that the independence assumption (which states that $\mathbf{v}(n)$ is independent of $\mathbf{x}(n)$ and $d(n)$) is also applicable to the transformed (prime) variables in (6.35). That is, the

random vector $\mathbf{v}'(n)$ is independent of $\mathbf{x}'(n)$ and $d(n)$. This is immediately observed if we note that $\mathbf{x}'(n)$ and $\mathbf{v}'(n)$ are independently obtained from $\mathbf{x}(n)$ and $\mathbf{v}(n)$, respectively. Also, the assumption that $d(n)$ and $x(n)$ are zero-mean and mutually Gaussian-distributed implies that $d(n)$ and $\mathbf{x}'(n)$ are also zero-mean and jointly Gaussian. Furthermore, using the definition $\mathbf{x}'(n) = \mathbf{Q}^{\mathrm{T}}\mathbf{x}(n)$, we note that the principle of orthogonality, i.e. $\mathrm{E}[e_{\mathrm{o}}(n)\mathbf{x}(n)] = \mathbf{0}$, may also be written as

$$\mathrm{E}[e_{\mathrm{o}}(n)\mathbf{x}'(n)] = \mathbf{0}. \tag{6.36}$$

Noting this, which shows that $e_{\mathrm{o}}(n)$ and $\mathbf{x}'(n)$ are uncorrelated, and the fact that $d(n)$ and $\mathbf{x}'(n)$ and, thus, $e_{\mathrm{o}}(n)$ and $\mathbf{x}'(n)$ are jointly Gaussian, we can say that the random variables $e_{\mathrm{o}}(n)$ and $\mathbf{x}'(n)$ are independent of each other.[3] Also, the independence of $\mathbf{v}'(n)$ from $d(n)$ and $\mathbf{x}(n)$ implies that $\mathbf{v}'(n)$ and $e_{\mathrm{o}}(n)$ are independent, since $e_{\mathrm{o}}(n)$ depends only on $d(n)$ and $\mathbf{x}(n)$. With these points in mind, the expectations on the right-hand side of (6.35) can be simplified as follows:

$$\begin{aligned}\mathrm{E}[\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\mathbf{v}'(n)\mathbf{v}'^{\mathrm{T}}(n)] &= \mathrm{E}[\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)]\mathrm{E}[\mathbf{v}'(n)\mathbf{v}'^{\mathrm{T}}(n)] \\ &= \mathbf{\Lambda}\mathbf{K}'(n)\end{aligned} \tag{6.37}$$

where we have noted that $\mathrm{E}[\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)] = \mathbf{\Lambda}$. Similarly

$$\mathrm{E}[\mathbf{v}'(n)\mathbf{v}'^{\mathrm{T}}(n)\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)] = \mathbf{K}'(n)\mathbf{\Lambda}. \tag{6.38}$$

Simplification of the third expectation requires some algebraic manipulations. These are provided in Appendix 6A. The result is

$$\mathrm{E}[\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\mathbf{v}'(n)\mathbf{v}'^{\mathrm{T}}(n)\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)] = 2\mathbf{\Lambda}\mathbf{K}'(n)\mathbf{\Lambda} + \mathrm{tr}[\mathbf{\Lambda}\mathbf{K}'(n)]\mathbf{\Lambda}. \tag{6.39}$$

Using the independence of $e_{\mathrm{o}}(n)$, $\mathbf{x}'(n)$ and $\mathbf{v}'(n)$ and noting that $e_{\mathrm{o}}(n)$ has zero mean, we get

$$\mathrm{E}[e_{\mathrm{o}}(n)\mathbf{x}'(n)\mathbf{v}'^{\mathrm{T}}(n)] = \mathrm{E}[e_{\mathrm{o}}(n)]\mathrm{E}[\mathbf{x}'(n)\mathbf{v}'^{\mathrm{T}}(n)] = \mathbf{0}, \tag{6.40}$$

where $\mathbf{0}$ denotes the $N \times N$ zero matrix. Similarly,

$$\mathrm{E}[e_{\mathrm{o}}(n)\mathbf{v}'(n)\mathbf{x}'^{\mathrm{T}}(n)] = \mathbf{0}, \tag{6.41}$$

$$\mathrm{E}[e_{\mathrm{o}}(n)\mathbf{x}'(n)\mathbf{v}'^{\mathrm{T}}(n)\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)] = \mathbf{0}, \tag{6.42}$$

$$\mathrm{E}[e_{\mathrm{o}}(n)\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\mathbf{v}'(n)\mathbf{x}'^{\mathrm{T}}(n)] = \mathbf{0}, \tag{6.43}$$

[3] We recall that when the random variables x and y are jointly Gaussian and uncorrelated, they are also independent (see Papoulis, 1991).

and

$$\begin{aligned} \mathrm{E}[e_o^2(n)\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)] &= \mathrm{E}[e_o^2(n)]\mathrm{E}[\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)] \\ &= \xi_{\min}\mathbf{\Lambda}. \end{aligned} \tag{6.44}$$

Substituting (6.37)–(6.44) in (6.35), we obtain

$$\begin{aligned} \mathbf{K}'(n+1) = \mathbf{K}'(n) &- 2\mu(\mathbf{\Lambda}\mathbf{K}'(n) + \mathbf{K}'(n)\mathbf{\Lambda}) \\ &+ 8\mu^2\mathbf{\Lambda}\mathbf{K}'(n)\mathbf{\Lambda} + 4\mu^2\mathrm{tr}[\mathbf{\Lambda}\mathbf{K}'(n)]\mathbf{\Lambda} + 4\mu^2\xi_{\min}\mathbf{\Lambda}. \end{aligned} \tag{6.45}$$

The difference equation (6.45) is difficult to handle. However, the fact that $\mathbf{\Lambda}$ is a diagonal matrix can be used to simplify the analysis. Consider the ith diagonal element of $\mathbf{K}'(n)$ and note that its corresponding time-update equation, obtained from (6.45), is

$$k'_{ii}(n+1) = \rho_i k'_{ii}(n) + 4\mu^2\lambda_i \sum_{j=0}^{N-1} \lambda_j k'_{jj}(n) + 4\mu^2\xi_{\min}\lambda_i, \tag{6.46}$$

where

$$\rho_i = 1 - 4\mu\lambda_i + 8\mu^2\lambda_i^2 \tag{6.47}$$

and we have noted that

$$\mathrm{tr}[\mathbf{\Lambda}\mathbf{K}'(n)] = \sum_{j=0}^{N-1} \lambda_j k'_{jj}(n). \tag{6.48}$$

The important feature of (6.46) to be noted is that the update of $k'_{ii}(n)$ is independent of the off-diagonal elements of $\mathbf{K}'(n)$. Furthermore, we note that since $\mathbf{K}'(n)$ is a correlation matrix, $k'^2_{ij}(n) \leq k'_{ii}(n)k'_{jj}(n)$ for all values of i and j. This suggests that the convergence of the diagonal elements of $\mathbf{K}'(n)$ is sufficient to ensure the convergence of all elements of it, which, in turn, are required to guarantee the stability of the LMS algorithm. Thus, we concentrate on (6.46), for $i = 0, 1, \ldots, N-1$.

Let us define the column vectors

$$\mathbf{k}'(n) = [k'_{00}(n) \;\; k'_{11}(n) \;\; \ldots \;\; k'_{N-1,N-1}(n)]^{\mathrm{T}} \tag{6.49}$$

and

$$\boldsymbol{\lambda} = [\lambda_0 \;\; \lambda_1 \;\; \ldots \;\; \lambda_{N-1}]^{\mathrm{T}} \tag{6.50}$$

and the matrix

$$\mathbf{F} = \mathrm{diag}[\rho_0, \rho_1, \ldots, \rho_{N-1}] + 4\mu^2\boldsymbol{\lambda}\boldsymbol{\lambda}^{\mathrm{T}}, \tag{6.51}$$

where diag[$\cdots$] refers to a diagonal matrix consisting of the indicated elements. Considering these definitions and the time-update equation (6.46), for $i = 0, 1, \ldots, N-1$, we get

$$\mathbf{k}'(n+1) = \mathbf{F}\mathbf{k}'(n) + 4\mu^2\xi_{\min}\boldsymbol{\lambda}. \tag{6.52}$$

The difference equation (6.52) can be used to study the stability of the LMS algorithm. As was noted before, the stability of the LMS algorithm is guaranteed if the elements of $\mathbf{K}(n)$ (or, equivalently, the elements of $\mathbf{k}'(n)$) remain bounded, as n increases. The necessary and sufficient condition for this to happen is that all the eigenvalues of the coefficient matrix $\mathbf{F}$ of (6.52) be less than one in magnitude. Feuer and Weinstein (1985) have discussed the eigenvalues of $\mathbf{F}$ and given the condition required to keep the LMS algorithm stable. Here, we will comment on the stability of the LMS algorithm in an indirect way. This is done after we find an expression for the excess MSE of the LMS algorithm, which is defined below.

6.3.3 Excess MSE and misadjustment

We note that even when the filter tap-weight vector $\mathbf{w}(n)$ approaches its optimal value, $\mathbf{w}_o$, and the mean of the stochastic gradient vector $\nabla e^2(n)$ tends to zero, the instantaneous value of this gradient may not be zero. This results in a perturbation of the tap-weight vector $\mathbf{w}(n)$ around its optimal value, $\mathbf{w}_o$, even after convergence of the algorithm. This, in turn, increases the MSE of the LMS algorithm to a level above the minimum MSE that would be obtained if the filter tap weights were fixed at their optimal values. This additional error is called the *excess MSE*. In other words, the excess MSE of an adaptive filter is defined as the difference between its steady-state MSE and its minimum MSE.

The steady-state MSE of the LMS algorithm can be found from (6.28) or, equivalently, (6.30) or (6.32) by letting the time-index n tend to infinity. Thus, subtracting $\xi_{\min}$ from both sides of (6.28), we obtain

$$\xi_{\text{excess}} = \text{tr}[\mathbf{K}(\infty)\mathbf{R}] \tag{6.53}$$

where ξ_{excess} denotes the excess MSE. Alternatively, if (6.32) is used, we get

$$\xi_{\text{excess}} = \sum_{i=0}^{N-1} \lambda_i k'_{ii}(\infty) = \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{k}'(\infty). \tag{6.54}$$

When the LMS algorithm is convergent, $\mathbf{k}'(n)$ converges to a bounded steady-state value and we can say $\mathbf{k}'(n+1) = \mathbf{k}'(n)$, when $n \to \infty$. Noting this, from (6.52) we obtain

$$\mathbf{k}'(\infty) = 4\mu^2\xi_{\min}(\mathbf{I} - \mathbf{F})^{-1}\boldsymbol{\lambda}. \tag{6.55}$$

Substituting this in (6.54) we get

$$\xi_{\text{excess}} = 4\mu^2\xi_{\min}\boldsymbol{\lambda}^{\mathrm{T}}(\mathbf{I} - \mathbf{F})^{-1}\boldsymbol{\lambda}. \tag{6.56}$$

We note that ξ_{excess} is proportional to ξ_{min}. This is intuitively understandable if we note that when $\mathbf{w}(n)$ has converged to a vicinity of $\mathbf{w}_o$ the variance of the elements of the stochastic gradient vector $\nabla e^2(n)$ are proportional to ξ_{min} (see Problem P6.1). We also note that similar to ξ_{min}, ξ_{excess} also has the units of power. It is convenient to normalize ξ_{excess} to ξ_{min}, so that a dimension-free degradation measure is obtained. The result is called misadjustment and denoted as $\mathcal{M}$. For the LMS algorithm, from (6.56) we obtain

$$\mathcal{M} = \frac{\xi_{excess}}{\xi_{min}} = 4\mu^2 \boldsymbol{\lambda}^T (\mathbf{I} - \mathbf{F})^{-1} \boldsymbol{\lambda}. \tag{6.57}$$

The special structure of the matrix $(\mathbf{I} - \mathbf{F})$ can be used to find its inverse.

We note from (6.51) that

$$\mathbf{I} - \mathbf{F} = \text{diag}[1 - \rho_0, 1 - \rho_1, \ldots, 1 - \rho_{N-1}] - 4\mu^2 \boldsymbol{\lambda}\boldsymbol{\lambda}^T. \tag{6.58}$$

On the other hand, we note that according to the matrix inversion lemma,[4] for an arbitrary positive-definite $N \times N$ matrix $\mathbf{A}$, any $N \times 1$ vector $\mathbf{a}$ and a scalar α,

$$(\mathbf{A} + \alpha \mathbf{a}\mathbf{a}^T)^{-1} = \mathbf{A}^{-1} - \frac{\alpha \mathbf{A}^{-1}\mathbf{a}\mathbf{a}^T\mathbf{A}^{-1}}{1 + \alpha \mathbf{a}^T\mathbf{A}^{-1}\mathbf{a}}. \tag{6.59}$$

Letting $\mathbf{A} = \text{diag}[1 - \rho_0, 1 - \rho_1, \ldots, 1 - \rho_{N-1}]$, $\mathbf{a} = \boldsymbol{\lambda}$ and $\alpha = -4\mu^2$ in (6.59) to obtain the inverse of $(\mathbf{I} - \mathbf{F})$, substituting the result in (6.57), and after some straightforward manipulations, we get

$$\mathcal{M} = \frac{\displaystyle\sum_{i=0}^{N-1} \frac{\mu\lambda_i}{1 - 2\mu\lambda_i}}{1 - \displaystyle\sum_{i=0}^{N-1} \frac{\mu\lambda_i}{1 - 2\mu\lambda_i}}. \tag{6.60}$$

It is useful to simplify this result by making some appropriate approximations, so that it can conveniently be used for the selection of the step-size parameter, μ. In practice, one usually selects μ so that a misadjustment of 10% ($\mathcal{M} = 0.1$) or less is achieved. In that case, we may find that

$$\sum_{i=0}^{N-1} \frac{\mu\lambda_i}{1 - 2\mu\lambda_i} \approx \mu \sum_{i=0}^{N-1} \lambda_i = \mu \,\text{tr}[\mathbf{R}], \tag{6.61}$$

---

[4] The general form of the matrix inversion lemma states that if $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are, respectively, $N \times N$, $M \times M$ and $N \times M$ matrices and the necessary inverses exist, then

$$(\mathbf{A} + \mathbf{C}\mathbf{B}\mathbf{C}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{C}(\mathbf{B}^{-1} + \mathbf{C}^T\mathbf{A}^{-1}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{A}^{-1}.$$

Clearly, the identity (6.59) is a special case of this.

where the last equality is obtained from (4.24). This approximation is understood if we note that when $\mathcal{M}$ is small, the summation on the left-hand side of (6.61) is also small. Moreover, when the latter summation is small, $\mu\lambda_i \ll 1$, for $i = 0, 1, \ldots, N-1$, and thus these may be deleted from the denominators of the terms under the summation on the right-hand side of (6.60). Thus, we obtain

$$\mathcal{M} = \frac{\mu \operatorname{tr}[\mathbf{R}]}{1 - \mu \operatorname{tr}[\mathbf{R}]}. \tag{6.62}$$

Furthermore, we note that when $\mathcal{M}$ is small, say $\mathcal{M} \leq 0.1$, $\mu \operatorname{tr}[\mathbf{R}]$ is also small, and thus it may be ignored in the denominator of (6.62) to obtain

$$\mathcal{M} \approx \mu \operatorname{tr}[\mathbf{R}]. \tag{6.63}$$

This is a very convenient equation, as $\operatorname{tr}[\mathbf{R}]$ is equal to the sum of the powers of the signal samples at the filter tap inputs. This can be easily measured and used for the selection of the step-size parameter, μ, for achieving a certain level of misadjustment. Furthermore, when the input process to the filter is non-stationary, estimate of $\operatorname{tr}[\mathbf{R}]$ may be updated recursively and the step-size parameter, μ, chosen accordingly to keep a certain level of misadjustment.

6.3.4 Stability

In Chapter 5 we noted that the steepest-descent algorithm remains stable only when its corresponding step-size parameter, μ, takes a value between zero and an upper bound value which was found to be dependent on the statistics of the filter input. The same is true for the LMS algorithm. However, the use of a stochastic gradient in the LMS algorithm makes it more sensitive to the value of its step-size parameter, μ, and, as a result, the upper bound of μ, which can ensure the stable behaviour of the LMS algorithm, is much lower than the corresponding bound in the case of the steepest-descent algorithm. To find the upper bound of μ that guarantees the stability of the LMS algorithm, we elaborate on the misadjustment equation (6.60).

We define

$$\mathcal{J} = \sum_{i=0}^{N-1} \frac{\mu\lambda_i}{1 - 2\mu\lambda_i} \tag{6.64}$$

and note that

$$\mathcal{M} = \frac{\mathcal{J}}{1 - \mathcal{J}}. \tag{6.65}$$

We also note that

$$\frac{\partial \mathcal{J}}{\partial \mu} = \sum_{i=0}^{N-1} \frac{\lambda_i}{(1 - 2\mu\lambda_i)^2}. \tag{6.66}$$

From (6.66) we note that $\mathcal{J}$ is an increasing function of μ, since its derivative with respect to μ is always positive. In a similar way, we can show that $\mathcal{M}$ is an increasing function of $\mathcal{J}$. This, in turn, implies that *the misadjustment $\mathcal{M}$ of* (6.60) *is an increasing function of the step-size parameter, μ*. Thus, starting with $\mu = 0$ (i.e. the lower bound of μ) and increasing μ, we find that $\mathcal{J}$ and $\mathcal{M}$ also start from zero and increase with μ. We also note that when $\mathcal{J}$ approaches unity, $\mathcal{M}$ tends to infinity. This clearly coincides with the upper bound of the step-size parameter, say $\mu_{\max}$, below which μ has to remain to ensure the stable behaviour of the LMS algorithm. Thus, the value of $\mu_{\max}$ is obtained by finding the first positive root of the equation

$$\sum_{i=0}^{N-1} \frac{\mu\lambda_i}{1 - 2\mu\lambda_i} = 1. \tag{6.67}$$

Finding the exact solution of this problem, in general, turns out to be a difficult mathematical task. Furthermore, from a practical point of view, such a solution is not rewarding because it depends on the statistics of the filter input in a complicated way. Here, we give an upper bound of μ that depends only on $\sum_{i=0}^{N-1} \lambda_i = \text{tr}[\mathbf{R}]$. This results in a smaller (more stringent) value as the upper bound of μ, but a value that can easily be measured in practice. For this we note that when

$$0 < \mu < \frac{1}{2\sum_{i=0}^{N-1} \lambda_i}, \tag{6.68}$$

the following inequality always holds:

$$\sum_{i=0}^{N-1} \frac{\mu\lambda_i}{1 - 2\mu\lambda_i} \leq \frac{\mu\sum_{i=0}^{N-1} \lambda_i}{1 - 2\mu\sum_{i=0}^{N-1} \lambda_i}. \tag{6.69}$$

The proof of this inequality is discussed in Problem P6.4. From (6.69), we find that the value of μ that satisfies the equation

$$\frac{\mu\sum_{i=0}^{N-1} \lambda_i}{1 - 2\mu\sum_{i=0}^{N-1} \lambda_i} = 1, \tag{6.70}$$

satisfies the inequality

$$\sum_{i=0}^{N-1} \frac{\mu\lambda_i}{1 - 2\mu\lambda_i} \leq 1. \tag{6.71}$$

Furthermore, any value of μ that remains between zero and the solution of (6.70) satisfies (6.71). This means that (6.70) gives an upper bound for μ which is sufficient for the stability of the LMS algorithm, but is not necessary in general. If we call the solution of (6.70) $\mu'_{\max}$, we obtain

$$\mu'_{\max} = \frac{1}{3\sum_{i=0}^{N-1} \lambda_i} = \frac{1}{3\text{tr}[\mathbf{R}]}. \tag{6.72}$$

To summarize, we found that, under the assumptions made at the beginning of this section, the LMS algorithm remains stable when

$$0 < \mu < \frac{1}{3\text{tr}[\mathbf{R}]}. \tag{6.73}$$

The significance of the upper bound of μ, which is provided by (6.73), is that it can easily be measured from the filter input samples. We also note that the range of μ that is provided by (6.73) is sufficient for the stability of the LMS algorithm, but is not necessary. The first positive root of (6.67) gives a more accurate upper bound of μ. However, this depends on the filter input statistics in a very complicated way which prohibits its applicability in actual practice.

6.3.5 The effect of initial values of tap weights on the transient behaviour of the LMS algorithm

As was noted before, the LMS algorithm on average follows the same trajectory as the steepest-descent algorithm. As a result, the learning curves of the two algorithms are found to be similar when the same step-size parameter is used for both. In particular, the learning curve equation (5.31) is also (approximately) applicable to the LMS algorithm. Thus, we may write

$$\xi(n) \approx \xi_{\min} + \sum_{i=0}^{N-1} \lambda_i (1 - 2\mu\lambda_i)^{2n} v_i'^2(0). \tag{6.74}$$

In most applications the filter tap weights are all initialized to zero. In that case,

$$\mathbf{v}(0) = \mathbf{w}(0) - \mathbf{w}_o = -\mathbf{w}_o. \tag{6.75}$$

Using this result and recalling the definition $\mathbf{v}'(0) = \mathbf{Q}^T\mathbf{v}(0)$, we get

$$\mathbf{v}'(0) = -\mathbf{w}_o' \tag{6.76}$$

where $\mathbf{w}_o' = \mathbf{Q}^T\mathbf{w}_o$. Using (6.76) in (6.74), we obtain

$$\xi(n) \approx \xi_{\min} + \sum_{i=0}^{N-1} \lambda_i (1 - 2\mu\lambda_i)^{2n} w_{o,i}'^2 \tag{6.77}$$

where $w_{o,i}'$ is the ith element of $\mathbf{w}_o'$.

The contribution of various modes of convergence of the LMS algorithm (i.e. the terms under the summation on the right-hand side of (6.74)) on its learning curve depends on the $\lambda_i w_{o,i}'^2$ coefficients. As a result, we find that even for a similar eigenvalue distribution, the convergence behaviour of the LMS algorithm is application dependent. For instance, if the $w_{o,i}'$s corresponding to the smaller eigenvalues of $\mathbf{R}$ are all close to zero, then the transient behaviour of the LMS algorithm is determined by the larger

eigenvalues of **R** whose associated time constants are small, thus a fast convergence is observed. On the contrary, if the $w'_{o,i}$s corresponding to the smaller eigenvalues of **R** are significantly large, then we find that the slower modes of the LMS algorithm are prominent on its learning curve. Examples given in the next section show that these two extreme cases can happen in practice.

6.4 Computer Simulations

Computer simulation plays a major role in the study of adaptive filters. In the analysis presented in the previous section, we had to consider a number of assumptions to make the problem mathematically tractable. The validity of these assumptions and the matching between mathematical results and the actual performance of adaptive filters are usually verified through computer simulations.

In this section we present a few examples of computer simulations. We present examples of four different applications of adaptive filters:

- System modelling
- Channel equalization
- Adaptive line enhancement (this is an example of prediction)
- Beamforming.

Our objectives in this presentation are to:

1. Help the novice reader to make a quick start at doing computer simulations.
2. Check the accuracy of the developed theoretical results.
3. Enhance the understanding of the theoretical results by careful observation and interpretation of simulation results.

All the results given below have been generated by using the MATLAB numerical package. The MATLAB programs used to generate the results presented in this section and other parts of this book are available on an accompanying diskette. A list of these

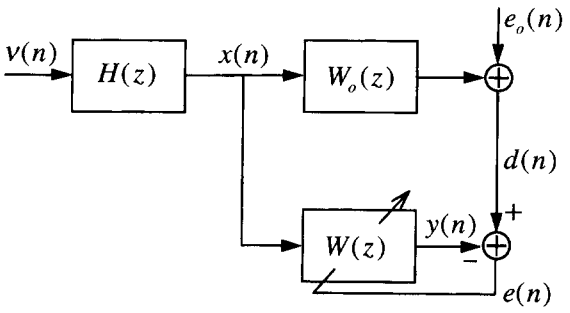


Figure 6.5 Adaptive modelling of an FIR plant

We present the results of simulations for two choices of input which are characterized by

$$H(z) = H_1(z) = 0.35 + z^{-1} - 0.35z^{-2} \tag{6.79}$$

and

$$H(z) = H_2(z) = 0.35 + z^{-1} + 0.35z^{-2}. \tag{6.80}$$

The first choice results in an input, $x(n)$, whose corresponding correlation matrix has an eigenvalue spread of 1.45. This is close to white input. On the contrary, the second choice of $H(z)$ results is a highly coloured input with an associated eigenvalue spread of 28.7. From the results of Chapter 4, we recall that the eigenvalue spread figures can approximately be obtained from the underlying power spectral densities. Figure 6.6

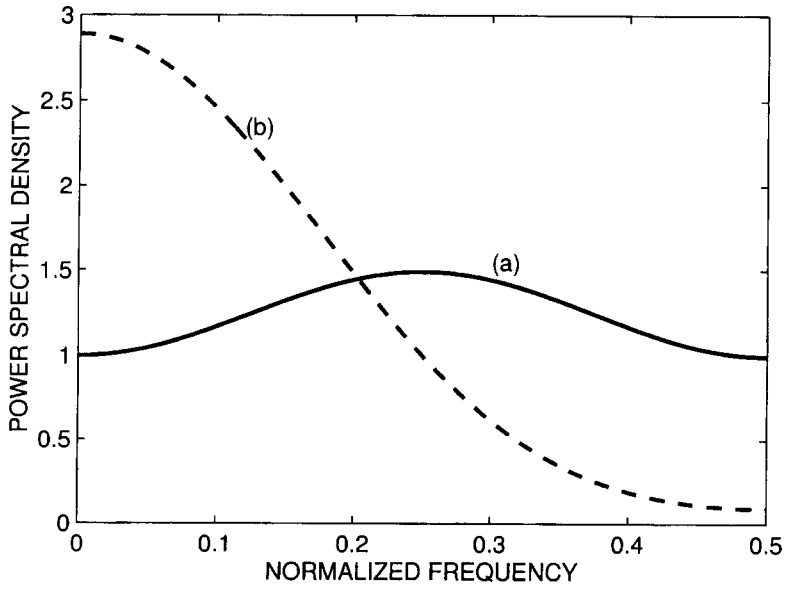


Figure 6.6 Power spectral densities of the two input processes used for the simulation of the modelling problem: (a) $H(z) = H_1(z)$, (b) $H(z) = H_2(z)$

shows the power spectral densities of the two inputs generated by using the filters $H_1(z)$ and $H_2(z)$. These plots are obtained by noting that

$$\Phi_{xx}(e^{j\omega}) = \Phi_{\nu\nu}(e^{j\omega})|H(e^{j\omega})|^2 \tag{6.81}$$

and $\Phi_{\nu\nu}(e^{j\omega}) = 1$, since $\nu(n)$ is a unit variance white noise process. The fact that $H_2(z)$ generates a process that is highly coloured, while the process generated by $H_1(z)$ is relatively flat, is clearly seen.

Figures 6.7(a) and (b) show the learning curves of the LMS algorithm for the two choices of $H(z)$. The step-size parameter, μ, is selected according to the simplified misadjustment equation (6.63) for the misadjustment values 10%, 20% and 30%. The filter tap weights are initialized to zero. Each plot is obtained by an ensemble average of 100 independent simulation runs. We note that $\xi_{\min} = \sigma_o^2 = 0.001$, and this is achieved when the model and plant coefficients match. Careful examination of the results presented in Figures 6.7(a) and (b) reveals that the predictions made by (6.63) are accurate for the cases where μ is set for a misadjustment of 10% (or less). For larger values of μ, we find that a more accurate theoretical estimate of the misadjustment is obtained by using equation (6.60). Such an estimate, of course, requires calculation of the eigenvalues of the correlation matrix **R**. The MATLAB program 'modelling.m' on the accompanying diskette contains instructions that generate the matrix **R** and the other parameters required for these calculations. The reader is encouraged to use this program and experiment with it to examine the effect of various parameters, such as the step size, μ, the plant model, $W_o(z)$, and the input sequence to the adaptive filter. Such experiments will greatly enhance the reader's understanding of the concepts of convergence and misadjustment.

Experiments with the LMS algorithm show that the accuracy of the misadjustment equations developed above varies with the statistics of the filter input and the step-size parameter. For example, we find that all of the three plots in Figuse 6.7(a) and two of the plots in Figure 6.7(b) match the theoretical predictions made by (6.60), but the third plot in Figure 6.7(b) (i.e. the case $\mathcal{M} = 30\%$) does not match (6.60). In the latter case the LMS algorithm experiences some instability problem. The mismatch between the theory and experiments here is attributed to the fact that the independence assumption made in the development of the theoretical results is badly violated for larger values of μ.

6.4.2 Channel equalization

Figure 6.8 depicts a channel equalization problem. The input sequence to the channel is assumed to be binary (taking values of $+1$ and -1) and white. The channel system function is denoted by $H(z)$. The channel noise, $\nu_c(n)$, is modelled as an additive white Gaussian process with variance $\sigma_{\nu_c}^2$. The equalizer is implemented as an N-tap transversal filter. The desired output of the equalizer is assumed to be $s(n - \Delta)$, i.e. a delayed replica of the transmitted data symbols. For the training of the equalizer, it is assumed that the transmitted data symbols are available at the receiver. This is called the training mode. Once the equalizer is trained and switched to the data mode, its output, after passing through a slicer, gives the transmitted symbols. A discussion on the training and data mode of equalizers can be found in Chapter 1.

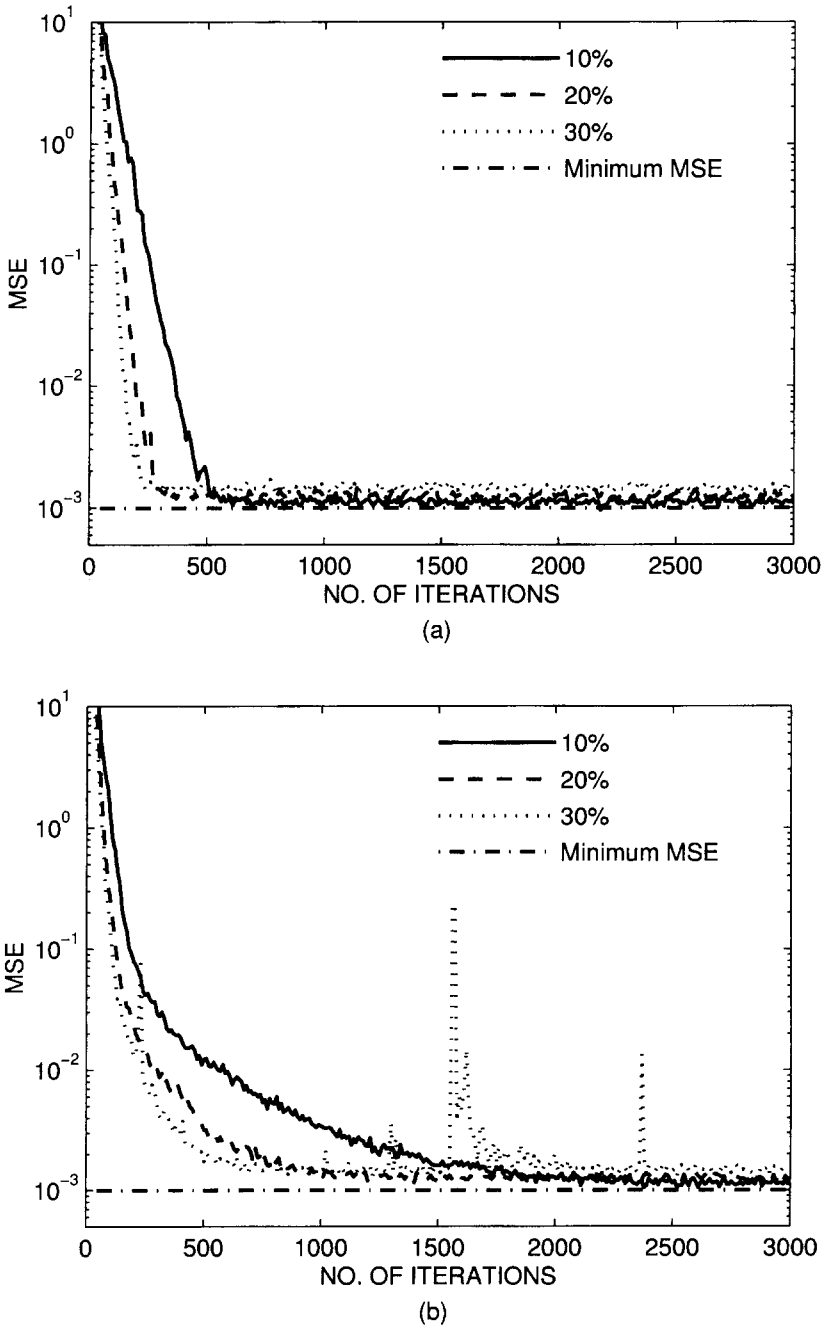


Figure 6.7 Learning curves of the LMS algorithm for the modelling problem of Figure 6.5, for the two input processes discussed in the text: (a) $H(z) = H_1(z)$ and (b) $H(z) = H_2(z)$. The step-size parameter, μ, is selected for the misadjustment values 10%, 20% and 30%, according to the simplified equation (6.63)

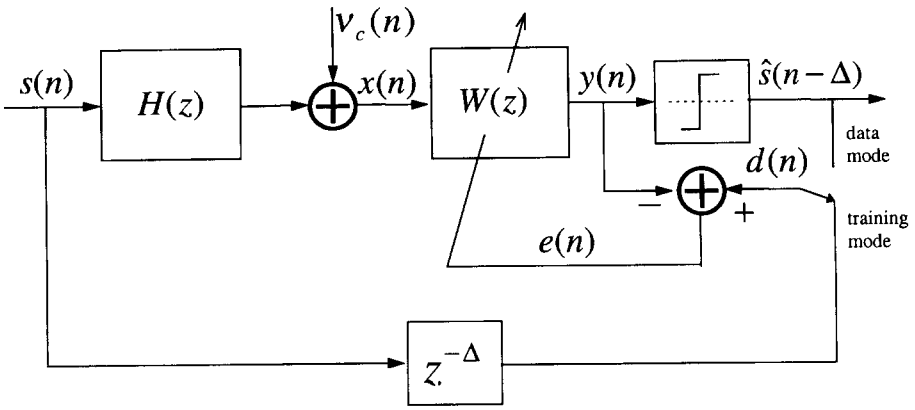


Figure 6.8 Adaptive channel equalization

Two choices of the channel response, $H(z)$, are considered for our study here. These are purposefully selected to be the same as the two choices of $H(z)$ in the modelling problem above, where $H(z)$ was used to shape the power spectral density of the input process to the plant and model. This facilitates a comparison of the results in the two cases. In particular, we note that, in the present problem,

$$\begin{aligned}\Phi_{xx}(e^{j\omega}) &= \Phi_{ss}(e^{j\omega})|H(e^{j\omega})|^2 + \Phi_{\nu_c\nu_c}(e^{j\omega}) \\ &= |H(e^{j\omega})|^2 + \sigma_{\nu_c}^2. \end{aligned} \tag{6.82}$$

Comparing (6.81) and (6.82) we note that when a similar $H(z)$ is used for both cases and the signal-to-noise ratio at the channel output is high (i.e. $\sigma_{\nu_c}^2$ is small), the power spectral densities of the input samples to the two adaptive filters are almost the same. This, in turn, implies that the convergence of both filters is controlled by the same set of eigenvalues. As a result, on average we may expect to see similar learning curves for both cases.

Figures 6.9(a) and (b) present the learning curves of the equalizer for the two choices of the channel response, i.e. $H_1(z)$ and $H_2(z)$ of (6.79) and (6.80), respectively. The equalizer length, N, and the delay, Δ, are set equal to 15 and 9, respectively. The step-size parameter, μ, is chosen according to the simplified equation (6.63) for the three misadjustment values 10%, 20% and 30%. The equalizer tap weights are initialized to zero. Each plot is based on an ensemble average of 100 independent simulation runs. The MATLAB program used to obtain these results is available on the accompanied diskette. It is called 'equalizer.m'. Careful study of Figures 6.9(a) and (b) and further numerical tests (using the 'equalizer.m' or any similar simulation program) reveal that similar to the modelling case, the theoretical and simulation results match well when the step-size parameter, μ, is small. However, the accuracy of the theoretical results is lost for larger values of μ. The latter effect is more noticeable when the eigenvalue spread of the correlation matrix $\mathbf{R}$ is large.

Comparing the results presented in Figures 6.7(a) and 6.9(a), we find that the performance of the adaptive filters in both cases are about the same. Moreover, these

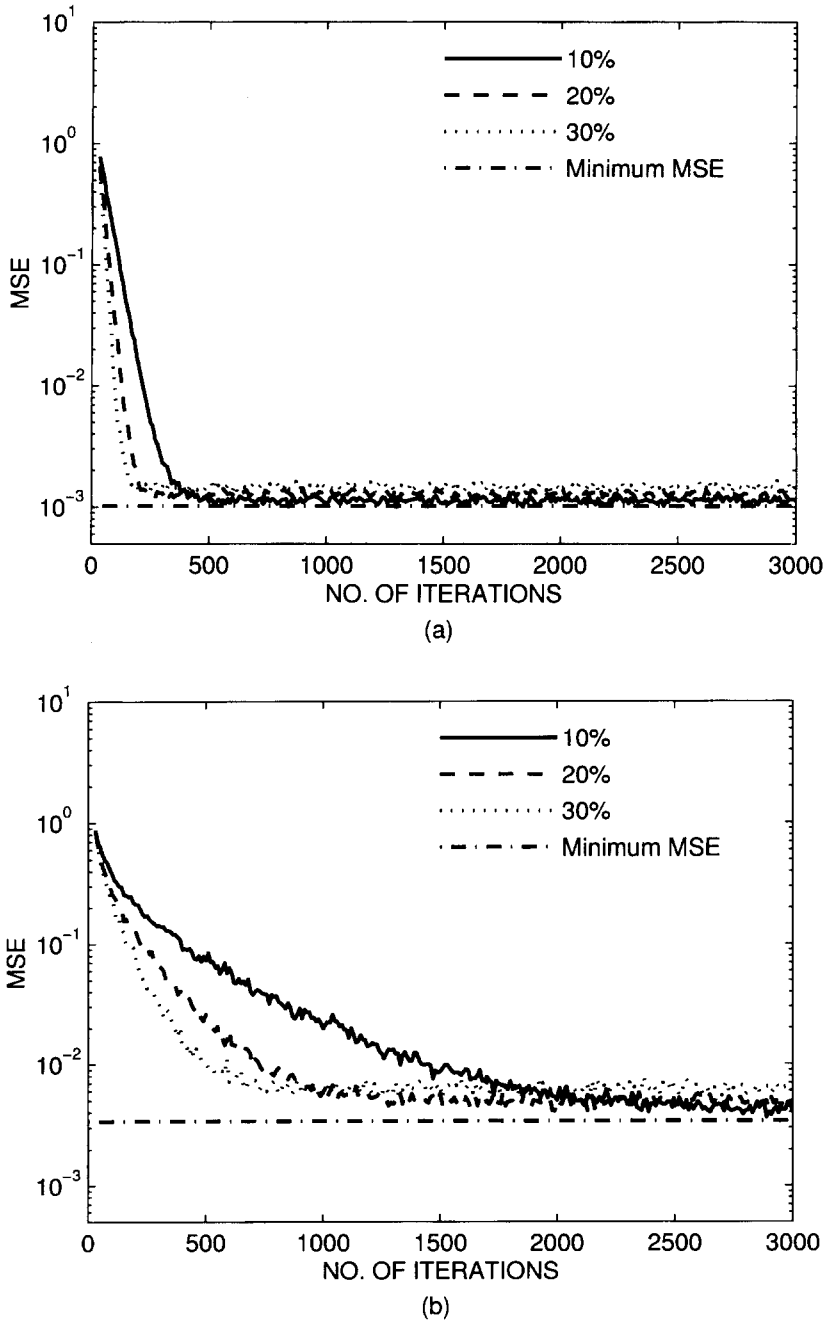


Figure 6.9 Learning curves of the LMS algorithm for the channel equalizer, for the two choices of channel responses discussed in the text: (a) $H(z) = H_1(z)$ and (b) $H(z) = H_2(z)$. The step-size parameter, μ, is selected for the misadjustment values 10%, 20% and 30%, according to the simplified equation (6.63).

results compare very well with the predictions made by theory. We recall that these correspond to the case where the eigenvalue spread of the correlation matrix **R** is small. Some differences between the results of the two cases are observed as the eigenvalue spread of **R** increases. In particular, a comparison of Figures 6.7(b) and 6.9(b) shows

there is that when the equalizer length N is relatively long

$$\lambda_i w_{o,i}'^2 \approx \frac{1}{N}, \quad \text{for } i = 0, 1, \ldots, N-1. \tag{6.85}$$

Substituting this in (6.77) we get, for an N-tap channel equalizer,

$$\xi(n) \approx \xi_{\min} + \frac{1}{N} \sum_{i=0}^{N-1} (1 - 2\mu\lambda_i)^{2n}. \tag{6.86}$$

The difference between the learning curves of the modelling and channel equalization problems may now be explained by comparing (6.83) and (6.86). When the eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$ are widely spread and n is small (i.e. the adaptation has just started), the summation on the right-hand side of (6.83) is predominantly determined by the larger λ_is. However, noting that the geometrical regressor factors, the $(1 - 2\mu\lambda_i)^{2n}$s, corresponding to the larger λ_is, converge to zero at a relatively fast rate, the summation on the right-hand side of (6.83) experiences a fast drop to a level significantly below its initial value when $n = 0$. The slower modes of the LMS algorithm are observed after this initial fast drop of the MSE. This, of course, is what we observe in Figure 6.7(b). In the case of channel equalizer, we note that when n is small all the terms under the summation on the right-hand side of (6.86) are about the same. This means that there is no dominant term in the latter summation and, as a result, unlike the modelling problem case, the convergence of the faster modes of the LMS algorithm may not reduce $\xi(n)$ significantly. A significant reduction in $\xi(n)$ after convergence of the faster modes of the LMS algorithm may only be observed when the filter length, N, is large and only a few of the eigenvalues of $\mathbf{R}$ are small.

6.4.3 Adaptive line enhancement

Adaptive line enhancement refers to the case where a noisy signal consisting of a few sinusoidal components is available and the aim is to filter out the noise part of the signal. The filtering solution to this problem is trivial. The noisy signal is passed through a filter which is tuned to the sinusoidal components. When the frequency of the sine waves present in the noisy signal are known, of course, a fixed filter will suffice. However, when the sine-wave frequencies are unknown or may be time-varying, an adaptive solution has to be adopted.

Figure 6.10 depicts the block schematic of an adaptive line enhancer. It is basically an M-step-ahead predictor. The assumption is that the noise samples which are more than M samples apart are uncorrelated with one another. As a result, the predictor can only make a prediction about the sinusoidal components of the input signal, and when adapted to minimize the output MSE, the line enhancer will be a filter tuned to the sinusoidal components. The maximum possible rejection of the noise will also be achieved, since any portion of the noise that passes through the prediction filter will enhance the output MSE whose minimization is the criterion in adapting the filter tap weights.

Here, to simplify our discussion, we assume that the enhancer input consists of a single sinusoidal component and the additive noise is white. More specifically, we

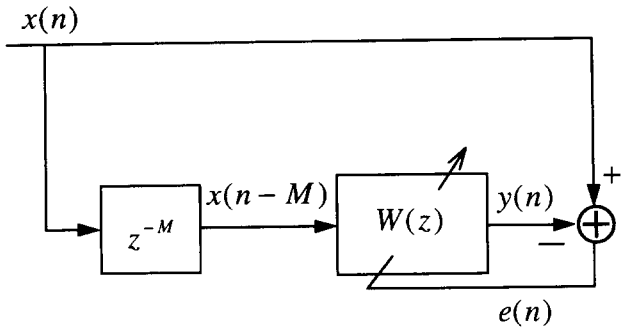
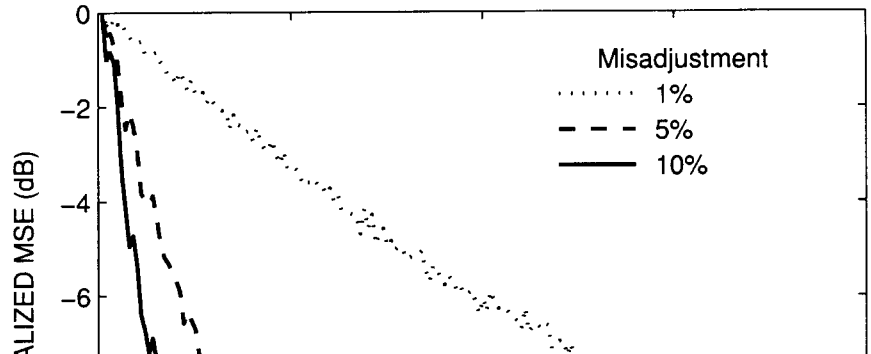


Figure 6.10 Adaptive line enhancer

where $\nu(n)$ is a white noise sequence. The delay parameter M is set to 1, since $\nu(n)$ is white.

Figure 6.11 shows the learning curves of the adaptive line enhancer when $x(n)$ is chosen as in (6.87). The following parameters are used to obtain these results: $N = 30$, $M = 1$, $a = 1$, $\omega_o = 0.1$, and θ is chosen to be a random variable with constant



distribution in the range of 0 to 2π, for different simulation runs. The variance of $\nu(n)$ is chosen as 10 dB below the sinusoidal signal energy. The learning curves are given for three choices of the step-size parameter, μ, which result in 1%, 5% and 10% misadjustment. The predictor tap weights are initialized to zero. The program used to obtain these results is available on the accompanying diskette. It is called 'lenhncr.m'.

From the results presented in Figure 6.11, it appears that the convergence of the line enhancer is governed by only one mode. Examination of the eigenvalues of the underlying process and the resulting time constants of the various modes of the line enhancer reveals that the mode that is observed in Figure 6.11 coincides with the fastest convergence mode of the LMS algorithm in the present case. An explanation of this phenomenon is instructive.

We note that the optimized predictor of the line enhancer is a filter tuned to the peak of the spectrum of $x(n)$. Furthermore, from the minimax theorem (of Chapter 4) we may say that the latter is the eigenfilter associated with the maximum eigenvalue of the correlation matrix $\mathbf{R}$ of the underlying process. This implies that the optimum tap-weight vector of the line enhancer coincides with the eigenvector associated with the largest eigenvalue of its corresponding correlation matrix. In other words, in the Euclidian space associated with the tap weights of the line enhancer, the line connecting the origin to the point defined by the optimized tap weights is along the eigenvector associated with largest eigenvalue of its corresponding correlation matrix. This clearly explains why the learning curves of the line enhancer presented in Figure 6.11 are predominantly controlled by only one mode and this coincides with the fastest mode of convergence of the corresponding LMS algorithm.

6.4.4 Beamforming

Consider a two-element antenna array similar to the one discussed in Example 3.6. The array consists of two omni-directional (equally sensitive to all directions) antennas A and B, as in Figure 6.12. The desired signal $s(n) = \alpha(n)\cos(n\omega_0 + \phi_1)$ arrives in the direction perpendicular to the line connecting A and B. An interferer (jammer) signal $\nu(n) = \beta(n)\cos(n\omega_0 + \phi_2)$ arrives at an angle θ_0 relative to $s(n)$. The signal sequences

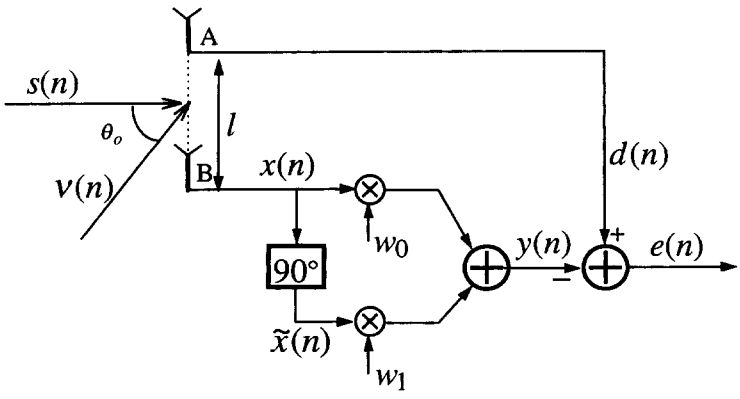


Figure 6.12 A two-element antenna array

$s(n)$ and $\nu(n)$ are assumed to be narrow-band processes with random phases ϕ_1 and ϕ_2, respectively. It is also assumed that the random amplitudes $\alpha(n)$ and $\beta(n)$ are zero-mean and uncorrelated with each other. The two omnis are separated by a distance of $l = \lambda_c/2$ metres, where λ_c is the wavelength associated with the continuous time carrier frequency

$$\omega_c = \frac{\omega_o}{T}, \tag{6.88}$$

with T being the sampling period. The coefficients, w_0 and w_1, of the beamformer are adjusted so that the output error, $e(n)$, is minimized in the mean-square sense.

As in Example 3.6, the adaptive beamformer of Figure 6.12 is characterized by the following signal sequences:[5]

1. Primary input

$$d(n) = \alpha(n)\cos(n\omega_o + \phi_1) + \beta(n)\cos(n\omega_o + \phi_2 - \phi_o). \tag{6.89}$$

2. Reference tap-input vector

$$\begin{aligned}\mathbf{x}(n) &= \begin{bmatrix} x(n) \\ \tilde{x}(n) \end{bmatrix} \\ &= \begin{bmatrix} \alpha(n)\cos(n\omega_o + \phi_1) + \beta(n)\cos(n\omega_o + \phi_2) \\ \alpha(n)\sin(n\omega_o + \phi_1) + \beta(n)\sin(n\omega_o + \phi_2) \end{bmatrix}.\end{aligned} \tag{6.90}$$

The phase shift ϕ_o is introduced because of the difference between the arrival time of the jammer at A and B. It is given by

$$\phi_o = \frac{l\sin\theta_o}{c}\omega_c, \tag{6.91}$$

where c is the propagation speed. Replacing l with $\lambda_c/2$ in (6.91) and noting that $\omega_c/c = 2\pi/\lambda_c$, we obtain

$$\phi_o = \pi\sin\theta_o. \tag{6.92}$$

We note that, as expected, ϕ_o is independent of the sampling period T. It depends only on the angle of arrival of the jammer signal, θ_o.

The beamformer coefficients, w_0 and w_1, are selected (adapted) so that the difference,

$$e(n) = d(n) - \mathbf{w}^T\mathbf{x}(n),$$

where $\mathbf{w} = [w_0 \;\; w_1]^T$, is minimized in the mean-square sense. The error signal $e(n)$ is the beamformer output.

For a given set of beamformer coefficients w_0 and w_1 and a signal arriving at an angle θ, the array power gain, $\mathcal{G}(\theta)$, is defined as the ratio of the signal power in the output $e(n)$ to the signal power at one of the omnis. Assuming that a narrow-band signal

[5] In Example 3.6, to simplify the derivations ϕ_1 and ϕ_2 were assumed to be zero.

$\gamma(n)\cos n\omega_o$ is arriving at an angle θ,

$$\begin{aligned} e(n) &= \gamma(n)[\cos(n\omega_o - \pi\sin\theta) - w_0\cos n\omega_o - w_1\sin n\omega_o] \\ &= \gamma(n)[(\cos(\pi\sin\theta) - w_0)\cos n\omega_o + (\sin(\pi\sin\theta) - w_1)\sin n\omega_0] \\ &= a(\theta)\gamma(n)\sin(n\omega_o + \varphi(\theta)), \end{aligned} \tag{6.93}$$

where

$$a(\theta) = \sqrt{(\cos(\pi\sin\theta) - w_0)^2 + (\sin(\pi\sin\theta) - w_1)^2}$$

and

$$\varphi(\theta) = \tan^{-1}\left(\frac{\cos(\pi\sin\theta) - w_0}{\sin(\pi\sin\theta) - w_1}\right).$$

Using these, we get

$$\mathcal{G}(\theta) = a^2(\theta) = (\cos(\pi\sin\theta) - w_0)^2 + (\sin(\pi\sin\theta) - w_1)^2. \tag{6.94}$$

$\mathcal{G}(\theta)$, when plotted against the angle of arrival of the received signal, is called the *directivity pattern* of the array (beamformer). The names *beam pattern*, *array pattern* and *spatial response* are also used to refer to $\mathcal{G}(\theta)$. The directivity patterns are usually plotted in polar coordinates.

Figure 6.13 shows the directivity pattern of the two-element beamformer of Figure 6.12 when its coefficients have been adjusted near their optimal values using

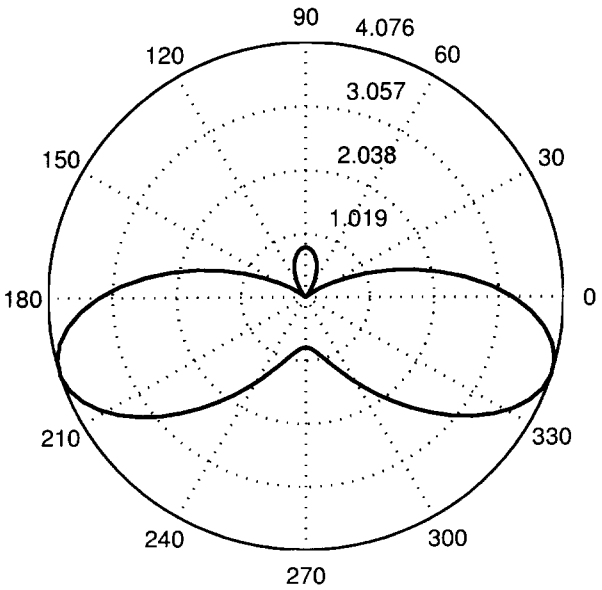


Figure 6.13 The directivity pattern of the two element antenna array when a jammer arrives from the direction 45° with respect to the desired signal, as defined in Figure 6.12

the LMS algorithm. The following parameters have been used to obtain these results:

$$\theta_0 = 45^\circ, \quad \sigma_\alpha^2 = 0.01, \quad \sigma_\beta^2 = 1,$$

where σ_α^2 and σ_β^2 are the variances of $\alpha(n)$ and $\beta(n)$, respectively. The results, as could be predicted from the theory, show a clear deep null in the direction from which the jammer arrives ($\theta = \theta_0$) and a reasonably good gain in the direction of the desired signal ($\theta = 0$). The array pattern is symmetrical with respect to the line connecting A to B because of the omni-directional properties of the antennas. The MATLAB program used to obtain this result is available on the accompanying diskette. It is called 'bformer.m'. We encourage the reader to try this program for different values of θ_0, σ_α^2 and σ_β^2. An interesting observation that can be made is that a null is always produced in the direction of arrival of the desired signal or jammer, whichever is stronger. The theoretical results related to these observations can be found in Chapter 3, Section 3.6.5.

6.5 Simplified LMS Algorithms

Over the years a number of modifications which simplify the hardware implementation of the LMS algorithm have been proposed (Hirsch and Wolf, 1970; Claasen and Mecklenbräuker, 1981; and Duttweiler, 1982). These simplifications are discussed in this section. The most important members of this class of algorithms are:

The Sign Algorithm This algorithm is obtained from the conventional LMS recursion (6.9) by replacing $e(n)$ with its sign. This leads to the following recursion:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu \operatorname{sign}(e(n))\mathbf{x}(n). \tag{6.95}$$

Because of the replacement of $e(n)$ by its sign, implementation of this recursion may be cheaper than the conventional LMS recursion, especially in high speed applications where a hardware implementation of the adaptation recursion may be necessary. Furthermore, the step-size parameter is usually selected to be a power-of-two so that no multiplication would be required for implementing the recursion (6.95). A set of shift and add/subtract operations would suffice to update the filter tap weights.

The Signed-Regressor Algorithm The signed-regressor algorithm is obtained from the conventional LMS recursion (6.9) by replacing the tap-input vector $\mathbf{x}(n)$ with the vector $\operatorname{sign}(\mathbf{x}(n))$, where the sign function is applied to the vector $\mathbf{x}(n)$ on an element-by-element basis. The signed-regressor recursion is then

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\operatorname{sign}(\mathbf{x}(n)). \tag{6.96}$$

Although quite similar in form, the signed-regressor algorithm performs much better than the sign algorithm. This will be shown later through a simulation example.

The Sign–Sign Algorithm The sign–sign algorithm, as may be understood from its name, combines the sign and signed-regressor recursions, resulting in the following recursion:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu \operatorname{sign}(e(n)) \operatorname{sign}(\mathbf{x}(n)). \tag{6.97}$$

It may be noted that even though in many practical cases all the above algorithms are likely to converge to the optimum Wiener–Hopf solution, this may not be true in general. For example, the sign–sign algorithm converges toward a set of tap weights that satisfy the equation

$$\mathrm{E}[\mathrm{sign}(e(n)\mathbf{x}(n))] = 0, \tag{6.98}$$

which in general may not be equivalent to the principle of orthogonality

$$\mathrm{E}[e(n)\mathbf{x}(n)] = 0 \tag{6.99}$$

which leads to the Wiener–Hopf equation. For instance, when the elements of the vector $\mathbf{x}(n)$ are zero-mean but have a non-symmetrical distribution around zero, the elements of $e(n)\mathbf{x}(n)$ may also have a non-symmetrical distribution around zero. In that case, it is likely that the solutions to (6.98) and (6.99) lead to two different set of tap weights. Nevertheless, we shall emphasize that in most of the practical applications the scenario that was just mentioned is unlikely to happen. Even if it happens, the solutions obtained from (6.98) and (6.99) are usually about the same.

To compare the performance of the algorithms that were introduced above with the conventional LMS algorithm and among themselves, we run the system modelling problem that was introduced in Section 6.4.1. Figure 6.14 shows the convergence behaviour of the algorithms when the input colouring filter $H(z) = H_1(z)$ is used and

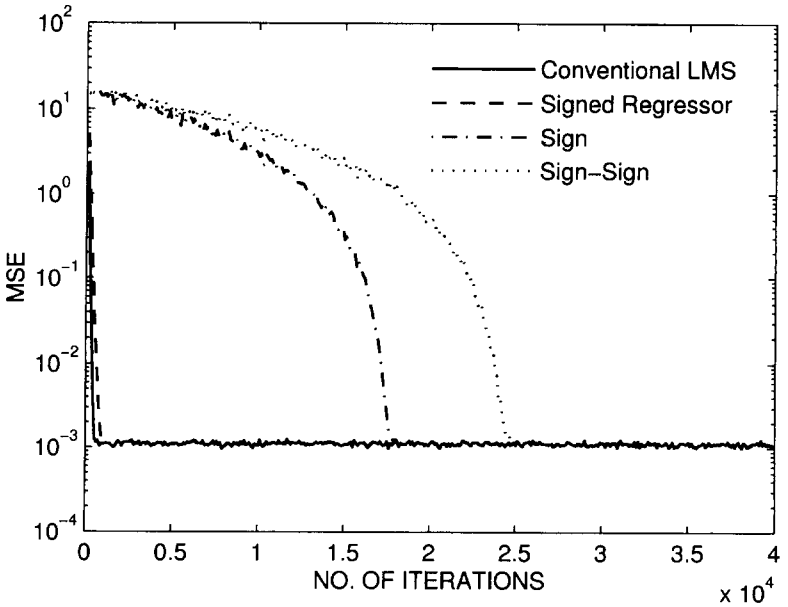


Figure 6.14 Learning curves of the conventional LMS algorithm and its simplified versions. Different step-size parameters are used. These have been selected experimentally so that all algorithms approach the same steady-state MSE

the step-size parameters for different algorithms are selected experimentally so that they all reach the same steady-state MSE.

From the results presented in Figure 6.14, we see that the performance of the signed-regressor algorithm is only slightly worse than the conventional LMS algorithm. However, the sign and sign–sign algorithms are both much slower than the conventional LMS algorithm. Their convergence behaviour is also rather peculiar. They converge very slowly at the beginning, but speed up as the MSE level drops. This can be explained as follows.

Consider the sign algorithm recursion and note that it may be written as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu \frac{e(n)}{|e(n)|}\mathbf{x}(n), \tag{6.100}$$

since $\text{sign}(e(n)) \triangleq e(n)/|e(n)|$. This may be rearranged as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\frac{\mu}{|e(n)|}e(n)\mathbf{x}(n). \tag{6.101}$$

Inspection of (6.101) reveals that the sign algorithm may be thought of as an LMS algorithm with a variable step-size parameter $\mu'(n) = \mu/|e(n)|$. The step-size parameter $\mu'(n)$ increases, on an average, as the sign algorithm converges, since $e(n)$ decreases in magnitude. Thus, to keep the sign algorithm stable, with a small steady-state error, a very small step-size parameter μ has to be used. Choosing a very small μ leads to an equally small value (on average) for $\mu'(n)$ in the initial portion of the sign algorithm. This clearly explains why the sign algorithm initially converges very slowly. However, as the algorithm converges and $e(n)$ becomes smaller in magnitude, the step-size parameter $\mu'(n)$ becomes larger, on average, and this, of course, leads to a faster convergence of the algorithm. A rigorous analysis of the sign algorithm for a non-stationary case can be found in Eweda (1990b).

The same procedure may be followed to explain the behaviour of the signed-regressor algorithm. In this case, each tap of the filter is controlled by a separate variable step-size parameter. In particular, the step-size parameter of the ith tap of the filter at the nth iteration is $\mu_i'(n) = \mu/|x(n-i)|$, where μ is a common parameter to all taps. The fundamental difference between the variable step-size parameters, the $\mu_i'(n)$s, here and what was observed above for the sign algorithm is that in the present case the variations in the $\mu_i'(n)$s are independent of the filter convergence. The selection of the common parameter μ is based on the average size of $|x(n)|$. This leads to a more homogeneous convergence of the signed-regressor algorithm when compared with the sign algorithm. In fact, the analysis of the signed-regressor algorithm given by Eweda (1990a) shows that for Gaussian signals the convergence behaviour of the signed-regressor algorithm is very similar to the conventional LMS algorithm. The replacement of the $x(n-i)$ terms by their signs leads to an increase in the time constants of the algorithm learning curve by a fixed factor of $\pi/2$. This, clearly, increases the convergence time of the signed-regressor algorithm by the same factor when it is compared with the conventional LMS algorithm. Problem P6.13 contains the necessary theoretical elements which lead to this result.

Another interesting proposal which also leads to some simplification of the LMS algorithm was suggested by Duttweiler (1982). He suggested that in calculating the

gradient vector $e(n)\mathbf{x}(n)$, $e(n)$ and/or $\mathbf{x}(n)$ may be quantized to their respective nearest power-of-two. This leads to an algorithm that performs very similar to the conventional LMS algorithm.

6.6 Normalized LMS Algorithm

The normalized LMS (NLMS) algorithm may be viewed as a special implementation of the LMS algorithm which takes into account the variation in the signal level at the filter input and selects a normalized step-size parameter which results in a stable as well as fast converging adaptation algorithm. The NLMS algorithm may be developed from different viewpoints. Goodwin and Sin (1984) formulated the NLMS algorithm as a constrained optimization problem; see also Haykin (1991). Nitzberg (1985) obtained the NLMS recursion by running the conventional LMS algorithm many times, for every new sample of the input. Here, we start with a rather straightforward derivation of the NLMS recursion and later show that the recursion obtained satisfies the constrained optimization criterion of Goodwin and Sin and also that it matches the result of Nitzberg.

We consider the LMS recursion

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu(n)e(n)\mathbf{x}(n), \tag{6.102}$$

where the step-size parameter $\mu(n)$ is time-varying. We select $\mu(n)$ so that the *a posteriori* error,

$$e^{+}(n) = d(n) - \mathbf{w}^{\mathrm{T}}(n+1)\mathbf{x}(n), \tag{6.103}$$

is minimized in magnitude. Substituting (6.102) in (3.103) and rearranging, we obtain

$$e^{+}(n) = (1 - 2\mu(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n))e(n). \tag{6.104}$$

Minimizing $\left(e^{+}(n)\right)^2$ with respect to $\mu(n)$ results in the following:

$$\mu(n) = \frac{1}{2\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)}, \tag{6.105}$$

which forces $e^{+}(n)$ to zero. Substituting (6.105) in (6.102) we obtain

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)}e(n)\mathbf{x}(n). \tag{6.106}$$

This is the NLMS recursion. When this is combined with the filtering equation (6.1) and the error estimation equation (6.2) we obtain the NLMS algorithm. There have been a variety of interpretations to the NLMS algorithm. We review some of these below, since it can help in enhancing our understanding of this algorithm.

1. The use of $\mu(n)$ as in (6.105) is appealing, since it selects a step-size parameter proportional to the inverse of the instantaneous signal sample's energy at the adaptive filter input. This matches the misadjustment equation (6.63) which suggests that a

step-size parameter for the LMS algorithm should be selected proportional to the inverse of the average total energy at the filter tap inputs. Note that

$$\text{tr}[\mathbf{R}] = \sum_{i=0}^{N-1} \text{E}[x^2(n-i)] = \text{E}\left[\sum_{i=0}^{N-1} x^2(n-i)\right],$$

and $\sum_{i=0}^{N-1} x^2(n-i)$ is the total instantaneous signal energy at the filter tap inputs.

2. The NLMS recursion (6.106) is equivalent to running the LMS recursion for every new sample of input for many iterations until it converges (Nitzberg, 1985); see Problem P6.14.
3. The NLMS recursion may also be derived by solving the following constrained optimization problem (Goodwin and Sin, 1984):

   Given the tap-input vector $\mathbf{x}(n)$ and the desired output sample $d(n)$, choose the updated tap-weight vector $\mathbf{w}(n+1)$ so as to minimize the squared Euclidian norm of the difference

$$\boldsymbol{\eta}(n) = \mathbf{w}(n+1) - \mathbf{w}(n) \tag{6.107}$$

   subject to the constraint

$$\mathbf{w}^{\text{T}}(n+1)\mathbf{x}(n) = d(n). \tag{6.108}$$

Observe that the solution given by (6.106) satisfies the constraint (6.108). Hence, we define $\boldsymbol{\eta}_{\text{NLMS}}(n)$ as

$$\boldsymbol{\eta}_{\text{NLMS}}(n) = \mathbf{w}(n+1) - \mathbf{w}(n) = \frac{1}{\mathbf{x}^{\text{T}}(n)\mathbf{x}(n)} e(n)\mathbf{x}(n). \tag{6.109}$$

We will now show that $\boldsymbol{\eta}_{\text{NLMS}}(n)$ is indeed the solution to the problem posed above.

Let the optimum $\boldsymbol{\eta}(n)$ be given by

$$\boldsymbol{\eta}_{\text{o}}(n) = \boldsymbol{\eta}_{\text{NLMS}}(n) + \boldsymbol{\eta}_{\text{I}}(n), \tag{6.110}$$

where $\boldsymbol{\eta}_{\text{I}}(n)$ indicates any difference that may exist between $\boldsymbol{\eta}_{\text{o}}(n)$ and $\boldsymbol{\eta}_{\text{NLMS}}(n)$. Since the updated vector $\mathbf{w}(n+1) = \mathbf{w}(n) + \boldsymbol{\eta}_{\text{NLMS}}(n)$ satisfies the constraint (6.108), we get

$$(\mathbf{w}(n) + \boldsymbol{\eta}_{\text{NLMS}}(n))^{\text{T}}\mathbf{x}(n) = d(n). \tag{6.111}$$

The tap-weight vector $\mathbf{w}(n+1) = \mathbf{w}(n) + \boldsymbol{\eta}_{\text{o}}(n)$ also satisfies the constraint (6.108), since $\boldsymbol{\eta}_{\text{o}}(n)$ is the optimum solution. Thus,

$$(\mathbf{w}(n) + \boldsymbol{\eta}_{\text{o}}(n))^{\text{T}}\mathbf{x}(n) = d(n). \tag{6.112}$$

Subtracting (6.111) from (6.112)) and using (6.110), we get

$$\boldsymbol{\eta}_{\text{I}}^{\text{T}}(n)\mathbf{x}(n) = 0. \tag{6.113}$$

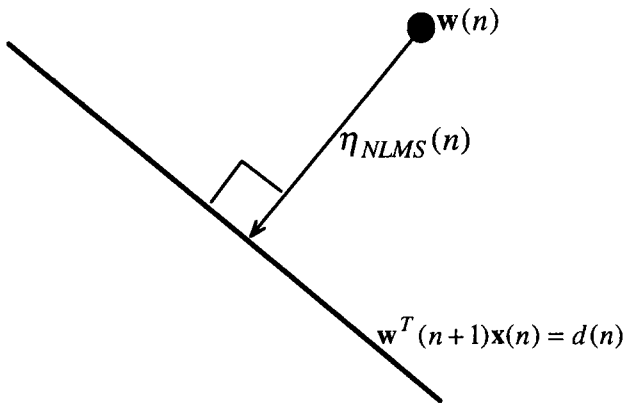


Figure 6.15 Geometrical interpretation of the NLMS recursion

Multiplying the left-hand and right-hand sides of (6.110) by their respective transposes, from left, we obtain

$$\begin{aligned}\boldsymbol{\eta}_{\mathrm{o}}^{\mathrm{T}}(n)\boldsymbol{\eta}_{\mathrm{o}}(n) &= (\boldsymbol{\eta}_{\mathrm{NLMS}}(n) + \boldsymbol{\eta}_{\mathrm{l}}(n))^{\mathrm{T}}(\boldsymbol{\eta}_{\mathrm{NLMS}}(n) + \boldsymbol{\eta}_{\mathrm{l}}(n)) \\ &= \boldsymbol{\eta}_{\mathrm{NLMS}}^{\mathrm{T}}(n)\boldsymbol{\eta}_{\mathrm{NLMS}}(n) + \boldsymbol{\eta}_{\mathrm{l}}^{\mathrm{T}}(n)\boldsymbol{\eta}_{\mathrm{l}}(n) + 2\boldsymbol{\eta}_{\mathrm{NLMS}}^{\mathrm{T}}(n)\boldsymbol{\eta}_{\mathrm{l}}(n). \end{aligned} \tag{6.114}$$

Premultiplying (6.109) by $\boldsymbol{\eta}_{\mathrm{l}}^{\mathrm{T}}(n)$ and using (6.113), we obtain

$$\boldsymbol{\eta}_{\mathrm{l}}^{\mathrm{T}}(n)\boldsymbol{\eta}_{\mathrm{NLMS}}(n) = 0. \tag{6.115}$$

Substituting (6.115) in (6.114) we obtain

$$\boldsymbol{\eta}_{\mathrm{o}}^{\mathrm{T}}(n)\boldsymbol{\eta}_{\mathrm{o}}(n) = \boldsymbol{\eta}_{\mathrm{NLMS}}^{\mathrm{T}}(n)\boldsymbol{\eta}_{\mathrm{NLMS}}(n) + \boldsymbol{\eta}_{\mathrm{l}}^{\mathrm{T}}(n)\boldsymbol{\eta}_{\mathrm{l}}(n). \tag{6.116}$$

This suggests that the squared Euclidian norm of the vector $\boldsymbol{\eta}_{\mathrm{o}}(n)$, i.e. $\boldsymbol{\eta}_{\mathrm{o}}^{\mathrm{T}}(n)\boldsymbol{\eta}_{\mathrm{o}}(n)$, attains its minimum when the squared Euclidian norm of the vector $\boldsymbol{\eta}_{\mathrm{l}}(n)$ is minimum. This, of course, is achieved when $\boldsymbol{\eta}_{\mathrm{l}}(n) = \mathbf{0}$. Thus, we obtain

$$\boldsymbol{\eta}_{\mathrm{o}}(n) = \boldsymbol{\eta}_{\mathrm{NLMS}}(n), \tag{6.117}$$

This completes our proof.[6]

Figure 6.15 gives a geometrical interpretation of the above result. The tap-weight vector $\mathbf{w}(n)$ is represented by a point. The constraint $\mathbf{w}^{\mathrm{T}}(n+1)\mathbf{x}(n) = d(n)$ limits $\mathbf{w}(n+1)$ to the points in a subspace whose dimension is one less than the filter length, N, i.e. $N-1$. This is represented as a line in Figure 6.15. The vector $\boldsymbol{\eta}_{\mathrm{NLMS}}(n)$ is

[6] The above results could also be derived by application of the method of the Lagrange multiplier; see Section 6.10.1 for an example of the use of the Lagrange multiplier. Here we have selected to give a direct derivation of the results from the first principles of vector calculus. This derivation is also instructive, since its application leads to the geometrical interpretation of the NLMS recursion depicted in Figure 6.15.

Table 6.2 Summary of the normalized LMS algorithm

Input:	Tap-weight vector, $\mathbf{w}(n)$, Input vector, $\mathbf{x}(n)$, and desired output, $d(n)$.
Output:	Filter output, $y(n)$, Tap-weight vector update, $\mathbf{w}(n+1)$.

1. Filtering:
$$y(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n)$$
2. Error estimation:
$$e(n) = d(n) - y(n)$$
3. Tap-weight vector adaptation:
$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\tilde{\mu}}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n) + \psi} e(n)\mathbf{x}(n)$$

orthogonal to this subspace. It is also the vector connecting the point associated with $\mathbf{w}(n)$ to its projection on the subspace. This clearly shows that $\boldsymbol{\eta}_{\mathrm{NLMS}}(n)$ is the minimum length vector that results in the updated tap-weight vector $\mathbf{w}(n+1) = \mathbf{w}(n) + \boldsymbol{\eta}_{\mathrm{NLMS}}(n)$ subject to the constraint (6.108).

Despite its appealing interpretations, the NLMS recursion (6.106) is seldom used in actual applications. Instead, it is often observed that the following relaxed recursion results in a more reliable implementation of adaptive filters:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\tilde{\mu}}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n) + \psi} e(n)\mathbf{x}(n). \tag{6.118}$$

In this recursion $\tilde{\mu}$ and ψ are positive constants which should be selected appropriately. The rationale for the introduction of the constant ψ is to prevent division by a small value when the squared Euclidian norm, $\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)$, is small. This results in a more stable implementation of the NLMS algorithm. The constant $\tilde{\mu}$ may be thought of as a step-size parameter which controls the rate of convergence of the algorithm and also its misadjustment. We also note that the recursion (6.118) reduces to (6.106) when $\tilde{\mu} = 1$ and $\psi = 0$. Table 6.2 summarizes the NLMS algorithm.

6.7 Variable Step-Size LMS Algorithm

The analysis presented in Section 6.3 shows that the step-size parameter, μ, plays a significant role in controlling the performance of the LMS algorithm. On the one hand, the speed of convergence of the LMS algorithm changes in proportion to its step-size parameter. As a result, a large step-size parameter may be required to minimize the transient time of the LMS algorithm. On the other hand, to achieve a small misadjustment a small step-size parameter has to be used. These are conflicting requirements and, thus, a compromise solution has to be adopted. The variable step-size LMS (VSLMS) algorithm which is introduced in this section is an effective solution to this problem (Harris, Chabries and Bishop, 1986).

The VSLMS algorithm works on the basis of a simple heuristic that comes from the mechanism of the LMS algorithm. Each tap of the adaptive filter is given a separate time-varying step-size parameter and the LMS recursion is written as

$$w_i(n+1) = w_i(n) + 2\mu_i(n)e(n)x(n-i), \quad \text{for } i = 0, 1, \ldots, N-1, \tag{6.119}$$

where $w_i(n)$ is the ith element of the tap-weight vector $\mathbf{w}(n)$ and $\mu_i(n)$ is its associated step-size parameter at iteration n. The adjustment of the step-size parameter $\mu_i(n)$ is done as follows. The corresponding stochastic gradient term $g_i(n) = e(n)x(n-i)$ is monitored over successive iterations of the algorithm and $\mu_i(n)$ is increased if the latter term consistently shows a positive or negative direction. This happens when the adaptive filter has not yet converged. As the adaptive filter tap weights converge to some vicinity of their optimum values, the averages of the stochastic gradient terms approach zero and hence they change signs more frequently. This is detected by the algorithm and the corresponding step-size parameters are gradually reduced to some minimum values. If the situation changes and the algorithm begins to hunt for a new optimum point, then the gradient terms will indicate consistent (positive or negative) directions, resulting in an increase in the corresponding step-size parameters. To ensure that the step-size parameters do not become too large (which may result in system instability) or too small (which may result in a slow reaction of the system to sudden changes), upper and lower limits should be specified for each step-size parameter.

Following the above argument, the VSLMS algorithm step-size parameters, the $\mu_i(n)$s, may be adjusted using the following recursion:

$$\mu_i(n) = \mu_i(n-1) + \rho \operatorname{sign}[g_i(n)] \operatorname{sign}[g_i(n-1)] \tag{6.120}$$

where ρ is a small positive step-size parameter. The 'sign' functions may be dropped from (6.120). This results in the following alternative step-size parameter update equation:

$$\mu_i(n) = \mu_i(n-1) + \rho g_i(n) g_i(n-1). \tag{6.121}$$

Both update equations (6.120) and (6.121) work well in practice. Which of the two choices works better is application dependent. The choice of one over the other may also be decided on the basis of the available hardware/software platform on which the algorithm is to be implemented. For instance, if a digital signal processor is being used, then recursion (6.121) may be much easier to implement. On the other hand, if a custom chip is to be designed, then the update equation (6.120) may be preferred.

The derivation of an inequality similar to (6.73) to determine the range of the step-size parameters that ensure the stability of the VSLMS algorithm is rather difficult, because of the time-variation of the step-size parameters. Here, we adopt a simple approach by assuming that the step-size parameters vary slowly so that for the stability analysis they may be assumed fixed and use the analogy between the resulting VSLMS algorithm equations and the conventional LMS algorithm to arrive at a result which, through computer simulations, has been found to be reasonable. Further results on the VSLMS algorithm misadjustment and its tracking behaviour, along with computer simulation results, can be found in Chapter 14.

The set of update equations (6.119) may be written in vector form as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\boldsymbol{\mu}(n)e(n)\mathbf{x}(n), \tag{6.122}$$

Table 6.3 Summary of an implementation of variable step-size LMS algorithm

Input: Tap-weight vector, $\mathbf{w}(n)$,
input vector, $\mathbf{x}(n)$,
Gradient terms $g_0(n-1), g_1(n-1), \ldots, g_{N-1}(n-1)$,
Step-size parameters, $\mu_0(n-1), \mu_1(n-1), \ldots, \mu_{N-1}(n-1)$,
and desired output, $d(n)$.

Output: Filter output, $y(n)$,
tap-weight vector update, $\mathbf{w}(n+1)$,
Gradient terms $g_0(n), g_1(n), \ldots, g_{N-1}(n)$,
and updated step-size parameters $\mu_0(n), \mu_1(n), \ldots, \mu_{N-1}(n)$.

1. Filtering:
   $y(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n)$
2. Error estimation:
   $e(n) = d(n) - y(n)$
3. Tap weights and step-size parameters adaptation:
   For $i = 0, 1, \ldots, N-1$
   $g_i(n) = e(n)x(n-i)$
   $\mu_i(n) = \mu_i(n-1) + \rho \text{sign}[g_i(n)]\text{sign}[g_i(n-1)]$
   if $\mu_i(n) > \mu_{\max}$, $\mu_i(n) = \mu_{\max}$
   if $\mu_i(n) < \mu_{\min}$, $\mu_i(n) = \mu_{\min}$
   $w_i(n+1) = w_i(n) + 2\mu_i(n)g_i(n)$
   end

where $\boldsymbol{\mu}(n)$ is a diagonal matrix consisting of the step-size parameters $\mu_0(n), \mu_1(n), \ldots, \mu_{N-1}(n)$. Equation (6.122) may further be rearranged as

$$\mathbf{v}(n+1) = (\mathbf{I} - 2\boldsymbol{\mu}(n)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n))\mathbf{v}(n) + 2\boldsymbol{\mu}(n)e_{\mathrm{o}}(n)\mathbf{x}(n), \tag{6.123}$$

where notations follow those of Section 6.2. Comparing (6.123) with (6.13) and the subsequent discussions on the stability of the conventional LMS algorithm in Section 6.3, we may argue that to ensure the stability of the VSLMS algorithm, the scalar step-size parameter μ in (6.73) should be replaced by the diagonal matrix $\boldsymbol{\mu}(n)$. This leads to the inequality[7]

$$\text{tr}[\boldsymbol{\mu}(n)\mathbf{R}] < \tfrac{1}{3} \tag{6.124}$$

as a sufficient condition which assures the stability of the VSLMS algorithm. Although the inequality (6.124) may be used to impose some dynamic bounds on the step-size parameters $\mu_i(n)$ as the adaptation of the filter proceeds, this leads to a rather complicated process. Instead, in practice we usually prefer to use (6.73) to limit all $\mu_i(n)$s to the same maximum value, say $\mu_{\max}$.

The minimum bound that may be imposed on the variable step-size parameters, the $\mu_i(n)$s, can be as low as zero. However, in actual practice a positive bound is usually used

[7] See Chapter 14 for a formal derivation of (6.124).

Table 6.4 Summary of the complex LMS algorithm

Input:	Tap-weight vector, $\mathbf{w}(n)$, Input vector, $\mathbf{x}(n)$, and desired output, $d(n)$.
Output:	Filter output, $y(n)$, Tap-weight vector update, $\mathbf{w}(n+1)$.

1. Filtering:
   $y(n) = \mathbf{w}^{\mathrm{H}}(n)\mathbf{x}(n)$
2. Error estimation:
   $e(n) = d(n) - y(n)$
3. Tap-weight vector adaptation:
   $\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e^{*}(n)\mathbf{x}(n)$

so that the adaptation process will be on all the time and possible variations in the adaptive filter optimum tap weights can always be tracked. Here, we use the notation $\mu_{\min}$ to refer to this lower bound. Table 6.3 summarizes an implementation of the VSLMS algorithm.

6.8 LMS Algorithm for Complex-Valued Signals

In applications such as data transmission with quadrature amplitude modulation (QAM) signalling, and beamforming with baseband processing of signals, the underlying data signals and filter coefficients are complex-valued. To modify the LMS recursion for such applications, we use the definition of the gradient of real-valued functions of complex-valued variables as was defined in Section 3.5. We consider an adaptive filter with a complex-valued tap-input vector $\mathbf{x}(n)$, a tap-weight vector $\mathbf{w}(n) = [w_0^*(n) \;\; w_1^*(n) \;\; \ldots \;\; w_{N-1}^*(n)]^{\mathrm{T}}$, output $y(n) = \mathbf{w}^{\mathrm{H}}(n)\mathbf{x}(n)$, and desired output $d(n)$.

The LMS algorithm in this case works on the basis of the update equation

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\nabla_{\mathbf{w}}^{\mathrm{C}}|e(n)|^2, \tag{6.125}$$

where $\nabla_{\mathbf{w}}^{\mathrm{C}}$ denotes complex gradient operator with respect to the variable vector $\mathbf{w}$. This

where w_R and w_I are the real and imaginary parts of w, respectively, and $j = \sqrt{-1}$. We also note that in (6.126) the elements of the gradient vector $\nabla_{\mathbf{w}}^{C}$ are complex gradients with respect to the elements of $\mathbf{w}$ and these elements are the conjugates of the actual tap weights, i.e. $w_0^*, w_1^*, \ldots, w_{N-1}^*$. Furthermore, we note that a direct substitution in (6.127) gives

$$\nabla_{w_i^*}^{C} = \frac{\partial}{\partial w_{i,R}} - j\frac{\partial}{\partial w_{i,I}}. \tag{6.128}$$

Replacing $|e(n)|^2$ by $e(n)e^*(n)$, using (6.128), and following a derivation similar to the one that led to equation (3.63), we obtain

$$\nabla_{w_i^*}^{C}|e(n)|^2 = -2e^*(n)x(n-i), \quad \text{for } i = 0, 1, \ldots, N-1, \tag{6.129}$$

where the asterisk denotes complex conjugation. Substituting (6.129) and the definition (6.126) in (6.125), we obtain

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e^*(n)\mathbf{x}(n). \tag{6.130}$$

This is the desired LMS recursion for the case where the underlying processes are complex-valued. Table 6.4 summarizes implementation of the LMS algorithm for complex-valued signals.

The convergence properties of the LMS algorithm for complex-valued signals are very similar to those of the real-valued signals. These properties are summarized below for reference:

- The time constant equation (6.33) is also applicable to adaptive filters with complex-valued signals.
- The misadjustment equation (6.60) has to be modified slightly. This modification is the result of the fact that for complex-valued jointly Gaussian random variables the equality (6A-6) has to be replaced by

$$E[x_1x_2^*x_3x_4^*] = E[x_1x_2^*]E[x_3x_4^*] + E[x_1x_4^*]E[x_2^*x_3]. \tag{6.131}$$

Taking note of this and following a similar derivation as in Section 6.3, we obtain[8]

$$\mathcal{M} = \frac{\displaystyle\sum_{i=0}^{N-1} \frac{\mu\lambda_i}{1-\mu\lambda_i}}{1 - \displaystyle\sum_{i=0}^{N-1} \frac{\mu\lambda_i}{1-\mu\lambda_i}}. \tag{6.132}$$

- When the step-size parameter, μ, is small, so that $\mu\lambda_i \ll 1$, for $i = 0, 1, \ldots, N-1$, (6.132) reduces to (6.63). Thus, the approximation (6.63) is also applicable to the case where the underlying signals are complex-valued.

[8] A detailed derivation of this result can be found in Haykin (1991).

- Using (6.132) and following the same arguments as in Section 6.3.4, we find that in the case of complex-valued signals, the LMS algorithm remains stable when

$$0 < \mu < \frac{1}{2\,\mathrm{tr}[\mathbf{R}]}. \tag{6.133}$$

Comparing this result with (6.73), we find that in the case of complex-valued signals, the upper bound of μ is more relaxed when compared with the corresponding bound for real-valued signals.

6.9 Beamforming (Revisited)

The beamforming structure presented in Example 3.6, as well as earlier in this chapter, works with signals at their associated radio frequency (RF) or an intermediate frequency (IF). We also recall that the carrier phase plays a major role in implementing a desired beam pattern. To extract and use the carrier phase angles of the signals picked up by the array elements, it was previously proposed that modulated carrier signals and their associated 90° phase-shifted version be processed simultaneously. The amplitude and phase angle of an amplitude modulated signal, such as $x(t) = a(t)\cos(\omega_c t + \phi)$ (where t

In the implementation of beamformers, working with phasor signals is more convenient than RF (or IF) signals. In particular, from an implementation point of view, the digital processing of RF (or IF) signals requires a very high sampling rate to prevent aliasing and to allow any post-processing of the sampled signals, while the required sampling (Nyquist) rate for equivalent baseband signals is much lower.

Example 6.3

In this example we discuss the implementation of the beamformer of Figure 6.12 at the baseband using phasor signals. Figure 6.17 shows an equivalent implementation of Figure 6.12 when all signals are converted to their equivalent phasors. This implementation, as shown, involves an adaptive filter with only one complex tap weight, w, whose optimum value is obtained by minimizing

$$\xi = \mathrm{E}\left[|\underline{d}(n) - w\underline{x}(n)|^2\right]. \tag{6.134}$$

Using definition (6.127) to obtain the gradient of ξ with respect to w and setting the result equal to zero, we obtain

$$w_{\mathrm{o}} = \frac{\mathrm{E}[\underline{d}(n)\underline{x}^*(n)]}{\mathrm{E}[|\underline{x}(n)|^2]}, \tag{6.135}$$

where w_{o} is the optimum value of w.

Converting $d(n)$ of (6.89) to its equivalent phasor, we get

$$\underline{d}(n) = \alpha(n)\mathrm{e}^{j\phi_1} + \beta(n)\mathrm{e}^{j(\phi_2 - \phi_{\mathrm{o}})}. \tag{6.136}$$

Similarly,

$$\underline{x}(n) = \alpha(n)\mathrm{e}^{j\phi_1} + \beta(n)\mathrm{e}^{j\phi_2}. \tag{6.137}$$

Substituting (6.136) and (6.137) in (6.135) and recalling that $\alpha(n)$ and $\beta(n)$ are zero-mean, real-valued and uncorrelated random variables, we obtain

$$w_{\mathrm{o}} = \frac{\sigma_\alpha^2 + \sigma_\beta^2 \mathrm{e}^{-j\phi_{\mathrm{o}}}}{\sigma_\alpha^2 + \sigma_\beta^2}. \tag{6.138}$$

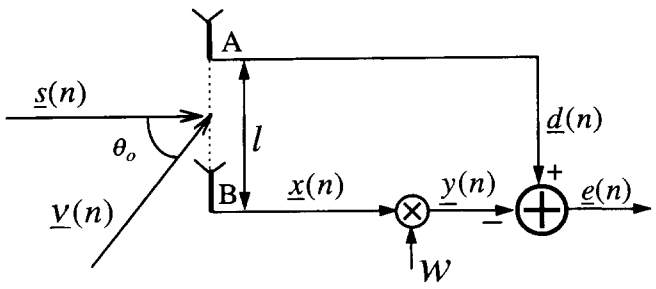


Figure 6.17 Baseband implementation of a two-element beamformer

With this value of w_o, the array power gain, $\mathcal{G}(\theta)$, for a narrow-band signal arriving at an angle θ, is obtained as follows.

Assuming that the signal arriving at the angle θ is $\gamma(n)\cos\omega_c nT$ and using (6.92), with θ_o replaced by θ, we get $\underline{d}(n) = \gamma(n)e^{-j\pi\sin\theta}$. We also note that $\underline{x}(n) = \gamma(n)$. Thus,

$$\underline{e}(n) = \gamma(n)e^{-j\pi\sin\theta} - w_o\gamma(n) = \gamma(n)(e^{-j\pi\sin\theta} - w_o)$$

and

$$\mathcal{G}(\theta) = \frac{E[|\underline{e}(n)|^2]}{E[|\underline{x}(n)|^2]} = |e^{-j\pi\sin\theta} - w_o|^2. \tag{6.139}$$

Careful examination of (6.94) and (6.139) reveals that, as might be expected, both implementations of the beamformer (i.e. Figures 6.12 and 6.17) result in the same optimized power gain. The beamformer tap weights w_0 and w_1 in (6.94) correspond to the real and negative of the imaginary parts, respectively, of the complex tap weight w_o in (6.139). This, in turn, confirms that the two implementations are equivalent.

To adjust w adaptively we may use the complex LMS algorithm of Table 6.4, with the following substitutions:

$$\mathbf{x}(n) = \underline{x}(n), \quad d(n) = \underline{d}(n), \quad e(n) = \underline{e}(n),$$

and

$$\mathbf{w}(n) = w^*(n).$$

If we run the resulting algorithm for a sufficient number of iterations and then use the converged tap weight in (6.139), we will obtain the same directivity pattern as the one presented in Figure 6.13, since the two implementations are equivalent.

So far we have introduced beamformers that are limited to only two antennas. Such beamformers are capable of canceling only one jammer. The use of more elements, as shown in Figure 6.18, allows cancellation of more than one jammer. In general, to cancel M jammers, we require at least $M + 1$ antennas. We may also recall that the implementation proposed in Example 6.3 and also those that were discussed before do not differentiate between the jammer(s) and the desired signal. They simply adapt so that the stronger signal(s) is (are) cancelled, leaving behind the weaker signal(s). In cases where no jammer is present or the desired signal is strong, the latter is deleted by the beamformer. This problem can be prevented by using an amended version of the LMS algorithm which imposes a linear constraint on the tap weight of the adaptive filter. This, which is known as linearly constrained LMS algorithm, is introduced in the next section. To be able to apply the latter algorithm, the beamformer structure has to be modified as in Figure 6.19, where $M + 1$ antennas are used for cancellation of up to M jammers arriving from different directions.

The fundamental difference between the two structures in Figure 6.18 and 6.19 is that in the latter there is no primary input. The tap weights of the beamformer of Figure 6.19 are optimized so that its output, $y(n)$, is minimized in the mean-square sense. To prevent the trivial solution of $w_i = 0$, for all i, a linear constraint that ensures a non-zero gain in the desired direction is imposed on the beamformer tap weights prior to their optimization. The discussions provided in the next section and, especially, Example 6.4 will clarify this concept.

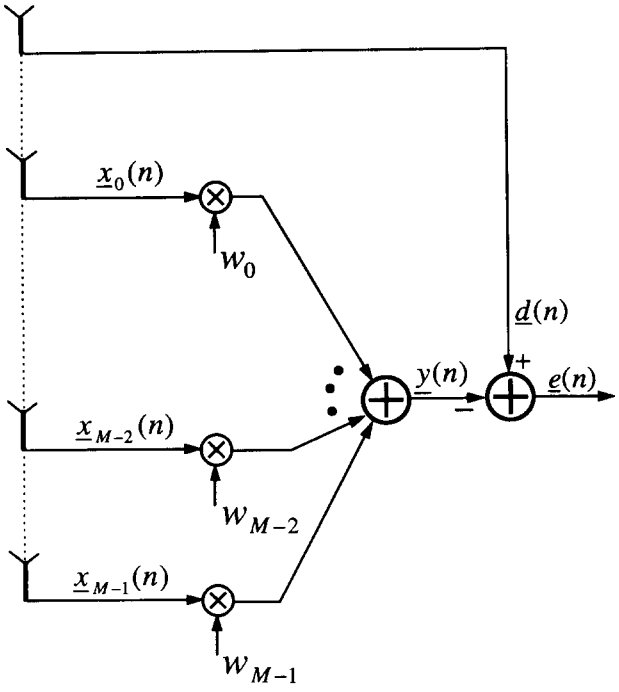


Figure 6.18 Baseband implementation of an $(M + 1)$-element beamformer

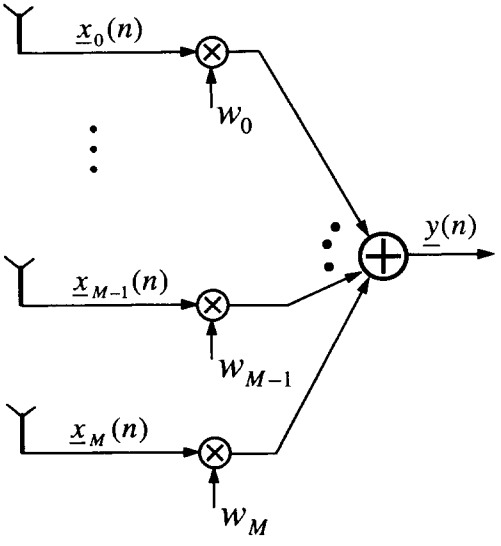


Figure 6.19 Alternative implementation of the $(M + 1)$-element baseband beamformer

We may also recall that the beamformer structures depicted in Figures 6.18 and 6.19 assume that the signals picked up by array elements (antennas) are narrow-band. When the underlying signals are wide-band, the output of each element has to go through a transversal filter, so that there would be some control over different frequency bins (Widrow and Stearns, 1985; and Johnson and Dudgeon, 1993).

6.10 Linearly Constrained LMS Algorithm

In this section we discuss the problem of Wiener filtering with a linear constraint imposed on the filter tap weights. We also present an LMS algorithm for adaptive adjustment of the filter tap weights subject to the required constraint. For the sake of simplicity, all derivations are given for the case of real-valued signals. However, we also give a summary of the final results for the case of complex-valued signals. The application of the proposed algorithm to narrow-band beamforming is then discussed as an example.

6.10.1 Statement of the problem and its optimal solution

Given an observation vector $\mathbf{x}(n)$ and a desired response $d(n)$, we wish to find a tap-weight vector $\mathbf{w}$ so that

$$e(n) = d(n) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(n) \tag{6.140}$$

is minimized in the mean-square sense, subject to the constraint

$$\mathbf{c}^{\mathrm{T}}\mathbf{w} = a, \tag{6.141}$$

where a is a scalar and $\mathbf{c}$ is a fixed column vector.

This problem can be solved using the method of Lagrange multiplier. According to the method of Lagrange multiplier, we define (the superscript c stands for constraint)

$$\xi^{c} = \mathrm{E}[e^2(n)] + \lambda(\mathbf{c}^{\mathrm{T}}\mathbf{w} - a), \tag{6.142}$$

where λ is the Lagrange multiplier, and solve the equations

$$\nabla_{\mathbf{w}}\xi^{c} = \mathbf{0} \quad \text{and} \quad \frac{\partial \xi^{c}}{\partial \lambda} = 0 \tag{6.143}$$

simultaneously. We note that $\partial \xi^{c}/\partial \lambda = 0$ results in the constraint (6.141).

Substituting (6.140) in (6.142) and going through some manipulations similar to those in Chapter 4 (Section 4.3), we obtain

$$\xi^{c} = \xi_{\min} + \mathbf{v}^{\mathrm{T}}\mathbf{R}\mathbf{v} + \lambda(\mathbf{c}^{\mathrm{T}}\mathbf{v} - a'), \tag{6.144}$$

where $\mathbf{v} = \mathbf{w} - \mathbf{w}_{\mathrm{o}}, \mathbf{w}_{\mathrm{o}} = \mathbf{R}^{-1}\mathbf{p}, \mathbf{R} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)], \mathbf{p} = \mathrm{E}[d(n)\mathbf{x}(n)]$, and $a' = a - \mathbf{c}^{\mathrm{T}}\mathbf{w}_{\mathrm{o}}$. With this, the above problem is reduced to the minimization of $\mathbf{v}^{\mathrm{T}}\mathbf{R}\mathbf{v}$, subject to

the constraint $\mathbf{c}^T\mathbf{v} = a'$. The solution to this problem is obtained by simultaneous solution of

$$\nabla_{\mathbf{v}}\xi^c = 2\mathbf{R}\mathbf{v}_o^c + \lambda\mathbf{c} = \mathbf{0} \tag{6.145}$$

and

$$\frac{\partial\xi^c}{\partial\lambda} = \mathbf{c}^T\mathbf{v}_o^c - a' = 0, \tag{6.146}$$

where $\mathbf{v}_o^c$ is the constrained optimum value of $\mathbf{v}$.

From (6.145), we obtain

$$\mathbf{v}_o^c = -\frac{\lambda}{2}\mathbf{R}^{-1}\mathbf{c}. \tag{6.147}$$

Substituting (6.147) in (6.146) we get

$$-\frac{\lambda}{2}\mathbf{c}^T\mathbf{R}^{-1}\mathbf{c} - a' = 0$$

or

$$\lambda = -\frac{2a'}{\mathbf{c}^T\mathbf{R}^{-1}\mathbf{c}}. \tag{6.148}$$

Finally, substituting (6.148) in (6.147) we obtain

$$\mathbf{v}_o^c = \frac{a'\mathbf{R}^{-1}\mathbf{c}}{\mathbf{c}^T\mathbf{R}^{-1}\mathbf{c}}. \tag{6.149}$$

The minimum value of ξ^c is obtained by substituting (6.149) in (6.144). This gives

$$\xi_{\min}^c = \xi_{\min} + \frac{a'^2}{\mathbf{c}^T\mathbf{R}^{-1}\mathbf{c}}. \tag{6.150}$$

We note that the second term on the right-hand side of (6.150) is the excess MSE which is introduced as a result of the imposed constraint.

Also, noting that $\mathbf{w} = \mathbf{v} + \mathbf{w}_o$, and using (6.149), we obtain

$$\mathbf{w}_o^c = \mathbf{w}_o + \frac{a'\mathbf{R}^{-1}\mathbf{c}}{\mathbf{c}^T\mathbf{R}^{-1}\mathbf{c}}. \tag{6.151}$$

6.10.2 Update equations

The adaptation of the tap-weight vector $\mathbf{w}$, while the constraint (6.141) holds, may be done in two steps as follows:

Table 6.5 Summary of the linearly constrained LMS algorithm

Input:	Tap-weight vector, $\mathbf{w}(n)$, Input vector, $\mathbf{x}(n)$, and desired output, $d(n)$.
Output:	Filter output, $y(n)$, Tap-weight vector update, $\mathbf{w}(n+1)$.

1. Filtering:
   $y(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n)$
2. Error estimation:
   $e(n) = d(n) - y(n)$
3. Tap-weight vector adaptation:
   $\mathbf{w}^{+}(n) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n)$
   $$\mathbf{w}(n+1) = \mathbf{w}^{+}(n) + \frac{a - \mathbf{c}^{\mathrm{T}}\mathbf{w}^{+}(n)}{\mathbf{c}^{\mathrm{T}}\mathbf{c}}\mathbf{c}$$

Step 1:

$$\mathbf{w}^{+}(n) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n). \tag{6.152}$$

Step 2:

$$\mathbf{w}(n+1) = \mathbf{w}^{+}(n) + \boldsymbol{\vartheta}(n), \tag{6.153}$$

where $\boldsymbol{\vartheta}(n)$ is chosen so that $\mathbf{c}^{\mathrm{T}}\mathbf{w}(n+1) = a$, while $\boldsymbol{\vartheta}^{\mathrm{T}}(n)\boldsymbol{\vartheta}(n)$ is minimized. That is, *we choose $\boldsymbol{\vartheta}(n)$ so that the constraint* (6.141) *holds after Step 2, while the perturbation introduced by $\boldsymbol{\vartheta}(n)$ is minimized.*

The latter problem can also be solved using the Lagrange multiplier and following a procedure similar to the one used above to obtain $\mathbf{v}_{\mathrm{o}}^{\mathrm{c}}$. This gives

$$\boldsymbol{\vartheta}(n) = \frac{a - \mathbf{c}^{\mathrm{T}}\mathbf{w}^{+}(n)}{\mathbf{c}^{\mathrm{T}}\mathbf{c}}\mathbf{c}. \tag{6.154}$$

Substituting this result in (6.153), we obtain

$$\mathbf{w}(n+1) = \mathbf{w}^{+}(n) + \frac{a - \mathbf{c}^{\mathrm{T}}\mathbf{w}^{+}(n)}{\mathbf{c}^{\mathrm{T}}\mathbf{c}}\mathbf{c}. \tag{6.155}$$

The above derivations are summarized in Table 6.5.

6.10.3 Extension to the complex-valued case

When the underlying signal/variables are complex-valued, the following amendments have to be made to the previous results:

- The constraint equation (6.141) is written as

$$\mathbf{w}^{\mathrm{H}}\mathbf{c} = a, \tag{6.156}$$

where the vector $\mathbf{c}$ and the scalar a are both complex-valued.

- The constrained optimum tap-weight vector of the filter is obtained according to the equation

$$\mathbf{w}_{\mathrm{o}}^{\mathrm{c}} = \mathbf{w}_{\mathrm{o}} + \frac{a'^{*}\mathbf{R}^{-1}\mathbf{c}}{\mathbf{c}^{\mathrm{H}}\mathbf{R}^{-1}\mathbf{c}}, \tag{6.157}$$

where $a' = a - \mathbf{w}_{\mathrm{o}}^{\mathrm{H}}\mathbf{c}$.

- The adaptation of the filter tap weights is made according to the following equations:

$$e(n) = d(n) - \mathbf{w}^{\mathrm{H}}(n)\mathbf{x}(n), \tag{6.158}$$

$$\mathbf{w}^{+}(n) = \mathbf{w}(n) + 2\mu e^{*}(n)\mathbf{x}(n) \tag{6.159}$$

and

$$\mathbf{w}(n+1) = \mathbf{w}^{+}(n) + \frac{a^{*} - \mathbf{c}^{\mathrm{H}}\mathbf{w}^{+}(n)}{\mathbf{c}^{\mathrm{H}}\mathbf{c}}\mathbf{c}. \tag{6.160}$$

Example 6.4

As an example of the linearly constrained LMS algorithm, we consider the two-element narrowband beamformer of Figure 6.20. Here, we consider the processing of signals in the baseband, i.e. phasor (complex-valued) signals are considered. We note that with $s(n)$ and $v(n)$, as defined in Section 6.4.4,

$$\underline{x}_0(n) = \alpha(n)\,\mathrm{e}^{j\phi_1} + \beta(n)\,\mathrm{e}^{j(\phi_2 - \phi_o)}, \tag{6.161}$$

$$\underline{x}_1(n) = \alpha(n)\,\mathrm{e}^{j\phi_1} + \beta(n)\,\mathrm{e}^{j\phi_2}. \tag{6.162}$$

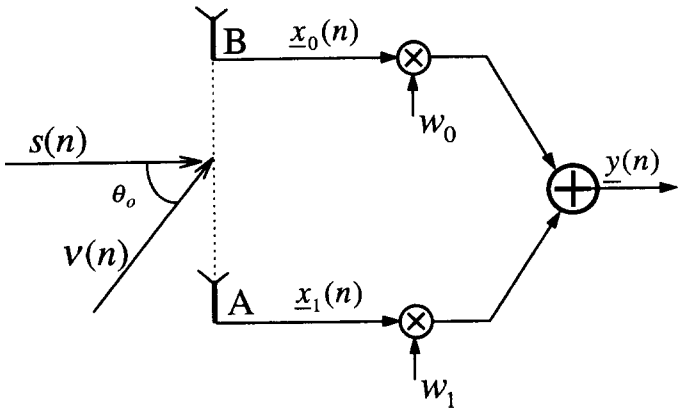


Figure 6.20 Baseband implementation of a two-element beamformer with a linearly constrained beam pattern

The beamformer tap weights, w_0 and w_1, are adjusted so that their outputs, $y(n)$, are minimized in the mean-square sense. This is equivalent to saying that $\underline{d}(n) = 0$. It is clear that if there is no constraint on the tap weights and they are adjusted to minimize $\mathrm{E}[|y(n)|^2]$, then we obtain the undesirable result of $w_{0,o} = w_{1,o} = 0$, which cancels both the jammer and the desired signal. To ensure that the desired signal, $s(n)$, arriving in the direction perpendicular to the line connecting A to B, passes through the beamformer with no distortion, the following constraint must hold:

$$w_0 + w_1 = 1.$$

Using vector notations, this may be written as

$$\mathbf{w}^{\mathrm{H}}\mathbf{c} = 1, \tag{6.163}$$

where $\mathbf{c} = [1 \;\; 1]^{\mathrm{T}}$ and $\mathbf{w} = [w_0^* \;\; w_1^*]^{\mathrm{T}}$. We note that, in general, the value of $\mathbf{c}$ will depend on the angle of arrival of the desired signal $s(n)$ with respect to the perpendicular to the line connecting A to B; see Problems P6.24.

Letting $\mathbf{x}(n) = [\underline{x}_0(n) \;\; \underline{x}_1(n)]^{\mathrm{T}}$ and noting that $\alpha(n)$ and $\beta(n)$ are uncorrelated with each other, we get

$$\mathbf{R} = \begin{bmatrix} \sigma_\alpha^2 + \sigma_\beta^2 & \sigma_\alpha^2 + \sigma_\beta^2 \mathrm{e}^{-j\phi_0} \\ \sigma_\alpha^2 + \sigma_\beta^2 \mathrm{e}^{j\phi_0} & \sigma_\alpha^2 + \sigma_\beta^2 \end{bmatrix}. \tag{6.164}$$

Using this result and noting that in the present case $\mathbf{c} = [1 \;\; 1]^{\mathrm{T}}$, $a = 1$, and $\mathbf{w}_0 = \mathbf{0}$, we obtain, from (6.157),

$$\mathbf{w}_0^{\mathrm{c}} = \frac{1}{2(1 - \cos\phi_0)} \begin{bmatrix} 1 - \mathrm{e}^{-j\phi_0} \\ 1 - \mathrm{e}^{j\phi_0} \end{bmatrix}. \tag{6.165}$$

Using this, we get

$$y_0(n) = (\mathbf{w}_0^{\mathrm{c}})^{\mathrm{H}}\mathbf{x}(n) = \alpha(n)\mathrm{e}^{j\phi_1}$$

which means that the desired signal, $\underline{s}(n)$, passes through the beamformer with no distortion, while the jammer, $\underline{\nu}(n)$, is completely cancelled.

Problems

P6.1 Show that when an adaptive filter with Gaussian underlying signals has converged and $\mathbf{w}(n) \approx \mathbf{w}_0$

$$\text{the variance of } [\nabla e^2(n)]_i \approx 4\xi_{\min}\mathrm{E}[x^2(n)]$$

where $[\nabla e^2(n)]_i$ is the ith element of the gradient vector $\nabla e^2(n)$.

P6.2 Formulate the LMS algorithm for a one-step-ahead N-tap linear predictor, i.e. a filter that predicts $x(n)$ based on a linear combination of its past samples, $x(n-1)$, $x(n-2), \ldots, x(n-N)$.

P6.3 By multiplying $\mathbf{A} + \alpha\mathbf{a}\mathbf{a}^{\mathrm{T}}$ with the right-hand side of (6.59) give a proof of that.

P6.4 Prove that if a and b are two positive values and $a + b < 1$,

$$\frac{a}{1-a} + \frac{b}{1-b} < \frac{a+b}{1-(a+b)}.$$

Use this result to establish the inequality (6.69).

P6.5 A 10-tap transversal adaptive filter is adapted using the LMS algorithm. Consider five cases of the filter input which are characterized by the following eigenvalues:

Case	λ_0	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
3	1.0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
4	1.0	1.0	1.0	1.0	1.0	0.1	0.1	0.1	0.1	0.1
5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.1

(i) Make a plot of $\mathcal{J}$ (as defined in (6.64)) for each case when μ varies from 0 to 1.
(ii) Find the range of μ in each case which results is a stable LMS algorithm.
(iii) Discuss on the various ranges obtained in (ii) and try to relate those to the distribution of the eigenvalues associated with the filter input.

P6.6 Equations (6.83) and (6.86) provide approximate expressions for the expected learning curves of the LMS algorithm in the two cases of system modelling and channel equalization. For the five cases noted in Problem P6.5, plot the expected learning curves of the LMS algorithm for system modelling and channel equalization and discuss on your observation.

P6.7 Consider a channel equalization problem similar to the one depicted in Figure 6.8. The magnitude response of the channel, $|H(z)|$, is as shown is Figure P6.7. The additive noise at the channel output has a variance of $\sigma_\nu^2 = 0.04$. The transmitted data symbols, the $s(n)$s, take the values of $+1$ and -1, and are samples of a white noise process.

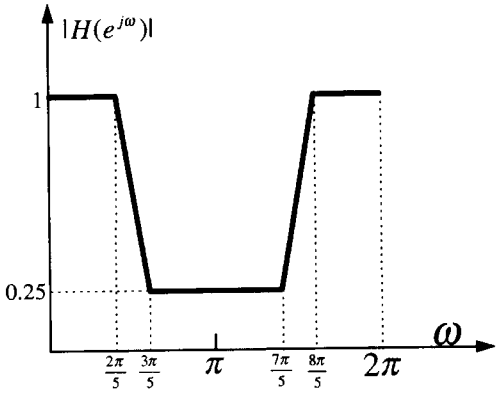


Figure P6.7

(i) Draw the power spectral density of the sequence $x(n)$ and obtain an estimate of $E[|x(n)|^2]$.
(ii) Give estimates of the maximum and minimum eigenvalues of the correlation matrix of the input process to the equalizer.
(iii) When the conventional LMS algorithm is used to adjust the equalizer tap weights and the equalizer has 20 taps, what is the value of the step-size parameter μ that results in a 10% misadjustment?
(iv) Obtain the range of time constants of the LMS algorithm in the present case and plot a typical learning curve for it.

P6.8 The LMS algorithm is used to adapt an adaptive filter with tap-weight vector $\mathbf{w}(n)$. Define $\bar{\mathbf{v}}(n) = E[\mathbf{w}(n) - \mathbf{w}_o]$, where $E[\cdot]$ denotes statistical expectation and $\mathbf{w}_o$ is the optimum value of the filter tap-weight vector.

(i) Show that if the step-size parameter, μ, is properly selected, $|\bar{\mathbf{v}}(n)|^2 = \bar{\mathbf{v}}^T(n)\bar{\mathbf{v}}(n)$ will approach zero, as n increases.
(ii) Find the range of μ that guarantees convergence of $|\bar{\mathbf{v}}(n)|^2$. Does this range guarantee the convergence of the LMS algorithm?
(iii) Find the time constants that govern the convergence of $|\bar{\mathbf{v}}(n)|^2$.

P6.9 A communication channel with a finite impulse response shorter than or equal to M bit interval is to be identified using the set-up shown in Figure P6.9. The transmitted data bits, $s(n)$, which take values of $+1$ and -1, are passed through the channel, $H_o(z)$. The same data bits are passed through an adaptive filter, $H(z)$, which is adapted through the LMS algorithm so that its output matches the output of the channel in the mean-square sense. The channel noise is modelled as an additive noise sequence $\nu(n)$ with variance σ_ν^2. The sequences $s(n)$ and $\nu(n)$ are independent of each other. Define the length M column vector $\mathbf{g}(n) = \mathbf{h}(n) - \mathbf{h}_o$, where $\mathbf{h}(n)$ is the channel model tap-weight vector at iteration n and the elements of the vector $\mathbf{h}_o$ are the samples of channel response.

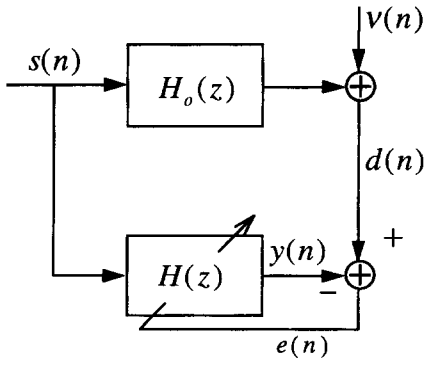


Figure P6.9

(i) Show that

$$\mathbf{g}(n+1) = (\mathbf{I} - 2\mu\mathbf{s}(n)\mathbf{s}^{\mathrm{T}}(n))\mathbf{g}(n) + 2\mu\nu(n)\mathbf{s}(n)$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{s}(n) = [s(n)\ \ s(n-1)\ \ \ldots\ \ s(n-M+1)]^{\mathrm{T}}$, and μ is the LMS algorithm step-size parameter.

(ii) Use the independence assumption to show that

$$\|\mathbf{g}(n+1)\|^2 = \mathrm{E}[\mathbf{g}^{\mathrm{T}}(n)(\mathbf{I} - (4\mu - 4M\mu^2)\mathbf{R}_{ss})\mathbf{g}(n)] + 4M\mu^2\sigma_\nu^2,$$

where $\|\mathbf{g}(n)\|^2 = \mathrm{E}[\mathbf{g}^{\mathrm{T}}(n)\mathbf{g}(n)]$ and $\mathbf{R}_{ss} = \mathrm{E}[\mathbf{s}(n)\mathbf{s}^{\mathrm{T}}(n)]$.

(iii) Use the result of part (ii) to find the range of μ that guarantees the convergence of $\|\mathbf{g}(n)\|^2$. Does this also guarantee the convergence of the LMS algorithm?

(iv) Compare the range obtained in part (iii) with the range of μ given in (6.73).

P6.10 In this problem we discuss the effect of the power level of the input process to an adaptive filter and its variation on the convergence of the LMS algorithm.

(i) Consider the LMS recursion (6.9) and assume that the time constants of its different modes of convergence are $\tau_0, \tau_1, \ldots, \tau_{N-1}$. Keep μ fixed, replace $\mathbf{x}(n)$ by $\mathbf{x}'(n) = \alpha\mathbf{x}(n)$, where α is a constant, and obtain the corresponding time constants of the resulting recursion, in terms of the τ_is, under the condition that the step-size parameter μ is small enough to guarantee the convergence of the algorithm.

(ii) Under the condition that the power levels of the elements of $\mathbf{x}(n)$ are time varying and fluctuate slowly between high and low levels, what is the shortcoming of the LMS algorithm (discuss)? Can you suggest any solution to this?

P6.11 This problem attempts to show the validity of the approximation (6.85) in a non-rigorous manner. Consider a random process $x(n)$ and its associated $(2M+1)\times(2M+1)$ correlation matrix $\mathbf{R}$. Let

$$\mathbf{q}_i = [q_{i,-M}\ \ \cdots\ \ q_0\ \ \cdots\ \ q_{i,M}]^{\mathrm{T}}, \quad \text{with } \mathbf{q}_i^{\mathrm{H}}\mathbf{q}_i = 1,$$

be the ith eigenvector of $\mathbf{R}$ and λ_i be its corresponding eigenvalue.

(i) Show that the expansion of the relationship $\mathbf{R}\mathbf{q}_i = \lambda_i\mathbf{q}_i$ leads to

$$\sum_{k=-M}^{M} \phi_{xx}(k-l)q_{i,k} = \lambda_i q_{i,l}, \quad \text{for } -M \le l \le M, \tag{P6.11-1}$$

where $\phi_{xx}(k-l)$ is the autocorrelation function of $x(n)$ for lag $k-l$.

(ii) Let $M \to \infty$ and take the Fourier transform of both sides of (P6.11-1). Show that this leads to the identity

$$\Phi_{xx}(\mathrm{e}^{j\omega})Q_i(\mathrm{e}^{j\omega}) = \lambda_i Q_i(\mathrm{e}^{j\omega}), \tag{P6.11-2}$$

where $\Phi_{xx}(\mathrm{e}^{j\omega})$ is the power spectral density of $x(n)$ and

$$Q_i(\mathrm{e}^{j\omega}) = \sum_{k=-\infty}^{\infty} q_{i,k}\,\mathrm{e}^{-j\omega k}.$$

(iii) Consider the case when $\Phi_{xx}(e^{j\omega})$ is a single-valued function of the angular frequency ω. Using (P6.11-2), show that

$$Q_i(e^{j\omega}) = \begin{cases} \text{a non-zero value,} & \text{for } \omega = \omega_i, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, argue that when M is large, the set of vectors

$$\mathbf{q}_i = \frac{1}{\sqrt{2M+1}}[e^{-j\frac{2\pi iM}{2M+1}} \quad e^{-j\frac{2\pi i(M-1)}{2M+1}} \quad \cdots \quad e^{j\frac{2\pi iM}{2M+1}}]^T$$

for $-M \leq i \leq M$, may be considered as an approximation to the eigenvectors of $\mathbf{R}$. Also, from the Parseval's relation (see Chapter 2) recall that

$$\mathbf{q}_i^H \mathbf{q}_i = \frac{1}{2\pi}\int_{-\pi}^{\pi} |Q_i(e^{j\omega})|^2 \, d\omega.$$

Thus, conclude that

$$|Q_i(e^{j\omega})|^2 = 2\pi\delta(\omega - \omega_i)$$

where $\delta(\cdot)$ is the Kronecker delta function and $\omega = \omega_i$ is the solution of the equation $\Phi_{xx}(e^{j\omega}) = \lambda_i$.

(iv) Extend the above result and argue that the latter approximation is also valid in the cases where $\Phi_{xx}(e^{j\omega})$ is not necessarily single-valued.

Now consider the case where $x(n)$ is the output of a channel with system function $H(z)$ and also the input to a $(2M+1)$-tap equalizer $W(z)$, as in Section 6.4.2. Ignore the channel noise and recall that, if the system delay Δ is assumed to be zero, the transmitted data symbols are assumed to be uncorrelated, and the equalizer is allowed to be non-causal,

$$W_o(z) \approx \frac{1}{H(z)}$$

where $W_o(z)$ is the optimum setting of $W(z)$.

(v) Using the approximation derived in Parts (iii) and (iv), show that when M is large

$$w'_{o,i} = \mathbf{q}_i^H \mathbf{w}_o \approx \frac{1}{\sqrt{2M+1}} \cdot \frac{1}{H(e^{j\frac{2\pi i}{2M+1}})} \quad \text{for } -M \leq i \leq M$$

where $\mathbf{w}_o$ is the optimum tap-weight vector of the equalizer.

(vi) Using this result show that

$$\lambda_i |w'_{o,i}|^2 \approx \frac{1}{2M+1}.$$

P6.12 The sequence $u(n) = \cos(n\omega_0 + \phi(n))$ is a narrow-band phase modulated sampled signal. The phase angle $\phi(n)$ is random, but varies slowly in time so that $\phi(n) \approx \phi(n-1) \approx \phi(n-2)$. The aim is to detect the carrier frequency ω_0 of $u(n)$. It is proposed that the set-up shown in Figure P6.12 be used. The coefficient w has to be adjusted so as to minimize the output, $y(n)$, in the mean-square sense.

(i) Show that the optimized value of w is

$$w_0 \approx -2\cos\omega_0.$$

(ii) Formulate the LMS algorithm for the present problem. In particular, specify the filter tap-weight vector, $\mathbf{w}(n)$, input vector, $\mathbf{x}(n)$, the desired output, $d(n)$, and how the output error is defined in the present case.

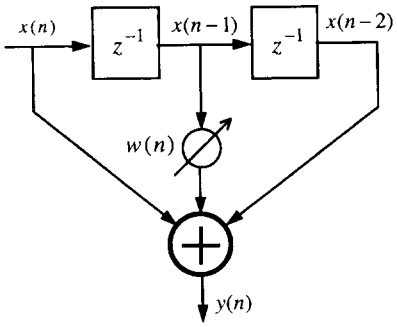


Figure P6.12

P6.13 This problem, the aim of which is to study the signed-regressor algorithm in some detail, is based on the derivations of Eweda (1991a).

Using Price's theorem (Papoulis, 1991), we can show that if x and y are a pair of zero-mean jointly Gaussian random variables, then

$$\mathrm{E}[x \cdot \mathrm{sign}(y)] = \frac{1}{\sigma_y}\sqrt{\frac{2}{\pi}}\,\mathrm{E}[xy].$$

Consider the signed-regressor algorithm introduced in Section 6.5, and let the assumptions made at the beginning of Section 6.3 apply. Show that:

(i) $$\mathrm{E}[\mathrm{sign}(\mathbf{x}(n))\mathbf{x}^{\mathrm{T}}(n)] = \mathrm{E}[\mathbf{x}(n)\mathrm{sign}(\mathbf{x}^{\mathrm{T}}(n))] = \frac{1}{\sigma_x}\sqrt{\frac{2}{\pi}}\mathbf{R}.$$

(ii) $$\mathbf{v}(n+1) = (\mathbf{I} - 2\mu\,\mathrm{sign}(\mathbf{x}(n))\mathbf{x}^{\mathrm{T}}(n))\mathbf{v}(n) + 2\mu e_0(n)\,\mathrm{sign}(\mathbf{x}(n)).$$

(iii) $$\mathrm{E}[\mathbf{v}(n+1)] = \left(\mathbf{I} - 2\mu\frac{1}{\sigma_x}\sqrt{\frac{2}{\pi}}\mathbf{R}\right)\mathrm{E}[\mathbf{v}(n)], \qquad \text{(P6.13-1)}$$

and from there argue that the signed-regressor algorithm follows the same trajectory as the conventional LMS algorithm.

(iv) Define $\|\mathbf{v}(n)\|^2 = \mathrm{E}[\mathbf{v}^{\mathrm{T}}(n)\mathbf{v}(n)]$ and show that

$$\|\mathbf{v}(n+1)\|^2 = \|\mathbf{v}(n)\|^2 - 4\left(\frac{1}{\sigma_x}\sqrt{\frac{2}{\pi}}\,\mu - \mu^2 N\right)\mathrm{E}[\mathbf{v}^{\mathrm{T}}(n)\mathbf{R}\mathbf{v}(n)] + 4\mu^2 N\xi_{\min}.$$

(v) Assuming that the signed-regressor algorithm is convergent, show that its misadjustment is given by

$$\mathcal{M} = \frac{\mu N}{\dfrac{1}{\sigma_x}\sqrt{\dfrac{2}{\pi}} - \mu N}. \tag{P6.13-2}$$

(vi) From this result and following a line of argument similar to the one in Section 6.3.5, show that the signed-regressor algorithm remains stable when

$$0 < \mu < \frac{1}{N\sigma_x}\sqrt{\frac{2}{\pi}}.$$

(vii) When the step-size parameter, μ, is small (P6.13-2) reduces to

$$\mathcal{M} \approx \mu N \sigma_x \sqrt{\frac{\pi}{2}}.$$

Using this result and (P6.13-1) and comparing these with their counterparts in the conventional LMS algorithm, show that when the step-size parameters of the two algorithms are chosen so that both result in the same misadjustment, the signed-regressor algorithm is $2/\pi$ times slower than the conventional LMS algorithm.

P6.14 Consider a case where the input vector, $\mathbf{x}(n)$, to an adaptive filter and its desired output, $d(n)$, are fixed for all values of n. Assuming an initial value, $\mathbf{w}(0)$, for the filter tap weight and running the LMS algorithm with a small step-size parameter (which guarantees its stability), find the final setting of the filter tap weights after the convergence of the LMS algorithm. Using the result obtained, confirm Nitzberg's interpretation (see Section 6.6) of the NLMS algorithm.

P6.15 In the derivation of the NLMS recursion (6.106), we searched for the step-size parameter, $\mu(n)$, that would minimize $(e^+(n))^2$, where $e^+(n)$ is as defined in (6.103) . We may also note that $e^+(n) = e(n) - \mathbf{x}^{\mathrm{T}}(n)\boldsymbol{\eta}(n)$, where $\boldsymbol{\eta}(n) = \mathbf{w}(n+1) - \mathbf{w}(n)$, as defined in (6.107). Furthermore, we note that to have an LMS algorithm with a variable step-size parameter, the increment $\boldsymbol{\eta}(n)$ has to be in the direction of $\mathbf{x}(n)$, i.e. we may write $\boldsymbol{\eta}(n) = \alpha(n)\mathbf{x}(n)$, where $\alpha(n)$ is scalar.

(i) Give an alternative derivation of the NLMS algorithm by optimizing $\alpha(n)$ so that $(e^+(n))^2$ is minimized.

(ii) To limit the perturbation introduced by the vector $\boldsymbol{\eta}(n)$, it is proposed that $(e^+(n))^2 + \psi\boldsymbol{\eta}^{\mathrm{T}}(n)\boldsymbol{\eta}(n)$ be minimized. Show that this leads to the recursion

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n) + \psi}e(n)\mathbf{x}(n).$$

P6.16 The following recursions have been proposed for implementation of a variable step-size LMS algorithm:

$$\boldsymbol{\mu}(n) = \boldsymbol{\mu}(n-1) - \rho \nabla_{\boldsymbol{\mu}} e^2(n)$$

and

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \boldsymbol{\mu}(n) \nabla_{\mathbf{w}} e^2(n),$$

where $\boldsymbol{\mu}(n)$ is a diagonal matrix consisting of N separate variable step-size parameters, $\mu_0(n), \mu_1(n), \ldots, \mu_{N-1}(n)$, ρ is a small positive step-size parameter, the gradient $\nabla_{\boldsymbol{\mu}} e^2(n)$ is a diagonal matrix compatible with $\boldsymbol{\mu}(n)$, and is evaluated at $\boldsymbol{\mu} = \boldsymbol{\mu}(n-1)$, the gradient $\nabla_{\mathbf{w}} e^2(n)$ is a column vector, as usual, and is evaluated at $\mathbf{w} = \mathbf{w}(n)$.

Show that the above proposal leads to the VSLMS algorithm that was introduced in Section 6.7 and summarized in Table 6.3.

P6.17 Assuming that a scalar variable step-size parameter, $\mu(n)$, is used for all taps of a transversal adaptive filter, show that a derivation similar to the one discussed in Problem P6.16 leads to the following recursion:

$$\mu(n) = \mu(n-1) + 4\rho e(n) e(n-1) \mathbf{x}^{\mathrm{T}}(n) \mathbf{x}(n-1).$$

P6.18 Give details of the derivation of (6.60) from (6.57).

P6.19 This problem looks at a variation of the LMS algorithm called the *leaky LMS algorithm*. The leaky LMS algorithm works on the basis of the recursion

$$\mathbf{w}(n+1) = \beta \mathbf{w}(n) + 2\mu e(n) \mathbf{x}(n),$$

where β is a constant slightly smaller than one.

(i) Define $\overline{\mathbf{w}}(n) = \mathrm{E}[\mathbf{w}(n)]$, where $\mathrm{E}[\cdot]$ denotes statistical expectation, and use the *independence assumption* of Section 6.3 to show that the following recursive equation holds:

$$\overline{\mathbf{w}}(n+1) = (\mathbf{I} - 2\mu \mathbf{R}') \overline{\mathbf{w}}(n) + 2\mu \mathbf{p}.$$

Specify $\mathbf{R}'$ and $\mathbf{p}$, and obtain the time constants of the learning curve of the leaky LMS algorithm in terms of the eigenvalues of the correlation matrix $\mathbf{R} = E[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]$, and the parameters β and μ.

(ii) Assuming that the step-size parameter μ is small enough to guarantee the convergence of the leaky LMS algorithm, derive an equation for $\overline{\mathbf{w}}(\infty)$ in terms of $\mathbf{R}'$ and $\mathbf{p}$.

(iii) Show that the difference between $\overline{\mathbf{w}}(\infty)$ and the optimum tap-weight vector of the adaptive filter is given by the following equation:

$$\overline{\mathbf{w}}(\infty) - \mathbf{w}_{\mathrm{o}} = -\gamma \sum_{i=0}^{N-1} \frac{\mathbf{q}_i^{\mathrm{T}} \mathbf{p}}{\lambda_i(\lambda_i + \gamma)} \mathbf{q}_i,$$

where $\gamma = (1-\beta)/2\mu$, and the λ_is and $\mathbf{q}_i$s are the eigenvalues and eigenvectors of $\mathbf{R}$, respectively.

P6.20 Define the scalar value $\|\mathbf{v}(n)\|^2 = \mathrm{E}[\mathbf{v}^{\mathrm{T}}(n)\mathbf{v}(n)]$ as the misalignment of an adaptive filter tap-weight vector.

(i) Show that

$$\|\mathbf{v}(n)\|^2 = \mathrm{tr}[\mathbf{K}'(n)]$$

where the correlation matrix $\mathbf{K}'(n)$ is defined as in (6.31).

(ii) Use (6.52) to show that

$$\|\mathbf{v}(\infty)\|^2 = \frac{\displaystyle\sum_{i=0}^{N-1} \frac{\mu\xi_{\min}}{1-2\mu\lambda_i}}{1 - \displaystyle\sum_{i=0}^{N-1} \frac{\mu\lambda_i}{1-2\mu\lambda_i}}.$$

(iii) Show that when μ is small, the above result reduces to

$$\|\mathbf{v}(\infty)\|^2 \approx \mu\xi_{\min} N.$$

P6.21 A complex-valued sinusoidal process, $u(n) = a\,\mathrm{e}^{j(\omega n+\phi)}$, where a and ϕ are random, but fixed for every realization, is mixed with an unknown complex-valued noise sequence, $\nu(n)$, which may not be white. Assuming that the frequency, ω, of $u(n)$ is known, propose an adaptive filter and its associated adaptation algorithm to filter out $\nu(n)$ and enhance $u(n)$ in the minimum MSE sense, preserving its phase, ϕ, and its amplitude, a.

P6.22 Repeat Problem P6.21 when $u(n) = a\cos(\omega n + \phi)$ and $\nu(n)$ is a real-valued noise sequence.

P6.23 Give details of the linearly constrained LMS algorithm required for adaptation of the tap-weight vector, $\mathbf{w}$, of the beamformer of Example 6.4.

P6.24 The beamformer discussed in Example 6.4 assumes that the desired signal is arriving in the direction perpendicular to the line connecting A to B. What constraint had to be imposed on the tap weights w_0 and w_1, if the desired signal was arriving at an angle $\theta = \theta_d$?

P6.25 Griffiths and Jim (1982) have proposed a structure for beamforming the performance of which is similar to that of the Frost algorithm; however, it does not need any constraint to applied. Example 6.4 is an example of Frost algorithm. The equivalent implementation of Figure 6.20 which follows the idea of Griffiths and Jim is shown in Figure P6.25. Note that here the beamformer has only one tap weight, as opposed to Figure 6.20 which has two tap weights. Also, adaptation of the tap weight w of Figure P6.25 is based on the conventional LMS algorithm, as opposed to the Frost implementation (Figure 6.20) which requires the use of the linearly constrained LMS algorithm.

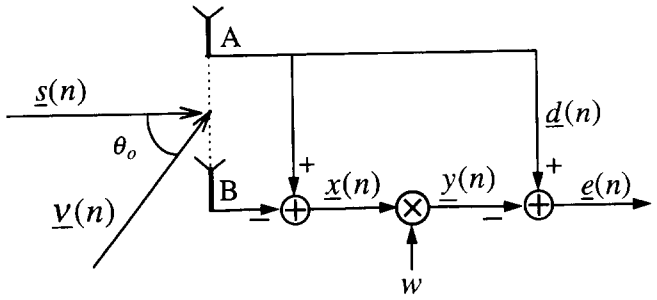


Figure P6.25

(i) Explore the validity of the Griffiths and Jim algorithm in the present case.
(ii) Figure P6.25 assumes that the desired signal is arriving in the direction perpendicular to the line connecting A to B. Modify this structure for the case when the desired signal is arriving at an angle $\theta = \theta_d$.

Simulation-Oriented Problems

All the programs that have been used to generate the results of the various examples of this chapter are written in MATLAB software package and are available on an accompanying diskette. The reader is encouraged to run these programs and confirm the results of this chapter. It would also be useful and enlightening if the reader tries other variations of the simulation parameters, such as the colouring filter in modelling, the channel response in channel equalization, and the noise and sinusoidal signal powers in the adaptive line enhancer. The following problems (case studies) are designed to guide the reader to many other interesting results.

P6.26 Consider the transfer function $H_1(z)$, of (6.79), as the channel response in the equalizer set-up of Figure 6.8. Set the equalizer length, N, equal to 15. Find the minimum MSE of the equalizer for values of the delay, Δ, in the range 3 to 15. Repeat this experiment for the transfer function $H_2(z)$, of (6.80), as well. For each case find the optimum value of the delay, Δ, that results in the minimum MSE. From these observations arrive at a rule-of-thumb for the selection of Δ in terms of the duration of the channel response and equalizer length.

P6.27 In the line enhancer problem that was studied in Section 6.4.3, we noted that no slow mode appears on its learning curve when the tap weights are initialized to zero. To observe the slow modes of the line enhancer, perform the following experiment. Run the line enhancer program 'lenhncr.m' (available on the accompanying diskette), starting with zero tap weights and using an input similar to the one used to obtain the result of Figure 6.11. Rerun the line enhancer program, with randomized initial tap weights. The program 'lenhncr.m' has the option of randomizing the initial tap weights. Compare the two learning curves that you have obtained and explain your observation.

P6.28 For the modelling problem that was discussed in Section 6.4.1, develop a program to study the convergence behaviour of the NLMS algorithm. Compare your results with those of the conventional LMS algorithm.

P6.29 Repeat Problem P6.28 when the VSLMS algorithm is used for adaptation of the filter.

P6.30 Repeat Problems P6.28 and P6.29 for the case where the adaptive filter of interest is the channel equalizer discussed in Section 6.4.2.

P6.31 Write a program to study the convergence behaviour of the line enhancer of Section 6.4.3 when the input, $x(n)$, is given by

$$x(n) = a\sin(\omega_1 n + \theta_1) + b\sin(\omega_2 n + \theta_2) + \nu(n),$$

where θ_1 and θ_2 are random phases that are uniformly distributed in the range 0 to 2π. Obtain the learning curves of the line enhancer for the following choices of the signal parameters:

(i) $\omega_1 = \pi/6$, $\omega_2 = 5\pi/8$, $a = 1$, $b = 1$, $\sigma_\nu^2 = 0.1$.
(ii) $\omega_1 = \pi/6$, $\omega_2 = 5\pi/8$, $a = 5$, $b = 1$, $\sigma_\nu^2 = 0.1$.
(iii) $\omega_1 = \pi/6$, $\omega_2 = \pi/4$, $a = 5$, $b = 1$, $\sigma_\nu^2 = 0.1$.

Run your program for the cases when the filter tap weights are initialized to zero, and also when they are initialized to some random values. Study the results that you obtain and explain your observations.

P6.32 Write your own program to confirm the results of Figure 6.13.

P6.33 Consider a communication channel with a complex-valued impulse response consisting of the following samples:

$$-0.1 + j0.2, \quad 0.15 - j0.4, \quad 1, \quad 0.5 - j0.2, \quad -0.2 + j0.1,$$

where $j = \sqrt{-1}$. The input data symbols are randomly selected from the alphabets

$$1 + j, \quad -1 + j, \quad -1 - j, \quad \text{and} \quad 1 - j$$

with equal probability. The channel noise is white and at 30 dB below the signal level at the equalizer input. Develop a program to simulate this scenario and study the convergence behaviour of the LMS algorithm in this case.

P6.34 Consider the scenario discussed in Example 6.4. Develop a program for the adaptive adjustment of the coefficients w_0 and w_1. By running your program for different choices of σ_α^2, σ_β^2, ϕ_1 and ϕ_2, study the behaviour of the constrained LMS algorithm in this case.

Appendix 6A: Derivation of (6.39)

Using the independence of $\mathbf{x}'(n)$ and $\mathbf{v}'(n)$, we get

$$\begin{aligned}
&\mathrm{E}\left[\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\mathbf{v}'(n)\mathbf{v}'^{\mathrm{T}}(n)\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\right] \\
&\quad = \mathrm{E}\left[\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\mathrm{E}[\mathbf{v}'(n)\mathbf{v}'^{\mathrm{T}}(n)]\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\right] \\
&\quad = \mathrm{E}\left[\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\mathbf{K}'(n)\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\right]. \qquad \text{(6A-1)}
\end{aligned}$$

To expand the right-hand side of (6A-1), we first note that

$$\mathbf{x}'^{\mathrm{T}}(n)\mathbf{K}'(n)\mathbf{x}'(n) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_i'(n)x_j'(n)k_{ij}'(n), \qquad \text{(6A-2)}$$

where $x_i'(n)$ is the ith element of the vector $\mathbf{x}'(n)$. We also define

$$\mathbf{C}(n) = \mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\mathbf{K}'(n)\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n) \qquad \text{(6A-3)}$$

and note that it is an $N \times N$ matrix. The lmth element of $\mathbf{C}(n)$ is

$$c_{lm}(n) = x_l'(n)x_m'(n)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_i'(n)x_j'(n)k_{ij}'(n). \qquad \text{(6A-4)}$$

Taking statistical expectation on both sides of (6A-4), we obtain

$$\mathrm{E}[c_{lm}(n)] = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \mathrm{E}[x_l'(n)x_m'(n)x_i'(n)x_j'(n)]k_{ij}'(n). \qquad \text{(6A-5)}$$

Next, if we consider the assumption that the input samples $x(n)$ (and, thus, the $x_i'(n)$s) are a set of mutually Gaussian random variables, and note that for any set of real-valued mutually Gaussian random variables x_1, x_2, x_3 and x_4

$$\mathrm{E}[x_1x_2x_3x_4] = \mathrm{E}[x_1x_2]\mathrm{E}[x_3x_4] + \mathrm{E}[x_1x_3]\mathrm{E}[x_2x_4] + \mathrm{E}[x_1x_4]\mathrm{E}[x_2x_3], \qquad \text{(6A-6)}$$

and, also,

$$\mathrm{E}[x_i'(n)x_j'(n)] = \lambda_i\delta(i-j), \qquad \text{(6A-7)}$$

where $\delta(\cdot)$ is the Kronecker delta function, we obtain

$$\begin{aligned}
\mathrm{E}[x_l'(n)x_m'(n)x_i'(n)x_j'(n)] &= \lambda_l\lambda_i\delta(l-m)\delta(i-j) + \lambda_l\lambda_m\delta(l-i)\delta(m-j) \\
&\quad + \lambda_l\lambda_m\delta(l-j)\delta(m-i). \qquad \text{(6A-8)}
\end{aligned}$$

Substituting (6A-8) in (6A-5) we obtain

$$\begin{aligned}
\mathrm{E}[c_{lm}(n)] &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \lambda_l\lambda_i\delta(l-m)\delta(i-j)k'_{ij}(n) \\
&\quad + \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \lambda_l\lambda_m\delta(l-i)\delta(m-j)k'_{ij}(n) \\
&\quad + \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \lambda_l\lambda_m\delta(l-j)\delta(m-i)k'_{ij}(n) \\
&= \lambda_l\delta(l-m)\sum_{i=0}^{N-1}\lambda_i k'_{ii}(n) + \lambda_l\lambda_m k'_{lm}(n) + \lambda_m\lambda_l k'_{ml}(n), \qquad \text{(6A-9)}
\end{aligned}$$

for $l = 0, 1, \ldots, N-1$ and $m = 0, 1, \ldots, N-1$. Noting that $k'_{lm}(n) = k'_{ml}(n)$, $\sum_{i=0}^{N-1}\lambda_i k'_{ii}(n) = \mathrm{tr}[\mathbf{\Lambda}\,\mathbf{K}'(n)]$, and using the result of (6A-9) to construct the matrix

$$\mathrm{E}[\mathbf{C}(n)] = \mathrm{E}\left[\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\mathbf{v}'(n)\mathbf{v}'^{\mathrm{T}}(n)\mathbf{x}'(n)\mathbf{x}'^{\mathrm{T}}(n)\right],$$

we obtain (6.39).

7

Transform Domain Adaptive Filters

In the previous chapter we noted that the convergence behaviour of the LMS algorithm depends on the eigenvalues of the correlation matrix, **R**, of the adaptive filter input process. Furthermore, we also saw that the eigenvalues of **R** are directly related to the power spectral density of the underlying process. Hence, we may say that the convergence behaviour of the LMS algorithm is frequency dependent in the sense that for an adaptive filter with the transfer function $W(e^{j\omega})$, the rate of convergence of $W(e^{j\omega})$ toward its optimum value, $W_o(e^{j\omega})$, at a given frequency $\omega = \omega_o$, depends on the relative value of the power spectral density of the underlying input signal at $\omega = \omega_o$, i.e. $\Phi_{xx}(e^{j\omega_o})$. A large value of $\Phi_{xx}(e^{j\omega_o})$ (relative to the values of $\Phi_{xx}(e^{j\omega})$ at other frequencies) indicates that the adaptive filter is well excited at $\omega = \omega_o$. This results in fast convergence around $\omega = \omega_o$. On the other hand, the LMS algorithm converges very slowly over those frequency bands in which the adaptive filter is poorly excited. This concept, which is intuitively understandable, may also be confirmed through computer simulations. (See simulation exercise P7.13 at the end of this chapter.)

A solution that we might intuitively consider for solving the above mentioned problem of slow convergence of the LMS algorithm may be to employ a set of bandpass filters to partition the adaptive filter input into a few subbands and use a normalization process to equalize the energy content in each of the subbands. The equalized subband signals can then be used for adaptation of the filter tap weights. The content of this chapter is an elaboration of this principle for developing adaptive algorithms with better convergence behaviour than the conventional LMS algorithm.

In this chapter we present an adaptive filtering scheme that uses an orthogonal transform for partitioning the filter input into subbands. This is called the transform domain adaptive filter (TDAF), for obvious reasons. We present a thorough study of the TDAF which includes not only its convergence behaviour but also its efficient implementation.

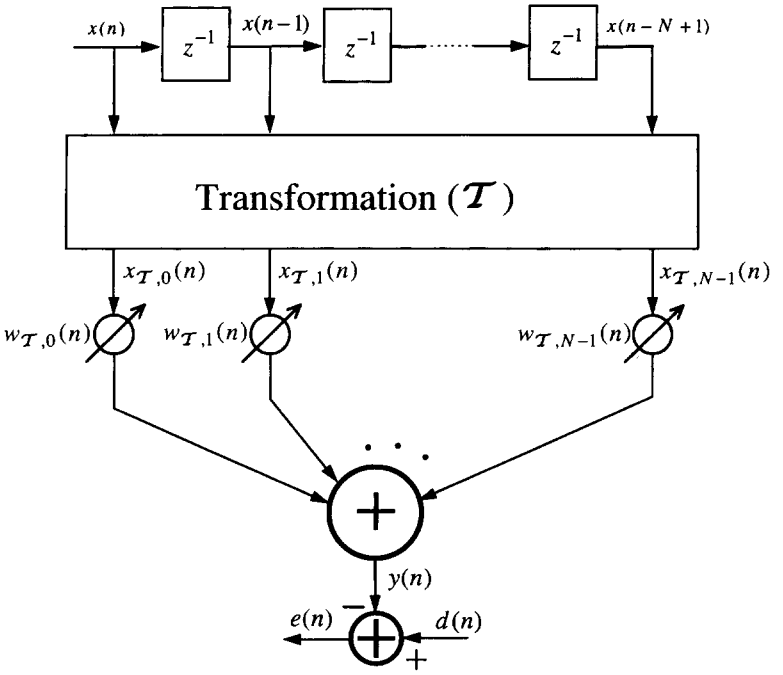


Figure 7.1 Transform domain adaptive filter

7.1 Overview of Transform Domain Adaptive Filters

Figure 7.1 depicts a block schematic of a TDAF.[1] The set of input samples $x(n), x(n-1), \ldots, x(n-N+1)$ to the filter are transformed to a new set of samples, $x_{T,0}(n), x_{T,1}(n), \ldots, x_{T,N-1}(n)$, through an orthogonal transform ($\mathcal{T}$), prior to the filtering process. The tap weights $w_{T,0}, w_{T,1}, \ldots, w_{T,N-1}$ are optimized so that the output error, $e(n)$, is minimized in the mean-square sense.

The orthogonal transform ($\mathcal{T}$) is implemented according to the equation

$$\mathbf{x}_T(n) = \mathcal{T}\mathbf{x}(n), \tag{7.1}$$

where $\mathbf{x}(n) = [x(n)\ \ x(n-1)\ \ \ldots\ \ x(n-N+1)]^{\mathrm{T}}$ is the filter tap-input vector in the time domain, $\mathbf{x}_T(n) = [x_{T,0}(n)\ \ x_{T,1}(n)\ \ \ldots\ \ x_{T,N-1}(n)]^{\mathrm{T}}$ is the filter tap-input vector in the transform domain, and $\mathcal{T}$ is the transformation matrix which is selected to be a unitary matrix,[2] i.e.

$$\mathcal{T}^{\mathrm{T}}\mathcal{T} = \mathcal{T}\mathcal{T}^{\mathrm{T}} = \mathbf{I}. \tag{7.2}$$

[1] The TDAF was first proposed by Narayan and Peterson (1981) and Narayan, Peterson and Narasimha (1983).

[2] Throughout this chapter, the symbol $\mathcal{T}$ will be used to represent a unitary matrix satisfying (7.2).

Here, we assume that the elements of $\mathcal{T}$ are real-valued. When the elements of $\mathcal{T}$ are complex-valued, the superscript T in (7.2), indicating transposition, has to be replaced by H, i.e. Hermitian transposition.

The filter output is obtained according to the equation

$$y(n) = \mathbf{w}_T^{\mathrm{T}} \mathbf{x}_T(n), \tag{7.3}$$

where $\mathbf{w}_T = [w_{T,0}\ w_{T,1}\ \ldots\ w_{T,N-1}]^{\mathrm{T}}$. We may note that although $\mathbf{x}_T(n)$ is in the transform domain, the filter output, $y(n)$, is in the time domain. The estimation error

$$e(n) = d(n) - y(n) \tag{7.4}$$

is also in the time domain.

The cost function used to optimize the filter tap weights is

$$\xi = \mathrm{E}[e^2(n)]. \tag{7.5}$$

Substituting (7.3) and (7.4) in (7.5) we obtain

$$\xi = \mathbf{w}_T^{\mathrm{T}} \mathbf{R}_T \mathbf{w}_T - 2\mathbf{w}_T^{\mathrm{T}} \mathbf{p}_T + \mathrm{E}[d^2(n)], \tag{7.6}$$

where $\mathbf{R}_T = \mathrm{E}[\mathbf{x}_T(n)\mathbf{x}_T^{\mathrm{T}}(n)]$ and $\mathbf{p}_T = \mathrm{E}[d(n)\mathbf{x}_T(n)]$. Setting the gradient of ξ with respect to $\mathbf{w}_T$ equal to zero, we obtain the corresponding Wiener–Hopf equation the solution of which gives the optimum tap-weight vector of the TDAF as

$$\mathbf{w}_{T,o} = \mathbf{R}_T^{-1} \mathbf{p}_T. \tag{7.7}$$

Substituting this result in (7.6), the minimum mean-square error (MSE) of the TDAF is obtained as

$$\xi_{\min} = \mathrm{E}[d^2(n)] - \mathbf{p}_T^{\mathrm{T}} \mathbf{R}_T^{-1} \mathbf{p}_T. \tag{7.8}$$

To compare this with the minimum MSE associated with the conventional transversal structure case (i.e. without the orthogonal transformation, $\mathcal{T}$), we note that

$$\begin{aligned} \mathbf{R}_T &= \mathrm{E}[\mathbf{x}_T(n)\mathbf{x}_T^{\mathrm{T}}(n)] \\ &= \mathcal{T}\mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]\mathcal{T}^{\mathrm{T}} \\ &= \mathcal{T}\mathbf{R}\mathcal{T}^{\mathrm{T}} \end{aligned} \tag{7.9}$$

and

$$\begin{aligned} \mathbf{p}_T &= \mathrm{E}[d(n)\mathbf{x}_T(n)] \\ &= \mathcal{T}\mathrm{E}[d(n)\mathbf{x}(n)] \\ &= \mathcal{T}\mathbf{p}. \end{aligned} \tag{7.10}$$

Substituting (7.9) and (7.10) in (7.8) and using (7.2), after some straightforward manipulations, we get

$$\xi_{\min} = \mathrm{E}[d^2(n)] - \mathbf{p}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{p}. \tag{7.11}$$

Comparing this result with (3.27) we find that the minimum MSE associated with a conventional transversal filter and its corresponding TDAF are the same. This could also be understood, intuitively, if we note that the transformation $\mathbf{x}_{\mathcal{T}}(n) = \mathcal{T}\mathbf{x}(n)$ is reversible (i.e. $\mathbf{x}(n) = \mathcal{T}^{\mathrm{T}}\mathbf{x}_{\mathcal{T}}(n)$) and, thus, any output $y(n) = \mathbf{w}^{\mathrm{T}}\mathbf{x}(n)$ can also be obtained from $\mathbf{x}_{\mathcal{T}}(n)$ by using an appropriate tap-weight vector $\mathbf{w}_{\mathcal{T}}$. To find the relationship between $\mathbf{w}$ and $\mathbf{w}_{\mathcal{T}}$, we simply let $\mathbf{w}^{\mathrm{T}}\mathbf{x}(n) = \mathbf{w}_{\mathcal{T}}^{\mathrm{T}}\mathbf{x}_{\mathcal{T}}(n)$ and use (7.1), to obtain

$$\mathbf{w}_{\mathcal{T}} = \mathcal{T}\mathbf{w}. \tag{7.12}$$

Before going into further details of transform domain adaptive filters, in the next two sections we study a very specific feature of orthogonal transforms which makes them suitable for adaptive filtering algorithms.

7.2 The Band-Partitioning Property of Orthogonal Transforms

We explore the DCT (discrete cosine transform) as an example of orthogonal transforms. The DCT of a sequence $\{x(n), x(n-1), \ldots, x(n-N+1)\}$ is defined as

$$x_{\mathrm{DCT},k}(n) = \sum_{l=0}^{N-1} c_{kl}x(n-l), \quad \text{for} \quad k = 0, 1, \ldots, N-1 \tag{7.13}$$

where

$$c_{kl} = \begin{cases} \dfrac{1}{\sqrt{N}}, & k = 0 \quad \text{and} \quad l = 0, 1, \ldots, N-1, \\ \sqrt{\dfrac{2}{N}}\cos\dfrac{\pi(2l+1)k}{2N}, & k = 1, 2, \ldots, N-1 \\ & \text{and} \quad l = 0, 1, \ldots, N-1 \end{cases} \tag{7.14}$$

are the DCT coefficients. It is also worth noting that (7.13) may be written as

$$\mathbf{x}_{\mathrm{DCT}}(n) = \mathcal{T}_{\mathrm{DCT}}\mathbf{x}(n), \tag{7.15}$$

where $\mathcal{T}_{\mathrm{DCT}}$ is the $N \times N$ DCT matrix. The klth element of $\mathcal{T}_{\mathrm{DCT}}$ is c_{kl}, as defined in (7.14) and

$$\mathbf{x}_{\mathrm{DCT}}(n) = [x_{\mathrm{DCT},0}(n) \;\; x_{\mathrm{DCT},1}(n) \;\; \ldots \;\; x_{\mathrm{DCT},N-1}(n)]^{\mathrm{T}}.$$

Besides being a linear transformation, the process defined by (7.13) (or (7.15)) may also be viewed as an implementation of a bank of finite impulse response (FIR) filters

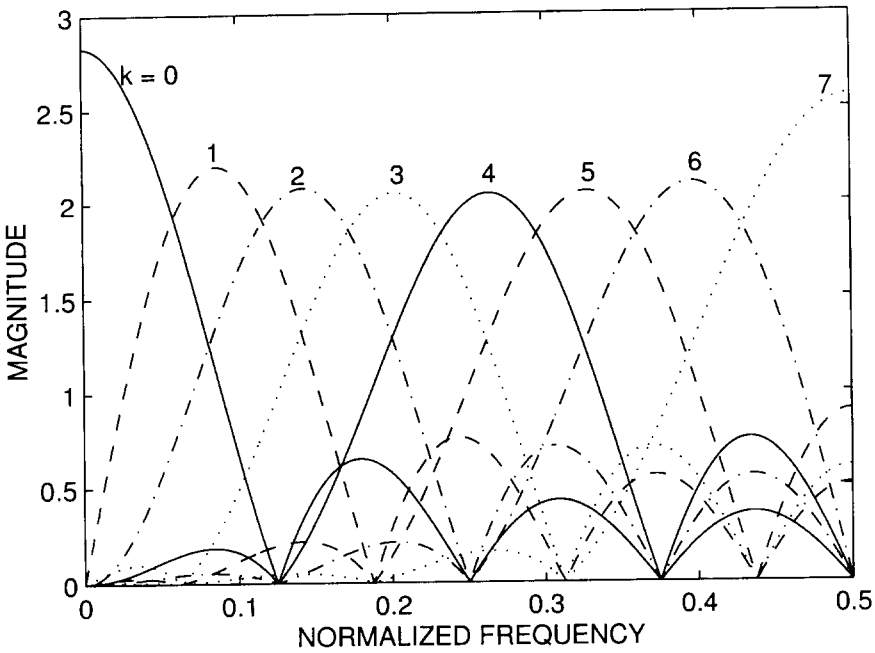


Figure 7.2 The magnitude responses of the DCT filters, for $N = 8$

whose coefficients are the c_{kl}s. Here, these are referred to as DCT filters. The transfer function of the kth DCT filter is

$$C_k(z) = \sum_{l=0}^{N-1} c_{kl} z^{-l}. \tag{7.16}$$

Figure 7.2 shows the magnitude responses of the DCT filters when $N = 8$. The plots clearly show the band-partitioning property of the DCT filters. Each response has a large main lobe which may be identified as its passband, and a number of side lobes which correspond to its stop-band. Similar plots (with some variations in the shapes) are also obtained for other commonly used orthogonal transforms, e.g. DFT.

Before we elaborate on how or why this band-partitioning property of orthogonal transforms is important to us, we look at this property from a different angle in the next section.

7.3 The Orthogonalization Property of Orthogonal Transforms

The band-partitioning property of orthogonal transforms gives a frequency domain view of them. The dual of this in the time domain is the orthogonalization property of such transforms. This property can be deduced intuitively from the band-partitioning property observed in Section 7.2. We recall that processes with mutually exclusive

spectral bands are uncorrelated with one another (Papoulis, 1991). On the other hand, from the band-partitioning property, we note that the elements of the transformed tap-input vector, $\mathbf{x}_T(n)$, constitute a set of random processes with approximately mutually exclusive spectral bands. This implies that the elements of $\mathbf{x}_T(n)$ are (at least) approximately uncorrelated with one another. This, in turn, implies that the correlation matrix $\mathbf{R}_T = \mathrm{E}[\mathbf{x}_T(n)\mathbf{x}_T^{\mathrm{T}}(n)]$ is closer to a diagonal matrix than $\mathbf{R}$ is. An appropriate normalization can convert $\mathbf{R}_T$ to a normalized matrix $\mathbf{R}_T^N$ whose eigenvalue spread will be much smaller than that of $\mathbf{R}$, thereby improving the convergence behaviour of the LMS algorithm in the transform domain. This can be best explained though a numerical example.

Consider the case where $x(n)$ is a first order autoregressive process generated by passing a white noise process through the system function[3]

$$H(z) = \frac{\sqrt{1-\alpha^2}}{1-\alpha z^{-1}}, \tag{7.17}$$

where α is a constant in the range -1 to $+1$. For $\alpha = 0.9$, we obtain

$$\mathbf{R} = \begin{bmatrix} 1.0000 & 0.9000 & 0.8100 & 0.7290 \\ 0.9000 & 1.0000 & 0.9000 & 0.8100 \\ 0.8100 & 0.9000 & 1.0000 & 0.9000 \\ 0.7290 & 0.8100 & 0.9000 & 1.0000 \end{bmatrix}. \tag{7.18}$$

For a derivation of $\mathbf{R}$, see Example 4.1. Using the DCT as the transformation, we get

$$\mathbf{R}_T = \begin{bmatrix} 3.5245 & 0.0000 & -0.0855 & 0.0000 \\ 0.0000 & 0.3096 & 0.0000 & -0.0032 \\ -0.0855 & 0.0000 & 0.1045 & 0.0000 \\ 0.0000 & -0.0032 & 0.0000 & 0.0614 \end{bmatrix}. \tag{7.19}$$

This clearly is much closer to the diagonal (i.e. its off-diagonal elements are relatively closer to zero) when compared with $\mathbf{R}$.

The normalization performed in the implementation of the LMS algorithm in the transform domain (as we shall see later), in effect, is equivalent to normalization of the elements of $\mathbf{x}_T(n)$ to the power of unity. This is done by premultiplying $\mathbf{x}_T(n)$ with a diagonal matrix, $\mathbf{D}^{-1/2}$, prior to the filtering and adaptation process, where $\mathbf{D}^{-1/2}$ is the inverse of the square root of the diagonal matrix

$$\mathbf{D} = \begin{bmatrix} \mathrm{E}[x_{T,0}^2(n)] & 0 & \cdots & 0 \\ 0 & \mathrm{E}[x_{T,1}^2(n)] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathrm{E}[x_{T,N-1}^2(n)] \end{bmatrix}. \tag{7.20}$$

[3] The reader may recall that we used the same process in many examples in previous chapters.

Thus, we get

$$\mathbf{x}_T^{\mathcal{N}}(n) = \mathbf{D}^{-1/2}\mathbf{x}_T(n), \tag{7.21}$$

where $\mathbf{x}_T^{\mathcal{N}}(n)$ is the normalized tap-input vector. The correlation matrix associated with $\mathbf{x}_T^{\mathcal{N}}(n)$ is

$$\mathbf{R}_T^{\mathcal{N}} = \mathbf{D}^{-1/2}\mathbf{R}_T\mathbf{D}^{-1/2}. \tag{7.22}$$

Furthermore, we note that

$$\mathbf{D} = \text{diag}[\mathbf{R}_T], \tag{7.23}$$

where $\text{diag}[\mathbf{R}_T]$ denotes the diagonal matrix consisting of the diagonal elements of $\mathbf{R}_T$. The reader may easily verify that the mean-square values of the elements of $\mathbf{x}_T^{\mathcal{N}}(n)$ as well as the diagonal elements of $\mathbf{R}_T^{\mathcal{N}}$ are all equal to unity as a result of this normalization.

For the above example, we get

$$\mathbf{D}^{-1/2} = \begin{bmatrix} 0.5327 & 0 & 0 & 0 \\ 0 & 1.7972 & 0 & 0 \\ 0 & 0 & 3.0934 & 0 \\ 0 & 0 & 0 & 4.0357 \end{bmatrix} \tag{7.24}$$

and

$$\mathbf{R}_T^{\mathcal{N}} = \begin{bmatrix} 1.0000 & 0.0000 & -0.1409 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 & -0.0231 \\ -0.1409 & 0.0000 & 1.0000 & 0.0000 \\ 0.0000 & -0.0231 & 0.0000 & 1.0000 \end{bmatrix}. \tag{7.25}$$

To compare the performance of the conventional LMS algorithm (in the time domain) with its associated implementation in a transform domain (explained in the following section), the eigenvalue spreads of $\mathbf{R}$ and $\mathbf{R}_T^{\mathcal{N}}$ have to be examined. For the above example, we obtain

$$\text{eigenvalue spread of } \mathbf{R} = 57.5$$

and

$$\text{eigenvalue spread of } \mathbf{R}_T^{\mathcal{N}} = 1.33.$$

For the present example, these results predict a much superior performance of the LMS algorithm in the transform domain as compared with its conventional implementation in the time domain. This, clearly, is a direct consequence of the orthogonalization property of the DCT, as was demonstrated above. This argument justifies the

application of the orthogonal transforms for improving the performance of the LMS algorithm.

In our study in this chapter we find that for a given transform the degree of improvement achieved by replacing the conventional LMS algorithm with its transform domain counterpart depends on the power spectral density of the underlying input process. We emphasize the band-partitioning property of orthogonal transforms and present some theoretical results that explain this phenomenon. We also find that a rough estimate of the power spectral density of the underlying input process is sufficient for the purpose of selecting an appropriate transform.

7.4 The Transform Domain LMS Algorithm

In the implementation of the transform domain LMS (TDLMS) algorithm the filter tap weights are updated according to the following recursion:

$$\mathbf{w}_T(n+1) = \mathbf{w}_T(n) + 2\mu\hat{\mathbf{D}}^{-1}e(n)\mathbf{x}_T(n), \tag{7.26}$$

where $\hat{\mathbf{D}}$ is an estimate of the diagonal matrix $\mathbf{D}$ defined in the last section. This vector recursion can be decomposed into the following N scalar recursions:

$$w_{T,i}(n+1) = w_{T,i}(n) + 2\frac{\mu}{\hat{\sigma}^2_{x_{T,i}}(n)}e(n)x_{T,i}(n), \quad i = 0, 1, \ldots, N-1 \tag{7.27}$$

where $\hat{\sigma}^2_{x_{T,i}}(n)$ is an estimate of $\mathrm{E}[x^2_{T,i}(n)]$. This shows that the presence of $\hat{\mathbf{D}}^{-1}$ in (7.26) is equivalent to using different step-size parameters at various taps of the TDAF. Each step-size parameter is chosen proportional to the inverse of the power of its associated input signal. Noting this, we refer to (7.26) as a step-normalized LMS recursion. In the present literature, the term normalized LMS algorithm has often been used to refer to (7.26) (Marshall, Jenkins and Murphy, 1989, and Narayan, Peterson and Narasimha, 1983, for example). In this book we use the term step-normalized (when necessary) to prevent any confusion between the normalization applied to TDAFs and the normalized LMS algorithm that was introduced in the previous chapter (Section 6.6).

In the implementation of (7.26) we need to obtain the estimates of the signal powers at various taps of the filter, i.e. the $\hat{\sigma}^2_{x_{T,i}}(n)$s. The following recursions are usually used for this purpose:

$$\hat{\sigma}^2_{x_{T,i}}(n) = \beta\hat{\sigma}^2_{x_{T,i}}(n-1) + (1-\beta)x^2_{T,i}(n), \quad i = 0, 1, \ldots, N-1, \tag{7.28}$$

where β is a positive constant close to but less than one. This recursion estimates the power by calculating a weighted average of the present and past samples of the $x^2_{T,i}(n)$s using an exponential weighting function given by $1, \beta, \beta^2, \ldots$ (see Problem P7.2). The TDLMS algorithm, including this signal power estimation, is summarized in Table 7.1.

The step-normalization, as applied in (7.26), is equivalent to the normalization of the elements of the transformed tap-input vector, $\mathbf{x}_T(n)$, to the power of unity. To show this we multiply (7.26) on both sides by $\hat{\mathbf{D}}^{1/2}$ (the diagonal matrix consisting of the square roots of the diagonal elements of $\hat{\mathbf{D}}$) and define $\mathbf{w}^N_T(n) = \hat{\mathbf{D}}^{1/2}\mathbf{w}_T(n)$ and

Table 7.1 Summary of the TDMS algorithm

Input:	Tap-weight vector, $\mathbf{w}_T(n)$, Input vector, $\mathbf{x}(n)$, Past tap-input power estimates, $\hat{\sigma}^2_{x_{T,i}}(n-1)$, and desired output, $d(n)$.
Output:	Filter output, $y(n)$, Tap-input power estimate updates, $\hat{\sigma}^2_{x_{T,i}}(n)$, Tap-weight vector update, $\mathbf{w}_T(n+1)$.

1. Transformation:
$$\mathbf{x}_T = \mathcal{T}\mathbf{x}(n)$$
1. Filtering:
$$y(n) = \mathbf{w}_T^{\mathrm{T}}(n)\mathbf{x}_T(n)$$
2. Error estimation:
$$e(n) = d(n) - y(n)$$
3. Tap-input power estimate update:
for $i = 0$ to $N-1$
$$\hat{\sigma}^2_{x_{T,i}}(n) = \beta\hat{\sigma}^2_{x_{T,i}}(n-1) + (1-\beta)x^2_{T,i}(n)$$
4. Tap-weight vector adaptation:
$$\mathbf{w}_T(n+1) = \mathbf{w}_T(n) + 2\mu\hat{\mathbf{D}}^{-1}e(n)\mathbf{x}_T(n),$$
where $\hat{\mathbf{D}} = \mathrm{diag}[\hat{\sigma}^2_{x_{T,0}}(n), \hat{\sigma}^2_{x_{T,1}}(n), \ldots, \hat{\sigma}^2_{x_{T,N-1}}(n)]$.

$\mathbf{x}_T^{\mathcal{N}}(n) = \hat{\mathbf{D}}^{-1/2}\mathbf{x}_T(n)$, to obtain

$$\mathbf{w}_T^{\mathcal{N}}(n+1) = \mathbf{w}_T^{\mathcal{N}}(n) + 2\mu e(n)\mathbf{x}_T^{\mathcal{N}}(n). \tag{7.29}$$

We may also note that

$$\begin{aligned} e(n) &= d(n) - y(n) \\ &= d(n) - \mathbf{w}_T^T(n)\mathbf{x}_T(n) \\ &= d(n) - \mathbf{w}_T^{\mathcal{N}T}(n)\mathbf{x}_T^{\mathcal{N}}(n). \end{aligned} \tag{7.30}$$

Equations (7.29) and (7.30) suggest that *the TDLMS algorithm, in effect, is equivalent to a conventional LMS algorithm with the normalized tap-input vector* $\mathbf{x}_T^{\mathcal{N}}(n)$.

The significance of this result is that since (7.29) is a conventional LMS recursion, the analytical results of the previous chapter can immediately be applied to evaluate the performance of the TDLMS algorithm. In particular, we note that the various modes of convergence of the TDLMS algorithm are determined by the eigenvalues of the correlation matrix $\mathbf{R}_T^{\mathcal{N}} = \mathrm{E}[\mathbf{x}_T^{\mathcal{N}}(n)\mathbf{x}_T^{\mathcal{N}T}(n)]$. This matches our conjecture in the previous section. Also, by substituting the eigenvalues of $\mathbf{R}_T^{\mathcal{N}}$ for $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$, in (6.60), (6.62) or (6.63), the misadjustment of the TDLMS algorithm can be evaluated. In particular, we note that $\mathrm{tr}[\mathbf{R}_T^{\mathcal{N}}] = N$, since the diagonal elements of $\mathbf{R}_T^{\mathcal{N}}$ are all normalized to unity. Thus, using (6.63), the misadjustment of the TDLMS algorithm is obtained as

$$\mathcal{M} \approx \mu N. \tag{7.31}$$

7.5 The Ideal LMS–Newton Algorithm and its Relationship with TDLMS

In Chapter 5 we introduced two search methods: the method of steepest-descent and Newton's algorithm. The LMS algorithm was introduced in Chapter 6 as a stochastic implementation of the method of steepest-descent. In the LMS algorithm, the gradient vector $\nabla_{\mathbf{w}}\xi$ is replaced by its instantaneous estimate, $\nabla_{\mathbf{w}}e^2(n)$. A imilar substitution of the gradient vector in the Newton's method given by (5.44) results in the recursion

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu\mathbf{R}^{-1}e(n)\mathbf{x}(n), \tag{7.32}$$

which may be called the *ideal LMS–Newton algorithm*. The term *ideal* refers to the fact that knowledge of the true $\mathbf{R}^{-1}$ is assumed here. In actual practice, of course, this cannot be true. We can only obtain an estimate of $\mathbf{R}^{-1}$. Methods for obtaining such estimates are available in the literature (see Widrow and Stearns, 1985, Marshall and Jenkins, 1992, and Farhang-Boroujeny, 1993, for example). Our aim in this section is to show that there is a close relationship between the LMS–Newton and the TDLMS algorithms. We show that when the transformation matrix $\mathcal{T}$ is selected to be the Karhunen Loéve transform (KLT) of the filter input, the TDLMS and LMS–Newton are two different formulations of the same algorithm. Thus, we conclude that when a proper transformation is used, the TDLMS algorithm may be considered as an efficient implementation of the LMS–Newton algorithm.

We recall that the correlation matrix $\mathbf{R}$ can be decomposed as $\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{T}}$, where $\mathbf{Q}$ is the $N \times N$ matrix whose columns are the eigenvectors of $\mathbf{R}$ and $\mathbf{\Lambda}$ is the diagonal matrix consisting of the associated eigenvalues of $\mathbf{R}$. This, in turn, implies that

$$\mathbf{R}^{-1} = \mathbf{Q}\mathbf{\Lambda}^{-1}\mathbf{Q}^{\mathrm{T}} \tag{7.33}$$

since $\mathbf{Q}\mathbf{Q}^{\mathrm{T}} = \mathbf{I}$.

Substituting (7.33) and (7.32) and premultiplying the result by $\mathbf{Q}^{\mathrm{T}}$, we obtain

$$\mathbf{w}'(n+1) = \mathbf{w}'(n) + 2\mu\mathbf{\Lambda}^{-1}e(n)\mathbf{x}'(n), \tag{7.34}$$

where $\mathbf{w}'(n) = \mathbf{Q}^{\mathrm{T}}\mathbf{w}(n)$ and $\mathbf{x}'(n) = \mathbf{Q}^{\mathrm{T}}\mathbf{x}(n)$. Also, from our discussions in Chapter 4, we recall that the eigenvalues of $\mathbf{R}$, i.e. the diagonal elements of $\mathbf{\Lambda}$, are equal to the powers (mean-square values) of the elements of the vector $\mathbf{x}'(n)$. Combining this with our discussion in the previous section, we see that the LMS–Newton algorithm is an alternative formulation of the TDLMS algorithm when $\mathcal{T} = \mathbf{Q}^{\mathrm{T}}$. Furthermore, we note that for a given input process, $x(n)$, with correlation matrix $\mathbf{R}$, the transform $\mathcal{T} = \mathbf{Q}^{\mathrm{T}}$ is the ideal one in the sense that it results in a diagonal $\mathbf{R}_{\mathcal{T}} = \mathbf{\Lambda}$. When $\mathcal{T} \neq \mathbf{Q}^{\mathrm{T}}$, but results in an approximately diagonal $\mathbf{R}_{\mathcal{T}}$, we may say that the TDLMS algorithm is equivalent to a *quasi* LMS–Newton algorithm.

7.6 Selection of the Transform $\mathcal{T}$

It turns out that for a given process, $x(n)$, the performance of the TDLMS algorithm may vary significantly depending on the selection of the transformation matrix, $\mathcal{T}$. A

transform which may perform well for a given input process may perform poorly once the statistics of the input change. This happens to be more prominent when the filter length is short. For long filters, we find that most of the commonly used transforms perform well and result in a significant performance improvement as compared with the conventional LMS algorithm.

In this section we present some theoretical results that explain these observations. This presentation includes a geometrical interpretation of the TDLMS process which will be given for a two-tap filter. This interpretation will then be generalized by using a special performance index which is also introduced in this section. This leads to a very instructive view of the band-partitioning property of orthogonal transforms which helps us to select a proper transform once a rough estimate of the power spectral density of the underlying input process is known.

7.6.1 A geometrical interpretation[4]

We recall that the performance surface of a transversal filter with input correlation matrix $\mathbf{R}$ may be written as

$$\xi(\mathbf{v}) = \xi_{\min} + \mathbf{v}^{\mathrm{T}}\mathbf{R}\mathbf{v}, \tag{7.35}$$

where the vector $\mathbf{v}$ is the difference between the filter tap-weight vector, $\mathbf{w}$, and its optimum value, $\mathbf{w}_o$.

For the sake of illustrating the principles, let us consider a two-tap filter problem with $\mathbf{R}$ given by

$$\mathbf{R} = \begin{bmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{bmatrix}. \tag{7.36}$$

With this $\mathbf{R}$, equation (7.35) becomes

$$\xi(v_0, v_1) = \xi_{\min} + v_0^2 + v_1^2 + 1.8v_0v_1. \tag{7.37}$$

Figure 7.3(a) shows the contour plot associated with this performance surface. As we may recall from our discussions in the previous chapters, the eccentricity of the contour ellipses in Figure 7.3(a), is related to the eigenvalue spread of the correlation matrix $\mathbf{R}$. A large eccentricity is due to a large eigenvalue spread and that, in turn, results in certain slow mode(s) of convergence when the conventional LMS algorithm is used to adjust the filter tap weights.

Application of an orthogonal transform, $\mathcal{T}$, converts the tap-input vector $\mathbf{x}(n)$ to $\mathbf{x}_{\mathcal{T}}(n) = \mathcal{T}\mathbf{x}(n)$, whose associated correlation matrix, $\mathbf{R}_{\mathcal{T}}$, is related to $\mathbf{R}$ according to (7.9). As a numerical example, let us choose

$$\mathcal{T} = \begin{bmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{bmatrix}. \tag{7.38}$$

[4] The geometrical interpretation presented here has been adopted from Marshall, Jenkins and Murphy (1989).

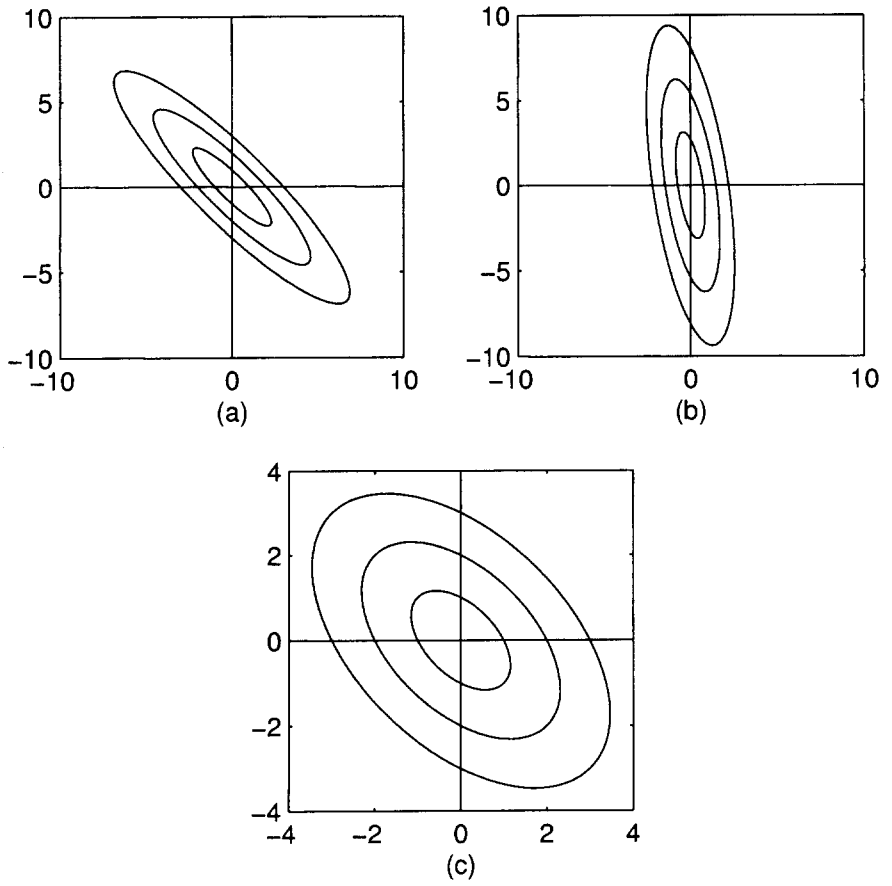


Figure 7.3 A geometrical interpretation of the TDLMS algorithm. (a) The performance surface before transformation. (b) The performance surface after transformation, but without normalization. (c) The performance surface after transformation and normalization (adopted from Marshall, Jenkins and Murphy, 1989)

This, with $\mathbf{R}$ as given in (7.36), results in

$$\mathbf{R}_T = \begin{bmatrix} 1.864 & 0.252 \\ 0.252 & 0.136 \end{bmatrix}. \tag{7.39}$$

If no normalization is applied to the transformed samples, then the performance surface associated with the TDAF will be

$$\xi_T(\mathbf{v}_T) = \xi_{\min} + \mathbf{v}_T^{\mathrm{T}} \mathbf{R}_T \mathbf{v}_T. \tag{7.40}$$

which for the present numerical example can be expanded as

$$\xi_T(v_{T,0}, v_{T,1}) = \xi_{\min} + 1.864 v_{T,0}^2 + 0.136 v_{T,1}^2 + 0.504 v_{T,0} v_{T,1}. \tag{7.41}$$

Figure 7.3(b) shows the contours associated with the performance surface defined by (7.41). Note that the effect of the transformation is only to rotate the performance surface with respect to the coordinate axes. The shape of the performance surface, i.e. the eccentricity of the contour ellipses, has not changed. This can be explained mathematically by noting that, since $\mathcal{T}\mathcal{T}^{\mathrm{T}} = \mathcal{T}^{\mathrm{T}}\mathcal{T} = \mathbf{I}$,

$$\begin{aligned}\xi(\mathbf{v}) &= \xi_{\min} + \mathbf{v}^{\mathrm{T}}\mathbf{R}\mathbf{v} \\ &= \xi_{\min} + \mathbf{v}^{\mathrm{T}}\mathcal{T}^{\mathrm{T}}\mathcal{T}\mathbf{R}\mathcal{T}^{\mathrm{T}}\mathcal{T}\mathbf{v} \\ &= \xi_{\min} + \mathbf{v}_T^{\mathrm{T}}\mathbf{R}_T\mathbf{v}_T = \xi_T(\mathbf{v}_T),\end{aligned} \tag{7.42}$$

where $\mathbf{v}$ and $\mathbf{v}_T$ are related according to the equation $\mathbf{v}_T = \mathcal{T}\mathbf{v}$. This result, which can also be written as $\xi_T(\mathbf{v}_T) = \xi(\mathcal{T}^{\mathrm{T}}\mathbf{v}_T)$, shows that the performance surface defined by (7.40) is obtained from the one defined by (7.35), by a rotation of the coordinate axes according to the relationship $\mathbf{v}_T = \mathcal{T}\mathbf{v}$, or, equivalently, by keeping the coordinate axes fixed and rotating the performance surface in the opposite direction. This observation shows that *transformation without normalization has no effect on the convergence behaviour of the steepest-descent method and, thus, the LMS algorithm.* Thus, we emphasize that normalization has to be considered as an integrated part of any

transform domain adaptive algorithm (as introduced in the case of the TDLMS algorithm in Section 7.4), otherwise transformation adds up to the filter complexity without any gain in convergence.

When the elements of $\mathbf{x}_T(n)$ are normalized to the power of unity,[5] the corresponding correlation matrix is given by (7.22) and its associated performance surface is defined as

$$\xi_T^{\mathcal{N}}(\mathbf{v}_T^{\mathcal{N}}) = \xi_{\min} + \mathbf{v}_T^{\mathcal{N}T}\mathbf{R}_T^{\mathcal{N}}\mathbf{v}_T^{\mathcal{N}}. \tag{7.43}$$

For the present example, we obtain

intersection points of each contour (hyper-ellipse) with the v-axes are at equal distance from the origin. This, which is clearly observed in Figure 7.3(a), can be shown to be true, in general, if we note that, for any i

$$\xi_i(v_i) = \xi_{\min} + r_{ii}v_i^2,$$

where r_{ii} is the ith diagonal element of $\mathbf{R}$ and $\xi_i(v_i)$ is the performance function $\xi(\mathbf{v})$ when all elements of the vector $\mathbf{v}$, except its ith element, v_i, have been set equal to zero. We also note that for a transversal filter, r_{ii} is the same for all values of i. Thus, the identity

$$\xi_i(v_i) = \xi_j(v_j), \quad \text{for all } i \text{ and } j,$$

implies that

$$|v_i| = |v_j|$$

which, in turn, shows that the hyper-ellipses associated with the performance surface of a transversal filter are hyper-spherical at the points of their intersection with the v-axes.

Following the same argument, we find that since the diagonal elements of $\mathbf{R}_{\mathcal{T}}$ are likely to be unequal (unless the underlying input process, $x(n)$, is white, i.e. when $\mathbf{R}$ is a multiple of the identity matrix), the contour ellipses associated with $\xi_{\mathcal{T}}(\mathbf{v}_{\mathcal{T}})$ are most likely non-hyper-spherical at the points of their intersection with the $v_{\mathcal{T}}$-axes. This is clearly observed in Figure 7.3(b).

On the other hand, normalization of the transformed samples to the power of unity equalizes the diagonal elements of $\mathbf{R}_{\mathcal{T}}^{\mathcal{N}}$. Thus, the hyper-ellipses associated with the performance surface $\xi_{\mathcal{T}}^{\mathcal{N}}(\mathbf{v}_{\mathcal{T}}^{\mathcal{N}})$ are hyper-spherical at the points of their intersection with the corresponding coordinate axes. To get a better insight, we may also note that

$$\begin{aligned}
\xi_{\mathcal{T}}(\mathbf{v}_{\mathcal{T}}) &= \xi_{\min} + \mathbf{v}_{\mathcal{T}}^{\mathrm{T}}\mathbf{R}_{\mathcal{T}}\mathbf{v}_{\mathcal{T}} \\
&= \xi_{\min} + \mathbf{v}_{\mathcal{T}}^{\mathrm{T}}\mathbf{D}^{1/2}\mathbf{D}^{-1/2}\mathbf{R}_{\mathcal{T}}\mathbf{D}^{-1/2}\mathbf{D}^{1/2}\mathbf{v}_{\mathcal{T}} \\
&= \xi_{\min} + (\mathbf{D}^{1/2}\mathbf{v}_{\mathcal{T}})^{\mathrm{T}}\mathbf{R}_{\mathcal{T}}^{\mathcal{N}}(\mathbf{D}^{1/2}\mathbf{v}_{\mathcal{T}}) = \xi_{\mathcal{T}}^{\mathcal{N}}(\mathbf{v}_{\mathcal{T}}^{\mathcal{N}}),
\end{aligned} \tag{7.46}$$

where $\mathbf{v}_{\mathcal{T}}^{\mathcal{N}} = \mathbf{D}^{1/2}\mathbf{v}_{\mathcal{T}}$, and we have noted that $(\mathbf{D}^{1/2})^{\mathrm{T}} = \mathbf{D}^{1/2}$, since $\mathbf{D}$ is a diagonal matrix. This result, which can also be written as $\xi_{\mathcal{T}}^{\mathcal{N}}(\mathbf{v}_{\mathcal{T}}^{\mathcal{N}}) = \xi_{\mathcal{T}}(\mathbf{D}^{-1/2}\mathbf{v}_{\mathcal{T}}^{\mathcal{N}})$, shows that the performance surface defined by (7.43) is obtained from the one defined by (7.40) by scaling its coordinate axes according to the relationship $\mathbf{v}_{\mathcal{T}}^{\mathcal{N}} = \mathbf{D}^{1/2}\mathbf{v}_{\mathcal{T}}$. For the example shown in Figure 7.3, this is equivalent to stretching the contour ellipses of Figure 7.3(b) along the $v_{\mathcal{T},0}$-axis and shrinking them along the $v_{\mathcal{T},1}$-axis. This clearly reduces the eccentricity of the ellipses. Furthermore, we note that the ellipses in Figure 7.3(c) would become circles, resulting in maximum improvement in convergence if the ellipses in Figure 7.3(b), had been rotated so that their principal axes were along the $v_{\mathcal{T}}$-axes. It is interesting to note that this corresponds to the case where $\mathcal{T}$ is the Karhunen–Loéve transform (KLT) associated with the correlation matrix $\mathbf{R}$. We may also recall from Chapter 4 that in the latter case $\mathbf{R}_{\mathcal{T}} = \mathbf{\Lambda}$, i.e. the diagonal matrix consisting of the eigenvalues of $\mathbf{R}$. Furthermore, from the minimax theorem (of Chapter 4) we find that this corresponds to the case where the diagonal elements of $\mathbf{R}_{\mathcal{T}}$ are maximally spread.

Moreover, a closer look at the present example reveals that the effect of normalization in reducing the eccentricity of the contour ellipses depends on the relative size (i.e. spread) of the diagonal elements of $\mathbf{R}_{\mathcal{T}}$. In other words, the spread of the signal power at the filter taps after transformation appears to be the key factor which determines the success of a TDAF. The discussion that follows in the rest of this section aims at exploring further this aspect of TDAFs.

7.6.2 A useful performance index

In the study of the LMS algorithm, eigenvalue spread, i.e. $\lambda_{\max}/\lambda_{\min}$, of the correlation matrix, $\mathbf{R}$, of the underlying input process is the most widely used performance index. In this book also, so far, we have emphasized the significance of eigenvalue spread. Unfortunately, there is no way of getting closed-form (explicit) equations for the maximum, $\lambda_{\max}$, and minimum, $\lambda_{\min}$, eigenvalues of a matrix $\mathbf{R}$, in general. As a result, application of this index for any further study of the TDLMS algorithm and its comparison with the conventional LMS algorithm is not possible. Hence, we shall look for other possible performance indices that may be mathematically tractable.

Farhang-Boroujeny and Gazor (1991, 1992) proposed an index that is mathematically tractable and able to give some further insight into the effect of orthogonal transforms in improving the performance of the LMS algorithm. The proposed index is

$$\rho(\mathbf{R}) = \left(\frac{\lambda_a}{\lambda_g}\right)^N, \tag{7.47}$$

where λ_a and λ_g are the arithmetic and geometric averages, respectively, of the eigenvalues of $\mathbf{R}$. Namely,

$$\lambda_a = \frac{\sum_i \lambda_i}{N} \tag{7.48}$$

and

$$\lambda_g = \sqrt[N]{\prod_i \lambda_i}. \tag{7.49}$$

We note that the value of $\rho(\mathbf{R})$ depends on the distribution of the eigenvalues of $\mathbf{R}$. It is always greater than or equal to one. It approaches one when all the eigenvalues of $\mathbf{R}$ assume about the same values, and increases as the eigenvalues of $\mathbf{R}$ spread apart. Furthermore, the lower bound $\rho(\mathbf{R}) = 1$ is reached when the eigenvalues of $\mathbf{R}$ are all equal. Using the identities (see Chapter 4)

$$\sum_i \lambda_i = \text{tr}[\mathbf{R}],$$

where $\text{tr}[\mathbf{R}]$ is the trace of $\mathbf{R}$, and

$$\prod_i \lambda_i = \det[\mathbf{R}],$$

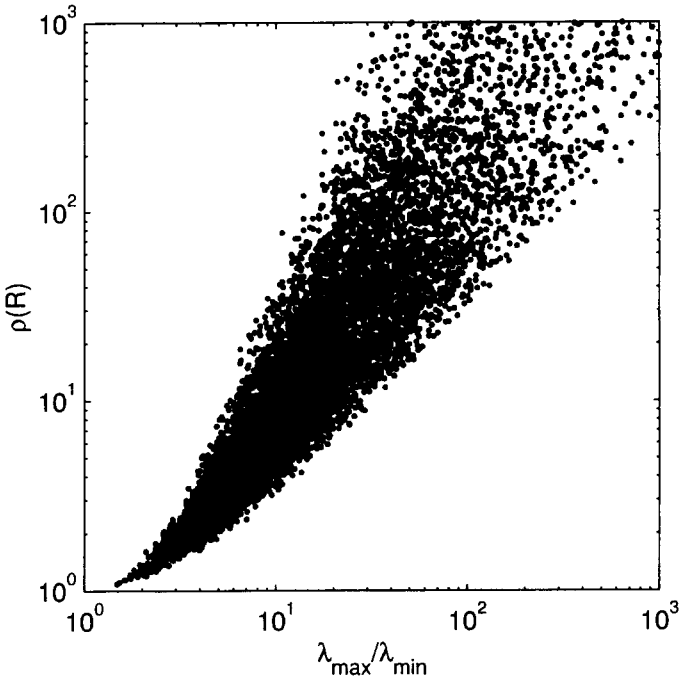


Figure 7.4 Variation of $\rho(\mathbf{R})$ versus eigenvalue spread of $\mathbf{R}$

where det[$\mathbf{R}$] is the determinant of $\mathbf{R}$, we obtain

$$\rho(\mathbf{R}) = \frac{(\text{tr}[\mathbf{R}]/N)^N}{\det[\mathbf{R}]}. \tag{7.50}$$

Now we may appreciate the index $\rho(\mathbf{R})$ because of its closed-form nature in terms of the elements of $\mathbf{R}$.

Before we proceed with the application of the performance index $\rho(\mathbf{R})$ to study the TDLMS algorithm further, we may remark that the relationship between $\rho(\mathbf{R})$ and the eigenvalue spread of $\mathbf{R}$, i.e. $\lambda_{max}/\lambda_{min}$, is rather complicated. The index $\rho(\mathbf{R})$ depends not only on $\lambda_{max}/\lambda_{min}$, but also, the distribution of the rest of the eigenvalues of $\mathbf{R}$ in the range λ_{min} to λ_{max}. However, the general trend is that a large eigenvalue spread of $\mathbf{R}$ implies a large $\rho(\mathbf{R})$ and vice versa. Similarly, a $\rho(\mathbf{R})$ close to one implies that the eigenvalue spread of $\mathbf{R}$ is small. Figure 7.4 shows how $\rho(\mathbf{R})$ varies as a function of $\lambda_{max}/\lambda_{min}$ when $N = 10$ and the eigenvalues of $\mathbf{R}$ are assumed to be a set of random numbers distributed in the range zero to one.

7.6.3 Improvement factor and comparisons

To compare a pair of LMS-based algorithms, say LMS_1 and LMS_2, we define an improvement factor, I_ρ, as the natural logarithm of the ratio of the performance index

$\rho(\cdot)$ in the two cases. In particular, when LMS_1 is compared with LMS_2, we define

$$I_\rho = \ln \rho(\mathbf{R}_1) - \ln \rho(\mathbf{R}_2), \tag{7.51}$$

where $\mathbf{R}_1$ and $\mathbf{R}_2$ are the associated correlation matrices in LMS_1 and LMS_2, respectively. A positive I_ρ indicates that LMS_2 is superior, and a negative I_ρ indicates that LMS_2 is inferior. In comparing a TDLMS algorithm and its conventional LMS counterpart, we shall let $\mathbf{R}_1 = \mathbf{R}$ and $\mathbf{R}_2 = \mathbf{R}_\mathcal{T}^N$. On the other hand, if no normalization is applied in the implementation of the TDLMS algorithm, then we shall let $\mathbf{R}_2 = \mathbf{R}_\mathcal{T}$. Thus, for the latter case, the corresponding improvement factor is

$$I_{\rho,\mathcal{T}} = \ln \rho(\mathbf{R}) - \ln \rho(\mathbf{R}_\mathcal{T}). \tag{7.52}$$

We note that

$$\begin{aligned} \rho(\mathbf{R}_\mathcal{T}) &= \frac{(\text{tr}[\mathbf{R}_\mathcal{T}]/N)^N}{\det[\mathbf{R}_\mathcal{T}]} \\ &= \frac{(\text{tr}[\mathcal{T}\mathbf{R}\mathcal{T}^{\text{T}}]/N)^N}{\det[\mathcal{T}\mathbf{R}\mathcal{T}^{\text{T}}]}. \end{aligned} \tag{7.53}$$

To simplify this we recall the following results from matrix algebra. If $\mathbf{A}$ and $\mathbf{B}$ are $N \times M$ and $M \times N$ matrices, respectively, then

$$\text{tr}[\mathbf{AB}] = \text{tr}[\mathbf{BA}]. \tag{7.54}$$

Also, when $\mathbf{A}$ and $\mathbf{B}$ are square matrices

$$\det[\mathbf{AB}] = \det[\mathbf{BA}] = \det[\mathbf{A}] \cdot \det[\mathbf{B}]. \tag{7.55}$$

Using (7.54) and (7.55) in (7.53) we obtain

$$\rho(\mathbf{R}_\mathcal{T}) = \frac{(\text{tr}[\mathcal{T}^{\text{T}}\mathcal{T}\mathbf{R}]/N)^N}{\det[\mathcal{T}^{\text{T}}\mathcal{T}\mathbf{R}]} = \rho(\mathbf{R}), \tag{7.56}$$

where the last equality follows since $\mathcal{T}^{\text{T}}\mathcal{T} = \mathbf{I}$. Substituting (7.56) in (7.52) we get

$$I_{\rho,\mathcal{T}} = 0.$$

This shows that transformation without normalization has no effect in improving the performance of the LMS algorithm. This result, which was also predicted by the geometrical interpretation of the TDLMS algorithm before (see Figure 7.3), can also be understood if we recall the definition of $\rho(\mathbf{R})$ (i.e. (7.47)) while noting that for an arbitrary orthogonal transformation $\mathcal{T}$, with $\mathcal{T}\mathcal{T}^{\text{T}} = \mathbf{I}$, the eigenvalues of $\mathbf{R}$ and $\mathbf{R}_\mathcal{T} = \mathcal{T}\mathbf{R}\mathcal{T}^{\text{T}}$ are the same (see Problem P7.7).

Another case of interest to be noted here is the comparison of the conventional LMS algorithm and the ideal LMS–Newton algorithm. In this case, $\mathbf{R}_1 = \mathbf{R}$ and $\mathbf{R}_2 = \mathbf{I}$. Thus, the improvement achieved using an ideal LMS–Newton algorithm instead of its

conventional LMS counterpart is

$$I_{\rho,\max} = \ln \rho(\mathbf{R}) - \ln \rho(\mathbf{I}) = \ln \rho(\mathbf{R}), \tag{7.57}$$

since $\rho(\mathbf{I}) = 1$. The notation $I_{\rho,\max}$ reflects the fact that the ideal LMS–Newton algorithm results in the maximum possible improvement that can be achieved by modifying the conventional LMS algorithm. Furthermore, (7.57) indicates that $\ln \rho(\mathbf{R})$ can be considered as a measure of the *distance* of the LMS algorithm from the ideal LMS–Newton algorithm. Similarly, in evaluating a particular implementation of the TDLMS algorithm, the value of $\ln \rho(\mathbf{R}_T^{\mathcal{N}})$ shows the distance of the TDLMS algorithm from the ideal LMS–Newton algorithm and, thus, it may be considered as a parameter indicating the extent of decorrelation that is achieved by the transformation.

The following theorem shows an easy way to compare a TDNLMS algorithm with its conventional LMS counterpart.

Theorem *When the conventional LMS algorithm is replaced by its TDLMS counterpart, the resulting improvement factor is*

$$I_{\rho,T}^{\mathcal{N}} = \ln \rho(\operatorname{diag}[\mathbf{R}_T]), \tag{7.58}$$

where $\mathbf{R}_T = \mathcal{T}\mathbf{R}\mathcal{T}^T$, $\mathbf{R}$ *is the correlation matrix of the underlying input process,* $\mathcal{T}$ *is the transformation matrix, and* $\operatorname{diag}[\mathbf{R}_T]$ *denotes the diagonal matrix consisting of the diagonal elements of* $\mathbf{R}_T$.

Proof According to (7.51), the improvement factor is

$$I_{\rho,T}^{\mathcal{N}} = \ln \rho(\mathbf{R}) - \ln \rho(\mathbf{R}_T^{\mathcal{N}}). \tag{7.59}$$

We recall that the diagonal elements of the normalized $N \times N$ matrix $\mathbf{R}_T^{\mathcal{N}}$ are all equal to one. This implies that

$$\operatorname{tr}[\mathbf{R}_T^{\mathcal{N}}] = N. \tag{7.60}$$

Noting this and using (7.22) and (7.55), we may proceed as follows:

$$\begin{aligned}
\rho(\mathbf{R}_T^{\mathcal{N}}) &= \frac{(\operatorname{tr}[\mathbf{R}_T^{\mathcal{N}}]/N)^N}{\det[\mathbf{R}_T^{\mathcal{N}}]} \\
&= \frac{1}{\det[\mathbf{D}^{-1/2}\mathbf{R}_T\mathbf{D}^{-1/2}])} \\
&= \frac{1}{\det[\mathbf{D}^{-1}\mathbf{R}_T]} \\
&= \frac{1}{\det[\mathbf{D}^{-1}] \cdot \det[\mathbf{R}_T]} \\
&= \frac{\det[\mathbf{D}]}{\det[\mathbf{R}_T]},
\end{aligned} \tag{7.61}$$

where the last equality follows from the identity $\det[\mathbf{D}^{-1}] = (\det[\mathbf{D}])^{-1}$. Next, substituting for $\mathbf{D}$ from (7.23) and noting that $\text{tr}[\mathbf{R}_{\mathcal{T}}] = \text{tr}[\text{diag}[\mathbf{R}_{\mathcal{T}}]]$, we get

$$\begin{aligned}\rho(\mathbf{R}_{\mathcal{T}}^{\mathcal{N}}) &= \frac{\det[\text{diag}[\mathbf{R}_{\mathcal{T}}]]}{\det[\mathbf{R}_{\mathcal{T}}]} \\ &= \frac{\det[\text{diag}[\mathbf{R}_{\mathcal{T}}]]}{(\text{tr}[\text{diag}[\mathbf{R}_{\mathcal{T}}]]/N)^N} \cdot \frac{(\text{tr}[\mathbf{R}_{\mathcal{T}}]/N)^N}{\det[\mathbf{R}_{\mathcal{T}}]} \\ &= \frac{\rho(\mathbf{R}_{\mathcal{T}})}{\rho(\text{diag}[\mathbf{R}_{\mathcal{T}}])}. \end{aligned} \tag{7.62}$$

Substituting (7.62) in (7.59) and noting that $\rho(\mathbf{R}) = \rho(\mathbf{R}_{\mathcal{T}})$, completes the proof.

The following corollary follows:

Corollary *Since* $\ln \rho(\text{diag}[\mathbf{R}_{\mathcal{T}}])$ *is always non-negative, the performance of a TDLMS algorithm can never be worse than its conventional LMS counterpart.*

The following remark may also be made. When comparing a TDLMS algorithm with its conventional LMS counterpart, the degree of improvement achieved depends on the distribution of the signal power at various outputs of the transformation, i.e. the tap inputs $x_{\mathcal{T},i}(n)$. A wide spread of signal power at the taps indicates a significant improvement. Similarly, a small spread in signal powers indicates that the improvement achievable is much less.

7.6.4 Filtering view

The quantitative result of the above theorem suggests that for a given input process a transformation matrix will effectively decorrelate the samples of input, if it implements a set of parallel FIR filters whose output powers are close to maximal spread. The maximally spread signal powers, here, are quantified by the *minimax theorem*, which was introduced in Chapter 4. When the correlation matrix of the underlying input process is known, the minimax theorem suggests a procedure for the optimal selection of a set of filters that achieve maximum power spreading. It starts with the design of a set of filters (with orthogonal coefficient vectors) whose output powers are maximized. Instead, it may also start with the design of another set of filters whose output powers are minimized. We also note that these two optimization procedures are implemented independent of each other, but both result in the same set of eigenvectors. This gives an intuitive feeling of how the minimax theorem (procedure) finds a transformation with a maximum spread of signal powers at its outputs.

We note that while the minimax theorem suggests a procedure for the design of the optimal transform for a given input process, the above theorem gives a measure of the effectiveness of a transformation matrix in decorrelating the samples of an underlying input process. We note that for a given input process with correlation matrix $\mathbf{R}$, the maximum attainable improvement factor is $I_{\rho,\max} = \ln \rho(\mathbf{R})$, and this is achieved when $\mathcal{T}$ is the KLT of the underlying input process. On the other hand, for a given transformation, $\mathcal{T}$, $I_{\rho,\mathcal{T}}^{\mathcal{N}} = \ln \rho(\text{diag}[\mathbf{R}_{\mathcal{T}}])$. Thus, the difference $I_{\rho,\max} - I_{\rho,\mathcal{T}}^{\mathcal{N}}$ gives a

measure of the success of $\mathcal{T}$ in decorrelating the input samples. A small value of $I_{\rho,\max} - I_{\rho,\mathcal{T}}^{\mathcal{N}}$ indicates that the transformation used is close to optimal and vice versa. Furthermore, as explained in Section 7.6.3, $I_{\rho,\max} - I_{\rho,\mathcal{T}}^{\mathcal{N}} = \ln \rho(\mathbf{R}_{\mathcal{T}}^{\mathcal{N}})$ is also the distance of the TDLMS from the ideal LMS–Newton algorithm.

It is instructive to elaborate more on the power-spreading effect of a transformation $\mathcal{T}$ and relate that to the above findings. We recall that the output power of a filter with the transfer function $F(e^{j\omega})$, input $x(n)$ and output $y(n)$ is given by (see Chapter 2)

$$E[y^2(n)] = \frac{1}{2\pi}\int_0^{2\pi} \Phi_{xx}(e^{j\omega})|F(e^{j\omega})|^2 \, d\omega, \tag{7.63}$$

where $\Phi_{xx}(e^{j\omega})$ is the power spectral density of $x(n)$. Now, if $F(e^{j\omega})$ is the transfer function of a filter whose coefficients constitute the elements of a row of a transformation matrix $\mathcal{T}$, with $\mathcal{T}\mathcal{T}^{\mathrm{T}} = \mathbf{I}$, then $F(e^{j\omega})$ is constrained to satisfy the following identity:

$$\frac{1}{2\pi}\int_0^{2\pi} |F(e^{j\omega})|^2 \, d\omega = 1. \tag{7.64}$$

This follows from Parseval's relation (see Chapter 2, Section 2.2). Noting this, we may say that the diagonal elements of $\mathbf{R}_{\mathcal{T}}$ (i.e. the signal powers at the outputs of the FIR filters defined by the rows of $\mathcal{T}$) are a set of averaged values of the power spectral density function, $\Phi_{xx}(e^{j\omega})$, of the underlying input process. The weighting functions used to obtain these averages are the squared magnitude responses of the FIR filters associated with the various rows of $\mathcal{T}$.

The numerical example that was given in Section 7.3 shows that the DCT is very effective in decorrelating the samples of the input process, $x(n)$, which was considered there. A closer look at this particular example is very instructive. Figure 7.5 shows the power spectral density, $\Phi_{xx}(e^{j\omega})$, of the underlying input process, $x(n)$. The main characteristic of this process to be noted here is that it is of lowpass nature, i.e. most of its spectral energy is concentrated over low frequencies. We also refer to Figure 7.2 where the magnitude responses of the DCT filters are shown, for $N = 8$, and note the following features. The side lobes of the filters whose passbands are over higher frequencies (closer to 0.5) are smaller than the side lobes of the filters whose passbands are over lower frequencies (close to zero). This, as we show next, is a very special characteristic of the DCT which make it an effective transform when it is applied for decorrelating the samples of a process that is dominantly lowpass in nature. To see this, we refer to (7.63) and note that when the main (passband) lobe of $F(e^{j\omega})$ lies in frequency bands where $\Phi_{xx}(e^{j\omega})$ is large (relative to its values in other frequency bands), the value of $E[y^2(n)]$ is not much affected by the size of the side lobes of $F(e^{j\omega})$. On the other hand, when the main lobe of $F(e^{j\omega})$ lies in frequency bands where $\Phi_{xx}(e^{j\omega})$ is relatively small, the value of $E[y^2(n)]$ may be significantly affected by the side lobes of $F(e^{j\omega})$, as these side lobes, although small, are multiplied by some large values of $\Phi_{xx}(e^{j\omega})$ before integration. In the context of orthogonal transforms and signal power spreading, the minimization of the side lobes of $F(e^{j\omega})$ in the latter case to reduce the value of $E[y^2(n)]$ is very critical. Referring back to the DCT filters and the size of their associated side lobes, we find that the DCT has the necessary properties to be effective in achieving a close to maximum signal power spreading when applied to any lowpass signal.

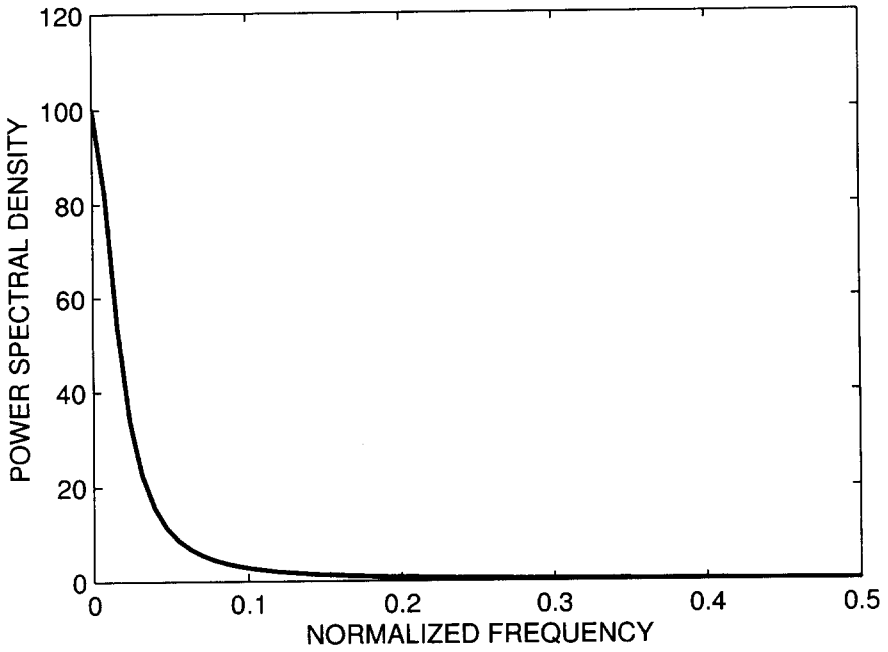


Figure 7.5 Power spectral density of the process $x(n)$ which is generated by the colouring filter (7.17), for $\alpha = 0.9$

To get further insight into the above results, we consider two more examples. We consider two choices of the inputs, $x_1(n)$ and $x_2(n)$, that are generated by passing a unit variance white noise process through two colouring filters which are specified by the system functions

$$H_1(z) = 0.1 + 0.2z^{-1} + 0.3z^{-2} + 0.4z^{-3} + 0.4z^{-4} + 0.2z^{-5} + 0.1z^{-6}$$

and

$$H_2(z) = 0.1 - 0.2z^{-1} - 0.3z^{-2} + 0.4z^{-3} + 0.4z^{-4} - 0.2z^{-5} - 0.1z^{-6},$$

respectively. Figures 7.6 and 7.7 show the power spectral densities of $x_1(n)$ and $x_2(n)$. We note that $x_1(n)$ and $x_2(n)$ are low- and bandpass processes, respectively.

We also consider two choices of $\mathcal{T}$:

1. The DCT matrix whose coefficients are specified by (7.14).
2. The discrete sine transform (DST) which is specified by the coefficients

$$s_{kl} = \left(\frac{2}{N+1}\right)^{1/2} \sin\frac{kl\pi}{N+1}, \quad k, l = 1, 2, \ldots, N. \tag{7.65}$$

We expect the DCT to perform well when applied to $x_1(n)$, since this is a lowpass process.

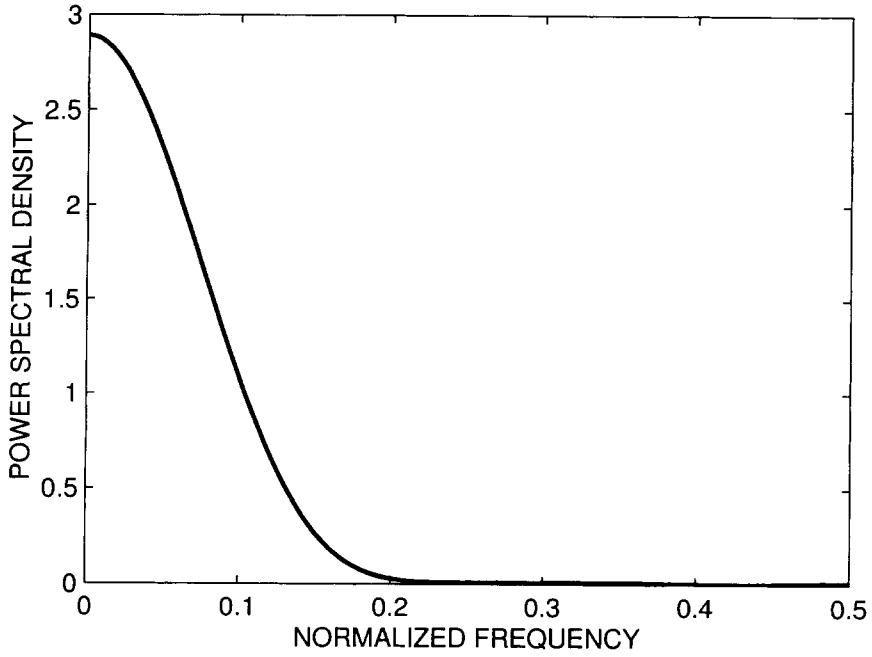


Figure 7.6 Power spectral density of the process $x_1(n)$

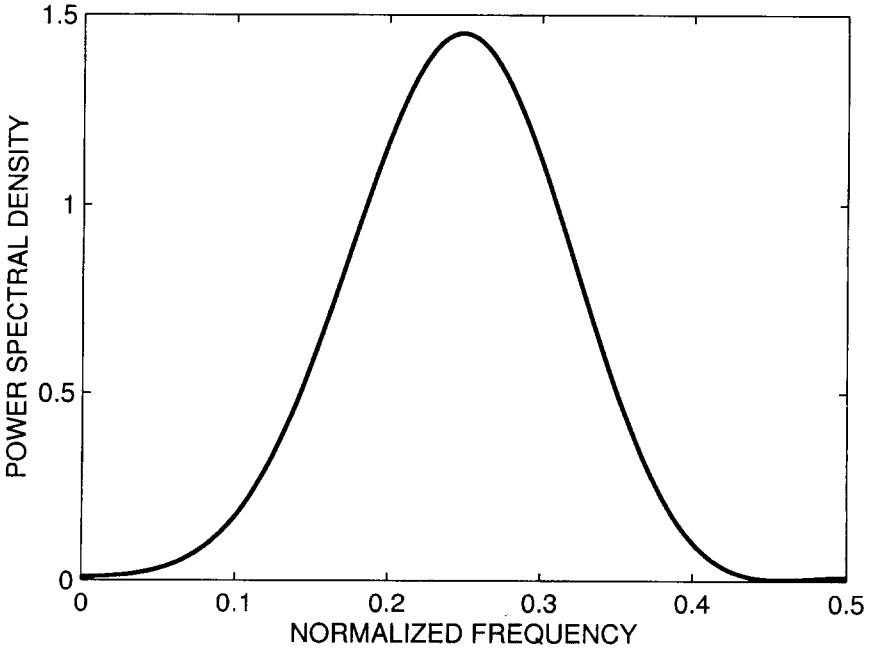


Figure 7.7 Power spectral density of the process $x_2(n)$

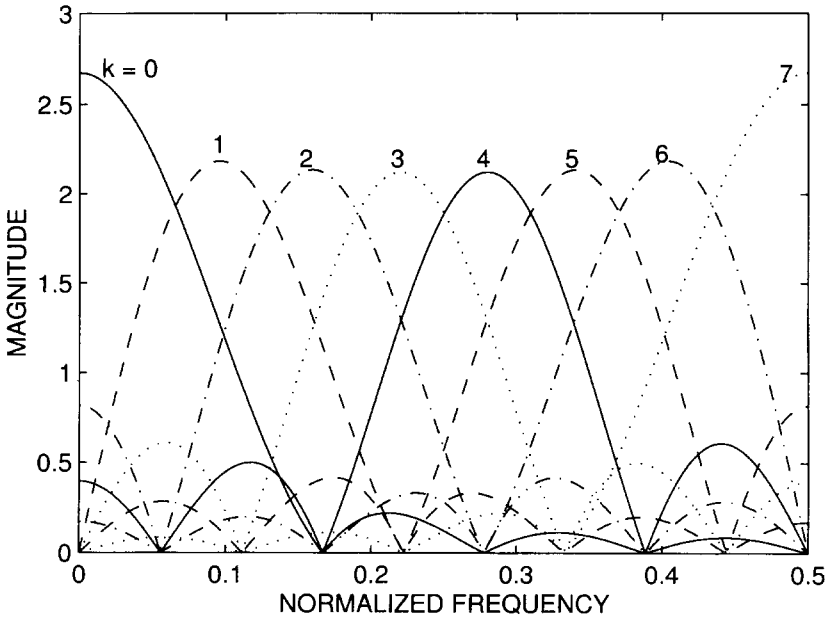


Figure 7.8 shows the magnitude responses of the DST filters for $N = 8$. For the DST, we observe that the side lobes of the filters whose passbands belong to high or low frequencies are relatively smaller than the side lobes of the filters whose passbands are within the midband frequencies. Thus, according to our discussion above, we expect DST to perform well when applied to $x_2(n)$.

Table 7.2 shows the results of some numerical calculations that have been performed to observe the effect of the two transformations in decorrelating the samples of $x_1(n)$ and $x_2(n)$. These results compare the eigenvalue spread of $\mathbf{R}$ and $\mathbf{R}_T^{\mathcal{N}}$ of the respective processes, for three values of filter length, N. Also, to illustrate that the improvement factor, I_ρ, and variation in eigenvalue spread of the respective matrices are tightly

Table 7.2 Comparison of the DST and DCT transformations when applied to lowpass process $x_1(n)$ and bandpass process $x_2(n)$.

		$\lambda_{\max}/\lambda_{\min}$			I_ρ	
Process	N	$\mathbf{R}$	$\mathbf{R}_{\text{DCT}}^{\mathcal{N}}$	$\mathbf{R}_{\text{DST}}^{\mathcal{N}}$	DCT	DST

related, values of I_ρ are also presented in Table 7.2. As was predicted, the DCT performs better for $x_1(n)$ and the DST performs better for $x_2(n)$.

Reviewing the above observations, the following guidelines may be drawn for the selection of the transformation $\mathcal{T}$:

> *In general, transforms whose associated band-partitioning filters have smaller side lobes are expected to perform better than those with larger side lobes. When an estimate of the power spectral density, $\Phi_{xx}(e^{j\omega})$, of the underlying input process, $x(n)$, is known, selection of those transforms whose associated filters have smaller side lobes within the frequency bands where $\Phi_{xx}(e^{j\omega})$ is large, leads to a more significant performance improvement.*

7.7 Transforms

Although, in general, there are infinite possible choices of the transformation matrix $\mathcal{T}$, only a few transforms have been widely used in practice. The main feature of such transforms is that there are many fast algorithms for their efficient implementation. They also exhibit a good signal separation, i.e. from a band-partitioning point of view, and they all offer well-behaved sets of parallel FIR filters with approximately mutually exclusive passbands. In the application of TDAFs, the most commonly used transforms are:

1. *Discrete Fourier transform (DFT):* The DFT is the most widely used transform in various applications of signal processing. The klth element of the DFT transformation matrix, $\mathcal{T}_{\text{DFT}}$, is

$$f_{kl} = \frac{1}{\sqrt{N}} e^{-j2\pi kl/N}, \quad \text{for} \quad 0 \le k, l \le N-1. \tag{7.66}$$

   The factor $1/\sqrt{N}$ on the right-hand side of (7.66) is to normalize the DFT coefficients so that $\mathcal{T}_{\text{DFT}}\mathcal{T}^{\text{H}}_{\text{DFT}} = \mathbf{I}$.

   The distinct feature of DFT, as compared with other transforms, is that it distinguishes between positive and negative frequencies. This, among all the widely used transforms, makes DFT the most effective transform in cases where the underlying input process has a non-symmetrical power spectral density with respect to $\omega = 0$, i.e. for complex-valued inputs. If the input is real-valued, then DFT has no advantage over the other transforms. In fact, its complex-valued coefficients add some unnecessary redundancy to the transformed signal samples, which then increases the complexity of the system.
2. *Real DFT (RDFT):* When N is even, the coefficients of RDFT are given by

$$f^{\text{R}}_{kl} = \begin{cases} \dfrac{1}{\sqrt{N}}, & k = 0, \ 0 \le l \le N-1, \\ \sqrt{\dfrac{2}{N}} \cos \dfrac{2\pi kl}{N}, & 1 \le k \le \frac{1}{2}N - 1, \ 0 \le l \le N-1, \\ \dfrac{1}{\sqrt{N}}(-1)^l, & k = \frac{1}{2}N, \ 0 \le l \le N-1, \\ \sqrt{\dfrac{2}{N}} \sin \dfrac{2\pi kl}{N}, & \frac{1}{2}N + 1 \le k \le N-1, \ 0 \le l \le N-1. \end{cases} \tag{7.67}$$

3. *Discrete Hartley transform (DHT):* The DHT coefficients are defined as

$$h_{kl} = \frac{1}{\sqrt{N}}\left(\cos\frac{2\pi kl}{N} + \sin\frac{2\pi kl}{N}\right), \quad \text{for } 0 \le k, l \le N-1. \tag{7.68}$$

   Both RDFT and DHT may be viewed as derivatives of the DFT which, for real-valued signals, exploit the redundancy of the transformed samples and suggest a lower complexity implementation of TDAFs. Experiments on TDAFs with DFT, RDFT and DHT show that they all perform about the same when the underlying input process is real-valued.
4. *Discrete cosine transform (DCT):* There are a few variations of DCT (Ersoy, 1997). However, the most widely used DCT is the one defined in (7.14).
5. *Discrete sine transform (DST):* Similar to DCT, there are also a few variations of DST (Ersoy, 1997). However, the most widely used DST is the one defined in (7.65).
6. *Walsh–Hadamard transform (WHT):* The WHT is defined when the transformation length, N, is a power of 2. The WHT coefficients are

$$w_{kl} = \frac{1}{\sqrt{N}}\prod_{p=0}^{m-1}(-1)^{b_p(k)b_{m-1-p}(l)}, \quad \text{for} \quad 0 \le k, l \le N-1, \tag{7.69}$$

   where $m = \log_2 N$, and $b_p(k)$ is the pth bit (with $p = 0$ referring to the least significant bit) of the binary representation of k.

   The main characteristic of the WHT is its simplicity, since all of its coefficients are $+1$ or -1 and, as a result, its implementation does not involve any multiplication. We note that in the implementation of the TDLMS algorithm, the common coefficient, $1/\sqrt{N}$, which is just a normalization factor, can be dropped since the step-size normalization of the TDLMS algorithm takes care of signal normalization. The price paid for this simplicity of the WHT is its higher side lobes as compared with other transforms. This, of course, results in poorer performance of the WHT when applied to TDAFs in general.

7.8 Sliding Transforms

The conventional fast algorithms available for implementation of the transforms introduced in the previous section all require $O(N \log N)$ operations (additions, subtractions or multiplications), where $O(\cdot)$ denotes *order of*. The term *order of* x means a value proportional to x with a fixed proportionality constant. In the context of transversal filters and their corresponding transform domain implementation, there is an important property of the filter tap-input vector, $\mathbf{x}(n)$, that can be used to reduce further the complexity of the latter transforms. Namely, when $\mathbf{x}(n) = [x(n)\ x(n-1)\ \ldots\ x(n-N+1)]^T$, $\mathbf{x}(n)$ and $\mathbf{x}(n+1)$ have $N-1$ elements in common. $\mathbf{x}(n+1)$ is obtained from $\mathbf{x}(n)$ by shifting (sliding) the elements of $\mathbf{x}(n)$ one element down, dropping out the last element of $\mathbf{x}(n)$, and adding the new sample of input, $x(n+1)$, as the first element of $\mathbf{x}(n+1)$. In this section we exploit this data redundancy in the successive tap-input vectors $\mathbf{x}(n)$ and $\mathbf{x}(n+1)$ and introduce two $O(N)$ complexity schemes for efficient implementation of the transformation part of TDAFs. These are called sliding transforms.

7.8.1 Frequency sampling filters

A useful common property of the transforms that were introduced in the previous section (with the exception of the WHT) is that the transfer functions of their corresponding FIR filters can be written in a compact recursive form. These transfer functions can then be used for efficient implementation of the respective transforms. To clarify this, we consider the DFT filters as an example.

The transfer function of the kth DFT filter is

$$H_{\mathrm{DFT}}^{k\mathcal{N}}(z) = \sum_{l=0}^{N-1} f_{kl} z^{-l}. \tag{7.70}$$

The superscript $\mathcal{N}$ in (7.70) emphasizes that the f_{kl} coefficients have been normalized so that $\sum_{l=0}^{N-1} |f_{kl}|^2 = 1$.

Substituting (7.66) in (7.70) we obtain

$$\begin{aligned} H_{\mathrm{DFT}}^{k\mathcal{N}}(z) &= \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} \mathrm{e}^{-j2\pi kl/N} z^{-l} \\ &= \frac{1}{\sqrt{N}} \frac{1 - (\mathrm{e}^{-j2\pi k/N} z^{-1})^N}{1 - \mathrm{e}^{-j2\pi k/N} z^{-1}} \\ &= \frac{1}{\sqrt{N}} \frac{1 - z^{-N}}{1 - \mathrm{e}^{-j2\pi k/N} z^{-1}}. \end{aligned} \tag{7.71}$$

When the TDLMS algorithm is used to adapt a DFT-based TDAF, the constant factor $1/\sqrt{N}$ may be dropped from the right-hand side of (7.71), since signal-normalization is taken care of by the step-normalization in the TDLMS algorithm, as discussed in earlier sections. Thus, the (unnormalized) transfer function of the kth DFT filter may be defined as

$$H_{\mathrm{DFT}}^{k}(z) = \frac{1 - z^{-N}}{1 - \mathrm{e}^{-j2k\pi/N} z^{-1}}. \tag{7.72}$$

The transfer functions associated with other transforms can also be derived in a similar way. For transforms with real-valued coefficients, we have to start with expanding the sine and cosine coefficients in terms of their associated complex exponents, then proceed as in the case of DFT filters and pack the results. At the end, any fixed scale factor in front of the final results is dropped.

Table 7.3 summarizes the transfer functions that are associated with various transforms. We have not included WHT here since its transfer functions do not have any closed-form equivalent. Hence, a different approach has to be adopted to arrive at an efficient implementation of the WHT. This is discussed in Problem P7.12.

The term *frequency sampling filter* is used to refer to the filters defined by the transfer functions given in Table 7.3. This is because each transfer function corresponds to a narrow-band filter which samples a small band of the spectrum of the underlying input process.

Table 7.3 Transfer functions associated with the various transforms (frequency sampling filters)

$$H^k_{\mathrm{DFT}}(z) = \frac{1 - z^{-N}}{1 - \mathrm{e}^{-j2\pi k/N} z^{-1}}$$

$$H^k_{\mathrm{RDFT}}(z) = \begin{cases} \dfrac{1 - z^{-N}}{1 - z^{-1}}, & \text{for } k = 0 \\ \dfrac{\left(1 - \cos\dfrac{2\pi k}{N} z^{-1}\right)(1 - z^{-N})}{1 - 2\cos\dfrac{2\pi k}{N} z^{-1} + z^{-2}}, & \text{for } 1 \le k \le \frac{1}{2}N - 1 \\ \dfrac{1 - z^{-N}}{1 + z^{-1}}, & \text{for } k = \frac{1}{2}N \\ \dfrac{z^{-1}(1 - z^{-N})}{1 - 2\cos\dfrac{2\pi k}{N} z^{-1} + z^{-2}}, & \text{for } \frac{1}{2}N + 1 \le k \le N - 1 \end{cases}$$

$$H^k_{\mathrm{DHT}}(z) = \frac{\left(1 - \left(\cos\dfrac{2\pi k}{N} - \sin\dfrac{2\pi k}{N}\right) z^{-1}\right)(1 - z^{-N})}{1 - 2\cos\dfrac{2\pi k}{N} z^{-1} + z^{-2}}$$

$$H^k_{\mathrm{DCT}}(z) = \frac{(1 - z^{-1})(1 - (-1)^k z^{-N})}{1 - 2\cos\dfrac{\pi k}{N} z^{-1} + z^{-2}}$$

$$H^k_{\mathrm{DST}}(z) = \frac{1 + (-1)^k z^{-(N+1)}}{1 - 2\cos\dfrac{\pi(k+1)}{N+1} z^{-1} + z^{-2}}$$

Table 7.3 provides all the necessary information for the development of the two realizations of the sliding transforms which are categorized as recursive and non-recursive structures, and are discussed below.

7.8.2 Recursive realization of sliding transforms

A direct realization of the transfer functions given in Table 7.3 suggests a simple recursive scheme for the implementation of the associated transforms. As an example, we present here a recursive realization of the DCT filters. Recursive realization of the other transforms which follow the same concept is then straightforward.

From Table 7.3 we have

$$H^k_{\mathrm{DCT}}(z) = \frac{(1 - z^{-1})(1 - (-1)^k z^{-N})}{1 - 2\cos\dfrac{\pi k}{N} z^{-1} + z^{-2}}. \tag{7.73}$$

This is the transfer function of the kth DCT filter. Figure 7.9 depicts a detailed realization of (7.73). In this realization we have purposefully divided the transfer function of $H^k_{\mathrm{DCT}}(z)$ into three separate parts. Namely, the forward parts, $1 - z^{-1}$ and

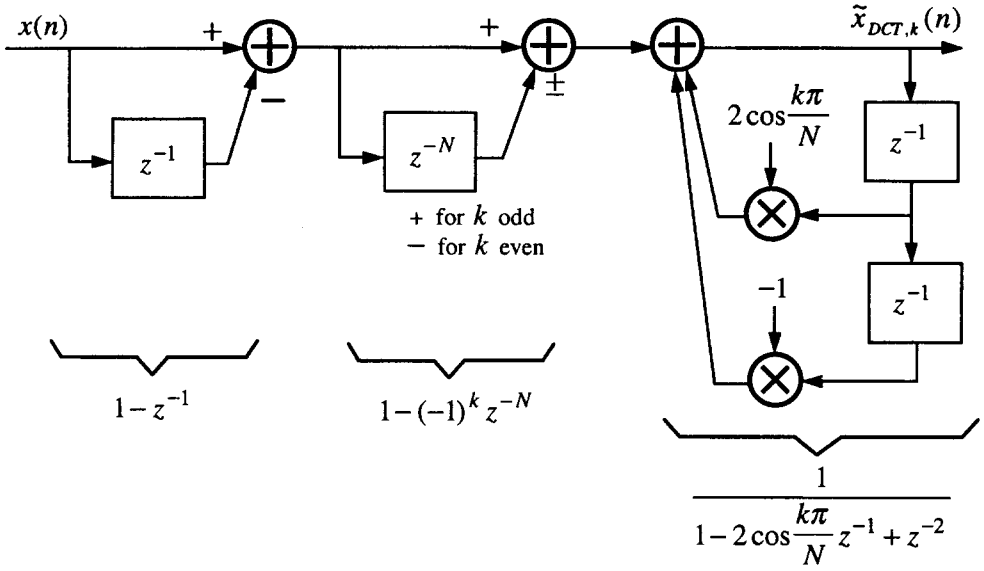


Figure 7.9 A realization of $H^k_{\mathrm{DCT}}(z)$

$1-(-1)^k z^{-N}$, and the feedback part, $[1-2\cos(\pi k/N)z^{-1}+z^{-2}]$. This separation facilitates the integration of the DCT filters (for $k=0,1,\ldots,N-1$) in a parallel structure.

Figure 7.10 depicts a block diagram of the DCT frequency sampling filters when they are put together in a parallel structure. Points to be noted here are:

1. For $k=0$,

$$\frac{1}{1-2\cos\dfrac{\pi k}{N}z^{-1}+z^{-2}}=\frac{1}{(1-z^{-1})^2}.$$

Substituting this result in (7.73) we obtain

$$H^0_{\mathrm{DCT}}(z)=\frac{1-z^{-N}}{1-z^{-1}}. \tag{7.74}$$

This has been considered in the block diagram of Figure 7.10 and, thus, the case $k=0$ has been treated separately.

2. We also note that

$$1-(-1)^k z^{-N}=\begin{cases}1-z^{-N}, & \text{for } k \text{ even},\\ 1+z^{-N}, & \text{for } k \text{ odd}.\end{cases}$$

Thus, the cases of k even and odd are separated at the first stage of Figure 7.10. However, when implementing the structure of Figure 7.10, we should note that the

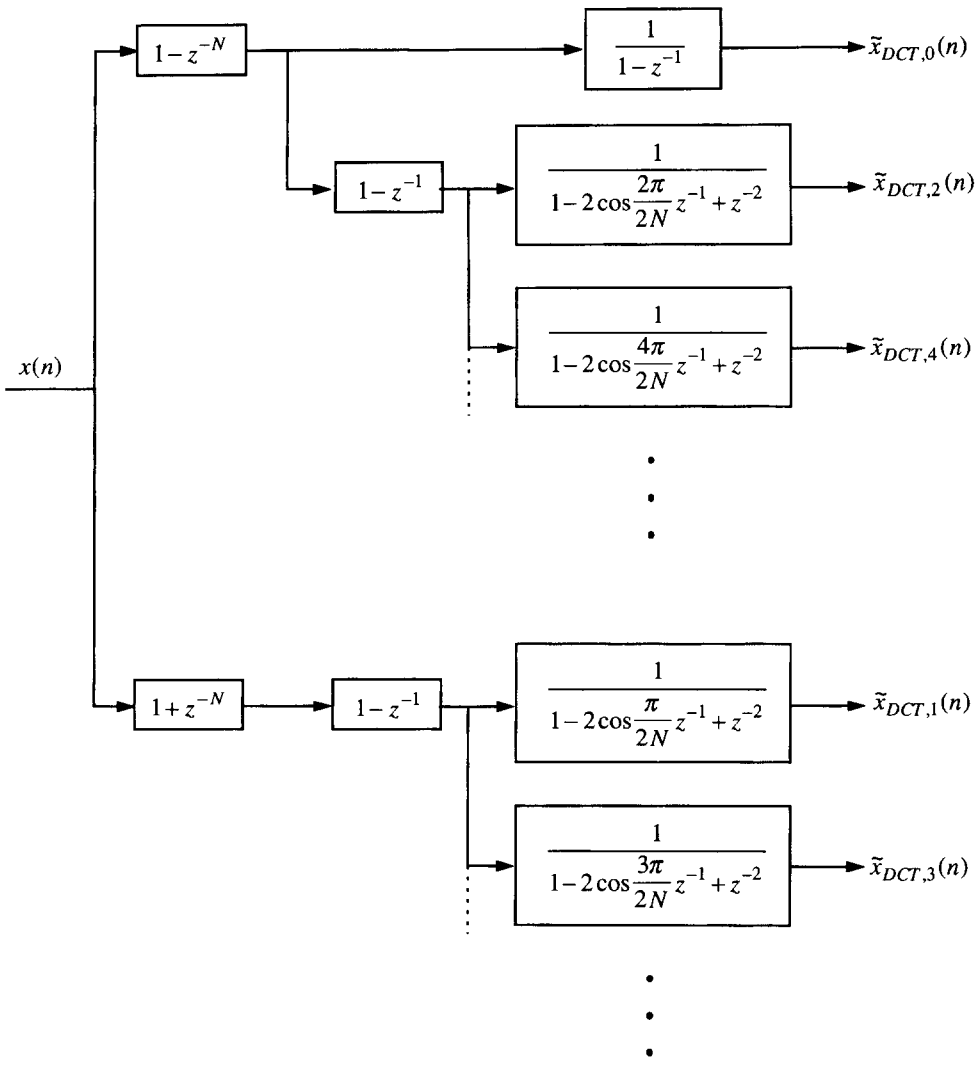


Figure 7.10 A parallel realization of the recursive DCT frequency sampling filters

blocks $1-z^{-N}$ and $1+z^{-N}$ have the same common input and, thus, can share the same delay line to hold the past samples of the input. This reduces the memory requirement of the system.

A common problem with the recursive realization of the frequency sampling filters that needs careful attention is that these filters are only marginally stable. They can easily run into instability problems, unless some special care is taken to ensure stability. This is because the poles of the frequency sampling filters are all on the unit circle and, as a result, any round-off error will accumulate and grow unbounded. Furthermore, quantization of the filter coefficients may result in poles outside the unit circle and thus result in unstable filters.

The above problem can be alleviated by replacing z^{-1} with βz^{-1}, where β is a constant smaller than, but close to, one. This shifts all the poles and zeros of the frequency sampling filters which are ideally on the unit circle to a circle with radius $\beta < 1$. This stabilizes the filters at the cost of some additional complexity in their realization, since the addition of β changes some of the filter coefficients that otherwise would have been unity.

7.8.3 Non-recursive realization of sliding transforms

The non-recursive sliding transforms that are introduced in this section use the following common property of the frequency sampling filters:

> *The frequency sampling filters associated with each transform have a common set of zeros out of which each filter selects $N-1$.*

Bruun (1978) noted the significance of the above property in the case of DFT and used that to develop a fast Fourier transform (FFT) structure. Farhang-Boroujeny, Lee and Ko (1996) noted that a rearrangement of Bruun's algorithm leads to a sliding DFT structure and extended the concept to the other transforms. In the rest of this section we present the sliding transforms that have been proposed in Farhang-Boroujeny, Lee and Ko (1996) and demonstrate their efficiency in the implementation of TDAFs.

Bruun's algorithm as a sliding DFT

The transfer functions of the DFT frequency sampling filters are (from Table 7.3):

$$H_{\mathrm{DFT}}^{k}(z) = \frac{1 - z^{-N}}{1 - \mathrm{e}^{-j2\pi k/N} z^{-1}}, \quad \text{for} \quad k = 0, 1, \ldots, N-1. \tag{7.75}$$

We note that the zeros of these filters are all taken from the set of N th roots of unity, i.e. $\mathrm{e}^{-j2\pi k/N}$, for $k = 0, 1, \ldots, N-1$. We also note that each DFT filter has one pole which belongs to the same set. As a result, we find that a pole–zero cancellation occurs and, thus, each DFT filter has effectively $N-1$ zeros out of the set of Nth roots of unity and no pole.

Bruun used this simple concept and suggested an elegant factorization of $1 - z^{-N}$ and used these results to form a tree structure as shown in Figure 7.11 (for $N = 16$) to realize the various FIR frequency sampling filters of DFT. The following identities are used for the factorization of $1 - z^{-N}$:

$$1 - z^{-2M} = (1 - z^{-M})(1 + z^{M}) \tag{7.76}$$

and

$$1 + az^{-2M} + z^{-4M} = (1 + \sqrt{2 - a}\, z^{-M} + z^{-2M})(1 - \sqrt{2 - a}\, z^{-M} + z^{-2M}). \tag{7.77}$$

These factorizations which are used until the last stage of the tree structure, have the following two features:

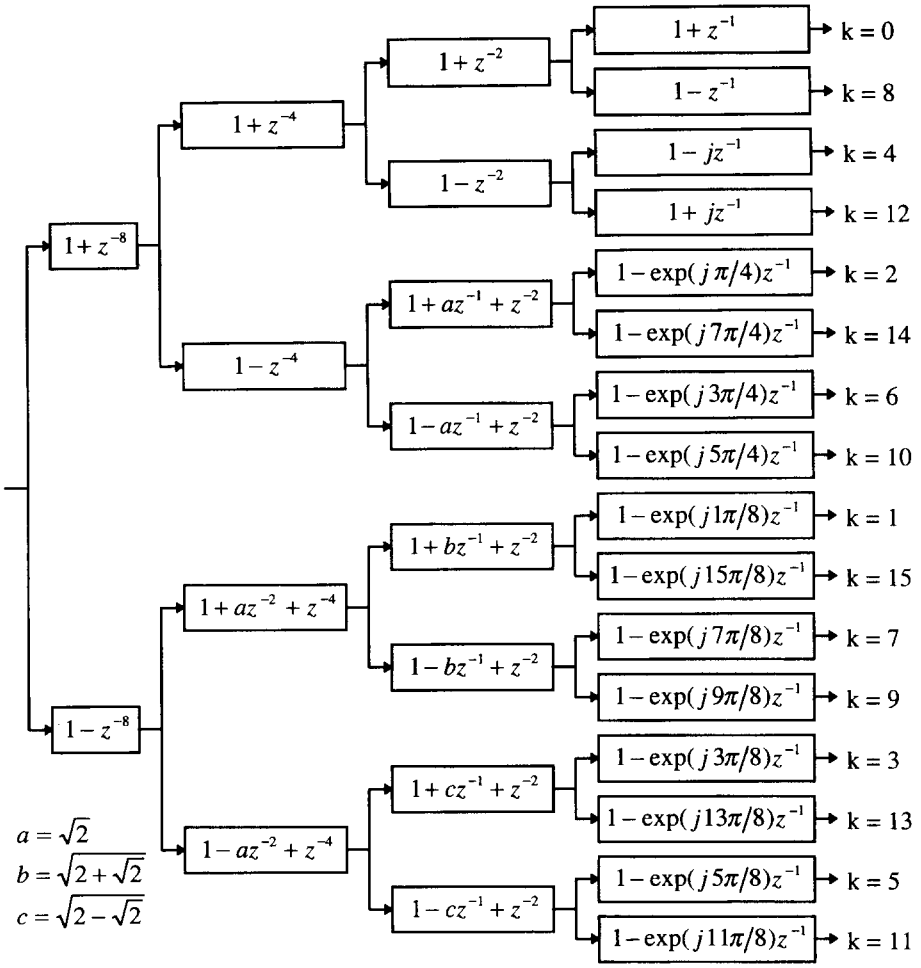


Figure 7.11 Non-recursive sliding DFT: $N = 16$ (Bruun, 1978). Reprinted from *Signal Processing*, vol. 52, B. Farhang-Boroujeny, Y. Lee and C.C. Ko, 'Sliding transforms for efficient implementation of transform domain adaptive filters', pp. 83–96, copyright (1996), with permission from Elsevier Science

1. Each factor consists of either two or three sparse taps.
2. There is at most one non-trivial, real-valued coefficient in each factor.

To see how the above identities could be used to develop the tree structure of Figure 7.11, we note that the factors that appear in the first stage are those of $1 - z^{-16} = (1 - z^{-8})(1 + z^{-8})$. The branches that follow after the factor $1 + z^{-8}$ are made of the factors of the other branch of the first stage, i.e. $1 - z^{-8} = (1 - z^{-4})(1 + z^{-4})$. Similarly, the branches that follow the factor $1 - z^{-8}$ are made of the factors of $1 + z^{-8} = (1 + \sqrt{2}z^{-2} + z^{-4})(1 - \sqrt{2}z^{-2} + z^{-4})$. The same procedure is used to determine the other branches of the structure. At the end of the third stage (in our particular example), each path of the tree covers 14 out of the 16 zeros of $1 - z^{-16}$. The remaining two zeros

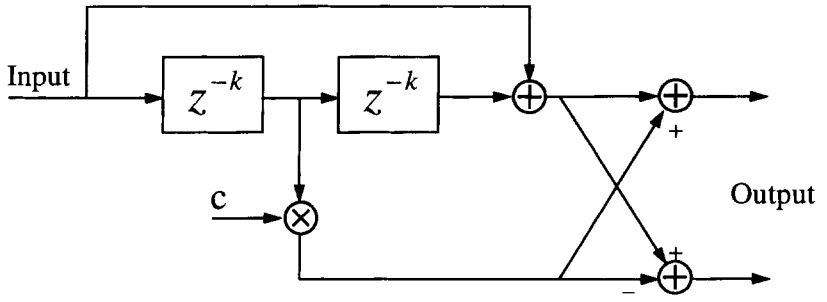


Figure 7.12 An implementation of the filter pair $1 \pm cz^{-k} + z^{-2k}$, when they share a common input. Reprinted from *Signal Processing*, vol. 52, B. Farhang-Boroujeny, Y. Lee and C.C. Ko, 'Sliding transforms for efficient implementation of transform domain adaptive filters', pp. 83–96, copyright (1996), with permission from Elsevier Science

that have not been covered by each path are complex conjugates, except for the top path whose corresponding missing zeros are $z = \pm 1$. One out of the two missing zeros is then added at the last stage. The same procedure can be used to develop the same structure for any value of N (the transform length) which is a power of 2.

Bruun elaborated on the tree structure of Figure 7.11 and proposed his FFT structure (Bruun, 1978). In the context of the TDLMS algorithm, we are interested in an efficient implementation of the DFT frequency sampling filters and updating their outputs after the arrival of each new data sample. The tree structure of Figure 7.11 is exactly what we are looking for. Thus, we hold on to this structure as an efficient way of implementing the non-recursive sliding DFT filters.

To appreciate the efficiency of the structure given in Figure 7.11, we shall elaborate on it further. We note that the pair of filters which originate from a common node at any stage share the same coefficients and, thus, they can be implemented jointly as depicted

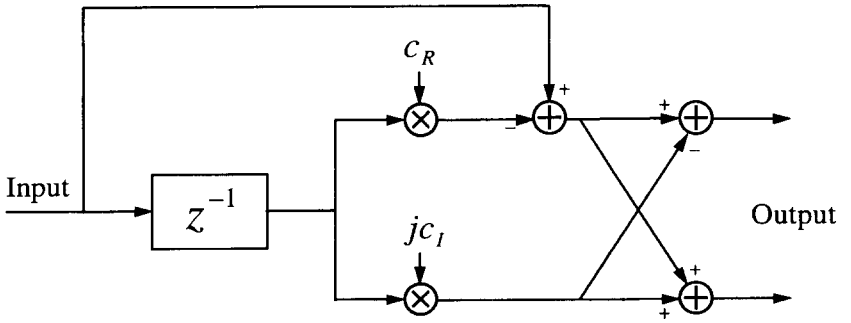


Figure 7.13 An implementation of the filter pair $(1 - cz^{-1}, 1 - c^*z^{-1})$, when they share a common input. Reprinted from *Signal Processing*, vol. 52, B. Farhang-Boroujeny, Y. Lee and C.C. Ko, 'Sliding transforms for efficient implementation of transform domain adaptive filters', pp. 83–96, copyright (1996), with permission from Elsevier Science

in Figure 7.12. For a real-valued sequence, this implementation requires only one multiplication and three additions. For a complex-valued input, the number of operations is twice this figure. We may also note that each filter pair at the output stage in Figure 7.11 uses a pair of complex conjugate coefficients, and therefore the corresponding multiplications can be shared. Figure 7.13 depicts a joint implementation of a filter pair of the output stage of Figure 7.11. In this implementation, c_R and c_I denote the real and imaginary parts of c, respectively, where c and c^* are the pair of filter coefficients. For a complex-valued input, this implementation requires four real multiplications and six real additions.

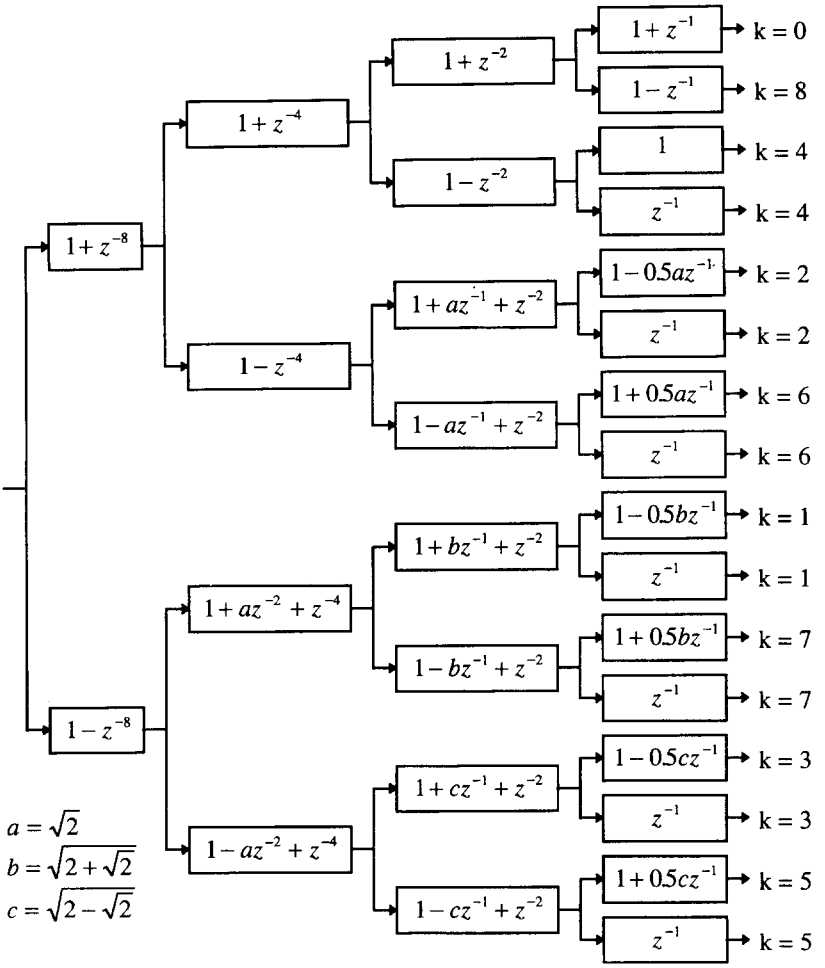
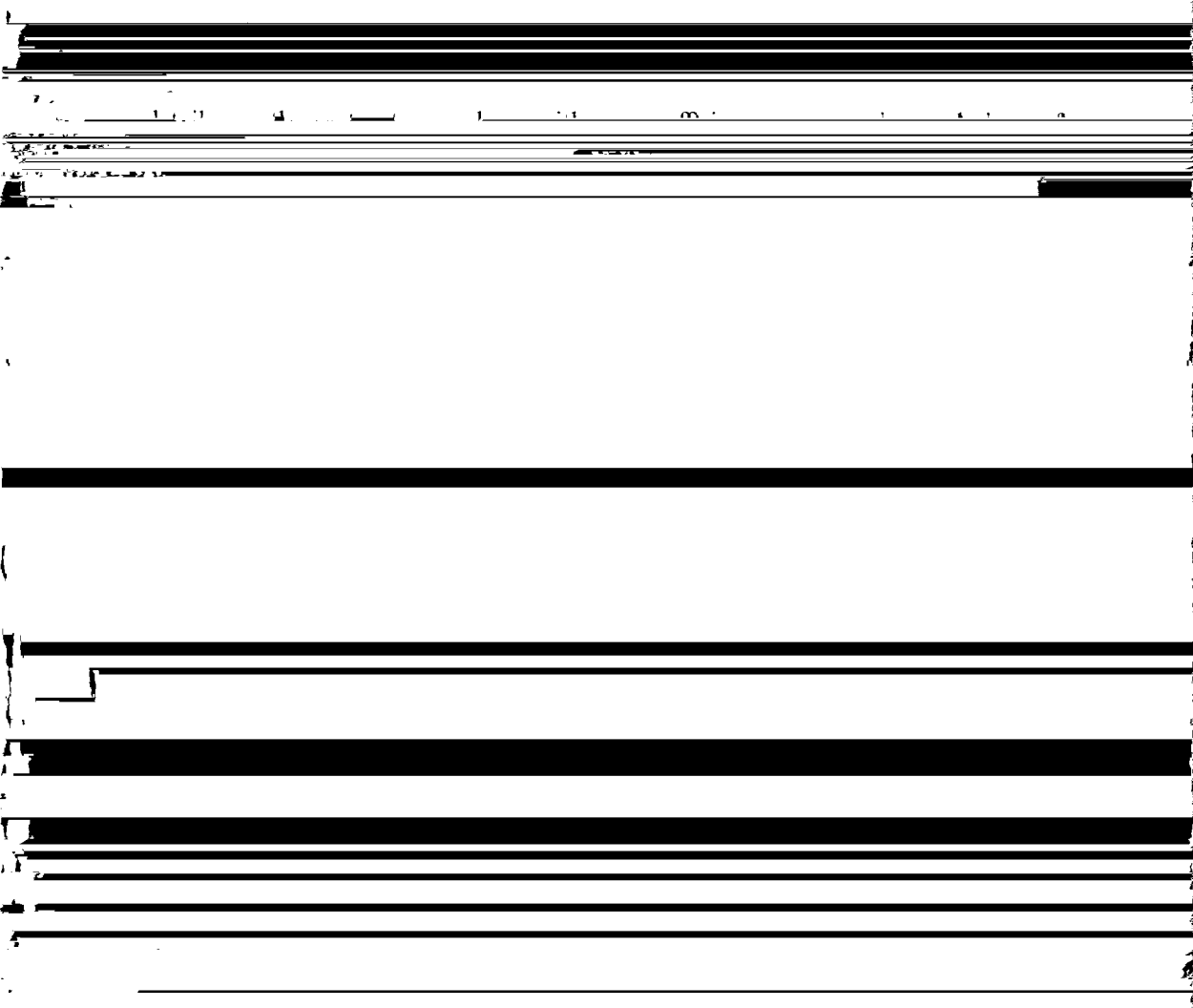
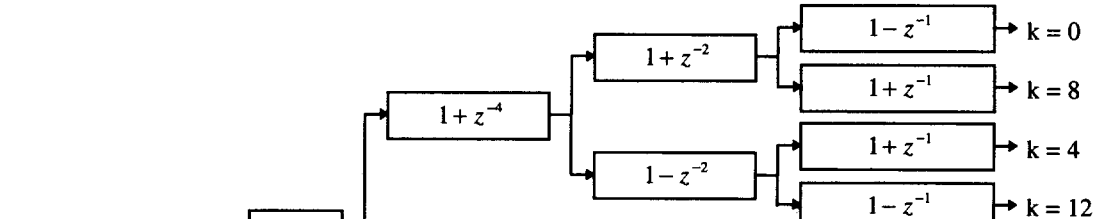


Figure 7.14 Non-recursive sliding RDFT: $N = 16$ (Farhang-Boroujeny, Lee and Ko, 1996). Reprinted from *Signal Processing*, vol. 52, B. Farhang-Boroujeny, Y. Lee and C.C. Ko, 'Sliding transforms for efficient implementation of transform domain adaptive filters', pp. 83–96, copyright (1996), with permission from Elsevier Science

Real-valued transforms

As was noted before, when the filter input is real-valued, about 50% of the DFT outputs are redundant because they appear in complex conjugate pairs. In such situations, transforms with real-valued coefficients are preferred. Following Bruun's factorization technique, it is not difficult to come up with tree structures similar to Figure 7.11 for other transforms. Figures 7.14–7.17 show a set of such tree structures for non-recursive sliding RDFT, DHT, DCT, and DST, respectively. Note that for the examples shown, the value of N is 16 for RDFT, DHT, and DCT and 15 in the case of DST. Further





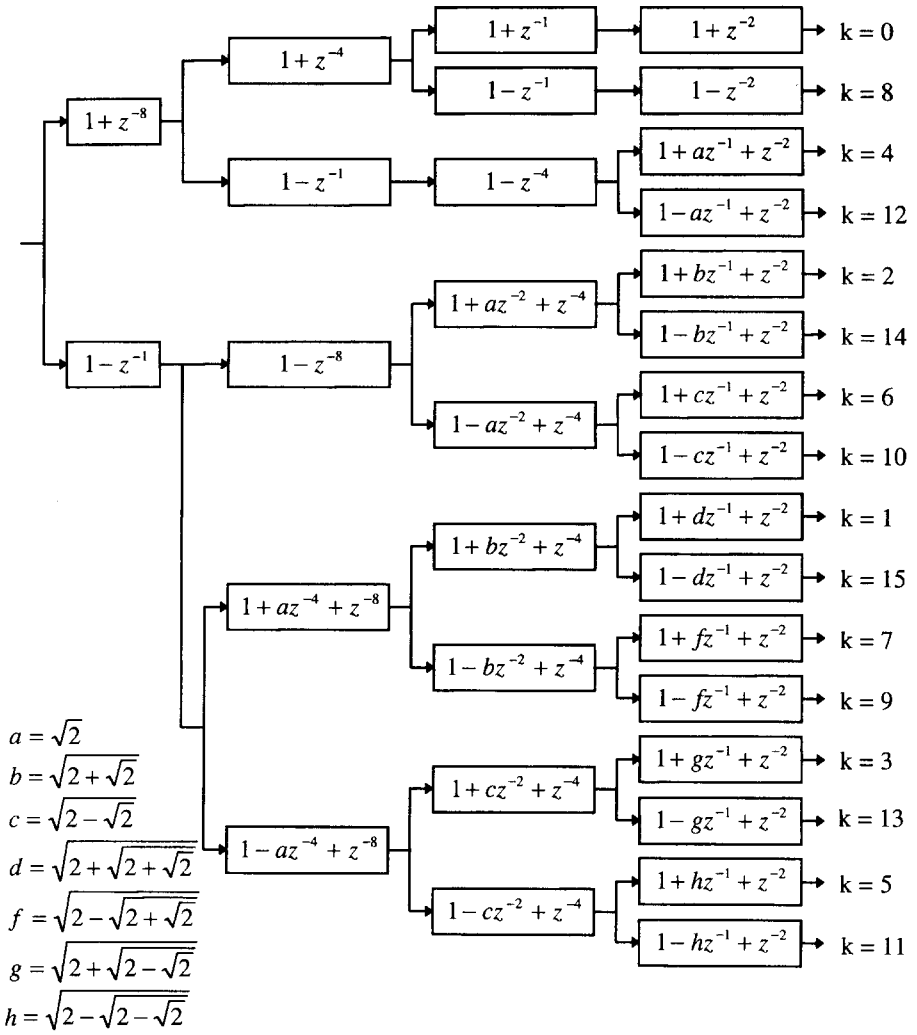


Figure 7.16 Non-recursive sliding DCT: $N = 16$ (Farhang-Boroujeny, Lee and Ko, 1996). Reprinted from *Signal Processing*, vol. 52, B. Farhang-Boroujeny, Y. Lee and C.C. Ko, 'Sliding transforms for efficient implementation of transform domain adaptive filters', pp. 83–96, copyright (1996), with permission from Elsevier Science

their software implementations, can be found in Farhang–Boroujeny, Lee and Ko (1996).

7.8.4 Comparison of recursive and non-recursive sliding transforms

In terms of robustness to numerical round-off errors, the non-recursive sliding transforms are superior to their recursive counterparts. A simple inspection of the non-recursive

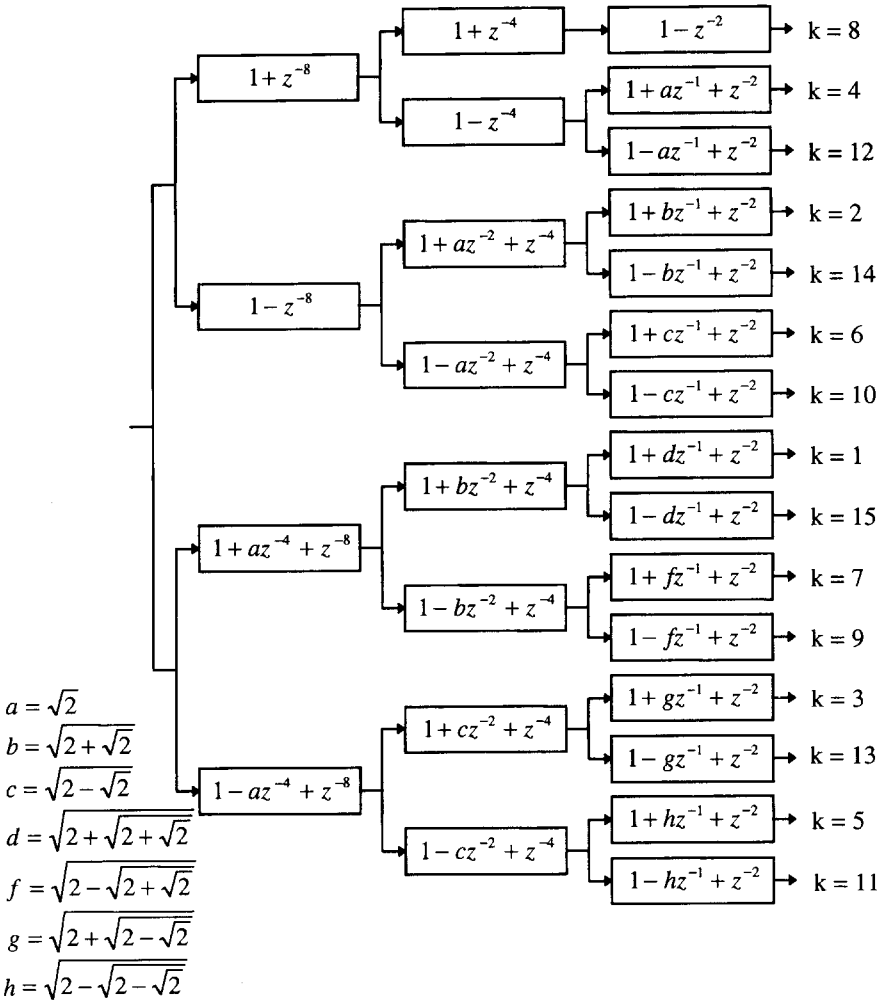


Figure 7.17 Non-recursive sliding DST: $N = 15$ (Farhang-Boroujeny, Lee and Ko, 1996). Reprinted from *Signal Processing*, vol. 52, B. Farhang-Boroujeny, Y. Lee and C.C. Ko, 'Sliding transforms for efficient implementation of transform domain adaptive filters', pp. 83–96, copyright (1996), with permission from Elsevier Science

sliding structures shows that each output in these structures is calculated on the basis of a very limited number of multiplications and additions. Furthermore, there is no feedback of numerical errors, thereby avoiding error accumulation. This property, which is inherent to all FFT-like structures, results in very low sensitivity to finite wordlength effects (Rabiner and Gold, 1975, and Oppenheim and Schafer, 1975). On the contrary, the recursive sliding transforms are highly sensitive to numerical error accumulation because of the feedback. The variances of such errors are proportional to $1/(1-\beta^2)$, where β is the stabilizing factor as defined before. Noting that β has to be selected close to one so

Table 7.4 Computation counts of the non-recursive and recursive sliding transforms

	Non-recursive		Recursive	
	Mults	Adds/Subs	Mults	Adds/Subs
DFT	$3N - 2m - 8$	$6N - 2m - 8$	$4N - 6$	$4N - 6$
RDFT	$N - m - 2$	$2N - m - 2$	$\frac{5N}{2} - 5$	$\frac{5N}{2} - 5$
DHT	$\frac{3N}{2} - m - 6$	$\frac{5N}{2} - m - 4$	$3N - 7$	$3N - 7$
DST	$N - m$	$3N - m - 3$	$2N$	$2N + 1$
DCT	$N - m - 1$	$3N - 5$	$2N + 1$	$2N + 2$

$m = \log_2 N$ for DFT, RDFT, DHT, and DCT
$m = \log_2(N + 1)$ for DST

that the deviation of the realized filters from the ideal frequency sampling filters would be minimum, these variances can be excessively large.

In terms of the number of operations per input sample, the non-recursive sliding transforms are also found to be superior to their recursive counterparts. Table 7.4 gives details of the operation counts of the two schemes. For the case of recursive implementations, the figures given in Table 7.4 have taken into account the effect of the stabilizing factor β.

The major drawback of the non-recursive sliding transforms is that they are limited to the cases where the filter length, N (filter length plus one in the case of DST) is a power of 2. On the contrary, the recursive sliding transforms can be used for any value of N.

7.9 Summary and Discussion

In this chapter we reviewed a class of adaptive filters known as transform domain adaptive filters (TDAFs). We gave a filtering interpretation of orthogonal transforms and demonstrated that a transformation may be viewed as a bank of bandpass filters which are used to separate different parts of the spectrum of the underlying input process. This led to a band-partitioning view of orthogonal transforms. It was thus concluded that the outputs from an orthogonal transformation constitute a set of partially decorrelated processes, since they belong to (partially) mutually exclusive bands.

Implementation of the LMS algorithm in the transform domain was then presented. This was called the transform domain LMS (TDLMS) algorithm. It was shown that significant improvement in the convergence behaviour of the TDLMS algorithm can be achieved if a proper set of normalized step-size parameters is used. This, which was called step-normalization, is assumed to be part of the TDLMS algorithm.

We showed that the TDLMS algorithm could equivalently be reformulated by normalizing the transformed samples of the underlying input process to the power of

unity and then using the conventional LMS algorithm (with a single step-size parameter for all taps) to adapt the filter tap weights. This formulation is theoretically of interest, since it allows the results of the conventional LMS algorithm to be used in evaluating the performance of the TDLMS algorithm.

The ideal LMS–Newton algorithm was introduced as a stochastic implementation of the Newton's search method of Chapter 5. The relationship between the TDLMS and ideal LMS–Newton algorithms was also established. We found that the TDLMS algorithm is in fact an approximation to the ideal LMS–Newton algorithm.

We noted that for a given input process the success of different transforms in decorrelating the samples of an input process varies. We presented a theory which relates the signal decorrelation property of orthogonal transforms to the distribution of signal powers after transformation. We demonstrated how this concept is related to the Karhunen–Loéve transform and drew some general guidelines for the selection of an appropriate transform when a rough estimate of the power spectral density of the underlying input process is known.

We also introduced various standard transforms which can be implemented efficiently using fast transforms. The sliding fast implementation of these transforms was then presented. We found that in the application of transform domain adaptive filters the commonly used transforms can all be implemented with an order of N computational complexity, where N is the filter length.

Problems

P7.1 Figure P7.1 shows the power spectral densities of four processes and the magnitude responses of their associated eigenfilters for $N = 5$, in some arbitrary order. Considering the maximum signal power-spreading property of the KLT, identify the magnitude response associated with each power spectral density.

P7.2 By substituting for past values of $\hat{\sigma}^2_{x_{T,i}}(n)$ in (7.28), show that $\hat{\sigma}^2_{x_{T,i}}(n)$ is an exponentially weighted average of the present and past samples of the $x^2_{T,i}(n)$s using the weighting function characterized by the coefficients $1, \beta, \beta^2, \ldots$, i.e.

$$\hat{\sigma}^2_{x_{T,i}}(n) = \frac{\sum_{k=0}^{\infty} \beta^k x^2_{T,i}(n-k)}{\sum_{k=0}^{\infty} \beta^k}.$$

P7.3 Assume a noisy sinusoidal sequence $s(n) = a\sin(\omega n + \phi) + \nu(n)$, where $\nu(n)$ is an uncorrelated noise sequence. The angular frequency ω is known a priori. However, the magnitude a and phase ϕ are unknown. To obtain an estimate of these parameters, a two-tap transversal filter whose input is chosen to be $x(n) = \sin \omega n$ is set up and its tap weights, $w_0(n)$ and $w_1(n)$, are adapted so that the difference between $s(n)$ and the filter output, $y(n)$, is minimized in the mean-square sense. The filter output, $y(n)$, is then a noise-free estimate of the sinusoidal sequence. The LMS algorithm is used for this purpose.

(i) Using time averages, find the correlation matrix $\mathbf{R}$ of the filter tap inputs.

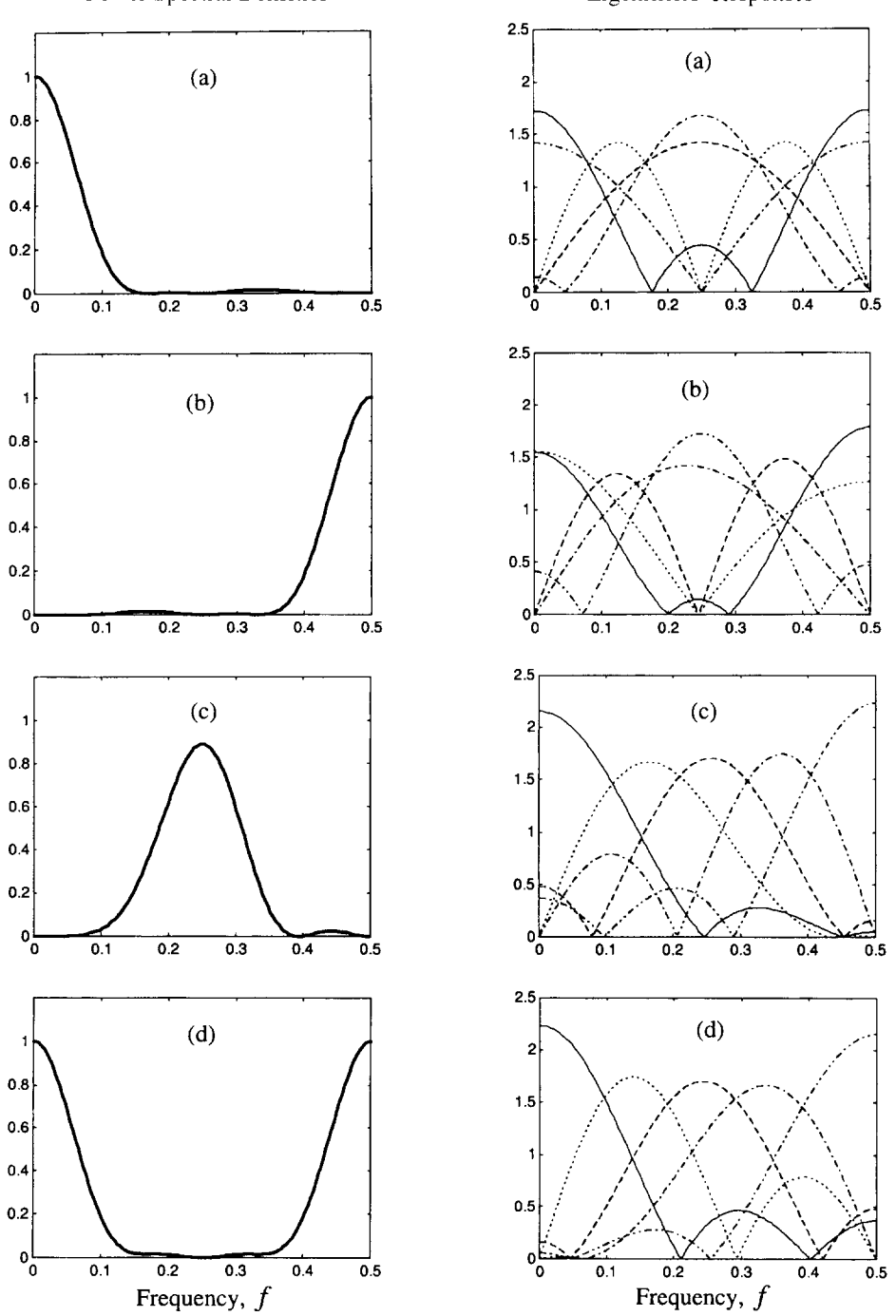

Power Spectral Densities
Eigenfilters' Responses
(a)
(b)
(c)
(d)
Frequency, f
Frequency, f

Figure P7.1

(ii) Find the step-size parameter, μ, of the LMS algorithm which results in 5% misadjustment.

(iii) For the step-size parameter obtained in (ii), find the time constants of the learning curve of the filter and show that the convergence of the LMS algorithm becomes slower as ω decreases.

(iv) Show that the problem of slow convergence of the LMS algorithm can be solved if a TDLMS algorithm with the transformation matrix

$$\mathcal{T} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

is used.

P7.4 An adaptive transversal filter is excited by two different inputs, $u(n)$ and $v(n)$, whose power spectral densities are presented in Figures P7.4(a) and (b), respectively.

(i) If the LMS algorithm is used in both cases and its step-size parameter is selected accordingly for a fixed level of misadjustment (say, 10%), which of the two inputs will result in the shortest transient time for the algorithm? Explain.

(ii) What will be your answer to part (i) if a DCT-based transform domain implementation of the adaptive filter is employed?

(iii) Will your answer to part (ii) change if the DCT is replaced by DST?

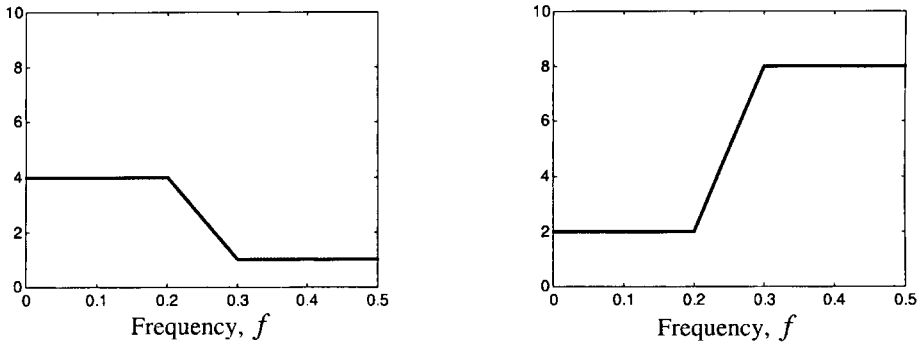


Figure P7.4

P7.5 Figure P7.5 shows the structure of a special adaptive filter whose tap inputs are the samples of the processes $u(n)$ and $v(n)$ that are generated from a stationary input process $x(n)$ as shown. Assume that the filter length, N, is an even number.

(i) Define the length N column vector

$$\tilde{\mathbf{x}}(n) = [u(n)\ \ v(n)\ \ u(n-2)\ \ v(n-2)\ \ \ldots\ \ u(n-N+2)\ \ v(n-N+2)]^{\mathrm{T}},$$

and show that

$$\tilde{\mathbf{x}}(n) = \mathcal{T}_2\mathbf{x}(n),$$

where

$$\mathcal{T}_2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & -1 \end{bmatrix}.$$

(ii) Show that $\mathcal{T}_2$ is an orthogonal matrix and, thus, conclude that the structure presented in Figure P7.5 corresponds to a TDAF with $\mathcal{T} = \mathcal{T}_2$.

(iii) You may note that $\mathcal{T}_2\mathcal{T}_2^{\mathrm{T}} = 2\mathbf{I}$. This is different from the unitary condition $\mathcal{T}\mathcal{T}^{\mathrm{T}} = \mathbf{I}$ which is usually assumed for the transformation matrix $\mathcal{T}$. Does this deviation affect the performance of the TDLMS algorithm?

(iv) If the TDLMS algorithm (with the step-normalization) is to be used for fast adaptation of this structure, give details of the equations required for such an implementation.

(v) Compare the structure of Figure P7.5 with that of a conventional LMS-based transversal adaptive filter, both in terms of computational complexity and memory requirement.

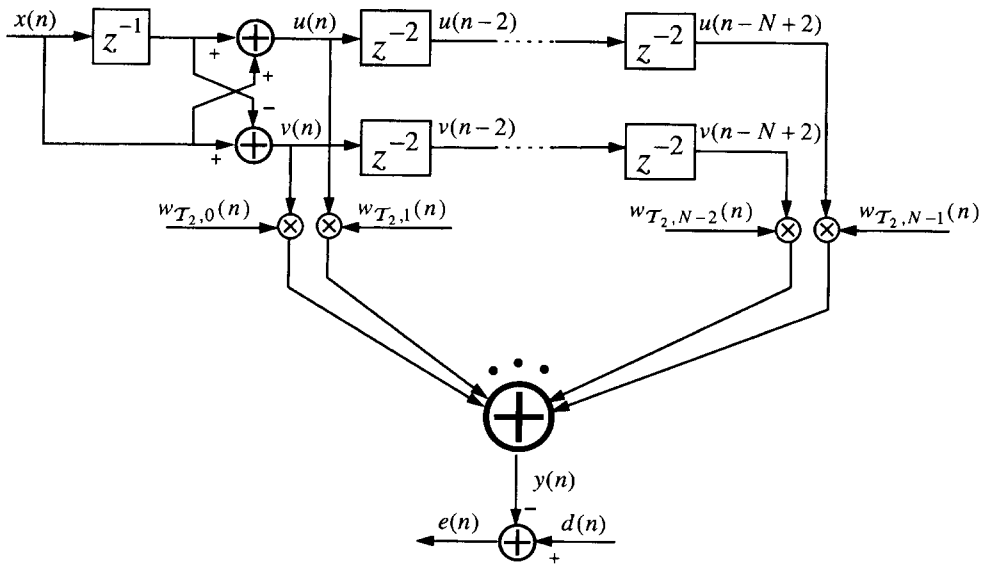


Figure P7.5

P7.6 Generalization of the adaptive filter structure given in Problem P7.5 may be done as follows.[6] Define the $N \times N$ matrix

$$\mathcal{T} = \begin{bmatrix} \mathcal{T}_{\text{sub}} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathcal{T}_{\text{sub}} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathcal{T}_{\text{sub}} \end{bmatrix},$$

where $\mathcal{T}_{\text{sub}}$ is an orthogonal square matrix and the $\mathbf{0}$ are zero matrices of appropriate dimensions.

(i) Show that $\mathcal{T}$ is an orthogonal matrix.
(ii) Considering the analogy between the transformation matrix $\mathcal{T}$ here and the one in Problem P7.5, construct a generalized version of Figure P7.5.
(iii) Noting that, in general, larger matrices achieve a higher degree of signal decorrelation (orthogonalization), discuss the convergence behaviour of the proposed structure as the size of $\mathcal{T}_{\text{sub}}$ increases.
(iv) Discuss the memory requirement and computational complexity of the proposed structure as the size of $\mathcal{T}_{\text{sub}}$ increases.

P7.7 Show that the identity $\mathcal{T}\mathcal{T}^{\mathrm{T}} = \mathbf{I}$ implies that the eigenvalues of $\mathbf{R}$ and $\mathbf{R}_{\mathcal{T}} = \mathcal{T}\mathbf{R}\mathcal{T}^{\mathrm{T}}$ are the same. Thus, conclude that $\rho(\mathbf{R}) = \rho(\mathbf{R}_{\mathcal{T}})$.

P7.8 With reference to the notations in Section 7.6, show that

$$I_{\rho,\max} - I_{\rho,\mathcal{T}}^{\mathcal{N}} = \ln \rho(\mathbf{R}_{\mathcal{T}}^{\mathcal{N}}).$$

P7.9 Consider a two-tap transversal filter that is characterized by the performance function

$$\xi(v_0, v_1) = 0.1 + [v_0 \;\; v_1]\mathbf{R}\begin{bmatrix} v_0 \\ v_1 \end{bmatrix},$$

where

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

Assume that the filter input is a real-valued random process.

(i) Find the points $(a, 0)$ and $(0, b)$ where the contour ellipse given by $\xi(v_0, v_1) = 1.1$ meets the v_0 and v_1 axes and show that $a = b$. Show that this result is directly related to the fact that the diagonal elements of $\mathbf{R}$ are the same which, in turn, implies that the signal energies at various taps of the filter are equal.

[6] This problem has been designed based on the work of Petraglia and Mitra (1993).

(ii) By sketching an arbitrary ellipse that passes through the points $(a, 0)$ and $(0, b)$ of part (i), verify that the principal axes of the sketched ellipse are always in the directions obtained by 45° rotation of the coordinate axes v_0 and v_1.

(iii) Define an orthogonal transformation matrix

$$\mathcal{T} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

and show that the transformation $\mathbf{v}_{\mathcal{T}} = \mathcal{T}\mathbf{v}$, where $\mathbf{v} = [v_0 \ v_1]^{\mathrm{T}}$, is equivalent to rotating the coordinate axes v_0 and v_1 by θ radian counterclockwise.

(iv) Find the rotation angle θ that maximizes the ratio of the diagonal elements of $\mathbf{R}_{\mathcal{T}} = \mathcal{T}\mathbf{R}\mathcal{T}^{\mathrm{T}}$ and show that it is independent of α.

(v) Noting that the diagonal elements of $\mathbf{R}_{\mathcal{T}}$ are the input signal energies after transformation, comment on your results in part (iv) and show that for two-tap transversal filters with real-valued input processes, the optimum transformation matrix, $\mathcal{T}_{\mathrm{opt}}$, is fixed and independent of the statistics of the underlying input process. What is $\mathcal{T}_{\mathrm{opt}}$?

P7.10 The autocorrelation matrix $\mathbf{R}$ of the input process to an adaptive filter is known. To use this information to speed up the adaptation of the filter, the following algorithm is proposed:[7]

$$\begin{aligned} \mathbf{x}_{\mathcal{T}}(n) &= \mathcal{T}\mathbf{x}(n), \\ y(n) &= \mathbf{w}_{\mathcal{T}}^{\mathrm{T}}(n)\mathbf{x}_{\mathcal{T}}(n), \\ e(n) &= d(n) - y(n), \\ \mathbf{w}_{\mathcal{T}}(n+1) &= \mathbf{w}_{\mathcal{T}}(n) + 2\mu e(n)\mathbf{x}_{\mathcal{T}}(n), \end{aligned}$$

where $\mathcal{T} = \mathbf{R}^{-1/2}$ which is the inverse of the square root of $\mathbf{R}$ (as defined in Chapter 4) and μ is a scalar step-size parameter. Note that the matrix $\mathcal{T}$ here is not an orthogonal matrix, and thus the proposed algorithm is different from the TDLMS algorithm introduced in this chapter. In particular, we may note that the proposed algorithm does not have step-normalization.

(i) Obtain the correlation matrix of the transformed samples, $\mathbf{x}_{\mathcal{T}}(n)$, and discuss the significance of $\mathcal{T} = \mathbf{R}^{-1/2}$ in increasing the speed of convergence of the adaptive filter.

(ii) Give an approximate equation for the misadjustment of the proposed algorithm.

(iii) Define $\mathbf{w}(n) = \mathbf{R}^{-1/2}\mathbf{w}_{\mathcal{T}}(n)$ and use that to show that the proposed algorithm is equivalent to the ideal LMS–Newton algorithm.

P7.11 In Section 7.8.1, a derivation of the DFT frequency sampling filters was given. Following the procedure used there, derive the rest of the system functions listed in Table 7.3.

---

[7] This problem has been designed based on the work of Widrow and Walach (1984).

P7.12 The transfer functions associated with the WHT cannot be written in a recursive form such as those given in Table 7.3 for the other transforms. However, we still find that each WHT filter may be implemented as a cascade of $\log_2 N$ non-recursive sparse coefficients filters, similar to the other transforms. In this problem we clarify this by exploring the WHT for the transformation length $N = 8$. The generalization of the results to any value of N which is a power of 2 is then obvious.

(i) Use (7.69) to find the coefficients of the WHT when $N = 8$.
(ii) Use the results of part (i) to write down the transfer functions associated with various rows of the WHT when $N = 8$.
(iii) Show that the transfer functions obtained in part (ii) can be factorized as

$$\frac{1}{\sqrt{8}}(1 \pm z^{-4})(1 \pm z^{-2})(1 \pm z^{-1}),$$

where the various combinations of the $\pm$ signs cover all the 8 filter transfer functions.
(iv) Using the latter factorization, propose a tree structure, similar to the non-recursive sliding transforms introduced in Section 7.8.3, for an $O(N)$ implementation of the WHT.

Simulation-Oriented Problems

P7.13 Consider a modelling problem where a plant

$$W_o(z) = 0.4 + z^{-1} - 0.3z^{-1}$$

is modelled using a three-tap transversal adaptive filter. The plant is assumed to be noise free. The input to the plant and adaptive filter is generated by passing a unit-variance white process through the colouring filter

$$H(z) = 0.1 - 0.3z^{-1} - 0.5z^{-2} + z^{-3} + z^{-4} - 0.5z^{-5} - 0.3z^{-6} + 0.1z^{-7}.$$

(i) Write a program to simulate this scenario. In your program, after every 10 iterations plot the magnitude response of the adaptive filter and observe how it converges toward the corresponding response of the plant.
(ii) Obtain and plot the power spectral density of the adaptive filter input, and try to relate that to your observation in part (i). You should find that the convergence of the magnitude response of the adaptive filter towards the plant response is frequency dependent. Over the frequency bands where the filter input has higher power, convergence is faster. On the other hand, the slow modes of the adaptive filter correspond to the bands where the filter input is poorly excited, i.e. having low power spectral density.

P7.14 Develop and run your own program(s) to confirm the results of Table 7.2.

P7.15 Consider a modelling problem where the plant is a 16-tap transversal filter. The plant output is contaminated with an additive white noise, $e_o(n)$, with variance

$\sigma_o^2 = 10^{-4}$. The plant input is generated by passing a unit variance white process through a colouring filter. Here, we consider the following choices of the noise colouring filter:

$$H_1(z) = 0.1 + 0.2z^{-1} + 0.3z^{-2} + 0.4z^{-3} + 0.4z^{-4} + 0.2z^{-5} + 0.1z^{-6},$$

$$H_2(z) = 0.1 - 0.2z^{-1} - 0.3z^{-2} + 0.4z^{-3} + 0.4z^{-4} - 0.2z^{-5} - 0.1z^{-6},$$

and

$$H_3(z) = 0.1 - 0.2z^{-1} + 0.3z^{-2} - 0.4z^{-3} + 0.4z^{-4} - 0.2z^{-5} + 0.1z^{-6}.$$

Note that the first two filters are those that were used in Section 7.6.4, to obtain the results of Table 7.2. We also note that the outputs of $H_1(z)$ and $H_2(z)$ are lowpass and bandpass processes, respectively (see Figures 7.6 and 7.7). The colouring filter $H_3(z)$ generates a highpass process.

Develop a program (or a set of programs) to study the convergence behaviour of the TDLMS algorithm for these choices of input and various choices of transforms. Examine your results and see how consistent these are with the general conclusions of Section 7.6.

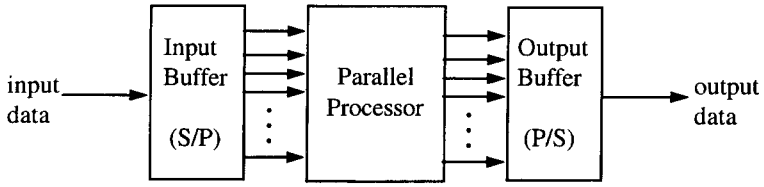
8

Block Implementation of Adaptive Filters

There are certain applications of signal processing that require adaptive filters the lengths of which exceed a few hundred or even a few thousand taps. For instance, to prevent the return of speaker echo to the far end of the telephone line, in the application of hand-free telephony, the use of an acoustic echo canceller the length of which exceeds a few thousand taps is not uncommon. Other applications, such as active noise control and the equalization of some communication channels, may also require adaptive filters with exceedingly long lengths. In such applications we find that even the conventional LMS algorithm, which is known for its simplicity, is computationally expensive to implement.

In this chapter we show how block processing of the data samples can significantly reduce the computational complexity of adaptive filters. In *block processing* (or *block implementation*), a block of samples of the filter input and desired output are collected and then processed together to obtain a block of output samples. Thus, the process involves serial to parallel conversion of the input data, parallel processing of the collected data, and parallel to serial conversion of the generated output data. This is illustrated in Figure 8.1. The computational complexity of the adaptive filter can then be reduced significantly through elegant parallel processing of the data samples. We note that the parallel processing involved in Figure 8.1 is repeated only after the collection of every block of data samples. Thus, a good measure of the computational complexity in a block processing system is given by the number of operations required to process one block of data divided by the block length. We may then note that the sharing of the processing time among the samples in each block is the key to achieving high computational efficiency.

In this chapter we discuss an efficient technique for block processing of data samples in the adaptive filtering context. This involves a special implementation of the LMS algorithm which is called the *block* LMS (BLMS). We introduce a computationally efficient implementation of the BLMS algorithm in the frequency domain. This is called the *fast* BLMS (FBLMS) algorithm. The high computational efficiency of the FBLMS algorithm is achieved by employing the following result from the theory of digital signal processing. Linear convolution of time domain sequences can be efficiently implemented using frequency domain processing. In particular, the linear convolution of an indefinite



S/P: serial-to-parallel
P/S: parallel-to-serial

Figure 8.1 Schematic of a block processing system

length sequence, $x(n)$, with a finite length sequence, $h(n)$ (which may be that of the impulse response of a FIR filter), is obtained by partitioning $x(n)$ into a set of overlapping finite duration blocks, finding the circular convolution of $h(n)$ (appended with some extra zeros) with these blocks, and then choosing the portions of the circular convolutions that match the desired linear convolution samples. The circular convolutions can be very efficiently performed in the frequency domain, using the properties of the discrete Fourier transform (DFT).

Throughout this chapter we adopt the following notations. As in the previous chapters, bold lower-case letters represent vectors, bold upper-case letters denote matrices, and non-bold lower-case letters represent scalars. As before, we use n as the time (sample) index. The letter k is reserved for block index. The subscript $\mathcal{F}$ is used to refer to frequency domain signals, e.g. the DFT of the time domain vector $\mathbf{x}$ is denoted as $\mathbf{x}_{\mathcal{F}}$. In the derivations that follow we frequently need to extend the dimensions of vectors and matrices to some certain dimensions by appending zeros. We use $\mathbf{0}$ (in bold) to refer to zero vectors and zero matrices and the dimensions of these zero vectors and/or matrices will be clear from the context.

Our discussion in this chapter is limited to the case where the filter input, $x(n)$, and the desired output, $d(n)$, are real-valued processes. However, we note that the frequency domain equivalent of these processes is complex-valued and hence the LMS recursion that is used is the complex LMS algorithm.

8.1 Block LMS Algorithm

The conventional LMS algorithm that was introduced in Chapter 6 uses the following recursion to adjust the tap weights of an adaptive filter:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n) \tag{8.1}$$

where $\mathbf{x}(n) = [x(n)\ x(n-1)\ \ldots\ x(n-N+1)]^{\mathrm{T}}$ and $\mathbf{w}(n) = [w_0(n)\ w_1(n)\ \ldots\ w_{N-1}(n)]^{\mathrm{T}}$ are the column vectors consisting of the filter tap inputs and tap weights, respectively, $e(n) = d(n) - y(n)$ is the output error, $d(n)$ and $y(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n)$ are the desired and actual output of the filter, respectively, and μ is the step-size parameter. We also recall that the conventional LMS algorithm is a stochastic implementation of the steepest-descent method using the instantaneous gradient vector

$$\nabla_{\mathbf{w}} e^2(n) = -2e(n)\mathbf{x}(n). \tag{8.2}$$

The block LMS (BLMS) algorithm works on the basis of the following strategy. The filter tap weights are updated once after the collection of every block of data samples. The gradient vector used to update the filter tap weights is an average of the instantaneous gradient vectors of the form (8.2) which are calculated during the current block. Using k to denote the block index, the BLMS recursion is obtained as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu_{\mathrm{B}} \frac{\sum_{i=0}^{L-1} e(kL+i)\mathbf{x}(kL+i)}{L}, \tag{8.3}$$

where L is the block length and μ_{B} is the algorithm step-size parameter. We also note that for the computation of the output error samples $e(kL+i) = d(kL+i) - y(kL+i)$, for $i = 0, 1, \ldots, L-1$, the output samples $y(kL+i) = \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(kL+i)$ are calculated using the update of the filter tap-weight vector, $\mathbf{w}(k)$, from the previous block.

The derivations presented in the following sections make use of, to a large extent, the vector formulation of the BLMS algorithm. Hence, we now present this formulation.

Define the matrix

$$\mathbf{X}(k) = [\mathbf{x}(kL) \;\; \mathbf{x}(kL+1) \;\; \ldots \;\; \mathbf{x}(kL+L-1)]^{\mathrm{T}}, \tag{8.4}$$

and the column vectors

$$\mathbf{d}(k) = [d(kL) \;\; d(kL+1) \;\; \ldots \;\; d(kL+L-1)]^{\mathrm{T}}, \tag{8.5}$$

$$\mathbf{y}(k) = [y(kL) \;\; y(kL+1) \;\; \ldots \;\; y(kL+L-1)]^{\mathrm{T}}, \tag{8.6}$$

$$\mathbf{e}(k) = [e(kL) \;\; e(kL+1) \;\; \ldots \;\; e(kL+L-1)]^{\mathrm{T}}, \tag{8.7}$$

and note that

$$\mathbf{y}(k) = \mathbf{X}(k)\mathbf{w}(k) \tag{8.8}$$

and

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{y}(k). \tag{8.9}$$

We also note that

$$\sum_{i=0}^{L-1} e(kL+i)\mathbf{x}(kL+i) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{e}(k). \tag{8.10}$$

Substituting (8.10) in (8.3) we obtain

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\frac{\mu_{\mathrm{B}}}{L}\mathbf{X}^{\mathrm{T}}(k)\mathbf{e}(k). \tag{8.11}$$

Equations (8.8), (8.9) and (8.11), which correspond to filtering, error estimation and tap-weight vector updating, respectively, define one iteration of the BLMS algorithm.

On the basis of our background from the method of steepest-descent and, also, the conventional LMS algorithm, the following comments may be made intuitively:

1. The convergence behaviour of the BLMS algorithm is governed by the eigenvalues of the correlation matrix $\mathbf{R} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]$. This follows from the fact that similar to the conventional LMS algorithm, the BLMS algorithm is also a stochastic implementation of the steepest-descent method.
2. The BLMS algorithm has N modes of convergence which are characterized by the time constants

$$\tau_{B,i} = \frac{1}{4\mu_{\mathrm{B}}\lambda_i}, \quad \text{for} \quad i = 0, 1, \ldots, N-1, \tag{8.12}$$

   where the λ_is are the eigenvalues of the correlation matrix $\mathbf{R}$. These time constants are in the unit of iteration (block) interval.
3. Averaging the instantaneous (stochastic) gradient vectors as was done in the BLMS algorithm results in gradient vectors with a lower variance, as compared with that in the conventional LMS algorithm. This allows the use of a larger step-size parameter for the BLMS algorithm compared to the conventional LMS algorithm. For block lengths, L, comparable or less than the filter length, N, and small misadjustments, in the range of 10% or less, misadjustment, $\mathcal{M}_{\mathrm{B}}$, of the BLMS algorithm can be approximated by the following expression:

$$\mathcal{M}_{\mathrm{B}} \approx \frac{\mu_{\mathrm{B}}}{L}\mathrm{tr}[\mathbf{R}]. \tag{8.13}$$

   This result is derived in Appendix 8A.

   Comparing (8.13) with (6.63), and letting $\mathcal{M}_{\mathrm{B}} = \mathcal{M}$, where $\mathcal{M}$ denotes the misadjustment of the conventional LMS algorithm, we obtain

$$\mu_{\mathrm{B}} = L\mu, \tag{8.14}$$

   where μ is the step-size parameter of the conventional LMS algorithm. Substituting (8.14) in (8.12), we get

$$\tau_{B,i} = \frac{1}{4L\mu\lambda_i} \text{ block interval} \tag{8.15}$$

$$= \frac{1}{4\mu\lambda_i} \text{ sample interval.} \tag{8.16}$$

   Comparing this result with (6.33) and recalling that the time constants associated with the conventional LMS algorithm are in sample intervals, we conclude that the convergence behaviour of BLMS and conventional LMS algorithms are the same.

The following example illustrates the above remarks.

Example 8.1

Let us consider the modelling problem discussed in Section 6.4.1 and use the signal colouring filter $H_1(z)$ of (6.79) to generate the input process, $x(n)$. Figure 8.2 shows the results of

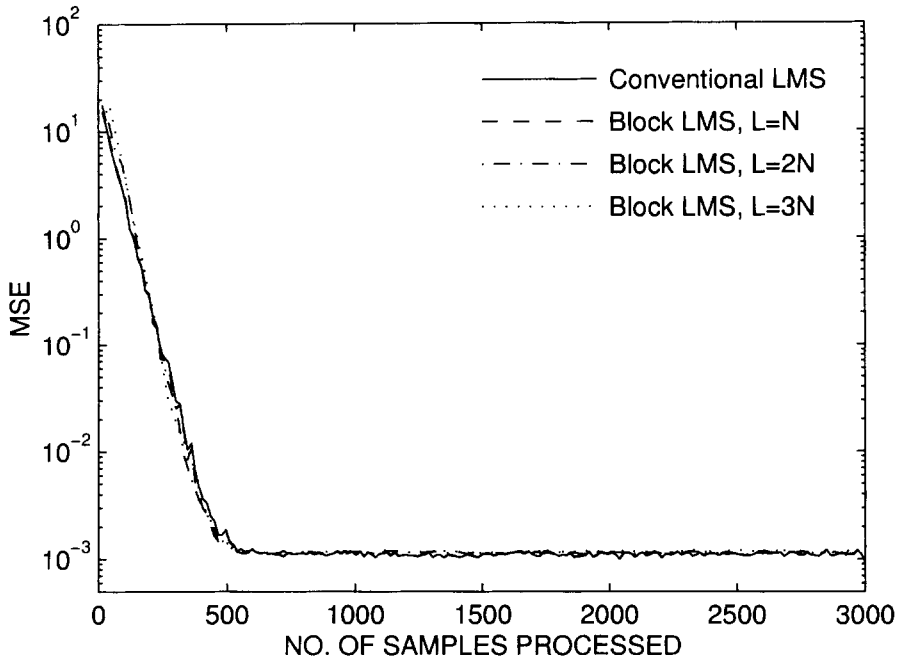


Figure 8.2 Convergence behaviour of the BLMS algorithm for various values of the block length, L. Results of the conventional LMS algorithm are also shown for comparison. The step-size parameters μ and μ_B are selected based on equations \protect (6.63) and (8.13), respectively, for 10% misadjustment

simulations that compare the conventional LMS algorithm and the BLMS algorithm, for different choices of the block length, L. The results presented here are based on an ensemble average of 100 independent runs for each plot. The step-size parameters μ and μ_B have been selected according to (6.63) and (8.13), respectively, for 10% misadjustment. We note that the difference between the various learning curves in Figure 8.2 is negligible. This confirms the theoretical predictions made above which suggest that the BLMS and conventional LMS algorithms perform about the same.

The program used to generate the results of Figures 8.2 is available on the accompanying diskette. It is called 'blk_mdlg.m'. The reader is encouraged to try this program for different choices of misadjustment and block length to study the effect of variations of these parameters on the behaviour of the BLMS algorithm.

8.2 Mathematical Background

The mathematical and signal processing tools required for the rest of this chapter are briefly reviewed in this section. In particular, we discuss how time domain linear convolutions can be efficiently performed using discrete Fourier transform (Oppenheim and Schafer, 1975, 1989). We also introduce circular matrices and review some of their properties that are relevant to our study of BLMS algorithms.

8.2.1 Linear convolution using the discrete Fourier transform

We consider the filtering of a sequence $x(n)$ through a FIR filter with coefficients $w_0, w_1, \ldots, w_{N-1}$. This involves computation of the linear convolution

$$y(n) = \sum_{i=0}^{N-1} w_i x(n-i). \tag{8.17}$$

This process requires N multiplications and $N-1$ additions for computing every sample of the output, $y(n)$. When N is large, the samples of $y(n)$ can be obtained with a reduced number of multiplications and additions, as discussed below.

Let us define the column vector $\tilde{\mathbf{x}}(k)$ of length $N' = N + L - 1$ as

$$\tilde{\mathbf{x}}(k) = [x(kL-N+1) \;\; x(kL-N+2) \;\; \ldots \;\; x(kL+L-1)]^{\mathrm{T}} \tag{8.18}$$

and $\tilde{\mathbf{w}}(k)$ of length N' as

$$\tilde{\mathbf{w}}(k) = \begin{bmatrix} \mathbf{w}(k) \\ \mathbf{0} \end{bmatrix}, \tag{8.19}$$

where $\mathbf{w}(k) = [w_0(k) \;\; w_2(k) \;\; \ldots \;\; w_{N-1}(k)]^{\mathrm{T}}$ is the filter tap-weight vector, and $\mathbf{0}$ refers to a column vector consisting of $L-1$ zeros. In order to maintain uniformity in the derivations of the subsequent sections, the block index k has been added to the filter tap weights, indicating that the weights vary only from block to block, as happens in the implementation of the BLMS algorithm.

From the properties of the DFT we know that the circular convolution of $\tilde{\mathbf{w}}(k)$ and $\tilde{\mathbf{x}}(k)$ can be obtained by transforming both vectors to their respective frequency domain equivalents (using the DFT), performing an element-by-element multiplication on the transformed samples, and transforming the result back to the time domain (using the inverse DFT (IDFT)). This process can be efficiently implemented by using the FFT and inverse FFT (IFFT) algorithms. Examining the circular convolution of $\tilde{\mathbf{w}}(k)$ and $\tilde{\mathbf{x}}(k)$ reveals that only the last L elements of the result coincide with the corresponding elements of the linear convolution (8.17); see Oppenheim and Schafer (1975) for example.[1] The rest of elements of the circular convolution do not provide any useful result, since the elements of $\tilde{\mathbf{x}}(k)$ are wrapped around and are not in the right order as required by the linear convolution (8.17). The computation of the circular convolution of $\tilde{\mathbf{w}}(k)$ and $\tilde{\mathbf{x}}(k)$ and the wraparound phenomenon are summarized in equation (8.20) on p. 253.

In (8.20) the elements represented by asterisks correspond to circular convolution results which do not coincide with linear convolution samples, as required by (8.17).

[1] In the original derivation of the FBLMS algorithm by Ferrara (1980), and most of the subsequent publications on this, the block length, L, is chosen equal to the filter length, N. Also, the column vector $\mathbf{0}$ in (8.19) has been assumed to be of length $L = N$, and not $L-1$, as we assume here.

$$\begin{bmatrix} * \\ * \\ \vdots \\ * \\ y(kL) \\ y(kL+1) \\ \vdots \\ y(kL+L-1) \end{bmatrix} = \begin{bmatrix} x(kL-N+1) & x(kL+L-1) & x(kL+L-2) & \dots & x(kL-N+2) \\ x(kL-N+2) & x(kL-N+1) & x(kL+L-1) & \dots & x(kL-N+3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x(kL-1) & x(kL-2) & x(kL-3) & \dots & x(kL) \\ x(kL) & x(kL-1) & x(kL-2) & \dots & x(kL+1) \\ x(kL+1) & x(kL) & x(kL-1) & \dots & x(kL+2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x(kL+L-1) & x(kL+L-2) & x(kL+L-3) & \dots & x(kL-N+1) \end{bmatrix} \begin{bmatrix} w_0(k) \\ w_1(k) \\ \vdots \\ w_{N-2}(k) \\ w_{N-1}(k) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{8.20}$$

Careful examination of the summations related to these elements reveals that the input samples experience some discontinuity in their order. For example, a jump from $x(kL - N + 1)$ to $x(kL + L - 1)$ is observed in the first row of the data matrix on the right-hand side of (8.20). When such discontinuities overlap with the non-zero portion of $\tilde{\mathbf{w}}(k)$, then the corresponding output samples will not correspond to valid linear convolution samples.

The procedure explained by (8.20) is commonly known as the *overlap–save method.* This name reflects the fact that in each block of input, $\tilde{\mathbf{x}}(k)$ consists of L new samples and $N - 1$ overlapped samples from the previous block(s). Another equally efficient method for the computation of linear convolutions using DFT is the *overlap–add method.* However, the overlap–add method has been found to be computationally less efficient than the overlap–save method when applied to the implementation of the BLMS algorithm. Noting this, we do not discuss the overlap–add method in this book.

8.2.2 Circular matrices

Circular matrices are used extensively in the derivation and analysis of the fast BLMS (FBLMS) algorithm. Hence, it is very useful as well as necessary to have a good understanding of the properties of these matrices before we start our discussion of the FBLMS algorithm.

Consider the $M \times M$ circular matrix

$$\mathbf{A}_c = \begin{bmatrix} a_0 & a_{M-1} & a_{M-2} & \cdots & a_1 \\ a_1 & a_0 & a_{M-1} & \cdots & a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{M-2} & a_{M-3} & a_{M-4} & \cdots & a_{M-1} \\ a_{M-1} & a_{M-2} & a_{M-3} & \cdots & a_0 \end{bmatrix}. \tag{8.21}$$

Clearly, the name 'circular' refers to the fact that each row (column) of $\mathbf{A}_c$ is obtained by circularly shifting the previous row (column) by one element. A special property of circular matrices that is extensively used in the following sections is that such matrices are diagonalized by DFT matrices. That is, if $\mathcal{F}$ is the $M \times M$ DFT matrix defined as

$$\mathcal{F} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-j2\pi/M} & e^{-j4\pi/M} & \cdots & e^{-j2\pi(M-1)/M} \\ 1 & e^{-j4\pi/M} & e^{-j8\pi/M} & \cdots & e^{-j4\pi(M-1)/M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-j2\pi(M-1)/M} & e^{-j4\pi(M-1)/M} & \cdots & e^{-j2\pi(M-1)^2/M} \end{bmatrix}, \tag{8.22}$$

then

$$\mathcal{A}_{\mathcal{F}} = \mathcal{F}\mathbf{A}_c\mathcal{F}^{-1} \tag{8.23}$$

is a diagonal matrix. Furthermore, the diagonal elements of $\mathcal{A}_{\mathcal{F}}$ correspond to the DFT of the first column of $\mathbf{A}_c$. In matrix notation, this may be written as

$$\mathcal{A}_{\mathcal{F}} = \text{diag}[\mathbf{a}_{\mathcal{F}}], \tag{8.24}$$

where $\mathbf{a}_{\mathcal{F}} = \mathcal{F}\mathbf{a}$, $\mathbf{a} = [a_0 \ a_1 \ \dots \ a_{M-1}]^T$ is the first column of $\mathbf{A}_c$, and $\text{diag}[\mathbf{a}_{\mathcal{F}}]$ denotes the diagonal matrix consisting of the elements of $\mathbf{a}_{\mathcal{F}}$. This can be proved as follows.

Since $\mathcal{F}$ is a DFT matrix, recall that

$$\mathcal{F}^{-1} = \frac{1}{M}\mathcal{F}^*, \tag{8.25}$$

where M is the length of DFT and an asterisk denotes complex conjugation. In other words, the lth column of $\mathcal{F}^{-1}$ can be given as

$$\mathbf{g}_l = \frac{1}{M}\mathbf{f}_l^*, \tag{8.26}$$

where $\mathbf{f}_l$ is the lth column of the DFT matrix $\mathcal{F}$ given by

$$\mathbf{f}_l = [1 \ e^{-j2\pi l/M} \ e^{-j4\pi l/M} \ \dots \ e^{-j2(M-1)\pi l/M}]^T. \tag{8.27}$$

Next, by direct insertion, one can easily show that (Problem P8.2)

$$\mathbf{A}_c\mathbf{g}_l = a_{\mathcal{F},l}\mathbf{g}_l, \quad \text{for } l = 0, 1, \dots, M-1, \tag{8.28}$$

where $a_{\mathcal{F},l} = \sum_{i=0}^{M-1} a_i e^{-j2\pi li/M}$ is the lth element of $\mathbf{a}_{\mathcal{F}}$. Using (8.24), the M equations in (8.28) may be put together to obtain

$$\mathbf{A}_c\mathcal{F}^{-1} = \mathcal{F}^{-1}\mathcal{A}_{\mathcal{F}}. \tag{8.29}$$

Premultiplying (8.29) on both sides by $\mathcal{F}$ gives (8.23).

Another important result of the circular matrices which will be useful for our later application is derived next. Applying Hermitian transposition on both sides of (8.23), we obtain

$$\mathcal{A}_{\mathcal{F}}^H = \mathcal{F}^{-H}\mathbf{A}_c^H\mathcal{F}^H, \tag{8.30}$$

where $\mathcal{F}^{-H}$ is the short-hand notation for $(\mathcal{F}^{-1})^H$. Since $\mathcal{A}_{\mathcal{F}}$ is diagonal, $\mathcal{A}_{\mathcal{F}}^H = \mathcal{A}_{\mathcal{F}}^*$. Furthermore, from (8.22) and (8.25), $\mathcal{F}^{-H} = (1/M)\mathcal{F}$ and $\mathcal{F}^H = M\mathcal{F}^{-1}$ since $\mathcal{F}^T = \mathcal{F}$. Using these in (8.30) we get

$$\mathcal{A}_{\mathcal{F}}^* = \mathcal{F}\mathbf{A}_c^H\mathcal{F}^{-1}. \tag{8.31}$$

When elements of $\mathbf{A}_c$ are real-valued, $\mathbf{A}_c^H = \mathbf{A}_c^T$ and thus (8.31) may be written as

$$\mathcal{A}_{\mathcal{F}}^* = \mathcal{F}\mathbf{A}_c^T\mathcal{F}^{-1}. \tag{8.32}$$

8.2.3 Window matrices and matrix formulation of the overlap–save method

Let us define the $N' \times N'$ circular matrix, for $N' = L + N - 1$, as

$$\mathbf{X}_c(k) = \begin{bmatrix} x(kL-N+1) & x(kL+L-1) & x(kL+L-2) & \dots & x(kL-N+2) \\ x(kL-N+2) & x(kL-N+1) & x(kL+L-1) & \dots & x(kL-N+3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x(kL+L-1) & x(kL+L-2) & x(kL+L-3) & \dots & x(kL-N+1) \end{bmatrix}. \tag{8.33}$$

We note that this is nothing but the data matrix on the right-hand side of (8.20).

We also define the length N' column vector

$$\tilde{\mathbf{y}}(k) = \begin{bmatrix} \mathbf{0} \\ \mathbf{y}(k) \end{bmatrix}, \tag{8.34}$$

where $\mathbf{y}(k)$, as defined in (8.6), is the column vector consisting of the output samples of the kth block, and $\mathbf{0}$ is the length $N-1$ zero vector. Let us denote by $\mathbf{y}_c(k)$ the column vector that appears on the left-hand side of (8.20) and note that $\tilde{\mathbf{y}}(k)$ can be obtained from $\mathbf{y}_c(k)$ by substituting all the $*$ elements in the latter with zeros. This substitution can be written in the form of a matrix–vector product as

$$\tilde{\mathbf{y}}(k) = \mathbf{P}_{0,L}\mathbf{y}_c(k), \tag{8.35}$$

where $\mathbf{P}_{0,L}$ is the $N' \times N'$ *windowing matrix* defined as

$$\mathbf{P}_{0,L} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_L \end{bmatrix}, \tag{8.36}$$

with $\mathbf{I}_L$ being the $L \times L$ identity matrix and the $\mathbf{0}$s are zero matrices with appropriate dimensions. Using (8.20), (8.19) and the above definitions, we obtain

$$\tilde{\mathbf{y}}(k) = \mathbf{P}_{0,L}\mathbf{X}_c(k)\tilde{\mathbf{w}}(k). \tag{8.37}$$

Implementation of (8.37) in the frequency domain can now be obtained simply by noting that (8.37) may be written as

$$\tilde{\mathbf{y}}(k) = \mathbf{P}_{0,L}\mathcal{F}^{-1}\mathcal{F}\mathbf{X}_c(k)\mathcal{F}^{-1}\mathcal{F}\tilde{\mathbf{w}}(k), \tag{8.38}$$

where $\mathcal{F}$ is the $N' \times N'$ DFT matrix. Next, define

$$\mathbf{w}_{\mathcal{F}}(k) = \mathcal{F}\tilde{\mathbf{w}}(k) \tag{8.39}$$

and

$$\mathcal{X}_{\mathcal{F}}(k) = \mathcal{F}\mathbf{X}_c(k)\mathcal{F}^{-1}, \tag{8.40}$$

and note that $\mathcal{X}_{\mathcal{F}}(k)$ is the diagonal matrix consisting of the elements of the DFT of the first column of $\mathbf{X}_c(k)$, since the latter is a circular matrix. We also note that the first column of $\mathbf{X}_c(k)$ is the input vector $\tilde{\mathbf{x}}(k)$, as defined in (8.18). Using (8.39) and (8/40) in (8.38) we obtain

$$\tilde{\mathbf{y}}(k) = \mathbf{P}_{0,L}\mathcal{F}^{-1}\mathcal{X}_{\mathcal{F}}(k)\mathbf{w}_{\mathcal{F}}(k). \tag{8.41}$$

This equation has the following interpretation. Since $\mathcal{X}_{\mathcal{F}}(k)$ is diagonal, $\mathcal{X}_{\mathcal{F}}(k)\mathbf{w}_{\mathcal{F}}(k)$ is nothing but the element-by-element multiplication of the filter input and its coefficients in the frequency domain. This gives the output samples of the filter in the frequency domain. Premultiplication of this result by $\mathcal{F}^{-1}$ converts the frequency domain samples of the output to the time domain. Furthermore, premultiplying the result by the windowing matrix $\mathbf{P}_{0,L}$ results in selecting only those samples that coincide with the required linear convolution samples.

With the background developed in this section, we are now ready to proceed with the derivation and analysis of the FBLMS algorithm.

8.3 The FBLMS Algorithm

The FBLMS algorithm, as mentioned in the introduction, is nothing but a fast (numerically efficient) implementation of the BLMS algorithm in the frequency domain.

Equation (8.41) corresponds to the filtering part of the FBLMS algorithm. Element-by-element multiplication of the frequency domain samples of the input and filter coefficients is followed by an IDFT and a proper windowing of the result to obtain the output vector $\tilde{\mathbf{y}}(k)$, in extended form, as defined by (8.34). The vector of desired outputs, in extended form, is defined as

$$\tilde{\mathbf{d}}(k) = \begin{bmatrix} \mathbf{0} \\ \mathbf{d}(k) \end{bmatrix}, \tag{8.42}$$

where $\mathbf{d}(k)$ is defined by (8.5) and $\mathbf{0}$ is the $N-1$ element zero column vector. We also define the extended error vector

$$\tilde{\mathbf{e}}(k) = \tilde{\mathbf{d}}(k) - \tilde{\mathbf{y}}(k). \tag{8.43}$$

To obtain the frequency domain equivalent of the recursion (8.11), we replace $\mathbf{w}(k)$ and $\mathbf{e}(k)$ by their extended versions and note that (8.11) may also be written as

$$\tilde{\mathbf{w}}(k+1) = \tilde{\mathbf{w}}(k) + 2\mu\mathbf{P}_{N,0}\mathbf{X}_c^{\mathrm{T}}(k)\tilde{\mathbf{e}}(k), \tag{8.44}$$

where $\mathbf{X}_c(k)$ is the circular matrix of samples of the filter input as defined by (8.33), $\mu = \mu_B/L$, and

$$\mathbf{P}_{N,0} = \begin{bmatrix} \mathbf{I}_N & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{8.45}$$

is an $N' \times N'$ windowing matrix which ensures that the last $L-1$ elements of the updated weight vector $\tilde{\mathbf{w}}(k+1)$ remain equal to zero after each iteration of (8.44). The fact that (8.11) and (8.44) are equivalent can easily be shown by substituting for the vectors and matrices in (8.44) and expanding the result (Problem P8.4).

Conversion of the recursion (8.44) to its frequency domain equivalent can be done by premultiplying it on both sides by the DFT matrix $\mathcal{F}$ and using the identity $\mathcal{F}^{-1}\mathcal{F} = \mathbf{I}$, to obtain

$$\mathbf{w}_{\mathcal{F}}(k+1) = \mathbf{w}_{\mathcal{F}}(k) + 2\mu\mathcal{F}\mathbf{P}_{N,0}\mathcal{F}^{-1}\mathcal{F}\mathbf{X}_c^{\mathrm{T}}(k)\mathcal{F}^{-1}\mathcal{F}\tilde{\mathbf{e}}(k). \tag{8.46}$$

Using (8.40) and the identity (8.32), (8.46) can be written as

$$\mathbf{w}_{\mathcal{F}}(k+1) = \mathbf{w}_{\mathcal{F}}(k) + 2\mu\mathcal{P}_{N,0}\mathcal{X}_{\mathcal{F}}^{*}(k)\mathbf{e}_{\mathcal{F}}(k), \tag{8.47}$$

where $\mathbf{e}_{\mathcal{F}}(k) = \mathcal{F}\tilde{\mathbf{e}}(k)$ and

$$\mathcal{P}_{N,0} = \mathcal{F}\mathbf{P}_{N,0}\mathcal{F}^{-1}. \tag{8.48}$$

Equations (8.41), (8.43) and (8.47) are the three steps required to complete each iteration of the FBLMS algorithm: namely, filtering, error estimation and tap-weight adaptation, respectively. Figure 8.3 depicts a block diagram of the FBLMS algorithm,

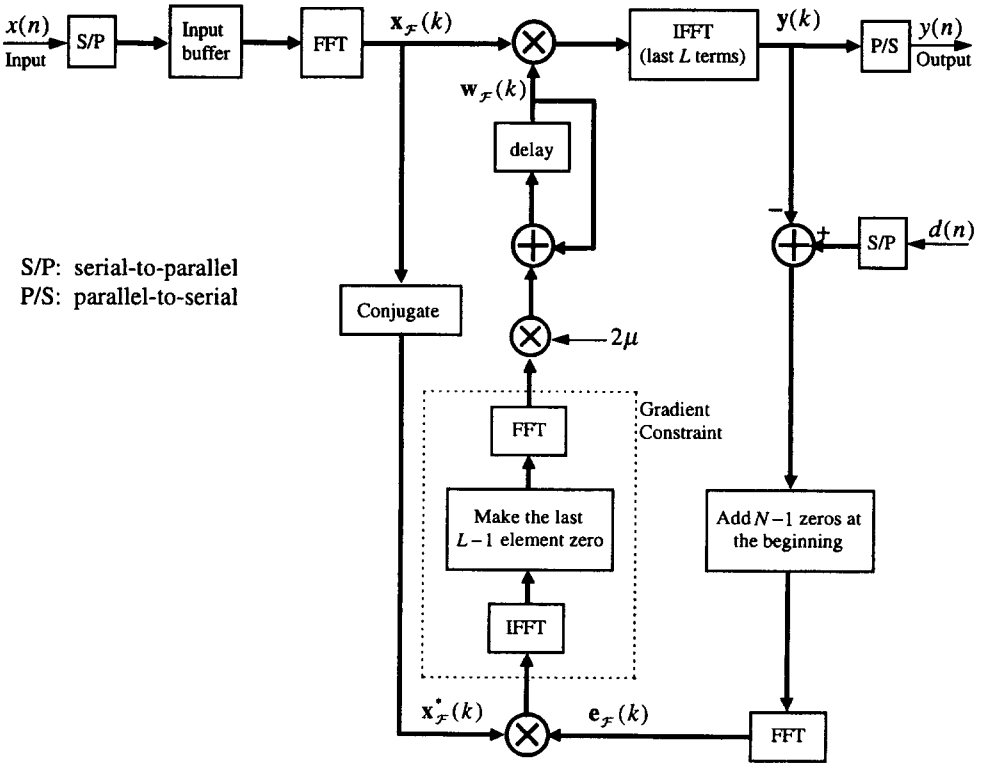


Figure 8.3 Implementation of the FBLMS algorithm

which shows how these steps are realized efficiently. The input samples are collected in an input buffer whose output is the vector $\tilde{\mathbf{x}}(k)$, consisting of L new samples and $N-1$ samples from the previous block(s). The vector $\tilde{\mathbf{x}}(k)$ is converted to the frequency domain and multiplied by the associated tap-weight vector, $\mathbf{w}_{\mathcal{F}}(k)$, on an element-by-element basis. This gives the samples of the filter output in the frequency domain which

this result correspond to the output samples of the current block and are sent to the output buffer as well as the error estimation section. The error vector, $\mathbf{e}(n)$, which consists of L elements is extended to the length of $N+L-1$ by appending $N-1$ zeros at its beginning and converted to the frequency domain using an FFT algorithm. An element-by-element multiplication of the error and conjugate of the input samples is performed in the frequency domain and the result is used to update the filter tap weights. Premultiplication of the gradient vector $\mathcal{X}^*_{\mathcal{F}}(k)\mathbf{e}_{\mathcal{F}}(k)$ by $\mathcal{P}_{N,0}$ is necessary to ensure that the last $L-1$ elements of the time domain equivalent of the tap-weight vector $\mathbf{w}_{\mathcal{F}}(k)$ are constrained to zero (see (8.19)). This constraining operation is implemented by converting the gradient vector $\mathcal{X}^*_{\mathcal{F}}(k)\mathbf{e}_{\mathcal{F}}(k)$ to the time domain, making the last $L-1$ elements zero, and converting back to the frequency domain, as shown in Figure 8.3.

8.3.1 Constrained and unconstrained FBLMS algorithms

The fact that the first $N-1$ elements of $\tilde{\mathbf{d}}(k)$ are all zero implies that $\tilde{\mathbf{d}}(k) = \mathbf{P}_{0,L}\tilde{\mathbf{d}}(k)$. Using this in (8.50) we obtain

$$\begin{aligned}\tilde{\mathbf{e}}(k) &= \mathbf{P}_{0,L}(\tilde{\mathbf{d}}(k) - \boldsymbol{\mathcal{F}}^{-1}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)\mathbf{w}_{\mathcal{F}}(k)) \\ &= \mathbf{P}_{0,L}\boldsymbol{\mathcal{F}}^{-1}(\boldsymbol{\mathcal{F}}\tilde{\mathbf{d}}(k) - \boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)\mathbf{w}_{\mathcal{F}}(k)).\end{aligned} \tag{8.51}$$

Premultiplying (8.51) on both sides by $\boldsymbol{\mathcal{F}}$, we get

$$\mathbf{e}_{\mathcal{F}}(k) = \boldsymbol{\mathcal{P}}_{0,L}(\mathbf{d}_{\mathcal{F}}(k) - \boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)\mathbf{w}_{\mathcal{F}}(k)) \tag{8.52}$$

where $\mathbf{d}_{\mathcal{F}}(k) = \boldsymbol{\mathcal{F}}\tilde{\mathbf{d}}(k)$ and

$$\boldsymbol{\mathcal{P}}_{0,L} = \boldsymbol{\mathcal{F}}\mathbf{P}_{0,L}\boldsymbol{\mathcal{F}}^{-1}. \tag{8.53}$$

Substituting (8.52) in (8.49), we obtain

$$\mathbf{w}_{\mathcal{F}}(k+1) = \mathbf{w}_{\mathcal{F}}(k) + 2\mu\boldsymbol{\mathcal{X}}^{*}_{\mathcal{F}}(k)\left[\boldsymbol{\mathcal{P}}_{0,L}(\mathbf{d}_{\mathcal{F}}(k) - \boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)\mathbf{w}_{\mathcal{F}}(k))\right]. \tag{8.54}$$

Next, we define the tap-weight error vector

$$\mathbf{v}_{\mathcal{F}}(k) = \mathbf{w}_{\mathcal{F}}(k) - \mathbf{w}_{\mathrm{o},\mathcal{F}}, \tag{8.55}$$

where $\mathbf{w}_{\mathrm{o},\mathcal{F}}$ is the optimum value of the filter tap-weight vector in the frequency domain. Using (8.55) in (8.54) we obtain, after some simple manipulation,

$$\mathbf{v}_{\mathcal{F}}(k+1) = (\mathbf{I} - 2\mu\boldsymbol{\mathcal{X}}^{*}_{\mathcal{F}}(k)\boldsymbol{\mathcal{P}}_{0,L}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k))\mathbf{v}_{\mathcal{F}}(k) + 2\mu\boldsymbol{\mathcal{X}}^{*}_{\mathcal{F}}(k)\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k), \tag{8.56}$$

where $\mathbf{e}_{\mathrm{o},\mathcal{F}}(k)$ is the optimum error vector obtained when $\mathbf{w}_{\mathcal{F}}(k)$ is replaced by $\mathbf{w}_{\mathrm{o},\mathcal{F}}$.

Now, if we use the *independence assumption* and follow the same procedure as in Section 6.2, we will find that the convergence of the unconstrained FBLMS algorithm is controlled by the eigenvalues of the matrix

$$\boldsymbol{\mathcal{R}}^{\mathrm{u}}_{xx} = \mathrm{E}[\boldsymbol{\mathcal{X}}^{*}_{\mathcal{F}}(k)\boldsymbol{\mathcal{P}}_{0,L}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)]. \tag{8.57}$$

The matrix $\boldsymbol{\mathcal{R}}^{\mathrm{u}}_{xx}$ may be evaluated as follows. Substituting (8.40) and (8.53) in (8.57), we obtain

$$\boldsymbol{\mathcal{R}}^{\mathrm{u}}_{xx} = \boldsymbol{\mathcal{F}}\mathbf{R}^{\mathrm{u}}_{xx}\boldsymbol{\mathcal{F}}^{-1}, \tag{8.58}$$

where

$$\mathbf{R}^{\mathrm{u}}_{xx} = \mathrm{E}[\mathbf{X}^{\mathrm{T}}_{\mathrm{c}}(k)\mathbf{P}_{0,L}\mathbf{X}_{\mathrm{c}}(k)]. \tag{8.59}$$

A careful examination of $\mathbf{R}^{\mathrm{u}}_{xx}$ reveals that when L and N are large and the autocorrelation function of the input process, $x(n)$, i.e. $\phi_{xx}(l)$, approaches zero for the lag values l

much smaller than L and N, $\mathbf{R}_{xx}^{\mathrm{u}}$ can be approximated by the $N' \times N'$ circular matrix whose first column is (Lee and Un, 1989)

$$\mathbf{r}_{xx}^{\mathrm{u}} = L \times [\phi_{xx}(0)\ \ \phi_{xx}(1)\ \ \ldots\ \ \phi_{xx}(l)\ \ 0\ \ \ldots\ \ 0\ \ \phi_{xx}(l)\ \ \phi_{xx}(l-1)\ \ \ldots\ \ \phi_{xx}(1)]^{\mathrm{T}}. \quad (8.60)$$

Using the properties of the circular matrices, this implies that

$$\mathcal{R}_{xx}^{\mathrm{u}} \approx L \times \mathrm{diag}(\Phi_{xx}(\mathrm{e}^{j2\pi \times 0/N'}), \Phi_{xx}(\mathrm{e}^{j2\pi/N'}), \ldots, \Phi_{xx}(\mathrm{e}^{j2\pi(N'-1)/N'})) \quad (8.61)$$

where $\Phi_{xx}(\mathrm{e}^{j\omega})$ is the power spectral density of the input process, $x(n)$. (See also Problem P8.11 for an alternative derivation of (8.61).) The samples of $\Phi_{xx}(\mathrm{e}^{j\omega})$ on the right-hand side of (8.61) are obtained by taking the DFT of the vector $\mathbf{r}_{xx}^{\mathrm{u}}/L$.

The fact that $\mathcal{R}_{xx}^{\mathrm{u}}$ is a diagonal matrix implies that its eigenvalues are equal to its diagonal elements. The diagonal elements of $\mathcal{R}_{xx}^{\mathrm{u}}$, as specified in (8.61), in turn are proportional to the samples of the power spectral density of the underlying input process. Thus, for coloured inputs, as happens with the conventional LMS algorithm, the unconstrained FBLMS algorithm will also perform poorly. The same is also true for the constrained FBLMS algorithm, since it is nothing but a fast implementation of the BLMS algorithm whose convergence behaviour was studied in Section 8.1 and found to perform very similar to the conventional LMS algorithm.

8.3.3 Step-normalization

The convergence performance of the FBLMS algorithm can be greatly improved by using individually normalized step-size parameters for each element of the tap-weight vector $\mathbf{w}_{\mathcal{F}}(k)$, rather than a common step-size parameter. This technique, known as *step-normalization*, is similar to one that was described in Chapter 7 for improving the convergence of the transform domain LMS algorithm. It is implemented by replacing the scalar step-size parameter μ by the diagonal matrix

$$\boldsymbol{\mu}(k) = \mathrm{diag}[\mu_0(k),\ \ \mu_1(j), \ldots, \mu_{N'-1}(k)], \quad (8.62)$$

where $\mu_i(k)$ is the normalized step-size parameters for the ith tap. These are obtained according to the equations

$$\mu_i(k) = \frac{\mu_{\mathrm{o}}}{\hat{\sigma}^2_{x_{\mathcal{F},i}}(k)}, \quad \text{for} \quad i = 0, 1, \ldots, N' - 1, \quad (8.63)$$

where μ_{o} is a common unnormalized step-size parameter and the $\hat{\sigma}^2_{x_{\mathcal{F},i}}(k)$s are the power estimates of the samples of the filter input in the frequency domain, the $x_{\mathcal{F},i}(k)$s. These estimates may be obtained using the following recursion:

$$\hat{\sigma}^2_{x_{\mathcal{F},i}}(k) = \beta \hat{\sigma}^2_{x_{\mathcal{F},i}}(k-1) + (1-\beta)|x_{\mathcal{F},i}(k)|^2, \quad (8.64)$$

for $i = 0, 1, \ldots, N' - 1$, where β is a constant close to, but smaller than, one.

8.3.4 Summary of the FBLMS algorithm

Using the results developed in the previous sections, Table 8.1 summarizes the FBLMS algorithm. This table is in a form which can be readily converted to an efficient program code for implementing the FBLMS algorithm. In particular, the diagonal matrix $\mathcal{X}_{\mathcal{F}}(k)$ is replaced by the vector $\mathbf{x}_{\mathcal{F}}(k)$ consisting of the diagonal elements of $\mathcal{X}_{\mathcal{F}}(k)$. Also, $\boldsymbol{\mu}(k)$ is redefined as a column vector. Furthermore, the constraining/windowing

Table 8.1 Summary of the FBLMS algorithm

Input: Tap-weight vector, $\mathbf{w}_{\mathcal{F}}(k)$,
Signal power estimates, $\hat{\sigma}^2_{x_{\mathcal{F},i}}(k-1)$s,
Extended input vector,
$\tilde{\mathbf{x}}(k) = [x(kL-N+1)\ \ x(kL-N+2)\ \ \ldots\ \ x(kL+L-1)]^{\mathrm{T}}$,
and desired output vector,
$\mathbf{d}(k) = [d(kL)\ \ d(kL+1)\ \ \ldots\ \ d(kL+L-1)]^{\mathrm{T}}$.

Output: Filter output,
$\mathbf{y}(k) = [y(kL)\ \ y(kL+1)\ \ \ldots\ \ y(kL+L-1)]^{\mathrm{T}}$,
Tap-weight vector update, $\mathbf{w}_{\mathcal{F}}(k+1)$.

1. Filtering:

$$\mathbf{x}_{\mathcal{F}}(k) = \mathrm{FFT}(\tilde{\mathbf{x}}(k))$$
$$\mathbf{y}(k) = \text{the last L elements of } \mathrm{IFFT}(\mathbf{x}_{\mathcal{F}}(k) \otimes \mathbf{w}_{\mathcal{F}}(k))$$

2. Error estimation:

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{y}(k)$$

3. Step-normalization:

$$\text{for } i = 0 \text{ to } N'-1$$
$$\hat{\sigma}^2_{x_{\mathcal{F},i}}(k) = \beta\hat{\sigma}^2_{x_{\mathcal{F},i}}(k-1) + (1-\beta)|x_{\mathcal{F},i}(k)|^2$$
$$\mu_i(k) = \mu_0/\hat{\sigma}^2_{x_{\mathcal{F},i}}(k)$$
$$\boldsymbol{\mu}(k) = [\mu_0(k)\ \ \mu_1(k)\ \ \ldots\ \ \mu_{N'-1}(k)]^{\mathrm{T}}$$

4. Tap-weight adaptation:

$$\mathbf{e}_{\mathcal{F}}(k) = \mathrm{FFT}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{e}(k)\end{bmatrix}\right)$$
$$\mathbf{w}_{\mathcal{F}}(k+1) = \mathbf{w}_{\mathcal{F}}(k) + 2\boldsymbol{\mu}(k) \otimes \mathbf{x}^*_{\mathcal{F}}(k) \otimes \mathbf{e}_{\mathcal{F}}(k)$$

5. Tap-weight constraint:

$$\mathbf{w}_{\mathcal{F}}(k+1) = FFT\left(\begin{bmatrix}\text{the first N elements of } \mathrm{IFFT}(\mathbf{w}_{\mathcal{F}}(k+1))\\ \mathbf{0}\end{bmatrix}\right)$$

Notes:

- N: filter length; L: block length; $N' = N + L - 1$.
- $\mathbf{0}$ denotes column zero vectors with appropriate length to extend vectors to the length of N'.
- $\otimes$ denotes element-by-element multiplication of vectors.
- Here, $\boldsymbol{\mu}(k)$ is defined as a column vector. This is different from the definition of $\boldsymbol{\mu}(k)$ in the text where it is defined as a diagonal matrix.
- Step 5 is applicable only for the constrained FBLMS algorithm.

operations defined by the matrices $\mathcal{P}_{0,L}$ and $\mathcal{P}_{N,0}$ are re-expressed more explicitly by replacing the unwanted elements of the corresponding vectors with zeros. We also use the terms FFT and IFFT to refer to DFT and IDFT operations. This is to emphasize that, in practice, fast Fourier transform algorithms are used to perform these operations efficiently.

In the derivations given in Section 8.3 it is assumed that the frequency domain tap-weight vector $\mathbf{w}_{\mathcal{F}}(k)$ satisfies the required time domain constraint, namely, the last $L-1$ elements of the inverse DFT of $\mathbf{w}_{\mathcal{F}}(k)$ are all zero. Thus, the constraint needs to be imposed only on the stochastic gradient vector $-2\mathcal{X}_{\mathcal{F}}^{*}(k)\mathbf{e}_{\mathcal{F}}(k)$; see (8.47). This assumption, although theoretically correct if $\mathbf{w}_{\mathcal{F}}(0)$ is initialized to a constraint-satisfying vector, may not continue to be true as the algorithm progresses. This is because the round-off noise that is added to the elements of $\mathbf{w}_{\mathcal{F}}(k+1)$ will accumulate and result in a vector that may seriously violate the constraint after some iterations. In the case of the unconstrained FBLMS algorithm, these errors are compensated by the adaptation process, since they propagate back to themselves through the unconstrained gradient vector $-2\mathcal{X}_{\mathcal{F}}^{*}(k)\mathbf{e}_{\mathcal{F}}(k)$. However, this does not happen in the case of constrained FBLMS algorithm, because the gradient vector $-2\mathcal{X}_{\mathcal{F}}^{*}(k)\mathbf{e}_{\mathcal{F}}(k)$ is constrained before being used for updating the tap weights. To resolve this problem, the tap-weight vector $\mathbf{w}_{\mathcal{F}}(k)$ should be regularly checked and constrained, as explained below.

Assume that $\mathbf{w}_{\mathcal{F}}(k)$ satisfies the required time domain constraint. That is, the last $L-1$ elements of the inverse DFT of $\mathbf{w}_{\mathcal{F}}(k)$ are all zero. This implies that

$$\mathbf{w}_{\mathcal{F}}(k) = \mathcal{P}_{N,0}\mathbf{w}_{\mathcal{F}}(k). \tag{8.65}$$

Using this in the constrained FBLMS recursion (8.47), we obtain

$$\mathbf{w}_{\mathcal{F}}(k+1) = \mathcal{P}_{N,0}[\mathbf{w}_{\mathcal{F}}(k) + 2\mu\mathcal{X}_{\mathcal{F}}^{*}(k)\mathbf{e}_{\mathcal{F}}(k)]. \tag{8.66}$$

This recursion constrains $\mathbf{w}_{\mathcal{F}}(k+1)$ after every iteration and thus prevents any accumulation of round-off noise errors. Implementation of the constrained FBLMS algorithm given in Table 8.1 is based on this recursion.

To emphasize the importance of the above-mentioned fine-tuning of the constrained FBLMS algorithm, the results of the two implementations of the constrained FBLMS algorithm (one based on the recursion (8.47) and the other based on the scheme proposed in Table 8.1) are presented in Figure 8.4 for comparison. These results are obtained by running a MATLAB program using the full precision of the floating point numbers available in the MATLAB environment. Observe that the constrained FBLMS algorithm implemented using (8.47) encounters a numerical problem, i.e. the associated MSE keeps growing after an initial convergence of the algorithm. Clearly, this problem will be more serious in situations where cost constraints require a lower precision to be used.

Before ending this section, some remarks on real- and complex-valued signal cases would be instructive. Although all the derivations in this chapter are given for real-valued signals in order to prevent unnecessary confusion, the final algorithm presented in Table 8.1 is applicable to both real- and complex-valued signals. Another point to be noted in the case of real-valued signals is that all the frequency domain vectors will be

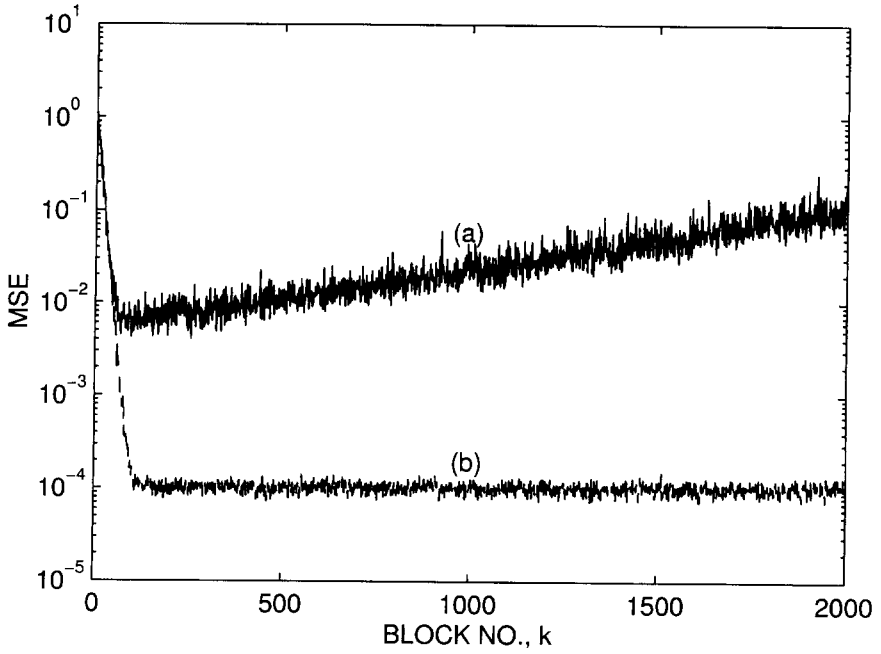


Figure 8.4 The two learning curves show the behaviour of the constrained FBLMS algorithm (a) when the constraining operation is applied on the gradient vector as in (8.47), and (b) when the constraining operation is applied on the tap-weight vector $\mathbf{w}_{\mathcal{F}}(k)$ as in Table 8.1

conjugate symmetric.[2] This implies that the first half of these frequency domain vectors contain all the necessary information and, hence, their second halves can be ignored. This reduces the computational complexity and memory requirement of the FBLMS algorithm by about 50%.

8.3.5 FBLMS misadjustment equations

Derivation of the misadjustment equations for the various implementations of the FBLMS algorithms is quite tedious and long. This is done in Appendix 8B. The derivations presented in Appendix 8B result in the following misadjustment equations:

$$\mathcal{M}^{\mathrm{c}}_{\mathrm{FBLMS}} \approx \mu N \phi_{xx}(0), \tag{8.67}$$

$$\mathcal{M}^{\mathrm{u}}_{\mathrm{FBLMS}} \approx \mu N' \phi_{xx}(0), \tag{8.68}$$

$$\mathcal{M}^{\mathrm{c},\mathcal{N}}_{\mathrm{FBLMS}} \approx \mu_{\mathrm{o}} N/N', \tag{8.69}$$

$$\mathcal{M}^{\mathrm{u},\mathcal{N}}_{\mathrm{FBLMS}} \approx \mu_{\mathrm{o}}. \tag{8.70}$$

[2] A length M vector $\mathbf{u} = [u_0 \;\; u_1 \;\; \cdots \;\; u_{M-1}]^{\mathrm{T}}$ is called conjugate symmetric when $u_i = u^*_{M-i}$, for $i = 0, 1, \cdots, \lfloor M/2 \rfloor$, where $\lfloor M/2 \rfloor$ denotes the integer part of $M/2$.

In these equations the superscripts c and u refer to the constrained and unconstrained versions of the FBLMS algorithm, respectively, and $\mathcal{N}$ indicates that the step-normalization has been applied. We also note that, similar to (6.63) and (8.13), equations (8.67)–(8.70) are valid only for misadjustment values of 10% or lower.

It can be immediately concluded from (8.67)–(8.70) that the constrained FBLMS algorithm outperforms its unconstrained counterpart, in the sense that the former results in a lower misadjustment for a given step-size parameter. Equivalently, for a given misadjustment the constrained FBLMS algorithm converges faster than its unconstrained counterpart. The difference between the two algorithms is determined by the ratio N/N' $(= N/(N + L - 1))$, which, in turn, is determined by the ratio L/N. Clearly, when $L \ll N$, $N/N' \approx 1$ and thus the difference between the constrained FBLMS algorithm and its unconstrained counterpart becomes insignificant. On the other hand, when L and N are comparable, the difference between the two algorithms will be significant.

8.3.6 Selection of the block length

Block processing of signals, in general, results in a certain time delay at the system output. In many applications this processing delay may be intolerable and hence it has to be minimized. It arises because a block of samples of input signal has to be collected before the processing of the data can begin. Consequently, the processing delay increases with block length. On the other hand, the per sample computational complexity of a block processing system varies with the block length, L. For values of L smaller than the filter length, N, per sample computational complexity of the FBLMS algorithm decreases as L increases. It reaches close to its minimum when $L \approx N$. Thus, in applications where the processing delay is not an issue, L is usually chosen close to N. The exact value of L depends on N. For a given N, one should choose L so that $N' = N + L - 1$ is an appropriate composite number so that efficient FFT and IFFT algorithms can be used in the realization of the FBLMS algorithm. On the other hand, in applications where it is important to keep the processing delay small, one may need to strike a compromise between system complexity and processing delay. In such applications an alternative implementation of the FBLMS algorithm, which is introduced in the next section, is found to be more efficient.

8.4 The Partitioned FBLMS Algorithm

When the filter length, N, is large and a block length, L, much smaller than N is used, an efficient implementation of the FBLMS algorithm can be derived by dividing (partitioning) the convolution sum of (8.17) into a number of smaller sums and proceeding as discussed below. The resulting implementation is called the partitioned FBLMS (PFBLMS) algorithm.[3]

[3] The PFBLMS algorithm was apparently discovered by a number of independent researchers and has been given different names: Asharif et al. (1986a, b) and Asharif and Amano (1994) call it frequency bin adaptive filtering; Soo and Pang (1987, 1990) refer to it as multidelay FBLMS; and Sommen (1989) uses the name partitioned FBLMS.

Let us assume that $N = P \cdot M$, where P and M are integers, and note that the convolution sum of (8.17) may be written as

$$y(n) = \sum_{l=0}^{P-1} y_l(n), \tag{8.71}$$

where

$$y_l(n) = \sum_{i=0}^{M-1} w_{i+lM} x(n - lM - i). \tag{8.72}$$

To develop a frequency domain implementation of these convolutions, we choose a block length $L = M$ and divide the input data into blocks of length $2M$ samples such that the last M samples of, say, the kth block are same as the first M samples of the $(k+1)$th block. Then, the convolution sum in (8.72) can be evaluated using circular convolution of these data blocks with the appropriate weight vectors having been padded with M zeros. Using $x(kM + M - 1)$ to represent the newest sample in the input, we define the vectors[4]

$$\begin{aligned}\mathbf{x}_{\mathcal{F},l}(k) = \text{FFT}([&x((k-l)M - M) \;\; x((k-l)M - M + 1) \\ &\ldots \;\; x((k-l)M + M - 1)]^{\mathrm{T}}),\end{aligned} \tag{8.73}$$

$$\mathbf{w}_{\mathcal{F},l}(k) = \text{FFT}([w_{lM}(k) \;\; w_{lM+1}(k) \;\; \ldots \;\; w_{lM+M-1}(k) \;\; \overbrace{0 \; 0 \; \ldots \; 0}^{M}]^{\mathrm{T}}), \tag{8.74}$$

$$\mathbf{y}_l(k) = [y_l(kM) \;\; y_l(kM+1) \;\; \ldots \;\; y_l(kM + M - 1)]^{\mathrm{T}}, \tag{8.75}$$

and note that

$$\mathbf{y}_l(k) = \text{the last } M \text{ elements of IFFT } \left(\mathbf{w}_{\mathcal{F},l}(k) \otimes \mathbf{x}_{\mathcal{F},l}(k)\right), \tag{8.76}$$

where $\otimes$ denotes multiplication on an element-by-element basis, k, as before, is the block index, and l is the partition index. We also define

$$\mathbf{y}(k) = [y(kM) \;\; y(kM+1) \;\; \ldots \;\; y(kM + M - 1)]^{\mathrm{T}} \tag{8.77}$$

and note that

$$\mathbf{y}(k) = \sum_{l=0}^{P-1} \mathbf{y}_l(k). \tag{8.78}$$

[4] It may be noted that according to the derivations in the earlier sections, for a filter (here, partition) length of M and a block length of $L = M$, the frequency domain vectors of length $2M - 1$ are sufficient to perform the necessary convolutions in the frequency domain. Here, we are using vectors which are of length $2M$, since this greatly simplifies the implementation of the PFBLMS algorithm. In particular, we note that (8.79) holds only when $L = M$.

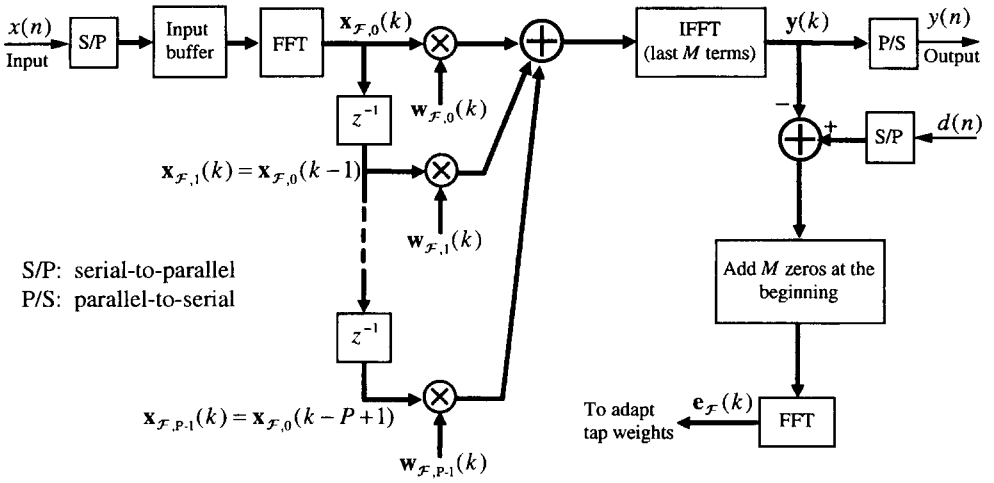


Figure 8.5 The partitioned FBLMS (PFBLMS) algorithm for $L = M$

Furthermore, from (8.73) we note that

$$\mathbf{x}_{\mathcal{F},l}(k) = \mathbf{x}_{\mathcal{F},0}(k-l). \tag{8.79}$$

Substituting (8.76) in (8.78), interchanging the order of summation and IFFT, and using (8.79), we obtain

$$\mathbf{y}(k) = \text{the last } M \text{ elements of IFFT} \left(\sum_{l=0}^{P-1} \mathbf{w}_{\mathcal{F},l}(k) \otimes \mathbf{x}_{\mathcal{F},0}(k-l) \right). \tag{8.80}$$

Using this result, the block diagram of the PFBLMS algorithm may be proposed as depicted in Figure 8.5. Here, the delays, the z^{-1}s, are in the unit of block size and the thick lines represent frequency domain vectors. Also, for future discussion it may be remarked that the implementation of the summation on the right-hand side of (8.80) can also be considered as a parallel bank of $2M$ transversal filters, each of length P, with the jth filter processing the frequency domain samples belonging to the jth frequency bin, for $j = 0, 1, \ldots, 2M-1$.

The adaptation of the filter tap weights is done according to the recursions

$$\begin{aligned} \mathbf{w}_{\mathcal{F},l}(k+1) &= \mathbf{w}_{\mathcal{F},l}(k) + \boldsymbol{\mu}(k) \otimes \mathbf{x}_{\mathcal{F},0}(k-l) \otimes \mathbf{e}_{\mathcal{F}}(k), \\ &\text{for } l = 0, 1, \ldots, P-1, \end{aligned} \tag{8.81}$$

where $\boldsymbol{\mu}(k)$ is the vector of the associated step-size parameters which may be normalized in a similar manner as (8.63),

$$\mathbf{e}_{\mathcal{F}}(k) = \text{FFT} \begin{bmatrix} \mathbf{d}(k) - \mathbf{y}(k) \\ \mathbf{0} \end{bmatrix}, \tag{8.82}$$

$\mathbf{d}(k) = [d(kM)\ \ d(kM+1)\ \ \ldots\ \ d(kM+M-1)]^{\mathrm{T}}$, and $\mathbf{0}$ is the length M zero column vector.

Recursion (8.81) corresponds to the unconstrained PFBLMS algorithm. The constrained PFBLMS algorithm recursion is obtained by constraining the filter tap weights after every iteration of (8.81).

8.4.1 Analysis of the PFBLMS algorithm

In this section we analyse the convergence behaviour of the PFBLMS algorithm. This analysis reveals that the PFBLMS algorithm suffers from slow convergence and hence we suggest some simple solutions to improve its convergence.[5] The main emphasis of this section is the convergence behaviour of the unconstrained PFBLMS algorithm. However, we also make some comments on the behaviour of the constrained PFBLMS algorithm.

From the analysis of the FBLMS algorithm, we recall that the frequency domain samples of input that belong to different frequency bins (i.e. the signal samples at the output of the first FFT in Figure 8.3) are approximately uncorrelated with one another and hence the associated correlation matrix may be approximated by a diagonal matrix

The step-normalization is then used to equalize the time constants of the various modes of convergence of the algorithm.

Extending the above result to the PFBLMS structure, we find that in this case there are $2M$ parallel transversal filters (one belonging to each frequency bin of the signal samples) whose associated input sequences are approximately uncorrelated with one another. Thus, a simple approach to analysing the PFBLMS algorithm is to assume that the transversal filters associated with each bin converge independently of one another so that we can concentrate on the convergence behaviour of these as independent filters.[6] We use the term *frequency bin filter* to refer to these independent filters. From (8.79) and Figure 8.5, we note that the tap-input vector of the ith frequency bin filter is

$$\mathbf{x}_{\mathcal{F}}^{b_i}(k) = [x_{\mathcal{F},0,i}(k)\ \ x_{\mathcal{F},0,i}(k-1)\ \ \ldots\ \ x_{\mathcal{F},0,i}(k-P+1)]^{\mathrm{T}}, \tag{8.83}$$

where $x_{\mathcal{F},0,i}(k)$ is the ith element of $\mathbf{x}_{\mathcal{F},0}(k)$. The convergence behaviour of the ith

However, we note that when the input process, $x(n)$, is stationary, the diagonal elements of $\mathcal{R}_{xx}^{b_i}$ are all identical and thus diag$[\mathcal{R}_{xx}^{b_i}]$ is proportional to the identity matrix. Hence, $\mathcal{R}_{xx}^{b_i}$ and $\mathcal{R}_{xx}^{b_i,\mathcal{N}}$ have the same eigenvalue spread.

Now, observe the fact that the matrix $\mathcal{R}_{xx}^{b_i}$ is a subdiagonal part of the dual of the matrix $\mathcal{R}_{xx}^{\mathrm{u}}$ which was obtained in Section 8.3.2 while analysing the unconstrained FBLMS algorithm (see equation (8.58)). As a result, the following analysis is applicable only to the unconstrained PFBLMS algorithm since it is based on a study of the matrix $\mathcal{R}_{xx}^{b_i,\mathcal{N}}$. The constrained PFBLMS algorithm requires further attention and we will make some comments on its convergence behaviour at the end of this subsection. A modified version of the constrained PFBLMS algorithm, with significantly less computational complexity, will be introduced in Section 8.4.5.

To keep the analysis simple, we consider the case where the input sequence, $x(n)$, is white. Even though this assumption simplifies the analysis greatly, the results obtained are still able to bring out the salient features of the algorithm. For example, the computer simulations given in the next section show that the conclusions drawn in this section remain valid even when $x(n)$ is highly coloured.

The ith element of $\mathbf{x}_{\mathcal{F},0}(k)$ (i.e. the ith frequency bin sample of the filter input) is

$$x_{\mathcal{F},0,i}(k) = \sum_{m=0}^{2M-1} x(kM - M + m)\, \mathrm{e}^{-j2\pi im/2M}. \tag{8.86}$$

When $x(n)$ is white, it is straightforward to show that

$$\mathrm{E}[x_{\mathcal{F},0,i}(k-l)x_{\mathcal{F},0,i}^{*}(k-m)] = \begin{cases} 2M\sigma_x^2, & \text{for } l = m, \\ (-1)^i \times M\sigma_x^2, & \text{for } l = m \pm 1, \\ 0, & \text{otherwise,} \end{cases} \tag{8.87}$$

where σ_x^2 is the variance of $x(n)$. Using this result, we obtain

$$\mathcal{R}_{xx}^{b_i,\mathcal{N}} = \begin{bmatrix} 1 & \alpha_i & 0 & 0 & \dots & 0 & 0 \\ \alpha_i & 1 & \alpha_i & 0 & \dots & 0 & 0 \\ 0 & \alpha_i & 1 & \alpha_i & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \alpha_i & 1 \end{bmatrix}, \tag{8.88}$$

where $\alpha_i = (-1)^i \times 0.5$.

The eigenvalues of $\mathcal{R}_{xx}^{b_i,\mathcal{N}}$ (which are independent of i) can be obtained numerically. These are presented in Table 8.2 for values of P in the range 2 to 10. It is noted that these are widely spread and their dispersion increases significantly as P grows. This means that for large values of P the PFBLMS algorithm may suffer from slow convergence and/or numerical instability, since in (8.88) $\mathcal{R}_{xx}^{b_i,\mathcal{N}}$ becomes badly ill-conditioned for large P.

Observe from the PFBLMS structure shown in Figure 8.5 that the successive partitions of the input samples are 50% overlapped. The value of $|\alpha_i| = 0.5$ in (8.88), which in turn results in the large eigenvalue spread in $\mathcal{R}_{xx}^{b_i,\mathcal{N}}$, is a direct consequence of this 50% overlapping. Numerical studies show that this eigenvalue spread reduces as $|\alpha_i|$

Table 8.2 Eigenvalues of $\mathcal{R}_{xx}^{b_i,\mathcal{N}}$ for different number of partitions, P

P	2	3	4	5	6	7	8	9	10
λ_0	1.500	1.707	1.809	1.866	1.901	1.924	1.940	1.951	1.959
λ_1	0.500	1.000	1.309	1.500	1.623	1.707	1.766	1.809	1.841
λ_2		0.293	0.691	1.000	1.222	1.383	1.500	1.588	1.655
λ_3			0.191	0.500	0.778	1.000	1.174	1.309	1.415
λ_4				0.134	0.376	0.617	0.826	1.000	1.142
λ_5					0.099	0.293	0.500	0.691	0.858
λ_6						0.076	0.234	0.412	0.585
λ_7							0.060	0.191	0.345
λ_8								0.049	0.159
λ_9									0.041
$\lambda_{\max}/\lambda_{\min}$	3	5.828	9.472	13.93	19.20	25.27	32.16	39.86	48.37

decreases. Furthermore, $|\alpha_i|$ can be reduced by reducing the amount of overlap of the successive partitions of the input samples. This is easily achieved by choosing a block length, L, smaller than the partition length, M, as explained in the next section.

Before proceeding with this modification of the PFBLMS algorithm, we shall make some comments on the convergence behaviour of the constrained PFBLMS algorithm. As was noted earlier, the correlation matrix $\mathcal{R}_{xx}^{b_i,\mathcal{N}}$ was the outcome of an analysis of the unconstrained PFBLMS algorithm. A detailed examination of the constrained PFBLMS algorithm is rather involved and beyond the scope of this book. As we will demonstrate through computer simulations later, the effect of overlapping of successive blocks is resolved when the tap weights of the filter are constrained. As a result, we find that the constrained PFBLMS algorithm does not have any convergence problems. It converges almost as fast as its non-partitioned counterpart.

8.4.2 The PFBLMS algorithm with $M > L$

Assuming a block length L and a partition length M, define the vector

$$\tilde{\mathbf{x}}_0(k) = [x(kL - M) \;\; x(kL - M + 1) \;\; \ldots \;\; x(kL + L - 1)]^{\mathrm{T}}. \tag{8.89}$$

Let us choose $M = pL$, where p is an integer. As we show later, this choice of L and M leads to an efficient implementation of the PFBLMS algorithm. We note that if we want to use the DFT to compute the output samples of various partitions in (8.71), then $\tilde{\mathbf{x}}_0(k)$ corresponds to the vector of input samples associated with the first partition, i.e. $y_0(n)$ in (8.71), with $n = kL + L - 1$. Observe that the first element of $\tilde{\mathbf{x}}_0(k)$ is $x(kL - M)$. Similarly, the vectors corresponding to the subsequent partitions start with samples $x(kL - 2M) = x((k - p)L - M)$, $x(kL - 3M) = x((k - 2p)L - M)$, and so on. We thus find that $\tilde{\mathbf{x}}_l(k) = \tilde{\mathbf{x}}_0(k - pl)$ or

$$\mathbf{x}_{\mathcal{F},l}(k) = \mathbf{x}_{\mathcal{F},0}(k - pl), \quad \text{for } l = 1, 2, \ldots, P - 1. \tag{8.90}$$

Table 8.3 Summary of the PFBLMS algorithm

Input: Tap-weight vectors, $\mathbf{w}_{\mathcal{F},l}(k)$, $l = 0, 1, \ldots, P-1$,
Extended input vector,
$\tilde{\mathbf{x}}_0(k) = [x(kL-M) \;\; x(kL-M+1) \;\; \ldots \;\; x(kL+L-1)]^{\mathrm{T}}$,
The past frequency domain vectors of input, $\mathbf{x}_{\mathcal{F},0}(k-l)$, for $k = 1, 2, \ldots, (P-1)p$,
and desired output vector,
$\mathbf{d}(k) = [d(kL) \;\; d(kL+1) \;\; \ldots \;\; d(kL+L-1)]^{\mathrm{T}}$.

Output: Filter output, $\mathbf{y}(k) = [y(kL) \;\; y(kL+1) \;\; \ldots \;\; y(kL+L-1)]^{\mathrm{T}}$,
Tap-weight vector update, $\mathbf{w}_{\mathcal{F},l}(k+1)$, $l = 0, 1, \ldots, P-1$.

1. Filtering:

$$\mathbf{x}_{\mathcal{F},0}(k) = \mathrm{FFT}(\tilde{\mathbf{x}}_0(k))$$
$$\mathbf{y}(k) = \text{the last L elements of } \mathrm{IFFT}\left(\sum_{l=0}^{P-1} \mathbf{w}_{\mathcal{F},l}(k) \otimes \mathbf{x}_{\mathcal{F},0}(k-pl)\right)$$

2. Error estimation:

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{y}(k)$$

3. Step-normalization:

$$\text{for } i = 0 \text{ to } M'-1$$
$$\hat{\sigma}^2_{x_{\mathcal{F},0,i}}(k) = \beta\hat{\sigma}^2_{x_{\mathcal{F},0,i}}(k-1) + (1-\beta)|x_{\mathcal{F},0,i}(k)|^2$$
$$\mu_i(k) = \mu_0/\hat{\sigma}^2_{x_{\mathcal{F},0,i}}(k)$$
$$\boldsymbol{\mu}(k) = [\mu_0(k) \;\; \mu_1(k) \;\; \ldots \;\; \mu_{M'-1}(k)]^{\mathrm{T}}$$

4. Tap-weight adaptation:

$$\mathbf{e}_{\mathcal{F}}(k) = \mathrm{FFT}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{e}(k) \end{bmatrix}\right)$$
$$\text{for } l = 0 \text{ to } P-1$$
$$\mathbf{w}_{\mathcal{F},l}(k+1) = \mathbf{w}_{\mathcal{F},l}(k) + 2\boldsymbol{\mu}(k) \otimes \mathbf{x}^*_{\mathcal{F},0}(k-pl) \otimes \mathbf{e}_{\mathcal{F}}(k)$$

5. Tap-weight constraint:

$$\text{for } l = 0 \text{ to } P-1$$
$$\mathbf{w}_{\mathcal{F},l}(k+1) = \mathrm{FFT}\left(\begin{bmatrix} \text{the first M elements of } \mathrm{IFFT}(\mathbf{w}_{\mathcal{F},l}(k+1)) \\ \mathbf{0} \end{bmatrix}\right)$$

Notes:
- M: partition length; L: block length; $M' = M + L$.
- $\mathbf{0}$ denotes column zero vectors with appropriate length to extend vectors to the length of M'.
- $\otimes$ denotes element-by-element multiplication of vectors.
- Step 5 is applicable only for the constrained PFBLMS algorithm.

where

$$\alpha_i = \frac{1}{p+1} e^{j2\pi pi/(p+1)}.$$

The eigenvalue spread of $\mathcal{R}_{xx}^{b_i,\mathcal{N}}$ (which is independent of i) for values of p changing from 1 to 10 and a fixed value of $P = 10$ are given in Table 8.4. The results clearly show that reducing the overlap of successive partitions significantly improves the convergence behaviour of the unconstrained PFBLMS algorithm.

Table 8.4 Eigenvalue spread, $\lambda_{max}/\lambda_{min}$, of $\mathcal{R}_{xx}^{b_i,\mathcal{N}}$ for $P = 10$ and different values of p

p	1	2	3	4	5	6	7	8	9	10
$\lambda_{max}/\lambda_{min}$	48.37	4.55	2.84	2.25	1.94	1.75	1.63	1.54	1.47	1.42

8.4.3 PFBLMS misadjustment equations

The following results can be derived for the PFBLMS algorithm by following the same line of derivations as in Appendix 8B:

$$\mathcal{M}_{\text{PFBLMS}}^{\text{c}} \approx \mu P M \phi_{xx}(0), \tag{8.95}$$

$$\mathcal{M}_{\text{PFBLMS}}^{\text{u}} \approx \mu P (M + L) \phi_{xx}(0), \tag{8.96}$$

$$\mathcal{M}_{\text{PFBLMS}}^{\text{c},\mathcal{N}} \approx \mu_{\text{o}} P \frac{M}{M + L}, \tag{8.97}$$

$$\mathcal{M}_{\text{PFBLMS}}^{\text{u},\mathcal{N}} \approx \mu_{\text{o}} P. \tag{8.98}$$

As in Section 8.3.5, here also we find that the constrained PFBLMS algorithm achieves a lower level of misadjustment compared with its unconstrained counterpart. The price paid for this is higher computational complexity.

8.4.4 Computational complexity and memory requirement

In this section we give some figures indicating the computational complexity and memory requirement of the PFBLMS algorithm. Instead of specifying the exact number of multiplications and additions, we specify a macro figure, such as the number of butterflies for quantifying computational complexity, since this may be more meaningful in the case of such algorithms. To estimate the memory requirement, we consider only its major blocks and ignore details, such as the temporary memory locations required since these will depend on the DSP system used and also on the efficiency of the code written. Furthermore, we only discuss the computational complexity of the unconstrained PFBLMS algorithm. The constrained PFBLMS algorithm has not been discussed here since its computational complexity will depend, to a great extent, on how the constraining step is implemented. We make some comments on this in the next subsection.

In the implementation of the unconstrained PFBLMS algorithm, the processing of each data block requires two $(p + 1)L$ $(= M + L)$ point FFTs and one IFFT of the same length. Assuming that the data signals are all real-valued, $(p + 1)L$ is chosen a power of 2, and an efficient FFT algorithm such as the Bergland (1968) is used, then $((p + 1)L/4)\log_2((p + 1)L/2)$ butterflies will have to be performed to complete each FFT. The computation of output samples in the frequency domain, i.e. $\mathbf{x}_{\mathcal{F},0}(k - l) \otimes \mathbf{w}_{\mathcal{F},l}(k)$, for $l = 0, 1, \ldots, P - 1$, and implementation of the tap-weight adaptation recursion (8.81) require two and half complex multiplications and two complex additions per data point. Since the step-size parameters are real-valued,

multiplication of a gradient term with its step-size parameter is counted as half a complex multiplication. Furthermore, step-normalization adds some more computations. To give a simple figure, we put all these computations (excluding the FFTs and IFFTs) together and roughly say that the complexity of processing each data point in the frequency domain is equivalent to performing two butterflies. Noting that each partition of input samples, which consists of $(p+1)L$ real-valued samples in the time domain, is converted to $(p+1)L/2$ complex-valued frequency domain samples and there are P such partitions, the total number of frequency domain samples is $(p+1)LP/2$. Adding these together and noting that L output samples are generated at the end of each block processing interval, we obtain the per sample computational complexity of the unconstrained PFBLMS algorithm as

$$C = \frac{(p+1)LP + \frac{3}{4}(p+1)L\log_2\dfrac{(p+1)L}{2}}{L}$$

$$= (p+1)P + \tfrac{3}{4}(p+1)\log_2\frac{(p+1)L}{2}. \tag{8.99}$$

The memory requirements of the unconstrained and constrained PFBLMS algorithms are about the same. The number of frequency domain data samples (including the intermediate results in the z^{-p} delay units) is $(p(P-1)+1)(p+1)L$. We also need $(p+1)LP$ memory words to store the filter coefficients. Some additional storage for input, output, error samples and step-size parameters is also required. Adding these together, the number of memory words required to implement the PFBLMS algorithm is approximately

$$S = (p+1)^2 LP \text{ words}. \tag{8.100}$$

To get a feeling of the above numbers, we give the following example.

Example 8.2

Let us consider an acoustic echo canceller which has to cover an echo spread of at least 250 ms at a sampling rate of 8 kHz. It is recommended that the algorithm latency (delay) in delivering the echo-free samples shall not exceed 16 ms. To cover an echo spread of 250 ms, an adaptive filter with at least 2000 taps should be used since 250 ms is equivalent to 2000 samples at a sampling frequency of 8 kHz. To achieve a latency of less than 16 ms, $L = 64$ is appropriate. Note that there will be a delay of L samples to collect a new block of input samples, and there will be an additional delay of up to one block period (i.e. L sample intervals) to calculate the corresponding block of output samples. This gives a total delay of up to $2L$ sample intervals, which for $L = 64$ and a sampling rate of 8 kHz is equivalent to 16 ms. Table 8.5 summarizes the computational complexity and memory requirement of the unconstrained PFBLMS algorithm for $p = 1$, 3 and 7. These values of p result in $(p+1)L$ being a power of 2 and, therefore, an efficient radix 2 FFT algorithm can be used. From these results we note that $p = 3$ is a good compromise choice since it results in some reduction in computational complexity and, as demonstrated in Section 8.5, significant improvement in convergence behaviour, at the cost of a slight increase in memory.

Table 8.5 Computational complexity and memory requirement of the unconstrained PFBLMS algorithm for the cases discussed in Example 8.2

	Computational complexity	Memory words
$p = 1, P = 32$	73	8192
$p = 3, P = 11$	65	11264
$p = 7, P = 5$	88	20480

8.4.5 Modified constrained PFBLMS algorithm

Our discussion of the PFBLMS algorithm, so far, suggests that the constrained PFBLMS algorithm is significantly more complicated than its unconstrained counterpart. This is because the tap weights of all partitions have to be constrained at the end of every iteration of the algorithm (see Step 5 in Table 8.3). McLaughlin (1996) has proposed a method that significantly reduces the computational complexity of the constrained PFBLMS algorithm while its convergence behaviour is almost unaffected. His method does not constrain the tap weights at the end of all iterations. In the context of a PFBLMS-based acoustic echo canceller, he has a special scheduling method for applying the constraint to various partitions. Although his method has not been supported by any rigorous analysis, computer simulations reveal that this is indeed an effective solution for efficient implementation of the constrained PFBLMS algorithm.

In the context of a general constrained PFBLMS algorithm, the following tap-weight constraint scheduling scheme is suggested here and studied using computer simulations in the next section. After every iteration of the PFBLMS algorithm the tap weights of one or a few of the partitions are constrained on a rotational basis. For example, in the first iteration the tap weights of the first and second partitions are constrained. In the second iteration, the constraint operation is applied to the third and fourth partitions. This process continues until all the partitions are constrained. The constraint operation then restarts with the first and second partitions. Clearly, in cases where the number of partitions, P, is large, this simple approach can significantly reduce the computational complexity of the constrained PFBLMS algorithm.

8.5 Computer Simulations

In this section we present some simulation results that confirm the theoretical results derived in the previous sections. These results also serve to enhance our understanding of the convergence behaviour of the FBLMS and PFBLMS algorithms.

We consider a modelling problem as shown in Figure 8.7. The plant, $W_o(z)$, which is to be identified by the adaptive filter, $W(z)$, is assumed to be a finite impulse response (FIR) system with an impulse response stretching over 1985 samples. This choice of filter length allows us to use a FBLMS algorithm with $L = 64$ and $N' = N + L - 1 = 2^{11}$ for modelling $W_o(z)$ (note that $1985 = 2^{11} - 64 + 1$). $W(z)$ is assumed to have sufficient taps to model $W_o(z)$ perfectly. Two cases of the input, $x(n)$, are considered:

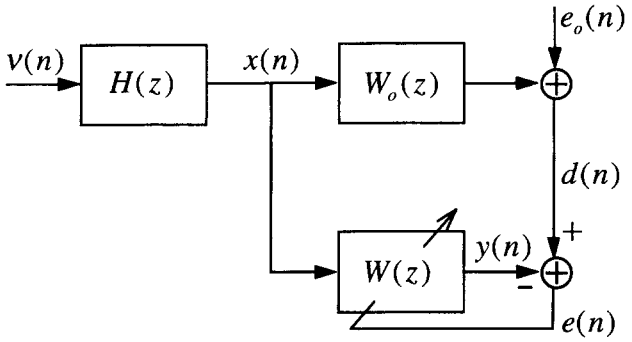


Figure 8.7 Adaptive modelling of an FIR plant

1. A white process.
2. A coloured process which is generated by passing a white noise through a colouring filter with the transfer function

$$H(z) = 0.1 - 0.2z^{-1} - 0.3z^{-2} + 0.4z^{-3} + 0.4z^{-4} - 0.2z^{-5} - 0.1z^{-6}.$$

Recall that this colouring filter is same as the filter $H_2(z)$ used in Chapter 7 (Section 7.6.4). The power spectral density of the process generated by this filter is shown in Figure 7.7. The samples of the plant impulse response, the $w_{o,i}$s, are chosen to be a set of identically, independent, random numbers, and they are normalized so that $\sum_i w_{o,i}^2 = 1$. The sequence $e_o(n)$ is an additive white Gaussian noise. It is independent of $x(n)$ and its variance is set equal to 0.001 for the simulations presented here. So the expected minimum MSE at the adaptive filter output is 0.001. Three cases of $p = 1$, 3 and 7 are considered. To completely cover the the impulse response of the plant, P is chosen to be 32, 11 and 5, respectively, for these cases. For each case, the step-size parameter μ_o is chosen using the misadjustment equations given earlier so as to result in 10% misadjustment. The algorithms used are of the step-normalized type. The learning curves presented here are based on ensemble averages of 100 independent runs for each curve. The averaged curves are smoothed before being plotted.

Figure 8.8 shows the results of the simulations for white input. As expected, the performance of the unconstrained PFBLMS algorithm is quite poor when the overlap is 50% among the successive partitions (i.e. the case $p = 1$) and improves as the overlap is reduced by increasing p. The case when no partitioning is applied, i.e. corresponding to the FBLMS algorithm, is also shown for comparison.

Figure 8.9 repeats Figure 8.8 for the case when $x(n)$ is generated using the colouring filter $H(z)$. In this case the eigenvalue spread of the correlation matrix of $x(n)$ can be as high as 459. Here also we find that reducing the amount of overlap between successive partitions improves the performance of the PFBLMS algorithm. Furthermore, there is very little difference between the results in Figures 8.9 and 8.8. This is in line with the theoretical results of the previous sections which predict that the step-normalized FBLMS and PFBLMS algorithms are insensitive to the power spectral density (eigenvalue spread) of the filter input.

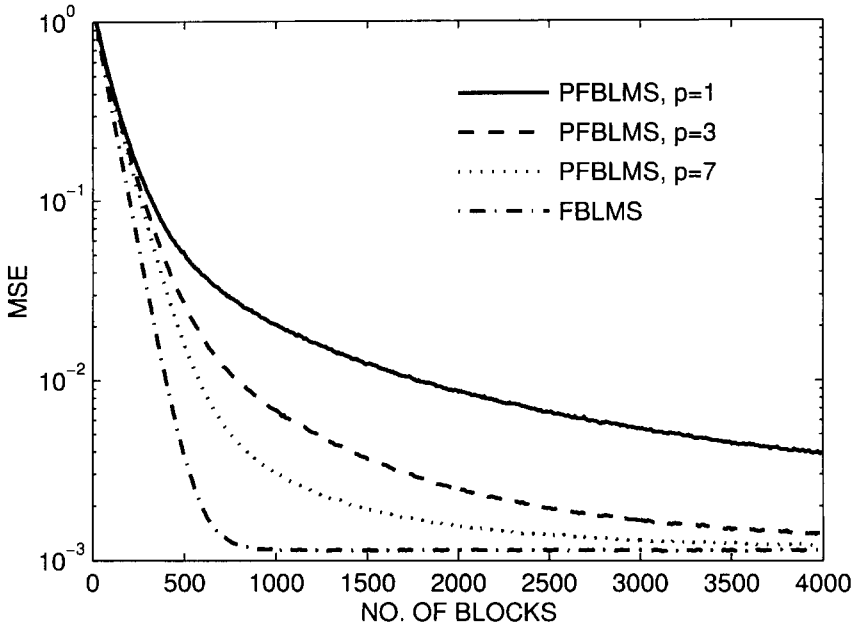


Figure 8.8 Learning curves of the FBLMS and PFBLMS algorithms with white input

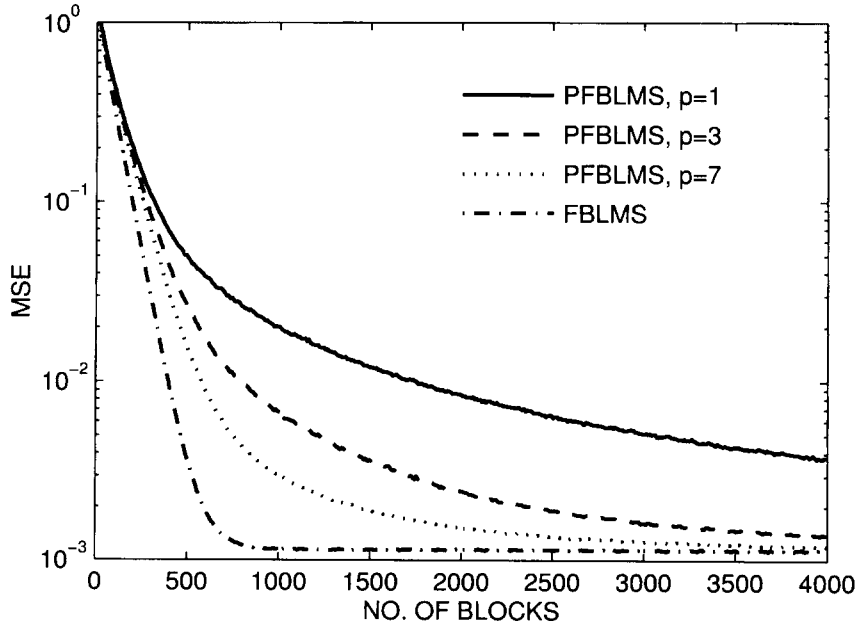


Figure 8.9 Learning curves of the FBLMS and PFBLMS algorithms for a coloured input

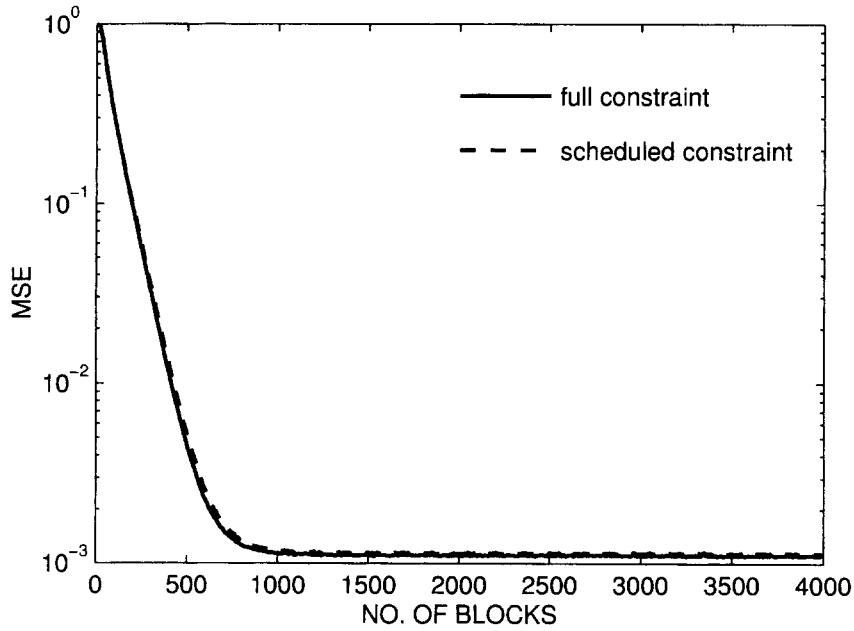


Figure 8.10 Learning curves of the constrained PFBLMS algorithm and its modified version

Figure 8.10 compares the convergence performance of the constrained PFBLMS algorithm and one of its modified versions, with $p = 1$ and $P = 32$. The filter input is coloured and is generated using the colouring filter $H(z)$. In the implementation of the modified PFBLMS algorithm, the tap-weight constraint operation is applied on a rotational basis to only one of the partitions in each iteration. Observe from the results that even though each partition in the modified constrained PFBLMS algorithm is constrained only once in every 32 iterations, the resulting performance loss is negligible. Also, by direct inspection of the learning curves of Figure 8.10, we see that overlap of the partitions has no significant effect on the convergence behaviour of the constrained PFBLMS algorithm. This is in view of the fact that there is only one dominant mode affecting the convergence behaviour of the constrained PFBLMS algorithm, as can be seen from the learning curves.

Problems

P8.1 Consider the BLMS recursion (8.11). In Appendix 8A it is shown that (8.11) can be rearranged as

$$\mathbf{v}(k+1) = \left(\mathbf{I} - 2\frac{\mu_B}{L}\mathbf{X}^T(k)\mathbf{X}(k)\right)\mathbf{v}(k) + 2\frac{\mu_B}{L}\mathbf{X}^T(k)\mathbf{e}_o(k),$$

where $\mathbf{v}(k) = \mathbf{w}(k) - \mathbf{w}_o$, $\mathbf{w}_o$ is the optimum tap-weight vector of the filter and $\mathbf{e}_o(k) = \mathbf{d}(k) - \mathbf{X}(k)\mathbf{w}_o$.

(i) Assuming $\mathbf{v}(k)$ and $\mathbf{X}(k)$ are independent of each other, show that

$$\mathrm{E}[\mathbf{v}(k+1)] = (\mathbf{I} - 2\mu_{\mathrm{B}}\mathbf{R})\mathrm{E}[\mathbf{v}(k)],$$

where $\mathbf{R}$ is the correlation matrix of the filter tap inputs.

(ii) Use the result of part (i) to obtain the time constants that control the convergence behaviour of $\mathrm{E}[\mathbf{v}(k)]$.

(iii) Based on the result obtained in part (ii), justify the validity of (8.12).

P8.2 By direct application of (8.21) and (8.26) confirm the identity (8.28).

P8.3 Define the time-reversed version of the vector $\mathbf{a} = [a_0 \ a_1 \ a_2 \ \dots \ a_{M-1}]^{\mathrm{T}}$ as $\mathbf{a}^{\mathrm{r}} = [a_0 \ a_{M-1} \ a_{M-2} \ \dots \ a_1]^{\mathrm{T}}$.

(i) Show that if $\mathbf{a}_{\mathcal{F}}$ and $\mathbf{a}^{\mathrm{r}}_{\mathcal{F}}$ are the DFTs of $\mathbf{a}$ and $\mathbf{a}^{\mathrm{r}}$, respectively, then

$$\mathbf{a}^{\mathrm{r}}_{\mathcal{F}} = \mathbf{a}^{*}_{\mathcal{F}},$$

where asterisk denotes complex conjugation.

(ii) Show that if $\mathbf{A}_{\mathrm{c}}$ is a circular matrix as in (8.21), then $\mathbf{A}_{\mathrm{c}}^{\mathrm{T}}$ is also a circular matrix. Compare the first columns of $\mathbf{A}_{\mathrm{c}}$ and $\mathbf{A}_{\mathrm{c}}^{\mathrm{T}}$ and show that they are time-reversed versions of each other.

(iii) Use the above observation to give an alternative derivation of (8.32).

P8.4 By direct application of (8.33), (8.34), (8.42), (8.43) and (8.45), show that (8.44) is just an alternative formulation of (8.11).

P8.5 In the derivation of the LMS algorithm, the instantaneous value of $e^2(n)$ was used as an estimate of the cost function $\xi = \mathrm{E}[e^2(n)]$. Give a direct derivation of the BLMS algorithm by considering

$$\hat{\xi}_{\mathrm{B}}(k) = \frac{1}{L}\sum_{i=0}^{L-1} e^2(kL+i)$$

as an estimate of the cost function ξ and running the steepest-descent recursion once after every L samples of the data.

P8.6 The estimate $\hat{\xi}_{\mathrm{B}}(k)$ defined in Problem P8.5 may equivalently be written as

$$\hat{\xi}_{\mathrm{B}}(k) = \frac{1}{L}\tilde{\mathbf{e}}^{\mathrm{T}}(k)\tilde{\mathbf{e}}(k) \qquad \text{(P8.6-1)}$$

where $\tilde{\mathbf{e}}(k)$ the output error vector of the filter in the extended form as defined by (8.43). Using the DFT properties, (P8.6-1) can be expressed in terms of $\mathbf{e}_{\mathcal{F}}(k) = \mathcal{F}\tilde{\mathbf{e}}(k)$ as

$$\hat{\xi}_{\mathrm{B}}(k) = \frac{1}{LN'}\mathbf{e}^{\mathrm{H}}_{\mathcal{F}}(k)\mathbf{e}_{\mathcal{F}}(k), \qquad \text{(P8.6-2)}$$

where $N' = N + L - 1$ is the length of the vector $\tilde{\mathbf{e}}(k)$.

To obtain the optimum frequency domain tap-weight vector $\mathbf{w}_{\mathcal{F}}$, the cost function $\xi(\mathbf{w}_{\mathcal{F}}) = \mathrm{E}[\hat{\xi}_{\mathrm{B}}(k)]$ should be minimized. Accordingly, the optimum solution obtained by the constrained FBLMS algorithm is the one minimizing $\xi(\mathbf{w}_{\mathcal{F}})$, subject to the constraint $\mathcal{P}_{N,0}\mathbf{w}_{\mathcal{F}} = \mathbf{w}_{\mathcal{F}}$. On the other hand, the unconstrained FBLMS minimizes $\xi(\mathbf{w}_{\mathcal{F}})$ without imposing any constraints on $\mathbf{w}_{\mathcal{F}}$.

(i) Show that

$$\xi(\mathbf{w}_{\mathcal{F}}) = \frac{1}{LN'}\left(\mathbf{w}_{\mathcal{F}}^{\mathrm{H}}\mathcal{R}_{xx}^{\mathrm{u}}\mathbf{w}_{\mathcal{F}} - \mathbf{p}_{\mathcal{F}}^{\mathrm{H}}\mathbf{w}_{\mathcal{F}} - \mathbf{w}_{\mathcal{F}}^{\mathrm{H}}\mathbf{p}_{\mathcal{F}} + \mathrm{E}[\mathbf{d}_{\mathcal{F}}^{\mathrm{H}}\mathbf{d}_{\mathcal{F}}]\right),$$

where $\mathcal{R}_{xx}^{\mathrm{u}}$ is as defined in (8.57) and $\mathbf{p}_{\mathcal{F}} = \mathrm{E}[\mathcal{X}_{\mathcal{F}}^{*}\mathcal{P}_{0,L}\mathbf{d}_{\mathcal{F}}]$.

(ii) Find the optimum value of $\mathbf{w}_{\mathcal{F}}$ that minimizes $\xi(\mathbf{w}_{\mathcal{F}})$. Show that the non-singularity of $\mathcal{R}_{xx}^{\mathrm{u}}$ is the necessary and sufficient condition for this solution to be unique.

(iii) It is understood that when a sufficiently small step-size parameter is used for both the constrained and unconstrained FBLMS algorithms so that the misadjustments of the two algorithms can be ignored, the unconstrained FBLMS algorithm converges to a mean-square error which is less than or equal to what can be achieved by the constrained FBLMS algorithm. With the knowledge developed in this problem, how do you explain this?

P8.7 Starting with equation (P8.6-2) of the previous problem, give a direct derivation of the unconstrained recursion (8.49).

P8.8 Consider a modelling problem with the desired output $d(n) = \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{x}(n) + e_{\mathrm{o}}(n)$, where the length of $\mathbf{w}_{\mathrm{o}}$ is less than or equal to the length of the adaptive filter, N. Assume that the plant noise, $e_{\mathrm{o}}(n)$, and its input, $\mathbf{x}(n)$, are uncorrelated with each other. Under these conditions, it is understood that the constrained and unconstrained FBLMS algorithms converge to exactly the same solution. Using the result obtained in Problem P8.6, and assuming that the inverse of $\mathcal{R}_{xx}^{\mathrm{u}}$ exists, give reasons that explain this. What is the common solution to which both the constrained and unconstrained FBLMS algorithms converge?

P8.9 Show that when the block length, L, is equal to the filter length, N, for a given misadjustment, the constrained FBLMS algorithm converges twice as fast as its unconstrained counterpart. Support your answer by giving careful consideration to the time constants associated with the two algorithms. Does your answer continue to hold if step-normalization is (i) used, (ii) not used?

P8.10 Consider the constrained FBLMS recursion (8.47). Show that when the block length, L, is one:

where $\mathbf{x}_{\mathcal{F}}(k)$ is the DFT of the first column of the circular matrix $\mathbf{X}_c(k)$ as defined by (8.33), $e(k)$ is the scalar output error at time k, and $\mathbf{\Gamma}$ is the diagonal matrix consisting of the elements $1, e^{j2\pi/N}, e^{j4\pi/N}, \ldots, e^{j2(N-1)\pi/N}$.

(iii)
$$y(k) = \text{the last term of } \mathcal{F}^{-1}(\mathbf{w}_{\mathcal{F}}(k) \otimes \mathbf{x}_{\mathcal{F}}(k))$$
$$= \left(\frac{1}{N}\mathbf{\Gamma}^{-1}\mathbf{w}_{\mathcal{F}}(k)\right)^{\mathrm{T}} \mathbf{x}_{\mathcal{F}}(k)$$

where $\otimes$ denotes element-by-element multiplication of vectors.

(iv) Now consider the TDLMS algorithm with $\mathcal{T} = \mathcal{F}$. Write down equations corresponding to this case and compare them with the above results. Verify that the FBLMS algorithm with block length $L = 1$ is equivalent to the TDLMS algorithm with $\mathcal{T} = \mathcal{F}$.

P8.11 An alternative procedure for the derivation of (8.61) is proposed in this problem.

(i) Show that

$$\mathbf{P}_{0,L} = \mathcal{F}\mathcal{P}_{0,L}^{*}\mathcal{F}^{-1}$$

and conclude that $\mathcal{P}_{0,L}$ is a circular matrix.

(ii) Show that

$$\text{the first column of } \mathcal{P}_{0,L} = \frac{1}{N'}\mathcal{F}\mathbf{p}_{0,L},$$

where $\mathbf{p}_{0,L}$ is the column vector consisting of the diagonal elements of $\mathbf{P}_{0,L}$.

(iii) Considering the fact that $\mathcal{X}_{\mathcal{F}}(k)$ is a diagonal matrix, show that

$$\mathcal{X}_{\mathcal{F}}^{*}(k)\mathcal{P}_{0,L}\mathcal{X}_{\mathcal{F}} = \mathcal{P}_{0,L} \otimes \left([\mathbf{x}_{\mathcal{F}}(k)\mathbf{x}_{\mathcal{F}}^{\mathrm{H}}(k)]^{*}\right),$$

where $\mathbf{x}_{\mathcal{F}}(k)$ is the column vector consisting of the diagonal elements of $\mathcal{X}_{\mathcal{F}}(k)$, and $\otimes$ denotes element-by-element multiplication of the matrices. Thus, show that

$$\mathcal{R}_{xx}^{\mathrm{u}} = \mathcal{P}_{0,L} \otimes \mathcal{R}_{xx}^{*},$$

where $\mathcal{R}_{xx} = \mathrm{E}[\mathbf{x}_{\mathcal{F}}(k)\mathbf{x}_{\mathcal{F}}^{\mathrm{H}}(k)]$.

(iv) Assuming that the cross-correlation between different elements of the vector $\mathbf{x}_{\mathcal{F}}(k)$ are negligible, show that

$$\mathcal{R}_{xx} \approx N' \times \mathrm{diag}\left(\Phi_{xx}(e^{j(2\pi\times 0)/N'}),\ \Phi_{xx}(e^{j2\pi/N'}),\ \ldots,\ \Phi_{xx}(e^{j2\pi(N'-1)/N'})\right).$$

(v) Using the results of Parts (iii) and (iv), derive (8.61).

(vi) Do a thorough study of the elements of the matrix $\mathcal{P}_{0,L}$. In particular, verify that the largest (in magnitude) elements of $\mathcal{P}_{0,L}$ are its diagonal elements. Use your findings to conclude that $\mathcal{R}_{xx}^{\mathrm{u}}$ is closer to diagonal than $\mathcal{R}_{xx}$, in the sense that the non-diagonal elements of the normalized matrix $(\mathrm{diag}[\mathcal{R}_{xx}^{\mathrm{u}}])^{-1}\mathcal{R}_{xx}^{\mathrm{u}}$ are smaller than the corresponding elements of $(\mathrm{diag}[\mathcal{R}_{xx}])^{-1}\mathcal{R}_{xx}$.

P8.12 Verify the results presented in (8.87).

P8.13 Verify the results presented in (8.93).

P8.14 For the case discussed in Example 8.2, evaluate the computational complexity and memory requirement of the FBLMS algorithm, for both the constrained and unconstrained cases, and compare your results with those given in Table 8.5.

P8.15 Discuss in detail why the selection of $M = pL$ results in less overlap among successive partitions, as p increases.

P8.16 In the results presented in Table 8.5, we find that the unconstrained PFBLMS algorithm with $p = 3$ is less complex than the case where $p = 1$. Explore the contribution of various parts of the algorithm to find out why this is happening.

P8.17 Evaluate the computational complexities of the constrained PFBLMS implementation and its modified version that were used to obtain the simulation results of Figure 8.10 and compare your results with those in Table 8.5.

Simulation-Oriented Problems

P8.18 The MATLAB program 'blk_mdlg.m', which was used to obtain the results of Example 8.1, is available on an accompanying diskette. Run this program and confirm the results of Figure 8.2. In addition, using this program, study the convergence behaviour of the BLMS algorithm for the following choices of L and $\mathcal{M}_{BMLS}$, and discuss your findings:

L	$\mathcal{M}_{BLMS}$
$4N, 5N$	10%
$N, 2N, 3N, 4N, 5N$	5%
$N, 2N, 3N, 4N, 5N$	20%

P8.19 Develop a simulation program to confirm the results that are presented in Figure 8.4.

P8.20 Consider a channel equalization problem similar to the one discussed in Section 6.4.2. Assume that the channel response is characterized by the transfer function

$$H(z) = 0.1 + 0.3z^{-1} + 0.6z^{-2} + z^{-3} + 0.5z^{-4} - 0.2z^{-5} + 0.1z^{-6},$$

the input data, $s(n)$, to the channel is binary and white, the channel noise, $\nu(n)$, is white and Gaussian, the signal-to-noise ratio at the channel output is 30 dB, and the equalizer length, N, and the delay, Δ, are set equal to 33 and 18, respectively. Develop a simulation program to study the performance of FBLMS algorithm in this application.

P8.21 Consider the channel equalization set-up of the previous problem. By running appropriate simulation programs, study the convergence behaviours of the conventional LMS algorithm and the TDLMS algorithm (with various transforms) and compare your results with that of the FBLMS algorithm.

Appendix 8A Derivation of a Misadjustment Equation for the BLMS Algorithm

In this appendix we present a simple derivation of the misadjustment of the BLMS algorithm. This derivation is different from the one used for the conventional LMS algorithm in Chapter 6. Because of certain assumptions used here (such as the step-size parameter, μ_B, is small, and an adaptive filter models the plant almost exactly), this derivation is rather less accurate.

We start with the recursion (8.11) and use the definition $\mathbf{v}(k) = \mathbf{w}(k) - \mathbf{w}_o$, where $\mathbf{w}_o$ is the optimum tap-weight vector of the filter, to obtain

$$\mathbf{v}(k+1) = \mathbf{v}(k) + 2\frac{\mu_B}{L}\mathbf{X}^T(k)\mathbf{e}(k). \tag{8A-1}$$

We also note that

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{X}(k)\mathbf{w}(k) = \mathbf{e}_o(k) - \mathbf{X}(k)\mathbf{v}(k), \tag{8A-2}$$

where $\mathbf{e}_o(k) = \mathbf{d}(k) - \mathbf{X}(k)\mathbf{w}_o$ is the output error when the optimum tap-weight vector, $\mathbf{w}_o$, is used.

Substituting (8A-2) in (8A-1) we get

$$\mathbf{v}(k+1) = \left(\mathbf{I} - 2\frac{\mu_B}{L}\mathbf{X}^T(k)\mathbf{X}(k)\right)\mathbf{v}(k) + 2\frac{\mu_B}{L}\mathbf{X}^T(k)\mathbf{e}_o(k). \tag{8A-3}$$

Next, we multiply both sides of (8A-3) from the left by their respective transposes and expand to obtain

$$\begin{aligned}\mathbf{v}^T(k+1)\mathbf{v}(k+1) = {} & \mathbf{v}^T(k)\left(\mathbf{I} - 2\frac{\mu_B}{L}\mathbf{X}^T(k)\mathbf{X}(k)\right)^2\mathbf{v}(k)) \\ & + 2\frac{\mu_B}{L}\mathbf{e}_o^T(k)\mathbf{X}(k)\left(\mathbf{I} - 2\frac{\mu_B}{L}\mathbf{X}^T(k)\mathbf{X}(k)\right)\mathbf{v}(k) \\ & + 2\frac{\mu_B}{L}\mathbf{v}^T(k)\left(\mathbf{I} - 2\frac{\mu_B}{L}\mathbf{X}^T(k)\mathbf{X}(k)\right)\mathbf{X}^T(k)\mathbf{e}_o(k) \\ & + 4\frac{\mu_B^2}{L^2}\mathbf{e}_o^T(k)\mathbf{X}(k)\mathbf{X}^T(k)\mathbf{e}_o(k). \end{aligned} \tag{8A-4}$$

[illegible]

and $\mathbf{e}_o(k)$. This results in

$$\|\mathbf{v}(k+1)\|^2 = E\left[\mathbf{v}^T(k)\left(\mathbf{I} - 2\frac{\mu_B}{L}\mathbf{X}^T(k)\mathbf{X}(k)\right)^2\mathbf{v}(k)\right] + 4\frac{\mu_B^2}{L^2}E[\mathbf{e}_o^T(k)\mathbf{X}(k)\mathbf{X}^T(k)\mathbf{e}_o(k)], \tag{8A-5}$$

where $\|\mathbf{v}(k)\|^2 = E[\mathbf{v}^T(k)\mathbf{v}(k)]$. The first term on the right-hand side of (8A-5) can be expanded as

$$\begin{aligned} &E\left[\mathbf{v}^T(k)\left(\mathbf{I} - 2\frac{\mu_B}{L}\mathbf{X}^T(k)\mathbf{X}(k)\right)^2\mathbf{v}(k)\right] \\ &\quad = \|\mathbf{v}(k)\|^2 - 4\frac{\mu_B}{L}E[\mathbf{v}^T(k)\mathbf{X}^T(k)\mathbf{X}(k)\mathbf{v}(k)] \\ &\qquad + 4\frac{\mu_B^2}{L^2}E[\mathbf{v}^T(k)\mathbf{X}^T(k)\mathbf{X}(k)\mathbf{X}^T(k)\mathbf{X}(k)\mathbf{v}(k)]. \end{aligned} \tag{8A-6}$$

To simplify this, we assume that μ_B is small so that the last term on the right-hand side of (8A-6) can be ignored. Furthermore, using the *independence assumption* between $\mathbf{v}(k)$ and $\mathbf{X}(k)$ and following the same line of argument as in Chapter 6, we obtain

$$\begin{aligned} &E\left[\mathbf{v}^T(k)\left(\mathbf{I} - 2\frac{\mu_B}{L}\mathbf{X}^T(k)\mathbf{X}(k)\right)^2\mathbf{v}(k)\right] \\ &\quad \approx \|\mathbf{v}(k)\|^2 - 4\frac{\mu_B}{L}E[\mathbf{v}^T(k)E[\mathbf{X}^T(k)\mathbf{X}(k)]\mathbf{v}(k)]. \end{aligned} \tag{8A-7}$$

Now note from (8.4) that

$$E[\mathbf{X}^T(k)\mathbf{X}(k)] = L\mathbf{R}, \tag{8A-8}$$

where $\mathbf{R}$ is the $N \times N$ correlation matrix of the filter tap inputs. Substituting (8A-8) in (8A-7) we get

$$E\left[\mathbf{v}^T(k)\left(\mathbf{I} - 2\frac{\mu_B}{L}\mathbf{X}^T(k)\mathbf{X}(k)\right)^2\mathbf{v}(k)\right] \approx \|\mathbf{v}(k)\|^2 - 4\mu_B E[\mathbf{v}^T(k)\mathbf{R}\mathbf{v}(k)]. \tag{8A-9}$$

To evaluate the second term on the right-hand side of (8A-5), we note that $\mathbf{e}_o^T(k)\mathbf{X}(k)\mathbf{X}^T(k)\mathbf{e}_o(k)$ is a scalar and use (6.23) to write

$$\begin{aligned} \mathbf{e}_o^T(k)\mathbf{X}(k)\mathbf{X}^T(k)\mathbf{e}_o(k) &= \text{tr}[\mathbf{e}_o^T(k)\mathbf{X}(k)\mathbf{X}^T(k)\mathbf{e}_o(k)] \\ &= \text{tr}[\mathbf{e}_o(k)\mathbf{e}_o^T(k)\mathbf{X}(k)\mathbf{X}^T(k)]. \end{aligned} \tag{8A-10}$$

Taking expectation on both sides of (8A-10) and noting that $\mathbf{e}_o(k)$ and $\mathbf{X}(k)$ are independent of each other, we obtain

$$E[\mathbf{e}_o^T(k)\mathbf{X}(k)\mathbf{X}^T(k)\mathbf{e}_o(k)] = \text{tr}[E[\mathbf{e}_o(k)\mathbf{e}_o^T(k)]E[\mathbf{X}(k)\mathbf{X}^T(k)]]. \tag{8A-11}$$

Next, we assume that the elements of $\mathbf{e}_o(k)$ are samples of a white noise process. This assumption is justified when the adaptive filter is long enough to model the plant almost exactly. This implies that

$$\mathrm{E}[\mathbf{e}_o(k)\mathbf{e}_o^{\mathrm{T}}(k)] = \xi_{\min}\mathbf{I}, \tag{8A-12}$$

where $\xi_{\min} = \mathrm{E}[e_o^2(n)]$ is the minimum MSE at the filter output, and the identity matrix $\mathbf{I}$ is $L \times L$. Substituting (8A-12) in (8A-11), noting that $\mathrm{tr}[\mathbf{X}(k)\mathbf{X}^{\mathrm{T}}(k)] = \mathrm{tr}[\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)]$, and using (8A-8), we get

$$\mathrm{E}[\mathbf{e}_o^{\mathrm{T}}(k)\mathbf{X}(k)\mathbf{X}^{\mathrm{T}}(k)\mathbf{e}_o(k)] = L\xi_{\min}\mathrm{tr}[\mathbf{R}]. \tag{8A-13}$$

Substituting (8A-13) and (8A-9) in (8A-5) we obtain

$$\|\mathbf{v}(k+1)\|^2 \approx \|\mathbf{v}(k)\|^2 - 4\mu_{\mathrm{B}}\mathrm{E}[\mathbf{v}^{\mathrm{T}}(k)\mathbf{R}\mathbf{v}(k)] + 4\frac{\mu_{\mathrm{B}}^2}{L}\xi_{\min}\mathrm{tr}[\mathbf{R}]. \tag{8A-14}$$

When the algorithm has converged and reached its steady state, $\|\mathbf{v}(k+1)\|^2 = \|\mathbf{v}(k)\|^2$. Using this in (8A-14), we obtain, in the steady state,

$$\mathrm{E}[\mathbf{v}^{\mathrm{T}}(k)\mathbf{R}\mathbf{v}(k)] \approx \frac{\mu_{\mathrm{B}}}{L}\xi_{\min}\mathrm{tr}[\mathbf{R}]. \tag{8A-15}$$

We recall that the left-hand side of (8A-15) is equal to the excess MSE of the algorithm after its convergence (see (6.21) and the subsequent discussion in the same section). Thus, we obtain

$$\text{excess MSE of the BLMS algorithm} \approx \frac{\mu_{\mathrm{B}}}{L}\xi_{\min}\mathrm{tr}[\mathbf{R}]. \tag{8A-16}$$

Dividing this excess MSE by the minimum MSE, $\xi_{\min}$, we obtain (8.13).

Appendix 8B Derivation of Misadjustment Equations for the FBLMS Algorithm

Let us start with the definition of misadjustment. We recall from Chapter 6 that for an adaptive algorithm, misadjustment is defined by the equation

$$\mathcal{M} = \frac{\xi_{\text{excess}}}{\xi_{\min}}, \tag{8B-1}$$

where ξ_{excess} is the excess MSE due to perturbation of the filter tap weights after the algorithm has reached its steady state, and $\xi_{\min}$ is the minimum MSE that would be achieved by the optimum tap weights. The excess MSE, as defined before (in Chapter 6), is given by the following equation and is evaluated after the convergence of the filter:

$$\xi_{\text{excess}} = \mathrm{E}[(\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n))^2]. \tag{8B-2}$$

Here, $\mathbf{v}(n) = \mathbf{w}(n) - \mathbf{w}_o$ is the tap-weight perturbation vector and thus $\mathbf{v}^T(n)\mathbf{x}(n)$ is an associated error quantity.

In the case of the FBLMS algorithm, where the perturbation vector $\mathbf{v}(k)$ varies only once every block, the excess MSE is defined as

$$\xi_{\text{excess}} = \frac{1}{L} E[(\mathbf{X}(k)\mathbf{v}(k))^T(\mathbf{X}(k)\mathbf{v}(k))] \tag{8B-3}$$

where $\mathbf{X}(k)\mathbf{v}(k)$ is the length L vector of error samples arising from the tap-weight perturbation $\mathbf{v}(k)$ during the kth block.

If $\mathbf{w}_{\mathcal{F}}(k)$ in (8.41) is replaced by $\mathbf{v}_{\mathcal{F}}(k)$, where $\mathbf{v}_{\mathcal{F}}(k)$ is defined as in (8.55), then the result would be the error due to the tap-weight error $\mathbf{v}_{\mathcal{F}}(k)$. Using this result, we obtain

$$\xi_{\text{excess}} = \frac{1}{L} E[(\mathbf{P}_{0,L}\mathcal{F}^{-1}\mathcal{X}_{\mathcal{F}}(k)\mathbf{v}_{\mathcal{F}}(k))^H(\mathbf{P}_{0,L}\mathcal{F}^{-1}\mathcal{X}_{\mathcal{F}}(k)\mathbf{v}_{\mathcal{F}}(k))]. \tag{8B-4}$$

Note that the transpose operator T is replaced by the Hermitian transpose operator H in (8B-4) since the frequency domain variables are, in general, complex-valued. It should also be noted that the $\mathbf{v}_{\mathcal{F}}(k)$ in (8B-4) need not to be constrained, i.e. the last $L-1$ samples of $\mathcal{F}^{-1}\mathbf{v}_{\mathcal{F}}(k)$ need not be zero. Accordingly, (8B-4) can be used for evaluating the excess MSE for both the constrained and unconstrained FBLMS algorithms.

Rearranging the terms under the expectation in (8B-4), and noting that $\mathbf{P}_{0,L}^H = \mathbf{P}_{0,L}$, $\mathbf{P}_{0,L}^2 = \mathbf{P}_{0,L}$, and $(\mathcal{F}^{-1})^H = (1/N')\mathcal{F}$, we obtain

$$\begin{aligned}\xi_{\text{excess}} &= \frac{1}{LN'} E[\mathbf{v}_{\mathcal{F}}^H(k)\mathcal{X}_{\mathcal{F}}^*(k)\mathcal{F}\mathbf{P}_{0,L}\mathcal{F}^{-1}\mathcal{X}_{\mathcal{F}}(k)\mathbf{v}_{\mathcal{F}}(k)] \\ &= \frac{1}{LN'} E[\mathbf{v}_{\mathcal{F}}^H(k)\mathcal{X}_{\mathcal{F}}^*(k)\mathcal{P}_{0,L}\mathcal{X}_{\mathcal{F}}(k)\mathbf{v}_{\mathcal{F}}(k)].\end{aligned} \tag{8B-5}$$

We recall that the length of $\mathbf{v}_{\mathcal{F}}(k)$ is $N' = N + L - 1$ and $\mathcal{X}_{\mathcal{F}}(k)$ is an $N' \times N'$ diagonal matrix. Assuming that $\mathbf{v}_{\mathcal{F}}(k)$ and $\mathcal{X}_{\mathcal{F}}(k)$ are independent of each other, we obtain from (8B-5)

$$\xi_{\text{excess}} = \frac{1}{LN'} E[\mathbf{v}_{\mathcal{F}}^H(k)\mathcal{R}_{xx}^u\mathbf{v}_{\mathcal{F}}(k)], \tag{8B-6}$$

where $\mathcal{R}_{xx}^u = E[\mathcal{X}_{\mathcal{F}}^*(k)\mathcal{P}_{0,L}\mathcal{X}_{\mathcal{F}}(k)]$, as defined in (8.57).

Proceeding in the same line of derivations as in Chapter 6 (Section 6.3), we obtain from (8B-6)

$$\xi_{\text{excess}} = \frac{1}{LN'} \text{tr}[\mathbf{K}_{\mathcal{F}}(k)\mathcal{R}_{xx}^u] \tag{8B-7}$$

where $\mathbf{K}_{\mathcal{F}}(k) = E[\mathbf{v}_{\mathcal{F}}(k)\mathbf{v}_{\mathcal{F}}^H(k)]$.

With the expressions (8B-6) and (8B-7) for ξ_{excess}, we are now ready to proceed with the derivation of the excess MSE for the various implementations of the FBLMS algorithm.

Unconstrained FBLMS algorithm without step-normalization. We multiply both sides of (8.56) from the right by their respective Hermitian transposes, expand, take expectation on both sides, and use similar assumptions as those used in deriving (8A-6), to obtain

$$\|\mathbf{v}_{\mathcal{F}}(k+1)\|^2 \approx \|\mathbf{v}_{\mathcal{F}}(k)\|^2 - 4\mu \mathrm{E}[\mathbf{v}_{\mathcal{F}}^{\mathrm{H}}(k)\boldsymbol{\mathcal{R}}_{xx}^{\mathrm{u}}\mathbf{v}_{\mathcal{F}}(k)] + 4\mu^2 \mathrm{E}[(\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}(\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}(k)})]. \tag{8B-8}$$

In the steady state, $\|\mathbf{v}_{\mathcal{F}}(k+1)\|^2 = \|\mathbf{v}_{\mathcal{F}}(k)\|^2$. Thus, when the algorithm has reached its steady state, we obtain from (8B-8)

$$\begin{aligned}\mathrm{E}[\mathbf{v}_{\mathcal{F}}^{\mathrm{H}}(k)\boldsymbol{\mathcal{R}}_{xx}^{\mathrm{u}}\mathbf{v}_{\mathcal{F}}(k)] &\approx \mu \mathrm{E}[(\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}(\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))] \\ &\approx \mu \mathrm{E}[(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))]. \end{aligned} \tag{8B-9}$$

We note that the last expectation in (8B-9) is a scalar and thus, using (6.23), it may be rearranged as

$$\begin{aligned}&\mathrm{E}[(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))] \\ &\quad = \mathrm{E}[\mathrm{tr}[(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))]] \\ &\quad = \mathrm{tr}[\mathrm{E}[(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)]] \\ &\quad = \mathrm{tr}[\mathrm{E}[(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}]\mathrm{E}[\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)]], \end{aligned} \tag{8B-10}$$

where the last equality follows from the independence assumption. Furthermore, we note that

$$\begin{aligned}\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k) &= \boldsymbol{\mathcal{F}}\mathbf{P}_{0,L}\boldsymbol{\mathcal{F}}^{-1}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k) \\ &= \boldsymbol{\mathcal{F}}\tilde{\mathbf{e}}_{\mathrm{o}}(k), \end{aligned} \tag{8B-11}$$

where $\tilde{\mathbf{e}}_{\mathrm{o}}(k) = [0\ 0\ \ldots\ 0\ e_{\mathrm{o}}(kL)\ e_{\mathrm{o}}(kL+1)\ \ldots\ e_{\mathrm{o}}(kL+L-1)]^{\mathrm{T}}$ is the optimum output error vector in the extended form. Using (8B-11), we obtain

$$\mathrm{E}[(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}})(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}})^{\mathrm{H}}] = N'\boldsymbol{\mathcal{F}}\mathrm{E}[\tilde{\mathbf{e}}_{\mathrm{o}}(k)\tilde{\mathbf{e}}_{\mathrm{o}}^{\mathrm{T}}(k)]\boldsymbol{\mathcal{F}}^{-1}, \tag{8B-12}$$

where we have noted that $\boldsymbol{\mathcal{F}}^{\mathrm{H}} = N'\boldsymbol{\mathcal{F}}^{-1}$ and $\tilde{\mathbf{e}}_{\mathrm{o}}^{\mathrm{H}}(k)$ is replaced by $\tilde{\mathbf{e}}_{\mathrm{o}}^{\mathrm{T}}(k)$, since $\tilde{\mathbf{e}}_{\mathrm{o}}(k)$ is assumed to be a real-valued vector. Assuming that the optimum error terms $e_{\mathrm{o}}(kL)$, $e_{\mathrm{o}}(kL+1), \ldots, e_{\mathrm{o}}(kL+L-1)$ are samples of a white noise process with variance $\mathrm{E}[e_{\mathrm{o}}^2(n)]$ and noting that $\mathrm{E}[e_{\mathrm{o}}^2(n)] = \xi_{\min}$, we get

$$\mathrm{E}[\tilde{\mathbf{e}}_{\mathrm{o}}(k)\tilde{\mathbf{e}}_{\mathrm{o}}^{\mathrm{T}}(k)] = \xi_{\min}\mathbf{P}_{0,L}, \tag{8B-13}$$

where $\mathbf{P}_{0,L}$ is defined as in (8.36).

Substituting (8B-13) in (8B-12) we get

$$\mathrm{E}[(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}] = N'\xi_{\min}\boldsymbol{\mathcal{P}}_{0,L}. \tag{8B-14}$$

Using this result and the identity (6.23), we obtain

$$\begin{aligned}
&\mathrm{tr}[\mathrm{E}[(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}]\mathrm{E}[\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)]] \\
&\quad = N'\xi_{\min}\mathrm{tr}[\mathrm{E}[\boldsymbol{\mathcal{P}}_{0,L}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)]] \\
&\quad = N'\xi_{\min}\mathrm{tr}[\mathrm{E}[\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)\boldsymbol{\mathcal{P}}_{0,L}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)]] \\
&\quad = N'\xi_{\min}\mathrm{tr}[\boldsymbol{\mathcal{R}}_{xx}^{\mathrm{u}}] \\
&\quad = LN'^{2}\phi_{xx}(0)\xi_{\min},
\end{aligned} \tag{8B-15}$$

where the last equality follows from the identity

$$\begin{aligned}
\mathrm{tr}[\boldsymbol{\mathcal{R}}_{xx}^{\mathrm{u}}] &= \mathrm{tr}[\boldsymbol{\mathcal{F}}\mathbf{R}_{xx}^{\mathrm{u}}\boldsymbol{\mathcal{F}}^{-1}] \\
&= \mathrm{tr}[\boldsymbol{\mathcal{F}}^{-1}\boldsymbol{\mathcal{F}}\mathbf{R}_{xx}^{\mathrm{u}}] \\
&= \mathrm{tr}[\mathbf{R}_{xx}^{\mathrm{u}}] = LN'\phi_{xx}(0)
\end{aligned} \tag{8B-16}$$

which is obtained from (8.57) and (8.60).

Substituting (8B-15) in (8B-10), and taking the result back to (8B-6) through (8B-9), we obtain

$$\xi_{\mathrm{excess}}^{\mathrm{u}} = \mu N'\phi_{xx}(0)\xi_{\min}. \tag{8B-17}$$

Substituting this result in (8B-1), we get the misadjustment for the unconstrained FBLMS algorithm without step-normalization as

$$\mathcal{M}_{\mathrm{FBLMS}}^{\mathrm{u}} = \mu N'\phi_{xx}(0). \tag{8B-18}$$

Unconstrained FBLMS algorithm with step-normalization. Following the same line of derivations as in Section 8.3.2, for the present case we obtain

$$\begin{aligned}
\mathbf{v}_{\mathcal{F}}(k+1) = (\mathbf{I} - 2\mu_{\mathrm{o}}\boldsymbol{\Lambda}^{-1}\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)\boldsymbol{\mathcal{P}}_{0,L}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k))\mathbf{v}_{\mathcal{F}}(k) \\
+ 2\mu_{\mathrm{o}}\boldsymbol{\Lambda}^{-1}\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k),
\end{aligned} \tag{8B-19}$$

where $\boldsymbol{\Lambda} = \mathrm{E}[\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)]$. If we post-multiply both sides of (8B-19) by their respective Hermitian transposes, take expectations, assume that $\mathbf{e}_{\mathrm{o},\mathcal{F}}(k)$ is zero mean and independent of $\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)$, and do some manipulations and approximation similar to what was done above, we obtain

$$\begin{aligned}
\mathbf{K}_{\mathcal{F}}(k+1) \approx \mathbf{K}_{\mathcal{F}}(k) - 2\mu_{\mathrm{o}}\boldsymbol{\Lambda}^{-1}\boldsymbol{\mathcal{R}}_{xx}^{\mathrm{u}}\mathbf{K}_{\mathcal{F}}(k) - 2\mu_{\mathrm{o}}\mathbf{K}_{\mathcal{F}}(k)\boldsymbol{\mathcal{R}}_{xx}^{\mathrm{u}}\boldsymbol{\Lambda}^{-1} \\
+ 4\mu_{\mathrm{o}}^{2}\boldsymbol{\Lambda}^{-1}\mathrm{E}[\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)]\boldsymbol{\Lambda}^{-1},
\end{aligned} \tag{8B-20}$$

where $\mathbf{K}_{\mathcal{F}}(k) = \mathrm{E}[\mathbf{v}_{\mathcal{F}}(k)\mathbf{v}_{\mathcal{F}}^{\mathrm{H}}(k)]$. Since $\mathbf{e}_{\mathrm{o},\mathcal{F}}(k)$ and $\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)$ are independent, we get, using (8B-14)

$$\begin{aligned}
&\mathrm{E}[\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)] \\
&\quad = E[\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)\mathrm{E}[(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))(\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}]\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)] \\
&\quad = N'\xi_{\min}\mathrm{E}[\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)\boldsymbol{\mathcal{P}}_{0,L}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)] \\
&\quad = N'\xi_{\min}\boldsymbol{\mathcal{R}}_{xx}^{\mathrm{u}}. \qquad \text{(8B-21)}
\end{aligned}$$

We note that $\boldsymbol{\Lambda}$ is a diagonal matrix consisting of the estimates of the powers of the input signal samples in the frequency domain. Considering the spectral separation property of the DFT (see Oppenheim and Schafer, 1975, for example), we obtain

$$\boldsymbol{\Lambda} \approx N' \times \mathrm{diag}(\Phi_{xx}(\mathrm{e}^{j(2\pi\times 0)/N'}), \Phi_{xx}(\mathrm{e}^{j2\pi/N'}), \ldots, \Phi_{xx}(\mathrm{e}^{j2\pi(N'-1)/N'})), \qquad \text{(8B-22)}$$

where $\Phi_{xx}(\mathrm{e}^{j\omega})$ is the power spectral density of the underlying input process, $x(n)$. The factor N' in (8B-22) is the length of the DFT in the present case. Comparing (8B-22) with (8.61) we find that

$$\boldsymbol{\Lambda} \approx \frac{N'}{L}\boldsymbol{\mathcal{R}}_{xx}^{\mathrm{u}}. \qquad \text{(8B-23)}$$

Substituting (8B-21) and (8B-23) in (8B-20) we get

$$\mathbf{K}_{\mathcal{F}}(k+1) \approx \mathbf{K}_{\mathcal{F}}(k) - 4\mu_{\mathrm{o}}\frac{L}{N'}\mathbf{K}_{\mathcal{F}}(k) + 4\mu_{\mathrm{o}}^{2}\xi_{\min}\frac{L^{2}}{N'}\boldsymbol{\mathcal{R}}_{xx}^{\mathrm{u}\,-1}. \qquad \text{(8B-24)}$$

In the steady state, when $\mathbf{K}_{\mathcal{F}}(k+1) = \mathbf{K}_{\mathcal{F}}(k)$, we obtain

$$\mathbf{K}_{\mathcal{F}}(k) \approx \mu_{\mathrm{o}}\xi_{\min}L\boldsymbol{\mathcal{R}}_{xx}^{\mathrm{u}\,-1}. \qquad \text{(8B-25)}$$

Substituting (8B-25) in (8B-7), we get

$$\xi_{\mathrm{excess}}^{\mathrm{u},\mathcal{N}} \approx \mu_{\mathrm{o}}\xi_{\min}. \qquad \text{(8B-26)}$$

Substituting this result in (8B-1), we obtain the misadjustment for the unconstrained FBLMS algorithm with step-normalization as

$$\mathcal{M}_{\mathrm{FBLMS}}^{\mathrm{u},\mathcal{N}} \approx \mu_{\mathrm{o}}. \qquad \text{(8B-27)}$$

Constrained FBLMS algorithm without step-normalization. In this case the FBLMS algorithm is an exact and fast implementation of the BLMS algorithm, i.e. with a reduced computational complexity. Hence, the corresponding excess MSE is given by (8A-16). Noting that $\mathbf{R}$ is $N \times N$ and its diagonal elements are all equal to $\phi_{xx}(0)$, (8A-16) may also be written as

$$\xi_{\mathrm{excess}}^{\mathrm{c}} = \mu N\xi_{\min}\phi_{xx}(0), \qquad \text{(8B-28)}$$

to be in line with the rest of the results in this appendix. Substituting this result in (8B-1), we obtain the corresponding misadjustment as

$$\mathcal{M}_{\text{FBLMS}}^{\text{c},\mathcal{N}} \approx \mu N \phi_{xx}(0). \tag{8B-29}$$

Constrained FBLMS algorithm with step-normalization. Premultiplication of the gradient vector $\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)\mathbf{e}_{\mathcal{F}}(k)$ by the matrix $\boldsymbol{\mathcal{P}}_{N,0}$ implements the constraining step (see (8.47)). Combining this step with step-normalization, we get the recursion

$$\begin{aligned}\mathbf{v}_{\mathcal{F}}(k+1) = (\mathbf{I} - 2\mu_{\text{o}}\boldsymbol{\Lambda}^{-1}\boldsymbol{\mathcal{P}}_{N,0}\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)\boldsymbol{\mathcal{P}}_{0,L}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k))\mathbf{v}_{\mathcal{F}}(k) \\ + 2\mu_{\text{o}}\boldsymbol{\Lambda}^{-1}\boldsymbol{\mathcal{P}}_{N,0}\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)\boldsymbol{\mathcal{P}}_{0,L}\mathbf{e}_{\text{o},\mathcal{F}}(k)\end{aligned} \tag{8B-30}$$

analogous to (8B-19). Following the same line of derivations as in the case of (8B-19), we obtain

$$\begin{aligned}-2\mu_{\text{o}}\boldsymbol{\Lambda}^{-1}\boldsymbol{\mathcal{P}}_{N,0}\boldsymbol{\mathcal{R}}_{xx}^{\text{u}}\mathbf{K}_{\mathcal{F}}(k) - 2\mu_{\text{o}}\mathbf{K}_{\mathcal{F}}(k)\boldsymbol{\mathcal{R}}_{xx}^{\text{u}}\boldsymbol{\mathcal{P}}_{N,0}\boldsymbol{\Lambda}^{-1} \\ + 4\mu_{\text{o}}^{2}N'\xi_{\min}\boldsymbol{\Lambda}^{-1}\boldsymbol{\mathcal{P}}_{N,0}\boldsymbol{\mathcal{R}}_{xx}^{\text{u}}\boldsymbol{\mathcal{P}}_{N,0}\boldsymbol{\Lambda}^{-1} = 0.\end{aligned} \tag{8B-31}$$

We shall now solve this equation to find $\mathbf{K}_{\mathcal{F}}(k)$.

To proceed, let us define

$$\mathbf{G} = \boldsymbol{\Lambda}^{-1}\boldsymbol{\mathcal{P}}_{N,0}\boldsymbol{\mathcal{R}}_{xx}^{\text{u}} \tag{8B-32}$$

and note that $\mathbf{G}^{\text{H}} = \boldsymbol{\mathcal{R}}_{xx}^{\text{u}}\boldsymbol{\mathcal{P}}_{N,0}\boldsymbol{\Lambda}^{-1}$ since $\boldsymbol{\mathcal{P}}_{N,0}$ is Hermitian and $\boldsymbol{\mathcal{R}}_{xx}^{\text{u}}$ and $\boldsymbol{\Lambda}^{-1}$ are diagonal matrices. Using these, (8B-31) may be rearranged as

$$\mathbf{G}\mathbf{K}_{\mathcal{F}}(k) - \mu_{\text{o}}N'\xi_{\min}\mathbf{G}\boldsymbol{\mathcal{P}}_{N,0}\boldsymbol{\Lambda}^{-1} + \mathbf{K}_{\mathcal{F}}(k)\mathbf{G}^{\text{H}} - \mu_{\text{o}}N'\xi_{\min}\boldsymbol{\Lambda}^{-1}\boldsymbol{\mathcal{P}}_{N,0}\mathbf{G}^{\text{H}} = 0 \tag{8B-33}$$

or

$$\mathbf{G}(\mathbf{K}_{\mathcal{F}}(k) - \mu_{\text{o}}N'\xi_{\min}\boldsymbol{\mathcal{P}}_{N,0}\boldsymbol{\Lambda}^{-1}) + (\mathbf{K}_{\mathcal{F}}(k) - \mu_{\text{o}}N'\xi_{\min}\boldsymbol{\Lambda}^{-1}\boldsymbol{\mathcal{P}}_{N,0})\mathbf{G}^{\text{H}} = 0 \tag{8B-34}$$

The general solution of (8B-34) turns out to be difficult. However, a trivial solution of that, which closely matches the simulation results, can be easily identified as

$$\mathbf{K}_{\mathcal{F}}(k) = \mu_{\text{o}}N'\xi_{\min}\boldsymbol{\mathcal{P}}_{N,0}\boldsymbol{\Lambda}^{-1}. \tag{8B-35}$$

Substituting (8B-35) in (8B-7) we get

$$\xi_{\text{excess}}^{\text{c},\mathcal{N}} = \mu_{\text{o}}\xi_{\min}\frac{1}{L}\text{tr}[\boldsymbol{\mathcal{P}}_{N,0}\boldsymbol{\Lambda}^{-1}\boldsymbol{\mathcal{R}}_{xx}^{\text{u}}]. \tag{8B-36}$$

Using (8B-23) in (8B–36) we obtain

$$\xi_{\text{excess}}^{\text{c},\mathcal{N}} = \mu_{\text{o}}\xi_{\min}\frac{1}{N'}\text{tr}[\boldsymbol{\mathcal{P}}_{N,0}]. \tag{8B-37}$$

Noting that

$$\text{tr}[\boldsymbol{\mathcal{P}}_{N,0}] = \text{tr}[\boldsymbol{\mathcal{F}}\mathbf{P}_{N,0}\boldsymbol{\mathcal{F}}^{-1}] = \text{tr}[\boldsymbol{\mathcal{F}}^{-1}\boldsymbol{\mathcal{F}}\mathbf{P}_{N,0}] = \text{tr}[\mathbf{P}_{N,0}] = N,$$

from (8B-37) we get

$$\xi_{\text{excess}}^{\text{c},\mathcal{N}} = \mu_{\text{o}}\xi_{\text{min}}\frac{N}{N'}. \tag{8B-38}$$

Substituting (8B-38) in (8B-1) we obtain the misadjustment for the constrained FBLMS algorithm with step-normalization as

$$\mathcal{M}_{\text{FBLMS}}^{\text{c},\mathcal{N}} = \mu_{\text{o}}\frac{N}{N'}. \tag{8B-39}$$

a deeper study on multirate signal processing, the reader may refer to Crochiere and Rabiner (1983) or Vaidyanathan (1993), for example.

Successful implementation of subband adaptive filters requires careful design of analysis and synthesis filters. Much of our effort in this chapter is thus devoted to the design of analysis and synthesis filters which are suitable for subband adaptive filtering.

9.1 DFT Filter Banks

Consider the case where a sequence, $x(n)$, has to be separated into a number of subbands. For this, we may start with a lowpass filter, $H(z)$, and proceed as follows. By passing $x(n)$ through $H(z)$, the low-frequency part of its spectrum is extracted. To extract any other part of the spectrum of $x(n)$, e.g. the part centred around the frequency $\omega = \omega_i$, we may shift the desired portion of the spectrum to the base-band (i.e. around $\omega = 0$) by multiplying $x(n)$ with the complex sinusoid $e^{-j\omega_i n}$, and then use the lowpass filter $H(z)$ to extract that. The filter $H(z)$, which is repeatedly used for extraction of different parts of the input spectrum, is called the *prototype* filter.

Using this method, a sequence, $x(n)$, can be partitioned into any set of arbitrary bands. Since the separated subband signals are in base-band and have a smaller bandwidth than the original full-band signal, they have a lower Nyquist rate and thus may be decimated (down-sampled) to a lower rate before any further processing. Figure 9.1 depicts the steps required for partitioning a sequence $x(n)$ into M equally spaced subbands, centred at frequencies

$$\omega_i = \frac{2\pi i}{M}, \quad \text{for} \quad i = 0, 1, \ldots, M-1,$$

and decimating the subband signals using a decimation factor L. The structure of Figure 9.1 is known as the *DFT analysis filter bank*, for reasons that will become clear shortly. In Figure 9.1, decimation is denoted by a downward arrow followed by the decimation factor, L. We may also note that, in Figure 9.1, the time index n is used for the full-band input sequence $x(n)$. In contrast, we use the time index k for subband sequences. These choices of time indices will be consistently followed throughout this chapter. Further-

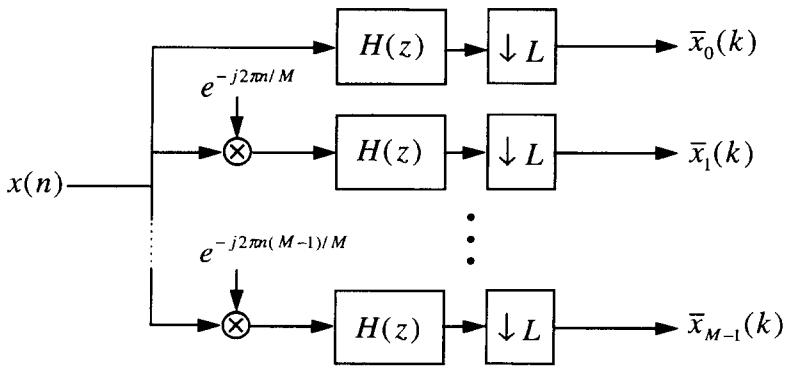


Figure 9.1 DFT analysis filter bank

more, the subband signals are represented by overbar variables, such as the $\bar{x}_i(k)$s in Figure 9.1, so as to distinguish from full-band signals.

We may note that in the structure of Figure 9.1 there is no restriction on the bandwidth of the prototype filter, $H(z)$, the number of subbands, M, and the decimation factor, L. Thus, there may be some overlap between different subbands. However, if L is chosen too large, the decimated subband signals may suffer from aliasing effects. Although aliasing is not desirable in most applications, we will show later that a small amount of aliasing may be beneficial in the implementation of subband adaptive filters.

A general procedure for efficient realization of the DFT filter banks, for any choice of L and M, is the *weighted overlap–add method.* When M is a multiple of L, a slightly different procedure which leads to the so called *polyphase* filter bank structure may be more useful from the point of view of *computational complexity* (see Crochiere and Rabiner, 1983, or Vaidyanathan, 1993). Since M is not necessarily a multiple of L in most applications of subband adaptive filters, we only discuss the weighted overlap–add method in the rest of this section.

9.1.1 The weighted overlap–add method for the realization of DFT analysis filter banks

To begin, let us define

$$W_M = e^{j(2\pi/M)},$$

where $j = \sqrt{-1}$. Then, the ith output of the DFT analysis filter bank may be expressed as[1]

$$\bar{x}_i(k) = \sum_{n=-\infty}^{\infty} h_n \tilde{x}_i(kL - n), \tag{9.1}$$

where

$$\tilde{x}_i(n) = x(n) W_M^{-in} \tag{9.2}$$

is the modulated version of the input, $x(n)$ (see Figure 9.1). Replacing n by $-n$, (9.1) may be rearranged as

$$\bar{x}_i(k) = \sum_{n=-\infty}^{\infty} h_{-n} \tilde{x}_i(kL + n). \tag{9.3}$$

Substituting (9.2) in (9.3) we get

$$\bar{x}_i(k) = W_M^{-ikL} \sum_{n=-\infty}^{\infty} h_{-n} x(kL + n) W_M^{-in}. \tag{9.4}$$

[1] We note that, in practice, the sequence h_n is always causal (i.e. $h_n = 0$, for $n < 0$) and has a finite duration. However, we let n to vary from $-\infty$ to $+\infty$ to keep the derivations simple.

Now, the method of *time aliasing* may be applied to the summation on the right-hand side of (9.4) for its evaluation in an efficient manner. To this end, we define the sequence

$$u_k(n) = h_{-n}x(kL + n) \tag{9.5}$$

and note that $u_k(n)$ is a windowed version of the input sequence, $x(n)$, the window being the time reverse of the prototype filter, h_n. Using (9.5) in (9.4) we get

$$\bar{x}_i(k) = W_M^{-ikL} \sum_{n=-\infty}^{\infty} u_k(n) W_M^{-in}. \tag{9.6}$$

With a change of variable $n = r + lM$ and noting that $W_M^{-ilM} = 1$, (9.6) may be rearranged as

$$\bar{x}_i(k) = W_M^{-ikL} \sum_{r=0}^{M-1} u_k^{\mathrm{a}}(r) W_M^{-ir}, \tag{9.7}$$

where

$$u_k^{\mathrm{a}}(r) = \sum_{l=-\infty}^{\infty} u_k(r + lM), \quad \text{for } r = 0, 1, \ldots, M-1. \tag{9.8}$$

We note that the M-point sequence $u_k^{\mathrm{a}}(r)$ is obtained by subdividing the sequence $u_k(n)$ into blocks of M samples and stacking and adding (i.e. time aliasing) these blocks.

From (9.7) we note that the subband signal samples, $\bar{x}_i(k)$, for $i = 0, 1, \ldots, M-1$, can be computed simultaneously, once the time aliased sequence $u_k^{\mathrm{a}}(r)$ is obtained. This is done by applying an M-point DFT to the samples $u_k^{\mathrm{a}}(r)$, for $r = 0, 1, \ldots, M-1$, and multiplying the DFT outputs by the coefficients W_M^{-ikL}, as suggested in (9.7). Furthermore, computation of the DFT may be done by using an efficient FFT algorithm.

9.1.2 The weighted overlap–add method for the realization of DFT synthesis filter banks

Consider the case where the subband signals $\bar{y}_i(k)$, for $i = 0, 1, \ldots, M-1$, are to be synthesized to reconstruct the full-band signal $y(n)$. Also, assume that these subband signals are in the baseband and at a decimated rate L times lower than the full-band rate. To generate $y(n)$, we may proceed as follows:

1. By appending $L-1$ zeros after every sample of subband signals, these signals are expanded to the full-band rate. This is referred to as *interpolation* and, accordingly, L is called the *interpolation factor*. Interpolation results in a set of full-band signals whose spectra consist of L repetitions of their associated baseband spectra (see Oppenheim and Schafer, 1989, for example).
2. The repetitions of the baseband spectra are removed by the lowpass filter.
3. The lowpass filtered full-band signals are then shifted to their respective bands, through appropriate modulators.

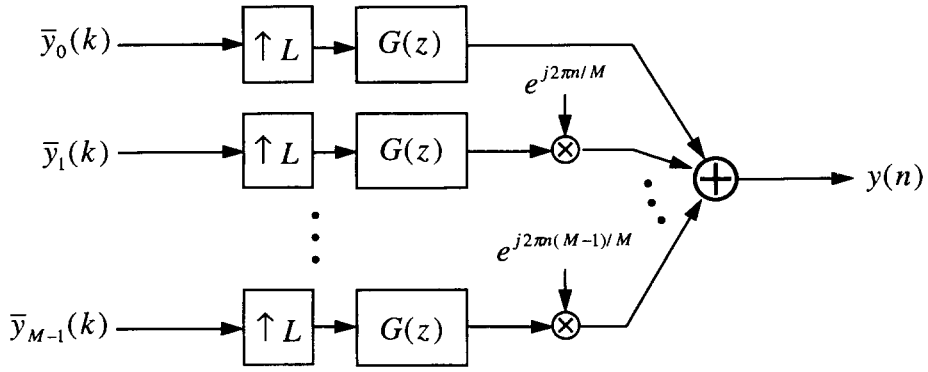


Figure 9.2 DFT synthesis filter bank

The combination of Steps 1, 2 and 3 can be expressed mathematically as

$$y_i(n) = W_M^{in} \sum_{k=-\infty}^{\infty} \bar{y}_i(k) g_{n-kL}, \quad \text{for } i = 0, 1, \ldots, M-1, \tag{9.9}$$

where the sequence g_n is the impulse response of the lowpass filter and the coefficients W_M^{in} are the modulating factors. The fact that the samples added in step 1, to expand subband signal sequences to full-band, are zero has been used to arrive at the special form of the summation on the right-hand side of (9.9). Verification of this is left to the reader as an exercise (see Problem P9.1)

4. Finally, the full-band signals, the $y_i(n)$s, are added together to obtain the synthesized sequence

$$y(n) = \frac{1}{M} \sum_{i=0}^{M-1} y_i(n). \tag{9.10}$$

The factor $1/M$ in (9.10) is added for convenience.

Figure 9.2 presents the block diagram of a synthesis filter bank, where interpolation is denoted by an upward arrow followed by the interpolation factor, L.

To obtain an efficient realization of synthesis filter banks, we proceed as follows. Substituting (9.9) in (9.10) and rearranging, we obtain

$$y(n) = \sum_{k=-\infty}^{\infty} g_{n-kL} \frac{1}{M} \sum_{i=0}^{M-1} \bar{y}_i(k) W_M^{in}. \tag{9.11}$$

Next, we define the following full-band sequence:

$$\hat{y}_k(n) = g_n \frac{1}{M} \sum_{i=0}^{M-1} \bar{y}_i(k) W_M^{i(n+kL)}. \tag{9.12}$$

Then, using (9.12) we can write (9.11) as

$$y(n) = \sum_{k=-\infty}^{\infty} \hat{y}_k(n - kL). \tag{9.13}$$

That is, the output sequence $y(n)$ is obtained by overlapping and adding the $\hat{y}_k(n)$ sequences, thus the name overlap–add.

Equation (9.12) may also be written as

$$\hat{y}_k(n) = g_n \tilde{y}_k(n) \tag{9.14}$$

where

$$\tilde{y}_k(n) = \frac{1}{M} \sum_{i=0}^{M-1} [\overline{y}_i(k) W_M^{ikL}] W_M^{in}. \tag{9.15}$$

Note that $\tilde{y}_k(n)$ is a periodic function of n with period M since W_M^{in} is periodic in n with period M, and the rest of the terms on the right-hand side of (9.15) are independent of n. Furthermore, it is straightforward to see that the values of $\tilde{y}_k(n)$, for $n = 0, 1, \ldots, M-1$ (i.e. the first period of $\tilde{y}_k(n)$), are samples of the inverse DFT of the sequence $\overline{y}_i(k) W_M^{ikL}$, for $i = 0, 1, \ldots, M-1$.

From the above observation, we may adopt the following procedure to generate the samples of the synthesized output sequence, $y(n)$:

1. Upon the receipt of the latest samples of the subband signals, say $\overline{y}_i(k)$, for $i = 0, 1, \ldots M-1$, we construct the vector

$$\overline{\mathbf{y}}(k) = [\overline{y}_0(k) \;\; \overline{y}_1(k) W_M^{kL} \;\; \overline{y}_2(k) W_M^{2kL} \;\; \cdots \;\; \overline{y}_{M-1}(k) W_M^{k(M-1)L}]$$

   and compute the inverse DFT of $\overline{\mathbf{y}}(k)$.
2. The result of this inverse DFT is repeated to generate a periodic sequence. This makes the sequence $\tilde{y}_k(n)$ of (9.15).
3. The sequence $\hat{y}_k(n)$ is obtained by multiplying the sequences $\tilde{y}_k(n)$ and g_n on an element-by-element basis, as in (9.14). Assuming that g_n is causal, $\hat{y}_k(n)$ will also be causal.
4. Finally, to generate the samples of $y(n)$, the sequence $\hat{y}_k(n)$ is added to a buffer holding the accumulated results of the previous iterations, i.e. $\sum_{l=-\infty}^{k-1} \hat{y}_l(n - lL)$. The first L elements of the updated buffer are the samples $y(kL), y(kL+1), \ldots, y(kL+L-1)$ of the synthesized output. While these samples are being sent to the output, the content of the buffer is shifted and filled with zeros from its other end and becomes ready for stacking the next set of samples, i.e. $\hat{y}_{k+1}(n)$, in the next iteration.

9.2 Complementary Filter Banks

In multirate signal processing, in general, analysis and synthesis filters need to satisfy certain conditions in order that the reconstructed full-band signals have no, or at least insignificant, distortion. For this to be true in subband adaptive filters, we find that the

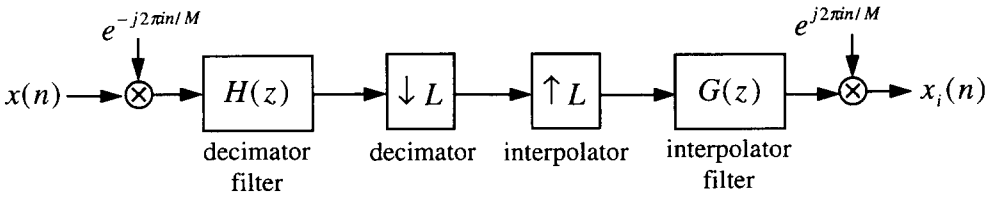


Figure 9.3 The ith channel of an M-band analysis–synthesis DFT filter bank

combined responses of the analysis and synthesis filters should be that of a *complementary filter bank*.

To explain what we mean by a complementary filter bank and also to derive the conditions required for a filter bank to be complementary, consider the ith channel (frequency band) of a pair of analysis–synthesis filter banks, as depicted in Figure 9.3. Figure 9.4 presents a set of plots showing the results of the various stages of Figure 9.3. Figure 9.4(a) shows a representative graph of the spectrum of the full-band input, $x(n)$. The portion of the spectrum of $x(n)$ that is centred around $\omega_i = 2\pi i/M$ is shifted to $\omega = 0$ and lowpass filtered through the decimator filter, $H(z)$. Let us choose $\omega_i = \pi/2$ and $L = 4$ for this example. Furthermore, let the lowpass filter $H(z)$ be an ideal filter with unit gain over the frequency range $-\pi/4 \leq \omega \leq \pi/4$ and zero elsewhere. Then, the spectrum of the output of $H(z)$ will be as shown in Figure 9.4(b). The decimation, which compresses the output of $H(z)$ along the time axis, results in expansion of the spectrum along the frequency axis, as in Figure 9.4(c). The interpolator, in contrast, expands the signal samples along the time axis and thus results in compression of the spectrum as in Figure 9.4(d). This leads to L repetitions of the spectrum of the decimated signal over the range $0 \leq \omega \leq 2\pi$. The interpolator filter, $G(z)$, selects the baseband part of the repeated spectrum and rejects its repetitions, thereby recovering the lowpass spectrum of Figure 9.4(b). Finally, the output of $G(z)$ is shifted to its respective band through a modulator. This results in a full-band signal $x_i(n)$ which is a bandpass filtered portion of the input, $x(n)$, as in Figure 9.4(e).

From the above example we also note that the effect of the decimator–interpolator blocks in Figure 9.3 is to repeat the baseband spectrum of Figure 9.4(b) as in Figure 9.4(d). However, since these repetitions are in turn rejected by the synthesis filter, we may delete these blocks from Figure 9.3, without affecting its input–output relationship. Furthermore, we can easily show that the combination of the modulator stages (i.e. multiplication of the input, $x(n)$, by $e^{-j2\pi in/M}$ and the interpolator filter output by $e^{j2\pi in/M}$) and the lowpass filters $H(z)$ and $G(z)$ is equivalent to the cascade of the bandpass filters $H(z e^{-j2\pi i/M})$ and $G(z e^{-j2\pi i/M})$ (see Problem P9.2). In an M-band analysis–synthesis filter bank, there are M such pairs of filters in parallel, as shown in Figure 9.5. For a sequence $x(n)$ to pass through this bank of filters without distortion, the overall transfer function of the system should resemble that of a pure delay. That is

$$\sum_{i=0}^{M-1} F(z e^{-j2\pi i/M}) = z^{-\Delta}, \tag{9.16}$$

where $F(z) = H(z)G(z)$ and Δ is the delay introduced by the cascade of the analysis and synthesis filters.

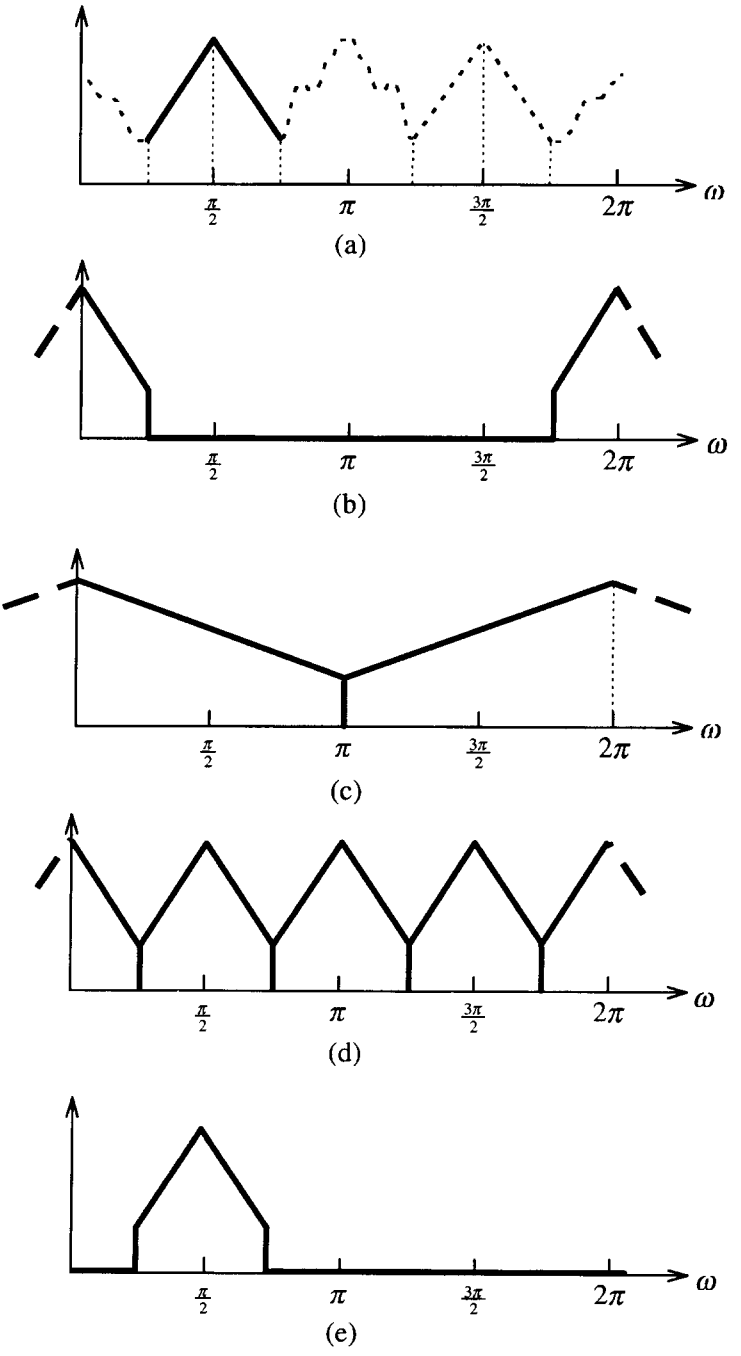


Figure 9.4 Spectra of the signal sequences at various stages of Figure 9.3: (a) input signal, $x(n)$, (b) decimator filter output, (c) decimator output, (d) interpolator output, (e) final output, $x_i(n)$

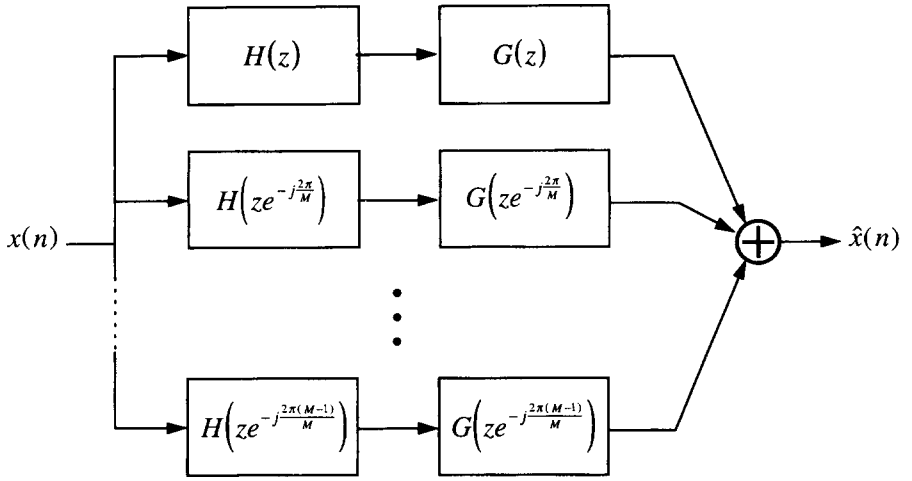


Figure 9.5 An equivalent block diagram of an M-band analysis–synthesis DFT filter bank

When (9.16) holds, we say that the filter bank is *complementary*. The complementary condition (9.16) implies that $\hat{x}(n) = x(n - \Delta)$. That is, the reconstructed signal, $\hat{x}(n)$, at the synthesis bank output is a delayed replica of the input, $x(n)$. Figure 9.6 gives a pictorial representation of the concept of complementary filters, where the magnitude responses of the filters $F(z\,e^{-j2\pi i/M}) = H(z\,e^{-j2\pi i/M})G(z\,e^{-j2\pi i/M})$ of a four-band filter bank are plotted for $i = 0, 1, 2$ and 3. As shown, there is some overlap among neighbouring filters. However, the filters are chosen so that the overall response adds up to unity across the full-band.

Figure 9.6, as well as equation (9.16), states that the condition in the frequency domain that should be satisfied for the filter bank to be complementary. In the design of complementary filter banks, however, we often find that it is more convenient to work with time domain constraints. So, we convert the constraint specified by (9.16) to its equivalent in the time domain. For this, we define the sequence f_n as the inverse z-transform of $F(z)$. That is,

$$F(z) = \sum_{n=-\infty}^{\infty} f_n z^{-n}. \tag{9.17}$$

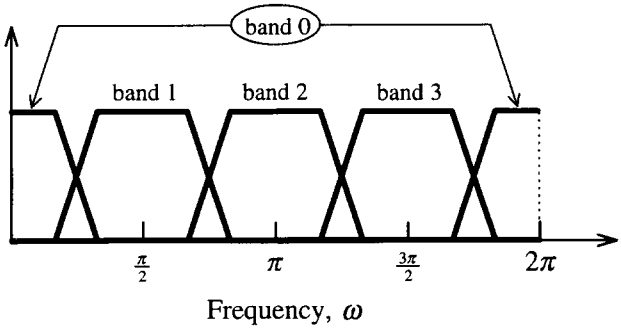


Figure 9.6 A pictorial representation of the concept of complementary filter banks

Using (9.17) in (9.16) and rearranging, we obtain

$$\sum_{n=-\infty}^{\infty} \left(\sum_{i=0}^{M-1} e^{-j(2\pi in)/M} \right) f_n z^{-n} = z^{-\Delta}. \tag{9.18}$$

Furthermore, it is straightforward to show that

$$\sum_{i=0}^{M-1} e^{-j(2\pi in)/M} = \begin{cases} M, & \text{when } n \text{ is a multiple of } M, \\ 0, & \text{otherwise.} \end{cases} \tag{9.19}$$

Using (9.19) in (9.18), we find that equation (9.18) can only be satisfied when $\Delta = KM$, where K is a positive integer, and

$$f_n = \begin{cases} 1/M, & n = KM, \\ 0, & n = \text{all multiples of } M \text{ except } KM, \\ \text{unspecified}, & \text{otherwise.} \end{cases} \tag{9.20}$$

Thus, the value of K determines the *total delay* introduced by the filter bank.

9.3 Subband Adaptive Filter Structures

Figure 9.7 depicts the schematic of a commonly used structure of subband adaptive filters.[2] The adaptive filter is used to model a plant, $W_o(z)$. The input, $x(n)$, and the plant output, $d(n)$, are passed through a pair of identical analysis filter banks to be partitioned into M subbands and decimated to a rate that is $1/L$ of the full-band rate. The subband adaptive filters, the $\overline{W}_i(z)$s, are thus running at a rate that is only $1/L$ of the full-band rate. To generate the adaptive filter output in the full band, the outputs from the subband filters are combined through a synthesis filter bank.

The subband adaptive filter structure presented in Figure 9.7 is referred to as *synthesis independent*, since the adaptation of the subband filters is independent of the synthesis filters. The assumption here is that the synthesis filters are ideal, in the sense that their stop-band attenuation is infinity and their cascade with the analysis filters results in a complementary filter bank. In practice, these ideal requirements can be satisfied only approximately. Hence, the synthesis independent subband adaptive filters are bound to have some distortion. This distortion can be reduced by using an alternative structure which is known as the *synthesis dependent* subband adaptive filter. This is shown in Figure 9.8. The delay Δ is to account for the combined delay due to the analysis and synthesis filters. In this structure, even though the filtering is still done in subbands, the computation of the output error, $e(n)$, is done in the full band. The full-band error, $e(n)$, is subsequently partitioned into subbands using an analysis filter bank, and the subband errors, the $\bar{e}_i(k)$s, are used for the adaptation of the associated subband filters.

[2] The concept of subband adaptive filtering was first introduced by Furukawa (1984) and Kellermann (1984, 1985).

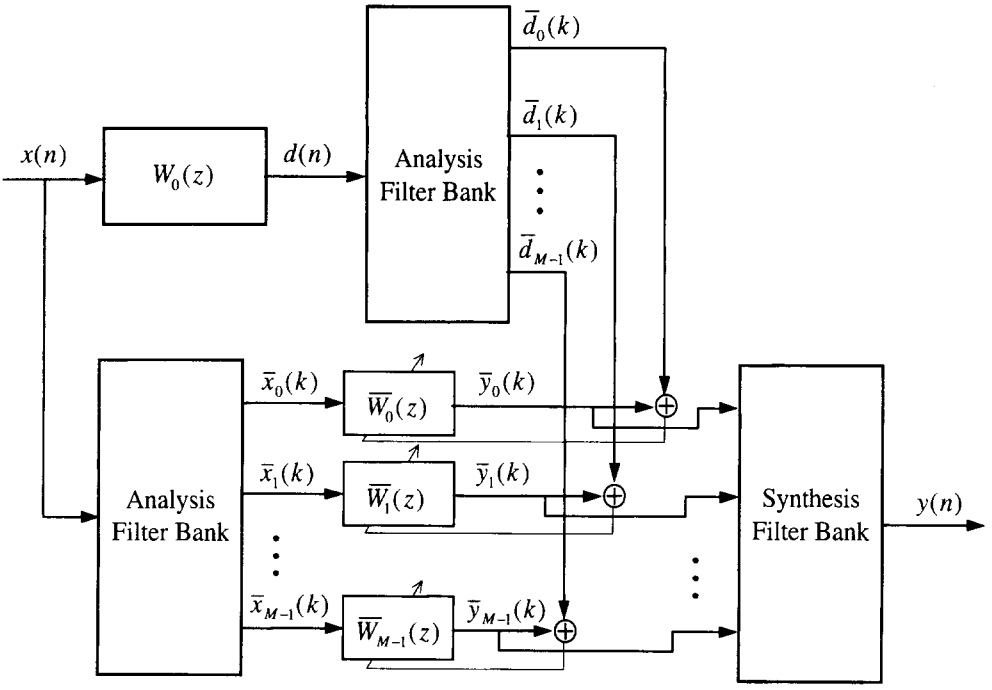


Figure 9.7 Subband adaptive filter (synthesis independent structure)

The synthesis dependent structure, although resolving the distortion introduced by the synthesis filters, has some drawbacks which hinder its application in practice (Sondhi and Kellermann, 1992). In particular, the cascade of synthesis and analysis filter banks in the adaptation loop introduces an undesirable delay which makes the filter more prone to instability. Furthermore, the presence of a delay in the adaptation loop increases the memory requirement of the filter (see Problem P9.3). Because of these problems, the synthesis dependent subband adaptive filter structure has been less popular than its synthesis independent counterpart. Noting this, our emphasis in the rest of this chapter will be on the synthesis independent structure. Nevertheless, most of the results we develop are applicable to the synthesis dependent structure as well.

9.4 Selection of Analysis and Synthesis Filters

The design of analysis and synthesis filters with well-behaved responses is crucial to the successful implementation of subband adaptive filters. In this section we look into the basic requirements of the analysis and synthesis filters. We note that there are many requirements that should be taken into account while selecting these filters and hence a compromise has to be struck to achieve an acceptable design. As a result, it is very difficult to give any specific criterion whose optimization will lead to the optimum set of filters. Instead, we find it more appropriate to deal with this problem in a subjective manner which would lead us to a number of specifications for a good compromise design.

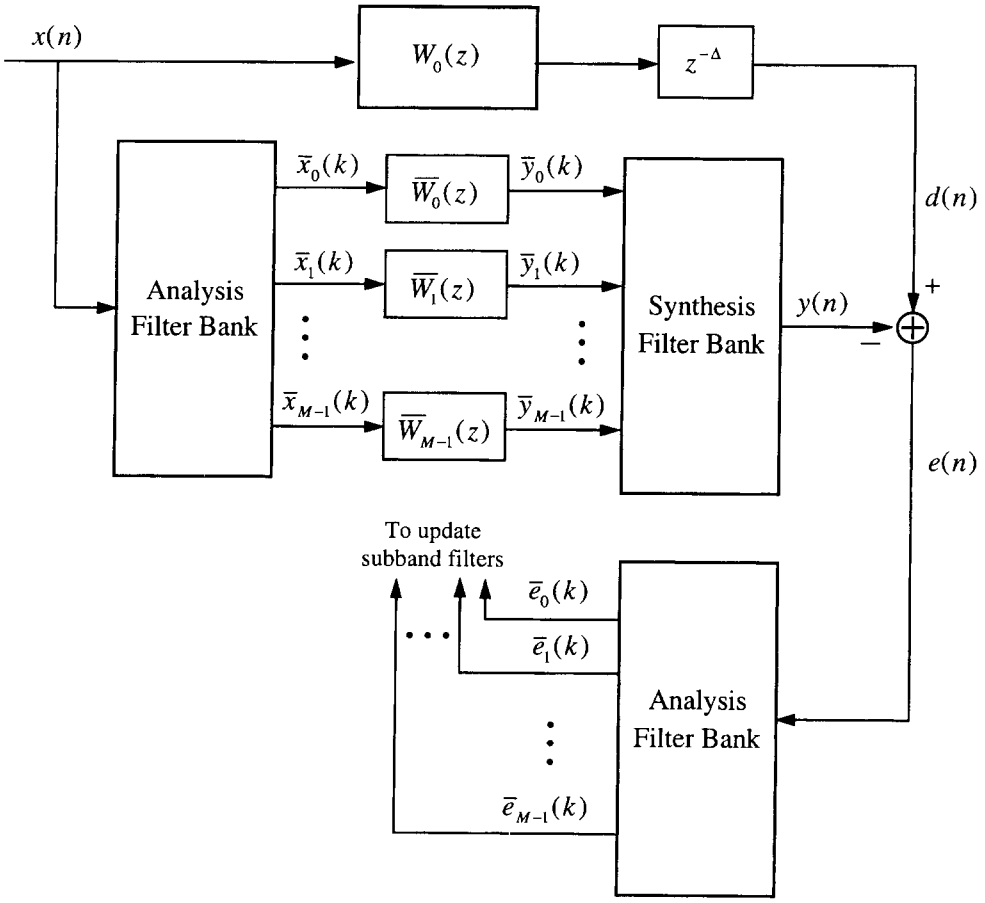


Figure 9.8 Subband adaptive filter (synthesis dependent structure)

As was noted earlier in Section 9.2, for the reconstructed output of a subband adaptive structure to have small distortions, the analysis and synthesis filters should form a complementary filter bank. There are many pairs of analysis and synthesis filter banks that satisfy the complementary condition. This provides some degrees of freedom which may be used to facilitate the design and/or enhance the performance of the subband adaptive filters.

A first attempt may be to use the same prototype filter for both analysis and synthesis. Unfortunately, this leads to subband signals whose spectra vary and decay to some small values near the ends of their respective bands. This, in turn, will result in the inputs to the subband adaptive filters being badly conditioned because of the low excitation levels near the band edges. Furthermore, from our discussions in the previous chapters, we know that such inputs will result in large eigenvalue spreads and thus poor convergence. This problem may be resolved as follows (Morgan, 1995, and De León and Etter, 1995).

Figure 9.9 presents a diagram showing a good choice of analysis and synthesis prototype filters which resolves the problem of slow convergence of subband adaptive

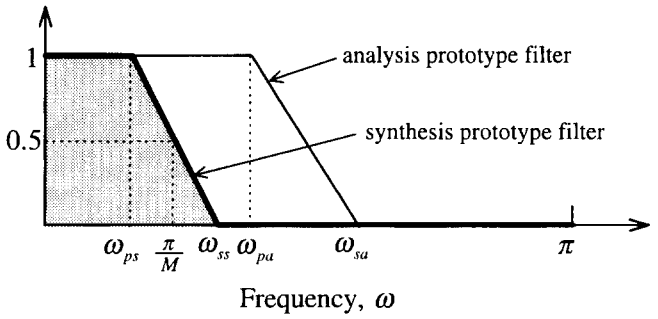


Figure 9.9 A possible choice of analysis and synthesis prototype filters which resolves the problem of slow convergence of subband adaptive filters

filters. The analysis prototype filter is chosen such that it has a flat magnitude response and linear phase response (constant group-delay[3]) between zero and a frequency larger than or equal to ω_{ss}, where ω_{ss} is the beginning of the stop-band (i.e. the end of the transition band) of the synthesis prototype filter. Moreover, the synthesis filters are chosen to be complementary. The cascade of the analysis and synthesis filters will then be a complementary filter bank since in this case the multiplication (cascade) of the analysis and synthesis prototype filters is just the same as the synthesis prototype filter. The analysis filters only introduce a fixed delay in the overall response of the subband structure.

Next, we explain why the choice of the analysis and synthesis prototype filters as in Figure 9.9 resolves the problem of poor convergence of subband adaptive filters. Assuming that the power spectral density of the full-band input, $x(n)$, does not vary significantly over each subband, using an analysis prototype filter similar to the one shown in Figure 9.9 would result in all decimated subband sequences having approximately flat spectra over the range of frequencies $|\omega| \leq \omega_{pa}$. On the other hand, the band of interest over which matching between the frequency response of each subband adaptive filter and its associated desired response from the respective band of the plant should be achieved is $|\omega| < \omega_{ss}$, since frequencies beyond this are cut off by the synthesis filters (see Figure 9.9). We may thus say that in a subband adaptive filter structure whose analysis and synthesis prototype filters are selected as in Figure 9.9, all the subband filters will be well excited over their respective bands of interest, and hence there will not be any slow mode which may affect the convergence behaviour of the overall filter.

Another consideration that should be noted in the implementation of subband adaptive filters, and hence in the design of analysis and synthesis filters, is the problem of delay (or latency) in the filter output, $y(n)$. This delay is caused by the analysis and synthesis filters. Minimization of this delay is exceedingly important since the maximum delay permitted in many applications is often very limited. For instance, in the application of acoustic echo cancellation, around which most of the theory of subband adaptive filters has been developed, the maximum delay allowed is usually very

[3] The group-delay of a system is defined as the derivative of its phase response with respect to the angular frequency, ω.

minimal.[4] The factors that influence the delay introduced by the analysis and synthesis filters are the number of subbands, M, the decimation factor, L, the accuracy of the analysis and synthesis filters (which may be defined in terms of their stop-band attenuation and pass-band ripple), and also the criterion used in designing analysis and synthesis filters. The last two issues are addressed in Section 9.7, where a method for designing analysis and synthesis filters with small delay is given.

The delay increases with the number of subbands, M. On the other hand, we may recall from our previous discussion that the idea of subband adaptive filtering is to partition the input signal into a number of narrow bands such that the signal spectrum is approximately flat over each band, thus giving an implementation which does not suffer due to large eigenvalue spreads. Hence, from a convergence point of view, larger values of M are preferred. The choice of the decimation factor, L, also affects the selection of the analysis and synthesis filters and hence the delay. In general, the delay increases with L as well. On the other hand, the computational complexity of a subband adaptive structure decreases as L increases. Thus, a compromise has to be struck when choosing L and M.

From the above discussion we find that the selection of the analysis and synthesis filters is not a straightforward or a clearly formulated problem. On the one hand, we should make sure that the delay introduced by the analysis and synthesis filters does not exceed a specified value. This is usually specified as one of the design requirements. On the other hand, we may choose the number of subbands, M, and the decimation factor, L, as large as possible, while designing analysis and synthesis filters with some (loosely defined) acceptable aspects. A procedure for the design of analysis and synthesis filters as well as selection of the values of L and M and the other parameters of the subband adaptive filters are given in Sections 9.7 and 9.8.

9.5 Computational Complexity

The computational complexity of subband adaptive filters, in general, decreases as the decimation factor, L, increases. To explore the impact of L in reducing the computational complexity of a subband adaptive filter, let us consider the implementation of an adaptive filter whose full-band implementation requires N taps. We also assume that the filter input, $x(n)$, and the desired signal, $d(n)$, are real-valued. The number of taps required for each subband filter is then N/L, since in subbands each sample interval is equivalent to L sample intervals in the full band. We also note that although the input, $x(n)$, is real-valued, the subband signals are, in general, complex-valued. They also appear in complex-conjugate pairs, with the exceptions of bands 0 and $M/2$ (we assume that M is even) whose corresponding inputs are real-valued, when the input, $x(n)$, is real-valued. Considering these two bands as one band with complex-valued input, the computational complexity of a subband adaptive filter with N real-valued full-band taps may be evaluated on the basis of $(M/2)(N/L) = MN/2L$ complex-valued subband taps. Furthermore, we note that processing in subbands is done at a rate that is only $1/L$ of the full-band rate. Noting these and counting each complex-valued tap as equivalent

[4] In the ITU-T standard G.167 it is stated that 'for end-to-end digital communications (for example wide-band teleconference systems), the delay shall be no more than 16 ms in each direction of speech transmission'.

to four real-valued taps, we obtain

$$\frac{\text{complexity of subband filter}}{\text{complexity of full-band filter}} = \frac{2M}{L^2}. \tag{9.21}$$

This result does not include the complexity of the analysis and synthesis filters. However, in applications where subband adaptive filters are found useful, the filter length, N, is usually very large, in the range of 1000 or above. For such values of N, the contribution from the complexity of analysis and synthesis filters is not that significant (usually in the range of 20% or less).

In typical designs, one of which is given in Section 9.9, we usually find that $L \approx M/2$. Thus, we obtain

$$\frac{\text{complexity of subband filter}}{\text{complexity of full-band filter}} \approx \frac{4}{L} \approx \frac{8}{M}. \tag{9.22}$$

9.6 Decimation Factor and Aliasing

With the choice of the analysis and synthesis filters as in Figure 9.9, the largest value of L that may be used without causing aliasing of signal spectra over the bands of interest, i.e. those selected by the synthesis filters, is given by

$$L_{\max} = \left\lfloor \frac{2\pi}{\omega_{ss} + \omega_{sa}} \right\rfloor \tag{9.23}$$

where $\lfloor x \rfloor$ denotes the largest integer smaller than or equal to x and ω_{sa} and ω_{ss}, as indicated in Figure 9.9, denote the ends of the transition bands of the analysis and synthesis prototype filters, respectively. This choice of L will result in some aliasing in the outputs of the analysis filters. However, the aliased portions of the spectra are those that will be filtered out by the synthesis filters, and hence will not affect the full-band output of the filter. Figure 9.10 illustrates this, where we have plotted the magnitude responses of the analysis and synthesis filters after decimation. The selection of $L = L_{\max}$, although seeming quite reasonable at first glance, has some drawbacks when it comes to adaptation of the subband filters. It results in significant augmentation of the misadjustment, as explained next.

The Fourier transform of the desired signal, $\bar{d}_i(n)$, in the ith subband is given by

$$\bar{D}_i(e^{j\omega}) = \sum_{m=i-1}^{i+1} X(e^{j(\omega - 2\pi m)/L}) W_o(e^{j(\omega - 2\pi m)/L}) H(e^{j\omega/L}), \tag{9.24}$$

where $X(e^{j\omega})$ is the Fourier transform of the input, $x(n)$, and $W_o(e^{j\omega})$ and $H(e^{j\omega})$ are the frequency responses of the plant and the analysis prototype filter, respectively. The three

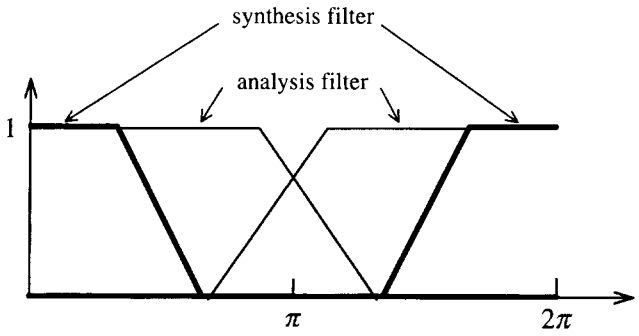


Figure 9.10 Magnitude responses of the analysis and synthesis filters after decimation, illustrating the fact that even though the decimated output samples of the analysis filters are aliased, the portion of the signal spectrum that is filtered by the synthesis filter is free of aliasing

ω, by L is due to the spectral expansion because of the L-fold decimation. Similarly, the Fourier transform of the output, $\bar{y}_i(k)$, of the ith subband filter is obtained as

$$\overline{Y}_i(e^{j\omega}) = \overline{W}_i(e^{j\omega}) \sum_{m=i-1}^{i+1} X(e^{j(\omega-2\pi m)/L}) H(e^{j\omega/L}). \tag{9.25}$$

Using (9.24) and (9.25), the Fourier transform of the subband error sequence $\bar{e}_i(k) = \bar{d}_i(k) - \bar{y}_i(k)$ is obtained as

$$\begin{aligned}
\overline{E}_i(e^{j\omega}) &= \overline{D}_i(e^{j\omega}) - \overline{Y}_i(e^{j\omega}) \\
&= [W_o(e^{j(\omega-2\pi i)/L}) - \overline{W}_i(e^{j\omega})] H(e^{j\omega/L}) X(e^{j(\omega-2\pi i)/L}) \\
&\quad + [W_o(e^{j(\omega-2\pi(i+1))/L}) - \overline{W}_i(e^{j\omega})] H(e^{j(\omega-2\pi)/L}) X(e^{j(\omega-2\pi(i+1))/L}) \\
&\quad + [W_o(e^{j(\omega-2\pi(i-1))/L}) - \overline{W}_i(e^{j\omega})] H(e^{j(\omega+2\pi)/L}) X(e^{j(\omega-2\pi(i-1))/L}).
\end{aligned} \tag{9.26}$$

Inspection of (9.26) reveals that to minimize

$$E[|e_i(k)|^2] = \frac{1}{2\pi} \int_{-\pi}^{\pi} |\overline{E}_i(e^{j\omega})|^2 \, d\omega,$$

$\overline{W}_i(e^{j\omega})$ has to be selected so that the three differences in the square brackets on the right-hand side of (9.26) reduce to some small values. Moreover, we note that the frequency interval $-\pi \leq \omega \leq \pi$, may be divided into three distinct ranges. The first range, defined as $-\omega_1 \leq \omega \leq \omega_1$, is where there is no overlap between $H(e^{j\omega/L})$ and its shifted versions, $H(e^{j(\omega+2\pi)/L})$ and $H(e^{j(\omega-2\pi)/L})$. In this range, the last two terms on the right-hand side of (9.26) are zero since this range coincides with the stop-bands of $H(e^{j(\omega+2\pi)/L})$ and $H(e^{j(\omega-2\pi)/L})$. The first term can also be made small by choosing $\overline{W}_i(e^{j\omega})$ to be close to the plant response, $W_o(e^{j(\omega-2\pi i)/L})$, for $-\omega_1 \leq \omega \leq \omega_1$. It is

important to note that selection of $L \leq L_{\max}$ implies that $\omega_{ss} \leq \omega_1$. This in turn means that the portions of the plant response that are picked up by the synthesis filters can be modeled well by the subband adaptive filters.

The second range, $\omega_1 < \omega \leq \pi$, is where the filters attached to the first two terms on the right-hand side of (9.26), i.e. $H(e^{j\omega/L})$ and $H(e^{j(\omega-2\pi)/L})$, overlap. In this range, for $|\overline{E}_i(e^{j\omega})|$ to reduce to a small value, $\overline{W}_i(e^{j\omega})$ has to match two different parts of the plant response; namely, $W_o(e^{j(\omega-2\pi i)/L})$ and $W_o(e^{j(\omega-2\pi(i+1))/L})$, for $\omega_1 < \omega \leq \pi$. This, of course, is not possible since $W_o(e^{j(\omega-2\pi i)/L}) \neq W_o(e^{j(\omega-2\pi(i+1))/L})$, in general. Similarly, in the third range also, where $-\pi \leq \omega < -\omega_1$, it may not be possible to reduce $|\overline{E}_i(e^{j\omega})|$ to a small value. As a result of these mismatches, $E[|e_i(k)|^2]$ may be very significant, even after the convergence of the subband adaptive filter. This will result in a large perturbation of the tap weights because of the use of stochastic gradients, thereby increasing the misadjustment of these filters, unless a very small step-size is used to reduce the level of perturbations. But, reducing the step-size is undesirable, since it proportionately reduces the convergence rate of the adaptive filter. Another solution that has been proposed to solve this problem is to add cross-filters between neighbouring subbands (Gilloire and Vetterli, 1992).[5] However, this increases the system complexity, and hence is not acceptable since the main goal of increasing L was to reduce the

complexity. Yet another solution, which is found to be more appropriate than the others, is to select the decimation factor, L, so that the overlapping of the adjacent analysis filters is limited only to those portions of the analysis filter responses that are below a certain level (Farhang-Boroujeny and Wang, 1997). However, no fixed value may be specified for this 'level'. It is a loosely defined design parameter which can only be found experimentally. Thus, a compromise value of L could only be selected through a trial-and-error design process (see Section 9.8).

9.7 Low-Delay Analysis and Synthesis Filter Banks

In this section we present a method for designing analysis and synthesis filters with low group-delay.[6] As was noted earlier, the design of low-delay filters is desirable in subband adaptive filters because it reduces the latency of the overall filter response.

9.7.1 Design method

From our discussion in Section 9.4 we recall that the analysis and synthesis filters should

Consider an FIR filter with length N_a and tap weights given by the real-valued coefficient-vector

$$\mathbf{a} = [a_0 \;\; a_1 \;\; \cdots \;\; a_{N_a-1}]^T,$$

where the superscript T denotes transposition. Then, the transfer function of the FIR filter is given by

$$A(e^{j\omega}) = \mathbf{a}^T\mathbf{\Omega}, \tag{9.27}$$

where

$$\mathbf{\Omega} = [1 \;\; e^{-j\omega} \;\; e^{-j2\omega} \cdots \;\; e^{-j(N_a-1)\omega}]^T.$$

Suppose we want this filter to have its stop-band to begin from ω_s. Then, the total energy in the stop-band is given by

$$E_s = \frac{1}{2\pi}\int_{\omega_s}^{2\pi-\omega_s} |A(e^{j\omega})|^2 \, d\omega. \tag{9.28}$$

Substituting (9.27) in (9.28), we obtain

$$E_s = \mathbf{a}^T\mathbf{\Phi}\mathbf{a}, \tag{9.29}$$

where

$$\mathbf{\Phi} = \frac{1}{2\pi}\int_{\omega_s}^{2\pi-\omega_s} \mathbf{\Omega}\mathbf{\Omega}^H \, d\omega \tag{9.30}$$

and the superscript H denotes Hermitian transposition. We note that $\mathbf{\Phi}$ is an $N_a \times N_a$ matrix whose klth element is

$$\phi_{kl} = \frac{1}{2\pi}\int_{\omega_s}^{2\pi-\omega_s} e^{-j\omega(k-l)} \, d\omega = \begin{cases} 1 - \dfrac{\omega_s}{\pi}, & k = l, \\ -\dfrac{\sin[\omega_s(k-l)]}{\pi(k-l)}, & k \neq l. \end{cases} \tag{9.31}$$

The optimum coefficients of the FIR filter are those that minimize the energy function E_s of (9.29). To prevent the trivial solution $a_i = 0$, for $i = 0, 1, \ldots, N_a - 1$, we impose the constraint $\mathbf{a}^T\mathbf{a} = 1$. The problem of minimizing E_s with respect to the vector $\mathbf{a}$, subject to the constraint $\mathbf{a}^T\mathbf{a} = 1$, is a standard eigenproblem whose optimum solution is the eigenvector of $\mathbf{\Phi}$ that corresponds to its minimum eigenvalue; see Property 7 of

coefficients a_i, for $i = 0, 1, \ldots, N_a - 1$, of the filter $A(e^{j\omega})$:

$$a_i = \begin{cases} 1/M, & i = KM, \\ 0, & i = \text{all multiples of } M \text{ except } KM, \\ \text{unspecified}, & \text{otherwise}, \end{cases} \tag{9.32}$$

where K is a constant integer which determines the group-delay of the filter bank, as discussed in Section 9.2.

To satisfy the conditions stated in (9.32), we may simply drop those a_is that have to be zero from (9.29). The new energy function to be minimized is then

$$\tilde{E}_s = \tilde{\mathbf{a}}^T \tilde{\mathbf{\Phi}} \tilde{\mathbf{a}}, \text{ subject to the constraint } \tilde{\mathbf{a}}^T \tilde{\mathbf{a}} = 1, \tag{9.33}$$

where $\tilde{\mathbf{a}}$ is obtained from $\mathbf{a}$ by deleting those elements that should be constrained to zero, and $\tilde{\mathbf{\Phi}}$ is obtained from $\mathbf{\Phi}$ by deleting the corresponding rows and columns so as to be made compatible with $\tilde{\mathbf{a}}$. The minimization of $\tilde{E}_s$ also is an eigenproblem. Its solution is the eigenvector of $\tilde{\mathbf{\Phi}}$ which corresponds to its minimum eigenvalue. The desired vector $\mathbf{a}$ is obtained from $\tilde{\mathbf{a}}$ by inserting the dropped-out zeros in the appropriate locations. Finally, to satisfy the condition $a_{KM} = 1/M$ of (9.32), a simple scaling is applied to $\mathbf{a}$.

We should note that the above procedure does not specify any specific range of frequencies for the pass-band and transition band. Only the stop-band is specified. To be more accurate on this, we recall that in an M-band complementary filter bank the frequency $\omega = \pi/M$ is located in the middle of the transition band of its prototype filter; see Figure 9.6 as an example. The stop-band of the prototype filter begins at $(1 + \alpha)\pi/M$, where α, known as the roll-off factor, determines the widths of the pass-band and transition band. The pass-band of the prototype filter is given as $0 \le \omega \le (1 - \alpha)\pi/M$ and the transition band as $(1 - \alpha)\pi/M < \omega < (1 + \alpha)\pi/M$. The numerical examples given next and the supporting discussions show that, for the filters designed by the proposed method, the pass-, stop- and transition bands will be clearly separated according to the above boundaries, once ω_s is set equal to $(1 + \alpha)\pi/M$.

9.7.2 Properties of the filters

In this subsection we look at the main features of the filters designed by the method presented above. We recall that an M-band complementary filter bank with the prototype filter $A(e^{j\omega})$ and the parameter K, as specified in (9.32), satisfies the equation

$$\sum_{i=0}^{M-1} A(e^{j(\omega - 2\pi i/M)}) = e^{-j\omega KM}. \tag{9.34}$$

On the other hand, the design procedure given in the previous subsection emphasizes only the stop-band of the prototype filter $A(e^{j\omega})$. But, there is no clear emphasis on how the pass-band and transition band of $A(e^{j\omega})$ are separated. Next, we show that the boundary between these two bands can be easily identified, once the design parameters M and ω_s are known.

From our discussion in Section 9.2, we recall that the mid-point of the transition band of the prototype filter of an M-band complementary filter bank is $\omega = \pi/M$. Moreover, from the pictorial representation of Figure 9.6 it is straightforward to conclude that if ω_p and ω_s are, respectively, the end of pass-band and the beginning of stop-band of the prototype filter of an M-band filter bank, then

$$\frac{\pi}{M} = \frac{\omega_p + \omega_s}{2}. \tag{9.35}$$

Hence, when M and ω_s are given, ω_p is obtained as

$$\omega_p = \frac{2\pi}{M} - \omega_s. \tag{9.36}$$

In the discussion that follows, it is convenient to specify ω_s in terms of the mid-point frequency π/M as

$$\omega_s = (1 + \alpha)\frac{\pi}{M}, \tag{9.37}$$

where α is a positive parameter that specifies the width of the transition band of the prototype filter of the filter bank as explained below. The parameter α, as noted above, is known as the roll-off factor.

Substituting (9.37) in (9.36) we get

$$\omega_p = (1 - \alpha)\frac{\pi}{M}. \tag{9.38}$$

Also, the width of the transition band of the prototype filter is obtained as

$$\omega_s - \omega_p = \frac{2\pi\alpha}{M}. \tag{9.39}$$

Equation (9.34) explicitly states that the group-delay introduced by the filter bank is KM. In most subband adaptive filtering applications, we want to keep this delay as small as possible. On the other hand, the optimum K that results in maximum attenuation in the stop-band is obtained by choosing K so that KM is the nearest multiple of M to $N_a/2$. However, this delay is generally large and thus we would instead strike a compromise between delay and stop-band attenuation. That is, we may accept a lower delay at the cost of lower stop-band attenuation.

For effective implementation of subband adaptive filters, it is important to understand the effect of reduced delay on the performance of the analysis and synthesis filters and its overall impact on the performance of the adaptive filter. This can be best understood through an example.

Figure 9.11 shows the magnitude and group-delay responses of three filters that have been designed by the above method. The filter length, N_a, the number of subbands, M, and the roll-off factor, α, are set equal to 97, 4 and 0.25, respectively, and the three designs are differentiated by the parameter K. The separation of the pass-band, transition band and stop-band can be clearly seen in the responses. In particular, we note that the

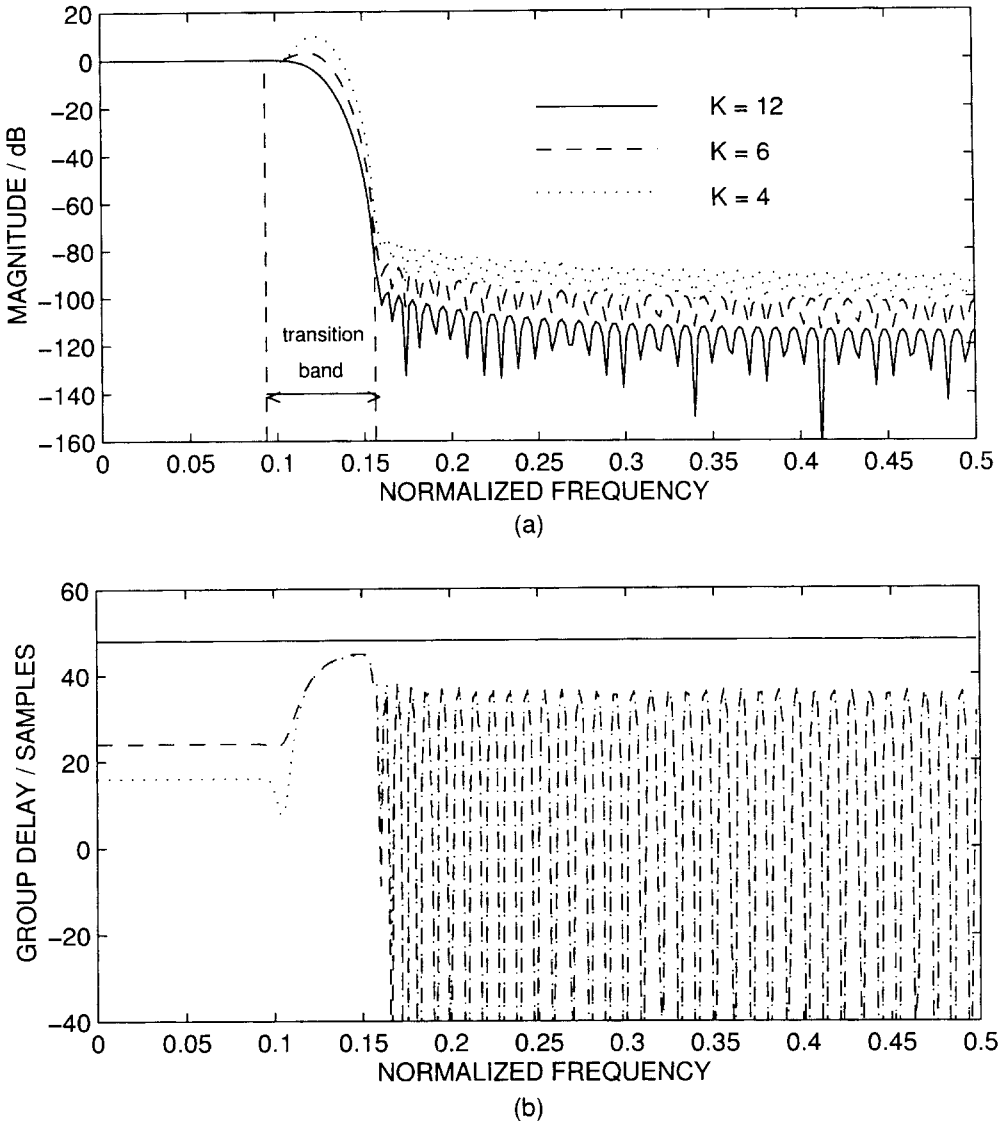


Figure 9.11 Magnitude and group-delay responses of three filters that have been designed by the method in (Farhang-Boroujeny and Wang, 1997). Reprinted from *Signal Processing*, vol. 61, B. Farhang-Boroujeny and Z. Wang, 'Adaptive filtering in subbands: Design issues and experimental results for acoustic echo cancellation', pp. 213–223, copyright (1997), with permission from Elsevier Science

transition bands in the three designs are the same and match the band edges predicted by (9.37) and (9.38). We also note that the price to be paid for achieving reduced delay is lower stop-band attenuation, an undesirable boost in the magnitude response in the transition band, and group-delay distortion in the transition and stop-bands. However,

the magnitude and group-delay responses in the pass-band remain nearly undistorted. This is a desirable feature of this design method which makes it very appropriate for designing analysis as well as synthesis filters in the application of subband adaptive filtering.

9.8 A Design Procedure for Subband Adaptive Filters

Since there are many compromises to be made in the overall design of subband adaptive filters, it is very hard to suggest a simple design procedure for such filters. In this section we present a procedure that the author has found useful in his research work. This procedure is iterative in nature and its application requires some experience. Hence, a novice needs to do some experiments with it before he can use it for actual design.

To choose all the parameters necessary for setting up a subband adaptive filter, we should take the following steps:

1. Choose a value for the number of subbands, M.
2. Choose an integer parameter, J, in the range of one-half to two-thirds of M, and select the pass-band, transition band and stop-band of the analysis and synthesis prototype filters as in Figure 9.12. This determines the values of the roll-off factors α_a and α_s of the analysis and synthesis filters, respectively.

   Note that the mid-points of the transition bands of the analysis and synthesis prototype filters are π/J and π/M, respectively. We note that when the method of Section 9.7 is used to design the analysis and synthesis filters, these choices of the mid-points of the transition bands lead to analysis filters that are J-band complementary, and synthesis filters that are M-band complementary. Furthermore, the positions of these mid-points determine the range of the roll-off factors α_a and α_s of the analysis and synthesis filters, respectively. The range of possible values that α_a and α_s may take can easily be worked out by inspection of Figure 9.12, and noting that the pass-band and transition band of the synthesis filter should be covered by the pass-band of the analysis filter (see also Problem P9.4).
3. Choose values for the lengths of analysis and synthesis filters. Call these N_a and N_s, respectively. Also, select values for the parameters K of the analysis and synthesis filters. Call these K_a and K_s, respectively.

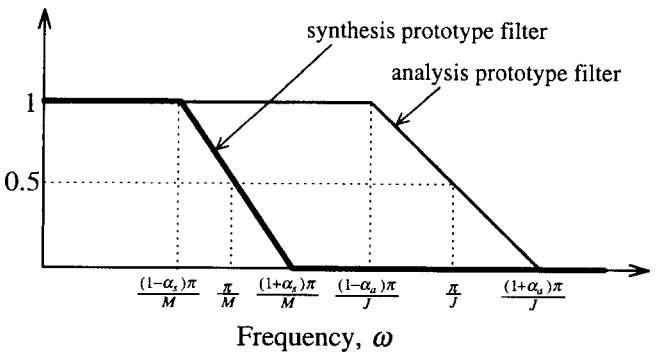


Figure 9.12 Definitions of the band edges in the analysis and synthesis prototype filters

4. Using the parameters selected above, design the analysis and synthesis prototype filters by following the method presented in Section 9.7.1.
5. Evaluate the stop-band rejection of the prototype filters. If satisfactory, proceed with the next step. Otherwise, reselect one or a few of the parameters N_a, N_s, K_a, K_s, J, α_a and α_s and redesign the prototype filters until the design is satisfactory.
6. Select a value of $L < J$ and evaluate aliasing of the decimated subband signals. A limited amount of aliasing may be allowed. However, because of the reason discussed in Section 9.6, such aliasing should be relatively small.
7. Evaluate the design by putting the designed analysis and synthesis filters in the subband adaptive filter structure and running a typical simulation of some application. If the performance is not satisfactory, then the filters need to be redesigned for other choices of the parameters listed above.

While putting the designed analysis and synthesis filters in a subband structure, we should note that the analysis filters are J-band complementary with the parameter $K = K_a$, while the synthesis filters are M-band complementary with the parameter $K = K_s$. As a result, the group-delay introduced by the analysis filters is K_aJ and that from the synthesis filters is K_sM. Then, the net group-delay due to a direct cascade of the two filter banks would be $K_aJ + K_sM$. On the other hand, according to our discussion in Section 9.2, the cascade of the analysis and synthesis filters must be M-band complementary. This suggests that the total group-delay must be an integer multiple of M. This may be achieved by either selecting K_a and J so that $K_aJ + K_sM$ is an integer multiple of M, or by padding the appropriate number of zero coefficients at the beginning of the analysis and/or synthesis filters so that the total group-delay is made an integer multiple of M. Accordingly, the following equation may be used to calculate the delay, Δ:

$$\Delta = \text{the first integer multiple of M which is greater than or equal to } K_aJ + K_sM. \tag{9.40}$$

9.9 An Example

In this section we discuss two design examples to demonstrate the effectiveness of the design technique that was introduced in the previous two sections.[7] The aim is to see how far we can go in reducing the delay and the price that we pay for it.
The following common parameters are used in both the designs:

$$M = 32, \quad J = 19, \quad \alpha_a = \tfrac{3}{16} \quad \text{and} \quad \alpha_s = \tfrac{7}{19}.$$

In the first design, we ignore the problem of delay and design the analysis and synthesis filters that result in maximum attenuation in their stop-bands. As was noted earlier, maximum stop-band attenuation is achieved when the delay introduced by each filter is about half of its respective length. We refer to this as the *conventional-delay design*. In the second design, a few attempts are made to obtain a pair of low-delay analysis and synthesis filters with stop-band attenuations comparable with that in the first design.

[7] The design examples presented here and their application in the study of an acoustic echo canceller is taken from Wang (1996).

Table 9.1 Summary of the two designs of analysis–synthesis prototype filters

Parameters	Conventional-delay	Low-delay
K_a	5	3
K_s	3	1
N_a	191	289
N_s	193	353
Δ	192	96
E_a	1.4×10^{-6}	5.0×10^{-7}
E_s	4.1×10^{-7}	4.6×10^{-7}

This is called the *low-delay design*. To achieve similar stop-band attenuations with reduced delay, the lengths of the filters need to be chosen longer than their conventional-delay counterparts.

The two designs are summarized in Table 9.1. The value of the delay, Δ, for each design is calculated according to (9.40). Note that the low-delay design achieves a delay

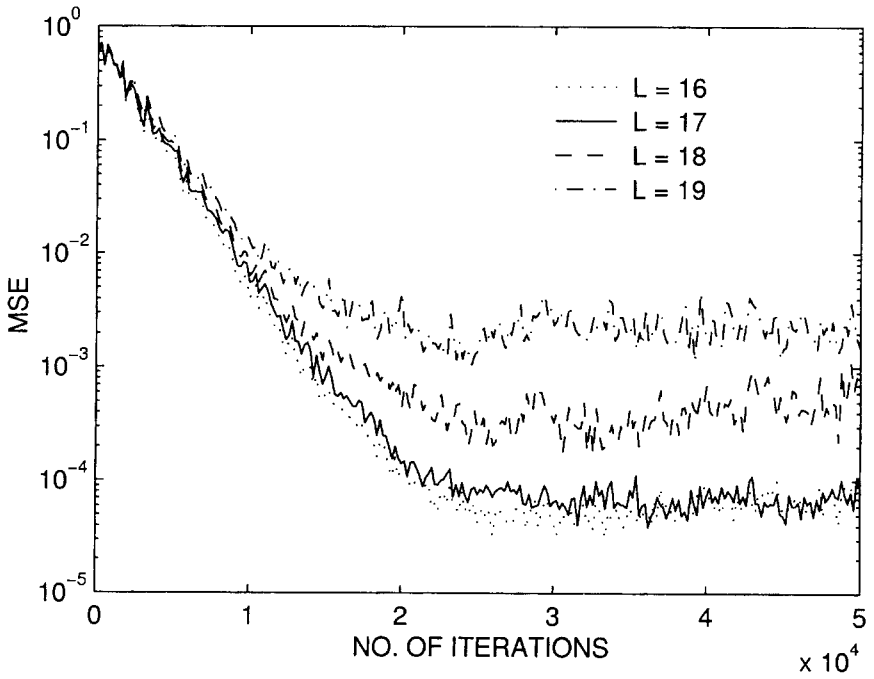


Figure 9.13 Learning curves of the subband adaptive filter for different values of the decimation factor, L. Reprinted from *Signal Processing*, vol. 61, B. Farhang-Boroujeny and Z. Wang, 'Adaptive filtering in subbands: Design issues and experimental results for acoustic echo cancellation', pp. 213–223, copyright (1997), with permission from Elsevier Science

which is half of that of the conventional-delay design. This, as expected, is at the cost of increased filter lengths, N_a and N_s. In Table 9.1, E_a and E_s are the stop-band energies of the analysis and synthesis filters, respectively.

To illustrate the effect of the decimation factor, L, on the overall performance of the filter, the designed low-delay analysis and synthesis filters are put into a subband structure that is used for modelling a 1600 taps plant. The plant response is that of an acoustic echo path of a normal size office room (see the next section). The plant input is assumed to be a white Gaussian noise. We also add some noise to the plant output. The normalized LMS algorithm of Section 6.6 is used for adaptation of the subband filters – see the next section. Figure 9.13 shows the learning curves of the subband adaptive filter for values of $L = 16$ to 19. $L = 16$ corresponds to the case where the decimated subband signals do not suffer from any aliasing. On the contrary, $L = 19$ corresponds to the case where the decimated subband signals are fully aliased over the transition bands of their respective analysis filters. However, the signals in their pass-bands do not suffer from any serious aliasing, except that due to non-ideal stop-band attenuations which are negligible. The case $L = 17$ does suffer from aliasing in the transition bands, though relatively low. These results clearly confirm our earlier conjecture that in the selection of the decimation factor, L, a small amount of aliasing is acceptable.

9.10 Application to Acoustic Echo Cancellation

In this section we present some results of subband adaptive filtering when applied to the application of acoustic echo cancellation (AEC). We recall from Chapter 1 (Section 1.6.4) that AEC is nothing but a system modelling problem. For our experiments, we use an echo path whose impulse response has been measured from an actual office room. At a sampling rate of 11.025 kHz, an adaptive filter with 1600 taps is found to be sufficient for modelling the room acoustics. The low-delay analysis and synthesis prototype filters presented in the previous section are used for implementing the subband adaptive filter. For purposes of comparison, we also present the results obtained with a 1600 taps full-band transversal filter.

We use the normalized LMS (NLMS) algorithm of Chapter 6 for the adaptation of the subband filters as well as the full-band filter. Since the signals as well as the filters' coefficients in the subband set-up are, in general, complex-valued, we use the complex form of the NLMS algorithm:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\tilde{\mu}}{\mathbf{x}^{\mathrm{H}}(n)\mathbf{x}(n) + \psi} e^{*}(n)\mathbf{x}(n), \tag{9.41}$$

where the superscript H denotes Hermitian transpose, the asterisk denotes complex conjugation, $\tilde{\mu}$ is the unnormalized step-size parameter and ψ is a positive constant added to prevent instability of the algorithm when $\mathbf{x}^{\mathrm{H}}(n)\mathbf{x}(n)$ is small. In the full-band case, where $\mathbf{x}(n)$ and $\mathbf{w}(n)$ are real-valued vectors, the real form of (9.41) is used, wherein the superscript H is replaced by T (transpose) and the conjugation is removed from $e(n)$. The step-size parameter $\tilde{\mu}$ is set to 0.4 in all the results. The parameter ψ is chosen equal to 1% of the average of $\mathbf{x}^{\mathrm{H}}(n)\mathbf{x}(n)$ over all subband signals, in the subband case, and equal to 1% of the average of $\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)$, in the full-band case.

The commonly used measure for evaluating the performance of acoustic echo cancellers is echo return loss enhancement (ERLE), which is defined as

$$\text{ERLE} = 10\log_{10}\left\{\frac{\text{E}\left[(d(n)-\nu(n))^2\right]}{\text{E}\left[(e(n)-\nu(n))^2\right]}\right\}, \tag{9.42}$$

where $d(n)$ is the signal at the microphone (this includes the echo from the speaker and other sound signals picked up by the microphone – see Figure 1.19), and $\nu(n)$ is the microphone signal minus the echo from the speaker. Therefore $d(n) - \nu(n)$ is the echo from the speaker that is picked up by the microphone and $e(n) - \nu(n)$ is the residual echo, thus the name ERLE.

To evaluate ERLE, a segment of speech signal is applied as input to the echo path and a white Gaussian noise sequence $\nu(n)$, which is at 35 dB below the speech signal level at the echo path output (microphone input), is added to produce the desired output sequence, $d(n)$. Figure 9.14 shows the results obtained with the subband adaptive echo canceller as well as its full-band counterpart. The superior performance of the subband adaptive filter is clearly observed. Both the full-band and subband echo cancellers exhibit fast initial convergence. However, at higher levels of ERLE the slower modes of

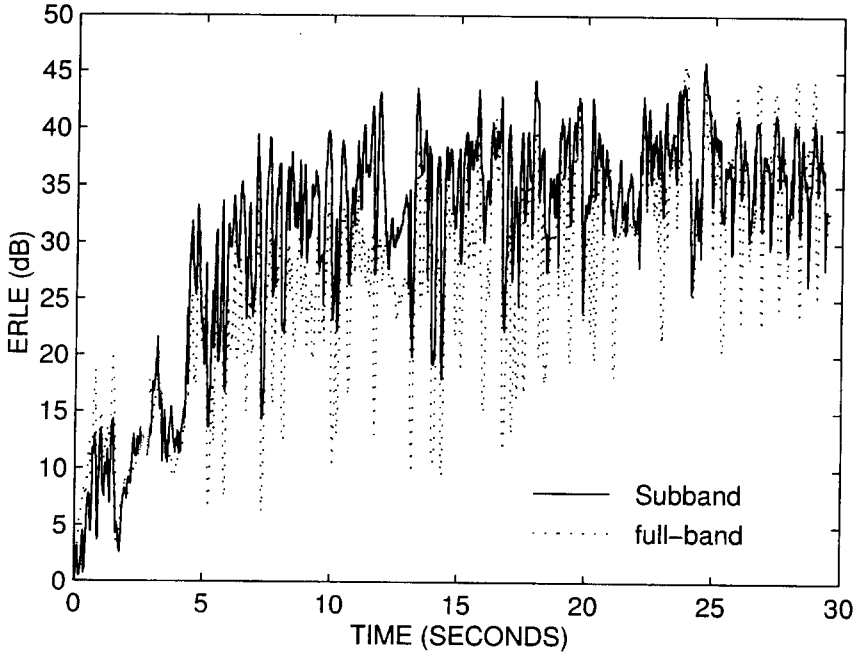


Figure 9.14 Echo return loss enhancement (ERLE) for the full-band and subband adaptive filters. Reprinted from *Signal Processing*, vol. 61, B. Farhang-Boroujeny and Z. Wang, 'Adaptive filtering in subbands: Design issues and experimental results for acoustic echo cancellation', pp. 213–223, copyright (1997), with permission from Elsevier Science

convergence of the full-band implementation become more prominent. The subband echo canceller reaches its steady state after about 7 s, while its full-band counterpart requires more than 20 s.

Further experiments with the subband adaptive filter using conventional-delay and low-delay analysis–synthesis filters have shown that no noticeable difference between the convergence behaviour of the two implementations could be observed (Wang, 1996).

9.11 Comparison with the FBLMS Algorithm

Adaptive filtering in subbands has many similarities with the fast block LMS (FBLMS) algorithm[8] which was introduced in the previous chapter. Firstly, both of these methods may be categorized under the class of block/parallel processing algorithms. As a result, both of these methods offer fast implementations of adaptive filters, i.e. implementations with reduced complexity as compared with the non-block methods, such as the conventional LMS algorithm. Furthermore, they resolve the problem of slow convergence of the LMS algorithm. These advantages are obtained at the cost of certain processing delay at the filter output, in both the methods. Hence, at this point it seems appropriate and essential to make some comments on the relative performance of the method of subband adaptive filtering and the FBLMS algorithm in terms of convergence behaviour, computational complexity, and processing delay. However, a quantitative comparison of the two methods is not straightforward. Thus, in the rest of this section we make an attempt to give some general comments on the above issues, leaving the discussion an open-ended one so that the reader can complete it by closely examining his/her specific application of interest.

We note that adaptation of each of the subband filters can be done using any of the adaptive filtering algorithms that have been introduced so far or will be introduced in the subsequent chapters. In the discussion that follows, for convenience we assume that the normalized LMS algorithm is used for this. We thus use the term 'subband NLMS algorithm' to refer to this implementation of the subband adaptive structure.

Simulations and experiments show that both the subband NLMS and FBLMS algorithms are quite successful in decorrelating the samples of the filter input. By careful selection of their parameters, both these algorithms can be tuned to offer learning curves that are predominantly governed by a single mode of convergence. For instance, in the application of acoustic echo cancellation that was discussed in the previous section, the presence of a predominant mode of convergence in the learning (ERLE) curve of the AEC can be clearly observed; see Figure 9.14. A similar performance can also be achieved by tuning the parameters of the FBLMS algorithm. Thus, in general, both the subband NLMS and FBLMS algorithms can offer very good and comparable convergence behaviour.

[8] In this section we use the term FBLMS algorithm in a general sense. It includes the FBLMS as well as partitioned FBLMS algorithms of the previous chapter.

A comparison of the two algorithms with respect to their computational complexity is also not straightforward. For a pair of designs with comparable convergence behaviour and processing delay, one may use the number of operations per sample for comparing the computational complexities of the two algorithms. This, although often used in the literature for comparing different algorithms, does not seem to be fair in the present case because of the many structural differences between the two algorithms. For instance, the subband NLMS algorithm has a more regular structure than the FBLMS algorithm. On the other hand, in typical applications of interest, say adaptive filters with at least a few hundred full-band taps, we usually find that the FBLMS algorithm has a lower operation count than the subband NLMS algorithm. Thus, to a great extent, the choice between the two algorithms depends on the available hardware/software platform. In software implementation on digital signal processors, the FBLMS algorithm is usually found to be more efficient than the subband NLMS algorithm. In contrast, the more regular structure of the subband NLMS algorithm may make it a better choice in a custom chip design. In particular, we may note that the subband filter structure can easily be divided into a number of separate blocks. For instance, each of the analysis/synthesis filter banks and the subband filters of Figure 9.7 may be treated as a separate block in a multi-processor chip.

Delay is an adjustable parameter in the subband structure as well as the FBLMS algorithm. In general, by allowing a larger delay (up to a certain limit), the complexities of both the methods can be reduced. The choice of the delay is usually limited by the system specification.

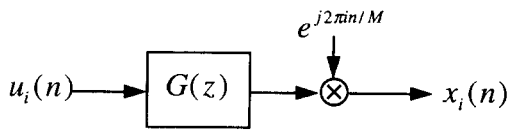
Problems

P9.1 Show that if a sequence $\bar{y}_i(k)$ is interpolated using an interpolation factor of L and passed through a filter with the impulse response g_n, the resulting output may be written as

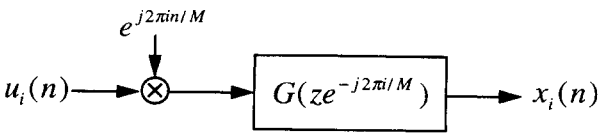
$$y_i(n) = \sum_{k=-\infty}^{\infty} \bar{y}_i(k) g_{n-kL}.$$

P9.2 Show that the following pairs of structures are equivalent:

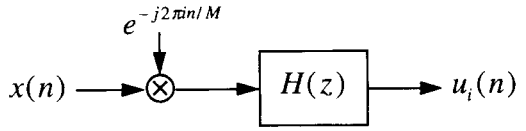
(i)



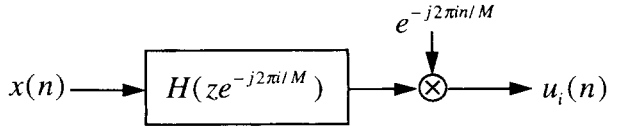
and



(ii)



and



P9.3 Consider a transversal adaptive filter with tap-input and tap-weight vectors $\mathbf{x}(n)$ and $\mathbf{w}(n)$, respectively. To adapt $\mathbf{w}(n)$ we wish to use the delayed LMS recursion

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n-\Delta)\mathbf{x}(n-\Delta),$$

where Δ is a constant delay, $e(n) = d(n) - \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n)$ is the output error, and $d(n)$ is the desired signal. Study the hardware/software implementation of this algorithm and discuss how the hardware/memory requirements of the filter vary with Δ.

Refer to the synthesis dependent structure given in Figure 9.8. Note that there is some delay introduced by the synthesis–analysis filters in the path from subband filters outputs, the $\bar{y}_i(k)$s, to the subband errors, the $\bar{e}_i(k)$s. Discuss how this delay leads to a set of delayed LMS recursions for adaptation of the subband filters, the $\overline{W}_i(z)$s, and how this affects the hardware/memory requirements of the subband structure.

P9.4 Consider an M-band subband structure with parameter J as defined in Section 9.8. Show that the condition necessary for the pass-bands and transition bands of synthesis filters to be covered by the pass-bands of analysis filters is the following:

$$J\alpha_{\mathrm{s}} + M\alpha_{\mathrm{a}} \leq M - J.$$

P9.5 Explore the validity of (9.23) in detail.

P9.6 Equations (9.21) and (9.22) are given for the subband adaptive filters with real-valued input. Derive similar equations for the case where the filter input and desired signal are complex-valued.

P9.7 In a pair of complementary analysis–synthesis filter banks, for each of the following set of parameters determine the number of zeros that need to be added in front of either the analysis or synthesis filters such that their combination is a complementary M-band filter bank:

(i) $M = 4$, $J = 3$, $K_{\mathrm{a}} = 7$, $K_{\mathrm{s}} = 5$.
(ii) $M = 64$, $J = 48$, $K_{\mathrm{a}} = 4$, $K_{\mathrm{s}} = 3$.

P9.8 Recall that in the realization of DFT analysis filters using the weighted overlap–add method, at the last stage we need to multiply the DFT outputs by the coefficients W_M^{-ikL}, for $k = 0, 1, \ldots, M-1$ (see (9.7)). On the other hand, in the realization of DFT

synthesis filters using the weighted overlap–add method, the subband signals should be multiplied by the coefficients W_M^{ikL}, for $k = 0, 1, \ldots, M-1$, prior to the application of the DFT (see (9.15)). Carefully examine the structure of the subband adaptive filter and show that these two operations may be deleted from the structure of the subband adaptive filter without affecting its performance.

Simulation-Oriented Problems

P9.9 The MATLAB program 'eigenfir.m' in the accompanying diskette can be used for designing complementary eigenfilters of the type discussed in Section 9.7.1. Use this program to design three filters with the following specifications:

(i) $N = 129$, $M = 4$, $\alpha = 0.25$, $K = 16$.
(ii) $N = 129$, $M = 4$, $\alpha = 0.25$, $K = 8$.
(iii) $N = 129$, $M = 4$, $\alpha = 0.25$, $K = 4$.

For each design, confirm that the band edges are realized as predicted in Section 9.7.2.

P9.10 Design a pair of analysis and synthesis prototype filters with the following parameters:

$$M = 16, \quad J = 10, \quad N_a = N_s = 257, \quad \alpha_a = \alpha_s = 0.15, \quad K_a = 3, \quad K_s = 4.$$

Put these into a subband structure and verify that the cascade of the analysis and synthesis filter banks is equivalent to a pure delay. For this, you may put a random sequence as input to the analysis filter bank and observe that the same sequence, with some delay, appears at the synthesis filter bank output. You may need to add an appropriate number of zeros at the beginning of the analysis or synthesis filter in order to get the right result from this experiment. Try your experiment for different values of the decimation factor $L = 7$, 8, 9 and 10. Do you observe any significant difference in the results? Explain your observation. Among these values of L, show that only $L = 7$ prevents aliasing of the subband signals.

P9.11 Use the analysis and synthesis filter banks of the previous problem to realize an NLMS-based subband adaptive filter to model a plant with 500 full-band taps. Choose a set of independent random numbers with variance 0.01 as the samples of the plant impulse response. Also, add a Gaussian noise with variance 10^{-4} to the plant output as the plant noise. Run your program for different values of the decimation factor, L, and verify that the subband adaptive filter converges towards an MSE which is much larger than the minimum MSE, when the aliasing of the subband signals is significant. To

10
IIR Adaptive Filters

In our study of adaptive filters in the previous chapters, we always limited ourselves to filters with a finite-impulse response (FIR). The main feature of FIR filters, which has made them the most attractive structure in the application of adaptive filters, is that they are non-recursive. That is, the filter output is computed based on only a finite number of input samples. This, as we noted in the previous chapters, results in a quadratic mean-square error (MSE) performance surface, allowing us to use any of the simple gradient-based algorithms for finding the optimum coefficients (tap weights) of the filter.

The use of *recursive* or infinite-impulse response (IIR) filters, on the other hand, has been less popular in the realization of adaptive filters for the following reasons:

1. IIR filters can easily become unstable since their poles may get shifted out of the unit circle (i.e. $|z| = 1$, in the z-plane) by the adaptation process.
2. The performance function (e.g. MSE as a function of filter coefficients) of an IIR filter, usually, has many local minima points.

The problem of instability is usually dealt with by checking the filter coefficients after each adaptation step and limiting them to the range that results in a stable transfer function. This, in general, is a difficult job and adds additional complexity which in many cases becomes significant when the filter order is large. This additional complexity tends to nullify the computational advantage provided by the recursive nature of these filters. Because of the multimodal nature of their performance surfaces, convergence of the IIR adaptive filters to their global minima is not guaranteed. The following approaches are usually used to deal with this problem:

1. Local minima are usually observed when the criterion used to adjust the filter coefficients is MSE. A modification to this criterion leads to quadratic performance surfaces similar to those of FIR filters, thereby eliminating the problem of local minima. This modification results in a special implementation of IIR adaptive filters known as the *equation error method.* The details of this method are discussed in Section 10.2. In contrast, the conventional formulation of IIR adaptive filters based on Wiener filter theory, which may suffer from the problem of local minima, is referred to as the *output error method.* This is discussed in Section 10.1.
2. For specific applications, we may limit ourselves to IIR transfer functions whose associated MSE performance surfaces are unimodal, i.e. they have no local minima.

In such cases the use of the output error method is the preferred choice since its convergence to the associated Wiener filter is guaranteed.

In this chapter we discuss both the output error and equation error methods. Since the performances of these methods are application dependent, we also present two case studies to highlight some of the implementation issues that should be considered when using IIR adaptive filters. The case studies that we have chosen are special applications that demonstrate the efficiency of IIR adaptive filters when their structure and/or design criteria are wisely selected and, at the same time, some peculiar behaviours of such filters which are hard to predict in general.

The first application that we consider is adaptive line enhancement. The problem of adaptive line enhancement was discussed in Chapter 1, where we reviewed various applications of adaptive filters, and also in Chapter 6, as an example of the application of the LMS algorithm. We used a transversal filter to implement the line enhancer. However, the problem of line enhancement may also be viewed as one of realizing/achieving narrow-band adaptive filters. But, to realize a narrow-band filter in transversal form, we would need very long filter lengths. In the example given in Chapter 6 (Section 6.4.3), we used a 30 tap transversal filter to achieve satisfactory enhancement of a single sinusoidal signal. On the contrary, as we will see later, a second-order IIR adaptive filter with four coefficients is sufficient for this problem. In Section 10.3, we introduce and study a special form of transfer function that has been found very appropriate for realization of IIR line enhancers. This is a good representative example of the second approach cited above, showing how a wise choice of the transfer function in a specific application can lead to a unimodal performance function, thereby solving the problem of local minima of the output error method.

The second application of IIR adaptive filters that we discuss is equalization of magnetic recording channels. In the case of magnetic recording channels, realization of equalizers in digital form turns out to be very costly because of very high data rates (a few hundred megabits per second). To solve this problem, the general trend in the present industry is to use analogue equalizers. We use the techniques presented in this chapter as tools to design analogue equalizers for magnetic recording channels. This serves as a good representative example of the use of the equation error method.

10.1 The Output Error Method

The output error method results when Wiener filter theory is made use of in a direct manner to develop algorithms for designing and/or adaptation of IIR filters. This can be best explained in the context of a system modelling problem as depicted in Figure 10.1. According to the Wiener theory, the coefficients of the recursive transfer function

$$W(z) = \frac{A(z)}{1 - B(z)}, \tag{10.1}$$

where $A(z)$ and $B(z)$ are polynomials in z, are obtained by minimizing the *output error*, $e(n)$, in the mean-square sense. We thus need to find the global minimum of the performance function $\xi = \mathrm{E}[e^2(n)]$ in an adaptive manner. However, we note that the performance function ξ is, in general, a multimodal function of the coefficients of

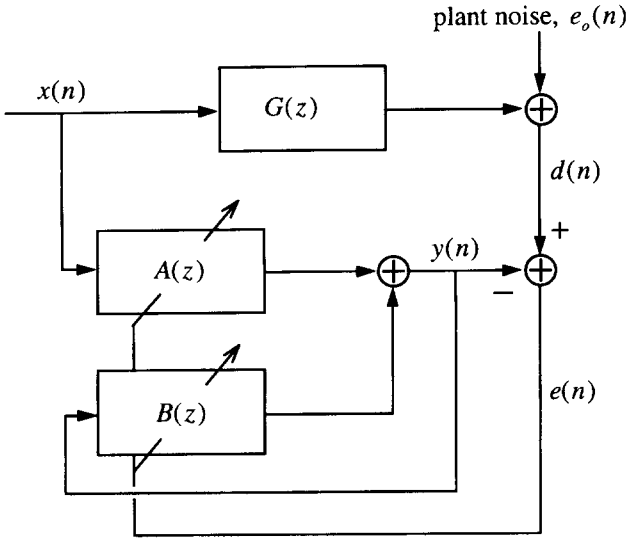


Figure 10.1 IIR adaptive filter with the output error adaptation method

the filter $W(z)$, i.e. ξ may have many local minima (see Chapter 3). This may lead to the convergence of any gradient-based (such as LMS) algorithm to a sub-optimal solution. In this section we ignore this problem and simply develop an LMS algorithm for adaptation of the coefficients of $W(z)$. Since the coefficients are obtained by minimizing the output error (in some sense), this approach is named the 'output error method'. The use of this name, hence, is to emphasize the special feature of the output error method as against the equation error method which is based on a different criterion (see next section).

Next, we develop a LMS algorithm for adaptation of the coefficients of IIR filters. To facilitate this we define the time-varying transfer functions

$$A(z,n) = \sum_{i=0}^{N} a_i(n) z^{-i} \tag{10.2}$$

and

$$B(z,n) = \sum_{i=1}^{M} b_i(n) z^{-i} \tag{10.3}$$

and note that the output, $y(n)$, of the adaptive IIR filter is obtained according to the equation

$$y(n) = \sum_{i=0}^{N} a_i(n) x(n-i) + \sum_{i=1}^{M} b_i(n) y(n-i). \tag{10.4}$$

The LMS algorithm, for the present case, can now be derived by following similar lines of derivations as those given in the previous chapters in the case of FIR filters. In

particular, we recall that the LMS algorithm makes use of the stochastic gradient vector given by

$$\hat{\nabla}(n) = \nabla_{\mathbf{w}} e^2(n) = 2e(n)\nabla_{\mathbf{w}} e(n), \tag{10.5}$$

where $\nabla_{\mathbf{w}}$ is the gradient operator with respect to the filter tap-weight vector, $\mathbf{w}(n)$, and

$$e(n) = d(n) - y(n) \tag{10.6}$$

is the output error. Here, the filter tap-weight vector $\mathbf{w}(n)$ is defined as

$$\mathbf{w}(n) = [a_0(n)\ \ a_1(n)\ \ \cdots\ \ a_N(n)\ \ b_1(n)\ \ \cdots\ \ b_M(n)]^{\mathrm{T}}. \tag{10.7}$$

Substituting (10.6) in (10.5) and noting that $d(n)$ is independent of $\mathbf{w}(n)$, we obtain

$$\begin{aligned}\hat{\nabla}(n) &= -2e(n)\nabla_{\mathbf{w}} y(n) \\ &= -2e(n)\left[\frac{\partial y(n)}{\partial a_0(n)}\ \ \frac{\partial y(n)}{\partial a_1(n)}\ \ \cdots\ \ \frac{\partial y(n)}{\partial a_N(n)}\ \ \frac{\partial y(n)}{\partial b_1(n)}\ \ \cdots\ \ \frac{\partial y(n)}{\partial b_M(n)}\right]^{\mathrm{T}}.\end{aligned} \tag{10.8}$$

The derivatives in (10.8) should be considered with special care, since $y(n)$ depends on its previous values, $y(n-1), y(n-2), \ldots$.

From (10.4) we get

$$\frac{\partial y(n)}{\partial a_i(n)} = x(n-i) + \sum_{l=1}^{M} b_l(n)\frac{\partial y(n-l)}{\partial a_i(n)}, \quad \text{for } i = 0, 1, \ldots, N \tag{10.9}$$

and

$$\frac{\partial y(n)}{\partial b_i(n)} = y(n-i) + \sum_{l=1}^{M} b_l(n)\frac{\partial y(n-l)}{\partial b_i(n)}, \quad \text{for } i = 1, 2, \ldots, M. \tag{10.10}$$

To proceed, it is convenient to define

$$\alpha_i(n) = \frac{\partial y(n)}{\partial a_i(n)}, \quad \text{for } i = 0, 1, \ldots, N \tag{10.11}$$

and

$$\beta_i(n) = \frac{\partial y(n)}{\partial b_i(n)}, \quad \text{for } i = 1, 2, \ldots, M. \tag{10.12}$$

Assuming that the $a_i(n)$ and $b_i(n)$ coefficients vary slowly in time, we get

$$\frac{\partial y(n-l)}{\partial a_i(n)} \approx \frac{\partial y(n-l)}{\partial a_i(n-l)} = \alpha_i(n-l) \tag{10.13}$$

and

$$\frac{\partial y(n-l)}{\partial b_i(n)} \approx \frac{\partial y(n-l)}{\partial b_i(n-l)} = \beta_i(n-l) \tag{10.14}$$

for $l = 1, 2, \ldots, M$. Substituting (10.13) and (10.14) in (10.9) and (10.10), respectively, we get the following recursive equations for obtaining successive samples of the $\alpha_i(n)$s and $\beta_i(n)$s:

$$\alpha_i(n) = x(n-i) + \sum_{l=1}^{M} b_l(n)\alpha_i(n-l) \tag{10.15}$$

and

$$\beta_i(n) = y(n-i) + \sum_{l=1}^{M} b_l(n)\beta_i(n-l). \tag{10.16}$$

Using these results, the LMS recursion for adaptation of IIR filters may be summarized as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\boldsymbol{\eta}(n), \tag{10.17}$$

where

$$\boldsymbol{\eta}(n) = [\alpha_0(n) \;\; \alpha_1(n) \;\; \cdots \;\; \alpha_N(n) \;\; \beta_1(n) \;\; \cdots \;\; \beta_M(n)]^{\mathrm{T}} \tag{10.18}$$

and the $\alpha_i(n)$s and $\beta_i(n)$s are obtained recursively according to (10.15) and (10.16).

Figure 10.2 depicts a block diagram showing the computations involved in calculating the $\alpha_i(n)$s and $\beta_i(n)$s, according to (10.15) and (10.16). From this diagram we see that the computation of the elements of $\boldsymbol{\eta}(n)$ requires parallel implementation of $M + N + 1$ recursive filters with the same transfer function $1/(1 - B(z, n))$, but different inputs, one for each element.

Figure 10.2 can be greatly simplified if we assume that the transfer function $1/(1 - B(z, n))$ varies only slowly with time. Then, we may use the following approximations:

$$\frac{1}{1 - B(z,n)} \approx \frac{1}{1 - B(z, n-i)}, \quad \text{for } i = 1, 2, \ldots, \max(N, M-1), \tag{10.19}$$

where $\max(N, M-1)$ denotes the maximum of N and $M - 1$. This allows us to write, from (10.15),

$$\alpha_i(n) \approx x(n-i) + \sum_{l=1}^{M} b_l(n-i)\alpha_i(n-l). \tag{10.20}$$

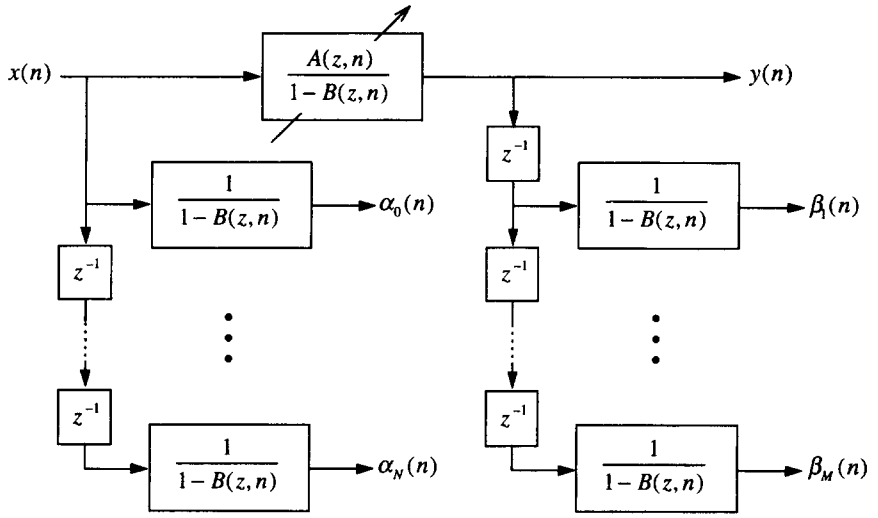


Figure 10.2 Implementation of the IIR adaptive filter using the output error adaptation method

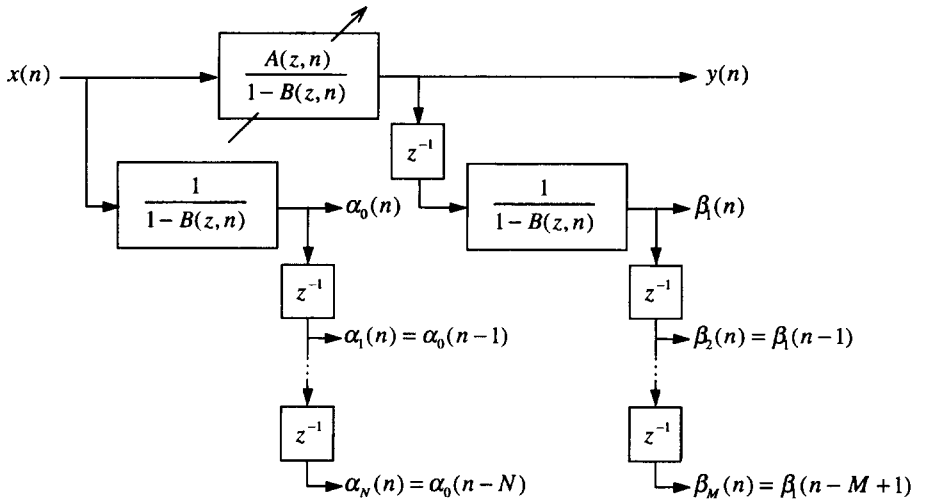


Figure 10.3 Simplified implementation of the IIR adaptive filter using the output error adaptation method

On the other hand, substituting i by 0 and n by $n - i$ in (10.15), we get

$$\alpha_0(n-i) = x(n-i) + \sum_{l=1}^{M} b_l(n-i)\alpha_0(n-i-l). \tag{10.21}$$

Now, comparing (10.21) and (10.20), we note that $\alpha_i(n)$ and $\alpha_0(n-i)$ are generated on the basis of the same input, $x(n-i)$, and approximately the same recursive equations. Thus, we get

$$\alpha_i(n) \approx \alpha_0(n-i), \quad \text{for } i = 1, 2, \ldots, N. \tag{10.22}$$

Similarly, we obtain

$$\beta_i(n) \approx \beta_1(n-i+1), \quad \text{for } i = 2, 3, \ldots, M. \tag{10.23}$$

Using these results, we obtain Figure 10.3 as an approximation to Figure 10.2. Note that in Figure 10.3, as opposed to Figure 10.2, we only need to use two filters with the transfer function $1/(1 - B(z, n))$ to calculate $\alpha_0(n)$ and $\beta_1(n)$. The rest of values of the $\alpha_i(n)$s and $\beta_i(n)$s are simply delayed versions of $\alpha_0(n)$ and $\beta_1(n)$, respectively. Table 10.1 summarizes the LMS algorithm which follows Figure 10.3.

Table 10.1 Summary of the output error LMS algorithm

Input: Tap-weight vector,
$\mathbf{w}(n) = [a_0(n)\ \ a_1(n)\ \ \cdots\ \ a_N(n)\ \ b_1(n)\ \ \cdots\ \ b_M(n)]^{\mathrm{T}}$,
Input vector,
$\mathbf{u}(n) = [x(n)\ \ x(n-1)\ \ \cdots\ \ x(n-N)\ \ y(n-1)\ \ \cdots\ \ y(n-M)]^{\mathrm{T}}$,
the previous samples of $\alpha_0(n)$ and $\beta_1(n)$,
and desired output, $d(n)$.

Output: Filter output, $y(n)$,
Tap-weight vector update, $\mathbf{w}(n+1)$,
and the samples of $\alpha_0(n)$ and $\beta_1(n)$ for next iteration.

1. Filtering:

$$y(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{u}(n)$$

2. Error estimation:

$$e(n) = d(n) - y(n)$$

3. $\boldsymbol{\eta}(n)$ update:

$$\alpha_0(n) = x(n) + \textstyle\sum_{l=1}^{M} b_l(n)\alpha_0(n-l)$$
$$\beta_1(n) = y(n) + \textstyle\sum_{l=1}^{M} b_l(n)\beta_1(n-l)$$
$$\boldsymbol{\eta}(n) = [\alpha_0(n)\ \ \alpha_0(n-1)\ \ \cdots\ \ \alpha_0(n-N)\ \ \beta_1(n)\ \ \cdots\ \ \beta_1(n-M)]^{\mathrm{T}}$$

4. Tap-weight vector adaptation:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\boldsymbol{\eta}(n)$$

10.2 The Equation Error Method

As was mentioned earlier, the main problem with direct minimization of the output error of an IIR filter is that the associated performance surface may have many local minima points, thereby resulting in convergence of the LMS algorithm to one of these local minima which may not be the desired global minimum. This problem may be resolved by using the equation error method as explained next.

Figure 10.4 depicts a block diagram illustrating the principle behind the equation error method. Here, the error used to adapt the transfer functions $A(z)$ and $B(z)$ is

$$e'(n) = d(n) - y'(n), \tag{10.24}$$

where

$$y'(n) = \sum_{i=0}^{N} a_i(n)x(n-i) + \sum_{i=1}^{M} b_i(n)d(n-i). \tag{10.25}$$

This equation may be thought of as a modified version of equation (10.4). It is obtained by replacing the past samples of output, $y(n-1), y(n-2), \ldots$, in (10.4), by the past samples of the desired output, $d(n-1), d(n-2), \ldots$. The name 'equation error' refers to this difference in the equation used to calculate the error $e'(n)$, as against the exact value of the output error, $e(n)$.

The adoption of the equation error method is based on the following rationale. When the structure and order of an adaptive filter are correctly selected, we would expect $d(n) \approx y(n)$ upon adaptation of the filter. In that case, the difference between the error sequences $e(n)$ and $e'(n)$, when both have converged towards their optimum values, is expected to be small. Hence, we may expect the performance surfaces associated with the

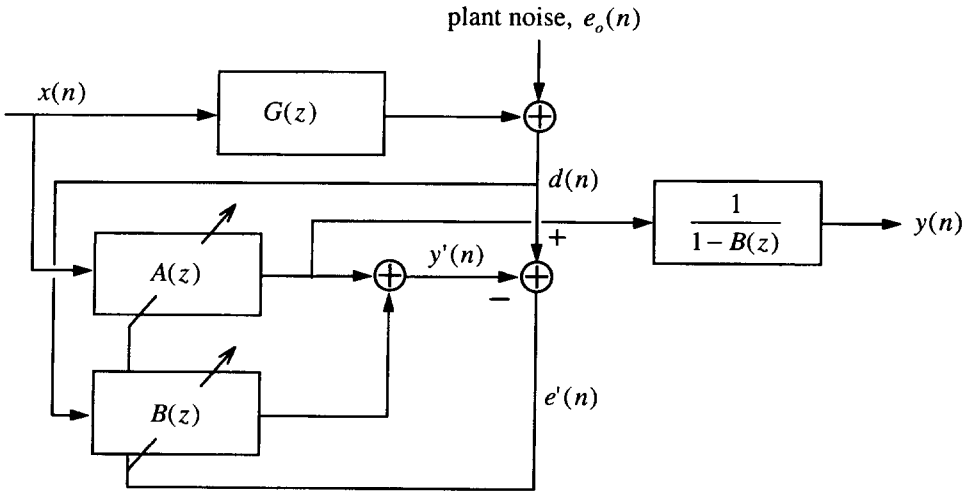


Figure 10.4 The IIR adaptive filter using the equation error method

output error and equation error methods to have approximately the same global minimum points. The use of the equation error method is then preferred, since its associated performance surface will be unimodal, i.e. will not have any local minimum. This unimodality results from the fact that the output of the filter, $y'(n)$, in (10.25) is no more recursive in nature, i.e. $y'(n)$ is effectively the output of a linear combiner with the tap-weight vector $\mathbf{w}(n)$, as defined by (10.7), and the tap-input vector

$$\mathbf{u}'(n) = [x(n)\ \ x(n-1)\ \ \cdots\ \ x(n-N)\ \ d(n-1)\ \ \cdots\ \ d(n-M)]^{\mathrm{T}}. \tag{10.26}$$

Now, we present a study of the equation error method that reveals its relationship as well as its difference with the output error method in greater detail. For this study we note that in the equation error method the criterion used for adaptation of the transfer functions $A(z)$ and $B(z)$ is

$$\xi' = \mathrm{E}[e'^2(n)]. \tag{10.27}$$

It is straightforward to show that this is a quadratic function of the coefficients of $A(z)$ and $B(z)$. This readily follows from the fact that the output $y'(n)$, as mentioned above, is the output of a linear combiner derived by the input sequences $x(n)$ and $d(n)$. Hence, convergence of the LMS or any other gradient based algorithm, which may be used to find the optimum tap weights of $A(z)$ and $B(z)$, is guaranteed.

We obtain a better understanding of the equation error method by finding the relationship between the output and equation errors, $e(n)$ and $e'(n)$, respectively. This relationship can be easily arrived at using the z-transform approach. From Figure 10.1, we note that

$$D(z) = X(z)G(z) + E_{\mathrm{o}}(z) \tag{10.28}$$

and

$$Y(z) = \frac{X(z)A(z)}{1 - B(z)}, \tag{10.29}$$

where $D(z)$, $X(z)$, $E_{\mathrm{o}}(z)$ and $Y(z)$ are the z-transforms of the sequences $d(n)$, $x(n)$, $e_{\mathrm{o}}(n)$ and $y(n)$, respectively. Then, since $e(n) = d(n) - y(n)$, we obtain from (10.28) and (10.29)

$$\begin{aligned} E(z) &= D(z) - Y(z) \\ &= X(z)\left(G(z) - \frac{A(z)}{1 - B(z)}\right) + E_{\mathrm{o}}(z), \end{aligned} \tag{10.30}$$

where $E(z)$ is the z-transform of the sequence $e(n)$. On the other hand, from Figure 10.4, we note that $e'(n) = d(n) - y'(n)$, with $y'(n)$ as given in (10.25). Hence, we get

$$E'(z) = D(z) - D(z)B(z) - X(z)A(z), \tag{10.31}$$

where $E'(z)$ is the z-transform of the sequence $e'(n)$. Substituting (10.28) in (10.31) and rearranging, we get

$$\begin{aligned} E'(z) &= X(z)[(1 - B(z))G(z) - A(z)] + [1 - B(z)]E_o(z) \\ &= \left[X(z)\left(G(z) - \frac{A(z)}{1 - B(z)}\right) + E_o(z)\right][1 - B(z)]. \end{aligned} \tag{10.32}$$

Finally, comparing (10.30) and (10.32), we obtain

$$E'(z) = E(z)(1 - B(z)). \tag{10.33}$$

This result shows that the equation error, $e'(n)$, is related to the output error, $e(n)$, through the transfer function $1 - B(z)$. In general, minimization of the mean-square values of the output and equation errors, $e(n)$ and $e'(n)$, respectively, could lead to two different sets of tap weights for the IIR filter. However, the two solutions may be very close for certain cases. For instance, when $\xi' = \mathrm{E}[e'^2(n)]$ converges to a very small value and $1 - B(z)$ is not very small for all values of z on the unit circle, $\xi = \mathrm{E}[e^2(n)]$ would also be very small; thus, we expect both the output and equation error methods to converge to about the same solutions. On the other hand, when the minimum value of ξ' is large or $1 - B(z)$ is very small over a range of frequencies, the two solutions may be significantly different. The following example clarifies this concept further.

Example 10.1

Consider Figures 10.1 and 10.4. Let the plant $G(z)$ be given by

$$G(z) = \frac{1}{1 - 0.5z^{-1}}$$

and choose the modelling filter as

$$W(z) = \frac{a_0}{1 - b_1 z^{-1}}.$$

Clearly, when the plant noise $e_o(n)$ is uncorrelated with the input, $x(n)$, the minimum MSE (Wiener) solution to this problem, i.e. what we obtain by using the output error method (assuming that the global minimum of the corresponding mean-square error function can be found), is $a_{0,o} = 1$ and $b_{1,o} = 0.5$. Here, the subscript 'o' emphasizes that the coefficients are those of the optimum Wiener filter. The minimum MSE in this case is

$$\xi_{\min} = \sigma_o^2,$$

where $\sigma_o^2 = \mathrm{E}[e_o^2(n)]$.

To find the optimum values of a_0 and b_1 in the case of the equation error method, we note that the filter tap-input and tap-weight vectors are, respectively, $\mathbf{u}'(n) = [x(n)\ \ d(n-1)]^{\mathrm{T}}$ and $\mathbf{w} = [a_0\ \ b_1]^{\mathrm{T}}$ and the desired signal is $d(n)$. The optimum value of $\mathbf{w}$ is then obtained by solving the normal equation

$$\mathbf{R}\mathbf{w} = \mathbf{p}, \tag{10.34}$$

where

$$\mathbf{R} = \mathrm{E}[\mathbf{u}'(n)\mathbf{u}'^{\mathrm{T}}(n)] = \begin{bmatrix} \mathrm{E}[x^2(n)] & \mathrm{E}[x(n)d(n-1)] \\ \mathrm{E}[d(n-1)x(n)] & \mathrm{E}[d^2(n-1)] \end{bmatrix}$$

and

$$\mathbf{p} = \begin{bmatrix} \mathrm{E}[d(n)x(n)] \\ \mathrm{E}[d(n)d(n-1)] \end{bmatrix}.$$

To facilitate evaluation of $\mathbf{R}$ and $\mathbf{p}$ and the subsequent calculations in this example, we assume that the input, $x(n)$, is white and has a variance of unity. This implies that the power spectral density of $x(n)$ is equal to one for all frequencies, i.e. $\Phi_{xx}(z) = 1$. Also, $\mathrm{E}[x^2(n)] = 1$. The rest of elements of the correlation matrix $\mathbf{R}$ and the cross-correlation vector $\mathbf{p}$ are obtained by using the results of Chapter 2. For example,

$$\begin{aligned} \mathrm{E}[d(n)x(n)] = \phi_{dx}(0) &= \frac{1}{2\pi j}\oint \Phi_{xx}(z)G(z)\frac{\mathrm{d}z}{z} \\ &= \frac{1}{2\pi j}\oint \frac{1}{1-0.5z^{-1}}\frac{\mathrm{d}z}{z} \\ &= \frac{1}{2\pi j}\oint \frac{\mathrm{d}z}{z-0.5} \\ &= \text{residue of } \frac{1}{z-0.5} \text{ at } z = 0.5 \\ &= 1. \end{aligned}$$

In similar way, we also obtain

$$\mathrm{E}[d^2(n)] = \tfrac{4}{3} + \sigma_{\mathrm{o}}^2, \quad \mathrm{E}[x(n)d(n-1)] = \mathrm{E}[d(n-1)x(n)] = 0$$

and

$$\mathrm{E}[d(n)d(n-1)] = \tfrac{2}{3}.$$

Substituting these results in (10.34) and solving for $\mathbf{w}$, we obtain

$$a_{0,\mathrm{e}} = 1 \quad \text{and} \quad b_{1,\mathrm{e}} = \frac{1}{2}\cdot\frac{1}{1+3\sigma_{\mathrm{o}}^2/4},$$

where the subscript 'e' signifies that the solutions correspond to the equation error method. We note that, in this particular case, $a_{0,\mathrm{e}}$ is unbiased, i.e. it is equal to its optimum value. However, $b_{1,\mathrm{e}}$ is different from its optimum value in the Wiener filter. The amount of bias in $b_{1,\mathrm{e}}$ is

$$b_{1,\mathrm{e}} - b_{1,\mathrm{o}} = -\frac{1}{2}\cdot\frac{3\sigma_{\mathrm{o}}^2/4}{1+3\sigma_{\mathrm{o}}^2/4}.$$

This bias is negligible when σ_{o}^2 is small. However, it becomes significant as σ_{o}^2 increases.

Further study of this example shows that when $x(n)$ is coloured (non-white), both $a_{0,\mathrm{e}}$ and $b_{1,\mathrm{e}}$ are biased and the amount of bias, as we expect, increases with σ_{o}^2. This is left as an exercise for the reader (see Problem P10.2).

10.3 Case Study I: IIR Adaptive Line Enhancement

As was noted earlier in this chapter, adaptive line enhancement is a special problem that can be best solved by using IIR filters. In this section we consider a special second-order IIR transfer function that was first proposed by David, et al. (1983) and subsequently used and developed further by the same authors and others (Ahmed et al., 1984; Hush et al., 1986; Cupo and Gitlin, 1989; Regalia, 1991; Cho and Lee, 1993; and Farhang-Boroujeny, 1997a).

Figure 10.5 depicts the block diagram of the adaptive line enhancer (ALE) that we wish to study in this section. Here, $W(z)$ is an IIR filter with the transfer function

$$W(z) = \frac{(1-s)(w-z^{-1})}{1-(1+s)wz^{-1}+sz^{-2}}. \tag{10.35}$$

This is a narrow-band filter that may be used to extract a portion of the spectrum of the input, $x(n)$. When $x(n)$ is the sum of a narrow-band and a wide-band processes and $W(z)$ is centred around the narrow-band part of $x(n)$, the output of $W(z)$ will contain mainly the narrow-band part of $x(n)$. The term line enhancer therefore refers to the fact that the narrow-band part of $x(n)$, which may be considered as a spectral line, is enhanced in the sense that it is separated from the wide-band part of $x(n)$ which may be thought of as a noise. In what follows we look into the details of the IIR ALE.

10.3.1 IIR ALE filter, W(z)

As was noted above, the transfer function $W(z)$ of (10.35) is that of a narrow-band filter. Its bandwidth is controlled by the parameter s, which may select any value in the range 0 to 1. Filters with a really narrow bandwidth can be realized by choosing values of s very close to one. The parameter w is related to the centre frequency, $w = \theta$, of the passband of $W(z)$ according to the equation

$$w = \cos\theta. \tag{10.36}$$

Substituting (10.36) in (10.35) and evaluating $W(z)$ at $z = e^{j\theta}$, i.e. the frequency response of $W(z)$ at the centre of its passband, we obtain

$$W(e^{j\theta}) = e^{j\theta}. \tag{10.37}$$

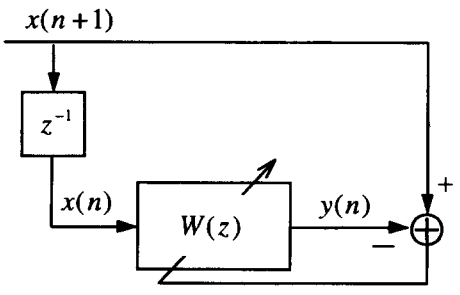


Figure 10.5 Adaptive line enhancer

This shows that at $z = e^{j \cos^{-1} w}$, $W(z) = z$ or, equivalently,

$$\text{at frequency } \omega = \cos^{-1} w, \quad z^{-1} W(z) = 1. \tag{10.38}$$

Noting that $z^{-1}W(z)$ is the transfer function between the input, $x(n)$, and the output, $y(n)$, of the line enhancer, the above result implies that the gain of the line enhancer to a sinusoid at frequency $\omega = \cos^{-1} w$ is exactly equal to one. This interesting property of the IIR line enhancer of (10.35) becomes advantageous in applications of notch filtering and also when multiple stages of line enhancers are cascaded together to enhance multiple sinusoids (spectral lines). Application of the line enhancer structure of Figure 10.5 as a notch filter is obvious if we note that the transfer function between the input, $x(n)$, and the error, $e(n)$, is $1 - z^{-1}W(z)$, and according to (10.38) this has a null at $\omega = \cos^{-1} w$.

10.3.2 Performance functions

To simplify our discussion, we assume that the input signal to the ALE is

$$x(n) = a \sin(\theta_o n) + \nu(n), \tag{10.39}$$

where a and θ_o are constants and $\nu(n)$ is a zero-mean white noise process with variance σ_ν^2. We refer to the first term in (10.39) as the (desired) signal and $\nu(n)$ as the noise. When $x(n)$ is given by (10.39), the performance function $\xi_w(s, w) = E[e^2(n)]$ of the IIR ALE is given by the following equation (see Problem P10.3):

$$\xi_w(s, w) = \frac{a^2}{2} \left| 1 - e^{-j\theta_o} W(e^{j\theta_o}) \right|^2 + \frac{2\sigma_\nu^2}{1 + s}. \tag{10.40}$$

The subscript w in $\xi_w(s, w)$ signifies the fact that, as we will see shortly, this is the performance function that is used to adjust w. In contrast, we define another performance function, $\xi_s(s, w)$, later, which will be used for adapting the parameter s. Figure 10.6 shows a set of plots of $\xi_w(s, w)$ as a function of w when s is given different values. These plots correspond to the case where $\theta_o = \pi/3$, $a = \sqrt{2}$ and $\sigma_\nu^2 = 1$. Observe from these plots that the performance function $\xi_w(s, w)$ is a unimodal function of w. Its minimum corresponds to $w = \cos\theta_o$, irrespective of the value of s. This can be easily proved analytically and is left as an exercise for the reader. This observation suggests that if s is kept fixed, then the optimum value of w can be obtained by using a gradient search method, such as the LMS algorithm. Furthermore, note also from Figure 10.6 that if w is set to its optimum value, then the minimum value of $\xi_w(s, w)$ reduces as s approaches one. This clearly improves the performance of the ALE. On the other hand, when w is not close to its optimum value, increasing the value of s results in slowing down the convergence of w since the gradient of $\xi_w(s, w)$ is quite small when w is away from its optimum value and s is close to one. To solve this problem, the parameter s may initially be given a smaller value and after or close to the convergence of w, it is changed to a larger value (Cho and Lee, 1993). To automate this, we need to find another performance function that allows us to quantify or detect the closeness of w to its optimum value. A possible performance function that may be used for this

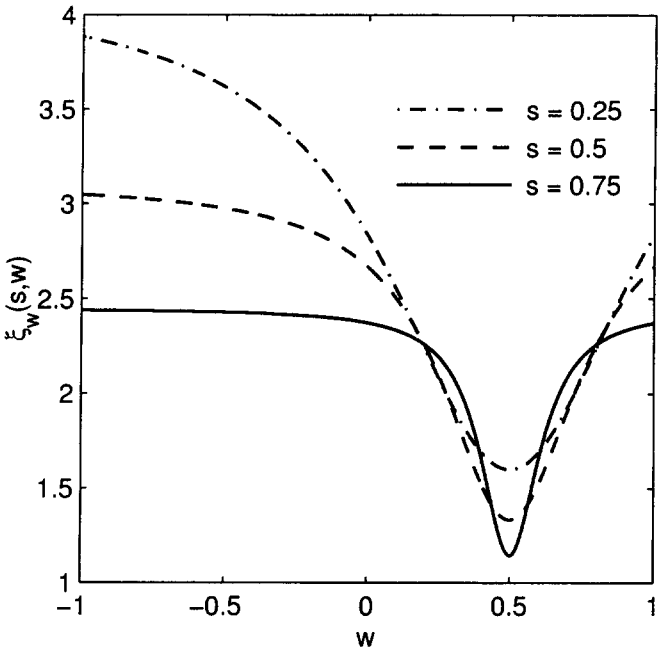


Figure 10.6 Plots of the performance function $\xi_w(s, w)$ for different values of s. Reprinted from Farhang-Boroujeny (1997a)

purpose is[1]

$$\xi_s(s, w) = \mathrm{E}[y_s^2(n)], \tag{10.41}$$

where

$$y_s(n) = \sqrt{\frac{1+s}{1-s}}\, y(n). \tag{10.42}$$

The adaptation of the parameter s is done by maximizing $\xi_s(s, w)$ with respect to s. It is straightforward to show that

$$\xi_s(s, w) = \frac{a^2}{2} \cdot \frac{1+s}{1-s} \cdot |W(\mathrm{e}^{j\theta_o})|^2 + \sigma_\nu^2. \tag{10.43}$$

Figure 10.7 shows the plots of $\xi_s(s, w)$, as a function of w, for $\theta_o = \pi/3$, $a = \sqrt{2}$, $\sigma_\nu^2 = 1$, and $s = 0.5, 0.7$ and 0.8. These plots clearly show that the performance function $\xi_s(s, w)$ is a proper choice for adjusting s. It perfectly satisfies the requirements stated above for changing s, namely, the maximization of $\xi_s(s, w)$ reduces s when w is far from its optimum value, and increases s as w approaches its optimum value. This can also be shown by observing the sign of $\partial\xi_s(s, w)/\partial s$ as w varies.

[1] The performance function $\xi_s(s, w)$ was first proposed by Farhang-Boroujeny (1997a).

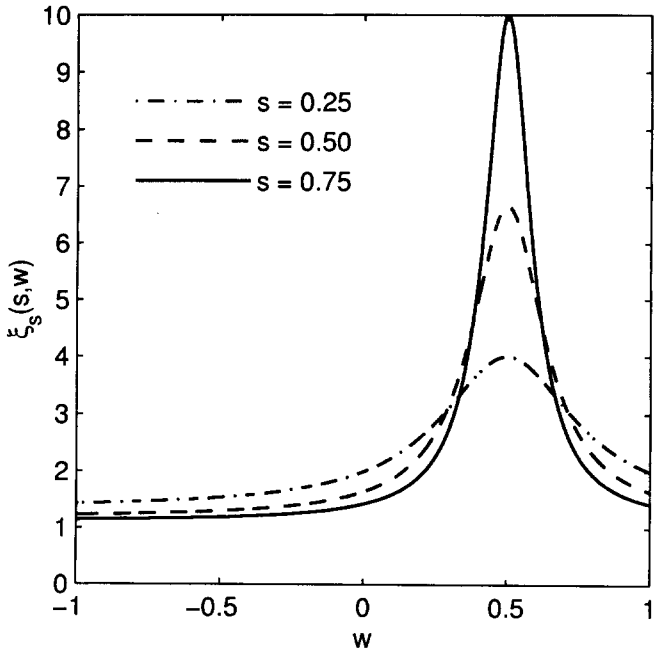


Figure 10.7 Plots of the performance function $\xi_s(s, w)$ for different values of s. Reprinted from Farhang-Boroujeny (1997a)

10.3.3 Simultaneous adaptation of s and w

Following similar derivations to those given in Section 10.1, we obtain the algorithm presented in Table 10.2, for simultaneous adaptation of s and w. We refer to this as Algorithm 1, for future reference. As in Table 10.2, henceforth we will use the notation $s(n)$ and $w(n)$ for s and w, respectively, since they vary with time because of adaptation. The derivations of the first four steps in Table 10.2 are straightforward. To derive the last

Table 10.2 Summary of the adaptive IIR ALE (Algorithm 1)

$$y(n) = (1 + s(n))w(n)y(n-1) - s(n)y(n-2) + (1 - s(n))(w(n)x(n) - x(n-1))$$
$$e(n) = x(n+1) - y(n)$$
$$\alpha(n) = (1 + s(n))w(n)\alpha(n-1) - s(n)\alpha(n-2) + (1 + s(n))y(n-1) + (1 - s(n))x(n)$$
$$w(n+1) = w(n) + 2\mu_w e(n)\alpha(n)$$
$$\beta(n) = (1 + s(n))w(n)\beta(n-1) - s(n)\beta(n-2) - (w(n)e(n-1) - e(n-2))$$
$$s(n+1) = s(n) + 2\mu_s \left[\frac{1}{(1 - s(n))^2} y^2(n) + \frac{1 + s(n)}{1 - s(n)} y(n)\beta(n) \right]$$

Definitions: $\alpha(n) = \dfrac{\partial y(n)}{\partial w(n)}$, $\beta(n) = \dfrac{\partial y(n)}{\partial s(n)}$

μ_w and μ_s are step-size parameters.

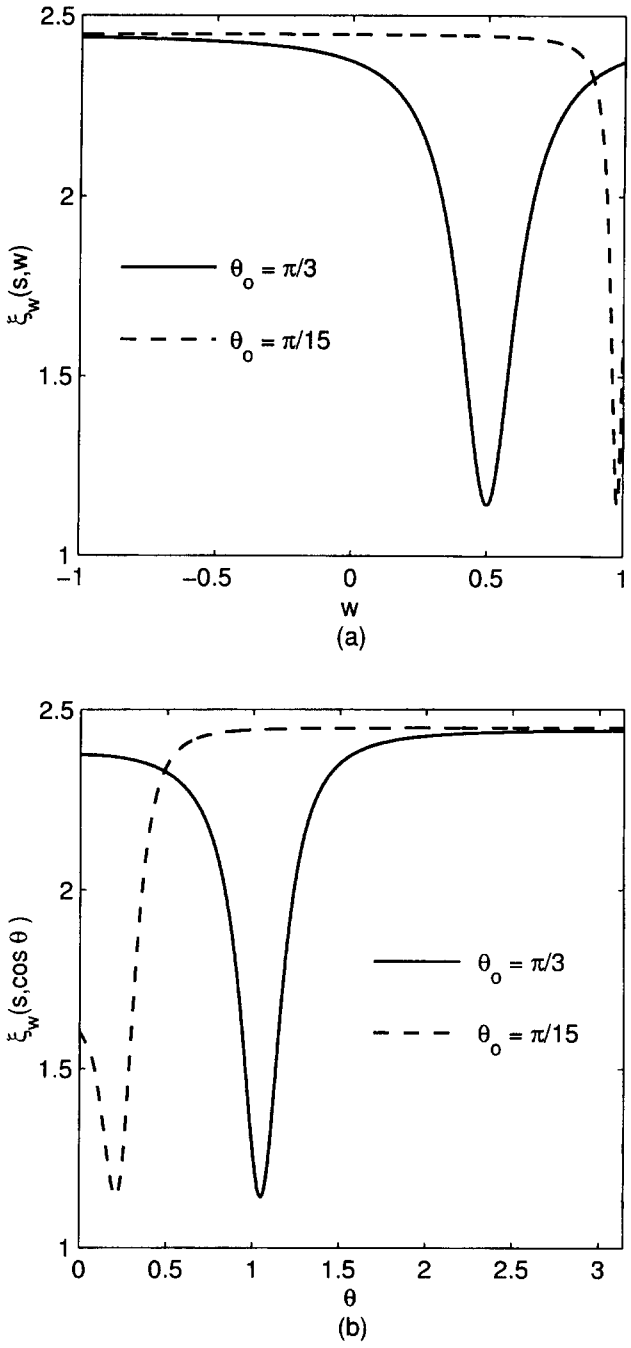


Figure 10.8 (a) Plots showing the variation of the performance function $\xi_w(s,w)$ as θ_o approaches zero. (b) Plots showing reduced sensitivity of the performance function $\xi_w(s,\cos\theta)$ to variations in θ_o. Reprinted from Farhang-Boroujeny (1997a)

two steps of the algorithm, we note that $s(n)$ is updated according to the recursive equation

$$s(n+1) = s(n) + \mu_s \cdot \frac{\partial y_s^2(n)}{\partial s(n)} \tag{10.44}$$

since our goal is to select $s(n)$ so that $\xi_s(s, w) = \mathrm{E}[y_s^2(n)]$ is maximized. Note also from (10.42) that

$$y_s^2(n) = \frac{1 + s(n)}{1 - s(n)} \cdot y^2(n).$$

10.3.4 Robust adaptation of w

Consider Figure 10.8(a), where two plots of $\xi_w(s, w)$ are given corresponding to $\theta_0 = \pi/3$ and $\pi/15$, with s fixed at 0.75. These plots show that the shape of the performance function $\xi_w(s, w)$ is sensitive to the value of θ_0. In particular, we note that when θ_0 is close to zero, the function $\xi_w(s, w)$ is nearly flat over most of the values of w, except when w is very close to its optimum value. This would result in extremely slow convergence for any gradient based algorithm, unless w is initialized close to its optimum value. The same sensitivity is observed when θ_0 is close to π.

This problem may be solved if we let $w = \cos\theta$ in (10.35) and adapt θ instead of w. With this amendment, the plots of the performance function $\xi_w(s, \cos\theta)$, as a function of θ, are as shown in Figure 10.8(b). We note that there is not much difference between the two plots in Figure 10.8(b), as opposed to the pair in Figure 10.8(a).

A robust implementation of the IIR ALE, which has reduced sensitivity to variations of θ_0, may thus be proposed by considering the change of variable $w = \cos\theta$ in (10.35) and adaptive adjustment of θ instead of w. Table 10.3 summarizes the resulting algorithm and is called Algorithm 2 for future reference. This algorithm, although

Table 10.3 Summary of the adaptive IIR ALE (Algorithm 2)

$w(n) = \cos\theta(n)$
$w'(n) = \sin\theta(n)$
$y(n) = (1 + s(n))w(n)y(n-1) - s(n)y(n-2) + (1 - s(n))(w(n)x(n) - x(n-1))$
$e(n) = x(n+1) - y(n)$
$\alpha(n) = (1 + s(n))w(n)\alpha(n-1) - s(n)\alpha(n-2) - w'(n)[(1 + s(n))y(n-1) + (1 - s(n))x(n)]$
$\theta(n+1) = \theta(n) + 2\mu_\theta e(n)\alpha(n)$
$\beta(n) = (1 + s(n))w(n)\beta(n-1) - s(n)\beta(n-2) - (w(n)e(n-1) - e(n-2))$
$s(n+1) = s(n) + 2\mu_s \left[\frac{1}{(1 - s(n))^2} y^2(n) + \frac{1 + s(n)}{1 - s(n)} y(n)\beta(n) \right]$

Definitions: $\alpha(n) = \frac{\partial y(n)}{\partial \theta(n)}$, $\beta(n) = \frac{\partial y(n)}{\partial s(n)}$
μ_θ and μ_s are step-size parameters

more complicated than Algorithm 1 (because of the involvement of the sine and cosine functions), has been found to be much more robust when θ_o is close to 0 or π (Farhang-Boroujeny, 1997a).

10.3.5 Simulation results

In this section we study the performance of the algorithms given in Tables 10.2 and 10.3 using computer simulations. We also discuss a cascade implementation of the IIR ALE which may be used for the enhancement of multiple sinusoidal signals.

Figure 10.9 presents a set of plots that shows the convergence as well as the tracking behaviour of Algorithms 1 and 2, when the ALE input is a single sinusoid in additive white Gaussian noise, as in (10.39). The simulated scenario consists of $\sigma_\nu^2 = 0.5$ and a unit amplitude sinusoid with angular frequency, $\theta_o(n)$, varying as shown in the figure. This corresponds to a signal-to-noise ratio (SNR) of 0 dB. The step-sizes are selected (empirically) according to the following equations:

$$\mu_s = 0.0005, \quad \mu_w(n) = 0.025(1 - s(n))^3, \quad \text{and} \quad \mu_\theta(n) = 0.05(1 - s(n))^3.$$

Note that the step-size parameters $\mu_w(n)$ and $\mu_\theta(n)$ are chosen to be time-varying and are selected according to the present value of $s(n)$. This results in large step-sizes when $s(n)$ is small and small step-sizes as $s(n)$ approaches one. The rationale behind this choice is the following. When $w(n)$ and $\theta(n)$ are far from their optimum values, $s(n)$ becomes small and hence it is better to use larger step-sizes to ensure faster convergence of the algorithm. On the other hand, when $w(n)$ and $\theta(n)$ are close to their optimum values, smaller step-sizes should be used to reduce the misadjustment of the algorithms. The above choices of $\mu_w(n)$ and $\mu_\theta(n)$ also compensate for the change in slope of the performance function $\xi_w(s, w)$ as $s(n)$ selects different values (see Figure 10.6). Thus, the equations proposed for adjusting $\mu_w(n)$ and $\mu_\theta(n)$ are based on these intuitions as well as a wide range of simulation tests. The parameter $s(n)$ is initialized to 0.25 at the beginning of each simulation and is allowed to vary in the range 0.25 to 0.9. For Algorithm 1, the parameter $w(n)$ is initialized to 0 $(= \cos^{-1}(\pi/2))$ and is confined to the range -0.999 to 0.999. Similarly, for Algorithm 2, $\theta(n)$ is initialized to $\pi/2$ and is confined to the range $\cos^{-1}(-0.999)$ to $\cos^{-1}(0.999)$. The results clearly show the superior performance of Algorithm 2. In particular, observe that as $\theta_o(n)$ approaches zero, its estimate, $\theta(n)$, becomes more noisy when Algorithm 1 is used. On the contrary, Algorithm 2 is much more robust.

To enhance or extract multiple sinusoids, we may use a cascade of a few IIR ALEs as in Figure 10.10. This configuration corresponds to an L-stage line enhancer, where each stage is responsible for the enhancement of one single sinusoid. The output error from each stage is the input to the next stage. The enhanced narrow-band outputs, the $y_i(n)$s, from the successive stages are added together to obtain the final output, $y(n)$, of the line enhancer. The adaptation of the multistage line enhancer begins with its first stage. The adaptation of the following stages begin once the previous stages have converged. Experiments have shown that this method works well (Cho and Lee, 1993). To decide on activating/deactivating the successive stages of the multistage IIR ALE we may use the $s(n)$ parameter of the previous stages. We know that for each stage $s(n)$ increases and approaches one only when $w(n)$ (or $\theta(n)$) is near its optimum value. Thus, by comparing

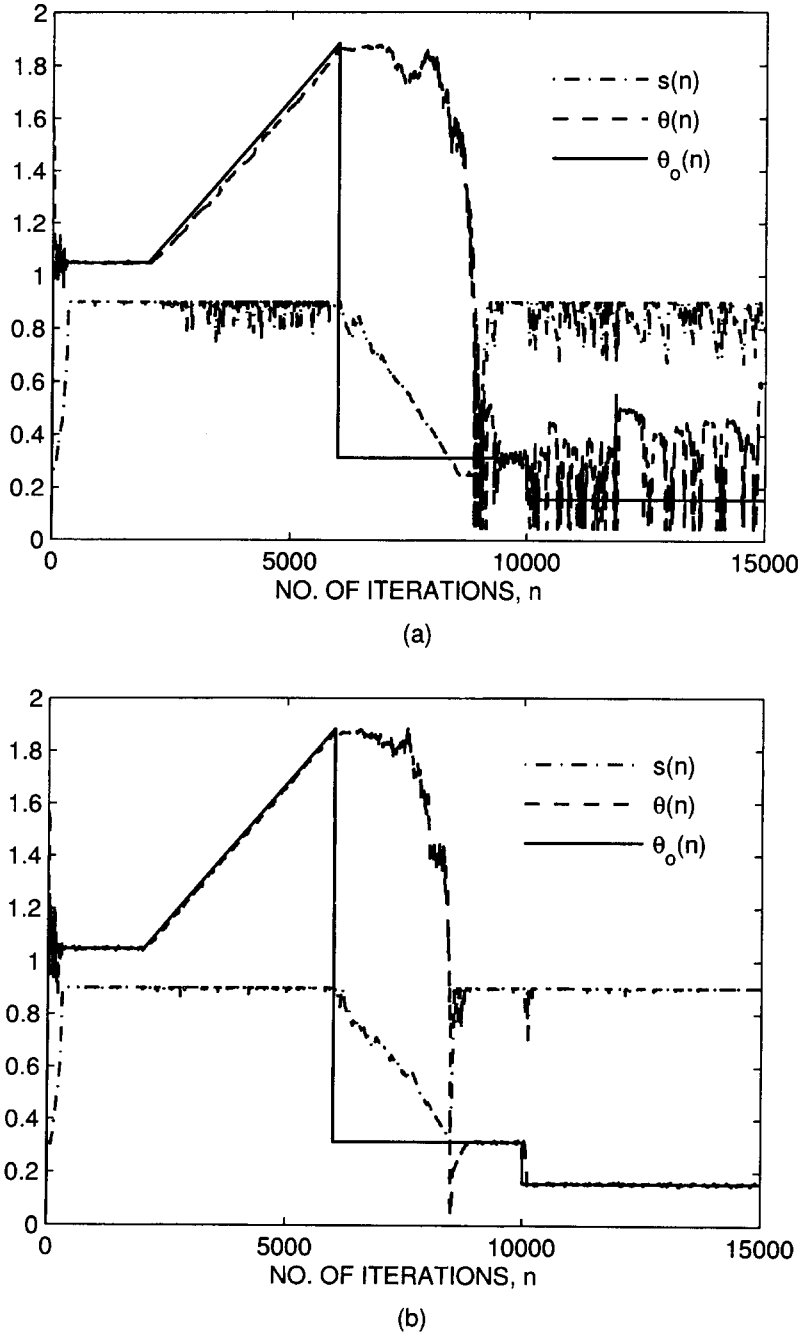


Figure 10.9 Simulation results illustrating the convergence as well as the tracking behaviour of the IIR ALE: (a) Algorithm 1, (b) Algorithm 2. Reprinted from Farhang-Boroujeny (1997a)

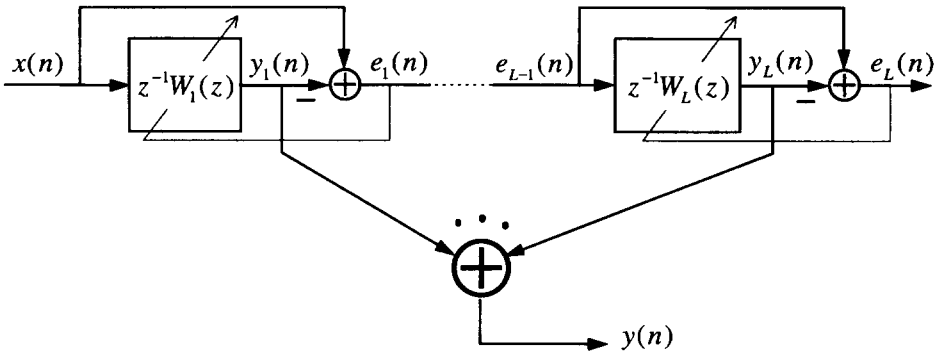


Figure 10.10 Cascaded IIR ALE for the enhancement of multiple sinusoids buried in white noise

$s(n)$ of each stage with a threshold level, we may decide on activating or deactivating the adaptation of the following stage(s). This provides a very simple and effective mechanism for controlling the adaptation of the cascaded IIR ALE.

Figure 10.11 illustrates the performance of the cascaded IIR ALE when Algorithm 2 is used. The input signal consists of the sum of four sinusoids in additive white Gaussian noise, and is given by

$$x(n) = \sin(\omega_1 n + \phi_1) + 2\sin(\omega_2 n + \phi_2) + 0.25\sin(\omega_3 n + \phi_3) + 0.5\sin(\omega_4 n + \phi_4) + \nu(n),$$

where ω_1, ω_2, ω_3 and ω_4 are equal to $\pi/1.8$, $\pi/3.5$, $\pi/6$ and $\pi/12$, respectively, ϕ_1 through ϕ_4 are random phases that are selected at the beginning of each simulation trial and remain fixed during that trial, and $\sigma_\nu^2 = 0.25$. This value corresponds to the SNRs of 3, 9, -9 and -3 dB, respectively, for the individual sinusoids. Since the power of the input signals to successive stages of the ALE are different, the step-sizes of each stage are normalized to the power of the input signal to that stage. The equations used for this purpose are $\mu_{s,i} = 0.0005/\hat{\sigma}_{x_i}^2$ and $\mu_{\theta,i}(n) = 0.01(1 - s(n))^3/\hat{\sigma}_{x_i}^2$. In these equations, i refers to the stage number, and $\hat{\sigma}_{x_i}^2$ is an estimate of the power of the input signal, $x_i(n)$, to the ith stage. The following recursive equation is used for estimation of $\hat{\sigma}_{x_i}^2$:

$$\hat{\sigma}_{x_i}^2(n) = 0.98\hat{\sigma}_{x_i}^2(n-1) + 0.02x_i^2(n).$$

The $s_i(n)$ parameters are allowed to change between 0.25 and 0.9, and the threshold level used for activating or deactivating the adaptation of the following stages is set at 0.85. The results in Figure 10.11 show that this mechanism works very well. It may also be noted that the first stage is tuned to the strongest sinusoid (ω_2), and the last stage is tuned to the weakest one (ω_3). Such observation is intuitively sound. The MATLAB programs used to generate the results of this section are available on an accompanying diskette. The reader is encouraged to examine these programs and run further simulations to learn more about the line enhancer as well as the difficulties that may be encountered in using IIR adaptive filters. It would be also interesting to compare the behaviour of FIR and IIR line enhancers. This is left as an exercise for the interested reader.

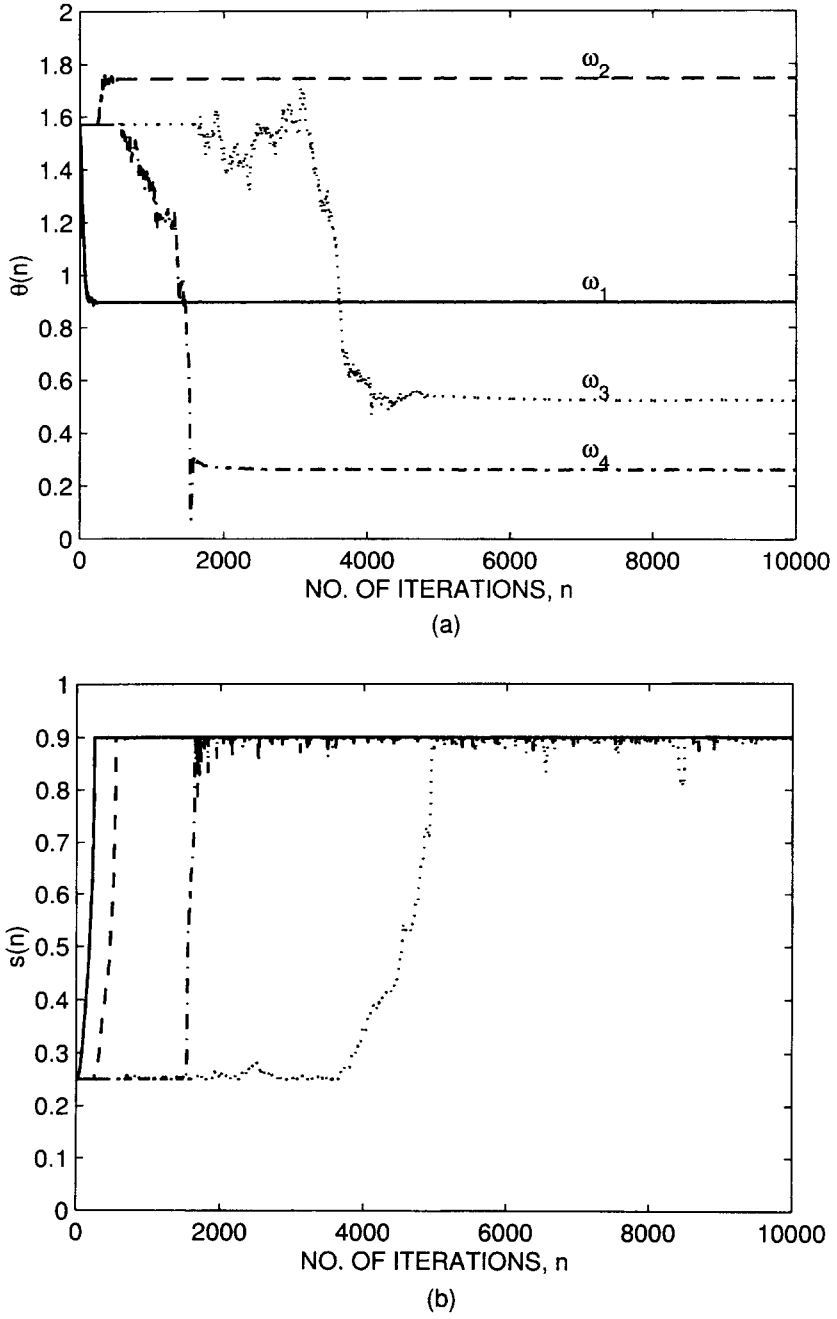


Figure 10.11 Simulation results showing convergence of the cascaded IIR ALE when used to detect/enhance multiple sinusoids: (a) angular frequencies, (b) $s(n)$ parameters. Algorithm 2 is used for the adaptation. Reprinted from Farhang-Boroujeny (1997a)

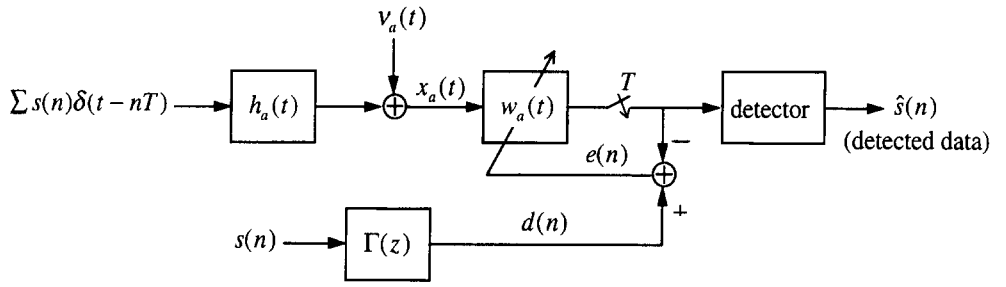


Figure 10.12 Model of a magnetic recording channel

10.4 Case Study II: Equalizer Design for Magnetic Recording Channels

Figure 10.12 depicts the block diagram of the magnetic recording channel that we wish to address in this section. This channel is characterized by its continuous time impulse response, $h_a(t)$. The subscript 'a' is to emphasize that $h_a(t)$ is an analogue quantity, i.e. it is a continuous function in amplitude as well as time, t. As was noted in Chapter 1 (Section 1.6.2), $h_a(t)$ is also called *the dibit response* and is usually modelled as the superposition of positive and negative Lorentzian pulses, separated by one bit interval, T. That is,

$$h_a(t) = g_a(t) - g_a(t - T), \tag{10.45}$$

where $g_a(t)$ is the Lorentzian pulse defined as

$$g_a(t) = \frac{1}{1 + \left(\dfrac{2t}{t_{50}}\right)^2} \tag{10.46}$$

and it is the response of the channel to a step input. The parameter t_{50}, which is the pulse-width of $g_a(t)$ measured at 50% of its maximum amplitude, is an indicator of the *recording density*. The recording density, D, is specified by the ratio t_{50}/T. Clearly, higher density implies denser storage and vice versa.

The response of the channel to the data bits,[2] $s(n)$, is then

$$x_a(t) = \sum_n s(n)h_a(t - kT) + \nu_a(t), \tag{10.47}$$

where $\nu_a(t)$ is the channel noise. The detector assumes that its input is the convolution of the data bits, $s(n)$, with a known response, called the target response, and it uses this information in doing the detection. Hence, our aim is to design an analogue equalizer

[2] The data bits $s(n)$ are assumed to take values $+1$ and -1.

(filter) whose impulse response, $w_a(t)$, when convolved with the dibit response, $h_a(t)$, matches the desired target response as closely as possible. In particular, we are interested in matching the combined response of the channel and equalizer, i.e.

$$\eta_a(t) = \int_{-\infty}^{\infty} w_a(\tau) h_a(t-\tau)\, d\tau, \tag{10.48}$$

with the target response at sampling instants separated by the bit interval, T. As was noted in Chapter 1 (Section 1.6.2), the target response in magnetic channels is usually one of the class-IV partial responses characterized by the transfer functions

$$\Gamma(z) = z^{-\Delta}(1+z^{-1})^K(1-z^{-1}), \tag{10.49}$$

where z^{-1} represents one bit delay, Δ is a parameter that takes care of the delays introduced by the channel and equalizer, and K is an integer greater than or equal to one. The choice of K depends on the recording density, D. The value of K also determines the complexity of the detector. The commonly used values of K are 1, 2 and 3.

Next, we go through a sequence of discussions which lead us to a design methodology, using the results of this chapter as well as the previous chapters, for designing analogue equalizers in the application of magnetic recording channels.

10.4.1 Channel discretization

Since all the derivations in this book are based on sampled signals, we would like to replace the continuous time channel and equalizer impulse responses, $h_a(t)$ and $w_a(t)$, respectively, by their associated discrete-time counterparts. Define the sequences

$$h_i = h_a(iT_s) \quad \text{and} \quad w_i = w_a(iT_s)$$

where T_s is the sampling period. When T_s is sufficiently small, we obtain, from (10.48),

$$\eta_i = \eta_a(iT_s) \approx T_s \cdot (h_i * w_i), \tag{10.50}$$

where an asterisk denotes convolution. The identity (10.50) follows from (10.48) by setting $t = iT_s$ and approximating the integration on the right-hand side of (10.48) by a summation.

[redacted]

10.4.2 Design steps

The following steps are taken in designing analogue equalizers for magnetic recording channels:[3]

1. Using the sampled channel response, h_i, and the statistics of the channel noise (e.g. the autocorrelation of the noise at the channel output), a fractionally tap-spaced FIR equalizer[4] is designed. The criterion that we use in this design is the mean-square error between the signal samples at the equalizer output and the desired signal that is obtained by passing the data sequence, $s(n)$, through the target response $\Gamma(z)$, as in Figure 10.12.
2. A discrete-time IIR filter whose impulse response matches best with the designed FIR equalizer is found. The equation error method will be used to find this match.
3. The discrete-time IIR filter obtained in Step 2 is then converted to an equivalent analogue filter as the desired analogue equalizer.

Next, we proceed with the details of the above steps.

10.4.3 FIR equalizer design

We define the equalizer tap-input and tap-weight vectors as

$$\mathbf{x}(n) = [x(n) \;\; x(n-1) \;\; \cdots \;\; x(n-N+1)]^{\mathrm{T}} \tag{10.52}$$

and

$$\mathbf{w} = [w_0 \;\; w_1 \;\; \cdots \;\; w_{N-1}]^{\mathrm{T}}, \tag{10.53}$$

respectively. The equalizer output is then

$$y(n) = \mathbf{w}^{\mathrm{T}}\mathbf{x}(n). \tag{10.54}$$

We note that the samples of the equalizer input, $x(n)$, and output, $y(n)$, are at T_s intervals. However, in the optimization of the equalizer tap weights, the w_is, we are only interested in samples of $y(n)$ at $T = LT_s$ intervals. Hence, we define the error as

$$e(n) = d(n) - y(nL) \tag{10.55}$$

and, accordingly, the performance function as

$$\xi = \mathrm{E}[e^2(n)], \tag{10.56}$$

where

$$d(n) = \sum_i \gamma_i s(n-i) \tag{10.57}$$

---

[3] The design procedure discussed here follows Mathew, Farhang-Boroujeny and Wood (1997).

[4] The term fractionally tap-spaced equalizer refers to the fact that the spacing between the successive taps of the equalizer, w_i, is T_s which, as noted earlier, is a few times smaller than the bit interval, T.

and the γ_is are the samples of the target response that are obtained by taking the inverse z-transform of $\Gamma(z)$ (see Figure 10.12). As an example, when $K = 2$,

$$\begin{aligned}\Gamma(z) &= z^{-\Delta}(1 + z^{-1})^2(1 - z^{-1}) \\ &= z^{-\Delta} + z^{-(\Delta+1)} - z^{-(\Delta+2)} - z^{-(\Delta+3)}\end{aligned}$$

and this gives

$$\gamma_i = \begin{cases} 1, & \text{for } i = \Delta \text{ and } \Delta + 1, \\ -1, & \text{for } i = \Delta + 2 \text{ for } \Delta + 3, \\ 0, & \text{otherwise.} \end{cases}$$

[redacted]

equalizer, we need to shift $h_a(t)$ to the right by a sufficient length, t_o, such that the remaining non-causal part of the shifted dibit could be ignored. This is done by replacing $h_a(t)$ with $h_a(t - t_o)$ in the earlier results and assuming that $h_a(t - t_o) = 0$, for $t < 0$. We also redefine the sampled dibit response, h_i, as

$$h_i = h_a(iT_s - t_o).$$

$$\mathbf{H} = \begin{bmatrix} h_0 & h_L & h_{2L} & h_{3L} & \cdots & h_{M-L} & 0 & \cdots \\ 0 & h_{L-1} & h_{2L-1} & h_{3L-1} & \cdots & h_{M-L-1} & h_{M-1} & \cdots \\ 0 & h_{L-2} & h_{2L-2} & h_{3L-2} & \cdots & h_{M-L-2} & h_{M-2} & \cdots \\ \vdots & \cdots & \vdots & \ddots & \vdots & \vdots & \ddots & \\ 0 & h_0 & h_L & h_{2L} & \cdots & h_{M-2L} & h_{M-L+1} & \cdots \\ 0 & 0 & h_{L-1} & h_{2L-1} & \cdots & h_{M-L-1} & h_{M-1} & \cdots \\ 0 & 0 & h_{L-2} & h_{2L-2} & \cdots & h_{M-L-2} & h_{M-2} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \ddots & \vdots & \vdots & \ddots \\ 0 & 0 & h_0 & h_L & \cdots & h_{M-2L} & h_{M-L} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \ddots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{10.61}$$

and $\boldsymbol{\nu}(nL)$ is the associated vector of samples of the channel noise, $\nu_a(t)$. We may also write (10.57) as

$$d(n) = \boldsymbol{\gamma}^T \mathbf{s}(n), \tag{10.62}$$

where $\boldsymbol{\gamma}$ is the column vector consisting of the samples of the target response, the γ_is. The length of $\boldsymbol{\gamma}$ is appropriately selected by appending extra zeros at its end so that it would be compatible with $\mathbf{s}(n)$.

Using the above results and assuming that the binary process, $s(n)$, and the noise process, $\nu(n)$, are white and independent of one another, we obtain

$$\mathbf{R} = \mathbf{H}\mathbf{H}^T + \sigma_\nu^2 \mathbf{I}, \tag{10.63}$$

where σ_ν^2 is the variance of $\nu(n)$ and $\mathbf{I}$ is the identity matrix. In arriving at (10.63), we have also used the fact that $E[\mathbf{s}(n)\mathbf{s}^T(n)] = \mathbf{I}$ since $s(n)$ is white with values ± 1. Similarly, we also obtain

$$\mathbf{p} = \mathbf{H}\boldsymbol{\gamma}. \tag{10.64}$$

Substituting (10.63) and (10.64) in (10.58), we obtain the following explicit equation for the desired optimum fractionally tap-spaced FIR equalizer:

$$\mathbf{w}_o = (\mathbf{H}\mathbf{H}^T + \sigma_\nu^2 \mathbf{I})^{-1} \mathbf{H}\boldsymbol{\gamma}. \tag{10.65}$$

10.4.4 Conversion from the FIR to the IIR equalizer

As the next step in designing analogue equalizers, we need to find an IIR filter whose response closely matches the designed FIR equalizer. For this, we use the method of equation error that was discussed in Section 10.2. With reference to Figure 10.4, in the present context of magnetic recording, $x(n)$ is the channel output, $G(z)$ is the designed FIR equalizer, $e_o(n) = 0$, for all n, $A(z)$ and $B(z)$ are polynomials that define the transfer function $W(z)$ of the desired IIR filter according to (10.1), and the unit delay z^{-1} is

equivalent to one T_s interval. We can use the LMS or any other adaptive filtering algorithm to find the coefficients of $A(z)$ and $B(z)$. We may also adopt an analytical method and develop a closed-form solution for the coefficients of $A(z)$ and $B(z)$, or use time averages to estimate the coefficients of the related Wiener–Hopf equation. We use the last method in the numerical examples discussed below for convenience.

10.4.5 Conversion from the z-domain to the s-domain

Among the different methods available for conversion between s-domain and z-domain transfer functions, we discuss the method of impulse invariance. To give a brief introduction to this method, we consider a causal continuous-time system with the transfer function

$$H_a(s) = \sum_i \frac{a_i}{s - s_i}, \tag{10.66}$$

with the s_is being the poles of the system. The impulse response of this system is

$$h_a(t) = \begin{cases} \sum_i a_i e^{s_i t}, & \text{for } t \geq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{10.67}$$

Now, if we consider a discrete-time system whose unit-sample (impulse) response is given by the samples $h_a(0)$, $h_a(T_s)$, $h_a(2T_s), \ldots$, its transfer function will be

$$\begin{aligned} H(z) &= \sum_{k=-\infty}^{\infty} h_a(kT_s) z^{-k} \\ &= \sum_{k=0}^{\infty} \sum_i a_i e^{s_i k T_s} z^{-k} \\ &= \sum_i \sum_{k=0}^{\infty} a_i e^{s_i k T_s} z^{-k} \\ &= \sum_i \frac{a_i}{1 - e^{s_i T_s} z^{-1}}. \end{aligned} \tag{10.68}$$

The reverse of this conversion is obvious. That is, if

$$H(z) = \sum_i \frac{a_i}{1 - z_i z^{-1}} \tag{10.69}$$

is the transfer function of a discrete-time system with unit-sample response h_n, then the transfer function of the continuous-time system whose impulse response samples, at T_s intervals, is the sequence h_n, is

$$H_a(s) = \sum_i \frac{a_i}{s - (1/T_s) \ln z_i}. \tag{10.70}$$

10.4.6 Numerical results

To highlight some of the features of the design method that was developed above, we present some numerical results using the Lorentzian pulse (see (10.45) and (10.46)) as the model for the magnetic recording channel. The measure used for evaluating the designed equalizer is the signal-to-noise ratio (SNR) at the detector input. It is defined as

$$\text{detection SNR} = \frac{\sum_i \gamma_i^2}{\sum_i (\eta_{iL} - \gamma_i)^2 + \sigma_\nu^2 \sum_i w_i^2}. \tag{10.71}$$

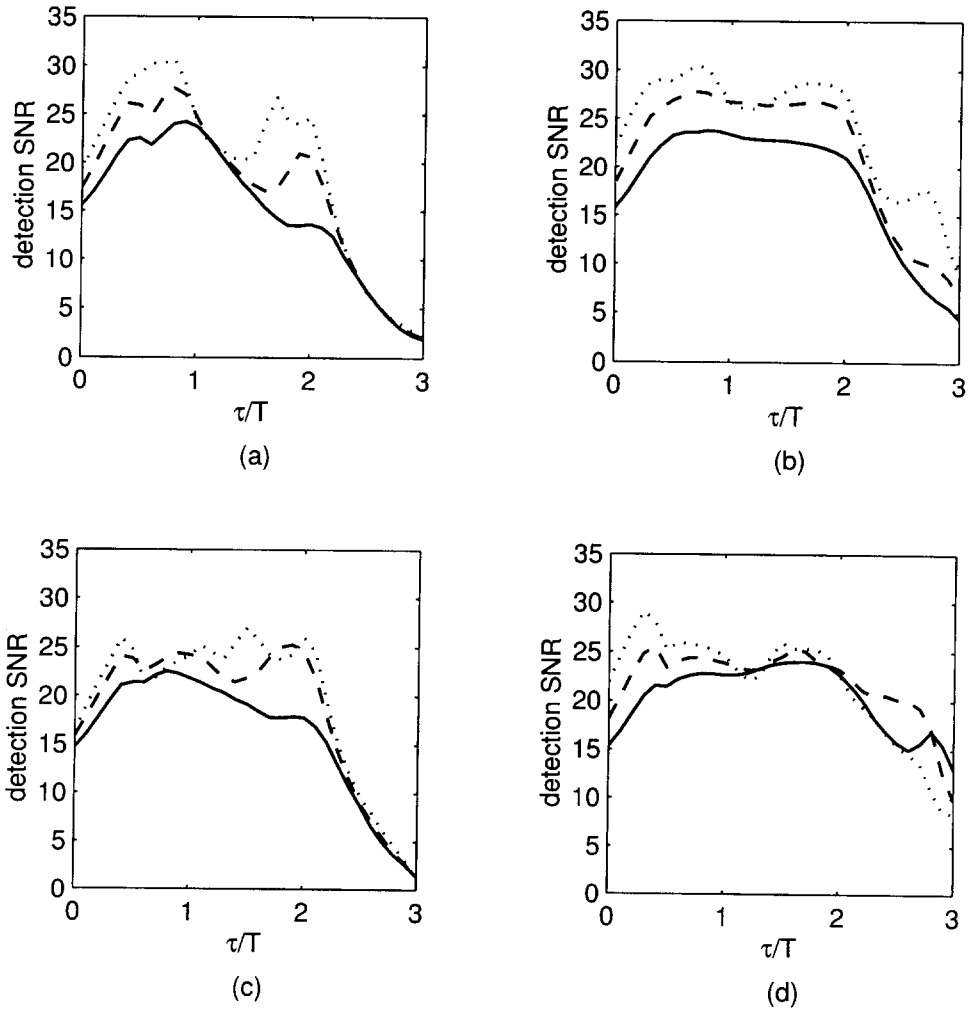


Figure 10.13 Performance of the IIR equalizers designed for different choices of the parameters: (a) three zeros and four poles, $D = 2$; (b) five zeros and six poles, $D = 2$; (c) three zeros and four poles, $D = 2.5$; (b) five zeros and six poles, $D = 2.5$. The three plots in each case correspond to the following choices of channel SNR: 25 dB (——); 30 dB (– – –), 35 dB (· · · · ·)

The value of the variance of the channel noise, σ_v^2, is selected on the basis of another SNR that is defined at the equalizer input (or channel output) as

$$\text{SNR at the equalizer input} = \frac{\sum_i h_i^2}{\sigma_v^2}. \tag{10.72}$$

It may be noted from (10.71) that the noise at the detector input is considered to be the sum of channel noise at the equalizer output and residual intersymbol interference (ISI). Further exploration of this definition is left as an exercise for the interested reader.

We present results of many designs that are obtained for various choices of channel parameters. Evaluating these results, we find that, among the different parameters, the performance of the IIR equalizer is highly affected by the choice of the delays t_0 and Δ. Furthermore, the choice of t_0 is closely related to the value of Δ. In a good design, usually $t_0 = \Delta T + \tau$, where T (as defined before) is the bit interval and τ is a relatively small delay in the range 0 to $3T$. The best value of τ which results in maximum detection SNR depends on the number of zeros and poles of the IIR equalizer, the noise level in the channel, and the recording density, D. The effect of these on the optimum value of τ is difficult to predict. It appears that the only way of finding the optimum τ is to design many IIR equalizers for different values of τ and choose the best among them. In Figure 10.13, we show how the detection SNR varies as a function of τ/T for certain selected

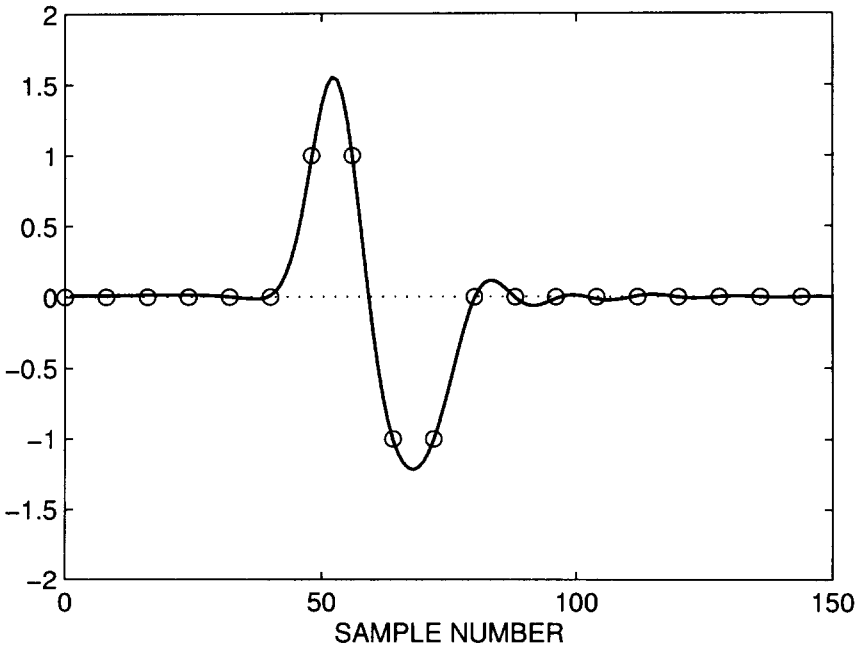


Figure 10.14 An example of the equalized dibit response of the magnetic recording channel for $K = 2$, $D = 2.5$, $\tau/T = 0.3$, channel SNR = 30 dB, and an IIR equalizer with five zeros and six poles. The circles correspond to the target response samples

choices of the recording density, D, the SNR at the equalizer input, and the number of poles and zeros of the IIR equalizer. The value of K is set at 2 in this set of results. These plots clearly indicate that the choice of τ is very critical in the final performance of the IIR equalizer.

Figure 10.14 shows an example of the equalized dibit response of the magnetic recording channel. This is obtained by passing the dibit response, h_i, through the designed IIR equalizer. The parameters used to obtain these results are: $D = 2.5$, $\tau/T = 0.3$, the SNR at the equalizer input = 30 dB, the IIR equalizer with five zeros and six poles, and $K = 2$. Observe that an almost perfect match between the equalizer output and the target response has been achieved here.

The MATLAB program that has been used to obtain these results is available on an accompanying diskette. It is called 'iirdsgn.m'. The reader is encouraged to run this program for other designs to enhance his/her understanding of the concepts that were discussed above.

10.5 Concluding Remarks

In this chapter we discussed the problem of IIR adaptive filtering. We noted that, unlike FIR adaptive filters, whose adaptation is a rather straightforward task, the adaptive adjustment of IIR filters is, in general, a complicated problem. IIR adaptive filters can easily become unstable since their poles may get shifted out of the unit circle by the adaptation process. Or, they can get trapped in one of the local minima points since the performance surfaces of IIR filters are, in general, multimodal. We saw that these problems could be resolved by either limiting ourselves to applications where special transfer functions with unimodal performance surfaces could be used, or using the method of equation error which leads to a suboptimal solution.

We also presented two case studies, one for each of the above solutions. These studies showed some of the difficulties that may be encountered while dealing with IIR adaptive filters – problems which do not arise when FIR adaptive filters are used. In the first case study we used a specific transfer function for realization of the line enhancers. There were many considerations that we had to take note of before getting to our final solution. For instance, we saw that our initial transfer function gets into difficulties when the frequency of the sinusoid is close to 0 or π. For this, we found a specific solution, namely replacement of the parameter w by $\cos\theta$ and adapting θ instead of w. We also had to take care of the parameter s of this structure in a very special way. In contrast to this, if we refer to Chapter 6 (Section 6.4.3) where we used an FIR filter to realize a line enhancer, we find that none of the above-mentioned kind of problems exists. The only point that we must consider while using an FIR filter is to include a sufficient number of taps. As was noted at the beginning of this chapter, the main advantage of IIR adaptive filters, as compared with their FIR counterparts, is their lower order which may lead to a lower computational complexity and hence a reduction in the cost of implementation.

The second case study that we discussed was equalization of magnetic recording channels. In this application we found that the optimum IIR equalizers are very sensitive to a delay parameter, τ. The results indicated that varying this delay even around its optimum value can significantly affect the performance of the resulting equalizer (Figure 10.13). A similar study for FIR equalizers shows that they do not exhibit such a level of

sensitivity. In fact, FIR equalizers are very robust in this respect. A study of this problem is left as an exercise for the reader.

To conclude, our study in this chapter showed that although the IIR adaptive filters are attractive for some specific applications, it may not be possible to use them (directly) for any arbitrary application. This is unlike the FIR (transversal) adaptive filters which are very versatile adaptive systems. While using IIR adaptive filters, special care has to be taken in the selection of the transfer function and/or the performance function depending upon the kind of application that we are dealing with.

At this point we should add that much of the research work in IIR adaptive filters has been carried out in the context of system modelling. The literature on this topic is much wider than what could be covered within the limits of a single chapter of this book. An excellent paper by Shynk (1989) provides a good review of the fundamental work done on this subject. A good bibliography of the key references is also provided in that paper. Another interesting and classic reference is the book by Ljung and Söderström (1983).

Problems

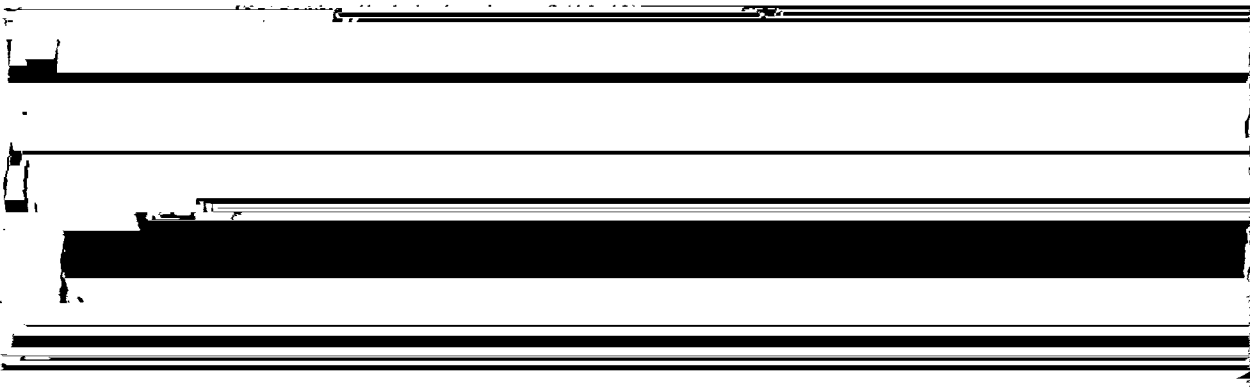
P10.1 Start with (10.4) and use (10.11) and (10.12) to give detailed derivations of (10.15) and (10.16), respectively.

P10.2 For the modelling problem that was discussed in Example 10.1, obtain the values of $a_{0,o}$, $b_{1,o}$, $a_{0,e}$ and $b_{1,e}$ for the cases where the power spectral density of the input, $x(n)$, is given as:

(i) $$\Phi_{xx}(e^{j\omega}) = \frac{1}{|1 - 0.3e^{-j\omega}|^2}.$$

(ii) $$\Phi_{xx}(e^{j\omega}) = \frac{1}{|1 - 0.8e^{-j\omega}|^2}.$$

From this study you should find that when $x(n)$ is coloured, both $a_{0,e}$ and $b_{1,e}$ are biased with respect to the optimum Wiener coefficients $a_{0,o}$ and $b_{1,o}$. Explain how these biases are affected by the shape of $\Phi_{xx}(e^{j\omega})$.



P10.4 Consider the case where the input to the line enhancer of Figure 10.5 is the sum of a sinusoid and a white noise as in (10.39).

(i) Show that

P10.5 Give a detailed derivation of (10.43).

P10.6 Give a detailed derivation of the LMS algorithm of Table 10.2.

P10.7 Give a detailed derivation of the LMS algorithm of Table 10.3.

P10.8 In the light of the result of Problem P10.4, adjustment of the parameter w of the line enhancer of Figure 10.5 may be done by maximizing the mean-square value of the output $y(n)$. Develop an LMS algorithm that works based on this principle. Also, develop another LMS algorithm that adapts $\theta = \cos^{-1} w$ instead of w, as in Algorithm 2 of Table 10.3.

P10.9 Give a formal proof of the fact that the performance function $\xi_w(s, w)$ of (10.40), for a given s, has only one minimum point and that corresponds to the value of $w = \cos\theta_0$.

P10.10 Study the transfer function between the input, $x(n)$, and output, $y(n)$, of Figure 10.10 and show that this is that of a filter with L narrow bands.

P10.11 Study the transfer function between the input, $x(n)$, and output error, $e_L(n)$, of Figure 10.10 and show that this is that of a filter with L notches.

P10.12 In line enhancers, signal enhancement is defined as the ratio of the SNR at the enhancer output to the SNR at its input. For the IIR ALE that was discussed in Section 10.3 show that when $x(n)$ is given by (10.39)

$$\text{the signal enhancement of IIR ALE} = \frac{1+s}{1-s}.$$

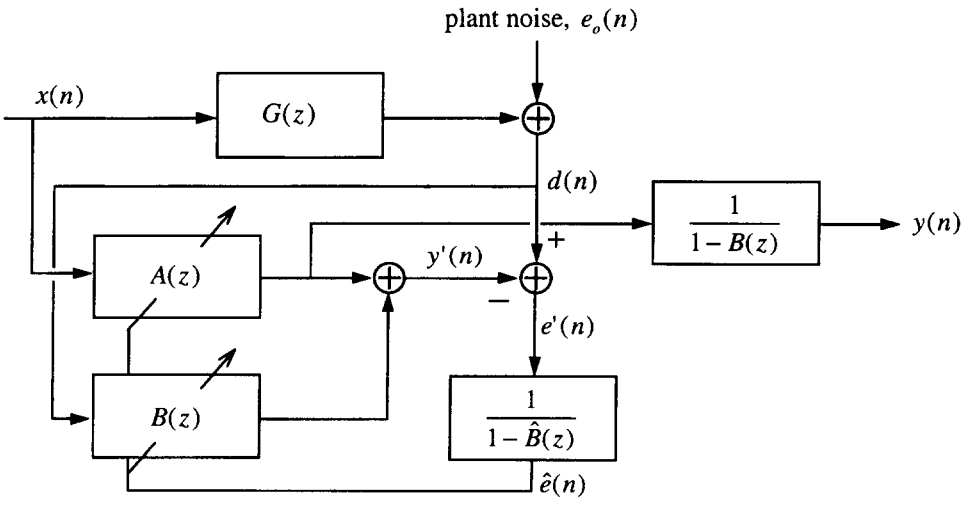


Figure P10.15

P10.13 Work out a detailed derivation of (10.59).

P10.14 For the magnetic recording channel that was discussed in Section 10.4, show that the mean-square value of the sum of residual ISI and noise at the equalizer output is $\sum_i(\eta_{iL} - \gamma_i)^2 + \sigma_\nu^2 \sum_i w_i^2$. Thus, justify the use of definition (10.71).

P10.15 From our discussion in Section 10.2, we recall that the output error, $e(n)$, and equation error, $e'(n)$, are related through the transfer function $1 - B(z)$. Assuming that a relatively good estimate of $B(z)$ (say, $\hat{B}(z)$) is available, it is proposed that the set-up of Figure P10.15 may be used to obtain better estimates of $A(z)$ and $B(z)$ compared with what could be achieved by the original equation error set-up of Figure 10.4. Elaborate on this diagram and explain why this set-up may give better estimates of $A(z)$ and $B(z)$, as compared with that in Figure 10.4. Also, develop an LMS algorithm for the adaptation of $A(z)$ and $B(z)$ in this set-up.

Simulation-Oriented Problems

P10.16 Develop programs for implementation of the LMS algorithms of Problem P10.8. For an input signal consisting of a sinusoid in additive white noise, as in (10.39), compare the convergence behaviour of these algorithms with Algorithms 1 and 2 of Tables 10.2 and 10.3, respectively. The MATLAB programs for Algorithms 1 and 2 are available on an accompanying diskette. As a benchmark for your comparisons, you may try to generate results similar to those in Figure 10.9.

P10.17 In magnetic recording, the best choice of the parameter K in the target response $\Gamma(z)$ depends on the recording density, D. In this exercise we study how the choice of K varies with D.

The program 'iirdsgn.m' on an accompanying diskette allows you to design IIR equalizers in the application of magnetic recording. Different parameters of interest (such as the recording density, the channel SNR and the equalizer order) are inputs to the program. Use this program to design IIR equalizers for the densities $D = 1.5$ to 3, in steps of 0.25, and the choices of the parameter $K = 1, 2$ and 3. Assume a channel SNR of 30 dB and an equalizer with three zeros and four poles in all your designs. However, each design has to be optimized with respect to the delay, τ. The criterion for the optimum design is the detection SNR. Tabulate your results and discuss how the choice of K varies with D.

11
Lattice Filters

In our discussions on FIR and IIR filters in the previous chapters, we always limited ourselves to implementation structures that were direct realizations of their corresponding system functions. In this chapter we introduce an alternative structure for the realization of FIR and IIR filters. This new structure, which is called a *lattice,* has a number of desirable properties that will become clear as we go along in this chapter. The lattice structure has most commonly been used for implementing linear predictors in the context of speech processing applications. Predictors may appear in two distinct forms: forward and backward. In a forward linear predictor the aim is to estimate the present sample of a signal $x(n)$ in terms of a linear combination of its past samples $x(n-1), x(n-2), \ldots, x(n-m)$. This corresponds to one-step forward prediction of order m. In backward linear prediction, on the other hand, an estimate of $x(n-m)$ is obtained as a linear combination of the future samples $x(n), x(n-1), \ldots, x(n-m+1)$.

In this chapter we start with a study of forward and backward linear predictors. We find that these two are closely related to each other. In particular, we introduce the so

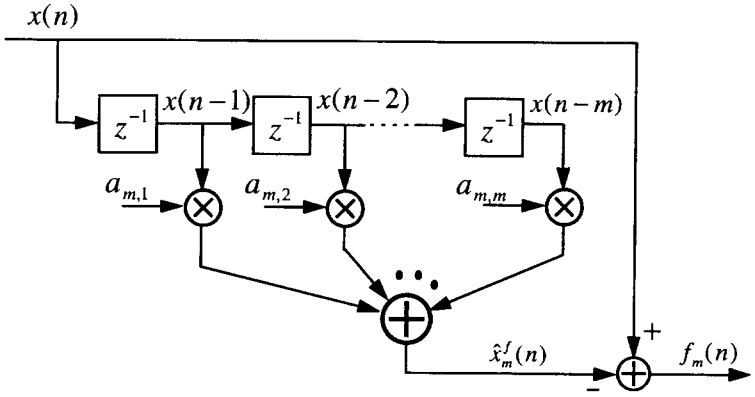


Figure 11.1 Forward linear predictor

of $\mathbf{a}_m$ to emphasize that the predictor order is m. The implementation structure of the type shown in Figure 11.1 is called the transversal or tapped delay line predictor, in contrast to the lattice structure that will be introduced later.

We assume that the input sequence, $x(n)$, is the realization of a stationary stochastic process. Furthermore, we assume that the predictor tap weights are optimized in the mean-square sense according to the Wiener filter theory. Thus, the optimum value of the predictor tap weights $a_{m,1}, a_{m,2}, \ldots, a_{m,m}$ is obtained by minimizing the function

$$P_m^{\mathrm{f}} = \mathrm{E}[f_m^2(n)], \tag{11.1}$$

where

$$f_m(n) = x(n) - \hat{x}_m^{\mathrm{f}}(n) \tag{11.2}$$

is the forward prediction error and

$$\hat{x}_m^{\mathrm{f}}(n) = \sum_{i=1}^{m} a_{m,i}x(n-i) = \mathbf{a}_m^{\mathrm{T}}\mathbf{x}_m(n-1) \tag{11.3}$$

is the mth order forward prediction of the input sample $x(n)$. This is a conventional Wiener filtering problem with the input vector $\mathbf{x}_m(n-1)$ and desired output $x(n)$. Hence, the corresponding Wiener–Hopf equation is obtained by direct substitution of $x(n)$ for $d(n)$ and $\mathbf{x}_m(n-1)$ for $\mathbf{x}(n)$ in (3.10) and (3.11), and recalling (3.24). The result is

$$\mathbf{R}\mathbf{a}_{m,\mathrm{o}} = \mathbf{r}, \tag{11.4}$$

where $\mathbf{R} = \mathrm{E}[\mathbf{x}_m(n-1)\mathbf{x}_m^{\mathrm{T}}(n-1)]$, $\mathbf{r} = \mathrm{E}[x(n)\mathbf{x}_m(n-1)]$ and $\mathbf{a}_{m,\mathrm{o}}$ denotes the optimum value of $\mathbf{a}_m$.

To simplify our notations in the discussion that follows, we assume that the predictor tap weights are always set to their optimum values and drop the extra subscript 'o' from

$\mathbf{a}_{m,o}$. Thus, (11.4) is simply written as

$$\mathbf{R}\mathbf{a}_m = \mathbf{r}. \tag{11.5}$$

When the predictor tap weights are set according to (11.5), P_m^f is minimized and this can be obtained using (3.26) as

$$\begin{aligned} P_m^f &= \mathrm{E}[x^2(n)] - \mathbf{r}^T\mathbf{a}_m \\ &= \mathrm{E}[x^2(n)] - \mathbf{r}^T\mathbf{R}^{-1}\mathbf{r}, \end{aligned} \tag{11.6}$$

assuming that $\mathbf{R}$ is non-singular.

For future use, we define the autocorrelation function of the input process for lag k as

$$r(k) = \mathrm{E}[x(n)x(n-k)]. \tag{11.7}$$

Using this definition, we note that

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) & \cdots & r(m-1) \\ r(1) & r(0) & \cdots & r(m-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(m-1) & r(m-2) & \cdots & r(0) \end{bmatrix} \tag{11.8}$$

and

$$\mathbf{r} = \begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(m) \end{bmatrix}. \tag{11.9}$$

We note that $\mathbf{R}$ and $\mathbf{r}$, with the exception of $r(0)$ and $r(m)$, share the same set of elements. This very close relationship between $\mathbf{R}$ and $\mathbf{r}$ is the key to many interesting properties of the linear predictors that will be brought out in this chapter.

11.2 Backward Linear Prediction

Figure 11.2 depicts an m th order backward linear predictor. A transversal filter with tap-input vector $\mathbf{x}_m(n) = [x(n)\ x(n-1)\ \ldots\ x(n-m+1)]^T$ and tap-weight vector $\mathbf{g}_m = [g_{m,1}\ g_{m,2}\ \cdots\ g_{m,m}]^T$ is used to obtain an estimate of the input sample $x(n-m)$. As in the forward prediction case, we assume that the backward predictor tap weights are optimized in the mean-square sense according to the Wiener filter theory. The optimum value of the predictor tap weights $g_{m,1}, g_{m,2}, \ldots, g_{m,m}$ are then obtained by minimizing the function

$$P_m^b = \mathrm{E}[b_m^2(n)], \tag{11.10}$$

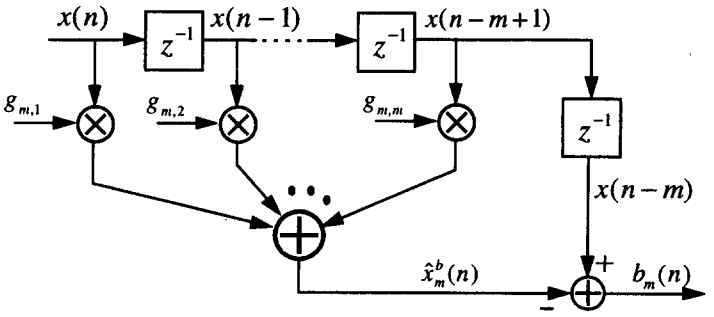


Figure 11.2 Backward linear predictor

where

$$b_m(n) = x(n-m) - \hat{x}_m^{\mathrm{b}}(n) \tag{11.11}$$

is the backward prediction error and

$$\hat{x}_m^{\mathrm{b}}(n) = \sum_{i=1}^{m} g_{m,i}\, x(n-i+1) = \mathbf{g}_m^{\mathrm{T}} \mathbf{x}_m(n) \tag{11.12}$$

is the m th order backward prediction of the input sample $x(n-m)$. This is a conventional Wiener filtering problem with the input vector $\mathbf{x}_m(n)$ and desired output $x(n-m)$. Hence, the corresponding Wiener–Hopf equation is obtained by direct substitution of $x(n-m)$ for $d(n)$ and $\mathbf{x}_m(n)$ for $\mathbf{x}(n)$ in (3.10) and (3.11), and recalling (3.24). The result is

$$\mathbf{R}\mathbf{g}_m = \mathbf{r}_b, \tag{11.13}$$

where $\mathbf{R} = \mathrm{E}[\mathbf{x}_m(n)\mathbf{x}_m^{\mathrm{T}}(n)]$ and $\mathbf{r}_b = \mathrm{E}[x(n-m)\mathbf{x}_m(n)]$.

Since $x(n)$ is stationary, the correlation matrix $\mathbf{R}$ in (11.13) is the same matrix as that in (11.5). However, the vector $\mathbf{r}_b$ on the right-hand side of (11.13) is different from the vector $\mathbf{r}$ in (11.5). Using the definition (11.7), we obtain

$$\mathbf{r}_b = \begin{bmatrix} r(m) \\ r(m-1) \\ \vdots \\ r(1) \end{bmatrix}. \tag{11.14}$$

Comparing (11.14) and (11.9), we note that $\mathbf{r}_b$ is the same as the vector $\mathbf{r}$ with its elements arranged in reverse order.

When the tap weights of the backward predictor are optimized according to (11.13),

$$\begin{aligned} P_m^{\mathrm{b}} &= \mathrm{E}[x^2(n-m)] - \mathbf{r}_b^{\mathrm{T}}\mathbf{g}_m \\ &= \mathrm{E}[x^2(n-m)] - \mathbf{r}_b^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{r}_b. \end{aligned} \tag{11.15}$$

11.3 The Relationship Between Forward and Backward Predictors

We now show that there is a close relationship between the tap-weight vectors of the forward and backward linear predictors of a process $x(n)$. To see this, we substitute (11.8) and (11.9) in (11.5) and write the result in scalar form as

$$\sum_{i=1}^{m} r(i-j)a_{m,i} = r(j), \quad \text{for } j = 1, 2, \ldots, m, \tag{11.16}$$

where we have used the property $r(i-j) = r(j-i)$. Also, substitution of (11.8) and (11.14) in (11.13) gives

$$\sum_{i=1}^{m} r(i-j)g_{m,i} = r(m+1-j), \quad \text{for } j = 1, 2, \ldots, m. \tag{11.17}$$

Next, we let $i = m+1-k$ and $j = m+1-l$ in (11.17) and use $r(k-l) = r(l-k)$ to obtain

$$\sum_{k=1}^{m} r(k-l)g_{m,m+1-k} = r(l), \quad \text{for } l = 1, 2, \ldots, m. \tag{11.18}$$

Replacing k and l in (11.18) by i and j, respectively, and comparing the result with (11.16), we get

$$a_{m,i} = g_{m,m+1-i}, \quad \text{for } i = 1, 2, \ldots, m \tag{11.19}$$

or

$$g_{m,i} = a_{m,m+1-i}, \quad \text{for } i = 1, 2, \ldots, m. \tag{11.20}$$

This result shows that *the optimum tap weights of the m th order forward predictor of a wide sense stationary process $x(n)$ are the same as the optimum tap weights of the corresponding backward predictor, but in reverse order.* Thus, we may write

$$f_m(n) = x(n) - \sum_{i=1}^{m} a_{m,i}x(n-i) \tag{11.21}$$

and

$$b_m(n) = x(n-m) - \sum_{i=1}^{m} a_{m,m+1-i}x(n-i+1). \tag{11.22}$$

11.4 Prediction-Error Filters

The forward predictor of Figure 11.1 uses an m-tap transversal filter to get an estimate of the present sample $x(n)$ of a sequence based on its past m samples $x(n-1)$,

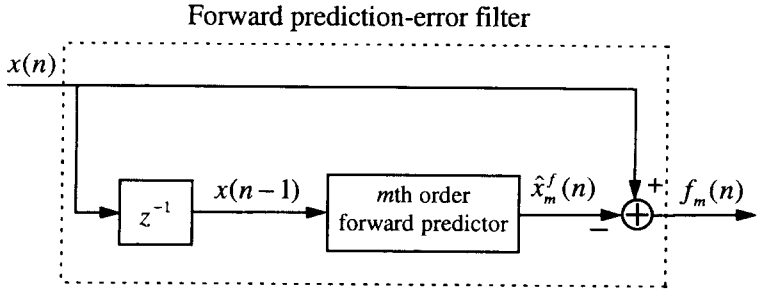


Figure 11.3 Block schematic showing the relationship between the forward predictor and the forward prediction-error filter

$x(n-2), \ldots, x(n-m)$. The mth order forward prediction-error filter for a sequence $x(n)$ is defined as the filter whose input is $x(n)$ and the forward prediction error $f_m(n)$ is its output. Figure 11.3 depicts a block schematic showing how the forward predictor and the forward prediction-error filter are related.

Similarly, the mth order backward prediction-error filter of a sequence $x(n)$ is the one whose input is $x(n)$ and its output is the backward prediction error $b_m(n)$. Figure 11.4 depicts a block schematic showing how the backward predictor and the backward prediction-error filter are related.

11.5 The Properties of Prediction Errors

The forward and backward prediction errors possess certain properties which are fundamental to the development of lattice structures. These properties are reviewed in this section.

Property 1 *For any sequence $x(n)$, the forward and backward prediction errors of the same order have the same power. In other words,*

$$P_m^{\mathrm{b}} = P_m^{\mathrm{f}}. \tag{11.23}$$

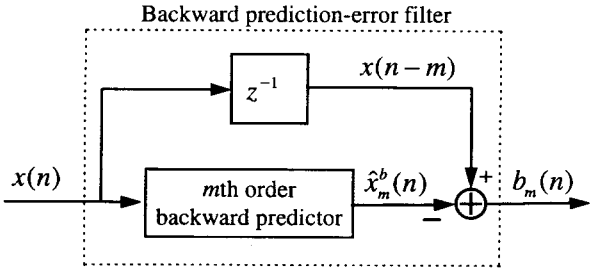


Figure 11.4 Block schematic showing the relationship between the backward predictor and the backward prediction-error filter

To show this, we note from (11.15),

$$P_m^b = \mathrm{E}[x^2(n-m)] - \sum_{i=1}^{m} g_{m,i} r(m+1-i). \tag{11.24}$$

Substituting (11.20) in (11.24) and noting that $\mathrm{E}[x^2(n-m)] = \mathrm{E}[x^2(n)]$ since $x(n)$ is stationary, we obtain

$$P_m^b = \mathrm{E}[x^2(n)] - \sum_{i=1}^{m} a_{m,m+1-i} r(m+1-i). \tag{11.25}$$

The proof of (11.23) is now complete since the right-hand side of (11.25) with $j = m+1-i$ is same as P_m^f given by (11.6).

This result shows that, *for a random process $x(n)$, the forward and backward predictors achieve the same level of minimum mean-square error, when their tap weights are optimized.* Noting this, we drop the superscripts f and b from P_m^f and P_m^b, respectively, in the rest of this chapter.

Property 2 *For any sequence $x(n)$ and its m th order forward prediction error $f_m(n)$*

$$\mathrm{E}[f_m(n)x(n-k)] = 0, \quad \text{for } k = 1, 2, \ldots, m. \tag{11.26}$$

This is easily proved by applying the principle of orthogonality to the forward predictor of Figure 11.1. Namely, the output error $f_m(n)$ is uncorrelated (orthogonal) with the samples $x(n-1), x(n-2), \ldots, x(n-m)$ at the filter (predictor) input.

Property 3 *For any sequence $x(n)$ and its m th order backward prediction error $b_m(n)$*

$$\mathrm{E}[b_m(n)x(n-k)] = 0, \quad \text{for } k = 0, 1, \ldots, m-1. \tag{11.27}$$

This also is proved by applying the principle of orthogonality to the backward predictor of Figure 11.2. Namely, the output error $b_m(n)$ is uncorrelated (orthogonal) with the samples $x(n), x(n-1), \ldots, x(n-m+1)$ at the filter (predictor) input.

Property 4 *The backward prediction errors $b_0(n), b_1(n), \ldots$ of a sequence $x(n)$ are always uncorrelated with one another. In other words, for any $k \neq l$,*

$$\mathrm{E}[b_k(n)b_l(n)] = 0. \tag{11.28}$$

To show this, with no loss of generality, we assume that $k < l$ and substitute for $b_k(n)$ from (11.22). This gives

$$\begin{aligned} \mathrm{E}[b_k(n)b_l(n)] &= \mathrm{E}\left[\left(x(n-k) - \sum_{i=1}^{k} a_{k,k+1-i} x(n-i+1)\right) b_l(n)\right] \\ &= \mathrm{E}[x(n-k)b_l(n)] - \sum_{i=0}^{k-1} a_{k,k-i} \mathrm{E}[x(n-i))b_l(n)]. \end{aligned} \tag{11.29}$$

Using Property 3 and noting that $k < l$, we find that all the expectations on the right-hand side of (11.29) are zero. This completes the proof.

11.6 Derivation of the Lattice Structure

In this section we present a derivation of the lattice structure for prediction-error filters. A distinct feature of the lattice structure, as we will show in this section, is that it is a direct implementation of the *order-update equations* for computing the mth order forward and backward prediction errors from the forward and backward prediction errors of order $m-1$. This is not possible in the transversal structure case. To derive these order-update equations and thereby the structure of the lattice filters, we start with the forward prediction error for an $(m+1)$th order predictor:

$$f_{m+1}(n) = x(n) - \sum_{i=1}^{m+1} a_{m+1,i}\, x(n-i). \tag{11.30}$$

The summation on the right-hand side of (11.30) can be rearranged as

$$\sum_{i=1}^{m+1} a_{m+1,i}\, x(n-i) = \sum_{i=1}^{m} a_{m+1,i}\, x(n-i) + a_{m+1,m+1}\, x(n-m-1). \tag{11.31}$$

From (11.22) we get

$$x(n-m-1) = b_m(n-1) + \sum_{i=1}^{m} a_{m,m+1-i}\, x(n-i). \tag{11.32}$$

Substituting (11.32) in (11.31) we obtain

$$\begin{aligned}\sum_{i=1}^{m+1} a_{m+1,i}\, x(n-i) &= \sum_{i=1}^{m} (a_{m+1,i} + a_{m+1,m+1} a_{m,m+1-i})\, x(n-i) + a_{m+1,m+1}\, b_m(n-1) \\ &= \sum_{i=1}^{m} a'_{m,i}\, x(n-i) + \kappa_{m+1}\, b_m(n-1),\end{aligned} \tag{11.33}$$

where

$$\kappa_{m+1} = a_{m+1,m+1} \tag{11.34}$$

and

$$a'_{m,i} = a_{m+1,i} + \kappa_{m+1}\, a_{m,m+1-i}, \quad \text{for } i = 1, 2, \ldots, m. \tag{11.35}$$

The above development shows that any linear combination of the input samples $x(n-1), x(n-2), \ldots, x(n-m-1)$ can also be obtained as a linear combination of $x(n-1), x(n-2), \ldots, x(n-m)$ and $b_m(n-1)$. We also note that the summation on the

left-hand side of (11.33) is the $(m+1)$th order forward prediction of $x(n)$, i.e. $\hat{x}^{\mathrm{f}}_{m+1}(n)$. We may thus argue that the estimate $\hat{x}^{\mathrm{f}}_{m+1}(n)$ can also be obtained as a linear combination of the past m samples of $x(n)$ and the backward prediction error $b_m(n-1)$, i.e.

$$\hat{x}^{\mathrm{f}}_{m+1}(n) = \sum_{i=1}^{m} a'_{m,i}x(n-i) + \kappa_{m+1}b_m(n-1). \tag{11.36}$$

Then, the $a'_{m,i}$ coefficients and κ_{m+1} can be obtained directly by minimizing the mean-square of the estimation error

$$\begin{aligned} f_{m+1}(n) &= x(n) - \hat{x}^{\mathrm{f}}_{m+1}(n) \\ &= x(n) - \sum_{i=1}^{m} a'_{m,i}x(n-i) - \kappa_{m+1}b_m(n-1). \end{aligned} \tag{11.37}$$

To proceed, we define the vectors

$$\mathbf{z}(n) = [\mathbf{x}^{\mathrm{T}}_m(n-1) \quad b_m(n-1)]^{\mathrm{T}} \tag{11.38}$$

and

$$\mathbf{w}_z = [\mathbf{a}'^{\mathrm{T}}_m \quad \kappa_{m+1}]^{\mathrm{T}}, \tag{11.39}$$

where

$$\mathbf{a}'_m = [a'_{m,1} \quad a'_{m,2} \quad \ldots \quad a'_{m,m}]^{\mathrm{T}} \tag{11.40}$$

and $\mathbf{x}_m(n-1)$ is as defined in Section 11.1. Using these definitions, (11.37) can be written as

$$f_{m+1}(n) = x(n) - \mathbf{w}^{\mathrm{T}}_z\mathbf{z}(n). \tag{11.41}$$

Then, the tap-weight vector $\mathbf{w}_z$, which minimizes $f_{m+1}(n)$ in the mean-square sense, can be obtained from the corresponding Wiener–Hopf equation

$$\mathbf{R}_{zz}\mathbf{w}_z = \mathbf{p}_{xz}, \tag{11.42}$$

where

$$\mathbf{R}_{zz} = \mathrm{E}[\mathbf{z}(n)\mathbf{z}^{\mathrm{T}}(n)] \tag{11.43}$$

is the correlation matrix of the observation vector, $\mathbf{z}(n)$, and

$$\mathbf{p}_{xz} = \mathrm{E}[x(n)\mathbf{z}(n)] \tag{11.44}$$

is the cross-correlation between $\mathbf{z}(n)$ and the desired output, $x(n)$.

Substituting (11.38) in (11.43) and using the definition (11.7) and Property 3 of the prediction errors, i.e. (11.27), we obtain

$$\mathbf{R}_{zz} = \begin{bmatrix} r(0) & r(1) & \cdots & r(m-1) & 0 \\ r(1) & r(0) & \cdots & r(m-2) & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r(m-1) & r(m-2) & \cdots & r(0) & 0 \\ 0 & 0 & \cdots & 0 & \mathrm{E}[b_m^2(n-1)] \end{bmatrix}. \tag{11.45}$$

We note that the $m \times m$ portion of the upper-left part of $\mathbf{R}_{zz}$ is nothing but the correlation matrix $\mathbf{R}$ of (11.8). Thus we may write

$$\mathbf{R}_{zz} = \begin{bmatrix} \mathbf{R} & \mathbf{0}_m \\ \mathbf{0}_m^{\mathrm{T}} & \mathrm{E}[b_m^2(n-1)] \end{bmatrix}, \tag{11.46}$$

where $\mathbf{0}_m$ denotes the length m zero column vector. Similarly, it is straightforward to show that

$$\mathbf{p}_{xz} = \begin{bmatrix} \mathbf{r} \\ \mathrm{E}[x(n)b_m(n-1)] \end{bmatrix}, \tag{11.47}$$

where $\mathbf{r}$ is the column vector defined in (11.9).

Substituting (11.39), (11.46) and (11.47) in (11.42) and solving for κ_{m+1} and $\mathbf{a}'_m$, we obtain

$$\kappa_{m+1} = \frac{\mathrm{E}[x(n)b_m(n-1)]}{\mathrm{E}[b_m^2(n-1)]} \tag{11.48}$$

and

$$\mathbf{a}'_m = \mathbf{R}^{-1}\mathbf{r}. \tag{11.49}$$

Comparing (11.49) with (11.5), we find that

$$\mathbf{a}'_m = \mathbf{a}_m. \tag{11.50}$$

Substituting this result in (11.37), we get

$$\begin{aligned} f_{m+1}(n) &= x(n) - \mathbf{a}_m^{\mathrm{T}}\mathbf{x}_m(n-1) - \kappa_{m+1}b_m(n-1) \\ &= f_m(n) - \kappa_{m+1}b_m(n-1), \end{aligned} \tag{11.51}$$

where use is made of (11.21). Thus, the $(m+1)$th order forward prediction error can be obtained from the mth order forward and backward prediction errors.

Following a similar procedure as above, a similar recursion for the backward prediction error can be derived. It is given by (Problem P11.1)

$$b_{m+1}(n) = b_m(n-1) - \kappa'_{m+1}f_m(n), \tag{11.52}$$

where

$$\kappa'_{m+1} = \frac{\mathrm{E}[x(n-m-1)f_m(n)]}{\mathrm{E}[f_m^2(n)]}. \tag{11.53}$$

We now show that the two quantities given for κ_{m+1} in (11.48) and κ'_{m+1} in (11.53) are the same. Consider (11.48). Using (11.21) we can write

$$\begin{aligned}\mathrm{E}[x(n)b_m(n-1)] &= \mathrm{E}\left[\left(f_m(n) + \sum_{i=1}^{m} a_{m,i}x(n-i)\right)b_m(n-1)\right] \\ &= \mathrm{E}[(f_m(n)b_m(n-1)] + \sum_{i=1}^{m} a_{m,i}\mathrm{E}[x(n-i)b_m(n-1)].\end{aligned} \tag{11.54}$$

We note from (11.27), with n replaced by $n-1$, that all the expectations under the summation on the right-hand side of (11.54) are zero. Thus, we obtain

$$\mathrm{E}[x(n)b_m(n-1)] = \mathrm{E}[f_m(n)b_m(n-1)]. \tag{11.55}$$

Similarly, one can easily show that

$$\mathrm{E}[x(n-m-1)f_m(n)] = \mathrm{E}[f_m(n)b_m(n-1)]. \tag{11.56}$$

The results in (11.55) and (11.56) show that the numerators of the two expressions on the right-hand sides of (11.48) and (11.53) are the same. The denominators of these expressions are also the same, since $\mathrm{E}[f_m^2(n)] = P_m^{\mathrm{f}}$, $\mathrm{E}[b_m^2(n-1)] = \mathrm{E}[b_m^2(n)] = P_m^{\mathrm{b}}$, and according to Property 1 of the prediction errors, $P_m^{\mathrm{f}} = P_m^{\mathrm{b}}$. Thus, we have established that the quantities κ_{m+1} in (11.48) and κ'_{m+1} in (11.53) are the same. This, of course, is true only when the predictors' coefficients are optimum.

We may also write

$$\kappa_{m+1} = \frac{\mathrm{E}[f_m(n)b_m(n-1)]}{\sqrt{\mathrm{E}[f_m^2(n)]\mathrm{E}[b_m^2(n-1)]}}. \tag{11.57}$$

Thus, κ_{m+1} is the normalized correlation between the forward and backward errors $f_m(n)$ and $b_m(n-1)$. In fact, κ_{m+1} is known as *the partial correlation* (PARCOR) coefficient since it represents the correlation that remains between the forward and backward prediction errors. Using the Cauchy–Schwartz inequality[1] it is straightforward to show that the following inequality is always true:

$$|\kappa_{m+1}| \leq 1. \tag{11.58}$$

[1] The Cauchy–Schwartz inequality states that for any set of numbers $\{a_i$ and b_i, for $i = 1, 2, \ldots, L\}$,

$$\left|\sum_{i=1}^{L} a_i b_i\right| \leq \sqrt{\left(\sum_{i=1}^{L} |a_i|^2\right)\left(\sum_{i=1}^{L} |b_i|^2\right)}.$$

We may also write

$$\kappa_{m+1} = \frac{\mathrm{E}[f_m(n)b_m(n-1)]}{P_m} \tag{11.59}$$

since $\mathrm{E}[f_m^2(n)] = \mathrm{E}[b_m^2(n-1)] = P_m$, according to Property 1 of the prediction errors.

Summarizing the above derived order-update equations for the prediction errors, we have

$$f_{m+1}(n) = f_m(n) - \kappa_{m+1} b_m(n-1), \tag{11.60}$$

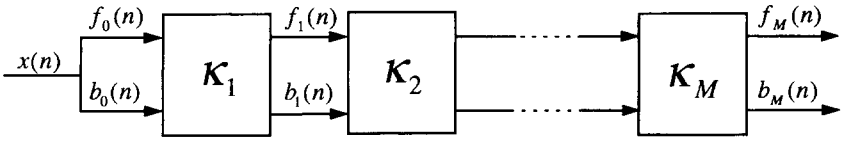
$$b_{m+1}(n) = b_m(n-1) - \kappa_{m+1} f_m(n), \tag{11.61}$$

where $m = 0, 1, 2, \ldots$, and κ_{m+1} is given by (11.57) or (11.59). Since $x(n)$ may be considered as the zeroth order forward or backward prediction errors, the initialization for the above recursions is given by

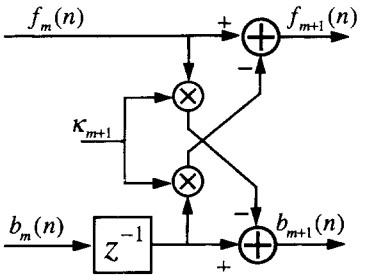
$$f_0(n) = b_0(n) = x(n). \tag{11.62}$$

The structure that implements the above recursions is called the lattice filter/predictor.

Figure 11.5(a) shows the lattice structure of an M-stage forward/backward predictor. Each stage has two inputs. These are the forward and backward prediction errors from the previous stage. The outputs of each stage are the forward and backward prediction errors of one order higher. These are calculated according to the order-update equations (11.60) and (11.61). The two inputs to Stage 1 are common and are equal to the predictor input $x(n)$. Figure 11.5(b) depicts the details of the mth



(a)

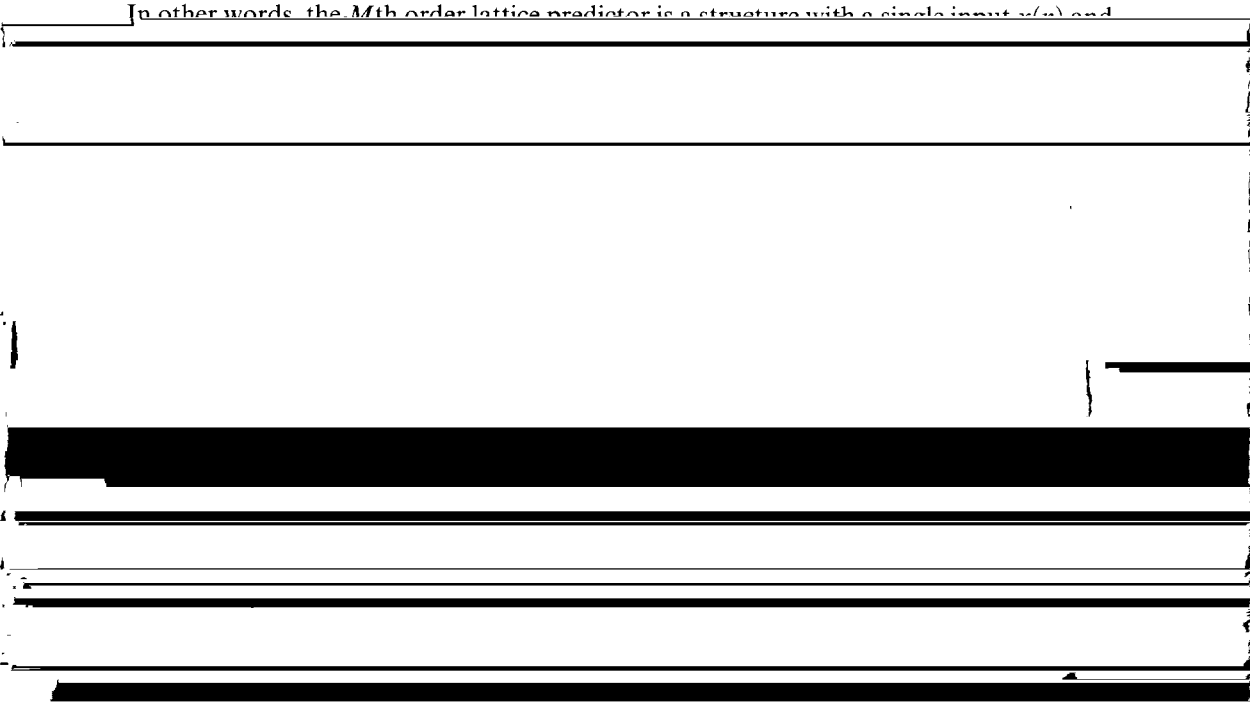


(b)

Figure 11.5 Lattice predictor: (a) overall structure, (b) details of stage m

stage of the lattice predictor. It follows from the order-update equations (11.60) and (11.61).

A special feature of the lattice predictor is that to obtain the Mth order prediction errors, all prediction errors (forward and backward) of lower orders are also calculated. In other words, the Mth order lattice predictor is a structure with a single input $x(n)$ and



$2M + 2$ outputs $f_0(n), b_0(n), f_1(n), b_1(n), \ldots, f_M(n)$ and $b_M(n)$. How many of these outputs will be used is application dependent. For example, in an Mth order forward predictor where the final goal is $f_M(n)$, the rest of the prediction errors (with the exception of $b_M(n)$ which, in this case, can be dropped from the structure) are required only as intermediate signal sequences.

A useful relationship which we shall establish before ending this section is an order-update equation for the mean-square value of the prediction errors. We not that

$$\begin{aligned} P_{m+1} &= \mathrm{E}[f_{m+1}^2(n)] \\ &= \mathrm{E}\left[f_{m+1}(n)\left(x(n) - \sum_{i=1}^{m+1} a_{m+1,i}x(n-i)\right)\right] \\ &= \mathrm{E}[f_{m+1}(n)x(n)] - \sum_{i=1}^{m+1} a_{m+1,i}\mathrm{E}[f_{m+1}(n)x(n-i)]. \end{aligned} \tag{11.63}$$

But from Property 2 of the prediction errors we know that all the expectations under the summation on the right-hand side of (11.63) are zero. Thus we obtain

$$P_{m+1} = \mathrm{E}[f_{m+1}(n)x(n)]. \tag{11.64}$$

Substituting for $f_{m+1}(n)$ and $x(n)$ in (11.64) from (11.60) and (11.21), respectively, we get

$$P_{m+1} = \mathrm{E}\left[(f_m(n) - \kappa_{m+1}b_m(n-1))\left(f_m(n) + \sum_{i=1}^{m} a_{m,i}x(n-i)\right)\right]$$

likely to be relatively larger (in magnitude) for the first few stages and drop to some values close to zero at later stages.

11.7 The Lattice as an Orthogonalization Transform

An important feature of the lattice predictor structure of Figure 11.5(a), in the context of adaptive filters, is that it may be viewed as an orthogonalization transform. Furthermore, as we shall see later, the PARCOR coefficients, which are central to the lattice structure, can be obtained adaptively. So, the lattice predictor structure may be used for adaptive implementation of orthogonalization in the transform domain adaptive filters discussed in Chapter 7.

Before looking at the adaptive techniques for the implementation of such orthogonalization, let us assume that the optimum PARCOR coefficients of the lattice structure are known and the corresponding prediction errors can be calculated. We also define the column vector

$$\mathbf{b}(n) = [b_0(n) \;\; b_1(n) \;\; \ldots \;\; b_{N-1}(n)]^{\mathrm{T}} \tag{11.67}$$

the elements of which are the backward prediction errors of orders 0 to $N-1$. The vector $\mathbf{b}(n)$ may be obtained through the lattice predictor of Figure 11.5(a), with $M = N-1$, or, equivalently, according to the equation

$$\mathbf{b}(n) = \mathbf{L}\mathbf{x}(n), \tag{11.68}$$

where

$$\mathbf{x}(n) = [x(n) \;\; x(n-1) \;\; \ldots \;\; x(n-N+1)]^{\mathrm{T}} \tag{11.69}$$

and

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -a_{1,1} & 1 & 0 & \cdots & 0 & 0 \\ -a_{2,2} & -a_{2,1} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -a_{N-1,N-1} & -a_{N-1,N-2} & -a_{N-1,N-3} & \cdots & -a_{N-1,1} & 1 \end{bmatrix}, \tag{11.70}$$

with $a_{i,j}$ denoting the jth coefficient of the ith order transversal forward predictor. We note that the matrix $\mathbf{L}$ is invertible, since $\det(\mathbf{L}) \neq 0$ (see Problem P11.3). Hence, (11.68) can also be written as

$$\mathbf{x}(n) = \mathbf{L}^{-1}\mathbf{b}(n). \tag{11.71}$$

Figure 11.6 depicts a block schematic obtained from (11.68). It consists of N prediction-error filters of orders 0 to $N-1$, in parallel. Compared with the lattice structure, this suggests a more direct way of converting (transforming) the input vector $\mathbf{x}(n)$ to the backward prediction errors $b_0(n), b_1(n), \ldots, b_{N-1}(n)$. It also resembles the

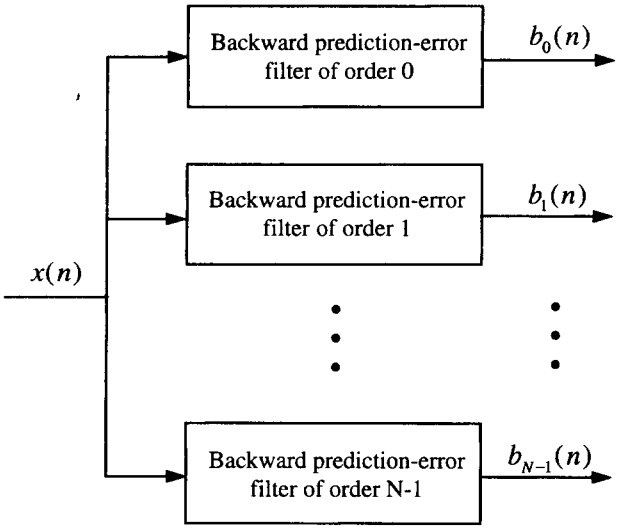


Figure 11.6 Block schematic for a direct implementation of (11.68)

idea of transformation in the context of the transform domain adaptive filters discussed in Chapter 7. However, it requires more computations compared with the lattice predictor. The lattice predictor requires only $2N$ multiplications and $2N$ additions/subtractions for each update of all the backward prediction errors once, while the computational complexity of the direct implementation presented in Figure 11.6 is about $N^2/2$ multiplications and a similar number of additions/subtractions for every input sample. However, in the discussions below, we use equation (11.68) as an expression for the vector $\mathbf{b}(n)$, since it will help in developing certain theoretical results. In actual implementation, of course, one can use the corresponding lattice structure.

11.8 The Lattice Joint Process Estimator

In the previous sections, our discussion of the lattice structure was limited to its use as a prediction-error filter. In this section we show how a general transversal filter, which is used to estimate a desired sequence $d(n)$ from another related sequence $x(n)$, can be implemented using the lattice structure.

Consider a transversal filter with a tap-input vector $\mathbf{x}(n)$, as in (11.69), a tap-weight vector

$$\mathbf{w} = [w_0 \;\; w_1 \;\; \ldots \;\; w_{N-1}]^{\mathrm{T}}, \tag{11.72}$$

output

$$y(n) = \mathbf{w}^{\mathrm{T}}\mathbf{x}(n) \tag{11.73}$$

and desired output $d(n)$. Substituting (11.71) in (11.73) we obtain

$$y(n) = \mathbf{w}^{\mathrm{T}}\mathbf{L}^{-1}\mathbf{b}(n). \tag{11.74}$$

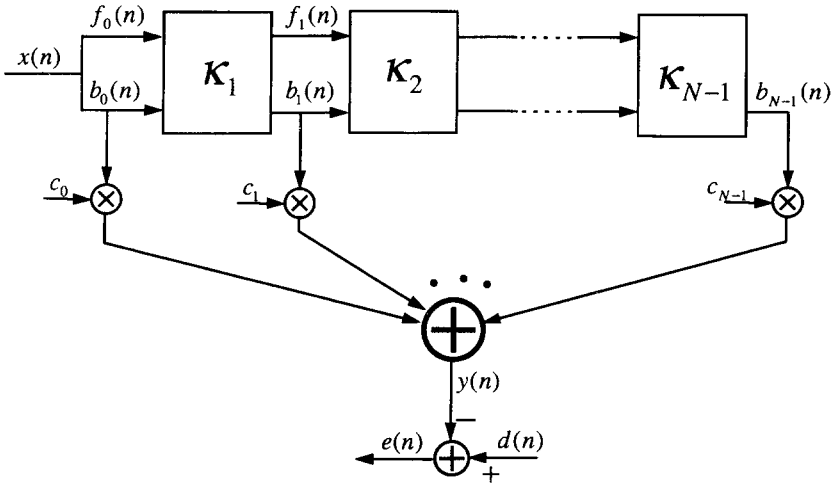


Figure 11.7 The lattice joint process estimator

We define the column vector

$$\mathbf{c} = \mathbf{L}^{-\mathrm{T}}\mathbf{w}, \tag{11.75}$$

where $\mathbf{L}^{-\mathrm{T}}$ is shorthand notation for $(\mathbf{L}^{\mathrm{T}})^{-1}$ or $(\mathbf{L}^{-1})^{\mathrm{T}}$ (note that $(\mathbf{L}^{\mathrm{T}})^{-1} = (\mathbf{L}^{-1})^{\mathrm{T}}$). Then, (11.74) simplifies as

$$y(n) = \mathbf{c}^{\mathrm{T}}\mathbf{b}(n). \tag{11.76}$$

This result shows that the output $y(n)$ of the transversal filter can equivalently be obtained as a linear combination of the backward prediction errors. This also suggests an alternative structure for the implementation of the system function of a transversal filter. Figure 11.7 depicts such an implementation. This is referred to as the *lattice joint process estimator*. It consists of two distinct parts: the lattice predictor and the linear combiner. The lattice predictor is used to convert the samples of input signal to backward prediction errors. The linear combiner uses the backward prediction errors to obtain the filter output according to (11.76).

We note that once the backward predictor coefficients are known, the coefficients of the linear combiner part of the lattice joint process estimator are uniquely determined through (11.75), provided the tap-weight vector $\mathbf{w}$ of the corresponding transversal filter is known. Furthermore, the existence of $\mathbf{c}$ is guaranteed since $\mathbf{L}$ is invertible (see Problem P11.3). The optimum values of the PARCOR coefficients in the lattice part of Figure 11.7 are determined from the statistics of the input sequence, $x(n)$.

11.9 System Functions

In this section we present a system function view of the lattice structure. This will be useful for our analyses in later sections. We define $H_{f_m}(z)$ and $H_{b_m}(z)$ as the system

functions relating the input sequence $x(n)$ and the mth order forward and backward prediction errors $f_m(n)$ and $b_m(n)$, respectively. Then, it follows from (11.60) and (11.61) that

$$H_{f_{m+1}}(z) = H_{f_m}(z) - \kappa_{m+1} z^{-1} H_{b_m}(z) \tag{11.77}$$

and

$$H_{b_{m+1}}(z) = z^{-1} H_{b_m}(z) - \kappa_{m+1} H_{f_m}(z). \tag{11.78}$$

These are order-update equations which may be used to obtain the system functions of forward and backward prediction-error filters of any order in terms of the system functions of one order lower prediction-error filters. The initial conditions to start these order-update equations are $H_{f_0}(z) = H_{b_0}(z) = 1$.

We also note that the system functions $H_{f_m}(z)$ and $H_{b_m}(z)$ may be realized directly using the expressions

$$H_{f_m}(z) = 1 - \sum_{i=1}^{m} a_{m,i} z^{-i} \tag{11.79}$$

and

$$H_{b_m}(z) = z^{-m} - \sum_{i=1}^{m} a_{m,m+1-i} z^{-i+1} \tag{11.80}$$

which follow directly from (11.21) and (11.22), respectively. Also, for future reference, we note that $H_{f_m}(z)$ and $H_{b_m}(z)$ are related according to the equation

$$H_{b_m}(z) = z^{-m} H_{f_m}(z^{-1}). \tag{11.81}$$

11.10 Conversions

From the results in the previous sections, and in particular the system functions presentation in the previous section, we may conclude that there is a close relationship between the PARCOR coefficients, the κ_ms, and the transversal predictor coefficients, the $a_{m,i}$s. In this section we present procedures for conversion between these two sets of coefficients.

11.10.1 Conversion between the lattice and transversal predictors

Given the PARCOR coefficients of a lattice predictor, the coefficients of the corresponding transversal structure can be calculated. This follows from the order-update equations (11.60) and (11.61) or (11.77) and (11.78). It is done by starting with the initial condition (system functions) $H_{f_0}(z) = H_{b_0}(z) = 1$ and iterating (11.77) and (11.78) until the required order is reached. In particular, substituting (11.79) and

(11.80) in (11.77) we get

$$1 - \sum_{i=1}^{m+1} a_{m+1,i} z^{-i} = 1 - \sum_{i=1}^{m} a_{m,i} z^{-i} - \kappa_{m+1} z^{-1} \left(z^{-m} - \sum_{i=1}^{m} a_{m,m+1-i} z^{-i+1} \right). \quad (11.82)$$

Rearranging this, we obtain

$$\sum_{i=1}^{m+1} a_{m+1,i} z^{-i} = \sum_{i=1}^{m} (a_{m,i} - \kappa_{m+1} a_{m,m+1-i}) z^{-i} + \kappa_{m+1} z^{-m-1}. \quad (11.83)$$

Equating the coefficients of similar powers of z on both sides of (11.83), we get

$$a_{m+1,i} = a_{m,i} - \kappa_{m+1} a_{m,m+1-i}, \quad \text{for } i = 1, 2, \ldots, m, \quad (11.84)$$

and

$$a_{m+1,m+1} = \kappa_{m+1}. \quad (11.85)$$

In order to obtain the coefficient of an Mth order transversal predictor, we shall start with the initial condition $a_{0,0} = 1$ (equivalent to $H_{f_0}(z) = H_{b_0}(z) = 1$) and iterate (11.84) and (11.85) M times. Table 11.1 summarizes this procedure.

Next, we derive a procedure for calculating the PARCOR coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$ from the coefficients $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$ of an Mth order transversal predictor. Consider (11.84) for a particular value of i and also when i is replaced by $m + 1 - i$. We get the following pair of simultaneous equations:

$$\begin{aligned} a_{m,i} - \kappa_{m+1} a_{m,m+1-i} &= a_{m+1,i}, \\ a_{m,m+1-i} - \kappa_{m+1} a_{m,i} &= a_{m+1,m+1-i}. \end{aligned} \quad (11.86)$$

Solving these for $a_{m,i}$, we get

$$a_{m,i} = \frac{a_{m+1,i} + \kappa_{m+1} a_{m+1,m+1-i}}{1 - \kappa_{m+1}^2}, \quad \text{for } i = 1, 2, \ldots, m. \quad (11.87)$$

This, with (11.85), suggest the procedure presented in Table 11.2 for calculating the PARCOR coefficients from the coefficients of the corresponding transversal predictor.

Table 11.1 Conversion from the lattice to the transversal predictor

Given:	$\kappa_1, \kappa_2, \ldots, \kappa_M$
Required:	$a_{M,1}, a_{M,2}, \ldots, a_{M,M}$

$a_{1,1} = \kappa_1$
for $m = 1$ to $M - 1$
  $a_{m+1,i} = a_{m,i} - \kappa_{m+1} a_{m,m+1-i}$, for $i = 1, 2, \ldots, m$
  $a_{m+1,m+1} = \kappa_{m+1}$
end

Table 11.2 Conversion from the transversal to the lattice predictor. The inverse Levinson–Durbin algorithm

Given: $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$
Required: $\kappa_1, \kappa_2, \ldots, \kappa_M$

$\kappa_M = a_{M,M}$
for $m = M-1, M-2, \ldots, 1$
$$a_{m,i} = \frac{a_{m+1,i} + \kappa_{m+1} a_{m+1,m+1-i}}{1 - \kappa_{m+1}^2}, \quad \text{for } i = 1, 2, \ldots, m$$
$\kappa_m = a_{m,m}$
end

This procedure is known as the *inverse Levinson–Durbin algorithm*. It may be noted that although we are only interested in the PARCOR coefficients, the coefficients of the transversal filters of orders $m = 1, 2, \ldots, M-1$ are also obtained as intermediate results. Thus, given the coefficients of an Mth order transversal predictor, the coefficients of the lower order transversal predictors are obtained by following the procedure provided in Table 11.2. In other words, given the last row of the matrix $\mathbf{L}$ of (11.70), for $M = N-1$, we can build the whole matrix $\mathbf{L}$ by following the inverse Levinson–Durbin algorithm.

11.10.2 The Levinson–Durbin algorithm

From (11.5) we note that the $a_{m,i}$ coefficients of a transversal predictor are directly related to the autocorrelation function of its input. The well-known Levinson–Durbin algorithm is a computationally efficient procedure for solving the Wiener–Hopf equation (11.5) of the transversal predictor. It also provides the PARCOR coefficients of the corresponding lattice predictor. The efficiency is achieved by exploiting the fact that the input $x(n)$ is a stationary process. This will be clarified further at the end of this subsection. With the background that we have already developed, derivation of the Levinson–Durbin algorithm is straightforward. We note from (11.59) that

$$\begin{aligned}\kappa_{m+1} P_m &= \mathrm{E}[f_m(n) b_m(n-1)] \\ &= \mathrm{E}\left[f_m(n)\left(x(n-m-1) - \sum_{i=1}^{m} a_{m,m+1-i} x(n-i)\right)\right] \\ &= \mathrm{E}[f_m(n) x(n-m-1)], \end{aligned} \tag{11.88}$$

where the last equality follows from the identity $\mathrm{E}[f_m(n)x(n-i)] = 0$, for $i = 1, 2, \ldots, m$. Substituting (11.21) in (11.88) we obtain

$$\begin{aligned}\kappa_{m+1} P_m &= \mathrm{E}\left[\left(x(n) - \sum_{i=1}^{m} a_{m,i} x(n-i)\right) x(n-m-1)\right] \\ &= r(m+1) - \sum_{i=1}^{m} a_{m,i} r(m+1-i) \end{aligned} \tag{11.89}$$

Table 11.3 The Levinson–Durbin algorithm

Given: $r(0), r(1), \ldots, r(M)$
Required: $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$ and $\kappa_1, \kappa_2, \ldots, \kappa_M$

$P_0 = r(0)$
$\kappa_1 = r(1)/P_0$
$a_{1,1} = \kappa_1$
$P_1 = (1 - \kappa_1^2)P_0$
for $m = 1$ to $M - 1$
$\quad \kappa_{m+1} = \dfrac{r(m+1) - \sum_{i=1}^{m} a_{m,i} r(m+1-i)}{P_m}$
$\quad a_{m+1,i} = a_{m,i} - \kappa_{m+1} a_{m,m+1-i}, \quad \text{for } i = 1, 2, \ldots, m$
$\quad a_{m+1,m+1} = \kappa_{m+1}$
$\quad P_{m+1} = (1 - \kappa_{m+1}^2)P_m$
end

or

$$\kappa_{m+1} = \frac{r(m+1) - \sum_{i=1}^{m} a_{m,i} r(m+1-i)}{P_m}. \tag{11.90}$$

Thus, given the autocorrelation coefficients $r(1), r(2), \ldots, r(m+1)$, the mth order transversal predictor coefficients $a_{m,i}$s, and P_m, we can calculate κ_{m+1} according to (11.90). The order-update equations (11.84) are then used to obtain the coefficients of the $(m+1)$th order transversal predictor, the $a_{m+1,i}$s. Equation (11.66) is used to obtain P_{m+1} for the next iteration. Table 11.3 summarizes these results. This is called the *Levinson–Durbin algorithm*. The most important feature of the Levinson–Durbin algorithm is its computational efficiency. Careful examination of Table 11.3 shows that implementation of the Levinson–Durbin algorithm requires about M^2 multiplications/divisions and the same number of additions/subtractions, where M is the order of the predictor. This must be compared with M^3 which is the order of computations required for solving a system of M linear equations without exploiting the structure in the system. The special structure that is exploited here is the symmetric toeplitz nature of the autocorrelation matrix $\mathbf{R}$. By definition, the autocorrelation matrix of any process (stationary or non-stationary) is symmetric. But, if the process $x(n)$ is stationary (at least wide-sense), then the autocorrelation matrix becomes toeplitz, in addition to being symmetric. That is, all the elements along any given diagonal are the same. For example, the k th sub-diagonal and super-diagonal will be constituted by the autocorrelation at lag k. In fact, all the results that we have derived in this chapter are under this stationarity assumption on input $x(n)$.

In the above two subsections we have derived procedures for (i) conversion between the coefficients of the lattice and transversal predictors, and (ii) obtained the coefficients of the lattice and transversal predictors from the autocorrolation values of the input process. Furthermore, given the power of the input sequence $x(n)$, i.e. $P_0 = r(0) = \mathrm{E}[x^2(n)]$, and the PARCOR coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$, we can develop a

procedure to obtain the autocorrelation coefficients $r(1), r(2), \ldots, r(M)$ (Problem P11.6). The latter coefficients can also be obtained if P_0 and the coefficients $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$ of an Mth order transversal predictor are available (Problem P11.6). All these possible conversions show that the three sets of coefficients $(P_0, \kappa_1, \kappa_2, \ldots, \kappa_M)$, $(P_0, a_{M,1}, a_{M,2}, \ldots, a_{M,M})$ and $(r(0), r(1), \ldots, r(M))$ are three different representations of the same information. When M tends to infinity, these may be thought as an alternative representation of the power spectral density of the input process $x(n)$.

11.10.3 Extension of the Levinson–Durbin algorithm

The solution provided by the Levinson–Durbin algorithm is only applicable to the case where the Wiener–Hopf equation to be solved corresponds to a predictor. In this section, the Levinson–Durbin algorithm is extended to the case of the joint process estimator so as to handle the general case of estimating a signal from another related signal. Consider the Wiener–Hopf equation

$$\mathbf{R}\mathbf{w} = \mathbf{p}, \tag{11.91}$$

where $\mathbf{R} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]$, $\mathbf{x}(n)$ is the input vector as defined in (11.69), $\mathbf{p} = \mathrm{E}[\mathbf{x}(n)d(n)]$, and $d(n)$ is the desired output of the estimator.

The solution to (11.91) consists of three steps:

Step 1. The conventional Levinson–Durbin algorithm of Table 11.3 is used to obtain the elements of the matrix $\mathbf{L}$ of (11.70). The PARCOR coefficients $\kappa_1, \kappa_2, \ldots, \kappa_{N-1}$ of the lattice predictor and the mean-square values of the prediction errors, i.e. $P_0, P_1, \ldots, P_{N-1}$, are also obtained in this process.

Step 2. The Wiener–Hopf equation corresponding to the linear combiner part of the lattice joint process estimator is built and solved. This gives the coefficient vector $\mathbf{c}$.

Step 3. The tap-weight vector $\mathbf{w}$ is obtained according to the equation

$$\mathbf{w} = \mathbf{L}^{\mathrm{T}}\mathbf{c}. \tag{11.92}$$

This is obtained by premultiplying (11.75) on both sides by $\mathbf{L}^{\mathrm{T}}$.

The Wiener–Hopf equation for $\mathbf{c}$ is (see Figure 11.7)

$$\mathbf{R}_{bb}\mathbf{c} = \mathbf{p}_{db}, \tag{11.93}$$

where $\mathbf{R}_{bb} = \mathrm{E}[\mathbf{b}(n)\mathbf{b}^{\mathrm{T}}(n)]$ and $\mathbf{p}_{db} = \mathrm{E}[d(n)\mathbf{b}(n)]$. Property 4 of the prediction errors implies that the correlation matrix $\mathbf{R}_{bb}$ is diagonal. Furthermore, the diagonal elements of $\mathbf{R}_{bb}$ are the mean-square values of the backward prediction errors $b_0(n)$, $b_1(n), \ldots, b_{N-1}(n)$, i.e. $P_0, P_1, \ldots, P_{N-1}$. Hence,

$$\mathbf{R}_{bb} = \mathrm{diag}(P_0, P_1, \ldots, P_{N-1}). \tag{11.94}$$

Also, the mth element of $\mathbf{p}_{db}$ is

$$\begin{aligned} p_{db}(m) &= \mathrm{E}[d(n)b_m(n)] \\ &= \mathrm{E}\left[d(n)\left(x(n-m) - \sum_{i=0}^{m-1} a_{m,m-i}x(n-i)\right)\right] \\ &= p(m) - \sum_{i=0}^{m-1} a_{m,m-i}p(i), \end{aligned} \tag{11.95}$$

where $p(i) = \mathrm{E}[d(n)x(n-i)]$ is the ith element of the vector $\mathbf{p}$. Substituting (11.94) and (11.95) in (11.93), we obtain

$$c_m = \begin{cases} \dfrac{p(0)}{P_0}, & m = 0, \\ \dfrac{p(m) - \sum_{i=0}^{m-1} a_{m,m-i}p(i)}{P_m}, & m = 1, 2, \ldots, N-1. \end{cases} \tag{11.96}$$

Table 11.4 summarizes the above results. The recursion for the transversal coefficients $w_{m,i}$ follows from (11.92) (Problem P11.7). The subscript m in $w_{m,i}$ denotes filter order.

Table 11.4 Extended Levinson–Durbin algorithm

Given: $\mathbf{R}$ and $\mathbf{p}$
Required: $\mathbf{w} = \mathbf{R}^{-1}\mathbf{p}$

$P_0 = r(0)$
$c_0 = \dfrac{p(0)}{P_0}$
$w_{0,0} = c_0$
$\kappa_1 = r(1)/P_0$
$a_{1,1} = \kappa_1$
$P_1 = (1 - \kappa_1^2)P_0$
$c_1 = \dfrac{p(1) - a_{1,1}p(0)}{P_1}$
$w_{1,0} = c_0 - a_{1,1}c_1$
$w_{1,1} = c_1$
for $m = 1$ to $N-2$
$\quad \kappa_{m+1} = \dfrac{r(m+1) - \sum_{i=1}^{m} a_{m,i}r(m+1-i)}{P_m}$
$\quad a_{m+1,i} = a_{m,i} - \kappa_{m+1}a_{m,m+1-i}, \quad$ for $i = 1, 2, \ldots, m$
$\quad a_{m+1,m+1} = \kappa_{m+1}$
$\quad P_{m+1} = (1 - \kappa_{m+1}^2)P_m$
$\quad c_{m+1} = \dfrac{p(m+1) - \sum_{i=0}^{m} a_{m+1,m+1-i}p(i)}{P_{m+1}}$
$\quad w_{m+1,i} = w_{m,i} - a_{m+1,m+1-i}c_{m+1} \quad$ for $i = 0, 1, \ldots, m$
$\quad w_{m+1,m+1} = c_{m+1}$
end

Here, the three steps listed above are combined under a single 'for loop'. Careful examination of Table 11.4 shows that the computational complexity of the extended Levinson–Durbin algorithm is about $2N^2$ multiplications/divisions and similar number of additions/subtractions. If the joint process estimator is to be implemented in lattice form, then the computations reduce to $1.5N^2$, since Step 3 of the algorithm can then be ignored.

11.11 All-Pole Lattice Structure

The system functions that we have considered so far are all in the form of all-zero filters. In this section we propose a lattice structure for the implementation of an all-pole filter which is characterized by the system function

$$F(z) = \frac{1}{H_{f_M}(z)} = \frac{1}{1 - \sum_{i=1}^{M} a_{M,i} z^{-i}}. \tag{11.97}$$

This choice of $F(z)$ is not restrictive except that it should be the system function of a stable system. This is the consequence of an important result of the theory of lattice filters which states that a forward prediction-error filter is always minimum phase. This result, proof of which is beyond the scope of our discussion in this book, implies that the zeros of the system function $H_{f_M}(z)$ of any prediction-error filter are all less than one in magnitude. It can also be shown that for any arbitrary minimum phase system function $H_{f_M}(z)$, we can always find a process whose forward prediction-error filter is $H_{f_M}(z)$. Since $H_{f_M}(z)$ is a prediction-error filter, if we excite $F(z) = 1/H_{f_M}(z)$ with the

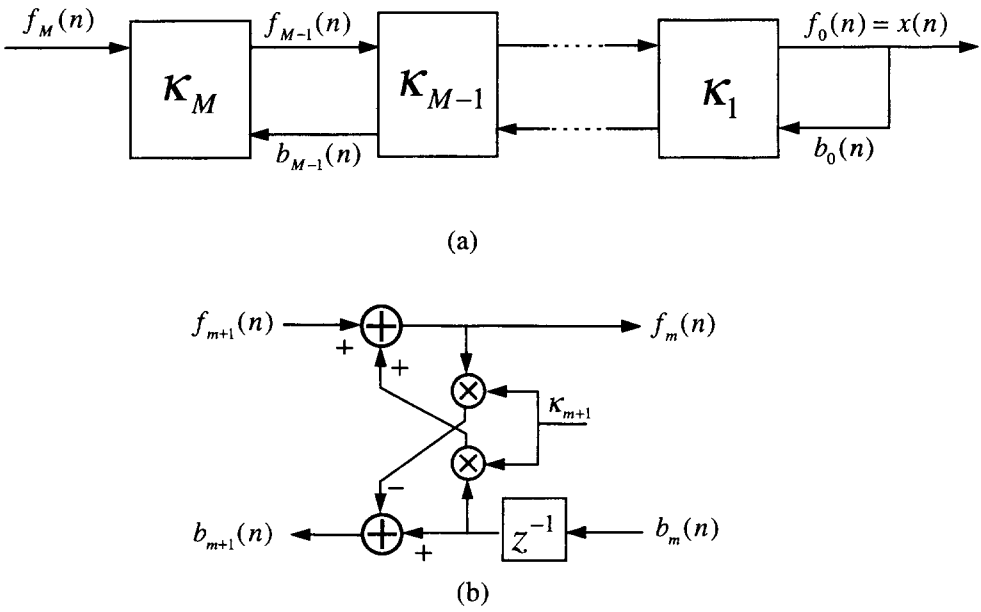


Figure 11.8 Lattice all-pole filter. (a) Overall structure, (b) details of one stage

M th order forward prediction error, $f_M(n)$, of $x(n)$, then the output will be the original process $x(n)$. In other words, the system function that relates $f_M(n)$ and $x(n)$ is $F(z)$. With this in view, we recall the order-update equations (11.60) and (11.61) and rearrange (11.60) as

$$f_m(n) = f_{m+1}(n) + \kappa_{m+1} b_m(n). \tag{11.98}$$

Considering (11.98) and (11.61), for values of $m = 0, 1, \ldots, M-1$, we can suggest the block diagram of Figure 11.8(a). The details of the m th stage of this block diagram are given in Figure 11.8(b). The input sequence in Figure 11.8(a) is $f_M(n)$ and the generated output is $f_0(n) = x(n)$. We also recall that $b_0(n) = x(n)$. From our discussion above, this observation implies that the block diagram of Figure 11.8(a) is the lattice realization of $F(z)$.

We shall comment that although in the development of the lattice structure of the all-pole system function $F(z)$ we used $f_M(n)$ as the input, the choice of the input to the latter structure is not limited to $f_M(n)$. It can be any arbitrary input.

11.12 Pole–Zero Lattice Structure

In this section we extend the all-pole lattice structure of the previous section to an arbitrary system function $G(z)$ with M zeros and M poles. With no loss of generality, we let

$$G(z) = \frac{\sum_{i=0}^{M} w_i z^{-i}}{H_{f_M}(z)}. \tag{11.99}$$

We note that the denominator of $G(z)$ is assumed to be the system function of a prediction-error filter. This, as was noted in the previous section, is not restrictive, since this condition only limits the poles of $G(z)$ to remain within the unit circle in the z-plane. In other words, the condition imposed on $G(z)$ is just to guarantee its stability.

To develop a lattice structure for $G(z)$, we first rearrange (11.75) as

$$\mathbf{w} = \mathbf{L}^{\mathrm{T}}\mathbf{c}. \tag{11.100}$$

Next, we define $\mathbf{z} = [1 \;\; z^{-1} \;\; z^{-2} \;\; \ldots \;\; z^{-(N-1)}]^{\mathrm{T}}$, where z is the z-domain complex variable, multiply the transpose of both sizes of (11.100) from the right by $\mathbf{z}$ and replace $N-1$ by M, to obtain

$$W(z) = \sum_{i=0}^{M} c_i H_{b_i}(z), \tag{11.101}$$

where the c_is are the elements of $\mathbf{c}$,

$$W(z) = \sum_{i=0}^{M} w_i z^{-i}, \tag{11.102}$$

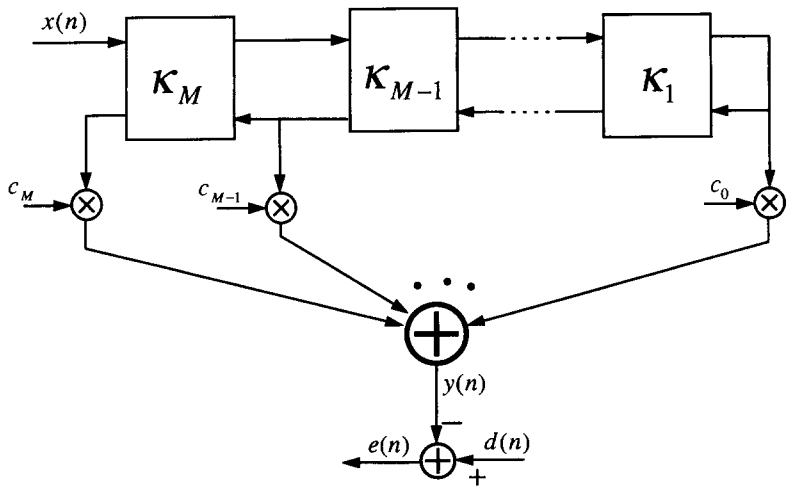


Figure 11.9 Pole–zero lattice for an arbitrary system function

and $H_{b_i}(z)$ is defined as in (11.80). Equation (11.101) shows that any arbitrary order M FIR system function $W(z)$ can equivalently be realized as a linear combination of the backward prediction-error-filter system functions $H_{b_0}(z), H_{b_1}(z), \ldots, H_{b_M}(z)$.

Using (11.101) and (11.102) we obtain

$$G(z) = \sum_{i=0}^{M} c_i \frac{H_{b_i}(z)}{H_{f_M}(z)}. \tag{11.103}$$

Furthermore, with reference to Figure 11.8(a), we note that $H_{b_i}(z)/H_{f_M}(z)$ is the transfer function relating $f_M(n)$ and $b_i(n)$. It is obtained as the cascade of the transfer function between $f_M(n)$ and $x(n)$, i.e. $1/H_{f_M}(z)$, and the transfer function between $x(n)$ and $b_i(n)$, i.e. $H_{b_i}(z)$. Using these results, we obtain Figure 11.9 as the lattice realization of the system function $G(z)$.

11.13 Adaptive Lattice Filter

In this section, an LMS algorithm for the adaptive adjustment of the parameters of the lattice joint process estimator, given in Figure 11.7, is developed. Simulation results and some discussions of the performance of adaptive lattice filters are provided in the next section.

Since the prediction-error power becomes minimum when the predictor coefficients are chosen optimally, the optimum PARCOR coefficient κ_m of the mth stage of a lattice predictor is obtained by minimizing the cost function

$$\xi_{p,m} = \mathrm{E}[f_m^2(n) + b_m^2(n)]. \tag{11.104}$$

The cost function $\xi_{p,m}$ is equivalent to either of the cost functions $P_m^{\mathrm{f}} = \mathrm{E}[f_m^2(n)]$ and $P_m^{\mathrm{b}} = \mathrm{E}[b_m^2(n)]$, since the forward and backward predictors of the same order share the same set of coefficients and also the same level of minimum mean-square error. By defining the cost function as in (11.104), we use both forward and backward prediction errors in the LMS algorithm, so that a lower misadjustment can be achieved.

The LMS algorithm for minimization of the cost function $\xi_{p,m}$ is implemented according to the recursive equation

$$\kappa_m(n+1) = \kappa_m(n) - \mu_{p,m}(n)\frac{\partial \hat{\xi}_{p,m}(n)}{\partial \kappa_m}, \tag{11.105}$$

where $\mu_{p,m}(n)$ is the algorithm step-size and

$$\hat{\xi}_{p,m}(n) = f_m^2(n) + b_m^2(n) \tag{11.106}$$

is an estimate of the cost function $\xi_{p,m}$ based on the most recent samples of the forward and backward prediction errors. Substituting (11.106) in (11.105) and using (11.60) and (11.61), we obtain

$$\kappa_m(n+1) = \kappa_m(n) + 2\mu_{p,m}(n)[f_m(n)b_{m-1}(n-1) + b_m(n)f_{m-1}(n)]. \tag{11.107}$$

To ensure fast convergence of the algorithm, the step-size $\mu_{p,m}(n)$ is normalized by the signal power at the input to the mth stage of the predictor. To estimate this power, we use the recursive equation

$$P_{m-1}(n) = \beta P_{m-1}(n-1) + 0.5(1-\beta)\big(f_{m-1}^2(n) + b_{m-1}^2(n-1)\big). \tag{11.108}$$

The normalized step-size parameter is then given by

$$\mu_{p,m}(n) = \frac{\mu_{\mathrm{p,o}}}{P_{m-1}(n) + \varepsilon}, \tag{11.109}$$

where $\mu_{\mathrm{p,o}}$ is an unnormalized step-size parameter common to all stages of the predictor, and ε is a small positive constant which is added to prevent instability of the algorithm when $P_{m-1}(n)$ assumes values close to zero.

The step-normalized LMS algorithm is also used for the adaptation of the c_i coefficients of the linear combiner part of the lattice joint process estimator. The derivation of this procedure is the same as the recursions developed in Chapter 7. The result is

$$\mathbf{c}(n+1) = \mathbf{c}(n) + 2\boldsymbol{\mu}_{\mathrm{c}} e(n)\mathbf{b}(n), \tag{11.110}$$

where $e(n) = d(n) - y(n)$, $y(n)$ is obtained according to (11.76), and $\boldsymbol{\mu}_{\mathrm{c}}$ is a diagonal matrix consisting the normalized step-size parameters

$$\mu_{\mathrm{c},m} = \frac{\mu_{\mathrm{c,o}}}{P_{m-1}(n) + \varepsilon}, \quad \text{for} \quad m = 0, 1, \ldots, N-1, \tag{11.111}$$

Table 11.5 LMS algorithm for adaptive lattice joint process estimator

Given: Estimator parameters:

$\kappa_1(n), \kappa_2(n), \ldots, \kappa_{N-1}(n)$

and $\mathbf{c}(n) = [c_0(n) \;\; c_1(n) \;\; \ldots \;\; c_{N-1}(n)]^{\mathrm{T}}$,

the most recent input sample $x(n)$, desired output $d(n)$,

backward prediction error vector

$\mathbf{b}(n-1) = [b_0(n-1) \;\; b_1(n-1) \;\; \ldots \;\; b_{N-1}(n-1)]^{\mathrm{T}}$,

and power estimates $P_0(n-1), P_1(n-1), \ldots, P_{N-1}(n-1)$.

Required: Estimator parameter updates:

$\kappa_0(n+1), \kappa_1(n+1), \ldots, \kappa_{N-1}(n+1)$

and $\mathbf{c}(n+1) = [c_0(n+1) \;\; c_1(n+1) \;\; \ldots \;\; c_{N-1}(n+1)]^{\mathrm{T}}$,

backward prediction error vector

$\mathbf{b}(n) = [b_0(n) \;\; b_1(n) \;\; \ldots \;\; b_{N-1}(n)]^{\mathrm{T}}$,

and power estimates $P_0(n), P_1(n), \ldots, P_{N-1}(n)$.

Lattice Predictor Part

$f_0(n) = b_0(n) = x(n)$

$P_0(n) = \beta P_0(n-1) + 0.5(1-\beta)[f_0^2(n) + b_0^2(n-1)]$

for $m = 1$ to $N-1$

$f_m(n) = f_{m-1}(n) - \kappa_m(n) b_{m-1}(n-1)$

$b_m(n) = b_{m-1}(n-1) - \kappa_m(n) f_{m-1}(n)$

$$\kappa_m(n+1) = \kappa_m(n) + \frac{2\mu_{\mathrm{p,o}}}{P_{m-1}(n) + \varepsilon}[f_{m-1}(n) b_m(n) + b_{m-1}(n-1) f_m(n)]$$

$P_m(n) = \beta P_m(n-1) + 0.5(1-\beta)[f_m^2(n) + b_m^2(n-1)]$

end

Linear Combiner Part

$y(n) = \mathbf{c}^{\mathrm{T}}(n)\mathbf{b}(n)$

$e(n) = d(n) - y(n)$

$\boldsymbol{\mu}_{\mathrm{c}} = \mu_{\mathrm{c,o}} \operatorname{diag}((P_0(n) + \varepsilon)^{-1}, (P_1(n) + \varepsilon)^{-1}, \ldots, (P_{N-1}(n) + \varepsilon)^{-1})$

$\mathbf{c}(n+1) = \mathbf{c}(n) + 2\boldsymbol{\mu}_{\mathrm{c}} e(n)\mathbf{b}(n)$

where $\mu_{\mathrm{c,o}}$ is an unnormalized step-size which, in general, may be different from $\mu_{\mathrm{p,o}}$. Table 11.5 summarizes the above results.

11.13.1 Discussion and simulations

Analysis of the convergence behaviour of the LMS algorithm when applied to a lattice structure is rather difficult. In particular, in the case of the lattice joint process estimator there are two sets of parameters that are being adapted simultaneously. The optimum values of the PARCOR coefficients, the κ_ms, depend only on the statistics of the input signal. The optimum value of the coefficient vector $\mathbf{c}$ of the linear combiner part depends on the current values of the PARCOR coefficients as well as the optimum value of the coefficient vector $\mathbf{w}$ in the original transversal filter, $\mathbf{w}_{\mathrm{o}}$, namely equation (11.75). An important point to be noted is that even if $\mathbf{w}_{\mathrm{o}}$, i.e. the optimum impulse response to be

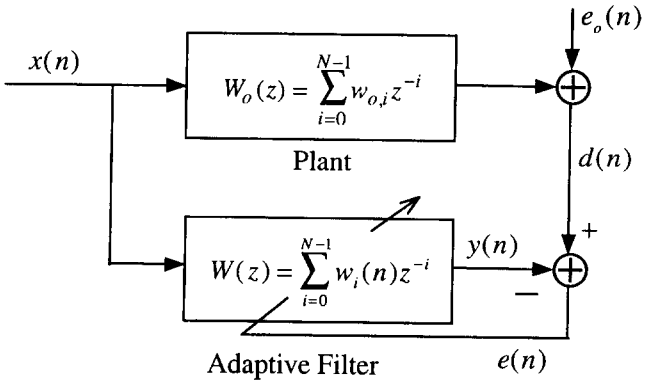


Figure 11.10 A modelling problem

realized by the estimator, is fixed, any change in the PARCOR coefficients will require readjustment of the coefficient vector **c**. This, as we will demonstrate by a simulation example, may lead to a significant increase in misadjustment.

To demonstrate the above phenomenon, we consider the modelling problem shown in Figure 11.10. A plant $W_o(z)$ is to be modelled by an adaptive filter $W(z)$. The common input signal to the plant and adaptive filter, $x(n)$, is generated by passing a unit variance white Gaussian process, $\nu(n)$, through a colouring filter with the system function

$$H_1(z) = K_1 \frac{1 - 1.2z^{-1}}{1 - 1.2z^{-1} + 0.8z^{-2}}. \tag{11.112}$$

The coefficient K_1 is set equal to 0.488. This results in a sequence with a unit variance. For future use, the coloured process generated by $H_1(z)$ will be called $x_1(n)$. Figure 11.11 shows the power spectral density of $x_1(n)$ evaluated using

$$\Phi_{x_1 x_1}(e^{j\omega}) = \Phi_{\nu\nu}(e^{j\omega})|H_1(e^{j\omega})|^2, \tag{11.113}$$

with $\Phi_{\nu\nu}(e^{j\omega}) = 1$. Observe that $x_1(n)$ is highly coloured and the eigenvalue spread of its corresponding correlation matrix can be as large as 338. This is obtained as the ratio of the maximum to the minimum value of the spectral density; see Chapter 4.

In this simulation example we consider realizing the adaptive filter $W(z)$ in transversal as well as lattice (i.e. the lattice joint process estimator of Figure 11.7) forms. The plant and the adaptive filter are both selected to have a length of $N = 30$. This choice of N for the present input sequence results in an eigenvalue spread of 300. The variance of the additive white noise sequence $e_o(n)$ at the plant output is set equal to $\sigma_{e_o}^2 = 10^{-4}$. So, when the adaptive filter $W(z)$ is set to its optimum choice, $W_o(z)$, the resulting minimum mean-square error will be $\sigma_{e_o}^2 = 10^{-4}$.

Figure 11.12 presents a pair of learning curves for the modelling problem. The curves correspond to the transversal and the lattice LMS, and each is an ensemble average of 50 independent runs. The final curves have been smoothed. In the case of the transversal LMS, the step-size parameter, μ, is selected according to (6.63) to result in a 10%

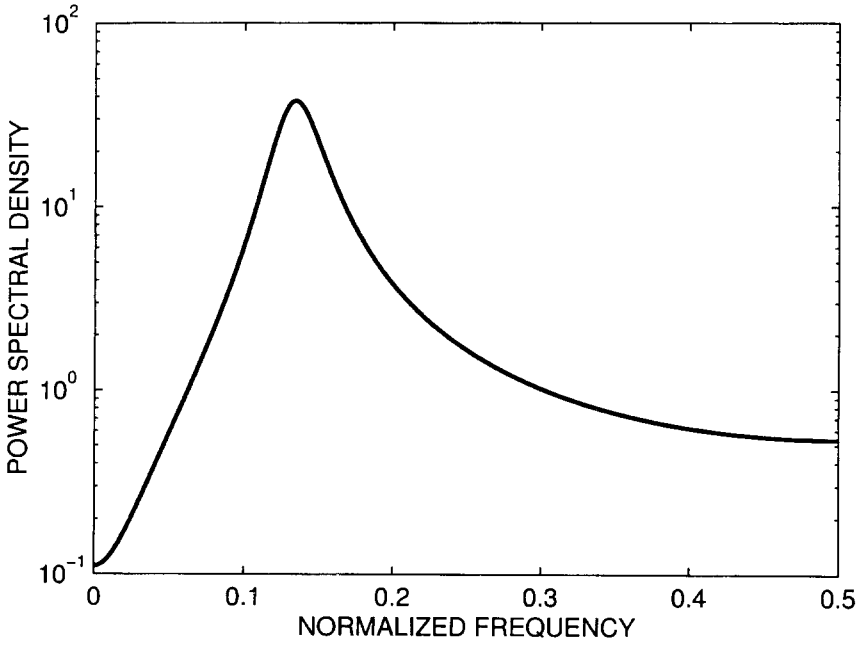


Figure 11.11 Power spectral density of the input process for the modelling problem simulations

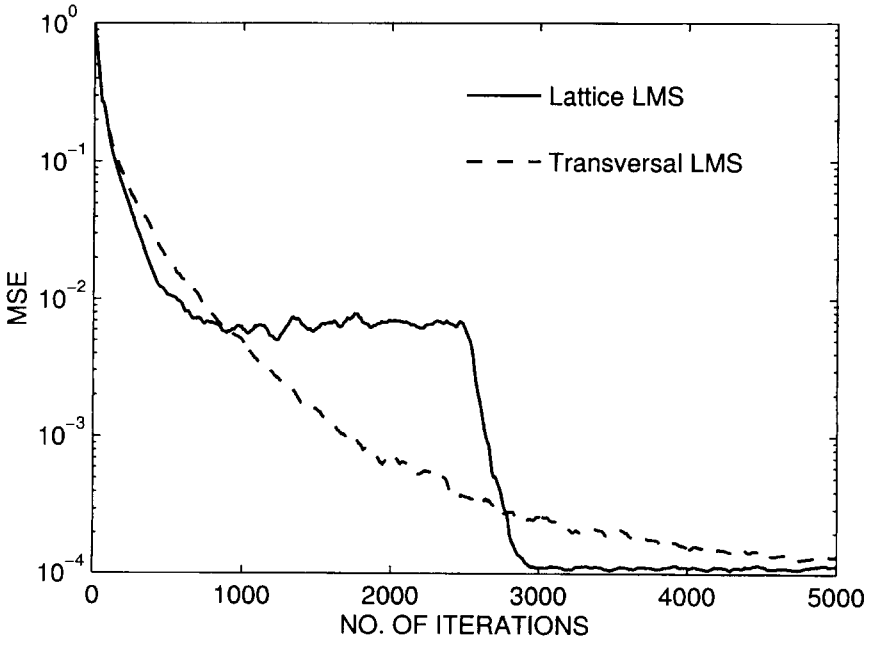


Figure 11.12 Learning curves showing convergence of the transversal and lattice LMS applied to the modelling problem of Figure 11.10

misadjustment. For the lattice LMS, the following parameters are used: $\varepsilon = 0.02$, $\mu_{p,o} = 0.001$ (for the first 2500 iterations only) and $\mu_{c,o} = 0.1/N = 0.0033$. This choice of $\mu_{c,o}$ would result in a misadjustment of about 10% if perturbation of the PARCOR coefficients, which arise due to their adaptation, could be ignored. To demonstrate the effect of the perturbation of the PARCOR coefficients, $\mu_{p,o}$ is forced to zero after the first 2500 iterations, so that the PARCOR coefficients remain fixed from iteration 2500. Observe that perturbation of the PARCOR coefficients has a significant impact on the misadjustment of the lattice LMS algorithm. Once adjustment of the PARCOR coefficients is stopped, we see a fast convergence of the algorithm with a misadjustment close to what we predicted before. At iteration 2500, the PARCOR coefficients are already near their optimal values and the backward prediction errors are almost uncorrelated with one another. This is why the lattice LMS converges faster than the transversal LMS, after iteration 2500.

The problem of perturbation of the PARCOR coefficients is a serious one which limits the application of the adaptive lattice joint process estimator. As the above example demonstrated, unless adjustment of the PARCOR coefficients is stopped after some initial convergence, the lattice LMS cannot be relied on as a good choice for improving the convergence performance of adaptive filters. The large misadjustment arising from adaptation of the PARCOR coefficients prohibits their applicability. The problem may be more serious when the input signal is non-stationary. In that case, the optimum PARCOR coefficients are time-varying, since they follow the time-varying statistics of the input. This in turn necessitates continuous adaptation of the PARCOR coefficients as well as the coefficient vector $\mathbf{c}$. The inevitable lag in the adaptation of $\mathbf{c}$ will result in a further increase in the mean-square error. This, in effect, means higher misadjustment.

11.14 Autoregressive Modelling of Random Processes

A random process $x(n)$ is said to be autoregressive (AR) of order M if it can be generated through a difference equation of the form

$$x(n) = \sum_{i=1}^{M} h_i x(n-i) + \nu(n), \tag{11.114}$$

where the h_is are AR coefficients and $\nu(n)$ is a zero-mean white noise process called the *innovation* of $x(n)$. This implies that any new sample of the process, $x(n)$, is related to its previous M samples according to the summation on the right-hand side of (11.114). In addition, there is a new piece of information (namely, the innovation $\nu(n)$) in $x(n)$ that is uncorrelated to its previous samples. This, in turn, implies that the best linear prediction of $x(n)$ based on its past M samples is nothing but the summation on the right-hand side of (11.114). Moreover, the latter estimate cannot be improved by increasing the order of the predictor beyond M, since the portion of $x(n)$ which could not be estimated by the latter summation, i.e. $\nu(n)$, has no correlation with previous samples of $x(n)$ which include its farther samples $x(n-M-1)$, $x(n-M-2), \ldots$. A procedure for the analytical derivation of these results is discussed in Problem P11.21.

An AR process $x(n)$ can be characterized by its model, which may be obtained by passing $x(n)$ through a forward (or backward) linear predictor and optimizing the

predictor coefficients by minimizing the mean-square error of its output. This results in a set of predictor coefficients that match the coefficients h_i of (11.114). In particular, if a predictor of order $M' \geq M$ is used, then we obtain

$$a_{M',i} = \begin{cases} h_i, & 1 \leq i \leq M, \\ 0, & M+1 \leq i \leq M', \end{cases} \tag{11.115}$$

and

$$P_{M'} = P_M = \sigma_\nu^2, \tag{11.116}$$

where σ_ν^2 is the variance of $\nu(n)$. An important point to be noted here is that the set of coefficients $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$ and P_M provide sufficient information to obtain the autocorrelation function of $x(n)$ for any arbitrary lag. This directly follows from (11.114) with $h_i = a_{M,i}$. To see this, multiplying (11.114) on both sides by $x(n-k)$ and taking expectations, we obtain

$$r(k) = \sum_{i=1}^{M} a_{M,i} r(k-i), \quad k > 0. \tag{11.117}$$

The value of $r(0)$ can be obtained using (11.66) as

$$r(0) = P_0 = \left(\prod_{i=1}^{M} (1 - \kappa_i^2) \right)^{-1} P_M, \tag{11.118}$$

where the PARCOR coefficients, the κ_is, can be obtained using the inverse Levinson–Durbin algorithm discussed in Section 11.10. Moreover, the values of $r(1), r(2), \ldots, r(m)$ are obtained by following either of the procedures discussed in Problem P11.6.

It may also be noted that the estimated AR coefficients may be used to obtain the power spectral density of $x(n)$ according to the equation

$$\Phi_{xx}(e^{j\omega}) = P_M |H_{AR}(e^{j\omega})|^2, \tag{11.119}$$

where it has been noted that $\Phi_{\nu\nu}(e^{j\omega}) = \sigma_\nu^2 = P_M$, and

$$H_{AR}(z) = \frac{1}{1 - \sum_{i=1}^{M} a_{M,i} z^{-i}}. \tag{11.120}$$

It is also instructive to note that the process $x(n)$ can be reconstructed by passing its innovation $\nu(n)$ through $H_{AR}(z)$.

Although many of the processes that arise in practice may not be truly AR, AR modelling of arbitrary processes for the purpose of spectral estimation has been found to be quite effective, provided that a sufficiently large order is considered. Usually, we find that a model order in the range 5 to 10 is more than sufficient to obtain an acceptable estimate of the power spectral density of most of the processes encountered in practice.

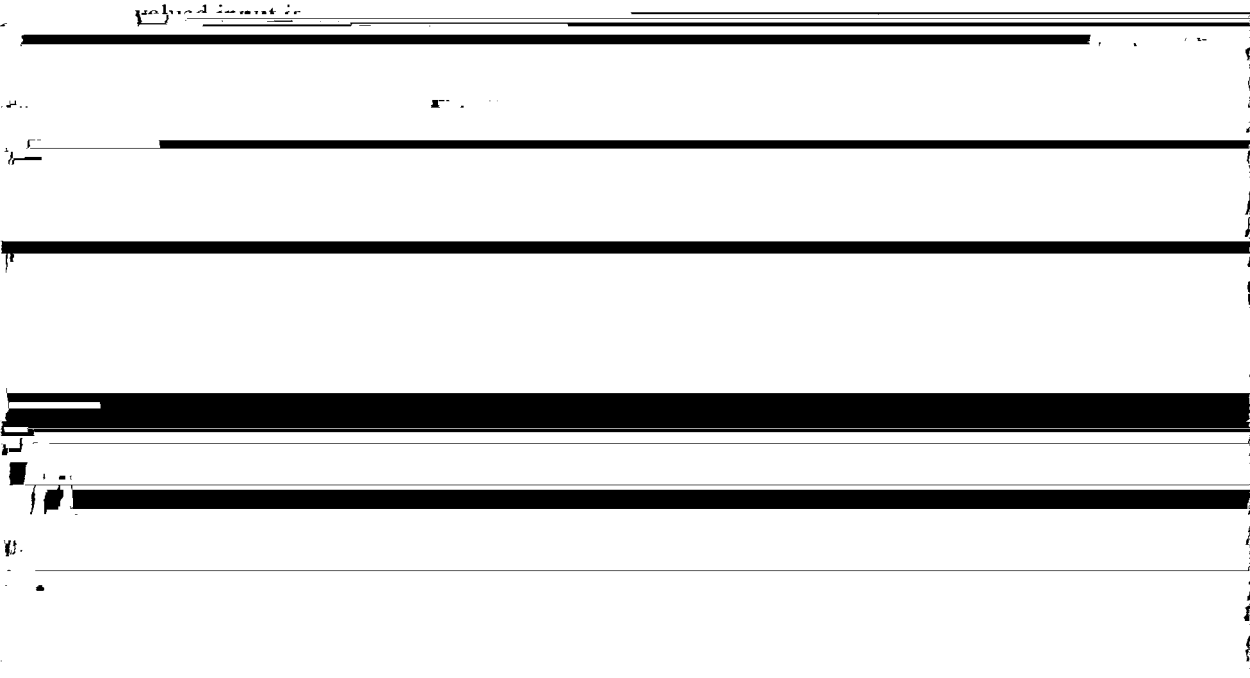
In the context of adaptive filters, the above results have the following implication. *The correlation matrix of the input process to an adaptive transversal filter may be*

characterized by an AR model whose order may be much less than the order of the adaptive filter. This, as we shall see in the next section, may effectively be used to improve the performance of adaptive filters, at very little computational cost.

11.15 Adaptive Algorithms Based on Autoregressive Modelling

The LMS–Newton algorithm was introduced in Chapter 7 as a method to solve the eigenvalue spread problem of adaptive filters whose inputs were coloured. In this section, the results of the previous sections are used to propose two efficient implementations of the LMS–Newton algorithm.[2] We assume that the input sequence to the adaptive filter can be modelled as an AR process whose order may be kept much lower than the adaptive filter length. The two implementations (referred to as Algorithm 1 and Algorithm 2) differ in their structural complexity. The first algorithm, which will be an exact implementation of the LMS–Newton algorithm, if the AR modelling assumption is accurate, is structurally complicated and fits best into a DSP-based implementation. On the other hand, the second algorithm is structurally simple and is tailored more towards VLSI custom chip design.

We recall that the LMS–Newton algorithm recursion for an adaptive filter with real-valued input is



where $\mathbf{w}(n) = [w_0(n)\ w_1(n)\ \ldots\ w_{N-1}]^T$ is the filter tap-weight vector, $\mathbf{x}(n) = [x(n)\ x(n-1)\ \ldots\ x(n-N+1)]^T$ is the filter input vector, $\hat{\mathbf{R}}_{xx}$ is an estimate of the input correlation matrix $\mathbf{R}_{xx} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^T(n)]$, μ is the algorithm step-size parameter, $e(n) = d(n) - y(n)$ is the measured error at the filter output, $d(n)$ is the desired output and $y(n) = \mathbf{w}^T(n)\mathbf{x}(n)$ is the filter output.

It should be noted that here we have added the subscript 'xx' to $\mathbf{R}_{xx}$ to emphasize that it corresponds to the input vector $\mathbf{x}(n)$. We follow this notation in the rest of this chapter, since we need to refer to a number of different correlation matrices.

To implement the LMS–Newton algorithm, we need to calculate $\hat{\mathbf{R}}_{xx}^{-1}\mathbf{x}(n)$ for

Inverting both sides of (11.122) and pre- and postmultiplying the result by $\mathbf{L}^T$ and $\mathbf{L}$, respectively, we obtain

$$\mathbf{R}_{xx}^{-1} = \mathbf{L}^T \mathbf{R}_{bb}^{-1} \mathbf{L}. \tag{11.123}$$

Next, we define $\mathbf{u}(n) = \mathbf{R}_{xx}^{-1}\mathbf{x}(n)$ and substitute for $\mathbf{R}_{xx}^{-1}$ from (11.123) to obtain

$$\begin{aligned} \mathbf{u}(n) &= \mathbf{L}^T \mathbf{R}_{bb}^{-1} \mathbf{L}\mathbf{x}(n) \\ &= \mathbf{L}^T \mathbf{R}_{bb}^{-1} \mathbf{b}(n). \end{aligned} \tag{11.124}$$

This result is fundamental to the derivation of the algorithms that follow.

In the rest of this section, for the sake of convenience, we shall use the notation $\mathbf{u}(n)$ even when $\mathbf{R}_{xx}$ is replaced by its estimate, $\hat{\mathbf{R}}_{xx}$.

11.15.1 Algorithms

Algorithm 1 Implementation of (11.124) requires a mechanism for converting the vector of input samples, $\mathbf{x}(n)$, to the vector of backward prediction-error samples, $\mathbf{b}(n)$. A lattice predictor may be used for the efficient implementation of this mechanism. Moreover, if we assume that the input sequence, $x(n)$, can be modelled as an AR process of order $M < N$, then a lattice predictor with order M will suffice, and the matrix $\mathbf{L}$ and vector $\mathbf{b}(n)$ take the following forms:

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ -a_{1,1} & 1 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -a_{M,M} & -a_{M,M-1} & \cdots & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & -a_{M,M} & \cdots & -a_{M,1} & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & -a_{M,M} & -a_{M,M-1} & \cdots & 1 \end{bmatrix} \tag{11.125}$$

and

$$\mathbf{b}(n) = [b_0(n) \;\; b_1(n) \;\; \ldots \;\; b_M(n) \;\; b_M(n-1) \;\; \ldots \;\; b_M(n-N+M+1)]^T. \tag{11.126}$$

The special structure in rows $M+1$ to N of $\mathbf{L}$ and elements $M+1$ to N of $\mathbf{b}(n)$ follows from (11.115).

In certain applications, such as acoustic echo cancellation, a value of M much smaller than N may be used. In such cases the computational burden of updating $\mathbf{b}(n)$ would be negligible when compared with the total computational complexity of the whole system, as only the first $M+1$ samples of $\mathbf{b}(n)$ require updating. The rest of the elements of $\mathbf{b}(n)$ are delayed versions of $b_M(n)$. Multiplication of $\mathbf{R}_{bb}^{-1}$ by $\mathbf{b}(n)$ (according to (11.124)) also requires only a small amount of computation. It involves estimation of the powers of $b_0(n)$ through $b_M(n)$ and normalization of these samples by their power estimates.

Multiplication of $\mathbf{L}^T$ by $\mathbf{R}_{bb}^{-1}\mathbf{b}(n)$, to complete the computation of $\mathbf{u}(n)$ (according to (11.124)), however, is more involved, since a structure such as a lattice is not applicable. It requires estimation of the elements of $\mathbf{L}$ and direct multiplication of $\mathbf{L}^T$ by $\mathbf{R}_{bb}^{-1}\mathbf{b}(n)$. Considering the forms of $\mathbf{L}$ and $\mathbf{b}(n)$, we find that only the first $M+1$ and the last M elements of $\mathbf{L}^T\mathbf{R}_{bb}^{-1}\mathbf{b}(n)$ need to be computed. The remaining elements of $\mathbf{L}^T\mathbf{R}_{bb}^{-1}\mathbf{b}(n)$ are delayed versions of its $(M+1)$th element.

The following procedure may be used for estimating the elements of $\mathbf{L}$, i.e. the coefficients of the predictors of orders 1 to M. An LMS-based adaptive lattice predictor is used to obtain the PARCOR coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$ of $x(n)$. The conversion algorithm of Table 11.1 is then used to obtain the predictor coefficients of orders 1 to M. Table 11.6 summarizes this procedure.

We have the following comments regarding the algorithm presented in Table 11.6. The role of the constant ε in the PARCOR updating equation is to ensure stability of the algorithm when $P_m(n)$ drops to very small values. Also, at every iteration, the PARCOR coefficients are constrained to lie within a maximum magnitude α. The predictor coefficients, the $a_{m,j}$s, and the backward errors obtained through the lattice predictor are used to update the first $M+1$ and the last M elements of the vector $\mathbf{u}(n)$. The iteration suggested by (11.66) is used to obtain the estimates of the backward error powers. These are denoted as the $\bar{P}_m(n)$s in Table 11.6. The power estimates obtained in the lattice predictor part of the algorithm, i.e. the $P_m(n)$s, could also be used. However, experiments have shown that the use of the $\bar{P}_m(n)$s results in a more reliable algorithm. The vectors $\mathbf{b}_h(n)$ and $\mathbf{b}_t(n)$ denote the backward error vectors which correspond to the input samples at the head and the end tail parts, respectively, of the tap-delay-line filter. When the input signal to the filter is stationary, the elements of $\mathbf{b}_t(n)$ can be obtained by delaying the output $b_M(n)$ of the lattice predictor at the head of $\mathbf{x}(n)$. This has been our assumption in Table 11.6. When the filter input is non-stationary and the filter length, N, is large, we may have to use a separate predictor for the samples at the tail of $\mathbf{x}(n)$.

Algorithm 2 Algorithm 1, although low in computational complexity, is structurally complicated, since the implementation of the Levinson–Durbin algorithm and ordering of the manipulated data is not straightforward. This would not be much of a problem if a DSP processor were used. Therefore, Algorithm 1 is suitable for software implementation. However, if we are interested in a custom chip implementation, we should use Algorithm 2, proposed below, which has a much simpler structure compared with Algorithm 1.

The reason why Algorithm 1 is not simple is because it needs to update the parameters of the lattice and transversal predictors of orders 1 to M at each time instant. Furthermore, only the middle samples in $\mathbf{u}(n)$ could be obtained as delayed versions of earlier samples. In Algorithm 2, we overcome these problems by extending the input and tap-weight vectors, $\mathbf{x}(n)$ and $\mathbf{w}(n)$, to the vectors

$$\mathbf{x}_E(n) = [x(n+M)\ldots x(n+1)\ \ x(n)\ldots x(n-N+1)\ldots x(n-N-M+1)]^T$$

and

$$\mathbf{w}_E(n) = [w_{-M}(n)\ldots w_{-1}(n)\ \ w_0(n)\ldots w_{N-1}(n)\ldots w_{N+M-1}(n)]^T,$$

respectively, and applying an LMS–Newton algorithm similar to (11.121) for updating $\mathbf{w}_E(n)$. Since the tap weights of the original filter correspond to $w_0(n)$ through $w_{N-1}(n)$, the first M and last M elements of $\mathbf{w}_E(n)$ may be frozen at zero. This can easily be done by initializing these weights to zero and assigning a zero step-size parameter to all of them. If this is done, then the computation of the first M and last M elements of $\mathbf{L}^T\mathbf{R}_{bb}^{-1}\mathbf{L}\mathbf{x}_E(n)$ (with appropriate dimensions for $\mathbf{L}$ and $\mathbf{R}_{bb}$) is immaterial and may be ignored. This results in the following recursive equation for updating the adaptive filter tap weights:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{u}_a(n), \tag{11.127}$$

where $\mathbf{w}(n)$ is the filter tap-weight vector as defined above, and

$$\mathbf{u}_a(n) = \mathbf{L}_2\mathbf{R}_{bb}^{-1}\mathbf{L}_1\mathbf{x}_E(n). \tag{11.128}$$

Here, $\mathbf{R}_{bb}$ is a diagonal matrix compatible with the column vector $\mathbf{L}_1\mathbf{x}_E(n)$ and the diagonal elements of $\mathbf{R}_{bb}$ are estimates of the powers of the elements of the latter vector. The matrices $\mathbf{L}_1$ and $\mathbf{L}_2$ are given by

$$\mathbf{L}_1 = \begin{bmatrix} -a_{M,M} & -a_{M,M-1} & \cdots & 1 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & -a_{M,M} & \cdots & -a_{M,1} & 1 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & -a_{M,M} & \cdots & -a_{M,1} & 1 \end{bmatrix} \tag{11.129}$$

and

$$\mathbf{L}_2 = \begin{bmatrix} 1 & -a_{M,1} & \cdots & -a_{M,M} & 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & -a_{M,M-1} & -a_{M,M} & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 1 & \cdots & -a_{M,M-1} & -a_{M,M} \end{bmatrix}. \tag{11.130}$$

The dimensions of $\mathbf{L}_1$ and $\mathbf{L}_2$ are $(N+M)\times(N+2M)$ and $N\times(N+M)$, respectively. The number of rows in $\mathbf{L}_1$ is only N since we do not want to compute the first M and last M elements of $\mathbf{L}^T\mathbf{R}_{bb}^{-1}\mathbf{L}\mathbf{x}_E(n)$.

Inspection of (11.128) reveals that each updating of $\mathbf{u}_a(n)$ requires only updating the first element of the vector $\mathbf{R}_{bb}^{-1}\mathbf{L}_1\mathbf{x}_E(n)$, and then the first element of the final result, $\mathbf{u}_a(n)$. The rest of the elements of the two vectors are delayed versions of their first elements. Putting these together, Figure 11.13 depicts a complete structure of Algorithm 2. It consists of a backward prediction-error filter $H_{b_M}(z)$ whose coefficients, the $a_{M,i}(n)$s, are updated using an adaptive algorithm. The time index 'n' is added to these coefficients to emphasize their variability in time and their adaptation as input statistics may change. Any adaptive algorithm may be used for the adjustment of these coefficients. The successive output samples from the backward prediction-error filter, i.e. $b_M(n+M)$, make the elements of the column vector $\mathbf{L}_1\mathbf{x}_E(n)$. Multiplication of $b_M(n+M)$ by the

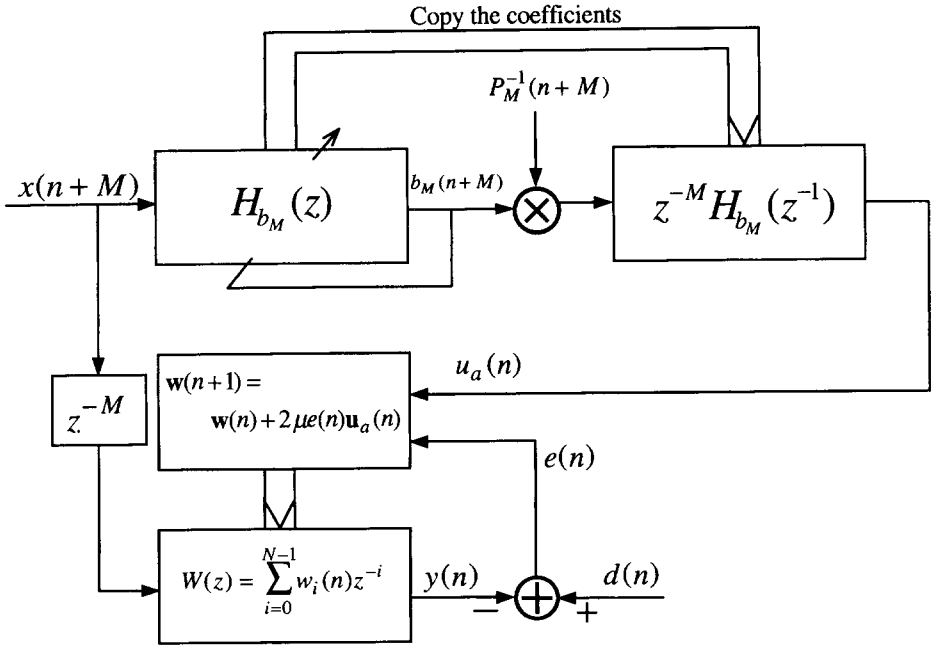


Figure 11.13 Block diagram depicting Algorithm 2. Reprinted from Farhang-Boroujeny (1997b)

inverse of an estimate of its power, denoted as $P_M^{-1}(n+M)$ in Figure 11.13, gives an update of $\mathbf{R}_{bb}^{-1}\mathbf{L}_1\mathbf{x}_E(n)$. Finally, filtering of the latter result by the next filter, whose coefficients are duplicates of those of the backward prediction-error filter in reverse order, provides the samples of the sequence $u_a(n)$, i.e. the elements of the vector $\mathbf{u}_a(n)$. It is also instructive to note that according to (11.81), the latter is nothing but the forward equivalent of the backward prediction-error filter $H_{b_M}(z)$.

We may note that the filter output, $y(n)$, is obtained at the time when $x(n+M)$ is available at the input of Figure 11.13. This is equivalent to saying that there is a delay of M samples at the filter output as compared with the reference input. Although this delay could easily be prevented by shifting the delay box, z^{-M}, from the filter input to its output, we avoid this here to keep the analysis given in the next section as simple as possible. Shifting the delay box to the filter output introduces a delay into the adjustment loop of the filter. The result would then be a delayed LMS algorithm which is known to be inferior to its non-delayed version (see Long, Ling and Proakis (1989)). However, in the cases of interest, when $M \ll N$, the difference between the two algorithms is negligible.

Table 11.7 presents an implementation of Algorithm 2 which follows Figure 11.13 closely, except that we assume the input to the backward prediction-error filter to be $x(n)$ instead of $x(n+M)$. In this implementation the backward prediction-error filter $H_{b_M}(z)$ is implemented in lattice form. Note also that the power normalization factor $P_M^{-1}(n+M)$ is shifted to the output of the filter $z^{-M}H_{b_M}(z^{-1})$. Experiments have shown that this amendment results in a more reliable algorithm.

Table 11.7 An LMS-based procedure for implementation of Algorithm 2

Given: Coefficient vectors $\boldsymbol{\kappa}(n) = [\kappa_1(n) \ \kappa_2(n) \ \ldots \ \kappa_{M-1}(n)]^T$ and $\mathbf{w}(n) = [w_0(n) \ w_1(n) \ \ldots \ w_{N-1}(n)]^T$, data vectors $\mathbf{x}(n), \mathbf{b}(n-1)$ and $\mathbf{u}_a(n-1)$, desired output $d(n)$, and power estimates $P_0(n-1), P_1(n-1), \ldots, P_M(n-1)$.

Required: Vector updates $\boldsymbol{\kappa}(n), \mathbf{w}(n+1), \mathbf{b}(n)$ and $\mathbf{u}_a(n)$, and power estimate updates $P_0(n), P_1(n), \ldots, P_M(n)$.

Lattice Predictor Part

$f_0(n) = b_0(n) = x(n)$
$P_0(n) = \beta P_0(n-1) + 0.5(1-\beta)[f_0^2(n) + b_0^2(n-1)]$
for $m = 1$ to M
$f_m(n) = f_{m-1}(n) - \kappa_m(n) b_{m-1}(n-1)$
$b_m(n) = b_{m-1}(n-1) - \kappa_m(n) f_{m-1}(n)$
$\kappa_m(n+1) = \kappa_m(n) + \dfrac{2\mu_{p,o}}{P_{m-1}(n) + \varepsilon}[f_{m-1}(n) b_m(n) + b_{m-1}(n-1) f_m(n)]$
$P_m(n) = \beta P_m(n-1) + 0.5(1-\beta)[f_m^2(n) + b_m^2(n-1)]$
if $|\kappa_m(n+1)| > \alpha$, $\kappa_m(n+1) = \kappa_m(n)$
end

$\mathbf{u}_a(n)$ update

$u_a(n-j) = u_a(n-j+1)$, for $j = N-1, \ N-2, \ \ldots, \ 2$
$f_0'(n) = b_0'(n) = b_M(n)$
for $m = 1$ to $M-1$
$f_m'(n) = f_{m-1}'(n) - \kappa_m(n) b_{m-1}'(n-1)$
$b_m'(n) = b_{m-1}'(n-1) - \kappa_m(n) f_{m-1}'(n)$
end
$u_a(n) = (P_M(n) + \varepsilon)^{-1}(f_{M-1}'(n) - \kappa_M(n) b_{M-1}'(n-1))$

Filtering

$y(n) = \mathbf{w}^T(n)\mathbf{x}(n-M)$
$e(n) = d(n-M) - y(n)$
$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{u}_a(n)$

11.15.2 Performance analysis

An analysis that reveals the differences between Algorithms 1 and 2 is presented. We assume that the input process, $x(n)$, is AR of order less than or equal to M. The predictors' coefficients, $\{a_{i,j}$, for $i = 1, 2, \ldots, M$ and $j = 1, 2, \ldots, i\}$, and the corresponding mean-square prediction error for different orders (i.e. the diagonal elements of $\mathbf{R}_{bb}$) are assumed to be known. In practice, when $M \ll N$, these assumptions are acceptable with a good approximation, since in that case the predictors' coefficients will converge much faster than the adaptive filter tap weights and they will be jittering near their optimum setting after an initial transient. With these assumptions we find that $\mathbf{u}(n)$ is an exact estimate of $\mathbf{R}_{xx}^{-1}\mathbf{x}(n)$ and, therefore, Algorithm 1 will be an exact implementation of the ideal

LMS–Newton algorithm, for which some theoretical results were presented in Chapter 7. We consider these results here as a base that determines the best performance that we may expect from Algorithm 1. Moreover, comparison of these results with what would be achieved by Algorithm 2, under the same ideal conditions, gives a good measure of the performance loss of Algorithm 2 as a result of simplifications made in its structure.

Under the ideal conditions stated above, the following results of the ideal LMS–Newton algorithm (presented in Chapter 7) are applicable to Algorithm 1.

- The algorithm does not suffer from any eigenvalue spread problem. It has only one mode of convergence which is characterized by the time constant

$$\tau = \frac{1}{4\mu}. \tag{11.131}$$

- For small values of the step-size parameter, μ, its misadjustment is given by the equation

$$\mathcal{M}_1 = \frac{\mu N}{1 - \mu(N+2)}. \tag{11.132}$$

  This result is obtained by letting $\lambda_0 = \lambda_1 = \ldots = \lambda_{N-1} = 1$ in (6.60).
- To guarantee the stability of the algorithm, its step-size parameter should remain within the limits

$$0 < \mu < \frac{1}{N+2}. \tag{11.133}$$

  This follows from (11.132) and the same line of arguments to those in Section 6.3.4.

The derivation of the above results has been based on a number of assumptions which we shall also assume here before proceeding to the analysis of Algorithm 2. A modelling problem such as Figure 11.10 is considered and the following assumptions are made:

1. The input samples, $x(n)$, and the desired output samples, $d(n)$, consist of jointly Gaussian-distributed random variables for all n.
2. At time n, $\mathbf{w}(n)$ is independent of the input vector $\mathbf{x}(n)$ and the desired output sample $d(n)$.
3. Noise samples $e_o(n)$, for all n, are zero-mean and uncorrelated with the input samples, $x(n)$.

The validity of the second assumption is justified for small values of μ, as discussed in Chapter 6. For the analysis of Algorithm 2, we extend these assumptions by replacing $\mathbf{x}(n)$ with $\mathbf{x}_E(n)$, so that it extends to include the independence of $\mathbf{u}_a(n)$ with $\mathbf{w}(n)$.

Now, we proceed with an analysis of Algorithm 2. First we present an analysis of the convergence of $\mathbf{w}(n)$ in the mean which gives a result similar to (11.131). Next we analyse the convergence of $\mathbf{w}(n)$ in the variance to obtain the misadjustment of the algorithm. This analysis also reveals the effect of replacing $\mathbf{u}(n)$ by $\mathbf{u}_a(n)$.

Convergence of the tap-weight vector in the mean

We look at the convergence of $E[\mathbf{w}(n)]$ as n increases. To this end, we note that

$$e(n) = d(n) - \mathbf{w}^T(n)\mathbf{x}(n) = e_o(n) - \mathbf{v}^T(n)\mathbf{x}(n), \tag{11.134}$$

where $\mathbf{v}(n) = \mathbf{w}(n) - \mathbf{w}_o$ is the weight-error vector and from Figure 11.10 we have noted that $d(n) = \mathbf{w}_o^T\mathbf{x}(n) + e_o(n)$. Substituting (11.134) in (11.127), we get

$$\mathbf{v}(n+1) = (\mathbf{I} - 2\mu\mathbf{u}_a(n)\mathbf{x}^T(n))\mathbf{v}(n) + 2\mu e_o(n)\mathbf{u}_a(n), \tag{11.135}$$

where $\mathbf{I}$ denotes the identity matrix with appropriate dimension. Taking expectations and using Assumptions 2 and 3 listed above, we obtain

$$\mathrm{E}[\mathbf{v}(n+1)] = \big(\mathbf{I} - 2\mu\mathrm{E}\big[\mathbf{u}_a(n)\mathbf{x}^T(n)\big]\big)\mathrm{E}[\mathbf{v}(n)]. \tag{11.136}$$

To evaluate $\mathrm{E}[\mathbf{u}_a(n)\mathbf{x}(n)]$, we first define

$$\mathbf{u}_E(n) = \mathbf{R}_{x_E x_E}^{-1}\mathbf{x}_E(n), \tag{11.137}$$

where $\mathbf{R}_{x_E x_E} = \mathrm{E}\big[\mathbf{x}_E(n)\mathbf{x}_E^T(n)\big]$, and note that postmultiplying (11.137) by $\mathbf{x}_E^T(n)$ and taking expectations on both sides gives

$$\mathrm{E}\big[\mathbf{u}_E(n)\mathbf{x}_E^T(n)\big] = \mathbf{I}. \tag{11.138}$$

This shows that the cross-correlation between the elements of $\mathbf{u}_E(n)$ and $\mathbf{x}_E(n)$ that are at the same position are unity and equal to zero for the other elements of the two vectors. Clearly, this also is applicable to the elements of $\mathbf{u}_a(n)$ and $\mathbf{x}(n)$, since they are truncated versions of $\mathbf{u}_E(n)$ and $\mathbf{x}_E(n)$, respectively. Thus

$$\mathrm{E}\big[\mathbf{u}_a(n)\mathbf{x}^T(n)\big] = \mathbf{I} \tag{11.139}$$

and therefore

$$\mathrm{E}[\mathbf{v}(n+1)] = (1 - 2\mu)\mathrm{E}[\mathbf{v}(n)]. \tag{11.140}$$

This shows that, similar to Algorithm 1, Algorithm 2 also is governed by a single mode of convergence. Furthermore, the time constant equation (11.131) is also applicable to Algorithm 2.

Convergence of the tap-weight vector in the mean square

We first develop a recursive equation for the time evolution of the correlation matrix of the weight-error vector $\mathbf{v}(n)$, which is defined as $\mathbf{K}(n) = \mathrm{E}\big[\mathbf{v}(n)\mathbf{v}^T(n)\big]$. For this, we find the outer products of the left-hand and right-hand sides of (11.135) and take expectations on both sides of the resulting equation. Then, using Assumptions 2 and 3 listed above, we obtain

$$\begin{aligned}
\mathbf{K}(n+1) &= \mathrm{E}\big[(\mathbf{I} - 2\mu\mathbf{u}_a(n)\mathbf{x}^T(n))\mathbf{K}(n)(\mathbf{I} - 2\mu\mathbf{x}(n)\mathbf{u}_a^T(n))\big] + 4\mu^2\xi_{\min}\mathbf{R}_{u_a u_a} \\
&= \mathbf{K}(n) - 2\mu\mathrm{E}\big[\mathbf{u}_a(n)\mathbf{x}^T(n)\big]\mathbf{K}(n) - 2\mu\mathbf{K}(n)\mathrm{E}\big[\mathbf{x}(n)\mathbf{u}_a^T(n)\big] \\
&\quad + 4\mu^2\mathrm{E}\big[\mathbf{u}_a(n)\mathbf{x}^T(n)\mathbf{K}(n)\mathbf{x}(n)\mathbf{u}_a^T(n)\big] + 4\mu^2\xi_{\min}\mathbf{R}_{u_a u_a} \\
&= (1 - 4\mu)\mathbf{K}(n) + 4\mu^2\mathrm{E}\big[\mathbf{u}_a(n)\mathbf{x}^T(n)\mathbf{K}(n)\mathbf{x}(n)\mathbf{u}_a^T(n)\big] + 4\mu^2\xi_{\min}\mathbf{R}_{u_a u_a},
\end{aligned} \tag{11.141}$$

where $\xi_{\min} = \mathrm{E}[e_o^2(n)]$ is the minimum mean-square error at the adaptive filter output and $\mathbf{R}_{u_a u_a} = \mathrm{E}[\mathbf{u}_a(n)\mathbf{u}_a^{\mathrm{T}}(n)]$.

The second term on the right-hand side of (11.141) can be evaluated by following a procedure similar to the one given in Chapter 6, Appendix 6A, for the case of the conventional LMS algorithm. This results in (see Appendix 11A for the derivation)

$$\mathrm{E}[\mathbf{u}_a(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{K}(n)\mathbf{x}(n)\mathbf{u}_a^{\mathrm{T}}(n)] = \mathbf{R}_{u_a u_a}\mathrm{tr}[\mathbf{K}(n)\mathbf{R}_{xx}] + 2\mathbf{K}(n). \tag{11.142}$$

Using this result in (11.141), we obtain

$$\mathbf{K}(n+1) = (1 - 4\mu + 8\mu^2)\mathbf{K}(n) + 4\mu^2\mathbf{R}_{u_a u_a}\mathrm{tr}[\mathbf{K}(n)\mathbf{R}_{xx}] + 4\mu^2\xi_{\min}\mathbf{R}_{u_a u_a}. \tag{11.143}$$

Next, we recall from Chapter 6 that the excess mean-square error of an adaptive filter with input and weight-error correlation matrices $\mathbf{R}_{xx}$ and $\mathbf{K}(n)$, respectively, is given by

$$\xi_{\mathrm{ex}}(n) = \mathrm{tr}[\mathbf{K}(n)\mathbf{R}_{xx}]. \tag{11.144}$$

Post-multiplying (11.143) on both sides by $\mathbf{R}_{xx}$ and equating the traces of the two sides of the resulting equation, we obtain

$$\xi_{\mathrm{ex}}(n+1) = \big(1 - 4\mu + 4\mu^2(\mathrm{tr}[\mathbf{R}_{u_a u_a}\mathbf{R}_{xx}] + 2)\big)\xi_{\mathrm{ex}}(n) + 4\mu^2\xi_{\min}\mathrm{tr}[\mathbf{R}_{u_a u_a}\mathbf{R}_{xx}]. \tag{11.145}$$

From (11.145) we note that the convergence of Algorithm 2 is guaranteed if

$$|1 - 4\mu + 4\mu^2(\mathrm{tr}[\mathbf{R}_{u_a u_a}\mathbf{R}_{xx}] + 2)| < 1. \tag{11.146}$$

This gives

$$0 < \mu < \frac{1}{\mathrm{tr}[\mathbf{R}_{u_a u_a}\mathbf{R}_{xx}] + 2}. \tag{11.147}$$

Also, when $n \to \infty$, $\xi(n+1) = \xi(n)$. Using this in (11.145) we obtain

$$\mathcal{M}_2 = \frac{\xi_{\mathrm{ex}}(\infty)}{\xi_{\min}} = \frac{\mu\mathrm{tr}[\mathbf{R}_{u_a u_a}\mathbf{R}_{xx}]}{1 - \mu(\mathrm{tr}[\mathbf{R}_{u_a u_a}\mathbf{R}_{xx}] + 2)}. \tag{11.148}$$

This is the misadjustment equation for Algorithm 2. The above results reduce to those of Algorithm 1 if $\mathbf{R}_{u_a u_a}$ is replaced by $\mathbf{R}_{uu} = \mathrm{E}[\mathbf{u}(n)\mathbf{u}^{\mathrm{T}}(n)]$ and we note that $\mathbf{R}_{uu} = \mathbf{R}_{xx}^{-1}$.

In view of (11.132) and (11.148), a good measure for comparing Algorithms 1 and 2 is the ratio

$$\gamma = \frac{\mathrm{tr}[\mathbf{R}_{u_a u_a}\mathbf{R}_{xx}]}{N}. \tag{11.149}$$

A value of $\gamma > 1$ indicates that Algorithm 1 performs better than Algorithm 2. Furthermore, the larger the value of γ, the greater would be the loss in replacing Algorithm 1 by Algorithm 2. However, if $\gamma \approx 1$, then the two algorithms perform about the same.

An evaluation of the parameter γ is provided in Appendix 11B. It is shown that γ is always greater than unity. This means that there is always a penalty to be paid for the simplification made in replacing the vector $\mathbf{u}(n)$ of Algorithm 1, by the vector $\mathbf{u}_a(n)$ of Algorithm 2. The amount of loss depends on the statistics of the input process, $x(n)$, and the

filter length N. Fortunately, the evaluation provided in Appendix 11B shows that γ approaches one as N increases. This means that the difference between the two algorithms may be insignificant for long filters. Numerical examples that verify this are given next.

11.15.3 Simulation results and discussion

We present some simulation results using the input process $x_1(n)$, which was introduced in Section 11.13.1, and also two other processes, $x_2(n)$ and $x_3(n)$, which are generated using the colouring filters

$$H_2(z) = \frac{K_2}{1 - 0.650z^{-1} + 0.693z^{-2} - 0.220z^{-3} + 0.309z^{-4} - 0.177z^{-5}} \tag{11.150}$$

and

$$H_3(z) = \frac{K_3}{1 - 2.059z^{-1} + 2.312z^{-2} - 1.893z^{-3} + 1.148z^{-4} - 0.293z^{-5}}, \tag{11.151}$$

respectively. The coefficients K_2 and K_3 are selected equal to 0.7208 and 0.2668, respectively, to normalize the resulting processes to unit power. We note that $x_2(n)$ and $x_3(n)$ are AR processes, but $x_1(n)$ is not.

To verify the theoretical results presented above, we start with some experiments using the AR processes $x_2(n)$ and $x_3(n)$. Figure 11.14 shows the power spectral densities of $x_2(n)$ and $x_3(n)$. From Chapter 4 we recall that the eigenvalue spread of the correlation matrix of a process is asymptotically determined by the maximum and minimum of its power spectral density. Noting this, we find that the eigenvalue spread of $x_2(n)$ is in the range of 100 and that of $x_3(n)$ can be as large as 10,000. This shows that $x_3(n)$ is a very badly conditioned process and one should expect difficulties in estimating the inverse of its correlation matrix.

To shed light on the differences between Algorithm 1 and Algorithm 2, we first present some simulation results for the case when the exact models of the AR inputs are known a priori. In this case, Algorithm 1 will be an exact implementation of the LMS–Newton algorithm and gives a good base for further comparisons. Figure 11.15 shows the variation in the parameter γ as a function of the filter length, N, for $x_2(n)$ and $x_3(n)$. As might be expected, the process $x_3(n)$, which suffers from a serious eigenvalue spread problem, shows higher sensitivity towards replacing Algorithm 1 by Algorithm 2. However, as N increases, γ approaches one and, therefore, the two algorithms are expected to perform about the same.

Figures 11.16 and 11.17 show the simulation results for the inputs $x_2(n)$ and $x_3(n)$ and a filter length $N = 30$. These results, as well as those presented in the rest of this section, are averaged over 50 independent runs. The results are then smoothed so that the various curves could be distinguished. The step-size parameter, μ, is selected equal to $0.1/N$, for all the results. This, according to equation (11.132), results in about a 10% misadjustment for Algorithm 1. According to the results of Figure 11.15 and equations (11.132) and (11.148), both algorithms should approach about the same misadjustment in the case of $x_2(n)$. However, their performance may be significantly different in the case of $x_3(n)$. To be more exact, from the data used for generating Figure 11.15 we have $\gamma = 1.13$, for $x_2(n)$, and $\gamma = 5.57$, for $x_3(n)$, for $N = 30$. Using these and equations

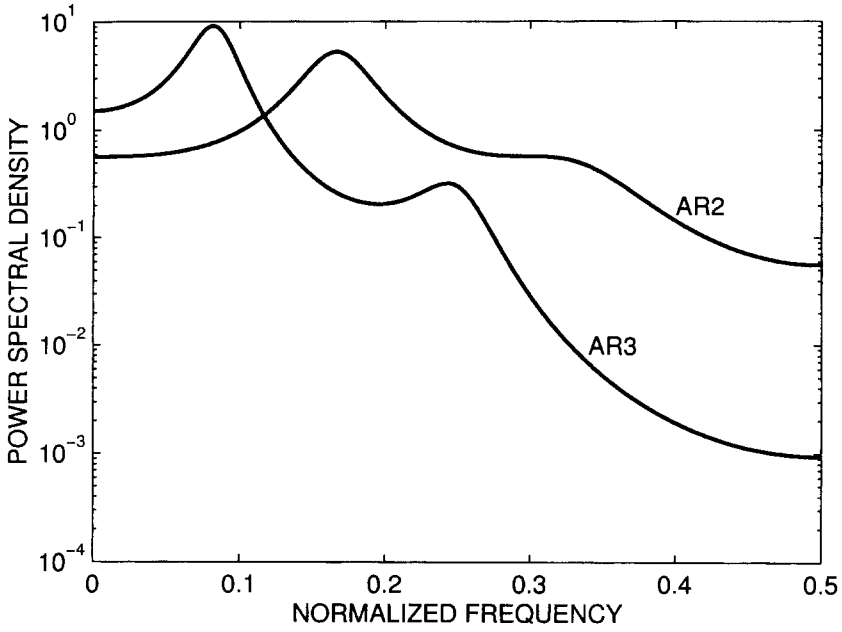


Figure 11.14 Power spectral densities of $x_2(n)$ (AR2) and $x_3(n)$ (AR3). Reprinted from Farhang-Boroujeny (1997b)

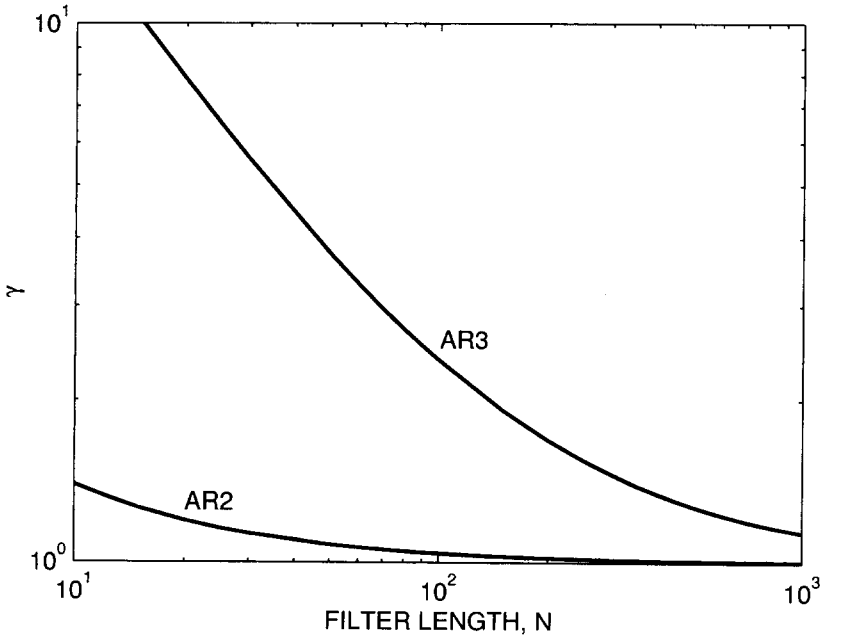


Figure 11.15 Variation of the parameter γ as a function of filter length for $x_2(n)$ (AR2) and $x_3(n)$ (AR3). Reprinted from Farhang-Boroujeny (1997b)

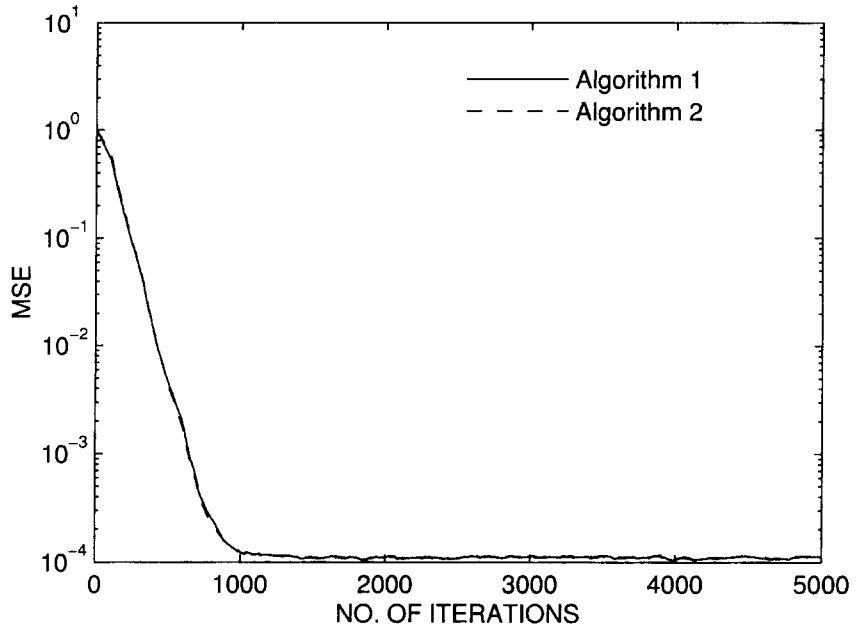


Figure 11.16 MSE vs. iteration number for $x_2(n)$, $N = 30$, and with the AR model of input assumed known. Reprinted from Farhang-Boroujeny (1997b)

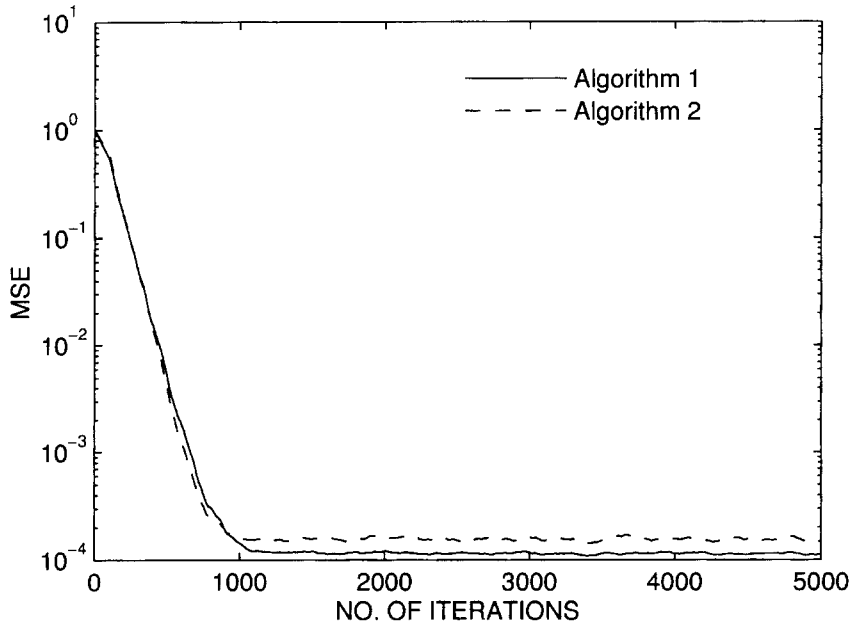


Figure 11.17 MSE vs. iteration number for $x_3(n)$, $N = 30$, and with the AR model of input assumed known. Reprinted from Farhang-Boroujeny (1997b)

(11.132) and (11.148), we obtain:

$$\text{for } x_2(n): \frac{\mathcal{M}_2}{\mathcal{M}_1} = 1.147;$$

$$\text{for } x_3(n): \frac{\mathcal{M}_2}{\mathcal{M}_1} = 11.32.$$

Careful examination of the numerical values obtained by simulations show that for the $x_2(n)$ process $\mathcal{M}_2/\mathcal{M}_1 = 1.152$. This matches well with the above ratio. However, for the $x_3(n)$ process the simulation results give $\mathcal{M}_2/\mathcal{M}_1 = 3.85$. This, which does not match the above theoretical ratio, may be explained as follows. Careful examination of the numerical results in simulations revealed that there are only a few terms in $\mathbf{u}_a(n)$ that have a major effect on the degradation of Algorithm 2 when compared with Algorithm 1. These terms, which greatly disturb the first and last few elements of the tap-weight vector $\mathbf{w}(n)$, are so large that their contribution violates the independence assumption 2 of the previous section. As a result, the theoretical derivation that led to (11.148) may not be valid unless the step-size parameter, μ, is set to a very small value so that the latter assumption could be justified. Nevertheless, the developed theory is able to predict conditions under which Algorithm 2 is more likely to go unstable, namely when the adaptive filter input is highly coloured.

To support the prediction made by the theory that the two algorithms perform about the same for long filters, we present another simulation example with the process $x_3(n)$ as the filter input. This time we increase the length of the filter, N, to 200. Figure 11.18

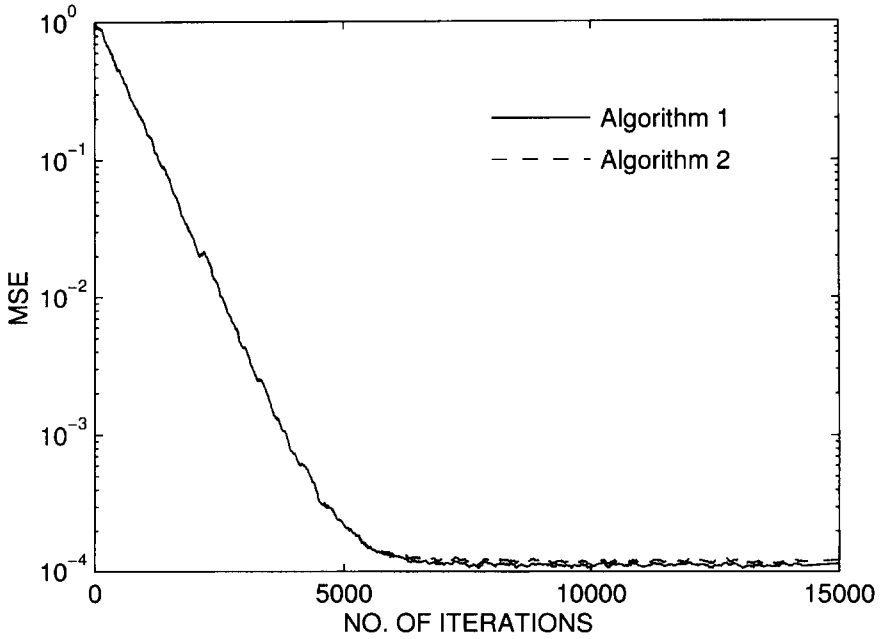
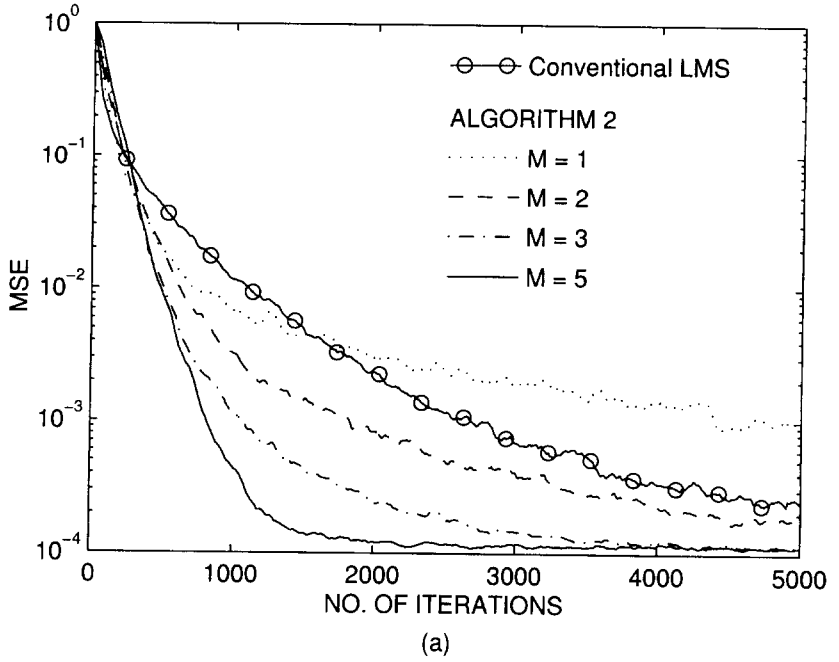


Figure 11.18 MSE vs. iteration number for $x_3(n)$, $N = 200$, and with the AR model of input assumed known. Reprinted from Farhang-Boroujeny (1997b)



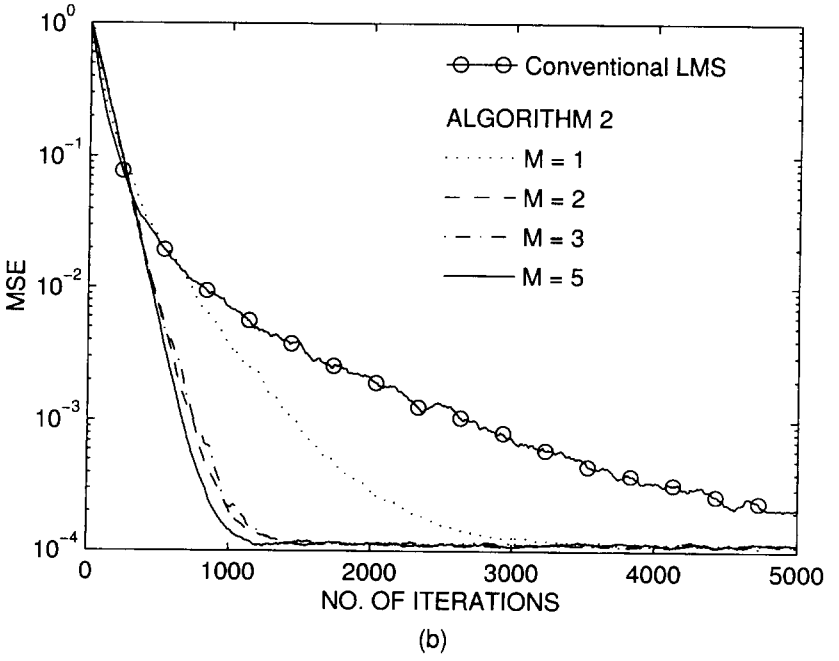


Figure 11.19 Comparison of the conventional LMS and Algorithm 2, for different inputs and various orders of AR model: (a) input process $x_1(n)$, (b) input process $x_2(n)$, and (c) input process $x_3(n)$ Reprinted from Farhang-Boroujeny (1997b)

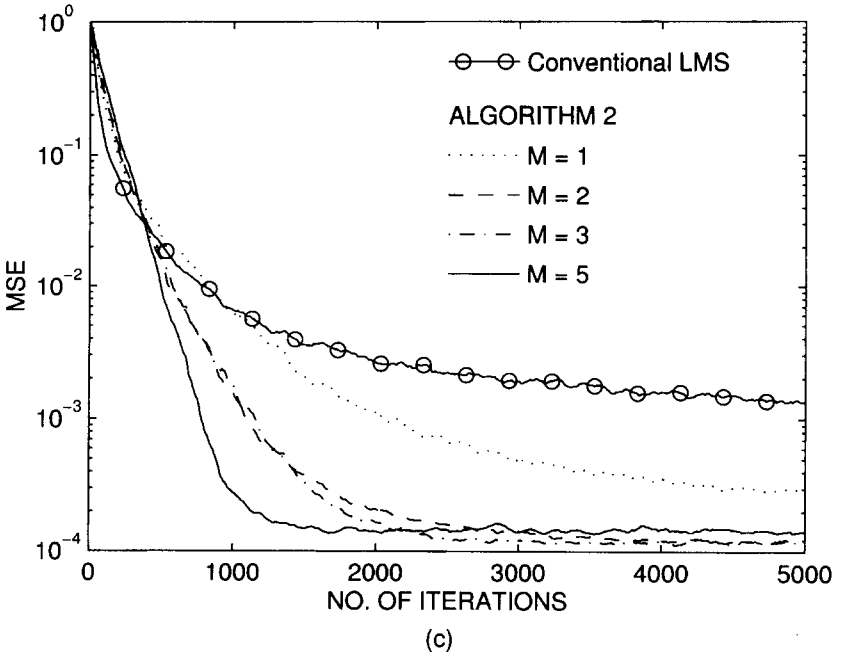


Figure 11.19 Continued

shows the results of this test. For this scenario the theory gives $\mathcal{M}_2/\mathcal{M}_1 = 1.69$ and simulation gives $\mathcal{M}_2/\mathcal{M}_1 = 1.64$, a good match, as was predicted.

Next, the simulation results of more realistic cases when the input process is unknown and its model has to be estimated along with the adaptive filter tap weights are presented. We present some results for Algorithm 2. The simulation program that we use follows Table 11.7. The following parameters are used: $\beta = 0.95$, $\alpha = 0.9$, $\varepsilon = 0.02$, $\mu_{p,o} = 0.01$, $\mu = 0.1/N$ and $N = 30$. Figures 11.19(a), (b) and (c) show the simulation results for the processes $x_1(n)$, $x_2(n)$ and $x_3(n)$, respectively. The results are given for the conventional LMS algorithm and Algorithm 2, for the cases where the order of the AR model, M, is set equal to 1, 2, 3 and 5. The results clearly show the improvement achieved by AR modelling. We note that for $x_2(n)$ and $x_3(n)$, even a first-order modelling of the input processes results in significant improvement in convergence compared with the conventional LMS algorithm. However, for $x_1(n)$ a modelling order of 2 or above is required to achieve some improvement.

Problems

P11.1 Give a detailed proof of (11.52) and find that such a proof leads to (11.53).

P11.2 Define the $m \times m$ matrix $\mathbf{J}$ whose ijth element is 1 for $j = m - i + 1$ and 0 for all other $i, j \in \{1, 2, \ldots, m\}$. This is called an exchange matrix. Show that if $\mathbf{R}$ is the correlation matrix of a stationary stochastic process, then $\mathbf{JRJ} = \mathbf{R}$. Use this result to derive an alternative proof for (11.19) or (11.20).

P11.3 Using the procedure mentioned in Problem P4.8, show that for the matrix $\mathbf{L}$ as defined in (11.70)

$$\det(\mathbf{L}) = 1.$$

Comment on the invertability of $\mathbf{L}$.

P11.4 Give a detailed derivation of (11.81).

P11.5 Consider the order-update equations (11.60) and (11.61). Show that optimization of κ_{m+1} in either of the two equations for minimization of the corresponding higher order errors in the mean-square sense gives (11.59).

P11.6 Equation (11.90) may be rearranged as

$$r(m+1) = P_m \kappa_{m+1} + \sum_{i=1}^{m} a_{m,i} r(m+1-i).$$

Use this result to develop procedures for:

(i) Conversion of the set of coefficients $(P_0, \kappa_1, \kappa_2, \ldots, \kappa_M)$ to $(r(0), r(1), \ldots, r(M))$.
(ii) Conversion of the set of coefficients $(P_0, a_{M,1}, a_{M,2}, \ldots, a_{M,M})$ to $(r(0), r(1), \ldots, r(M))$.

P11.7 Using (11.92), derive the recursion used for obtaining the transversal predictor coefficients $w_{m,i}$ in Table 11.4.

P11.8 Give the lattice equivalent of the forward prediction-error filter which is characterized by the system function

$$H_{f_4}(z) = 1 - 0.5z^{-2} + 0.5z^{-3} + 0.25z^{-4}.$$

P11.9 Give the transversal equivalent of the third-order forward and backward prediction-error filters of a process which is characterized by the PARCOR coefficients

$$\kappa_1 = 0.8, \quad \kappa_2 = 0.5, \quad \kappa_3 = -0.2.$$

P11.10 Find the lattice realization of the system function

$$H(z) = \frac{1 + z^{-1} + 2z^{-2}}{1 - 1.2z^{-1} + 0.5z^{-2}}.$$

P11.11 Use the Levinson–Durbin algorithm to find the fourth-order transversal and lattice predictors of a process $x(n)$ which is characterized by the correlation coefficients

$$r(0) = 5, \quad r(1) = 3, \quad r(2) = -1, \quad r(3) = 2, \quad r(4) = -0.5$$

P11.12 Use the Levinson–Durbin algorithm to solve the system of equations

$$\begin{bmatrix} 1.0 & 0.8 & -0.5 & 0.2 \\ 0.8 & 1.0 & 0.8 & -0.5 \\ -0.5 & 0.8 & 1.0 & 0.8 \\ 0.2 & -0.5 & 0.8 & 1.0 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 0.8 \\ -0.5 \\ 0.2 \\ 0 \end{bmatrix}.$$

P11.13 Use the extended Levinson–Durbin algorithm to solve the system of equations

$$\begin{bmatrix} 1.0 & 0.8 & -0.5 & 0.2 \\ 0.8 & 1.0 & 0.8 & -0.5 \\ -0.5 & 0.8 & 1.0 & 0.8 \\ 0.2 & -0.5 & 0.8 & 1.0 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 1 \\ 0.5 \\ 0 \end{bmatrix}.$$

P11.14 Consider an AR process described by the difference equation

$$x(n) = 0.7x(n-1) + 0.66x(n-2) - 0.432x(n-3) + \nu(n),$$

where $\nu(n)$ is a zero-mean white noise process with variance of unity.

(i) Find the system function $H(z)$ that relates $\nu(n)$ and $x(n)$.
(ii) Show that the poles of $H(z)$ are 0.9, −0.8 and 0.6.
(iii) Find the power of $x(n)$.
(iv) Find the PARCOR coefficients κ_1, κ_2 and κ_3 of $x(n)$.
(v) Find the prediction-error powers P_1, P_2 and P_3 of $x(n)$.
(vi) Comment on the values of κ_m and P_m, for values of $m \geq 4$.
(vii) Using the procedure developed in Problem P11.6, find the autocorrelation coefficients $r(0), r(1), \ldots, r(5)$ of $x(n)$.

P11.15 For a real-valued process $x(n)$ with mth order forward and backward prediction errors $f_m(n)$ and $b_m(n)$, respectively, prove the following results.

(i) $\mathrm{E}[f_5(n)f_3(n-2)] = 0$.
(ii) $\mathrm{E}[b_5(n)b_2(n-2)] = 0$.
(iii) $\mathrm{E}[f_m(n)x(n)] = \mathrm{E}[f_m^2(n)]$.
(iv) $\mathrm{E}[b_m(n)x(n-m)] = \mathrm{E}[b_m^2(n)]$.
(v) For $0 < k < m$, $\mathrm{E}[f_m(n)f_{m-k}(n-k)] = 0$.
(vi) For $0 < k < m$, $\mathrm{E}[b_m(n)b_{m-k}(n-k)] = \mathrm{E}[b_m^2(n)]$.

P11.16 For a real-valued process $x(n)$ with mth order forward and backward prediction errors $f_m(n)$ and $b_m(n)$, respectively, find the range of i for which the following results hold.

(i) $\mathrm{E}[f_m(n)f_{m-k}(n-i)] = 0$.
(ii) $\mathrm{E}[b_m(n)b_{m-k}(n-i)] = 0$.
(iii) $\mathrm{E}[f_m(n)b_{m-k}(n-i)] = 0$.
(iv) $\mathrm{E}[b_m(n)f_{m-k}(n-i)] = 0$.

P11.17 Consider a complex-valued process $x(n)$.

(i) Using the principle of orthogonality, derive the Wiener–Hopf equation that gives the coefficients $a_{m,i}$ of the order m forward linear predictor of $x(n)$.
(ii) Repeat (i) to derive the coefficients $g_{m,i}$ of the order m backward linear predictor of $x(n)$.
(iii) Show that

$$g_{m,i} = a^*_{m,m+1-i}, \quad \text{for } i = 1, 2, \ldots, m.$$

P11.18 In the case of complex-valued signals, the order-update equations (11.60) and (11.61) take the following forms:

$$f_{m+1}(n) = f_m(n) - \kappa_{m+1} b_m(n-1)$$

and

$$b_{m+1}(n) = b_m(n-1) - \kappa^*_{m+1} f_m(n)$$

where the asterisk denotes complex conjugation. Give a detailed proof of these equations and show that

$$\kappa_{m+1} = \frac{\mathrm{E}[f_m(n) b^*_m(n-1)]}{P_m},$$

where $P_m = \mathrm{E}[|f_m(n)|^2] = \mathrm{E}[|b_m(n-1)|^2]$.

P11.19 For the case of complex-valued signals, propose a lattice joint process estimator similar to Figure 11.7 and develop an LMS algorithm for its adaptation.

P11.20 For a complex-valued process $x(n)$ prove the following properties:

(i) $\mathrm{E}[f_m(n) x^*(n-k)] = 0, \quad \text{for } 1 \le k \le m.$
(ii) $\mathrm{E}[b_m(n) x^*(n-k)] = 0, \quad \text{for } 0 \le k \le m-1.$
(iii) $\mathrm{E}[b_k(n) b^*_l(n)] = 0, \quad \text{for } k \ne l.$

P11.21 Consider the difference equation (11.114) and note that it may be written as

$$x(n) = \mathbf{x}^{\mathrm{T}}_M(n-1)\mathbf{h} + \nu(n) \qquad \text{(P11.21-1)}$$

where $\mathbf{x}_M(n) = [x(n-1) \;\; x(n-2) \;\; \ldots \;\; x(n-M)]^{\mathrm{T}}$ and $\mathbf{h} = [h_1 \;\; h_2 \;\; \ldots \;\; h_M]^{\mathrm{T}}$.

(i) Starting with (P11.21-1) show that the vector $\mathbf{h}$ is related to the autocorrelation coefficients $r(0), r(1), \ldots, r(M)$ of $x(n)$ according to the equation

$$\mathbf{R}_M \mathbf{h} = \mathbf{r}_M,$$

where

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

and

$$\mathbf{r}_M = [r(1)\ \ r(2)\ \ \ldots\ \ r(M)]^{\mathrm{T}}.$$

(ii) Show that for any m

$$r(m) = \sum_{i=1}^{M} h_i r(m-i).$$

(iii) By combining the results of (i) and (ii) show that for any $M' > M$,

$$\mathbf{R}_{M'}\mathbf{h}' = \mathbf{r}_{M'},$$

where

$$\mathbf{h}' = \begin{bmatrix} \mathbf{h} \\ \mathbf{0} \end{bmatrix}$$

and $\mathbf{0}$, here, is the length $M' - M$ zero column vector.

(iv) Use the above results to justify the validity of the results presented in (11.115) and (11.116).

P11.22 Suggest a lattice structure for the realization of the transfer function $W(z)$ of the IIR line enhancer of Section 10.3 and obtain its coefficients in terms of the parameters s and w.

P11.23 Give a detailed derivation of (11.123).

P11.24 Give a derivation of (11.147) from (11.146).

P11.25 Recall that the unconstrained PFBLMS algorithm of Chapter 8 converges very slowly when the partition length, M, and the block length, L, are equal. In Section 8.4 it was noted that the slow convergence of the PFBLMS algorithm can be improved by choosing M a few times larger than L. We also noticed that the frequency domain processing involved in the implementation of the PFBLMS algorithm may be viewed as a parallel bank of a number of transversal filters, each belonging to one of the frequency bins. Furthermore, when the number of frequency bins is large, these filters operate (converge) almost independently of one another. Noting these, the following alternative solution may be proposed to improve the convergence behaviour of the PFBLMS algorithm. We may keep $L = M$ and use a lattice structure for decorrelating the samples of the input signal at each frequency bin. Explore this solution. In particular, note that when the filter input, $x(n)$, is a white process, the autocorrelation coefficients of the signal samples at various frequency bins are known a priori (see (8.88)). Explain how this known information can be exploited in the proposed implementation.

Simulation-Oriented Problems

P11.26 Write a simulation program to confirm the results presented in Figure 11.12. If you are looking for a short-cut and you have access to the MATLAB software package, then you may study and use the program 'ltc_mdlg.m' on the accompanying diskette. Also, run 'ltc_mdlg.m' or your program for the following values of the step-size parameters $\mu_{p,o}$ and $\mu_{c,o}$ and observe the impact of those on the performance of the algorithm. Comment on your observations.

$\mu_{p,o}$	$\mu_{c,o}$
0.01	0.003
0.001	0.010
0.001	0.001
0.0001	0.003

P11.27 Consider a process $x(n)$ that is characterized by the difference equation

$$x(n) = 1.2x(n-1) - 0.8x(n-2) + \nu(n) + \alpha\nu(n-1),$$

where α is a parameter and $\nu(n)$ is a zero-mean, unit-variance, white process.

(i) Derive an equation for the power spectral density $\Phi_{xx}(e^{j\omega})$ of $x(n)$ [illegible]

for values of $\alpha = 0$, 0.5, 0.8 and 0.95.

(ii) The autocorrelation coefficients $r(0)$, $r(1)$, ... of $x(n)$ can be obtained numerically by evaluating the inverse discrete Fourier transform (DFT) of samples of $\Phi_{xx}(e^{j\omega})$ taken at equally spaced intervals. The number of samples of $\Phi_{xx}(e^{j\omega})$ used for this purpose should be large enough to give an accurate result. Use this method to obtain the autocorrelation coefficients of $x(n)$.

(iii) Use the results of part (ii) to obtain the system functions of the backward prediction-error filters of $x(n)$ for predictor orders of 2, 5, 10 and 20, and values

(ii) Use the process $x(n)$ of P11.27 as the adaptive filter input and try that for values of $\alpha = 0$, 0.5, 0.8 and 0.95, and various values of the AR modelling order, i.e. the parameter M. Comment on your observations.

To get further insight, you may need to evaluate the parameter γ for each case. Write a program to generate γ. For this you need to calculate the autocorrelation functions of $x(n)$ and $u_a(n)$ and use them to build the matrices $\mathbf{R}_{xx}$ and $\mathbf{R}_{u_a u_a}$ required in (11.149). These can conveniently be obtained by calculating the inverse DFT [illegible] of the corresponding spectral densities of the corresponding processes

Substituting (11A-5) in (11A-3) and noting that $k_{lm}(n) = k_{ml}(n)$, we obtain

$$\begin{aligned} \mathrm{E}[c_{lm}(n)] &= r^{lm}_{u_a u_a} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} r^{ij}_{xx} k_{ij}(n) + 2k_{lm}(n) \\ &= r^{lm}_{u_a u_a} \mathrm{tr}[\mathbf{K}(n)\mathbf{R}_{xx}] + 2k_{lm}(n) \end{aligned} \tag{11A-7}$$

for $l = 0, 1, \ldots, N-1$ and $m = 0, 1, \ldots, N-1$.

Combining these elements to construct the matrix $\mathrm{E}[\mathbf{C}(n)] = \mathrm{E}\left[\mathbf{u}_a(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{K}(n)\mathbf{x}(n)\mathbf{u}_a^{\mathrm{T}}(n)\right]$, we get (11.142).

Appendix 11B: Evaluation of the Parameter γ

To evaluate γ, we proceed as follows:

$$\begin{aligned} \mathrm{tr}[\mathbf{R}_{u_a u_a}\mathbf{R}_{xx}] &= \mathrm{tr}[\mathrm{E}\left[\mathbf{u}_a(n)\mathbf{u}_a^{\mathrm{T}}(n)\right]\mathbf{R}_{xx}] \\ &= \mathrm{E}\left[\mathrm{tr}[\mathbf{u}_a(n)\mathbf{u}_a^{\mathrm{T}}(n)\mathbf{R}_{xx}]\right]. \end{aligned} \tag{11B-1}$$

We note that for any pair of matrices $\mathbf{A}$ and $\mathbf{B}$ with dimensions $N_1 \times N_2$ and $N_2 \times N_1$, respectively, $\mathrm{tr}[\mathbf{A}B] = \mathrm{tr}[\mathbf{B}A]$. Using this in (11B-1), we may write

$$\mathrm{tr}[\mathbf{R}_{u_a u_a}\mathbf{R}_{xx}] = \mathrm{E}\left[\mathbf{u}_a^{\mathrm{T}}(n)\mathbf{R}_{xx}\mathbf{u}_a(n)\right]. \tag{11B-2}$$

Note that the trace function has been dropped from the right-hand side of (11B-2), since $\mathbf{u}_a^{\mathrm{T}}(n)\mathbf{R}_{xx}\mathbf{u}_a(n)$ is a scalar.

Next, we recall that the correlation matrix $\mathbf{R}_{xx}$ may be decomposed as (see (4.23))

$$\mathbf{R}_{xx} = \sum_{i=0}^{N-1} \lambda_i \mathbf{q}_i \mathbf{q}_i^{\mathrm{T}}, \tag{11B-3}$$

where the λ_is and $\mathbf{q}_i$s are the eigenvalues and eigenvectors, respectively, of $\mathbf{R}_{xx}$. Using (11B-3) in (11B-2), we obtain

$$\mathrm{tr}[\mathbf{R}_{u_a u_a}\mathbf{R}_{xx}] = \sum_{i=0}^{N-1} \lambda_i \eta_i, \tag{11B-4}$$

where $\eta_i = \mathrm{E}[(\mathbf{q}_i^{\mathrm{T}}\mathbf{u}_a(n))^2]$.

Now, we shall analyse the terms $\lambda_i\eta_i$. For this, we refer to Figure 11B-1 which depicts a procedure for measuring $\lambda_i\eta_i$ through a sequence of filtering and averaging procedures. The AR process $x(n)$ is generated by passing its innovation, $\nu(n)$, through its model transfer function

$$H_{\mathrm{AR}}(\mathrm{e}^{j\omega}) = \frac{1}{1 - \sum_{m=1}^{M} a_{M,i}\mathrm{e}^{-jm\omega}}. \tag{11B-5}$$

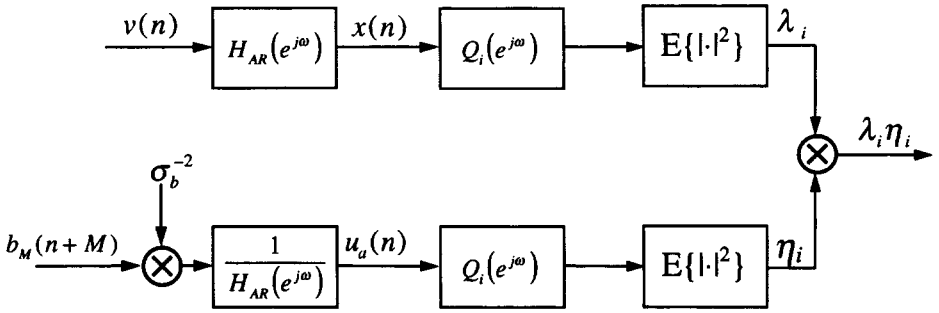


Figure 11B-1 Procedure for the evaluation of $\lambda_i\eta_i$.

The innovation $\nu(n)$ is a white noise process with variance σ_ν^2. Passing $x(n)$ through the eigenfilter $Q_i(\mathrm{e}^{j\omega})$ (the FIR filter whose coefficients are the elements of the eigenvector $\mathbf{q}_i$) generates a signal whose mean-square value is equal to λ_i. On the other hand, according to Figure 11.13, the sequence $u_\mathrm{a}(n)$ is generated from $b_M(n+M)$ by first multiplying that by σ_b^{-2} (the inverse of the variance of $b_M(n+M)$) and then passing the result through a FIR filter with the transfer function $z^{-M}H_{b_M}(z^{-1})$ which is nothing but $1/H_\mathrm{AR}(\mathrm{e}^{j\omega})$. Passing $u_\mathrm{a}(n)$ through the eigenfilter $Q_i(\mathrm{e}^{j\omega})$ generates the samples of the sequence $\mathbf{q}_i^\mathrm{T}\mathbf{u}_\mathrm{a}(n)$ whose mean-square value is then measured.

From Figure 11B-1 we may immediately write

$$\lambda_i = \frac{1}{2\pi}\int_0^{2\pi} \sigma_\nu^2 |H_\mathrm{AR}(\mathrm{e}^{j\omega})|^2 |Q_i(\mathrm{e}^{j\omega})|^2 \,\mathrm{d}\omega \tag{11B-6}$$

and

$$\eta_i = \frac{1}{2\pi}\int_0^{2\pi} \sigma_b^2 \frac{\sigma_b^{-4}}{|H_\mathrm{AR}(\mathrm{e}^{j\omega})|^2} |Q_i(\mathrm{e}^{j\omega})|^2 \,\mathrm{d}\omega. \tag{11B-7}$$

We also note that the innovation process $\nu(n)$ and the backward prediction error, $b_M(n)$ (or equivalently $b_M(n+M)$), are statistically the same. This implies that $\sigma_b^2 = \sigma_\nu^2$. Noting this, (11B-6) and (11B-7) give

$$\lambda_i\eta_i = \left(\frac{1}{2\pi}\right)^2 \left(\int_0^{2\pi} |H_\mathrm{AR}(\mathrm{e}^{j\omega})|^2 |Q_i(\mathrm{e}^{j\omega})|^2 \,\mathrm{d}\omega\right)\left(\int_0^{2\pi} \frac{1}{|H_\mathrm{AR}(\mathrm{e}^{j\omega})|^2} |Q_i(\mathrm{e}^{j\omega})|^2 \,\mathrm{d}\omega\right). \tag{11B-8}$$

Equation (11B-8) is in an appropriate form that may be used to give some argument with regard to the value of $\lambda_i\eta_i$ and the overall summation in (11B-4).

Now, if $f(x)$ and $g(x)$ are two arbitrary functions with finite energy in the interval (a, b), then the Cauchy–Schwartz inequality states that

$$\left|\int_\mathrm{a}^\mathrm{b} f(x)g(x)\,\mathrm{d}x\right|^2 \le \left(\int_\mathrm{a}^\mathrm{b} |f(x)|^2\,\mathrm{d}x\right)\left(\int_\mathrm{a}^\mathrm{b} |g(x)|^2\,\mathrm{d}x\right), \tag{11B-9}$$

with the equality valid when $f(x) = \alpha g(x)$, α being a scalar. Using this, (11B-8) gives

$$\lambda_i \eta_i \geq \left(\frac{1}{2\pi} \int_0^{2\pi} |Q_i(e^{j\omega})|^2 \, d\omega \right)^2. \tag{11B-10}$$

Noting that $Q_i(e^{j\omega})$ is a normalized eigenfilter in the sense that $\mathbf{q}_i^T \mathbf{q}_i = 1$, the right-hand side of (11B-10) is always equal to unity (see Chapter 4). Using this result in (11B-4) and recalling the definition of the parameter γ, we obtain

$$\gamma \geq 1. \tag{11B-11}$$

A particular case of interest for which the inequality (11B-10) (and thus (11B-11)) will be converted to equality is when $|Q_i(e^{j\omega})|^2$ is an impulse function in the form $2\pi\delta(\omega - \omega_i)$. In fact, this happens to be nearly the case as the filter length, N, increases to a large value. With this argument we can say that the above inequalities will all be close to equalities when the filter length, N, is large.

12
Method of Least Squares

The problem of filter design for estimating a desired signal based on another signal can be formulated from either a statistical or deterministic point of view, as was mentioned in Chapter 1. The Wiener filter and its adaptive version (the LMS algorithm and its derivatives) belong to the statistical framework since their design is based on minimizing a statistical quantity, *the mean-square error*. So far, all our discussions have been limited to this statistical class of algorithms. In the next two chapters we are going to consider the second class of algorithms which are derived based on the *method of least squares* which belongs to the deterministic framework. We have noted that the class of LMS-based algorithms is very wide and covers a large variety of algorithms, each having some merits over the others. The class of least-squares-based algorithms is also equally wide. The current literature contains a large number of scientific papers that report a diverse range of least-squares-based adaptive filtering algorithms.

We recall that in the derivation of the LMS algorithm the goal was to minimize the *mean square* of the estimation error. In the *method of least squares,* on the other hand, at any time instant $n > 0$ the adaptive filter parameters (tap weights) are calculated so that the quantity

$$\zeta(n) = \sum_{k=1}^{n} \rho_n(k) e_n^2(k) \tag{12.1}$$

is minimized, and hence the name *least squares*. In (12.1), $k = 1$ is the time at which the algorithm starts, $e_n(k)$, for $k = 1, 2, \ldots, n$, are the samples of error estimates that would be obtained if the filter were run from time $k = 1$ to n, using the set of filter parameters that are computed at time n, and $\rho_n(k)$ is a weighting function whose role will be discussed later. Thus, in the method of least squares the filter parameters are optimized by using all the observations from the time the filter begins until the present time and minimizing the sum of squared values of the error samples of the filter output. Clearly, this is a deterministic optimization of the filter parameters, based on the observed data.

An insightful interpretation of the method of least squares is its *curve-fitting* property. Consider a curve whose samples are the desired output samples of the adaptive filter. In the same manner, samples of the filter output (given some input sequence) can be considered to constitute another curve. Then, the problem of choosing the filter parameters to find the best fit between these two curves boils down to the method of

least squares if we define the *best fit* as one that minimizes a weighted sum of squared values of the differences between the samples of the two curves.

In this book, our discussion of the method of least squares is rather limited. In this chapter we first present a formulation of the problem of least squares for a linear combiner and discuss some of its properties. We also introduce the *standard recursive least-squares* (RLS) *algorithm* as an example of the class of least-squares-based adaptive filtering algorithms. Some results which compare the LMS and RLS algorithms are also given in this chapter. In the next chapter we present the development of *fast RLS algorithms*, which are computationally more efficient than the standard RLS algorithm, for recursive implementation of the method of least squares.

12.1 Formulation of the Least-Squares Estimation for a Linear Combiner

Consider a linear adaptive filter with the observed real-valued input vector $\mathbf{x}(n) = [x_0(n)\ x_1(n)\ \ldots\ x_{N-1}(n)]^T$, tap-weight vector $\mathbf{w}(n) = [w_0(n)\ w_1(n)\ \ldots\ w_{N-1}(n)]^T$, and desired output $d(n)$. The filter output is obtained as the inner product of $\mathbf{w}(n)$ and $\mathbf{x}(n)$, i.e. $\mathbf{w}^T(n)\mathbf{x}(n)$. Note that, here, we have not specified any particular structure for the elements of the input vector $\mathbf{x}(n)$. The elements of $\mathbf{x}(n)$ may be successive samples of a particular input process, as happens in the case of transversal filters, or may be samples of a parallel set of input sources, as in the case of antenna arrays.

In the *method of least squares*, at time instant n we choose $\mathbf{w}(n)$ so that the summation (12.1) is minimized. We define

$$y_n(k) = \mathbf{w}^T(n)\mathbf{x}(k), \quad \text{for } k = 1, 2, \ldots, n, \tag{12.2}$$

as the filter output generated by using the tap-weight vector $\mathbf{w}(n)$. The corresponding estimation error would then be

$$e_n(k) = d(k) - y_n(k). \tag{12.3}$$

Thus, we note from (12.1) and (12.2) that the addition of the subscript n to the samples of the filter output, $y_n(k)$, and the error estimates, $e_n(k)$, is to emphasize that these quantities are computed using the solution $\mathbf{w}(n)$, at instant n, which is obtained by minimizing the weighted sum of error squares over *all* the instants upto n.

To keep our derivations simple we assume that the weighting function $\rho_n(k)$ is equal to one, for all values of k in the first three sections of this chapter. We also adopt a matrix/vector formulation of the problem.

We define the following vectors:

$$\mathbf{d}(n) = [d(1)\ d(2)\ \ldots\ d(n)]^T, \tag{12.4}$$

$$\mathbf{y}(n) = [y_n(1)\ y_n(2)\ \ldots\ y_n(n)]^T \tag{12.5}$$

and

$$\mathbf{e}(n) = [e_n(1)\ e_n(2)\ \ldots\ e_n(n)]^T. \tag{12.6}$$

We also define the matrix of observed input samples as

$$\mathbf{X}(n) = [\mathbf{x}(1)\ \mathbf{x}(2)\ \ldots\ \mathbf{x}(n)]. \tag{12.7}$$

Then, using (12.2) and (12.3) in (12.4)–(12.7), we get

$$\mathbf{y}(n) = \mathbf{X}^{\mathrm{T}}(n)\mathbf{w}(n) \tag{12.8}$$

and

$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{y}(n). \tag{12.9}$$

Furthermore, with $\rho_n(k) = 1$, for all k, (12.1) can be written as

$$\zeta(n) = \mathbf{e}^{\mathrm{T}}(n)\mathbf{e}(n). \tag{12.10}$$

Substituting (12.8) and (12.9) in (12.10) we obtain

$$\zeta(n) = \mathbf{d}^{\mathrm{T}}(n)\mathbf{d}(n) - 2\boldsymbol{\theta}^{\mathrm{T}}(n)\mathbf{w}(n) + \mathbf{w}^{\mathrm{T}}(n)\boldsymbol{\Psi}(n)\mathbf{w}(n), \tag{12.11}$$

where

$$\boldsymbol{\Psi}(n) = \mathbf{X}(n)\mathbf{X}^{\mathrm{T}}(n) \tag{12.12}$$

and

$$\boldsymbol{\theta}(n) = \mathbf{X}(n)\mathbf{d}(n). \tag{12.13}$$

Setting the gradient of $\zeta(n)$ with respect to the tap-weight vector $\mathbf{w}(n)$ equal to zero and following the same line of derivations as in the case of Wiener filters (Chapter 3) we obtain

$$\boldsymbol{\Psi}(n)\hat{\mathbf{w}}(n) = \boldsymbol{\theta}(n), \tag{12.14}$$

where $\hat{\mathbf{w}}(n)$ is the estimate of filter tap-weight vector in the *least-squares sense*. Equation (12.14) is known as the *normal equation for a linear least-squares filter*. It results in the following least-squares solution:

$$\hat{\mathbf{w}}(n) = \boldsymbol{\Psi}^{-1}(n)\boldsymbol{\theta}(n). \tag{12.15}$$

Substituting (12.15) in (12.11), the minimum value of $\zeta(n)$ is obtained as

$$\begin{aligned}\zeta_{\min}(n) &= \mathbf{d}^{\mathrm{T}}(n)\mathbf{d}(n) - \boldsymbol{\theta}^{\mathrm{T}}(n)\boldsymbol{\Psi}^{-1}(n)\boldsymbol{\theta}(n) \\ &= \mathbf{d}^{\mathrm{T}}(n)\mathbf{d}(n) - \boldsymbol{\theta}^{\mathrm{T}}(n)\hat{\mathbf{w}}(n).\end{aligned} \tag{12.16}$$

12.2 The Principle of Orthogonality

We recall that in the case of Wiener filters the optimized output error, $e_o(n)$, is orthogonal to the filter tap inputs, in the sense that the following identities hold:

$$\mathrm{E}[e_o(n)x_i(n)] = 0, \quad \text{for } i = 0, 1, \ldots, N-1 \tag{12.17}$$

where $x_i(n)$ is the ith element of the tap-input vector $\mathbf{x}(n)$, and $\mathrm{E}[\cdot]$ denotes statistical expectation. This was called the *principle of orthogonality* for Wiener filters. Similar result can also be derived in the case of linear least-squares estimation by following the same line of derivations as those given in Chapter 3 (Section 3.3).

Using (12.6) and (12.10) we obtain

$$\frac{\partial \zeta(n)}{\partial w_i(n)} = 2\sum_{k=1}^{n} e_n(k)\frac{\partial e_n(k)}{\partial w_i(n)}. \tag{12.18}$$

Using the identity

$$e_n(k) = d(k) - \sum_{i=0}^{N-1} w_i(n)x_i(k) \tag{12.19}$$

to evaluate the second factor on the right-hand side of (12.18), we get

$$\frac{\partial \zeta(n)}{\partial w_i(n)} = -2\sum_{k=1}^{n} e_n(k)x_i(k). \tag{12.20}$$

Furthermore, we note that when $\mathbf{w}(n) = \hat{\mathbf{w}}(n)$,

$$\frac{\partial \zeta(n)}{\partial w_i(n)} = 0, \quad \text{for } i = 0, 1, \ldots, N-1. \tag{12.21}$$

Using (12.20) and (12.21) we find that when $\mathbf{w}(n) = \hat{\mathbf{w}}(n)$, the following identities hold:

$$\sum_{k=1}^{n} \hat{e}_n(k)x_i(k) = 0, \quad \text{for } i = 0, 1, \ldots, N-1, \tag{12.22}$$

where $\hat{e}_n(k)$ is the optimized estimation error in the least-squares sense. This result, which is equivalent to (12.17), is known as the *principle of orthogonality* in the least-squares formulation. We define the vectors

$$\hat{\mathbf{e}}(n) = [\hat{e}_n(1) \;\; \hat{e}_n(2) \;\; \ldots \;\; \hat{e}_n(n)]^{\mathrm{T}} \tag{12.23}$$

and

$$\mathbf{x}_i(n) = [x_i(1) \;\; x_i(2) \;\; \ldots \;\; x_i(n)]^{\mathrm{T}} \tag{12.24}$$

and note that (12.22) may also be expressed in terms of these vectors as

$$\hat{\mathbf{e}}^{\mathrm{T}}(n)\mathbf{x}_i(n) = 0, \quad \text{for } i = 0, 1, \ldots, N-1. \tag{12.25}$$

[redacted]

A comparison of (12.17) and (12.25) reveals that the definition of orthogonality is in terms of statistical averages in Wiener filtering, whereas it is in terms of the inner products of data vectors in the case of least-squares estimation. By dividing both sides of

and

$$\boldsymbol{\theta}(3) = \mathbf{X}(3)\mathbf{d}(3) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Thus,

$$\hat{\mathbf{w}}(3) = \begin{bmatrix} 5 & 4 \\ 4 & 5.01 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{9.05} \begin{bmatrix} 5.01 & -4 \\ -4 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{9.05} \begin{bmatrix} 9.01 \\ -9 \end{bmatrix},$$

$$\hat{\mathbf{y}}(3) = \hat{w}_0(3)\mathbf{x}_0(3) + \hat{w}_1(3)\mathbf{x}_1(3)$$

$$= \frac{9.01}{9.05} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} - \frac{9}{9.05} \begin{bmatrix} 1 \\ 2 \\ 0.1 \end{bmatrix} = \frac{1}{9.05} \begin{bmatrix} 9.02 \\ -8.99 \\ -0.9 \end{bmatrix}$$

and

$$\hat{\mathbf{e}}(3) = \mathbf{d}(3) - \hat{\mathbf{y}}(3) = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} - \frac{1}{9.05} \begin{bmatrix} 9.02 \\ -8.99 \\ -0.9 \end{bmatrix} = \frac{1}{9.05} \begin{bmatrix} 0.03 \\ -0.06 \\ 0.90 \end{bmatrix}.$$

We can now confirm the principle of orthogonality by noting that

$$\hat{\mathbf{e}}^{\mathrm{T}}(3)\mathbf{x}_0(3) = 0$$

and also that

$$\hat{\mathbf{e}}^{\mathrm{T}}(3)\mathbf{x}_1(3) = 0.$$

12.3 Projection Operator

An alternative interpretation to the solution of the least-squares problem can be given using the concept of a *projection operator*. Projection of a $1 \times n$ vector $\mathbf{d}(n)$ into the subspace spanned by a set of vectors $\mathbf{x}_0(n), \mathbf{x}_1(n), \ldots, \mathbf{x}_{N-1}(n)$ is a vector $\hat{\mathbf{d}}(n)$ with the following properties:

1. The vector $\hat{\mathbf{d}}(n)$ is obtained as a linear combination of the vectors $\mathbf{x}_0(n)$, $\mathbf{x}_1(n), \ldots, \mathbf{x}_{N-1}(n)$.
2. Among all the vectors in the subspace spanned by $\mathbf{x}_0(n)$, $\mathbf{x}_1(n), \ldots, \mathbf{x}_{N-1}(n)$, the vector $\hat{\mathbf{d}}(n)$ has the minimum Euclidian distance from $\mathbf{d}(n)$.
3. The difference $\mathbf{d}(n) - \hat{\mathbf{d}}(n)$ is a vector that is orthogonal to the subspace spanned by $\mathbf{x}_0(n), \mathbf{x}_1(n), \ldots, \mathbf{x}_{N-1}(n)$.

We may note that the least-squares estimate $\hat{\mathbf{y}}(n)$ satisfies the three properties listed above. Namely, we note from (12.28) that $\hat{\mathbf{y}}(n)$ is also obtained as a linear combination of the vectors $\mathbf{x}_0(n), \mathbf{x}_1(n), \ldots, \mathbf{x}_{N-1}(n)$. Furthermore, obtaining $\hat{\mathbf{y}}(n)$ by minimizing $\hat{\mathbf{e}}^{\mathrm{T}}(n)\hat{\mathbf{e}}(n)$, where $\hat{\mathbf{e}}(n) = \mathbf{d}(n) - \hat{\mathbf{y}}(n)$, is equivalent to minimizing the Euclidian distance between $\mathbf{d}(n)$ and $\hat{\mathbf{y}}(n)$. Also, from the principle of orthogonality, the error vector $\hat{\mathbf{e}}(n) = \mathbf{d}(n) - \hat{\mathbf{y}}(n)$ is orthogonal to the vectors $\mathbf{x}_0(n), \mathbf{x}_1(n), \ldots, \mathbf{x}_{N-1}(n)$. We thus

conclude that $\hat{\mathbf{y}}(n)$ *is nothing but the projection of* $\mathbf{d}(n)$ *into the subspace spanned by the vectors* $\mathbf{x}_0(n), \mathbf{x}_1(n), \ldots, \mathbf{x}_{N-1}(n)$.

We also note that from (12.8)

$$\hat{\mathbf{y}}(n) = \mathbf{X}^{\mathrm{T}}(n)\hat{\mathbf{w}}(n). \tag{12.29}$$

Substituting (12.12) and (12.13) in (12.15) and the result in (12.29) we obtain

$$\hat{\mathbf{y}}(n) = \mathbf{P}(n)\mathbf{d}(n), \tag{12.30}$$

where

$$\mathbf{P}(n) = \mathbf{X}^{\mathrm{T}}(n)\big(\mathbf{X}(n)\mathbf{X}^{\mathrm{T}}(n)\big)^{-1}\mathbf{X}(n). \tag{12.31}$$

Consequently, the matrix $\mathbf{P}(n)$ is known as the *projection operator.*

Using (12.30), we find that the optimized error vector $\hat{\mathbf{e}}(n) = \mathbf{d}(n) - \hat{\mathbf{y}}(n)$ can be expressed as

$$\hat{\mathbf{e}}(n) = [\mathbf{I} - \mathbf{P}(n)]\mathbf{d}(n), \tag{12.32}$$

where $\mathbf{I}$ is the identity matrix of the same dimension as $\mathbf{P}(n)$. As a result, the matrix $\mathbf{I} - \mathbf{P}(n)$ is referred to as the *orthogonal complement projection operator.*

12.4 The Standard Recursive Least-Squares Algorithm

The least-squares solution provided by (12.15) is of very little interest in the actual implementation of adaptive filters, since it requires that all the past samples of the input as well as the desired output be available at every iteration. Furthermore, the number of operations needed to calculate $\hat{\mathbf{w}}(n)$ grows proportional to n, as the number of columns of $\mathbf{X}(n)$ and the length of $\mathbf{d}(n)$ grow with n. These problems are solved by employing recursive methods. In this section, as an example of recursive methods, we present the *standard recursive least-squares (RLS) algorithm.*

12.4.1 RLS recursions

In the standard RLS algorithm (or just 'RLS' algorithm, for short), the weighting factor $\rho_n(k)$ is chosen as

$$\rho_n(k) = \lambda^{n-k}, \quad k = 1, 2, \ldots, n, \tag{12.33}$$

where λ is a positive constant close to, but smaller than, one. The ordinary method of least squares, discussed in the previous sections, corresponds to the case of $\lambda = 1$. The parameter λ is known as the *forgetting factor.* Clearly, when $\lambda < 1$, the weighting factors defined by (12.33) give more weight to the recent samples of the error estimates (and thus to the recent samples of the observed data) compared with the old ones. In other words, *the choice of* $\lambda < 1$ *results in a scheme that puts more emphasis on the recent samples of the observed data and tends to forget the past.* This is exactly what we may wish when we

develop an adaptive algorithm with some tracking capability. Roughly speaking, $1/(1-\lambda)$ is a measure of the *memory* of the algorithm. The case of $\lambda = 1$ corresponds to *infinite memory*.

Substituting (12.33) in (12.1) and using the vector/matrix notations of Section 12.1 we obtain

$$\zeta(n) = \mathbf{e}^{\mathrm{T}}(n)\mathbf{\Lambda}(n)\mathbf{e}(n), \tag{12.34}$$

where $\mathbf{\Lambda}(n)$ is the diagonal matrix consisting of the weighting factors $1, \lambda, \lambda^2, \ldots$, i.e.

$$\mathbf{\Lambda}(n) = \begin{bmatrix} \lambda^{n-1} & 0 & 0 & \cdots & 0 \\ 0 & \lambda^{n-2} & 0 & \cdots & 0 \\ 0 & 0 & \lambda^{n-3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{12.35}$$

Following the same line of derivations which led to (12.15) we obtain the minimizer of $\zeta(n)$ in (12.34) as

$$\hat{\mathbf{w}}(n) = \mathbf{\Psi}_{\lambda}^{-1}(n)\boldsymbol{\theta}_{\lambda}(n), \tag{12.36}$$

where

$$\mathbf{\Psi}_{\lambda}(n) = \mathbf{X}(n)\mathbf{\Lambda}(n)\mathbf{X}^{\mathrm{T}}(n) \tag{12.37}$$

and

$$\boldsymbol{\theta}_{\lambda}(n) = \mathbf{X}(n)\mathbf{\Lambda}(n)\mathbf{d}(n). \tag{12.38}$$

On substituting (12.4) and (12.7) in (12.37) and (12.38), and expanding the summations, we get

$$\mathbf{\Psi}_{\lambda}(n) = \mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n) + \lambda\mathbf{x}(n-1)\mathbf{x}^{\mathrm{T}}(n-1) + \lambda^2\mathbf{x}(n-2)\mathbf{x}^{\mathrm{T}}(n-2) + \ldots \tag{12.39}$$

and

$$\boldsymbol{\theta}_{\lambda}(n) = \mathbf{x}(n)d(n) + \lambda\mathbf{x}(n-1)d(n-1) + \lambda^2\mathbf{x}(n-2)d(n-2) + \ldots, \tag{12.40}$$

respectively. Using (12.39) and (12.40), it is straightforward to see that $\mathbf{\Psi}_{\lambda}(n)$ and $\boldsymbol{\theta}_{\lambda}(n)$ can be obtained recursively as

$$\mathbf{\Psi}_{\lambda}(n) = \lambda\mathbf{\Psi}_{\lambda}(n-1) + \mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n) \tag{12.41}$$

and

$$\boldsymbol{\theta}_{\lambda}(n) = \lambda\boldsymbol{\theta}_{\lambda}(n-1) + \mathbf{x}(n)d(n), \tag{12.42}$$

respectively. These two recursions and the following result from matrix algebra form the *basis* for the derivation of the RLS algorithm.

For an arbitrary non-singular $N \times N$ matrix $\mathbf{A}$, any $N \times 1$ vector $\mathbf{a}$ and a scalar α,

$$(\mathbf{A} + \alpha\mathbf{a}\mathbf{a}^{\mathrm{T}})^{-1} = \mathbf{A}^{-1} - \frac{\alpha\mathbf{A}^{-1}\mathbf{a}\mathbf{a}^{\mathrm{T}}\mathbf{A}^{-1}}{1 + \alpha\mathbf{a}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{a}}. \tag{12.43}$$

This identity, which was also used for some other derivations in Chapter 6, is a special form of the *matrix inversion lemma* (see page 153).

We let $\mathbf{A} = \lambda\boldsymbol{\Psi}_{\lambda}(n-1)$, $\mathbf{a} = \mathbf{x}(n)$ and $\alpha = 1$ to evaluate the inverse of $\boldsymbol{\Psi}_{\lambda}(n) = \lambda\boldsymbol{\Psi}_{\lambda}(n-1) + \mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)$. This results in the following recursive equation for updating the inverse of $\boldsymbol{\Psi}_{\lambda}(n)$:

$$\boldsymbol{\Psi}_{\lambda}^{-1}(n) = \lambda^{-1}\boldsymbol{\Psi}_{\lambda}^{-1}(n-1) - \frac{\lambda^{-2}\boldsymbol{\Psi}_{\lambda}^{-1}(n-1)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\boldsymbol{\Psi}_{\lambda}^{-1}(n-1)}{1 + \lambda^{-1}\mathbf{x}^{\mathrm{T}}(n)\boldsymbol{\Psi}_{\lambda}^{-1}(n-1)\mathbf{x}(n)}. \tag{12.44}$$

To simplify the subsequent steps, we define the column vector

$$\mathbf{k}(n) = \frac{\lambda^{-1}\boldsymbol{\Psi}_{\lambda}^{-1}(n-1)\mathbf{x}(n)}{1 + \lambda^{-1}\mathbf{x}^{\mathrm{T}}(n)\boldsymbol{\Psi}_{\lambda}^{-1}(n-1)\mathbf{x}(n)}. \tag{12.45}$$

The vector $\mathbf{k}(n)$ is referred to as the *gain vector* for reasons that will become apparent later in this section.

Substituting (12.45) in (12.44) we obtain

$$\boldsymbol{\Psi}_{\lambda}^{-1}(n) = \lambda^{-1}\big(\boldsymbol{\Psi}_{\lambda}^{-1}(n-1) - \mathbf{k}(n)\mathbf{x}^{\mathrm{T}}(n)\boldsymbol{\Psi}_{\lambda}^{-1}(n-1)\big). \tag{12.46}$$

By rearranging (12.45) we get

$$\mathbf{k}(n) = \lambda^{-1}\big(\boldsymbol{\Psi}_{\lambda}^{-1}(n-1) - \mathbf{k}(n)\mathbf{x}^{\mathrm{T}}(n)\boldsymbol{\Psi}_{\lambda}^{-1}(n-1)\big)\mathbf{x}(n). \tag{12.47}$$

Using (12.46) in (12.47) we obtain

$$\mathbf{k}(n) = \boldsymbol{\Psi}_{\lambda}^{-1}(n)\mathbf{x}(n). \tag{12.48}$$

Next, we substitute (12.42) in (12.36) and expand to obtain

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \lambda\boldsymbol{\Psi}_{\lambda}^{-1}(n)\boldsymbol{\theta}_{\lambda}(n-1) + \boldsymbol{\Psi}_{\lambda}^{-1}(n)\mathbf{x}(n)d(n) \\ &= \lambda\boldsymbol{\Psi}_{\lambda}^{-1}(n)\boldsymbol{\theta}_{\lambda}(n-1) + \mathbf{k}(n)d(n),\end{aligned} \tag{12.49}$$

where the last equality is obtained by using (12.48). Substituting (12.46) in (12.49) we get

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \boldsymbol{\Psi}_{\lambda}^{-1}(n-1)\boldsymbol{\theta}_{\lambda}(n-1) - \mathbf{k}(n)\mathbf{x}^{\mathrm{T}}(n)\boldsymbol{\Psi}_{\lambda}^{-1}(n-1)\boldsymbol{\theta}_{\lambda}(n-1) + \mathbf{k}(n)d(n) \\ &= \hat{\mathbf{w}}(n-1) - \mathbf{k}(n)\mathbf{x}^{\mathrm{T}}(n)\hat{\mathbf{w}}(n-1) + \mathbf{k}(n)d(n) \\ &= \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)(d(n) - \hat{\mathbf{w}}^{\mathrm{T}}(n-1)\mathbf{x}(n)).\end{aligned} \tag{12.50}$$

Finally, we define

$$\hat{e}_{n-1}(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{T}}(n-1)\mathbf{x}(n) \tag{12.51}$$

and use this in (12.50) to obtain

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\hat{e}_{n-1}(n). \tag{12.52}$$

This is the recursion used by the RLS algorithm to update $\hat{\mathbf{w}}(n)$. The amount of change to be made in the tap weights at the nth iteration is determined by the product of the *estimation error* $\hat{e}_{n-1}(n)$ and the *gain vector* $\mathbf{k}(n)$.

From (12.51) we note that $\hat{e}_{n-1}(n)$ is the estimation error at time n based on the tap-weight vector estimated at time $n-1$, $\hat{\mathbf{w}}(n-1)$. Hence, $\hat{e}_{n-1}(n)$ is referred to as the a priori estimation error. On the other hand, the a posteriori estimation error is given by

$$\hat{e}_n(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{x}(n), \tag{12.53}$$

which would be obtained if the *current* least-squares estimate of the filter tap weights, i.e. $\hat{\mathbf{w}}(n)$, were used to calculate the filter output.

Equations (12.45), (12.51), (12.52) and (12.46), in this order, describe one iteration of the standard RLS algorithm.

12.4.2 Initialization of the RLS algorithm

Actual implementation of the RLS algorithm requires proper initialization of $\boldsymbol{\Psi}_\lambda(0)$ and $\hat{\mathbf{w}}(0)$ prior to the start of the algorithm. In particular, we note that the matrix

$$\boldsymbol{\Psi}_\lambda(n) = \sum_{k=1}^{n} \lambda^{n-k}\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k), \tag{12.54}$$

for values of n smaller than the filter length, N, has a rank that is less than its dimension, N. This implies that the inverse of $\boldsymbol{\Psi}_\lambda(n)$ does not exist for $n < N$. A simple and commonly used solution to this problem is to start the RLS algorithm with an initial setting of

$$\boldsymbol{\Psi}_\lambda(0) = \delta\mathbf{I}, \tag{12.55}$$

where δ is a small positive constant. Then, iterating the recursive equation (12.41) we obtain

$$\boldsymbol{\Psi}_\lambda(n) = \sum_{k=1}^{n} \lambda^{n-k}\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k) + \delta\lambda^n\mathbf{I}. \tag{12.56}$$

We observe that, for $\lambda < 1$, the effect of $\boldsymbol{\Psi}_\lambda(0)$ reduces exponentially as n increases. Thus, this initialization of $\boldsymbol{\Psi}_\lambda(0)$ has very little effect on the steady-state performance of the RLS algorithm. Furthermore, the effect of $\boldsymbol{\Psi}_\lambda(0)$ on the convergence behaviour of the RLS algorithm can be minimized by choosing a very small value for δ.

As for the initialization of the filter tap weights, it is common practice to set

$$\hat{\mathbf{w}}(0) = \mathbf{0}, \tag{12.57}$$

where $\mathbf{0}$ is the $N \times 1$ zero vector. However, setting $\hat{\mathbf{w}}(0)$ equal to an arbitrary non-zero vector also, does not result in any significant effect on the convergence and steady-state behaviour of the RLS algorithm, provided that the elements of $\hat{\mathbf{w}}(0)$ are not very large. A study of the effect of a non-zero selection of $\hat{\mathbf{w}}(0)$ is discussed in Problem P12.6.

12.4.3 Summary of the standard RLS algorithm

Table 12.1 summarizes the standard RLS algorithm. This is one of the few possible implementations of the RLS algorithm. It exploits the special form of the gain vector, $\mathbf{k}(n)$, to simplify its computation by using an intermediate vector, $\mathbf{u}(n) = \boldsymbol{\Psi}_\lambda^{-1}(n-1)\mathbf{x}(n)$. We also note that by multiplying the numerator and denominator of the right-hand side of (12.45) by λ and using this definition of $\mathbf{u}(n)$ we obtain

$$\mathbf{k}(n) = \frac{1}{\lambda + \mathbf{x}^{\mathrm{T}}(n)\mathbf{u}(n)}\mathbf{u}(n).$$

Careful examination of Table 12.1 reveals that the computational complexity of this implementation is mainly determined by:

1. Computation of the vector $\mathbf{u}(n)$.
2. Computation of $\mathbf{x}^{\mathrm{T}}(n)\boldsymbol{\Psi}_\lambda^{-1}(n-1)$ in the $\boldsymbol{\Psi}_\lambda^{-1}(n)$ update equation.

Table 12.1 Summary of the standard RLS algorithm (version I)

Input:	Tap-weight vector estimate, $\hat{\mathbf{w}}(n-1)$, Input vector, $\mathbf{x}(n)$, desired output, $d(n)$, and the matrix $\boldsymbol{\Psi}_\lambda^{-1}(n-1)$.
Output:	Filter output, $\hat{y}_{n-1}(n)$, Tap-weight vector update, $\hat{\mathbf{w}}(n)$, and the updated matrix $\boldsymbol{\Psi}_\lambda^{-1}(n)$.

1. Computation of the gain vector:
$$\mathbf{u}(n) = \boldsymbol{\Psi}_\lambda^{-1}(n-1)\mathbf{x}(n)$$
$$\mathbf{k}(n) = \frac{1}{\lambda + \mathbf{x}^{\mathrm{T}}(n)\mathbf{u}(n)}\mathbf{u}(n)$$
2. Filtering:
$$\hat{y}_{n-1}(n) = \hat{\mathbf{w}}^{\mathrm{T}}(n-1)\mathbf{x}(n)$$
3. Error estimation:
$$\hat{e}_{n-1}(n) = d(n) - \hat{y}_{n-1}(n)$$
4. Tap-weight vector adaptation:
$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\hat{e}_{n-1}(n)$$
5. $\boldsymbol{\Psi}_\lambda^{-1}(n)$ update:
$$\boldsymbol{\Psi}_\lambda^{-1}(n) = \lambda^{-1}(\boldsymbol{\Psi}_\lambda^{-1}(n-1) - \mathbf{k}(n)[\mathbf{x}^{\mathrm{T}}(n)\boldsymbol{\Psi}_\lambda^{-1}(n-1)])$$

3. Computation of the outer product of $\mathbf{k}(n)$ and $\mathbf{x}^T(n)\boldsymbol{\Psi}_\lambda^{-1}(n-1)$ in the $\boldsymbol{\Psi}_\lambda^{-1}(n)$ update equation.
4. Subtraction of the two terms within brackets in the $\boldsymbol{\Psi}_\lambda^{-1}(n)$ update equation, and scaling of the result by λ^{-1}.

Each of these steps requires N^2 multiplications. In addition, Steps 1, 2 and 4 require N^2 additions/subtractions each. This brings the total computational complexity of the RLS algorithm of Table 12.1 to about $4N^2$ multiplications and $3N^2$ additions/subtractions.

The fact that $\boldsymbol{\Psi}_\lambda(n)$ is a symmetric matrix can be used to reduce the computational complexity of the RLS algorithm. Using this, we find that $\mathbf{x}^T(n)\boldsymbol{\Psi}_\lambda^{-1}(n-1) = (\boldsymbol{\Psi}_\lambda^{-1}(n-1)\mathbf{x}(n))^T = \mathbf{u}^T(n)$. The last step of Table 12.1 may then be simplified as

$$\boldsymbol{\Psi}_\lambda^{-1}(n) = \lambda^{-1}(\boldsymbol{\Psi}_\lambda^{-1}(n-1) - \mathbf{k}(n)\mathbf{u}^T(n)). \tag{12.58}$$

This amendment, although logical and precise, results in a useless implementation when applied in practice. Computer simulations and theoretical analysis (Verhaegen, 1989) show that this amended version of the RLS algorithm is numerically unstable. This behaviour of the RLS algorithm is due to the round-off error accumulation that makes $\boldsymbol{\Psi}_\lambda^{-1}(n-1)$ non-symmetric. This in turn invalidates the assumption $\mathbf{x}^T(n)\boldsymbol{\Psi}_\lambda^{-1}(n-1) = \mathbf{u}^T(n)$ that was used to introduce the above amendment.

To resolve this problem and come up with an efficient and stable implementation of the RLS algorithm, we may compute only the upper or lower triangular part of $\boldsymbol{\Psi}_\lambda^{-1}(n)$ according to (12.58), and copy the result to obtain the rest of the elements of $\boldsymbol{\Psi}_\lambda^{-1}(n)$ to preserve its symmetric structure (Verhaegen, 1989, and Yang, 1994). Table 12.2 summarizes this implementation of the RLS algorithm. Here, the operator $\text{Tri}\{\cdot\}$

Table 12.2 Summary of the standard RLS algorithm (version II)

Input:	Tap-weight vector estimate, $\hat{\mathbf{w}}(n-1)$, Input vector, $\mathbf{x}(n)$, desired output, $d(n)$, and the matrix $\boldsymbol{\Psi}_\lambda^{-1}(n-1)$.
Output:	Filter output, $\hat{y}_{n-1}(n)$, Tap-weight vector update, $\hat{\mathbf{w}}(n)$, and the updated matrix $\boldsymbol{\Psi}_\lambda^{-1}(n)$.

1. Computation of the gain vector:
$$\mathbf{u}(n) = \boldsymbol{\Psi}_\lambda^{-1}(n-1)\mathbf{x}(n)$$
$$\mathbf{k}(n) = \frac{1}{\lambda + \mathbf{x}^T(n)\mathbf{u}(n)}\mathbf{u}(n)$$
2. Filtering:
$$\hat{y}_{n-1}(n) = \hat{\mathbf{w}}^T(n-1)\mathbf{x}(n)$$
3. Error estimation:
$$\hat{e}_{n-1}(n) = d(n) - \hat{y}_{n-1}(n)$$
4. Tap-weight vector adaptation:
$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\hat{e}_{n-1}(n)$$
5. $\boldsymbol{\Psi}_\lambda^{-1}(n)$ update:
$$\boldsymbol{\Psi}_\lambda^{-1}(n) = \text{Tri}\{\lambda^{-1}(\boldsymbol{\Psi}_\lambda^{-1}(n-1) - \mathbf{k}(n)\mathbf{u}^T(n))\}$$

signifies that the computation of $\Psi_\lambda^{-1}(n)$ is based on either the upper or lower triangular parts. Clearly, this results in a significant saving in computations since a large portion of the complexity of the RLS algorithm arises from the computation of $\Psi_\lambda^{-1}(n)$. Basically, the computational complexity of Steps 3 and 4 above (corresponding to $\Psi_\lambda^{-1}(n)$ update) are halved and Step 2 is eliminated. This brings down the computational complexity of the RLS algorithm in Table 12.2 to about $2N^2$ multiplications and $1.5N^2$ additions/subtractions, which is about half of that of the algorithm of Table 12.1.

From the above discussion we may perceive the potential problem of the RLS algorithm. It is indeed true that *round-off errors may accumulate and result in undesirable behaviour for any algorithm that works based on some recursive update equations.* This statement is general and applicable to all LMS and least-squares-based algorithms. However, the problem turns out to be more serious in the case of least-squares-based algorithms. See Cioffi (1987a) for an excellent qualitative discussion of the round-off error in various adaptive filtering algorithms. *Engineers who use adaptive filtering algorithms should be aware of this potential problem and must evaluate the algorithms on this issue before going for their practical implementations.*

12.5 The Convergence Behaviour of the RLS Algorithm

In this section we study the convergence behaviour of the RLS algorithm in the context of a system modelling problem. As the plant, we consider a linear multiple regressor characterized by the equation

$$d(n) = \mathbf{w}_o^T \mathbf{x}(n) + e_o(n), \tag{12.59}$$

where $\mathbf{w}_o$ is the regressor tap-weight vector, $\mathbf{x}(n)$ is the tap-input vector, $e_o(n)$ is the plant noise, and $d(n)$ is the plant output. The noise samples, $e_o(n)$, are assumed to be zero-mean and white, and independent of the input samples, $\mathbf{x}(n)$. The tap-input vector, $\mathbf{x}(n)$, is also applied to an adaptive filter whose tap-weight vector, $\mathbf{w}(n)$, is adapted so that the difference between its output, $y(n) = \mathbf{w}^T(n)\mathbf{x}(n)$, and the plant output, $d(n)$, is minimized in the least-squares sense.

The derivations that follow use the vector/matrix formulation adopted in the previous sections. In particular, we note that with the definitions (12.4) and (12.7),

$$\mathbf{d}(n) = \mathbf{X}^T(n)\mathbf{w}_o + \mathbf{e}_o(n), \tag{12.60}$$

where $\mathbf{e}_o(n) = [e_o(1)\ \ e_o(2)\ \ \ldots\ \ e_o(n)]^T$.

12.5.1 Average tap-weight behaviour of the RLS algorithm

We show that the least-squares estimate $\hat{\mathbf{w}}(n)$ is an unbiased estimate of the tap-weight vector $\mathbf{w}_o$.

From (12.36), (12.37) and (12.38) we obtain

$$\hat{\mathbf{w}}(n) = \left(\mathbf{X}(n)\mathbf{\Lambda}(n)\mathbf{X}^T(n)\right)^{-1}\mathbf{X}(n)\mathbf{\Lambda}(n)\mathbf{d}(n). \tag{12.61}$$

Substituting (12.60) in (12.61), and using (12.37), we get

$$\hat{\mathbf{w}}(n) = \mathbf{w}_o + \boldsymbol{\Psi}_\lambda^{-1}(n)\mathbf{X}(n)\boldsymbol{\Lambda}(n)\mathbf{e}_o(n). \tag{12.62}$$

Taking expectations on both sides of (12.62) and recalling that $\mathbf{X}(n)$ and $\mathbf{e}_o(n)$ are independent of each other we obtain

$$\begin{aligned} \mathrm{E}[\hat{\mathbf{w}}(n)] &= \mathbf{w}_o + \mathrm{E}[\boldsymbol{\Psi}_\lambda^{-1}(n)\mathbf{X}(n)]\boldsymbol{\Lambda}(n)\mathrm{E}[\mathbf{e}_o(n)] \\ &= \mathbf{w}_o, \end{aligned} \tag{12.63}$$

where the last equality follows from the fact that $e_o(n)$ is a zero-mean process, i.e. $\mathrm{E}[e_o(n)] = 0$, for all values of n. This result shows that $\hat{\mathbf{w}}(n)$ is an unbiased estimate of $\mathbf{w}_o$.

The above derivation does not include the effect of initialization, i.e. $\boldsymbol{\Psi}^{-1}(0) = \delta^{-1}\mathbf{I}$, which is required for proper operation of the RLS algorithm. This initialization introduces some bias in $\hat{\mathbf{w}}(n)$ which is proportional to δ and decreases as n increases (see Problem P12.3).

12.5.2 Weight-error correlation matrix

Let us define the weight-error vector

$$\hat{\mathbf{v}}(n) = \hat{\mathbf{w}}(n) - \mathbf{w}_o. \tag{12.64}$$

From (12.62) we obtain

$$\hat{\mathbf{v}}(n) = \boldsymbol{\Psi}_\lambda^{-1}(n)\mathbf{X}(n)\boldsymbol{\Lambda}(n)\mathbf{e}_o(n). \tag{12.65}$$

We also define the weight-error correlation matrix

$$\hat{\mathbf{K}}(n) = \mathrm{E}[\hat{\mathbf{v}}(n)\hat{\mathbf{v}}^{\mathrm{T}}(n)]. \tag{12.66}$$

Substituting (12.65) in (12.66) and noting that $\left(\boldsymbol{\Psi}_\lambda^{-1}(n)\right)^{\mathrm{T}} = \boldsymbol{\Psi}_\lambda^{-1}(n)$ and $\boldsymbol{\Lambda}^{\mathrm{T}}(n) = \boldsymbol{\Lambda}(n)$ we obtain

$$\hat{\mathbf{K}}(n) = \mathrm{E}[\boldsymbol{\Psi}_\lambda^{-1}(n)\mathbf{X}(n)\boldsymbol{\Lambda}(n)\mathbf{e}_o(n)\mathbf{e}_o^{\mathrm{T}}(n)\boldsymbol{\Lambda}(n)\mathbf{X}^{\mathrm{T}}(n)\boldsymbol{\Psi}_\lambda^{-1}(n)]. \tag{12.67}$$

Recalling the independence of $e_o(n)$ and $\mathbf{x}(n)$, from (12.67) we obtain

$$\hat{\mathbf{K}}(n) = \mathrm{E}[\boldsymbol{\Psi}_\lambda^{-1}(n)\mathbf{X}(n)\boldsymbol{\Lambda}(n)\mathrm{E}[\mathbf{e}_o(n)\mathbf{e}_o^{\mathrm{T}}(n)]\boldsymbol{\Lambda}(n)\mathbf{X}^{\mathrm{T}}(n)\boldsymbol{\Psi}_\lambda^{-1}(n)]. \tag{12.68}$$

Since $e_o(n)$ is a white noise process,

$$\mathrm{E}[\mathbf{e}_o(n)\mathbf{e}_o^{\mathrm{T}}(n)] = \sigma_o^2\mathbf{I}, \tag{12.69}$$

where σ_o^2 is the variance of $e_o(n)$ and $\mathbf{I}$ is the identity matrix with appropriate dimension. Finally, substituting (12.69) in (12.68) we get

$$\hat{\mathbf{K}}(n) = \sigma_o^2\mathrm{E}[\boldsymbol{\Psi}_\lambda^{-1}(n)\boldsymbol{\Psi}_{\lambda^2}(n)\boldsymbol{\Psi}_\lambda^{-1}(n)], \tag{12.70}$$

where $\boldsymbol{\Psi}_{\lambda^2}(n) = \mathbf{X}(n)\boldsymbol{\Lambda}^2(n)\mathbf{X}^{\mathrm{T}}(n)$.

Rigorous evaluation of (12.70) is a difficult task. Hence, we make the following assumptions to facilitate an approximate evaluation of $\hat{\mathbf{K}}(n)$:

1. The observed input vectors $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(n)$ constitute the samples of an ergodic process. Thus, the time averages may be used instead of the ensemble averages.
2. The forgetting factor λ is very close to 1.
3. The time n at which $\hat{\mathbf{K}}(n)$ is evaluated is large.

We note from (12.39) that $\boldsymbol{\Psi}_\lambda(n)$ is a weighted sum of the outer products $\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)$, $\mathbf{x}(n-1)\mathbf{x}^{\mathrm{T}}(n-1)$, $\mathbf{x}(n-2)\mathbf{x}^{\mathrm{T}}(n-2), \ldots$. Thus, considering the above assumptions, we find that

$$\boldsymbol{\Psi}_\lambda(n) \approx \frac{1-\lambda^n}{1-\lambda}\mathbf{R}, \tag{12.71}$$

where $\mathbf{R} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]$ is the correlation matrix of the input.

Substituting (12.71) in (12.70) we obtain

$$\begin{aligned}\hat{\mathbf{K}}(n) &= \sigma_{\mathrm{o}}^2\left(\frac{1-\lambda}{1-\lambda^n}\right)^2 \times \frac{1-\lambda^{2n}}{1-\lambda^2}\mathbf{R}^{-1} \\ &= \sigma_{\mathrm{o}}^2\frac{1-\lambda}{1+\lambda}\cdot\frac{1+\lambda^n}{1-\lambda^n}\mathbf{R}^{-1}.\end{aligned} \tag{12.72}$$

In the steady state, i.e. when $n \to \infty$, from (12.72) we obtain

$$\hat{\mathbf{K}}(\infty) = \sigma_{\mathrm{o}}^2\frac{1-\lambda}{1+\lambda}\mathbf{R}^{-1}. \tag{12.73}$$

12.5.3 The learning curve

From the summary of the RLS algorithm in Table 12.1 (or Table 12.2), we find that the filter output at time n is obtained according to the equation

$$\hat{y}_{n-1}(n) = \hat{\mathbf{w}}^{\mathrm{T}}(n-1)\mathbf{x}(n). \tag{12.74}$$

Accordingly, the *learning curve* of the RLS algorithm is defined in terms of the a priori error $\hat{e}_{n-1}(n)$ as

$$\hat{\xi}_{n-1}(n) = \mathrm{E}[\hat{e}_{n-1}^2(n)]. \tag{12.75}$$

To evaluate $\hat{\xi}_{n-1}(n)$, we proceed as follows.

Using (12.64) and (12.59) we obtain

$$\begin{aligned}\hat{e}_{n-1}(n) &= d(n) - \hat{\mathbf{w}}^{\mathrm{T}}(n-1)\mathbf{x}(n) \\ &= d(n) - \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{x}(n) - \hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{x}(n) \\ &= e_{\mathrm{o}}(n) - \hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{x}(n).\end{aligned} \tag{12.76}$$

Substituting (12.76) in (12.75) we obtain

$$\begin{aligned}\hat{\xi}_{n-1}(n) &= \mathrm{E}[(e_{\mathrm{o}}(n) - \hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{x}(n))^2] \\ &= \mathrm{E}[e_{\mathrm{o}}^2(n)] - 2\mathrm{E}[\hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{x}(n)e_{\mathrm{o}}(n)] + \mathrm{E}[\hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\hat{\mathbf{v}}(n-1)], \quad (12.77)\end{aligned}$$

where the last term is obtained by noting that $\hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{x}(n) = \mathbf{x}^{\mathrm{T}}(n)\hat{\mathbf{v}}(n-1)$.

To simplify the evaluation of the last two terms on the right-hand side of (12.77), we make use of the assumptions we have made on $e_{\mathrm{o}}(n)$ and $\mathbf{x}(n)$. First, $e_{\mathrm{o}}(n)$ is a zero-mean white process, and second, $e_{\mathrm{o}}(n)$ and $\mathbf{x}(n)$ are independent. Consequently, $e_{\mathrm{o}}(n)$ is also independent of $\hat{\mathbf{v}}(n-1)\mathbf{x}(n)$, since $\hat{\mathbf{v}}(n-1)$ depends only on the past observations which include only the past samples of $e_{\mathrm{o}}(n)$. Noting these we obtain

$$\mathrm{E}[\hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{x}(n)e_{\mathrm{o}}(n)] = \mathrm{E}[\hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{x}(n)]\mathrm{E}[e_{\mathrm{o}}(n)] = 0, \quad (12.78)$$

since $\mathrm{E}[e_{\mathrm{o}}(n)] = 0$.

To simplify the third term on the right-hand side of (12.77), we also assume that $\hat{\mathbf{v}}(n-1)$ and $\mathbf{x}(n)$ are independent. This is similar to the independence assumption in the case of the LMS algorithm (see Chapter 6). Strictly speaking, the latter assumption is hard to justify, since $\hat{\mathbf{v}}(n-1)$ depends on the past samples of $\mathbf{x}(n)$, and the current $\mathbf{x}(n)$ may not be independent of its past samples. However, when n is large, $\hat{\mathbf{v}}(n-1)$, which is determined by a large number of past observations of $\mathbf{x}(n)$, depends only weakly on the present sample of $\mathbf{x}(n)$. This is because the older samples of $\mathbf{x}(n)$ are only loosely dependent on the present $\mathbf{x}(n)$, unless $\mathbf{x}(n)$ contains one or more significant narrow-band components. We exclude such special cases in our study here and assume that $\hat{\mathbf{v}}(n-1)$ and $\mathbf{x}(n)$ are independent of each other. This is referred to as the *independence assumption* in analogy with the independence assumption used in the case of the LMS algorithm. However, we note that, unlike the LMS algorithm, for which the independence assumption could be used for all (small and large) values of the time index n, in the case of the RLS algorithm the independence assumption is valid only for large values of n.

Using the independence assumption we get

$$\begin{aligned}\mathrm{E}[\hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\hat{\mathbf{v}}(n-1)] &= \mathrm{E}[\hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]\hat{\mathbf{v}}(n-1)] \\ &= \mathrm{E}[\hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{R}\hat{\mathbf{v}}(n-1)]. \quad (12.79)\end{aligned}$$

Next, we note that $\mathrm{E}[\hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{R}\hat{\mathbf{v}}(n-1)]$ is a scalar and proceed as follows:

$$\begin{aligned}\mathrm{E}[\hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{R}\hat{\mathbf{v}}(n-1)] &= \mathrm{tr}[\mathrm{E}[\hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{R}\hat{\mathbf{v}}(n-1)]] \\ &= \mathrm{E}[\mathrm{tr}[\hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{R}\hat{\mathbf{v}}(n-1)]] \\ &= \mathrm{E}[\mathrm{tr}[\hat{\mathbf{v}}(n-1)\hat{\mathbf{v}}^{\mathrm{T}}(n-1)\mathbf{R}]] \\ &= \mathrm{tr}[\mathrm{E}[\hat{\mathbf{v}}(n-1)\hat{\mathbf{v}}^{\mathrm{T}}(n-1)]\mathbf{R}] \\ &= \mathrm{tr}[\hat{\mathbf{K}}(n-1)\mathbf{R}], \quad (12.80)\end{aligned}$$

where tr[·] denotes the trace of the indicated matrix. In the derivation of (12.80) we have used the linearity property of the expectation and trace operators, the definition (12.66), and the identity

$$\text{tr}[\mathbf{AB}] = \text{tr}[\mathbf{BA}],$$

which is valid for any pair of $N \times M$ and $M \times N$, $\mathbf{A}$ and $\mathbf{B}$ matrices, respectively.

Substituting (12.80) and (12.78) in (12.77) we get

$$\hat{\xi}_{n-1}(n) = \xi_{\min} + \text{tr}[\hat{\mathbf{K}}(n-1)\mathbf{R}], \tag{12.81}$$

where $\xi_{\min} = \text{E}[e_o^2(n)]$ is the minimum MSE of the filter which is achieved when a perfect estimate of $\mathbf{w}_o$ is available. Substituting (12.72) in (12.81) we obtain

$$\hat{\xi}_{n-1}(n) = \xi_{\min} + \frac{1-\lambda}{1+\lambda} \cdot \frac{1+\lambda^{n-1}}{1-\lambda^{n-1}} \cdot N\xi_{\min}. \tag{12.82}$$

This describes the learning curve of the RLS algorithm. Note that we made the assumption of n being large in the derivation of (12.82). Thus, (12.82) can predict the behaviour of the RLS algorithm only after a certain initial transient period. Some

We thus note that the convergence behaviour of the RLS algorithm is controlled by only a single mode of convergence. Unlike the LMS algorithm, whose convergence behaviour is affected by the eigenvalues of the correlation matrix, $\mathbf{R}$, of the filter input, the above results show that the convergence behaviour of the RLS algorithm is independent of the eigenvalues of $\mathbf{R}$. This may be explained by substituting (12.48) in the RLS recursion (12.52), to obtain

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \boldsymbol{\Psi}_{\lambda}^{-1}(n)\hat{e}_{n-1}(n)\mathbf{x}(n). \tag{12.88}$$

When n is large, we may use the approximation (12.71) in (12.88) to get

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \frac{1-\lambda}{1-\lambda^n}\mathbf{R}^{-1}\hat{e}_{n-1}(n)\mathbf{x}(n). \tag{12.89}$$

When n is large such that $\lambda^n \ll 1$, the above recursion simplifies to

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + (1-\lambda)\mathbf{R}^{-1}\hat{e}_{n-1}(n)\mathbf{x}(n). \tag{12.90}$$

Letting $\lambda = 1 - 2\mu$, we find that this is nothing but the LMS–Newton algorithm introduced in Chapter 7 (Section 7.5) and its convergence behaviour was found to be independent of the eigenvalues of $\mathbf{R}$.

At this point, once again, we remind the reader that most of the results developed above are valid only when n is large. In particular, the similarity between the RLS and LMS–Newton algorithms that was noted above is valid in the sense that for large values of n, the RLS recursion approaches an update equation which is similar to the LMS–Newton recursion. However, this does not mean that the RLS and LMS–Newton algorithms have the same convergence behaviour, since the convergence behaviours of the two algorithms are completely different when n is small. This is further illustrated through the simulation results presented below in Section 12.5.5.

12.5.4 Excess MSE and misadjustment

As in Chapter 6, we define the excess MSE of the RLS algorithm as the difference between its steady-state MSE and the minimum achievable MSE. In other words, we define

$$\text{excess MSE} = \lim_{n\to\infty} \hat{\xi}_{n-1}(n) - \xi_{\min}. \tag{12.91}$$

Using (12.82) in (12.91) we obtain

$$\text{excess MSE} = \frac{1-\lambda}{1+\lambda} N\xi_{\min}. \tag{12.92}$$

As in the case of the LMS algorithm, the misadjustment of the RLS algorithm is given by

$$\mathcal{M}_{\text{RLS}} = \frac{\text{excess MSE}}{\xi_{\min}}. \tag{12.93}$$

Substituting (12.92) in (12.93) we obtain

$$\mathcal{M}_{\text{RLS}} = \frac{1-\lambda}{1+\lambda} N. \tag{12.94}$$

12.5.5 Initial transient behaviour of the RLS algorithm

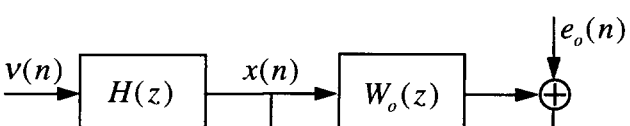
Much of the benefit of the RLS algorithm is attributed to the fact that it shows very fast convergence when it is started from a rest condition with the initial values of $\mathbf{w}(0) = \mathbf{0}$ and $\boldsymbol{\Psi}^{-1}(0) = \delta^{-1}\mathbf{I}$. This fast convergence is observed only after the first N samples of the input and desired output sequences are processed. In typical implementations of the RLS algorithm we always find that the MSE of the adaptive filter converges to a level close to its minimum value within a small number of iterations (usually two to three times the filter length) and then it proceeds with a fine-tuning process which may last much longer, before the MSE reaches its steady-state value. The initial transient behaviour of the RLS algorithm can be best explained through a numerical example (computer simulation).

As a numerical example, here we apply the RLS algorithm to the system modelling set-up of Section 6.4.1. Thus, a comparison of the RLS and LMS algorithms can be made. Figure 12.1 presents the schematic diagram of the modelling set-up. The common input, $x(n)$, to the plant, $W_o(z)$, and adaptive filter, $W(z)$, is obtained by passing a unit variance white Gaussian sequence, $\nu(n)$, through a filter with the system function $H(z)$. The plant noise, as before, is denoted by $e_o(n)$. It is assumed to be a white noise process independent of $x(n)$.

[redacted]

$$W_o(z) = \sum_{i=0}^{7} z^{-i} - \sum_{i=8}^{14} z^{-i}. \tag{12.95}$$

The length of the adaptive filter, N, is chosen equal to the length of $W_o(z)$, i.e. $N = 15$. Also, we present the results of simulations for two choices of input which are



characterized by

$$H(z) = H_1(z) = 0.35 + z^{-1} - 0.35z^{-2} \tag{12.96}$$

and

$$H(z) = H_2(z) = 0.35 + z^{-1} + 0.35z^{-2}. \tag{12.97}$$

The first choice results in an input, $x(n)$, whose corresponding correlation matrix has an eigenvalue spread of 1.45. This is close to white input. On the contrary, the second choice of $H(z)$ results is a highly coloured input with an associated eigenvalue spread of 28.7 (see Section 6.4.1).

Figures 12.2(a) and (b) show the learning curves of the RLS algorithm for the two choices of the input. Each plot is obtained by averaging over 100 independent simulation runs. In all the runs the RLS algorithm was started with zero initial tap weights and the parameter δ (used to initialize $\boldsymbol{\Psi}^{-1}(0) = \delta^{-1}\mathbf{I}$) was set to 0.0001. The forgetting factor, λ, was chosen according to (12.94) to achieve a misadjustment of 10%. From Figure 12.2 we see that the convergence behaviour of the RLS algorithm is independent of the eigenvalue spread of the correlation matrix, $\mathbf{R}$, of the filter input. This is in line with the theoretical predictions of the previous section. Furthermore, we find that the learning curves of the RLS algorithm are quite different from the learning curves of the LMS algorithm – compare the learning curves of Figures 12.2(a) and (b) with their LMS counterparts in Figures 6.7(a) and (b), respectively.

Each learning curve of the RLS algorithm may be divided into three distinct parts:

1. During the first N iterations (N is the filter length) the MSE remains almost unchanged at a high level.
2. The MSE converges at a very fast rate once the iteration number, n, exceeds N.
3. After this period of fast convergence, the RLS algorithm converges toward its steady state at a much slower rate.

The three separate parts of the learning curves of the RLS algorithm may be explained as follows.

During the first $N - 1$ iterations, i.e. when $n < N$, there are an infinite number of possible choices of the tap-weight vector $\hat{\mathbf{w}}(n)$ that satisfy the set of equations

$$\hat{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{x}(k) = d(k), \quad \text{for } k = 1, 2, \ldots, n, \tag{12.98}$$

since there are fewer equations than the number of unknown tap weights. This ambiguity in the solution of the least-squares problem is manifested in the coefficient matrix $\boldsymbol{\Psi}(n)$ whose rank remains less than its dimension, N. This rank deficiency problem, as suggested before, is solved by initializing $\boldsymbol{\Psi}(0)$ to a positive definite matrix, $\delta\mathbf{I}$, which will result a full-rank $\boldsymbol{\Psi}(n)$, and thus a solution for $\hat{\mathbf{w}}(n)$ with no ambiguity. However, the resulting solution, although satisfying (12.98) within a very good approximation, may not give an accurate estimate of the true tap weights of the plant, $\mathbf{w}_o$. As a result, we find that during the first $N - 1$ iterations while the a posteriori error, $\hat{e}_n(n)$, is very small, the a priori error, $\hat{e}_{n-1}(n)$, may still be large. This initial behaviour of the RLS algorithm

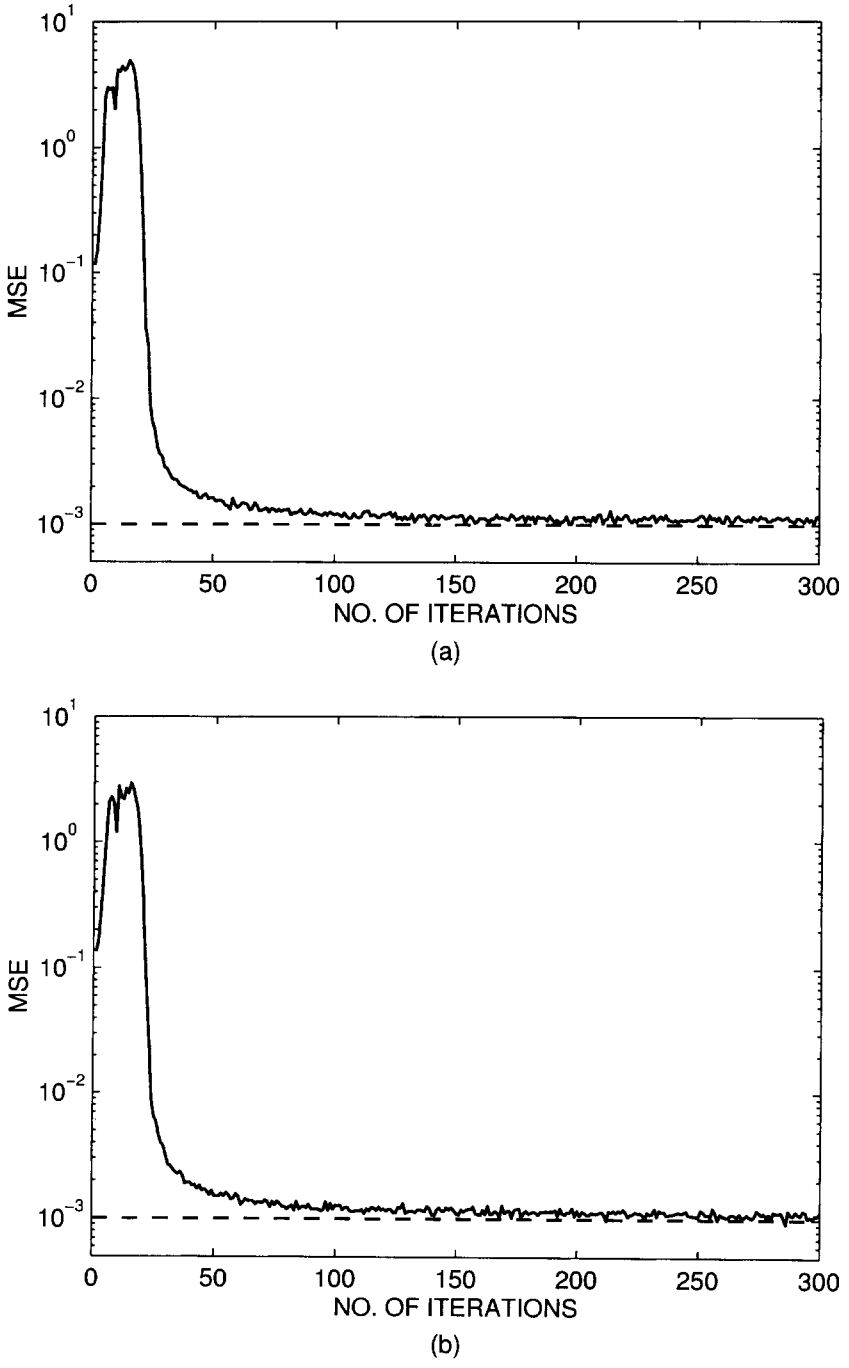


Figure 12.2 Learning curves of the RLS algorithm for the modelling problem of Figure 12.1. (a) $H(z) = H_1(z)$, (b) $H(z) = H_2(z)$. In both cases the forgetting factor, λ, is chosen according to (12.94) for a 10% misadjustment

also explains why in the definition of its learning curve we use $\hat{e}_{n-1}(n)$ and not $\hat{e}_n(n)$. Clearly, the latter does not reflect the fact that the least-squares estimate $\hat{\mathbf{w}}(n)$ may be far from its true value, $\mathbf{w}_o$.

On the contrary, when $n > N$ the number of equations in (12.98) is more than the number of unknown tap weights that we wish to estimate. In that case (12.98) cannot be satisfied exactly. However, a least-squares solution can be found without any ambiguity. The accuracy of the estimate $\hat{\mathbf{w}}(n)$ of $\mathbf{w}_o$ depends on the level of the plant noise and also on the number of observed points, i.e. the iteration number n. In particular, in the case where the plant noise, $e_o(n)$, is zero, (12.98) can be satisfied exactly, for all values of k, by choosing $\hat{\mathbf{w}}(n) = \mathbf{w}_o$. This clearly is the least-squares solution to the problem, since it results in $\zeta(n) = 0$ which is the minimum achievable value of the cost function $\zeta(n)$. The RLS algorithm, which is designed to minimize the cost function $\zeta(n)$, will find this optimum estimate of $\hat{\mathbf{w}}(n)$ once there are enough samples of the observed input and desired output such that the filter tap weights could be found without any ambiguity. This explains the sharp drop in the learning curve of the RLS algorithm when n exceeds N.

The last part of the learning curve of the RLS algorithm, which decays exponentially, matches the results of Section 12.5.3.

Problems

P12.1 The observed samples of the input to a three-tap filter are

$$\mathbf{x}(1) = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, \quad \mathbf{x}(2) = \begin{bmatrix} -2 \\ 1 \\ -1 \end{bmatrix}, \quad \mathbf{x}(3) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{x}(4) = \begin{bmatrix} 0 \\ -1 \\ -1 \end{bmatrix}.$$

(i) Find the projection and the orthogonal complement projection operators of the set of observed input vectors.

(ii) Using the results of (i) find the least-squares estimate $\hat{\mathbf{y}}(4)$ of the desired vector

$$\mathbf{d}(4) = [1 \quad 2 \quad -1 \quad -1]^T.$$

Also, obtain the associated estimation error vector $\hat{\mathbf{e}}(4)$. To check the accuracy of your results, evaluate $\hat{\mathbf{y}}^T(4)\hat{\mathbf{e}}(4)$ and show that it is equal to zero.

(iii) Repeat part (ii) for

$$\mathbf{d}(4) = [0 \quad -1 \quad 1 \quad -1]^T.$$

P12.2 Repeat Problem P12.1 when the observed samples of input are $\mathbf{x}(1)$, $\mathbf{x}(2)$ and $\mathbf{x}(3)$, as given there, and we wish to obtain the least-squares estimates, $\hat{\mathbf{y}}(3)$, of

(i) $\mathbf{d}(3) = [1\ 1\ 1]^T$,

(ii) $\mathbf{d}(3) = [1\ -1\ 2]^T$.

P12.3 Show that the initialization $\Psi_\lambda^{-1}(0) = \delta^{-1}\mathbf{I}$ will introduce a bias in $\hat{\mathbf{w}}(n)$ which is given by

$$\Delta\hat{\mathbf{w}}(n) = \mathrm{E}[\hat{\mathbf{w}}(n)] - \mathbf{w}_\mathrm{o} = -\delta\lambda^n \mathrm{E}[\Psi^{-1}(n)]\mathbf{w}_\mathrm{o}.$$

Simplify this result for the case when $\lambda = 1$ and n is large.

P12.4 Show that when $(\mathbf{X}^\mathrm{T}(N))^{-1}$ exists,

$$(\mathbf{X}^\mathrm{T}(N))^{-1} = (\mathbf{X}(N)\mathbf{X}^\mathrm{T}(N))^{-1}\mathbf{X}(N),$$

and thus conclude that the solution provided by the equation $\mathbf{X}^\mathrm{T}(N)\mathbf{w}(N) = \mathbf{d}(N)$ and the least-squares solution $\hat{\mathbf{w}}(N) = (\mathbf{X}(N)\mathbf{X}^\mathrm{T}(N))^{-1}\mathbf{X}(N)\mathbf{d}(N)$ are the same. What would be the a posteriori estimation error $e_N(N)$?

P12.5 Show that in the RLS algorithm, when $\lambda = 1$, and the iteration number, n, is large,

$$\mathcal{M}_\mathrm{RLS} = \frac{N}{n},$$

where N is the filter length.

P12.6 Consider the case where the RLS algorithm begins with a non-zero $\hat{\mathbf{w}}(0)$ and $\Psi_\lambda(0) = \delta\mathbf{I}$.

(i) Show that this initial condition is equivalent to solving the problem of least-squares according to the following procedure:

- Let $\Psi_\lambda(0) = \delta\mathbf{I}$ and $\boldsymbol{\theta}(0) = \delta\hat{\mathbf{w}}(0)$.
- Update $\Psi_\lambda(n)$ and $\boldsymbol{\theta}_\lambda(n)$ using the recursions (12.41) and (12.42).
- Calculate $\hat{\mathbf{w}}(n)$ using the equation $\hat{\mathbf{w}}(n) = \Psi_\lambda^{-1}(n)\boldsymbol{\theta}_\lambda(n)$.

(ii) Use the result of part (i) to show that when δ is small a non-zero choice of $\hat{\mathbf{w}}(0)$ has no significant effect on the convergence behaviour of the RLS algorithm.

P12.7 Give a detailed derivation of (12.71).

P12.8 In some modelling applications, the tap-weight misalignment defined as

$$\eta(n) = \mathrm{E}[(\mathbf{w}(n) - \mathbf{w}_\mathrm{o})^\mathrm{T}(\mathbf{w}(n) - \mathbf{w}_\mathrm{o})]$$

may be of interest.

(i) Using (12.73), find the steady-state misalignment of the RLS algorithm, $\eta_\mathrm{RLS}(\infty)$, and show that it is a function of the eigenvalues of the correlation matrix $\mathbf{R}$. How does the eigenvalue spread of $\mathbf{R}$ affect $\eta_\mathrm{RLS}(\infty)$?

(ii) Refer to Chapter 6, Section 6.3, and show that for the LMS algorithm

$$\eta_{LMS}(n) = \text{the sum of elements of the vector } \mathbf{k}'(n).$$

Then, starting with (6.55), show that

$$\eta_{LMS}(\infty) = \mu\xi_{\min} \frac{\sum_{i=0}^{N-1} \frac{1}{1-2\mu\lambda_i}}{1 - \sum_{i=0}^{N-1} \frac{\mu\lambda_i}{1-2\mu\lambda_i}}.$$

For the case where $\mu\lambda_i \ll 1$, for all values of i, simplify this result and show that $\eta_{LMS}(\infty)$ depends only on $\sum_{i=0}^{N-1} \lambda_i = \text{tr}[\mathbf{R}]$, and thus is independent of the eigenvalue spread of $\mathbf{R}$.

(iii) Discuss the significance of your observations in (i) and (ii).

P12.9 Consider the modified least-squares cost function

$$\zeta_a(n) = \sum_{k=1}^{n} \lambda^{n-k} e_n^2(k) + \lambda^n K(\hat{\mathbf{w}}(n) - \hat{\mathbf{w}}(0))^T(\hat{\mathbf{w}}(n) - \hat{\mathbf{w}}(0)),$$

where $\hat{\mathbf{w}}(0) \neq \mathbf{0}$ is the initial tap-weight vector and K is a constant.

(i) Use this cost function to derive a modified RLS algorithm.
(ii) Obtain an expression for the learning curve of the filter and discuss the convergence behaviour of the proposed RLS algorithm for small and large values of K.

P12.10 Formulate the problem of modelling a memoryless non-linear system with input $x(n)$ and output

$$y(n) = ax^3(n) + bx^2(n) + cx(n),$$

where a, b and c are the unknown coefficients of the system which should be found in the least-squares sense.

P12.11 Repeat Problem P12.10 for the case where

$$y(n) = ax^3(n) + bx^2(n) + cx(n) + d$$

and a, b, c and d are the unknown coefficients of the system.

Simulation-Oriented Problems

P12.12 The MATLAB program used to obtain the simulation results of Figure 12.2 is available on an accompanying diskette. This, which is written based on the version I of the RLS algorithm, as in Table 12.2, is called 'rlsI.m'. Run this program (or develop and run your own program) to verify the results of Figure 12.2. Also, to gain a better insight into the behaviour of the RLS algorithm, try the following runs:

(i) Run 'rlsI.m' (or your own program) for $\hat{\mathbf{w}}(0) = \mathbf{0}$ and values of $\delta = 0.001, 0.01, 0.1$ and 1, and compare your results with those in Figure 12.2.

(ii) Run 'rlsI.m' (or your own program) for $\delta = 0.0001$ and a few (randomly selected) non-zero values of $\hat{\mathbf{w}}(0)$. Comment on your observation.
(iii) Repeat part (ii) for values of $\delta = 0.001$, 0.01, 0.1 and 1.

P12.13 Develop a simulation program to study the variation in the a posteriori MSE, $\xi_n(n) = \mathrm{E}[\hat{e}_n^2(n)]$, of the RLS algorithm in the case of the modelling problem of Figure 12.1. Comment on your observation.

P12.14 Develop a simulation program to study the convergence behaviour of the RLS algorithm when applied to the channel equalization problem of Section 6.4.2. Compare your results with those of the LMS algorithm in Figures 6.9(a) and (b).

13
Fast RLS Algorithms

The standard recursive least-squares (RLS) algorithm that was introduced in the previous chapter has a computational complexity that grows in proportion to the square of the length of the filter. For long filters this may be unacceptable. During the past two decades many researchers have attempted to solve this drawback of the least-squares method and have come up with a variety of elegant solutions. These solutions, whose computational complexity grows in proportion to the length of the filter, are commonly referred to as *fast RLS algorithms.*

In this chapter we review the underlying principles that are fundamental in the development of fast RLS algorithms. Our intention by no means is to cover the whole spectrum of fast RLS algorithms. A thorough treatment of these algorithms requires many more pages than is allocated to this topic in this book. Moreover, such a treatment is beyond the scope of this book whose primary aim is to serve as an introductory textbook on adaptive filters. Our aim is to put together the basic concepts upon which most fast RLS algorithms are built. Once these basic concepts are understood by the reader, he/she should feel comfortable to proceed with reading the more advanced topics on this subject (see Haykin, 1991, 1996, and Kalouptsidis and Theodoridis, 1993, for more extensive treatments of RLS algorithms).

All fast RLS algorithms benefit from the order-update and time-update equations similar to those introduced in Chapter 11. In other words, fast RLS algorithms combine the concepts of prediction and filtering in an elegant way to come up with computationally efficient implementations. Among these implementations, the RLS lattice (RLSL) algorithm appears to be numerically the most robust implementation. The fast transversal RLS (FTRLS) algorithm (also known as the fast transversal filter – FTF), on the other hand, is an alternative solution that has a minimum number of operations among all the present RLS algorithms.

In this chapter our emphasis is on the RLSL algorithm. The derivation of the RLSL algorithm leads to a number of order- and time-update equations which are fundamental to the derivation of the whole class of fast RLS algorithms. We also present the FTRLS algorithm as a by-product of these equations. Since the lattice structure is closely related to forward and backward linear predictors, we begin with some preliminary discussion of these predictors.

13.1 Least-Squares Forward Prediction

Consider the mth order forward transversal predictor shown in Figure 13.1. The tap-weight vector $\mathbf{a}_m(n) = [a_{m,1}(n)\ a_{m,2}(n)\ \ldots\ a_{m,m}(n)]^{\mathrm{T}}$ is optimized in the least-squares sense over the entire observation interval $k = 1, 2, \ldots, n$. Accordingly, in the forward transversal predictor the observed tap-input vectors are $\mathbf{x}_m(0)$, $\mathbf{x}_m(1), \ldots, \mathbf{x}_m(n-1)$, where $\mathbf{x}_m(k) = [x(k)\ x(k-1)\ \ldots\ x(k-m+1)]^{\mathrm{T}}$, and the desired output samples are $x(1), x(2), \ldots, x(n)$. The *normal equations* of the forward transversal predictor are then obtained as (see Chapter 12)

$$\boldsymbol{\Psi}_m(n-1)\mathbf{a}_m(n) = \boldsymbol{\psi}_m^{\mathrm{f}}(n), \tag{13.1}$$

where

$$\boldsymbol{\Psi}_m(n) = \sum_{k=1}^{n} \lambda^{n-k}\mathbf{x}_m(k)\mathbf{x}_m^{\mathrm{T}}(k), \tag{13.2}$$

$$\boldsymbol{\psi}_m^{\mathrm{f}}(n) = \sum_{k=1}^{n} \lambda^{n-k}x(k)\mathbf{x}_m(k-1), \tag{13.3}$$

and λ is the forgetting factor. The least-squares sum of the estimation errors is then given by

$$\zeta_m^{\mathrm{ff}}(n) = \sum_{k=1}^{n} \lambda^{n-k}f_{m,n}^2(k), \tag{13.4}$$

where

$$f_{m,n}(k) = x(k) - \mathbf{a}_m^{\mathrm{T}}(n)\mathbf{x}_m(k-1). \tag{13.5}$$

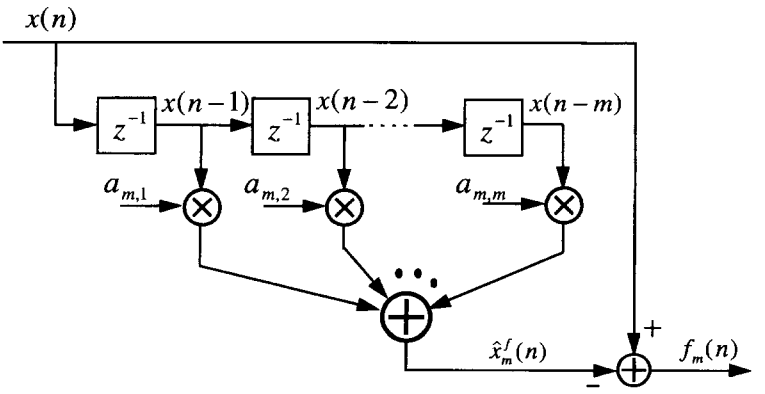


Figure 13.1 Transversal forward predictor

The sequence $f_{m,n}(k)$ is known as the *a posteriori prediction error* since its computation is based on the latest value of the predictor tap-weight vector $\mathbf{a}_m(n)$. In contrast, the *a priori prediction error* of the forward predictor is defined as

$$f_{m,n-1}(k) = x(k) - \mathbf{a}_m^{\mathrm{T}}(n-1)\mathbf{x}_m(k-1), \tag{13.6}$$

where $\mathbf{a}_m(n-1)$ is the previous value of the predictor tap-weight vector.

We recall that according to the *principle of orthogonality*, the a posteriori estimation error, $f_{m,n}(k)$, and the predictor tap-input vector, $\mathbf{x}_m(k-1)$, are orthogonal, in the sense that

$$\sum_{k=1}^{n} \lambda^{n-k} f_{m,n}(k)\mathbf{x}_m(k-1) = \mathbf{0}. \tag{13.7}$$

Using (13.5) and (13.7) in (13.4), we can show that (Problem P13.1)

$$\zeta_m^{\mathrm{ff}}(n) = \sum_{k=1}^{n} \lambda^{n-k} x^2(k) - \mathbf{a}_m^{\mathrm{T}}(n)\psi_m^{\mathrm{f}}(n). \tag{13.8}$$

This result could also be obtained by inserting the relevant variables in equation (12.16).

Application of the *standard RLS algorithm* for adapting the forward transversal predictor results in the following recursion:

$$\mathbf{a}_m(n) = \mathbf{a}_m(n-1) + \mathbf{k}_m(n-1)f_{m,n-1}(n), \tag{13.9}$$

where $f_{m,n-1}(n)$ is the latest sample of the a priori estimation error of the forward predictor and $\mathbf{k}_m(n-1)$ is the present *gain vector* of the algorithm. The time index $n-1$ in the gain vector here follows the predictor tap input whose latest value is $\mathbf{x}_m(n-1)$. The use of this notation also keeps our notations consistent as we proceed with similar formulations for the backward transversal predictor. Furthermore, following the results of the previous chapter (equation (12.48)), the gain vector of the forward transversal predictor is obtained as

$$\mathbf{k}_m(n-1) = \boldsymbol{\Psi}_m^{-1}(n-1)\mathbf{x}_m(n-1). \tag{13.10}$$

At this point we may note that there are a few differences between some of our notations here and those in the previous chapter. Since we will be making frequent use of the order-update equations in the derivations of this chapter, the order of the predictor, m, is explicitly reflected on all the variables. On the other hand, we have dropped the subscript λ (the forgetting factor) from the correlation matrix $\boldsymbol{\Psi}_m(n)$ and the vector $\psi_m(n)$ to simplify the notations. However, the subscript m has been added to them to indicate their dimensions. The *hat* sign that was used to refer to optimized tap-weight vectors and prediction errors is also dropped here in order to simplify the notations.

13.2 Least-Squares Backward Prediction

Figure 13.2 depicts an mth order backward transversal predictor. Here, the predictor tap-weight and tap-input vectors are $\mathbf{g}_m(n) = [g_{m,1}(n)\ g_{m,2}(n)\ \ldots\ g_{m,m}(n)]^{\mathrm{T}}$ and $\mathbf{x}_m(k) = [x(k)\ x(k-1)\ \ldots\ x(k-m+1)]^{\mathrm{T}}$, respectively. The tap-weight vector, $\mathbf{g}_m(n)$, of the predictor is optimized in the least-squares sense over the entire observation interval $k = 1, 2, \ldots, n$. The samples of the desired output are $x(1-m), x(2-m), \ldots, x(n-m)$. Accordingly, the *normal equations* associated with the backward transversal predictor are obtained as (see Chapter 12)

$$\boldsymbol{\Psi}_m(n)\mathbf{g}_m(n) = \boldsymbol{\psi}_m^{\mathrm{b}}(n), \tag{13.11}$$

where $\boldsymbol{\Psi}_m(n)$ is given in (13.2), and

$$\boldsymbol{\psi}_m^{\mathrm{b}}(n) = \sum_{k=1}^{n} \lambda^{n-k} x(k-m)\mathbf{x}_m(k). \tag{13.12}$$

The least-squares sum of the estimation errors is then given by

$$\zeta_m^{\mathrm{bb}}(n) = \sum_{k=1}^{n} \lambda^{n-k} b_{m,n}^2(k), \tag{13.13}$$

where

$$b_{m,n}(k) = x(k-m) - \mathbf{g}_m^{\mathrm{T}}(n)\mathbf{x}_m(k). \tag{13.14}$$

The sequence $b_{m,n}(k)$ is the *a posteriori estimation error* of the backward predictor. In contrast, the *a priori prediction error* is defined as

$$b_{m,n-1}(k) = x(k-m) - \mathbf{g}_m^{\mathrm{T}}(n-1)\mathbf{x}_m(k), \tag{13.15}$$

where $\mathbf{g}_m(n-1)$ is the previous value of the predictor tap-weight vector.

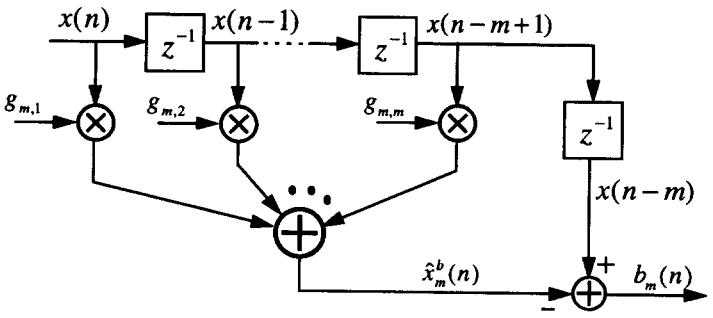


Figure 13.2 Transversal backward predictor

We recall that according to the *principle of orthogonality*, the a posteriori estimation error, $b_{m,n}(k)$, and the predictor tap-input vector, $\mathbf{x}_m(k)$, are orthogonal, in the sense that

$$\sum_{k=1}^{n} \lambda^{n-k} b_{m,n}(k)\mathbf{x}_m(k) = \mathbf{0}. \tag{13.16}$$

Using (13.14) and (13.16) in (13.13) we can show that (Problem P13.2)

$$\zeta_m^{\mathrm{bb}}(n) = \sum_{k=1}^{n} \lambda^{n-k} x^2(k-m) - \mathbf{g}_m^{\mathrm{T}}(n)\psi_m^{\mathrm{b}}(n). \tag{13.17}$$

This result could also be obtained by inserting the relevant variables in equation (12.16).

Application of the *standard RLS algorithm* for adapting the backward transversal predictor results in the following recursion:

$$\mathbf{g}_m(n) = \mathbf{g}_m(n-1) + \mathbf{k}_m(n)b_{m,n-1}(n), \tag{13.18}$$

where $b_{m,n-1}(n)$ is the latest sample of the a priori estimation error of the backward predictor and $\mathbf{k}_m(n)$ is the present *gain vector* of the algorithm, given by

$$\mathbf{k}_m(n) = \Psi_m^{-1}(n)\mathbf{x}_m(n). \tag{13.19}$$

It is instructive to note that the gain vector of an mth order forward predictor is equal to the previous value of the gain vector of the associated backward predictor. This, clearly, follows from the fact that the tap-input vectors of the forward predictor, $\mathbf{x}_m(k-1)$, and the backward predictor, $\mathbf{x}_m(k)$, are one sample apart for a given time instant k.

13.3 The Least-Squares Lattice

Figure 13.3 depicts the schematic of a lattice joint process estimator. This is similar to the lattice joint process estimator presented in Chapter 11 (see Figure 11.7). However, there are some changes made in the notations here so that they can serve our discussion in this chapter better. From the previous chapter we recall that in the least-squares optimization, at any instant of time, say n, the filter parameters are optimized based on the observed data samples from time 1 to n, so that a weighted sum of the error squares is minimized. With this view of the problem, in Figure 13.3 the time index of the signal sequences is chosen to be k, and at time n, k is varied from 1 to n. Furthermore, following the notations in the previous two sections, the estimation errors are labelled with two subscripts. The first subscript denotes the filter/predictor length/order. The second subscript indicates the length, n, of the observed data. The lattice PARCOR coefficients, $\kappa_m^{\mathrm{f}}(n)$ and $\kappa_m^{\mathrm{b}}(n)$, and the regressor coefficients, $c_m(n)$, are also labelled with time index n to emphasize that they are optimized in the least-squares sense based on the data samples up to time n. In addition, to facilitate the derivation of the recursive least-squares lattice in the next section, the summer at the output of the joint process estimator is divided into a set of distributed adders so that the estimation errors of order 1 to N (denoted $e_{1,n}(k)$

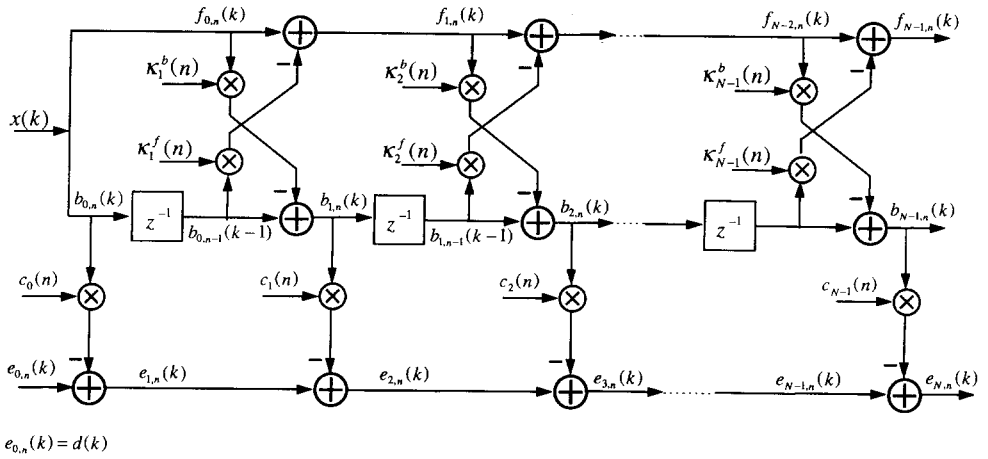


Figure 13.3 Least-squares lattice joint process estimator

through $e_{N,n}(k)$) can be obtained in a sequential manner, as will be explained later in this chapter.

From Chapter 11 we recall that when the input sequence, $x(n)$, is stationary, and the lattice coefficients are optimized to minimize the mean-square errors of the forward and backward predictors, the PARCOR coefficients, κ_m^{f} and κ_m^{b}, are found to be equal. However, we note that this is not the case when the optimization of the lattice coefficients is based on least-squares criteria. Noting this, throughout this chapter we keep the superscripts f and b on κ_m^{f} and κ_m^{b}, respectively, to differentiate between the two.

To optimize the coefficients of the lattice joint process estimator the following sums are minimized, simultaneously:

$$\zeta_m^{ff}(n) = \sum_{k=1}^{n} \lambda^{n-k} f_{m,n}^2(k), \quad \text{for } m = 1, 2, \ldots, N-1, \tag{13.20}$$

$$\zeta_m^{bb}(n) = \sum_{k=1}^{n} \lambda^{n-k} b_{m,n}^2(k), \quad \text{for } m = 1, 2, \ldots, N-1, \tag{13.21}$$

$$\zeta_m^{ee}(n) = \sum_{k=1}^{n} \lambda^{n-k} e_{m,n}^2(k), \quad \text{for } m = 1, 2, \ldots, N, \tag{13.22}$$

where $f_{m,n}(k)$ and $b_{m,n}(k)$ are the a posteriori estimation errors as defined before, and similarly $e_{m,n}(k)$ is defined as the *a posteriori estimation error of the length m joint process estimator*. Note from Figure 13.3 that there are effectively N forward and N backward predictors of order 1 to N, as well as N joint process estimators of length 1 to N, optimized simultaneously. Furthermore, it is important to note that the least-squares sums (13.20)–(13.22) are independent of whether the predictors and joint process estimators are implemented in transversal or lattice forms. We will exploit this fact to simplify the derivations that follow by switching between the equations derived for the transversal and lattice forms to arrive at the desired results.

In Chapter 11 we discussed a number of properties of the lattice structure. In particular, we noted that the backward prediction errors of different orders are orthogonal (uncorrelated) with one another. This and many other properties of the lattice structure that were discussed in Chapter 11 based on *stochastic averages,* are equally applicable to the lattice structure of Figure 13.3, where optimization of the filter coefficients is done based on *time averages* (because of the least-squares optimization). The most important properties of the least-squares lattice which are relevant to the derivation of the RLSL algorithm in the next section are the following:

1. At time n, the PARCOR coefficients $\kappa_m^{\mathrm{f}}(n)$ and $\kappa_m^{\mathrm{b}}(n)$ of the successive stages of the lattice structure can be optimized sequentially as follows. We first note that the sequences at the tap inputs of the first stage (i.e. the signals multiplied by the PARCOR coefficients $\kappa_1^{\mathrm{f}}(n)$ and $\kappa_1^{\mathrm{b}}(n)$) are $f_{0,n}(k)$ and $b_{0,n-1}(k-1)$, for $k = 1, 2, \ldots, n$ (see Figure 13.3). Using these sequences, the coefficients $\kappa_1^{\mathrm{f}}(n)$ and $\kappa_1^{\mathrm{b}}(n)$ are optimized so that the output sequences $f_{1,n}(k)$ and $b_{1,n}(k)$ of the first stage are minimized in the least-squares sense. Next, we note that the tap inputs in the second stage are $f_{1,n}(k)$ and $b_{1,n-1}(k-1)$. Accordingly, considering the sequences $f_{1,n}(k)$ and $b_{1,n-1}(k-1)$, for $k = 1, 2, \ldots, n$, as tap inputs to the second stage, the coefficients $\kappa_2^{\mathrm{f}}(n)$ and $\kappa_2^{\mathrm{b}}(n)$ are optimized so that the output sequences $f_{2,n}(k)$ and $b_{2,n}(k)$ of the second stage are minimized in the least-squares sense. This process continues for the rest of the stages as well. The above process leads to the following equations which are the bases for the derivation of the RLS algorithm, as explained in the next section:

$$\kappa_m^{\mathrm{f}}(n) = \frac{\sum_{k=1}^{n} \lambda^{n-k} f_{m-1,n}(k) b_{m-1,n-1}(k-1)}{\sum_{k=1}^{n} \lambda^{n-k} b_{m-1,n-1}^2(k-1)} \tag{13.23}$$

and

$$\kappa_m^{\mathrm{b}}(n) = \frac{\sum_{k=1}^{n} \lambda^{n-k} f_{m-1,n}(k) b_{m-1,n-1}(k-1)}{\sum_{k=1}^{n} \lambda^{n-k} f_{m-1,n}^2(k)}, \tag{13.24}$$

for $m = 1, 2, \ldots, N-1$.

2. Once the PARCOR coefficients are optimized, the backward prediction errors $b_{0,n}(k)$, $b_{1,n}(k), \ldots, b_{N,n}(k)$ are orthogonal with one another, in the sense that

$$\sum_{k=1}^{n} \lambda^{n-k} b_{i,n}(k) b_{j,n}(k) = 0 \tag{13.25}$$

for any pair of unequal i and j in the range of 0 to $N-1$.

3. The regressor coefficients $c_0(n)$, $c_1(n), \ldots, c_{N-1}(n)$ may also be optimized in a sequential manner. That is, first $c_0(n)$ is optimized by minimizing $\zeta_1^{ee}(n)$. We then hold $c_0(n)$, run the sequence $\{x(1), x(2), \ldots, x(n)\}$ through the first stage of the lattice and optimize $c_1(n)$ so that $\zeta_2^{ee}(n)$ is minimized. This process continues for the rest of the joint process estimator coefficients as well. To summarize, the regressor coefficients $c_0(n)$, $c_1(n), \ldots, c_{N-1}(n)$ are obtained according to the

following equations:

$$c_m(n) = \frac{\sum_{k=1}^{n} \lambda^{n-k} e_{m,n}(k) b_{m,n}(k)}{\sum_{k=1}^{n} \lambda^{n-k} b_{m,n}^2(k)}, \tag{13.26}$$

for $m = 0, 1, \ldots, N-1$.

Equations (13.23), (13.24) and (13.26), although fundamental in providing a clear understanding of the underlying principles in the development of the least-squares lattice algorithm, cannot be used for computation of the lattice coefficients in an adaptive application because their computational complexity grows with the number of data samples, n. As in the case of the standard RLS algorithm, the problem is solved by finding a set of equations that updates the filter coefficients in a recursive manner. This is the subject of the next section.

13.4 The RLSL Algorithm

In this section we go through a systematic step-by-step procedure to develop the recursions necessary for the derivation of the recursive least-squares lattice (RLSL) algorithm. The development of the RLSL algorithm involves a large number of variables, compared with any of the algorithms that we have derived/discussed thus far in this book. Because of this, it is often difficult for a novice to the topic to follow these equations. Thus, choosing the right set of notations that reduces this burden is crucial to the development of readable material on this topic. Bearing this in mind, our discussion on the RLSL algorithm begins with an introduction to the notations and some preliminaries. The derivations will be followed thereafter.

13.4.1 Notations and preliminaries

In the previous few sections we introduced a number of notations for formulating the least-squares solutions in the cases of forward and backward transversal predictors as well as the lattice joint process estimator. Here, we introduce some more notations and also some new definitions which are necessary for the derivations that follow.

Prewindowing of input data

Throughout the discussion in the rest of this chapter we assume that the samples of input signal, $x(k)$, are all zero for values of $k \leq 0$. This assumption on input signal is known as *prewindowing*. In this book we do not consider other variations of the fast RLS algorithms that are based on other types of windowing methods (Honig and Messerschmitt, 1984, and Alexander, 1986a, b).

A priori and a posteriori estimation errors

We noted that the subscript n in the sequences $f_{m,n}(k)$, $b_{m,n}(k)$ and $e_{m,n}(k)$ denotes that they are a posteriori estimation errors. The term a posteriori signifies that the errors are

obtained using the filter (predictor) coefficients that have been optimized using the past as well as the present samples of the input and desired output, i.e. $x(k)$ and $d(k)$, for $k = 1, 2, \ldots, n$. In other words, the a posteriori estimation errors are obtained when the lattice coefficients, the $\kappa_m^{\mathrm{f}}(n)$s, $\kappa_m^{\mathrm{b}}(n)$s and $c_m(n)$s, as given by (13.23), (13.24) and (13.26), are chosen. In contrast, if we compute these estimation errors using the last values of the joint process estimator coefficients, i.e. the $\kappa_m^{\mathrm{f}}(n-1)$s, $\kappa_m^{\mathrm{b}}(n-1)$s and $c_m(n-1)$s, then the resulting errors are known as a priori. We use the notations $f_{m,n-1}(k)$, $b_{m,n-1}(k)$ and $e_{m,n-1}(k)$ (with the subscripts $n-1$ signifying the use of the last values of the lattice coefficients, the $\kappa_m^{\mathrm{f}}(n-1)$s, $\kappa_m^{\mathrm{b}}(n-1)$s and $c_m(n-1)$s) to refer to the a priori estimation errors.

Conversion factor, $\gamma_m(n)$

A key result to the development of the fast RLS algorithms is the following relationship:

$$\frac{e_{m,n}(n)}{e_{m,n-1}(n)} = \frac{b_{m,n}(n)}{b_{m,n-1}(n)} = \frac{f_{m,n+1}(n+1)}{f_{m,n}(n+1)}. \tag{13.27}$$

Note that in (13.27) the denominators are the a priori estimation errors and the numerators are the a posteriori estimation errors. To appreciate this relationship, we note that the tap-input vector to the length m joint process estimator at time n is $\mathbf{x}_m(n) = [x(n)\ x(n-1)\ \ldots\ x(n-m+1)]^{\mathrm{T}}$. This is also the tap-input vector to the mth order backward predictor at time n and that of the forward predictor at time $n+1$. Noting this, it appears that, in general, the ratio of the a posteriori and a priori estimation errors depends only on the tap-input vector of the filter (predictor or joint process estimator). This ratio, which will be discussed in detail later, is called the *conversion factor*, denoted as $\gamma_m(n)$.

Least-squares error sums, $\zeta_m^{ee}(n)$, $\zeta_m^{ff}(n)$ and $\zeta_m^{bb}(n)$

We recall that $\zeta_m^{ee}(n)$, $\zeta_m^{ff}(n)$ and $\zeta_m^{bb}(n)$, as defined in (13.22), (13.20) and (13.21), respectively, refer to the least-squares error sums of the joint process estimator of length [redacted] forward and backward predictors of order m. Note also that these are all based

and

$$\zeta_m^{be}(n) = \sum_{k=1}^{n} \lambda^{n-k} e_{m,n}(k) b_{m,n}(k). \tag{13.29}$$

To follow the same terminology, we may refer to the least-squares sums $\zeta_m^{ff}(n)$, $\zeta_m^{bb}(n)$ and $\zeta_m^{ee}(n)$ as *autocorrelations*.

Using (13.20), (13.21), (13.22), (13.28) and (13.29), the set of equations (13.23), (13.24) and (13.26) are written as

$$\kappa_m^{\mathrm{f}}(n) = \frac{\zeta_{m-1}^{fb}(n)}{\zeta_{m-1}^{bb}(n-1)}, \tag{13.30}$$

$$\kappa_m^{\mathrm{b}}(n) = \frac{\zeta_{m-1}^{fb}(n)}{\zeta_{m-1}^{ff}(n)} \tag{13.31}$$

and

$$c_m(n) = \frac{\zeta_m^{be}(n)}{\zeta_m^{bb}(n)}. \tag{13.32}$$

Later in the chapter we develop a set of equations for recursive updating of the auto- and cross-correlations that were just defined. *The updated auto- and cross-correlation will then be substituted into equations* (13.30)–(13.32) *for computation of the lattice coefficients at every iteration.*

Augmented normal equations for forward and backward prediction

Using the definition (13.2), $\boldsymbol{\Psi}_{m+1}(n)$ may be extended as

$$\boldsymbol{\Psi}_{m+1}(n) = \begin{bmatrix} \psi^{00}(n) & \boldsymbol{\psi}_m^{\mathrm{fT}}(n) \\ \boldsymbol{\psi}_m^{\mathrm{f}}(n) & \boldsymbol{\Psi}_m(n-1) \end{bmatrix}, \tag{13.33}$$

where $\boldsymbol{\psi}_m^{\mathrm{f}}(n)$ is defined by (13.3) and

$$\psi^{00}(n) = \sum_{k=1}^{n} \lambda^{n-k} x^2(k). \tag{13.34}$$

Using (13.33) and (13.34), equations (13.1) and (13.8) may be combined to obtain

$$\boldsymbol{\Psi}_{m+1}(n)\tilde{\mathbf{a}}_m(n) = \begin{bmatrix} \zeta_m^{ff}(n) \\ \mathbf{0}_m \end{bmatrix}, \tag{13.35}$$

where

$$\tilde{\mathbf{a}}_m(n) = \begin{bmatrix} 1 \\ -\mathbf{a}_m(n) \end{bmatrix}, \tag{13.36}$$

$\mathbf{0}_m$ denotes the $m \times 1$ zero vector, and $\mathbf{a}_m(n)$ is the tap-weight vector of the transversal forward predictor, optimized in the least-squares sense. Equation (13.35) is known as the *augmented normal equation for the forward predictor of order m.*

The matrix $\mathbf{\Psi}_{m+1}(n)$ may also be extended as

$$\mathbf{\Psi}_{m+1}(n) = \begin{bmatrix} \mathbf{\Psi}_m(n) & \boldsymbol{\psi}_m^{\mathrm{b}}(n) \\ \boldsymbol{\psi}_m^{\mathrm{bT}}(n) & \psi^{mm}(n) \end{bmatrix}, \tag{13.37}$$

where $\boldsymbol{\psi}_m^{\mathrm{b}}(n)$ is defined by (13.12) and

$$\psi^{mm}(n) = \sum_{k=1}^{n} \lambda^{n-k} x^2(k-m). \tag{13.38}$$

Using (13.37) and (13.38), equations (13.11) and (13.17) may be combined to obtain

$$\mathbf{\Psi}_{m+1}(n)\tilde{\mathbf{g}}_m(n) = \begin{bmatrix} \mathbf{0}_m \\ \zeta_m^{bb}(n) \end{bmatrix}, \tag{13.39}$$

where

$$\tilde{\mathbf{g}}_m(n) = \begin{bmatrix} -\mathbf{g}_m(n) \\ 1 \end{bmatrix} \tag{13.40}$$

and $\mathbf{g}_m(n)$ is the tap-weight vector of the transversal backward predictor, optimized in the least-squares sense. Equation (13.39) is known as the *augmented normal equation for the backward predictor of order m.*

13.4.2 Update recursion for the least-squares error sums

Consider an N-tap transversal filter with the tap-input vector $\mathbf{x}(k) = [x(k)\ x(k-1)\ \ldots\ x(k-N+1)]^{\mathrm{T}}$ and desired output $d(k)$. From the previous chapter, we recall that the least-squares error sum of the filter at time n is[1]

$$\zeta_{\min}(n) = \mathbf{d}^{\mathrm{T}}(n)\mathbf{\Lambda}(n)\mathbf{d}(n) - \boldsymbol{\theta}^{\mathrm{T}}(n)\hat{\mathbf{w}}(n), \tag{13.41}$$

where $\hat{\mathbf{w}}(n)$ is the optimized tap-weight vector of the filter, $\mathbf{d}(n) = [d(1)\ d(2)\ \ldots\ d(n)]^{\mathrm{T}}$,

$$\boldsymbol{\theta}(n) = \sum_{k=1}^{n} \lambda^{n-k} d(k)\mathbf{x}(k)$$

[1] It may be noted that (13.41) is similar to (12.16) with the forgetting factor, λ, included in the results. Also, to be consistent with the rest of our notations in this chapter the subscript λ has been dropped from vectors and matrices.

and $\mathbf{\Lambda}(n)$ is the diagonal matrix consisting of the powers of the forgetting factor, λ, as defined in (12.35). Substituting (12.42), (12.52) and

$$\mathbf{d}^{\mathrm{T}}(n)\mathbf{\Lambda}(n)\mathbf{d}(n) = d^2(n) + \lambda\mathbf{d}^{\mathrm{T}}(n-1)\mathbf{\Lambda}(n-1)\mathbf{d}(n-1)$$

in (13.41) and rearranging, we get

$$\begin{aligned}\zeta_{\min}(n) &= \lambda(\mathbf{d}^{\mathrm{T}}(n-1)\mathbf{\Lambda}(n-1)\mathbf{d}(n-1) - \boldsymbol{\theta}^{\mathrm{T}}(n-1)\hat{\mathbf{w}}(n-1)) \\ &\quad + d(n)\big(d(n) - \hat{\mathbf{w}}^{\mathrm{T}}(n-1)\mathbf{x}^{\mathrm{T}}(n)\big) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{k}(n)e_{n-1}(n)d(n) \\ &\quad - \lambda\boldsymbol{\theta}^{\mathrm{T}}(n-1)\mathbf{k}(n)e_{n-1}(n) \\ &= \lambda\zeta_{\min}(n-1) + d(n)e_{n-1}(n) - (\mathbf{x}(n)d(n) + \lambda\boldsymbol{\theta}(n-1))^{\mathrm{T}}\mathbf{k}(n)e_{n-1}(n), \end{aligned} \tag{13.42}$$

where we have noted that $d(n) - \hat{\mathbf{w}}^{\mathrm{T}}(n-1)\mathbf{x}(n)$ is the a priori estimation error $e_{n-1}(n)$. Furthermore, from (12.42) we note that $\mathbf{x}(n)d(n) + \lambda\boldsymbol{\theta}(n-1) = \boldsymbol{\theta}(n)$. Using this result and (12.48), we obtain

$$\begin{aligned}(\mathbf{x}(n)d(n) + \lambda\boldsymbol{\theta}(n-1))^{\mathrm{T}}\mathbf{k}(n) &= \boldsymbol{\theta}^{\mathrm{T}}(n)\mathbf{\Psi}^{-1}(n)\mathbf{x}(n) \\ &= \hat{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{x}(n), \end{aligned} \tag{13.43}$$

where we have noted that $\boldsymbol{\theta}^{\mathrm{T}}(n)\mathbf{\Psi}^{-1}(n) = \big(\mathbf{\Psi}^{-1}(n)\boldsymbol{\theta}(n)\big)^{\mathrm{T}} = \hat{\mathbf{w}}^{\mathrm{T}}(n)$, since $\mathbf{\Psi}^{-1}(n)$ is symmetrical. Substituting (13.43) in (13.42) and rearranging, we get

$$\zeta_{\min}(n) = \lambda\zeta_{\min}(n-1) + e_n(n)e_{n-1}(n), \tag{13.44}$$

where $e_n(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{x}(n)$ is the a posteriori estimation error. Thus, to update $\zeta_{\min}(n-1)$, we only need to know the a priori and a posteriori estimation errors at instant n, i.e. $e_{n-1}(n)$ and $e_n(n)$, respectively.

Recursion (13.44) can readily be applied to update the least-squares error sums of the forward and backward predictors as well as the joint process estimator. The results are:

$$\zeta_m^{ff}(n) = \lambda\zeta_m^{ff}(n-1) + f_{m,n}(n)f_{m,n-1}(n), \tag{13.45}$$

$$\zeta_m^{bb}(n) = \lambda\zeta_m^{bb}(n-1) + b_{m,n}(n)b_{m,n-1}(n), \tag{13.46}$$

$$\zeta_m^{ee}(n) = \lambda\zeta_m^{ee}(n-1) + e_{m,n}(n)e_{m,n-1}(n), \tag{13.47}$$

where $f_{m,n-1}(n)$, $f_{m,n}(n)$, $b_{m,n-1}(n)$, $b_{m,n}(n)$, $e_{m,n-1}(n)$ and $e_{m,n}(n)$, as defined before, are the associated a priori and a posteriori estimation errors.

We note that the above update equations involve the use of both the a priori and a posteriori estimation errors. We next see that with the aid of the conversion factor, $\gamma_m(n)$, the above recursions may be written in terms of either the a priori or a posteriori estimation errors only.

13.4.3 Conversion factor

Using recursion (12.52), the a posteriori estimation error $e_n(n)$ of a transversal filter with the least-squares optimized tap-weight vector $\hat{\mathbf{w}}(n)$, tap-input vector $\mathbf{x}(n)$ and desired

output $d(n)$ may be expanded as

$$\begin{aligned} e_n(n) &= d(n) - \hat{\mathbf{w}}^{\mathrm{T}}(n)\mathbf{x}(n) \\ &= d(n) - (\hat{\mathbf{w}}(n-1) + \mathbf{k}(n)e_{n-1}(n))^{\mathrm{T}}\mathbf{x}(n) \\ &= d(n) - \hat{\mathbf{w}}^{\mathrm{T}}(n-1)\mathbf{x}(n) - \mathbf{k}^{\mathrm{T}}(n)\mathbf{x}(n)e_{n-1}(n) \\ &= e_{n-1}(n) - \mathbf{k}^{\mathrm{T}}(n)\mathbf{x}(n)e_{n-1}(n) \\ &= (1 - \mathbf{k}^{\mathrm{T}}(n)\mathbf{x}(n))e_{n-1}(n), \end{aligned} \tag{13.48}$$

where $e_{n-1}(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{T}}(n-1)\mathbf{x}(n)$ is the a priori estimation error. We note that the a priori and a posteriori estimation errors, $e_{n-1}(n)$ and $e_n(n)$, respectively, are related by the factor $1 - \mathbf{k}^{\mathrm{T}}(n)\mathbf{x}(n)$. This is called the conversion factor, as mentioned in Section 13.4.1, and is denoted by $\gamma(n)$. Thus,

$$\gamma(n) = 1 - \mathbf{k}^{\mathrm{T}}(n)\mathbf{x}(n). \tag{13.49}$$

Substituting (12.48) in (13.49) we also obtain

$$\gamma(n) = 1 - \mathbf{x}^{\mathrm{T}}(n)\boldsymbol{\Psi}^{-1}(n)\mathbf{x}(n). \tag{13.50}$$

An interesting interpretation of $\gamma(n)$, the study of which is left for the reader in Problem P13.8, reveals that $\gamma(n)$ is a positive quantity less than or equal to one. Another interesting property of $\gamma(n)$ is seen by noting that for a given forgetting factor, λ, $\boldsymbol{\Psi}(n)$ depends on the input samples to the filter only. Accordingly, $\gamma(n)$ can be found once the observed tap-input vectors to the filter are known. The following cases are then identified:

1. In an mth order forward predictor, with the observed tap-input vectors $\mathbf{x}_m(0)$, $\mathbf{x}_m(1), \ldots, \mathbf{x}_m(n-1)$ (with $\mathbf{x}_m(0) = \mathbf{0}$, because of prewindowing the input data), the conversion factor is recognized as

$$\begin{aligned} \gamma_m(n-1) &= 1 - \mathbf{x}_m^{\mathrm{T}}(n-1)\boldsymbol{\Psi}^{-1}(n-1)\mathbf{x}_m(n-1) \\ &= 1 - \mathbf{k}_m^{\mathrm{T}}(n-1)\mathbf{x}_m(n-1), \end{aligned} \tag{13.51}$$

   where $\mathbf{k}_m(n-1)$ is the gain vector of the forward predictor, as was identified before (Section 13.1). Accordingly, the a priori and a posteriori estimation errors, $f_{m,n-1}(n)$ and $f_{m,n}(n)$, of the forward predictor are related according to the equation

$$f_{m,n}(n) = \gamma_m(n-1)f_{m,n-1}(n). \tag{13.52}$$

2. Similarly, in an mth order backward predictor, with the observed tap-input vectors $\mathbf{x}_m(1), \mathbf{x}_m(2), \ldots, \mathbf{x}_m(n)$, the conversion factor is recognized as

$$\begin{aligned} \gamma_m(n) &= 1 - \mathbf{x}_m^{\mathrm{T}}(n)\boldsymbol{\Psi}_m^{-1}(n)\mathbf{x}_m(n) \\ &= 1 - \mathbf{k}_m^{\mathrm{T}}(n)\mathbf{x}_m(n). \end{aligned} \tag{13.53}$$

Accordingly, the a priori and a posteriori estimation errors $b_{m,n-1}(n)$ and $b_{m,n}(n)$ of the backward predictor are related according to the equation

$$b_{m,n}(n) = \gamma_m(n) b_{m,n-1}(n). \tag{13.54}$$

3. The observed tap-input vectors to an m-tap joint process estimator are $\mathbf{x}_m(1)$, $\mathbf{x}_m(2), \ldots, \mathbf{x}_m(n)$. Since these are similar to the tap-input vectors to the mth order backward predictor, the conversion factor of the m-tap joint process estimator is also $\gamma_m(n)$, given by (13.53). Accordingly, the a priori and a posteriori estimation errors, $e_{m,n-1}(n)$ and $e_{m,n}(n)$, of the joint process estimator are related according to the equation

$$e_{m,n}(n) = \gamma_m(n) e_{m,n-1}(n). \tag{13.55}$$

13.4.4 Update equation for the conversion factor

First, we show that

$$\boldsymbol{\Psi}_{m+1}^{-1}(n) = \begin{bmatrix} \boldsymbol{\Psi}_m^{-1}(n) & \mathbf{0}_m \\ \mathbf{0}_m^{\mathrm{T}} & 0 \end{bmatrix} + \frac{1}{\zeta_m^{bb}(n)} \tilde{\mathbf{g}}_m(n) \tilde{\mathbf{g}}_m^{\mathrm{T}}(n). \tag{13.56}$$

To this end we multiply the right-hand side of (13.56) by $\boldsymbol{\Psi}_{m+1}(n)$ and show that the result is the $(m+1) \times (m+1)$ identity matrix. The two separate expressions arising from this multiplication are

$$\mathbf{A} = \boldsymbol{\Psi}_{m+1}(n) \begin{bmatrix} \boldsymbol{\Psi}_m^{-1}(n) & \mathbf{0}_m \\ \mathbf{0}_m^{\mathrm{T}} & 0 \end{bmatrix} \tag{13.57}$$

and

$$\mathbf{B} = \frac{1}{\zeta_m^{bb}(n)} \boldsymbol{\Psi}_{m+1}(n) \tilde{\mathbf{g}}_m(n) \tilde{\mathbf{g}}_m^{\mathrm{T}}(n). \tag{13.58}$$

Substituting (13.37) in (13.57), we obtain

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} \boldsymbol{\Psi}_m(n) & \boldsymbol{\psi}_m^{\mathrm{b}}(n) \\ \boldsymbol{\psi}_m^{bT}(n) & \psi^{mm}(n) \end{bmatrix} \begin{bmatrix} \boldsymbol{\Psi}_m^{-1}(n) & \mathbf{0}_m \\ \mathbf{0}_m^{\mathrm{T}} & 0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_m \\ \boldsymbol{\psi}_m^{bT}(n) \boldsymbol{\Psi}_m^{-1}(n) & 0 \end{bmatrix}, \end{aligned} \tag{13.59}$$

where $\mathbf{I}_m$ is the $m \times m$ identity matrix. Furthermore, recalling that $\boldsymbol{\Psi}_m^{-1}(n)$ is a symmetric matrix and using (13.11), we get

$$\boldsymbol{\psi}_m^{\mathrm{bT}}(n) \boldsymbol{\Psi}_m^{-1}(n) = (\boldsymbol{\Psi}_m^{-1}(n) \boldsymbol{\psi}_m^{\mathrm{b}}(n))^{\mathrm{T}} = \mathbf{g}_m^{\mathrm{T}}(n). \tag{13.60}$$

Substituting (13.60) in (13.59), we obtain

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_m \\ \mathbf{g}_m^{\mathrm{T}}(n) & 0 \end{bmatrix}. \tag{13.61}$$

Also, substituting (13.39) in (13.58), we get

$$\mathbf{B} = \begin{bmatrix} \mathbf{0}_m \\ 1 \end{bmatrix} [-\mathbf{g}_m^{\mathrm{T}}(n) \quad 1] = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \\ -\mathbf{g}_m^{\mathrm{T}}(n) & & & 1 \end{bmatrix}. \tag{13.62}$$

Adding the results of (13.61) and (13.62) we obtain

$$\mathbf{A} + \mathbf{B} = \mathbf{I}_{m+1}.$$

This completes the proof of (13.56).

Premultiplying and postmultiplying (13.56) by $\mathbf{x}_{m+1}^{\mathrm{T}}(n)$ and $\mathbf{x}_{m+1}(n)$, respectively, and noting that

$$\mathbf{x}_{m+1}(n) = \begin{bmatrix} \mathbf{x}_m(n) \\ x(n-m) \end{bmatrix} \quad \text{and} \quad \mathbf{k}_m(n) = \boldsymbol{\Psi}_m^{-1}(n)\mathbf{x}_m(n),$$

we obtain

$$\mathbf{x}_{m+1}^{\mathrm{T}}(n)\mathbf{k}_{m+1}(n) = \mathbf{x}_m^{\mathrm{T}}(n)\mathbf{k}_m(n) + \frac{(\tilde{\mathbf{g}}_m^{\mathrm{T}}(n)\mathbf{x}_{m+1}(n))^2}{\zeta_m^{bb}(n)}. \tag{13.63}$$

Using (13.40), we get

$$\begin{aligned} \tilde{\mathbf{g}}_m^{\mathrm{T}}(n)\mathbf{x}_{m+1}(n) &= x(n-m) - \mathbf{g}_m^{\mathrm{T}}(n)\mathbf{x}_m(n) \\ &= b_{m,n}(n). \end{aligned} \tag{13.64}$$

Finally, substituting (13.64) in (13.63), subtracting both sides of the result from one, and recalling (13.53) we obtain

$$\gamma_{m+1}(n) = \gamma_m(n) - \frac{b_{m,n}^2(n)}{\zeta_m^{bb}(n)}. \tag{13.65}$$

13.4.5 Update equation for cross-correlations

The recursions that remain to complete the derivation of the RLSL algorithm are the update equations for cross-correlations $\zeta_m^{fb}(n)$ and $\zeta_m^{be}(n)$.

Recall the RLS recursion for the mth order forward transversal predictor

$$\mathbf{a}_m(n) = \mathbf{a}_m(n-1) + \mathbf{k}_m(n-1)f_{m,n-1}(n), \tag{13.66}$$

where $\mathbf{k}_m(n-1)$ and $f_{m,n-1}(n)$, as defined earlier, are the gain vector and the a priori estimation error of the forward predictor, respectively. The samples of the a posteriori estimation error of the forward predictor, for $k = 1, 2, \ldots, n$, are given by

$$f_{m,n}(k) = x(k) - \mathbf{a}_m^{\mathrm{T}}(n)\mathbf{x}_m(k-1). \tag{13.67}$$

Substituting (13.66) in (13.67) and rearranging, we obtain

$$f_{m,n}(k) = f_{m,n-1}(k) - \mathbf{k}_m^{\mathrm{T}}(n-1)\mathbf{x}_m(k-1)f_{m,n-1}(n), \tag{13.68}$$

where

$$f_{m,n-1}(k) = x(k) - \mathbf{a}_m^{\mathrm{T}}(n-1)\mathbf{x}_m(k-1), \tag{13.69}$$

for $k = 1, 2, \ldots, n$, are samples of the a priori forward prediction error.

Also, recall the RLS recursion for the mth order backward predictor

$$\mathbf{g}_m(n) = \mathbf{g}_m(n-1) + \mathbf{k}_m(n)b_{m,n-1}(n), \tag{13.70}$$

where $\mathbf{k}_m(n)$ and $b_{m,n-1}(n)$, as defined earlier, are the gain vector and a priori estimation error of the backward predictor, respectively. The samples of the a posteriori estimation error of the backward predictor, for $k = 0, 1, \ldots, n$, are given by

$$b_{m,n}(k) = x(k-m) - \mathbf{g}_m^{\mathrm{T}}(n)\mathbf{x}_m(k). \tag{13.71}$$

Substituting (13.70) in (13.71) and rearranging, we obtain

$$b_{m,n}(k) = b_{m,n-1}(k) - \mathbf{k}_m^{\mathrm{T}}(n)\mathbf{x}_m(k)b_{m,n-1}(n), \tag{13.72}$$

where

$$b_{m,n-1}(k) = x(k-m) - \mathbf{g}_m^{\mathrm{T}}(n-1)\mathbf{x}_m(k), \tag{13.73}$$

for $k = 1, 2, \ldots, n$, are samples of the a priori backward prediction error.

Next, substituting (13.68) and (13.72) in (13.28) and expanding we obtain

$$\begin{aligned}
\zeta_m^{fb}(n) = {} & \sum_{k=1}^{n} \lambda^{n-k} f_{m,n-1}(k)b_{m,n-2}(k-1) \\
& - \mathbf{k}_m^{\mathrm{T}}(n-1)b_{m,n-2}(n-1)\sum_{k=1}^{n} \lambda^{n-k} f_{m,n-1}(k)\mathbf{x}_m(k-1) \\
& - \mathbf{k}_m^{\mathrm{T}}(n-1)f_{m,n-1}(n)\sum_{k=1}^{n} \lambda^{n-k} b_{m,n-2}(k-1)\mathbf{x}_m(k-1) \\
& + \mathbf{k}_m^{\mathrm{T}}(n-1)\boldsymbol{\Psi}_m(n-1)\mathbf{k}_m(n-1)f_{m,n-1}(n)b_{m,n-2}(n-1),
\end{aligned} \tag{13.74}$$

where to obtain the last term we have noted that $\mathbf{k}_m^{\mathrm{T}}(n-1)\mathbf{x}_m(n-1) = \mathbf{x}_m^{\mathrm{T}}(n-1)\mathbf{k}_m(n-1)$ and also

$$\sum_{k=1}^{n} \lambda^{n-k}\mathbf{x}_m(k-1)\mathbf{x}_m^{\mathrm{T}}(k-1) = \sum_{k=1}^{n-1} \lambda^{n-1-k}\mathbf{x}_m(k)\mathbf{x}_m^{\mathrm{T}}(k) = \boldsymbol{\Psi}_m(n-1)$$

since $\mathbf{x}_m(0) = \mathbf{0}$ because of prewindowing.

We treat the four terms on the right-hand side of (13.74) separately:

- *First term:* We note that

$$\begin{aligned}
&\sum_{k=1}^{n} \lambda^{n-k} f_{m,n-1}(k) b_{m,n-2}(k-1) \\
&\quad = \lambda \sum_{k=1}^{n-1} \lambda^{n-1-k} f_{m,n-1}(k) b_{m,n-2}(k-1) + f_{m,n-1}(n) b_{m,n-2}(n-1) \\
&\quad = \lambda \zeta_m^{fb}(n-1) + f_{m,n-1}(n) b_{m,n-2}(n-1). \qquad (13.75)
\end{aligned}$$

- *Second term:* We first note that

$$\begin{aligned}
&\sum_{k=1}^{n} \lambda^{n-k} f_{m,n-1}(k) \mathbf{x}_m(k-1) \\
&\quad = \lambda \sum_{k=1}^{n-1} \lambda^{n-1-k} f_{m,n-1}(k) \mathbf{x}_m(k-1) + f_{m,n-1}(n) \mathbf{x}_m(n-1) \\
&\quad = f_{m,n-1}(n) \mathbf{x}_m(n-1),
\end{aligned}$$

where the last equality follows from (13.7), with n replaced by $n-1$. Using this result we obtain

$$\begin{aligned}
&\mathbf{k}_m^{\mathrm{T}}(n-1) b_{m,n-2}(n-1) \sum_{k=1}^{n} \lambda^{n-k} f_{m,n-1}(k) \mathbf{x}_m(k-1) \\
&\quad = \mathbf{k}_m^{\mathrm{T}}(n-1) \mathbf{x}_m(n-1) f_{m,n-1}(n) b_{m,n-2}(n-1). \qquad (13.76)
\end{aligned}$$

- *Third term:* Using the change of variable $l = k-1$, we get

$$\begin{aligned}
\sum_{k=1}^{n} \lambda^{n-k} b_{m,n-2}(k-1) \mathbf{x}_m(k-1) &= \sum_{l=0}^{n-1} \lambda^{n-1-l} b_{m,n-2}(l) \mathbf{x}_m(l) \\
&= \sum_{l=1}^{n-1} \lambda^{n-1-l} b_{m,n-2}(l) \mathbf{x}_m(l) \\
&= \lambda \sum_{l=1}^{n-2} \lambda^{n-2-l} b_{m,n-2}(l) \mathbf{x}_m(l) + b_{m,n-2}(n-1) \mathbf{x}_m(n-1) \\
&= b_{m,n-2}(n-1) \mathbf{x}_m(n-1),
\end{aligned}$$

where we have used $\mathbf{x}_m(0) = \mathbf{0}$ (because of prewindowing) in the second step and (13.16) with n replaced by $n-2$ for the last step. Using this result, we obtain

$$\begin{aligned}&\mathbf{k}_m^{\mathrm{T}}(n-1)f_{m,n-1}(n)\sum_{k=1}^{n}\lambda^{n-k}b_{m,n-1}(k-1)\mathbf{x}_m(k-1)\\&\quad=\mathbf{k}_m^{\mathrm{T}}(n-1)\mathbf{x}_m(n-1)f_{m,n-1}(n)b_{m,n-2}(n-1).\end{aligned}\tag{13.77}$$

- *Fourth term:* Using (13.10), we get

$$\boldsymbol{\Psi}_m(n-1)\mathbf{k}_m(n-1)=\mathbf{x}_m(n-1).$$

Thus,

$$\begin{aligned}&\mathbf{k}_m^{\mathrm{T}}(n-1)\boldsymbol{\Psi}_m(n-1)\mathbf{k}_m(n-1)f_{m,n-1}(n)b_{m,n-2}(n-1)\\&\quad=\mathbf{k}_m^{\mathrm{T}}(n-1)\mathbf{x}_m(n-1)f_{m,n-1}(n)b_{m,n-2}(n-1).\end{aligned}\tag{13.78}$$

Substituting (13.75), (13.76), (13.77) and (13.78) in (13.74) and rearranging, we obtain

$$\zeta_m^{fb}(n)=\lambda\zeta_m^{fb}(n-1)+(1-\mathbf{k}_m^{\mathrm{T}}(n-1)\mathbf{x}_m(n-1))f_{m,n-1}(n)b_{m,n-2}(n-1).\tag{13.79}$$

Next, noting that $1-\mathbf{k}_m^{\mathrm{T}}(n-1)\mathbf{x}_m(n-1)=\gamma_m(n-1)$ and $\gamma_m(n-1)b_{m,n-2}(n-1)=b_{m,n-1}(n-1)$, according to (13.53) and (13.54), respectively, (13.79) can be simplified as

$$\zeta_m^{fb}(n)=\lambda\zeta_m^{fb}(n-1)+f_{m,n-1}(n)b_{m,n-1}(n-1).\tag{13.80}$$

Following a similar line of derivations, we also obtain

$$\zeta_m^{be}(n)=\lambda\zeta_m^{be}(n-1)+e_{m,n-1}(n)b_{m,n-1}(n).\tag{13.81}$$

We have now developed all the basic equations/recursions necessary for implementation of the RLSL algorithms.

13.4.6 The RLSL algorithm using a posteriori errors

Table 13.1 presents a possible implementation of the RLSL algorithm that uses the a posteriori estimation errors. For every iteration the algorithm begins with the initial values of $f_{0,n}(n)$, $b_{0,n}(n)$, $e_{0,n}(n)$ and $\gamma_0(n)$ as inputs to the first stage and proceeds with updating the successive stages of the lattice in a *for loop*. The operations in this loop may be divided into those related to the forward and backward predictions and the operations related to the filtering. In the prediction part, the recursive equations (13.45), (13.46) and (13.80) are used to update $\zeta_m^{ff}(n)$, $\zeta_m^{bb}(n)$ and $\zeta_m^{fb}(n)$, respectively. Here, we have also used (13.52) and (13.54) to write the recursions in terms of the a posteriori estimation errors, $f_{m,n}(n)$ and $b_{m,n}(n)$, only. The results of these recursions are then used to calculate the PARCOR coefficients $\kappa_{m+1}^{\mathrm{f}}(n)$ and $\kappa_{m+1}^{\mathrm{b}}(n)$ according to

Table 13.1 RLSL algorithm using the a posteriori estimation errors

Input: Latest sample of input, $x(n)$,
Past values of
the backward a posteriori estimation errors, $b_{m,n-1}(n-1)$,
the auto- and cross-correlations, $\zeta_m^{ff}(n-1), \zeta_m^{bb}(n-1), \zeta_m^{fb}(n-1)$ and $\zeta_m^{be}(n-1)$,
the conversion factors, $\gamma_m(n-1)$,
for $m = 0, 1, \ldots, N-1$.

Output: The updated values of
the backward a posteriori estimation errors, $b_{m,n}(n)$
the auto- and cross-correlations, $\zeta_m^{ff}(n), \zeta_m^{bb}(n), \zeta_m^{fb}(n)$ and $\zeta_m^{be}(n)$,
the conversion factors, $\gamma_m(n)$,
for $m = 0, 1, \ldots, N-1$.
The lattice coefficients are also available at the end of each iteration.

$f_{0,n}(n) = b_{0,n}(n) = x(n)$
$e_{0,n}(n) = d(n)$
$\gamma_0(n) = 1$
for $m = 0$ to $N-1$

$$\zeta_m^{ff}(n) = \lambda\zeta_m^{ff}(n-1) + \frac{f_{m,n}^2(n)}{\gamma_m(n-1)}$$

$$\zeta_m^{bb}(n) = \lambda\zeta_m^{bb}(n-1) + \frac{b_{m,n}^2(n)}{\gamma_m(n)}$$

$$\zeta_m^{fb}(n) = \lambda\zeta_m^{fb}(n-1) + \frac{f_{m,n}(n)b_{m,n-1}(n-1)}{\gamma_m(n-1)}$$

$$\kappa_{m+1}^{\mathrm{f}}(n) = \frac{\zeta_m^{fb}(n)}{\zeta_m^{bb}(n-1)}$$

$$\kappa_{m+1}^{\mathrm{b}}(n) = \frac{\zeta_m^{fb}(n)}{\zeta_m^{ff}(n)}$$

$$f_{m+1,n}(n) = f_{m,n}(n) - \kappa_{m+1}^{\mathrm{f}}(n)b_{m,n-1}(n-1)$$

$$b_{m+1,n}(n) = b_{m,n-1}(n-1) - \kappa_{m+1}^{\mathrm{b}}(n)f_{m,n}(n)$$

$$\zeta_m^{be}(n) = \lambda\zeta_m^{be}(n-1) + \frac{e_{m,n}(n)b_{m,n}(n)}{\gamma_m(n)}$$

$$c_m(n) = \frac{\zeta_m^{be}(n)}{\zeta_m^{bb}(n)}$$

$$e_{m+1,n}(n) = e_{m,n}(n) - c_m(n)b_{m,n}(n)$$

$$\gamma_{m+1}(n) = \gamma_m(n) - \frac{b_{m,n}^2(n)}{\zeta_m^{bb}(n)}$$

end

(13.30) and (13.31), respectively. This follows with the order-update equations for computation of the a posteriori estimation errors of the forward and backward predictors. These follow from Figure 13.3 – see also Chapter 11. The filtering is done in a similar way using the recursion (13.47) and equations (13.32) and (13.55). Finally, the conversion factor $\gamma_m(n)$ is updated according to recursion (13.65).

Theoretically, the autocorrelations, $\zeta_m^{ff}(n)$ and $\zeta_m^{bb}(n)$, should be initialized to zero. However, since such initialization results in division by zeros during the first few iterations of the algorithm, $\zeta_m^{ff}(0)$ and $\zeta_m^{bb}(0)$, for $m = 0, 1, \ldots, N-1$, are initialized to a small positive number, δ, to prevent these numerical difficulties. The cross-correlations $\zeta_m^{fb}(0)$ and $\zeta_m^{be}(0)$ are initialized to the value of zero.

13.4.7 The RLSL algorithm with error feedback

The RLSL algorithm given in Table 13.1 uses the auto- and cross-correlations of the input signals to the successive stages of the lattice to calculate the coefficients $\kappa_m^{\mathrm{f}}(n)$, $\kappa_m^{\mathrm{b}}(n)$ and $c_m(n)$, according to equations (13.30), (13.31) and (13.32), respectively. Alternatively, we can develop a set of recursive equations for updating the coefficients $\kappa_m^{\mathrm{f}}(n)$, $\kappa_m^{\mathrm{b}}(n)$ and $c_m(n)$. This leads to an alternative implementation of the RLSL algorithm which has been found to be *less sensitive* to numerical errors as compared with the algorithm of Table 13.1 (Ling, 1993).

Table 13.2 summarizes this alternative implementation of the RLSL algorithm. We note that here all the errors are the a priori ones, while in Table 13.1 all the equations are in terms of the a posteriori errors. In addition, the update equations of the cross-correlations $\zeta_m^{fb}(n)$ and $\zeta_m^{be}(n)$ have been deleted in Table 13.2, as they are no longer required. Instead, there are three recursions for time-updating the coefficients of the lattice. Next, we explain the derivation of one of these recursions as an example. The other two can be derived by following the same line of derivation.

Recall that

$$\kappa_{m+1}^{\mathrm{f}}(n) = \frac{\zeta_m^{fb}(n)}{\zeta_m^{bb}(n-1)}. \tag{13.82}$$

Substituting (13.80) in (13.82), we get

$$\kappa_{m+1}^{\mathrm{f}}(n) = \frac{\lambda\zeta_m^{fb}(n-1)}{\zeta_m^{bb}(n-1)} + \frac{f_{m,n-1}(n)b_{m,n-1}(n-1)}{\zeta_m^{bb}(n-1)}. \tag{13.83}$$

But,

$$\begin{aligned}
\frac{\lambda\zeta_m^{fb}(n-1)}{\zeta_m^{bb}(n-1)} &= \frac{\zeta_m^{fb}(n-1)}{\zeta_m^{bb}(n-2)} \cdot \frac{\lambda\zeta_m^{bb}(n-2)}{\zeta_m^{bb}(n-1)} \\
&= \kappa_{m+1}^{\mathrm{f}}(n-1)\frac{\lambda\zeta_m^{bb}(n-2)}{\zeta_m^{bb}(n-1)} \\
&= \kappa_{m+1}^{\mathrm{f}}(n-1)\frac{\zeta_m^{bb}(n-1) - b_{m,n-1}(n-1)b_{m,n-2}(n-1)}{\zeta_m^{bb}(n-1)} \\
&= \kappa_{m+1}^{\mathrm{f}}(n-1) - \frac{\kappa_{m+1}^{\mathrm{f}}(n-1)b_{m,n-1}(n-1)b_{m,n-2}(n-1)}{\zeta_m^{bb}(n-1)},
\end{aligned} \tag{13.84}$$

Table 13.2 RLSL algorithm using the a priori estimation errors with error feedback

Input: Latest sample of input, $x(n)$,
Past values of
the backward a priori estimation errors, $b_{m,n-2}(n-1)$,
the autocorrelations, $\zeta_m^{ff}(n-1), \zeta_m^{bb}(n-1)$,
the lattice coefficients $\kappa_{m+1}^{f}(n-1), \kappa_{m+1}^{b}(n-1)$ and $c_m(n-1)$,
the conversion factors, $\gamma_m(n-1)$,
for $m = 0, 1, \ldots, N-1$

Output: The updated values of
the backward a priori estimation errors, $b_{m,n-1}(n)$
the autocorrelations, $\zeta_m^{ff}(n), \zeta_m^{bb}(n)$,
the lattice coefficients $\kappa_{m+1}^{f}(n), \kappa_{m+1}^{b}(n)$ and $c_m(n)$,
the conversion factors, $\gamma_m(n)$,
for $m = 0, 1, \ldots, N-1$.

$f_{0,n-1}(n) = b_{0,n-1}(n) = x(n)$

$e_{0,n-1}(n) = d(n)$

$\gamma_0(n) = 1$

for $m = 0$ to $N-1$

$$\zeta_m^{ff}(n) = \lambda\zeta_m^{ff}(n-1) + \gamma_m(n-1)f_{m,n-1}^2(n)$$

$$\zeta_m^{bb}(n) = \lambda\zeta_m^{bb}(n-1) + \gamma_m(n)b_{m,n-1}^2(n)$$

$$f_{m+1,n-1}(n) = f_{m,n-1}(n) - \kappa_{m+1}^{f}(n-1)b_{m,n-2}(n-1)$$

$$b_{m+1,n-1}(n) = b_{m,n-2}(n-1) - \kappa_{m+1}^{b}(n-1)f_{m,n-1}(n)$$

$$\kappa_{m+1}^{f}(n) = \kappa_{m+1}^{f}(n-1) + \frac{\gamma_m(n-1)b_{m,n-2}(n-1)}{\zeta_m^{bb}(n-1)}f_{m+1,n-1}(n)$$

$$\kappa_{m+1}^{b}(n) = \kappa_{m+1}^{b}(n-1) + \frac{\gamma_m(n-1)f_{m,n-1}(n)}{\zeta_m^{ff}(n)}b_{m+1,n-1}(n)$$

$$e_{m+1,n-1}(n) = e_{m,n-1}(n) - c_m(n-1)b_{m,n-1}(n)$$

$$c_m(n) = c_m(n-1) - \frac{\gamma_m(n)b_{m,n-1}(n)}{\zeta_m^{bb}(n)}e_{m+1,n-1}(n)$$

$$\gamma_{m+1}(n) = \gamma_m(n) - \frac{\gamma_m^2(n)b_{m,n-1}^2(n)}{\zeta_m^{bb}(n)}$$

end

where we have used (13.46) to replace $\lambda\zeta_m^{bb}(n-2)$ by $\zeta_m^{bb}(n-1) - b_{m,n-1}(n-1)b_{m,n-2}(n-1)$. Substituting (13.84) in (13.83) and rearranging, we get

$$\begin{aligned}\kappa_{m+1}^{f}(n) &= \kappa_{m+1}(n-1) + \frac{b_{m,n-1}(n-1)}{\zeta_m^{bb}(n-1)}\left(f_{m,n-1}(n) - \kappa_{m+1}^{f}(n-1)b_{m,n-2}(n-1)\right) \\ &= \kappa_{m+1}^{f}(n-1) + \frac{b_{m,n-1}(n-1)f_{m+1,n-1}(n)}{\zeta_m^{bb}(n-1)}. \end{aligned} \tag{13.85}$$

Finally, using (13.54) to convert the a posteriori estimation error $b_{m,n-1}(n-1)$ to its equivalent a priori estimation error $b_{m,n-2}(n-1)$, in (13.85), we get

$$\kappa_{m+1}^{f}(n) = \kappa_{m+1}^{f}(n-1) + \frac{\gamma_m(n-1)b_{m,n-2}(n-1)}{\zeta_m^{bb}(n-1)} f_{m+1,n-1}(n), \tag{13.86}$$

which is the recursion used in Table 13.2, for the adaptation of $\kappa_{m+1}^{f}(n)$. Following the same line of derivation, we can also obtain the recursions associated with the adaptation of $\kappa_{m+1}^{b}(n)$ and $c_m(n)$. These are left to the reader as exercises.

13.5 The FTRLS algorithm

Fast transversal filter (FTF) or fast transversal RLS (FTRLS) algorithm is another alternative numerical technique for solving the least-squares problem. The main advantage of the FTRLS algorithm is its reduced computational complexity as compared with other available solutions, such as the standard RLS and RLSL algorithms. Table 13.3 summarizes the number of operations (additions, multiplications and divisions) required in each iteration of the standard RLS algorithm (Table 12.2), the two versions of the RLSL algorithm presented in Tables 13.1 and 13.2, and also the two versions of the FTRLS algorithm that will be discussed in this section, as an indication of their computational complexity.[2] We note that as the filter length, N, increases, the standard RLS becomes a rather expensive algorithm because its computational complexity grows in proportion to the square of the filter length. On the other hand, the computational complexities of RLSL and FTRLS algorithms grow only linearly with filter length. In addition, we find that the FTRLS algorithm has only about half the complexity of the RLSL algorithm. However, unfortunately, such a significant reduction in the complexity of the FTRLS algorithm does not come for free. Computer simulations and also theoretical studies have shown that the FTRLS algorithms are, in general, highly sensitive to round-off error accumulation. Precautions have to be taken to deal with this problem to prevent the algorithm from becoming unstable. It is generally suggested that the algorithms should be reinitialized once a sign of instability is observed (Eleftheriou and Falconer, 1987, and Cioffi and Kailath, 1984). To reduce the chance of instability in the FTRLS algorithm, a new version that is more robust against round-off error accumulation has been proposed by Slock and Kailath (1988, 1991). This is called the stabilized FTRLS (SFTRLS) algorithm. However, studies show that even the SFTRLS algorithm has some limitation in the sense that it becomes unstable when the forgetting factor, λ, is not close enough to one. This definitely limits the applicability of the FTRLS algorithm in cases where smaller values of λ should be used to achieve fast tracking (see the next chapter).

[2] We note that the number of operations, in general, may not be a fair measure in comparing various algorithms. A fair comparison would only be possible if the platform over which the algorithms are implemented is known a priori. For example, in hardware implementation, the modular structure of the RLSL may be very beneficial when a pipe-line structure is considered - see Ling (1993).

Table 13.3 Computational complexity of various RLS algorithms

Algorithm	No. of $+$, $\times$ and $\div$ (added)
RLS (Table 12.2)	$3.5N^2$
RLSL (Table 13.1)	$28N$
RLSL (Table 13.2)	$31N$
FTRLS (Table 13.4)	$14N$
FTRLS (stabilized)	$18N$

13.5.1 Derivation of the FTRLS algorithm

The FTRLS algorithm, basically, takes advantage of the interrelationships that exist between the forward and backward predictors as well as the joint process estimator when they share the same set of input samples. In particular, in the development of the RLSL algorithm in Section 13.4 we found that the forward and backward predictors and also the joint process estimator share the same conversion factor and gain vector. These properties led to a number of order- and time-update equations which were eventually put together to obtain the RLSL algorithm. In the RLSL algorithm, the problem of prediction and filtering (joint process estimation) is solved for orders of 1 to N simultaneously. In cases where the goal is to solve the problem only for a filter of length N, this solution clearly has many redundant elements which may unnecessarily complicate the solution. Accordingly, a set of equations that is limited to order N predictors and also to a length N filter (joint process estimator) may give a more efficient solution. This is the main essence of the FTRLS algorithm when it is viewed as an improvement to the RLSL algorithm.

To have a clear treatment of the FTRLS algorithm, we proceed with the derivations of the necessary recursions separated into three subsections. Namely, forward prediction, backward prediction and filtering.

Forward prediction

Consider an Nth order forward transversal predictor with tap-weight vector $\mathbf{a}_N(n)$ and tap-input vector $\mathbf{x}_N(k-1) = [x(k-1)\ x(k-2)\ \ldots\ x(k-N)]^T$, for $k = 1, 2, \ldots, n$. The RLS recursion for the adaptive adjustment of $\mathbf{a}_N(n)$ is

$$\mathbf{a}_N(n) = \mathbf{a}_N(n-1) + \mathbf{k}_N(n-1)f_{N,n-1}(n), \tag{13.87}$$

where $\mathbf{k}_N(n-1)$ is the gain vector of the adaptation as defined in (13.10) and $f_{N,n-1}(n)$ is the a priori estimation error of the forward predictor.

Let us define the normalized gain vector

$$\bar{\mathbf{k}}_N(n) = \frac{\mathbf{k}_N(n)}{\gamma_N(n)}, \tag{13.88}$$

where $\gamma_N(n)$ is the conversion factor as defined before. Substituting (13.88) and (13.52) in (13.87), we get

$$\begin{aligned}\mathbf{a}_N(n) &= \mathbf{a}_N(n-1) + \bar{\mathbf{k}}_N(n-1)\gamma_N(n-1)f_{N,n-1}(n) \\ &= \mathbf{a}_N(n-1) + \bar{\mathbf{k}}_N(n-1)f_{N,n}(n),\end{aligned} \tag{13.89}$$

where $f_{N,n}(n)$ is the a posteriori estimation error of the forward predictor. Furthermore, using the definition (13.36), we may rewrite (13.89) as

$$\tilde{\mathbf{a}}_N(n) = \tilde{\mathbf{a}}_N(n-1) - \begin{bmatrix} 0 \\ \bar{\mathbf{k}}_N(n-1) \end{bmatrix} f_{N,n}(n). \tag{13.90}$$

Next, we note that

$$\boldsymbol{\Psi}_{N+1}^{-1}(n) = \begin{bmatrix} 0 & \mathbf{0}_N^{\mathrm{T}} \\ \mathbf{0}_N & \boldsymbol{\Psi}_N^{-1}(n-1) \end{bmatrix} + \frac{1}{\zeta_N^{ff}(n)} \tilde{\mathbf{a}}_N(n)\tilde{\mathbf{a}}_N^{\mathrm{T}}(n). \tag{13.91}$$

This identity, which appears similar to (13.56), can also be proved in the same way as (13.56). This is left to the reader as an exercise. Postmultiplying (13.91) by $\mathbf{x}_{N+1}(n)$, recalling (13.36), (13.5), (13.19) and (13.88), and noting that

$$\mathbf{x}_{N+1}(n) = \begin{bmatrix} x(n) \\ \mathbf{x}_N(n-1) \end{bmatrix},$$

we get

$$\gamma_{N+1}(n)\bar{\mathbf{k}}_{N+1}(n) = \gamma_N(n-1)\begin{bmatrix} 0 \\ \bar{\mathbf{k}}_N(n-1) \end{bmatrix} + \frac{f_{N,n}(n)}{\zeta_N^{ff}(n)} \tilde{\mathbf{a}}_N(n). \tag{13.92}$$

Substituting (13.90) in (13.92) and rearranging we obtain

$$\gamma_{N+1}(n)\bar{\mathbf{k}}_{N+1}(n) = \left(\gamma_N(n-1) - \frac{f_{N,n}^2(n)}{\zeta_N^{ff}(n)}\right)\begin{bmatrix} 0 \\ \bar{\mathbf{k}}_N(n-1) \end{bmatrix} + \frac{f_{N,n}(n)}{\zeta_N^{ff}(n)} \tilde{\mathbf{a}}_N(n-1). \tag{13.93}$$

On the other hand, post- and premultiplying (13.91) by $\mathbf{x}_{N+1}(n)$ and $\mathbf{x}_{N+1}^{\mathrm{T}}(n)$, respectively, subtracting both sides of the result from unity, and recalling (13.53) we obtain

$$\gamma_{N+1}(n) = \gamma_N(n-1) - \frac{f_{N,n}^2(n)}{\zeta_N^{ff}(n)}. \tag{13.94}$$

Substituting (13.94) in (13.93) and dividing both sides of the result by $\gamma_{N+1}(n)$, we get

$$\bar{\mathbf{k}}_{N+1}(n) = \begin{bmatrix} 0 \\ \bar{\mathbf{k}}_N(n-1) \end{bmatrix} + \frac{f_{N,n}(n)}{\gamma_{N+1}(n)\zeta_N^{ff}(n)} \tilde{\mathbf{a}}_N(n-1). \tag{13.95}$$

Moreover, combining (13.94) and (13.45), it is straightforward to show that (Problem P13.17)

$$\gamma_{N+1}(n)\zeta_N^{ff}(n) = \lambda\gamma_N(n-1)\zeta_N^{ff}(n-1). \tag{13.96}$$

Finally, substituting (13.96) in (13.95) and using (13.52), we obtain

$$\bar{\mathbf{k}}_{N+1}(n) = \begin{bmatrix} 0 \\ \bar{\mathbf{k}}_N(n-1) \end{bmatrix} + \lambda^{-1}\frac{f_{N,n-1}(n)}{\zeta_N^{ff}(n-1)}\tilde{\mathbf{a}}_N(n-1). \tag{13.97}$$

This recursion gives a time as well as order update of the normalized gain vector. Next, we develop another recursion that keeps the time index of the normalized gain vector fixed at n, but reduces its length from $N+1$ to N. This also leads to a time update of the tap-weight vector of the backward predictor.

Backward prediction

Consider (13.56) with $m = N$. Then, postmultiplying it by $\mathbf{x}_{N+1}(n)$ and recalling (13.40) and (13.88), we get

$$\gamma_{N+1}(n)\bar{\mathbf{k}}_{N+1}(n) = \gamma_N(n)\begin{bmatrix} \bar{\mathbf{k}}_N(n) \\ 0 \end{bmatrix} + \frac{b_{N,n}(n)}{\zeta_N^{bb}(n)}\tilde{\mathbf{g}}_N(n). \tag{13.98}$$

Equating the last elements of the vectors on both sides of (13.98) and rearranging, we obtain

$$\bar{k}_{N+1,N+1}(n) = \frac{b_{N,n}(n)}{\gamma_{N+1}(n)\zeta_N^{bb}(n)}, \tag{13.99}$$

where $\bar{k}_{N+1,N+1}(n)$ denotes the last element of $\bar{\mathbf{k}}_{N+1}(n)$. On the other hand, combining (13.46) and (13.65), and replacing m by N, it is straightforward to show that (Problem P13.18)

$$\gamma_{N+1}(n)\zeta_N^{bb}(n) = \lambda\gamma_N(n)\zeta_N^{bb}(n-1). \tag{13.100}$$

Substituting (13.100) in (13.99), recalling (13.54), and rearranging the result, we get

$$b_{N,n-1}(n) = \lambda\zeta_N^{bb}(n-1)\bar{k}_{N+1,N+1}(n). \tag{13.101}$$

Furthermore, solving (13.100) for $\gamma_N(n)$ and using (13.46), we obtain

$$\begin{aligned}\gamma_N(n) &= \frac{\zeta_N^{bb}(n)}{\lambda\zeta_N^{bb}(n-1)}\gamma_{N+1}(n) \\ &= \left(\frac{\lambda\zeta_N^{bb}(n-1)}{\zeta_N^{bb}(n)}\right)^{-1}\gamma_{N+1}(n) \\ &= \left(1 - \frac{b_{N,n}(n)b_{N,n-1}(n)}{\zeta_N^{bb}(n)}\right)^{-1}\gamma_{N+1}(n).\end{aligned} \tag{13.102}$$

Substituting (13.99) in (13.102) we get

$$\gamma_N(n) = \left(1 - b_{N,n-1}(n)\gamma_{N+1}(n)\bar{k}_{N+1,N+1}(n)\right)^{-1}\gamma_{N+1}(n). \tag{13.103}$$

Moreover, we note that the update recursion (13.18) (with $m = N$) may be rearranged as

$$\tilde{\mathbf{g}}_N(n) = \tilde{\mathbf{g}}_N(n-1) - \begin{bmatrix} \bar{\mathbf{k}}_N(n) \\ 0 \end{bmatrix} b_{N,n}(n). \tag{13.104}$$

Substituting (13.104) in (13.98) and rearranging, we obtain

$$\gamma_{N+1}(n)\bar{\mathbf{k}}_{N+1}(n) = \left(\gamma_N(n) - \frac{b_{N,n}^2(n)}{\zeta_N^{bb}(n)}\right)\begin{bmatrix} \bar{\mathbf{k}}_N(n) \\ 0 \end{bmatrix} + \frac{b_{N,n}(n)}{\zeta_N^{bb}(n)}\tilde{\mathbf{g}}_N(n-1). \tag{13.105}$$

Finally, substituting (13.65) with $m = N$ in (13.105), dividing both sides of the result by $\gamma_{N+1}(n)$, and using (13.99), we get

$$\begin{bmatrix} \bar{\mathbf{k}}_N(n) \\ 0 \end{bmatrix} = \bar{\mathbf{k}}_{N+1}(n) - \bar{k}_{N+1,N+1}(n)\tilde{\mathbf{g}}_N(n-1). \tag{13.106}$$

With this recursion we recover the updated value of the gain vector in the right order, N. We thus can proceed with the next iteration of predictions and also use $\bar{\mathbf{k}}_N(n)$ for adaptation of the tap-weight vector, $\hat{\mathbf{w}}_N(n)$, of an adaptive filter with tap-input vector $\mathbf{x}_N(n)$, as explained below.

Filtering

Having obtained the normalized gain vector $\bar{\mathbf{k}}_N(n)$, the following equations may be used for adaptation of the tap-weight vector, $\hat{\mathbf{w}}_N(n)$, of an adaptive filter with tap-input vector $\mathbf{x}_N(n)$.

We first obtain the a priori estimation error

$$e_{N,n-1}(n) = d(n) - \hat{\mathbf{w}}_N^{\mathrm{T}}(n-1)\mathbf{x}_N(n). \tag{13.107}$$

Then, we calculate the corresponding a posteriori estimation error

$$e_{N,n}(n) = \gamma_N(n)e_{N,n-1}(n). \tag{13.108}$$

Finally, the update of the tap-weight vector of the adaptive filter is done according to the recursion

$$\hat{\mathbf{w}}_N(n) = \hat{\mathbf{w}}_N(n-1) + \bar{\mathbf{k}}_N(n)e_{N,n}(n). \tag{13.109}$$

We note that (13.109) is the same as the recursion (12.52). The only difference is that (13.109) is written in terms of the normalized gain vector $\bar{\mathbf{k}}_N(n)$ and, as a result, the a priori estimation error $e_{N,n-1}(n)$ is replaced by the a posteriori estimation error $e_{N,n}(n)$.

13.5.2 Summary of the FTRLS algorithm

Table 13.4 summarizes the FTRLS algorithm by collecting together the relevant equations from Section 13.4 and some of the new results that were developed in this section.

As mentioned before, the FTRLS algorithm may experience numerical instability. To deal with this problem, it has been noted that the sign of the expression

$$\beta(n) = 1 - b_{N,n-1}(n)\gamma_{N+1}(n)\bar{k}_{N+1,N+1}(n) \tag{13.110}$$

is a good indication of the state of the algorithm with regard to its numerical instability. From (13.103), we note that $\beta(n) = \gamma_{N+1}(n)/\gamma_N(n)$ and this always has to be positive,

Table 13.4 The FTRLS algorithm

Input: Tap-input vector $\mathbf{x}_{N+1}(n-1)$, desired output $d(n)$,
Tap-weight vectors $\tilde{\mathbf{a}}_N(n-1)$, $\tilde{\mathbf{g}}_N(n-1)$ and $\hat{\mathbf{w}}_N(n-1)$,
Normalized gain vector $\bar{\mathbf{k}}_N(n-1)$, and least-squares sums $\zeta_N^{ff}(n-1)$ and $\zeta_N^{bb}(n-1)$.

Output: The updated values of
$\tilde{\mathbf{a}}_N(n), \tilde{\mathbf{g}}_N(n), \hat{\mathbf{w}}_N(n), \bar{\mathbf{k}}_N(n), \zeta_N^{ff}(n)$ and $\zeta_N^{bb}(n)$.

Prediction:

$$f_{N,n-1}(n) = \tilde{\mathbf{a}}_N^{\mathrm{T}}(n-1)\mathbf{x}_{N+1}(n)$$
$$f_{N,n}(n) = \gamma_N(n-1)f_{N,n-1}(n)$$
$$\zeta_N^{ff}(n) = \lambda\zeta_N^{ff}(n-1) + f_{N,n}(n)f_{N,n-1}(n)$$
$$\gamma_{N+1}(n) = \lambda\frac{\zeta_N^{ff}(n-1)}{\zeta_N^{ff}(n)}\gamma_N(n-1)$$
$$\bar{\mathbf{k}}_{N+1}(n) = \begin{bmatrix} 0 \\ \bar{\mathbf{k}}_N(n-1) \end{bmatrix} + \lambda^{-1}\frac{f_{N,n-1}(n)}{\zeta_N^{ff}(n-1)}\tilde{\mathbf{a}}_N(n-1)$$
$$\tilde{\mathbf{a}}_N(n) = \tilde{\mathbf{a}}_N(n-1) - \begin{bmatrix} 0 \\ \bar{\mathbf{k}}_N(n-1) \end{bmatrix} f_{N,n}(n)$$
$$b_{N,n-1}(n) = \lambda\zeta_N^{bb}(n-1)\bar{k}_{N+1,N+1}(n)$$
$$\beta(n) = 1 - b_{N,n-1}(n)\gamma_{N+1}(n)\bar{k}_{N+1,N+1}(n) \qquad \text{(rescue variable)}$$
$$\gamma_N(n) = \beta^{-1}(n)\gamma_{N+1}(n)$$
$$b_{N,n}(n) = \gamma_N(n)b_{N,n-1}(n)$$
$$\zeta_N^{bb}(n) = \lambda\zeta_N^{bb}(n-1) + b_{N,n}(n)b_{N,n-1}(n)$$
$$\begin{bmatrix} \bar{\mathbf{k}}_N(n) \\ 0 \end{bmatrix} = \bar{\mathbf{k}}_{N+1}(n) - \bar{k}_{N+1,N+1}(n)\tilde{\mathbf{g}}_N(n-1)$$
$$\tilde{\mathbf{g}}_N(n) = \tilde{\mathbf{g}}_N(n-1) - \begin{bmatrix} \bar{\mathbf{k}}_N(n) \\ 0 \end{bmatrix} b_{N,n}(n)$$

Filtering:

$$e_{N,n-1}(n) = d(n) - \hat{\mathbf{w}}_N^{\mathrm{T}}(n-1)\mathbf{x}_N(n)$$
$$e_{N,n}(n) = \gamma_N(n)e_{N,n-1}(n)$$
$$\hat{\mathbf{w}}_N(n) = \hat{\mathbf{w}}_N(n-1) + \bar{\mathbf{k}}_N(n)e_{N,n}(n)$$

since the conversion factors, $\gamma_N(n)$ and $\gamma_{N+1}(n)$, are non-negative quantities (see Problem P13.8). However, studies have shown that the FTRLS algorithm has some unstable modes that are not excited when infinite precision is assumed for arithmetics. Under finite precision arithmetics, these unstable modes receive some excitation which will lead to some misbehaviour of the algorithm and eventually result in its divergence. In particular, it has been noted that the quantity $\beta(n)$ becomes negative just before divergence of the algorithm occurs (Cioffi and Kailath, 1984). For this reason, $\beta(n)$ is called the *rescue variable* and it is suggested that once a negative value of $\beta(n)$ is observed, the normal execution of the FTRLS algorithm must be stopped and it should be restarted. In that case, the latest values of the filter coefficients may be used for a soft reinitialization of the algorithm (see Cioffi and Kailath, 1984, for the reinitialization procedure).

13.5.3 The stabilized FTRLS algorithm

Further developments in the FTRLS algorithm has shown that the use of a special *error feedback* mechanism can greatly stabilize the FTRLS algorithm. It has been noticed that by introducing *computational redundancy*, by computing certain quantities in different ways, specific measurements of the numerical errors present can be made. These measurements can then be *fed back* to modify the dynamics of error propagation such that the unstable modes of the FTRLS algorithm are stabilized (Slock and Kailath, 1988, 1991). The quantities that have been identified to be appropriate for this purpose are the backward prediction error $b_{N,n-1}(n)$, the conversion factor $\gamma_{N+1}(n)$ and the last element of the normalized gain vector $\bar{\mathbf{k}}_{N+1}(n)$, i.e. the three quantities used in the computation of $\beta(n)$ in (13.110). Slock and Kailath (1991) have proposed an elegant procedure for exploiting these redundancies in the FTRLS algorithm and have come up with a stabilized version of the FTRLS algorithm. However, as was noted before, even the stabilized FTRLS algorithm has to be treated with some special care which makes it rather restrictive in applications. In particular, it has been found that the stability of the SFTRLS can only be guaranteed when the forgetting factor, λ, is chosen very close to one. As a rule of thumb, it is suggested that λ should be kept within the range

$$1 - \frac{1}{2N} < \lambda < 1, \tag{13.111}$$

where N is the length of the filter.

Problems

P13.1 Starting with (13.4) and using the principle of orthogonality, derive (13.8). Also, by inserting the relevant variables in (12.16), suggest an alternative derivation of (13.8).

P13.2 Following similar lines of derivations as those in Problem P13.1, suggest two methods for the derivation of (13.17).

P13.3 Work out the details of derivations of the augmented normal equations (13.35) and (13.39).

P13.4 Give a detailed derivation of (13.23) and (13.24).

P13.5 By using the principle of orthogonality, prove (13.25).

P13.6 Consider the a posteriori forward and backward prediction errors $f_{i,n}(k)$ and $b_{j,n}(k)$, respectively, of a real-valued and prewindowed signal sequence $x(k)$. Prove the following results:

(i) $$\sum_{k=1}^{n} \lambda^{n-k} f_{6,n}(k) f_{5,n}(k-1) = 0.$$

(ii) $$\sum_{k=1}^{n} \lambda^{n-k} f_{m,n}(k) x(k) = \sum_{k=1}^{n} \lambda^{n-k} f_{m,n}^2(k).$$

(iii) $$\sum_{k=1}^{n} \lambda^{n-k} b_{m,n}(k) x(k-m) = \sum_{k=1}^{n} \lambda^{n-k} b_{m,n}^2(k).$$

(iv) For $0 < l < m$,

$$\sum_{k=1}^{n} \lambda^{n-k} b_{m-l,n-l}(k-l) f_{m,n}(k) = 0.$$

P13.7 Consider the a priori forward and backward prediction errors $f_{i,n-1}(k)$ and $b_{j,n-1}(k)$ and also the associated a posteriori errors $f_{i,n}(k)$ and $b_{j,n}(k)$, respectively, of a real-valued and prewindowed signal sequence $x(k)$. Prove the following results:

(i) $$\sum_{k=1}^{n} \lambda^{n-k} b_{m-1,n-1}(k) f_{m,n}(k) = 0.$$

(ii) $$\sum_{k=1}^{n} \lambda^{n-k} f_{m-1,n-1}(k) b_{m,n}(k) = 0.$$

P13.8 Consider a linear adaptive filter with tap-input vectors $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(n)$, and desired output sequence

$$d(k) = \begin{cases} 1, & k = n, \\ 0, & k = 1, 2, \ldots, n-1. \end{cases}$$

Find the least-squares error sum of this filter and show that it is equal to the conversion factor $\gamma(n)$ as given by (13.49) or (13.50). Then, prove that

$$0 \leq \gamma(n) \leq 1.$$

P13.9 Show that at any instant of time, n,

$$\gamma_{m+1}(n) \leq \gamma_m(n).$$

P13.10 Prove that

$$\frac{\zeta_m^{ff}(n)}{\zeta_{m-1}^{ff}(n)} = \frac{\zeta_m^{bb}(n)}{\zeta_{m-1}^{bb}(n-1)} = 1 - \kappa_m^{\mathrm{f}}(n)\kappa_m^{\mathrm{b}}(n).$$

P13.11 Use the result of Problem P13.10 to derive the following update equations for the least-squares sums $\zeta_m^{ff}(n)$ and $\zeta_m^{bb}(n)$:

$$\zeta_m^{ff}(n) = \zeta_{m-1}^{ff}(n) - \frac{(\zeta_{m-1}^{fb}(n))^2}{\zeta_{m-1}^{bb}(n-1)}$$

and

$$\zeta_m^{bb}(n) = \zeta_{m-1}^{bb}(n-1) - \frac{(\zeta_{m-1}^{fb}(n))^2}{\zeta_{m-1}^{ff}(n)}.$$

P13.12 Obtain the normal equation that results from the least-squares optimization of the a posteriori estimation error $e_{N,n}(k)$ of the joint process estimator of Figure 13.3 and show that this leads to the following set of independent equations:

$$c_m(n) = \frac{\sum_{k=1}^{n} \lambda^{n-k} b_{m,n}(k) d(k)}{\sum_{k=1}^{n} \lambda^{n-k} b_{m,n}^2(k)},$$

for $m = 0, 1, \ldots, N-1$. Then, use the orthogonality of the backward errors $b_{m,n}(k)$, for $m = 0, 1, \ldots, N$, to convert these equations to those given in (13.26).

P13.13 Give a detailed proof of (13.81).

P13.14 Derive the update equations of $\kappa_{m+1}^{\mathrm{b}}(n)$ and $c_m(n)$ that appeared in Table 13.2.

P13.15 Prove the following identity:

$$\boldsymbol{\Psi}_{m+1}^{-1}(n) = \begin{bmatrix} 0 & \mathbf{0}_m^{\mathrm{T}} \\ \mathbf{0}_m & \boldsymbol{\Psi}_m^{-1}(n-1) \end{bmatrix} + \frac{1}{\zeta_m^{ff}(n)} \tilde{\mathbf{a}}_m(n) \tilde{\mathbf{a}}_m^{\mathrm{T}}(n).$$

P13.16 Show that

$$\gamma_m(n) = \gamma_m(n-1) + \frac{b_{m,n}^2(n)}{\zeta_m^{bb}(n)} - \frac{f_{m,n}^2(n)}{\zeta_m^{ff}(n)}.$$

P13.17 Give a detailed derivation of (13.96).

P13.18 Give a detailed derivation of (13.100).

P13.19 Use the results of Problems P13.17 and P13.18 to obtain a time-update equation relating $\gamma_m(n)$ and $\gamma_m(n-1)$.

P13.20 Explore the possibility of rearranging the recursions/equations in Table 13.1 in terms of a priori estimation errors. Thus, suggest an alternative implementation RLSL algorithm using the a priori estimation errors.

14
Tracking

Our study of adaptive filters so far has been based on the assumption that the filter input and its desired output are jointly stationary processes. Under this condition, the correlation matrix, $\mathbf{R}$, of the filter input and the cross-correlation vector, $\mathbf{p}$, between the filter input and its desired output are fixed quantities. Consequently, the performance surface of the filter is also fixed, with its minimum point given by the Wiener–Hopf solution $\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p}$. A comparison of different algorithms would thus be based on their convergence behaviour. In this context, superior algorithms are those with shorter convergence times.

In this chapter we study another important aspect of adaptive filters. In many applications the underlying processes are non-stationary. As a result, the Wiener–Hopf solution, $\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p}$, varies with time, since $\mathbf{R}$ and $\mathbf{p}$ are time-varying. In such a situation the adaptive algorithm is expected to not only adapt the filter tap weights to a neighbourhood of their optimum values, but also to follow the variations of the optimum tap weights. The latter, which is the subject of this chapter, is known as *tracking*.

Before we start our study on tracking, we should remark that there is a clear distinction between convergence and tracking. *Convergence is a transient phenomenon.* It refers to the behaviour of a system (here, an adaptive filter) when it starts from an arbitrary initial condition and undergoes a transient period before it reaches its steady state. *Tracking,* on the other hand, *is a steady-state phenomenon.* It refers to the behaviour of a system in following variations in its surrounding environment, after it has reached its steady state. An algorithm with good convergence properties does not necessarily possess a fast tracking capability, and vice versa. Part of our effort in this chapter is to clarify this *seemingly* unusual behaviour of adaptive algorithms.

14.1 Formulation of the Tracking Problem

Much of the work related to the tracking behaviour of adaptive filters is done in the context of the modelling problem depicted in Figure 14.1. The plant is a linear multiple regressor characterized by the equation

$$d(n) = \mathbf{w}_o^T(n)\mathbf{x}(n) + e_o(n), \tag{14.1}$$

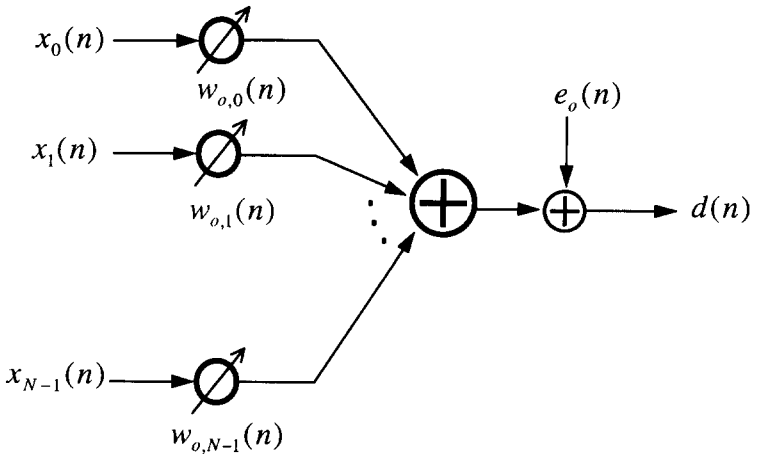


Figure 14.1 Linear multiple regressor

where $\mathbf{x}(n) = [x_0(n)\ x_1(n)\ \cdots\ x_{N-1}(n)]^T$ is the tap-input vector, $\mathbf{w}_o(n) = [w_{o,0}(n)\ w_{o,1}(n)\ \cdots\ w_{o,N-1}(n)]^T$ is the plant tap-weight vector, $e_o(n)$ is the plant noise, and $d(n)$ is the plant output. The presence of the time index n in $\mathbf{w}_o(n)$ is to emphasize that the plant tap-weight vector is time variant. This is unlike the notation $\mathbf{w}_o$ which was used in previous chapters to represent fixed plant weights. The role of the adaptive algorithm is to follow the variations in $\mathbf{w}_o(n)$.

The time-varying tap-weight vector $\mathbf{w}_o(n)$ is chosen to be a *multivariate random-walk process* characterized by the difference equation

$$\mathbf{w}_o(n+1) = \mathbf{w}_o(n) + \boldsymbol{\varepsilon}_o(n), \tag{14.2}$$

where $\boldsymbol{\varepsilon}_o(n)$ is the *process noise vector*.

The following assumptions are made throughout this chapter:

1. The sequences $e_o(n)$, $\boldsymbol{\varepsilon}_o(n)$ and $\mathbf{x}(n)$ are zero-mean and stationary random processes.
2. The sequences $e_o(n)$, $\boldsymbol{\varepsilon}_o(n)$ and $\mathbf{x}(n)$ are statistically independent of one another.
3. The successive increments, $\boldsymbol{\varepsilon}_o(n)$, of the plant tap weights are independent. However, the elements of $\boldsymbol{\varepsilon}_o(n)$, for a given n, may be statistically dependent.
4. At time n, the tap-weight vector $\mathbf{w}(n)$ of the adaptive filter is statistically independent of $e_o(n)$ and $\mathbf{x}(n)$.

The validity of the last assumption (which is known as the *independence assumption*) is justified only for small values of the step-size parameter(s) of the adaptation algorithm (see Chapter 6, Section 6.2). This is assumed to be true throughout our discussions in this chapter.

14.2 Generalized Formulation of the LMS Algorithm

In this section we present a generalized formulation of the LMS algorithm which can be used for a unified study of the tracking behaviour of various adaptive algorithms. The

LMS recursion that we consider is

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\boldsymbol{\mu}e(n)\mathbf{x}(n), \tag{14.3}$$

where $e(n) = d(n) - y(n)$ is the output error, $y(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n)$ is the filter output, $\mathbf{w}(n)$ and $\mathbf{x}(n)$ are the tap-weight and tap-input vectors, respectively, and $\boldsymbol{\mu}$ is a diagonal matrix consisting of the step-size parameters corresponding to various taps of the filter. These parameters, which are called μ_i, $i = 0, 1, \ldots, N-1$, are assumed fixed in our analysis. Furthermore, to keep (14.3) in its most general form, we follow the modelling problem of Figure 14.1 and choose $\mathbf{x}(n) = [x_0(n)\ x_1(n)\ \cdots\ x_{N-1}(n)]^{\mathrm{T}}$. This allows for the possibility that the tap inputs may not correspond to those from a tapped delay line.

The algorithms that are covered by (14.3) are:

The conventional LMS algorithm. By choosing $\boldsymbol{\mu} = \mu\mathbf{I}$, where μ is a scalar step-size parameter and $\mathbf{I}$ is the $N \times N$ identity matrix, (14.3) reduces to the conventional LMS recursion.

The TDLMS algorithm. The recursion (14.3) will be that of the TDLMS algorithm if $\mathbf{x}(n)$ is replaced by $\mathbf{x}_T(n) = \boldsymbol{\mathcal{T}}\mathbf{x}(n)$, where $\boldsymbol{\mathcal{T}}$ is a transformation matrix and $\mathbf{x}(n)$ is the filter tap-input vector before transformation. Moreover, we shall choose the normalized step-sizes as (see Chapter 7)

$$\mu_i = \frac{\mu'}{\mathrm{E}[x_{T,i}^2(n)]}, \quad \text{for } i = 0, 1, \ldots, N-1, \tag{14.4}$$

where μ' is a common scalar, $x_{T,i}(n)$ is the ith element of $\mathbf{x}_T(n)$, and $\mathrm{E}[\cdot]$ denotes statistical expectation. In actual implementation of the TDLMS algorithm, the values of $\mathrm{E}[x_{T,i}^2(n)]$ are estimated through time averaging. However, to simplify our discussion, we will assume that such averages are known *a priori*, thus in our study the μ_is are fixed.

The ideal LMS–Newton algorithm. From Chapter 7 we recall that the ideal LMS–Newton algorithm is equivalent to the TDLMS with $\boldsymbol{\mathcal{T}}$ replaced by the Karhunen–Loéve transform (KLT) of the input process. Thus, the analysis that we do for the TDLMS algorithm can be immediately applied to evaluate the tracking behaviour of

We note that

$$\begin{aligned} e(n) &= d(n) - \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n) \\ &= d(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{w}(n) \\ &= d(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{w}_{\mathrm{o}}(n) - \mathbf{x}^{\mathrm{T}}(n)[\mathbf{w}(n) - \mathbf{w}_{\mathrm{o}}(n)] \\ &= e_{\mathrm{o}}(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{v}(n) \end{aligned} \tag{14.5}$$

where $\mathbf{v}(n) = \mathbf{w}(n) - \mathbf{w}_{\mathrm{o}}(n)$ is the weight-error vector, and from (14.1), $e_{\mathrm{o}}(n) = d(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{w}_{\mathrm{o}}(n)$. Using (14.5) in (14.3) and using (14.2), we obtain

$$\mathbf{v}(n+1) = (\mathbf{I} - 2\mu\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n))\mathbf{v}(n) + 2\mu e_{\mathrm{o}}(n)\mathbf{x}(n) - \varepsilon_{\mathrm{o}}(n), \tag{14.6}$$

where $\mathbf{I}$ is the identity matrix. Next, we multiply both sides of (14.6) from the right by their respective transposes, take statistical expectation of the results and expanding to obtain

$$\begin{aligned} \mathbf{K}(n+1) = \mathbf{K}(n) &- 2\mu\mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)] - 2\mathrm{E}[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]\mu \\ &+ 4\mu\mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]\mu \\ &+ 2\mathrm{E}[(\mathbf{I} - 2\mu\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n))\mathbf{v}(n)\mathbf{x}^{\mathrm{T}}(n)e_{\mathrm{o}}(n)]\mu \\ &+ 2\mu\mathrm{E}[e_{\mathrm{o}}(n)\mathbf{x}(n)\mathbf{v}^{\mathrm{T}}(n)(\mathbf{I} - 2\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)\mu)] + 4\mu\mathrm{E}[|e_{\mathrm{o}}(n)|^2\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]\mu \\ &- \mathrm{E}[(\mathbf{I} - 2\mu\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n))\mathbf{v}(n)\varepsilon_{\mathrm{o}}^{\mathrm{T}}(n)] - \mathrm{E}[\varepsilon_{\mathrm{o}}(n)\mathbf{v}^{\mathrm{T}}(n)(\mathbf{I} - 2\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)\mu)] \\ &- 2\mu\mathrm{E}[e_{\mathrm{o}}(n)\mathbf{x}(n)\varepsilon_{\mathrm{o}}^{\mathrm{T}}(n)] - 2\mathrm{E}[e_{\mathrm{o}}(n)\varepsilon_{\mathrm{o}}(n)\mathbf{x}^{\mathrm{T}}(n)]\mu + \mathrm{E}[\varepsilon_{\mathrm{o}}(n)\varepsilon_{\mathrm{o}}^{\mathrm{T}}(n)], \end{aligned} \tag{14.7}$$

where $\mathbf{K}(n) = \mathrm{E}[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)]$. According to assumptions 1–4 of Section 14.1, $e_{\mathrm{o}}(n)$ is zero-mean and independent of $\mathbf{x}(n)$, $\mathbf{v}(n)$ and $\varepsilon_{\mathrm{o}}(n)$. The independence of $e_{\mathrm{o}}(n)$ and $\mathbf{v}(n) = \mathbf{w}(n) - \mathbf{w}_{\mathrm{o}}(n)$ follows from the fact that $e_{\mathrm{o}}(n)$ is independent of $\mathbf{w}(n)$ (assumption 4) and $\varepsilon_{\mathrm{o}}(n)$ (assumption 2). Consequently, the fifth, sixth, tenth and eleventh terms on the right-hand side of (14.7) become zero. Similarly, the eighth and ninth terms on the right-hand side of (14.7) are also zero since $\varepsilon_{\mathrm{o}}(n)$ is zero-mean and independent of $\mathbf{x}(n)$ and $\mathbf{v}(n)$. The independence of $\varepsilon_{\mathrm{o}}(n)$ and $\mathbf{v}(n)$ follows from the fact that $\mathbf{v}(n)$ is only affected by the past values of $\varepsilon_{\mathrm{o}}(n)$, and according to assumption 3, $\varepsilon_{\mathrm{o}}(n)$ is independent of its past observations. Furthermore, the independence of $\mathbf{x}(n)$ and $\mathbf{v}(n)$ implies that

$$\begin{aligned} \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)] &= \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]\mathrm{E}[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)] \\ &= \mathbf{R}\mathbf{K}(n) \end{aligned} \tag{14.8}$$

and

$$\begin{aligned} \mathrm{E}[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)] &= \mathrm{E}[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)]\mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)] \\ &= \mathbf{K}(n)\mathbf{R}, \end{aligned} \tag{14.9}$$

where $\mathbf{R} = \mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]$. Assumption 2 implies that

$$\mathrm{E}[|e_{\mathrm{o}}(n)|^2\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)] = \sigma_{e_{\mathrm{o}}}^2\mathbf{R}, \tag{14.10}$$

where $\sigma_{e_{\mathrm{o}}}^2 = \mathrm{E}[|e_{\mathrm{o}}(n)|^2]$ is the variance of the zero-mean random variable $e_{\mathrm{o}}(n)$. Finally, considering the independence of $\mathbf{v}(n)$ and $\mathbf{x}(n)$ and assuming that the elements of $\mathbf{x}(n)$ are Gaussian distributed and following a similar line of derivations as that which led to (6.39) (see Appendix 6A), we obtain

$$\mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)] = \mathbf{R}\mathrm{tr}[\mathbf{R}\mathbf{K}(n)] + 2\mathbf{R}\mathbf{K}(n)\mathbf{R}. \tag{14.11}$$

Using these results in (14.7) we obtain

$$\begin{aligned}\mathbf{K}(n+1) = \mathbf{K}(n) &- 2\boldsymbol{\mu}\mathbf{R}\mathbf{K}(n) - 2\mathbf{K}(n)\mathbf{R}\boldsymbol{\mu} + 4\boldsymbol{\mu}\mathbf{R}\boldsymbol{\mu}\mathrm{tr}[\mathbf{R}\mathbf{K}(n)] \\ &+ 8\boldsymbol{\mu}\mathbf{R}\mathbf{K}(n)\mathbf{R}\boldsymbol{\mu} + 4\sigma_{e_{\mathrm{o}}}^2\boldsymbol{\mu}\mathbf{R}\boldsymbol{\mu} + \mathbf{G},\end{aligned} \tag{14.12}$$

where

$$\mathbf{G} = \mathrm{E}[\boldsymbol{\varepsilon}_{\mathrm{o}}(n)\boldsymbol{\varepsilon}_{\mathrm{o}}^{\mathrm{T}}(n)] \tag{14.13}$$

is the correlation matrix of the plant tap-weight increments.

Next, we recall that

$$\xi_{\mathrm{ex}}(n) = \mathrm{E}\left[(\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n))^2\right], \tag{14.14}$$

where $\xi_{\mathrm{ex}}(n)$ is the excess MSE at time n. Using the independence of $\mathbf{v}(n)$ and $\mathbf{x}(n)$ and following the same line of derivations which led to (6.26), we obtain

$$\mathrm{E}\left[(\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n))^2\right] = \mathrm{tr}[\mathbf{R}\mathbf{K}(n)]. \tag{14.15}$$

Substituting this result in (14.14) we obtain

$$\xi_{\mathrm{ex}}(n) = \mathrm{tr}[\mathbf{R}\mathbf{K}(n)]. \tag{14.16}$$

Since all the underlying processes are assumed to be stationary (assumption 1 in Section 14.1), $\mathbf{K}(n)$ and $\xi_{\mathrm{ex}}(n)$ will be independent of n in the steady state. Hence, the time index n is dropped from $\mathbf{K}(n)$ and $\xi_{\mathrm{ex}}(n)$ henceforth.

Premultiplying (14.12) on both sides by $\frac{1}{2}\boldsymbol{\mu}^{-1}$, taking the trace, and assuming that the algorithm has reached its steady state so that $\mathbf{K}(n+1) = \mathbf{K}(n) = \mathbf{K}$, we obtain

$$\mathrm{tr}[\mathbf{R}\mathbf{K}] + \mathrm{tr}[\boldsymbol{\mu}^{-1}\mathbf{K}\mathbf{R}\boldsymbol{\mu}] = 2\mathrm{tr}[\mathbf{R}\boldsymbol{\mu}]\mathrm{tr}[\mathbf{R}\mathbf{K}] + 4\mathrm{tr}[\mathbf{R}\mathbf{K}\mathbf{R}\boldsymbol{\mu}] + 2\sigma_{e_{\mathrm{o}}}^2\mathrm{tr}[\mathbf{R}\boldsymbol{\mu}] + \tfrac{1}{2}\mathrm{tr}[\boldsymbol{\mu}^{-1}\mathbf{G}]. \tag{14.17}$$

Next, using the identity $\mathrm{tr}[\mathbf{A}\mathbf{B}] = \mathrm{tr}[\mathbf{B}\mathbf{A}]$, which is true for any pair of $M \times N$ and $N \times M$ matrices $\mathbf{A}$ and $\mathbf{B}$, we get $\mathrm{tr}[\mathbf{R}\boldsymbol{\mu}] = \mathrm{tr}[\boldsymbol{\mu}\mathbf{R}]$, $\mathrm{tr}[\mathbf{R}\mathbf{K}\mathbf{R}\boldsymbol{\mu}] = \mathrm{tr}[\boldsymbol{\mu}\mathbf{R}\mathbf{K}\mathbf{R}]$, and

$$\mathrm{tr}[\boldsymbol{\mu}^{-1}\mathbf{K}\mathbf{R}\boldsymbol{\mu}] = \mathrm{tr}[\mathbf{R}\boldsymbol{\mu}\boldsymbol{\mu}^{-1}\mathbf{K}] = \mathrm{tr}[\mathbf{R}\mathbf{K}].$$

Using these and (14.16) in (14.17) we obtain

$$2\xi_{ex} = 2\text{tr}[\boldsymbol{\mu}\mathbf{R}]\xi_{ex} + 4\text{tr}[\boldsymbol{\mu}\mathbf{R}\mathbf{K}\mathbf{R}] + 2\sigma_{e_o}^2\,\text{tr}[\boldsymbol{\mu}\mathbf{R}] + \tfrac{1}{2}\text{tr}[\boldsymbol{\mu}^{-1}\mathbf{G}] \qquad (14.18)$$

(14.18) can be ignored. Numerical examples and computer simulations shows that when N (the filter length) is large, $\text{tr}[\boldsymbol{\mu}\mathbf{R}\mathbf{K}\mathbf{R}]$ is usually at least an order of magnitude smaller than $\text{tr}[\boldsymbol{\mu}\mathbf{R}]\xi_{ex}$. See Problem P14.1 for more exposure over this approximation. This leads to the following result:

$$\xi_{ex} = \frac{1}{1 - \text{tr}[\boldsymbol{\mu}\mathbf{R}]}(\sigma_{e_o}^2\,\text{tr}[\boldsymbol{\mu}\mathbf{R}] + \tfrac{1}{4}\text{tr}[\boldsymbol{\mu}^{-1}\mathbf{G}]). \qquad (14.19)$$

Using (14.19) to evaluate the misadjustment of the generalized LMS algorithm, we obtain

$$\mathcal{M} = \frac{\xi_{ex}}{\xi_{min}} = \frac{1}{1 - \text{tr}[\boldsymbol{\mu}\mathbf{R}]}(\text{tr}[\boldsymbol{\mu}\mathbf{R}] + \tfrac{1}{4}\sigma_{e_o}^{-2}\text{tr}[\boldsymbol{\mu}^{-1}\mathbf{G}]), \qquad (14.20)$$

where $\xi_{min} = \sigma_{e_o}^2$ is the minimum MSE of the filter that is obtained when $\mathbf{w}(n) = \mathbf{w}_o(n)$.

To relate this result to the results of the previous chapters, let us consider the case of

and

$$\mathcal{M}_2 = \frac{\sigma_{e_o}^{-2}}{4} \cdot \frac{\text{tr}[\boldsymbol{\mu}^{-1}\mathbf{G}]}{1 - \text{tr}[\boldsymbol{\mu}\mathbf{R}]}. \tag{14.25}$$

With reference to recursion (14.6) and the subsequent derivations, we find that $\mathcal{M}_1$ originates from the term $2\boldsymbol{\mu}e_o(n)\mathbf{x}(n)$ on the right-hand side of (14.6). This, clearly, is contributed by the plant noise, $e_o(n)$. Similarly, we find that $\mathcal{M}_2$ is a direct contribution of the plant tap-weight increments $\boldsymbol{\varepsilon}_o(n)$. Accordingly, $\mathcal{M}_1$ is called the *noise misadjustment* and $\mathcal{M}_2$ is referred to as the *lag misadjustment*. We note that the noise misadjustment decreases with a decrease in the step-size parameters, the μ_is. On the other hand, a smaller lag misadjustment is achieved by increasing the step-size parameters. Thus, it becomes necessary to find a compromise choice of the step-size parameters which will result in the right balance between the noise and lag misadjustments. This is the subject of the next section.

14.4 Optimum Step-Size Parameters

To derive a set of equations for the optimum step-size parameters that minimizes the excess MSE and thus the misadjustment of the LMS algorithm, we first expand (14.19) to obtain

$$\xi_{ex} = \frac{1}{1 - \sum_i \mu_i \sigma_{x_i}^2} \sum_{i=0}^{N-1} \left(\mu_i \sigma_{e_o}^2 \sigma_{x_i}^2 + \frac{1}{4\mu_i} \sigma_{\varepsilon_{o,i}}^2 \right) \tag{14.26}$$

where $\sigma_{x_i}^2$ and $\sigma_{\varepsilon_{o,i}}^2$ are, respectively, the variances of $x_i(n)$ and the ith element of $\boldsymbol{\varepsilon}_o(n)$, i.e. the diagonal elements of the respective correlation matrices, $\mathbf{R}$ and $\mathbf{G}$.

The optimum values of the step-size parameters are obtained by setting the derivatives of ξ_{ex} with respect to μ_is equal to zero. Solving the set of simultaneous equations

$$\frac{\partial \xi_{ex}}{\partial \mu_i} = 0, \quad \text{for } i = 0, 1, \ldots, N-1, \tag{14.27}$$

we obtain (see Problem P14.4)

$$\mu_{o,i} = \frac{\sigma_{\varepsilon_{o,i}}}{2\sigma_{x_i}\sqrt{\xi_{ex,o} + \sigma_{e_o}^2}}, \qquad i = 0, 1, \ldots, N-1, \tag{14.28}$$

where the subscript 'o' is added to the $\mu_{o,i}$s to emphasize that they are the optimum values of the step-size parameters. Moreover, $\xi_{ex,o}$ refers to the excess MSE when the optimum step-size parameters, the $\mu_{o,i}$s, are used. This solution, of course, is not complete, since $\xi_{ex,o}$ depends on the $\mu_{o,i}$s. To complete the solution, we define $\eta = \sqrt{\xi_{ex,o} + \sigma_{e_o}^2}$ and replace (14.28) in (14.26). This results in a second-order equation in η whose solutions are

$$\eta = \frac{\sum_i \sigma_{\varepsilon_{o,i}} \sigma_{x_i} \pm \sqrt{\left(\sum_i \sigma_{\varepsilon_{o,i}} \sigma_{x_i}\right)^2 + 4\sigma_{e_o}^2}}{2}.$$

Noting that η cannot be negative, we find that

$$\eta = \frac{\sum_i \sigma_{\varepsilon_{o,i}} \sigma_{x_i} + \sqrt{(\sum_i \sigma_{\varepsilon_{o,i}} \sigma_{x_i})^2 + 4\sigma_{e_o}^2}}{2} \tag{14.29}$$

is the only acceptable solution of η. With this, we get

$$\mu_{o,i} = \frac{\sigma_{\varepsilon_{o,i}}}{2\eta\sigma_{x_i}}, \quad \text{for } i = 0, 1, \ldots, N-1. \tag{14.30}$$

It is instructive to note that (14.30) is intuitively sound. It suggests that those taps that have a larger tap perturbation should be given larger step-size parameters. It also suggests normalization of the step-size parameters proportional to the inverse of the signal level at various taps. However, this normalization is different from the one commonly used in the step-normalized algorithms, where μ_i is selected proportional to the inverse of the signal power at the respective tap, i.e. proportional to $1/\sigma_{x_i}^2$. Moreover, (14.30) suggests that the step-size parameters should be reduced as the error level at the filter output increases – note that η^2 is equal to the MSE of the filter after it has converged.

The validity of (14.30) is subject to the condition that the optimum step-size parameters remain in a range that does not result in instablility of the algorithm. For the case of the conventional LMS algorithm, where a single step-size parameter, μ, is employed, a useful and practically applicable upper bound for μ is the one derived in Chapter 6 and repeated below for convenience (see (6.73)):

$$\mu < \frac{1}{3\text{tr}[\mathbf{R}]} \tag{14.31}$$

or, equivalently,

$$\mu\text{tr}[\mathbf{R}] < \tfrac{1}{3}. \tag{14.32}$$

This result can be extended to the generalized LMS recursion (14.3) as follows.

Consider the recursion (14.3) and define $\tilde{\mathbf{x}}(n) = \boldsymbol{\mu}^{1/2}\mathbf{x}(n)$, where $\boldsymbol{\mu}^{1/2}$ is the diagonal matrix consisting of the square roots of the diagonal elements of $\boldsymbol{\mu}$. Then, multiplying both sides of (14.6) from the left by $\boldsymbol{\mu}^{-1/2}$ and, also, defining $\tilde{\mathbf{v}}(n) = \boldsymbol{\mu}^{-1/2}\mathbf{v}(n)$ and $\tilde{\boldsymbol{\varepsilon}}_o(n) = \boldsymbol{\mu}^{-1/2}\boldsymbol{\varepsilon}_o(n)$ we obtain

$$\tilde{\mathbf{v}}(n+1) = (\mathbf{I} - 2\tilde{\mathbf{x}}(n)\tilde{\mathbf{x}}^{\mathrm{T}}(n))\tilde{\mathbf{v}}(n) + 2e_o(n)\tilde{\mathbf{x}}(n) - \tilde{\boldsymbol{\varepsilon}}_o(n). \tag{14.33}$$

The recursion (14.33) is similar to the conventional LMS recursion with $\mu = 1$. Accordingly, (14.32) can be applied. Hence, we find that the stability of (14.33), and thus (14.6) or, equivalently, the recursion (14.3), is guaranteed if

$$\text{tr}[\tilde{\mathbf{R}}] < \tfrac{1}{3}, \tag{14.34}$$

where

$$\begin{aligned}\tilde{\mathbf{R}} &= \mathrm{E}[\tilde{\mathbf{x}}(n)\tilde{\mathbf{x}}^{\mathrm{T}}(n)] \\ &= \mathrm{E}[\boldsymbol{\mu}^{1/2}\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\boldsymbol{\mu}^{1/2}] \\ &= \boldsymbol{\mu}^{1/2}\mathrm{E}[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]\boldsymbol{\mu}^{1/2} \\ &= \boldsymbol{\mu}^{1/2}\mathbf{R}\boldsymbol{\mu}^{1/2}. \end{aligned} \tag{14.35}$$

Substituting this result in (14.34) and noting that $\mathrm{tr}[\boldsymbol{\mu}^{1/2}\mathbf{R}\boldsymbol{\mu}^{1/2}] = \mathrm{tr}[\boldsymbol{\mu}\mathbf{R}]$ (according to the identity $\mathrm{tr}[\mathbf{AB}] = \mathrm{tr}[\mathbf{BA}]$), we get

$$\mathrm{tr}[\boldsymbol{\mu}\mathbf{R}] < \tfrac{1}{3}. \tag{14.36}$$

This is a sufficient condition which may be imposed on the algorithm step-size parameters, the μ_is, to guarantee the stability of the generalized LMS recursion (14.3).

When (14.36) holds, the minimum excess MSE of the filter, $\xi_{\mathrm{ex,o}}$, is obtained by substituting (14.30) in (14.26). This gives

$$\xi_{\mathrm{ex,o}} = \frac{1}{2 - \dfrac{1}{\eta}\displaystyle\sum_i \sigma_{\varepsilon_{\mathrm{o},i}}\sigma_{x_i}} \left(\frac{\sigma_{e_\mathrm{o}}^2}{\eta} + \eta\right) \sum_i \sigma_{\varepsilon_{\mathrm{o},i}}\sigma_{x_i}. \tag{14.37}$$

Substituting for η from (14.29), we get

$$\xi_{\mathrm{ex,o}} = \frac{\sum_i \sigma_{\varepsilon_{\mathrm{o},i}}\sigma_{x_i} + \sqrt{(\sum_i \sigma_{\varepsilon_{\mathrm{o},i}}\sigma_{x_i})^2 + 4\sigma_{e_\mathrm{o}}^2}}{2} \sum_i \sigma_{\varepsilon_{\mathrm{o},i}}\sigma_{x_i}. \tag{14.38}$$

14.5 Comparisons of Conventional Algorithms

In this section we compare the tracking behaviour of various versions of the LMS algorithm in the context of the modelling problem discussed in the previous few sections. Noting that the tracking behaviours of the RLS and LMS–Newton algorithms are about the same, the comparisons also cover the RLS algorithm. *The indicator of better tracking behaviour (performance) is a lower steady-state excess MSE.*

To prevent divergence into many possible cases, we concentrate on a comparison of the direct implementation of a transversal filter, using the LMS algorithm and its implementation in the transform domain. We note that for a transversal filter $\mathbf{x}(n) = [x(n)\ \ x(n-1)\ \ \ldots\ \ x(n-N+1)]^{\mathrm{T}}$, and for its transform-domain implementation, $\mathbf{x}(n)$ is replaced by $\mathbf{x}_T(n) = \mathcal{T}\mathbf{x}(n)$, where $\mathcal{T}$ is an orthonormal transformation matrix satisfying the condition[1]

$$\mathcal{T}\mathcal{T}^{\mathrm{T}} = \mathbf{I}, \tag{14.39}$$

where $\mathbf{I}$ is the identity matrix.

[1] To avoid complex-valued coefficients/variables in our formulations in this chapter, we only consider transformations with real-valued coefficients.

Also, if $\varepsilon_o(n)$ represents the plant tap-weight increments in its transversal form, the corresponding increments in the transform domain are given by

$$\varepsilon_{T,o}(n) = \mathcal{T}\varepsilon_o(n). \tag{14.40}$$

We also define $\mathbf{R}_T = \mathrm{E}[\mathbf{x}_T(n)\mathbf{x}_T^{\mathrm{T}}(n)]$ and $\mathbf{G}_T = \mathrm{E}[\varepsilon_{T,o}(n)\varepsilon_{T,o}^{\mathrm{T}}(n)]$, and note that

$$\mathbf{R}_T = \mathcal{T}\mathbf{R}\mathcal{T}^{\mathrm{T}} \tag{14.41}$$

and

$$\mathbf{G}_T = \mathcal{T}\mathbf{G}\mathcal{T}^{\mathrm{T}}. \tag{14.42}$$

The ith diagonal elements of $\mathbf{R}_T$ and $\mathbf{G}_T$ are denoted as $\sigma^2_{x_{T,i}}$ and $\sigma^2_{\varepsilon_{T,o,i}}$, respectively. Moreover, to simplify our discussion, yet with no loss of generality, we assume that the input sequence to the transversal filter is normalized to unit power, i.e. $\sigma^2_{x_i} = \mathrm{E}[|x(n-i)|^2] = 1$, for $i = 0, 1, \ldots, N-1$. Then, the orthonormality of $\mathcal{T}$, i.e. the condition (14.39), implies that

$$\sum_{i=0}^{N-1} \sigma^2_{x_{T,i}} = \sum_{i=0}^{N-1} \sigma^2_{x_i} = N. \tag{14.43}$$

We note that in the case of the conventional LMS algorithm, a single step-size parameter, μ, is used for all taps. On the other hand, in the case of TDLMS algorithm, different step-size parameters are used for various taps and they are selected according to (14.4). Furthermore, for a fixed misadjustment, say $\mathcal{M}$, we have (see (6.63) and (7.31))

$$\mu = \frac{\mathcal{M}}{\mathrm{tr}[\mathbf{R}]} = \frac{\mathcal{M}}{\sum_i \sigma^2_{x_i}} \tag{14.44}$$

and

$$\mu' = \frac{\mathcal{M}}{N}. \tag{14.45}$$

Thus, in the light of (14.43), we find that $\mu' = \mu$ in the present case.

Using the above results in (14.26), the excess MSE of the conventional LMS and TDLMS algorithms are obtained as

$$\xi_{\mathrm{ex}}(\mathrm{LMS}) = \frac{1}{1-\mu N}\left(\mu N \sigma^2_{e_o} + \frac{1}{4\mu}\sum_{i=0}^{N-1} \sigma^2_{\varepsilon_{o,i}}\right) \tag{14.46}$$

and

$$\xi_{\mathrm{ex}}(\mathrm{TDLMS}) = \frac{1}{1-\mu N}\left(\mu N \sigma^2_{e_o} + \frac{1}{4\mu}\sum_{i=0}^{N-1} \sigma^2_{\varepsilon_{T,o,i}}\sigma^2_{x_{T,i}}\right), \tag{14.47}$$

respectively. In arriving at (14.47) and (14.46), we made use of the assumption $\sigma_{x_i}^2 = 1$, for $i = 0, 1, \ldots, N-1$, along with (14.43) and (14.4).

Now, let us consider a few specific cases.

Case 1: $\mathbf{G} = \sigma_{\varepsilon_o}^2 \mathbf{I}$ and $\mathbf{R}$ is arbitrary Substituting $\mathbf{G} = \sigma_{\varepsilon_o}^2 \mathbf{I}$ in (14.42) and recalling (14.39), we get $\mathbf{G}_T = \sigma_{\varepsilon_o}^2 \mathbf{I}$, which implies that

$$\sigma_{\varepsilon_{T,o,i}}^2 = \sigma_{\varepsilon_{o,i}}^2 = \sigma_{\varepsilon_o}^2, \quad \text{for } i = 0, 1, \ldots, N-1. \tag{14.48}$$

Substituting (14.48) and (14.43) in (14.47), we obtain

$$\begin{aligned} \xi_{\text{ex}}(\text{TDLMS}) &= \frac{1}{1-\mu N}\left(\mu N \sigma_{e_o}^2 + \frac{1}{4\mu}\sigma_{\varepsilon_o}^2 \sum_{i=0}^{N-1} \sigma_{x_{T,i}}^2\right) \\ &= \frac{N(\mu\sigma_{e_o}^2 + \frac{1}{4\mu}\sigma_{\varepsilon_o}^2)}{1-\mu N}. \end{aligned} \tag{14.49}$$

From this result we see that when $\mathbf{G} = \sigma_{\varepsilon_o}^2 \mathbf{I}$, $\xi_{\text{ex}}(\text{TDLMS})$ is independent of $\mathcal{T}$. Furthermore, with $\mathbf{G} = \sigma_{\varepsilon_o}^2 \mathbf{I}$, (14.46) also simplifies to (14.49). This, in turn, means that independent of the transformation used, the tracking performance of the TDLMS algorithm remains similar to that of the conventional LMS algorithm. Furthermore, noting that the LMS–Newton algorithm is equivalent to the TDLMS algorithm when KLT is used as its transformation, this conclusion also applies to the comparison of the conventional LMS and LMS–Newton algorithms. Moreover, noting that the RLS and LMS–Newton algorithms have similar tracking behaviour (see Section 14.2), we may also add that in the present case the conventional LMS and RLS algorithms have similar tracking behaviour.

Case 2: $\mathbf{R} = \mathbf{I}$ and $\mathbf{G}$ is arbitrary Using $\mathbf{R} = \mathbf{I}$ in (14.41), we find that $\mathbf{R}_T$ is also equal to the identity matrix. Thus,

$$\sigma_{x_{T,i}}^2 = 1, \quad \text{for } i = 0, 1, \ldots, N-1.$$

Using this in (14.47) we obtain

$$\xi_{\text{ex}}(\text{TDLMS}) = \frac{1}{1-\mu N}\left(\mu N \sigma_{e_o}^2 + \frac{1}{4\mu}\sum_{i=0}^{N-1} \sigma_{\varepsilon_{T,o,i}}^2\right). \tag{14.50}$$

Now,

$$\begin{aligned} \sum_{i=0}^{N-1} \sigma_{\varepsilon_{T,o,i}}^2 &= \text{tr}[\mathbf{G}_T] = \text{tr}[\mathcal{T}\mathbf{G}\mathcal{T}^{\text{T}}] = \text{tr}[\mathcal{T}^{\text{T}}\mathcal{T}\mathbf{G}] \\ &= \text{tr}[\mathbf{G}] = \sum_{i=0}^{N-1} \sigma_{\varepsilon_{o,i}}^2, \end{aligned} \tag{14.51}$$

where we have used (14.39) and (14.42), and the identity tr[**AB**] = tr[**BA**]. Substituting (14.51) in (14.50), we obtain

$$\xi_{ex}(\text{TDLMS}) = \frac{1}{1-\mu N}\left(\mu N \sigma_{e_o}^2 + \frac{1}{4\mu}\sum_{i=0}^{N-1}\sigma_{\varepsilon_{o,i}}^2\right). \tag{14.52}$$

Comparing this with (14.46) we find that in the present case also, irrespective of the transformation $\mathcal{T}$, there is no difference between the tracking behaviours of the conventional LMS and TDLMS algorithms. Thus, all the conclusions drawn for Case 1 continue to hold for Case 2 also, i.e. *the conventional LMS, TDLMS, LMS–Newton and RLS algorithms all have similar tracking behaviour.*

Case 3: R and G are arbitrary From (14.47) we note that to study the variation in the excess MSE of the TDLMS algorithm for different choices of $\mathcal{T}$, we need to study the summation

$$\sum_{i=0}^{N-1}\sigma_{\varepsilon_{T,o,i}}^2\sigma_{x_{T,i}}^2. \tag{14.53}$$

Moreover, we note that the orthonormality of $\mathcal{T}$, i.e. the identity $\mathcal{T}\mathcal{T}^T = \mathbf{I}$, implies that the summations $\sum_i \sigma_{x_{T,i}}^2$ and $\sum_i \sigma_{\varepsilon_{T,o,i}}^2$ are independent of $\mathcal{T}$. However, the individual terms under the summations, i.e. the $\sigma_{x_{T,i}}^2$s and $\sigma_{\varepsilon_{T,o,i}}^2$s, vary with $\mathcal{T}$. Thus, while the summations $\sum_i \sigma_{x_{T,i}}^2$ and $\sum_i \sigma_{\varepsilon_{T,o,i}}^2$ are fixed for different choices of $\mathcal{T}$, the distributions of the terms $\sigma_{x_{T,i}}^2$ and $\sigma_{\varepsilon_{T,o,i}}^2$ vary with $\mathcal{T}$. These distributions also depend on the correlation matrices **R** and **G**. When **R** and **G** are arbitrary, these distribution are also arbitrary. As a result, we find that *when no prior information about* **R** *and* **G** *is available, nothing can be said about the summation* (14.53), *and thus no specific comment can be made about the tracking behaviour of various algorithms.* The following numerical example clarifies this further. Let

$$\mathbf{R} = \begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 1.0 \end{bmatrix} \quad \text{and} \quad \mathbf{G} = \begin{bmatrix} 0.0010 & 0.0008 \\ 0.0008 & 0.0100 \end{bmatrix}.$$

Also, define

$$\mathcal{T} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

This is an arbitrary 2×2 orthonormal transformation matrix which varies with θ. Table 14.1 summarizes the results that we have obtained for the LMS and TDLMS algorithms for two choices of $\theta = \pi/8$ and $\pi/4$. It is noted that in the case of $\theta = \pi/8$, the TDLMS shows a better tracking behaviour than the LMS algorithm – compare the summations in the last line of Table 14.1. However, the LMS algorithm behaves better when $\theta = \pi/4$ is chosen. Incidentally, here $\theta = \pi/4$ makes $\mathcal{T}$ correspond to the KLT of the filter input, for which the TDLMS algorithm is also equivalent to the LMS–Newton algorithm.

Table 14.1 Comparison of the conventional LMS and TDLMS for a numerical example

	TDLMS	
LMS	$\theta = \pi/8$	$\theta = \pi/4$
$\sigma^2_{x_0} = 1.0000$	$\sigma^2_{x_{T,0}} = 1.3536$	1.5000
$\sigma^2_{x_1} = 1.0000$	$\sigma^2_{x_{T,1}} = 0.6464$	0.5000
$\sigma^2_{\varepsilon_0} = 0.0010$	$\sigma^2_{\varepsilon_{T,o,0}} = 0.0029$	0.0063
$\sigma^2_{\varepsilon_1} = 0.0100$	$\sigma^2_{\varepsilon_{T,o,1}} = 0.0081$	0.0047
$\sum_i \sigma^2_{x_i}\sigma^2_{\varepsilon_{0,i}} = 0.0110$	$\sum_i \sigma^2_{x_{T,i}}\sigma^2_{\varepsilon_{T,o,i}} = 0.0091$	0.0118

The comparisons given above assume that we have no information about the correlation matrix $\mathbf{G}$ of the plant tap-weight increments. Thus, the optimum step-size parameters derived in the previous section (see (14.30)) could not be used. In Section 14.7 we show that the optimum step-size parameters can, in fact, be obtained adaptively using the variable step-size LMS (VSLMS) algorithm introduced in Chapter 6 (Section 6.7). Noting this, we consider using the optimum step-size parameters given by (14.30) and present some more comparisons of the various algorithms in the next section.

14.6 Comparisons Based on the Optimum Step-Size Parameters

From the theoretical results of Section 14.4 and the definitions of $\mathbf{R}_T$ and $\mathbf{G}_T$ in the previous section, we recall that when the optimum step-size parameters given by (14.30) are used, the excess MSE of the TDLMS algorithm is given by (see (14.38))

$$\xi_{\text{ex,o}}(\text{TDLMS}) = \frac{\Gamma_T + \sqrt{(\Gamma_T)^2 + 4\sigma^2_{e_o}}}{2}\Gamma_T, \tag{14.54}$$

where

$$\Gamma_T = \sum_{i=0}^{N-1} \sigma_{\varepsilon_{T,o,i}}\sigma_{x_{T,i}}. \tag{14.55}$$

We note that Γ_T is a function of $\mathbf{R}$, $\mathbf{G}$ and $\mathcal{T}$.

We also note that when no transformation has been applied, but the optimum step-size parameters are used for different taps, the excess MSE of the LMS algorithm is given by

$$\xi_{\text{ex,o}}(\text{LMS}) = \frac{\Gamma_I + \sqrt{(\Gamma_I)^2 + 4\sigma^2_{e_o}}}{2}\Gamma_I, \tag{14.56}$$

where

$$\Gamma_I = \sum_{i=0}^{N-1} \sigma_{\varepsilon_{0,i}} \sigma_{x_i}. \tag{14.57}$$

Clearly, to achieve the best tracking performance of the TDLMS algorithm we should find the matrix $\mathcal{T}$ that minimizes $\Gamma_{\mathcal{T}}$. A general solution to this problem appears to be difficult. We thus limit ourselves to a few particular cases whose study is found to be instructive. The following lemma will be widely used in the study of the cases that follows.

Lemma *Consider the diagonal matrix* $\mathbf{\Lambda} = \text{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{N-1})$, *where the* λ_i*s are all real and non-negative. If* $\mathcal{T}$ *is an orthonormal matrix, i.e.* $\mathcal{T}\mathcal{T}^T = \mathbf{I}$, *and* $\mathbf{S} = \mathcal{T}\mathbf{\Lambda}\mathcal{T}^T$, *then the following inequality always holds:*

$$\sum_{i=0}^{N-1} \sqrt{\lambda_i} \leq \sum_{i=0}^{N-1} \sqrt{s_{ii}}, \tag{14.58}$$

where s_{ii} *is the ith diagonal element of* $\mathbf{S}$.

Proof We first note that for $x \geq 0, f(x) = \sqrt{x}$ is a concave function. Also, according to the theory of the convex functions (Rockafellar, 1970), if $f(x)$ is a concave function and $\zeta_0, \zeta_1, \ldots, \zeta_{N-1}$ are a set of non-negative numbers that satisfy $\sum_{i=0}^{N-1} \zeta_i = 1$, then for any set of numbers $x_0, x_1, \ldots, x_{N-1}$ in the domain of $f(x)$, the following inequality holds:

$$\sum_{i=0}^{N-1} \zeta_i f(x_i) \leq f\left(\sum_{i=0}^{N-1} \zeta_i x_i\right). \tag{14.59}$$

Next, we notice that

$$s_{ii} = \sum_{l=0}^{N-1} \lambda_l \tau_{il}^2, \tag{14.60}$$

where τ_{il} is the ilth element of $\mathcal{T}$. Also, the orthonormality of $\mathcal{T}$ implies that

$$\sum_{i=0}^{N-1} \tau_{il}^2 = 1. \tag{14.61}$$

Choosing $\zeta_i = \tau_{il}^2$, $x_i = \lambda_i$, and $f(x) = \sqrt{x}$ in (14.59), and using (14.60), we obtain

$$\sum_{l=0}^{N-1} |\tau_{il}|^2 \sqrt{\lambda_l} \leq \sqrt{s_{ii}}. \tag{14.62}$$

Summing up both sides of (14.62) over $i = 0, 1, \ldots, N-1$ and using (14.61) completes the proof.

We are now ready to consider a few specific cases:

Case 1: $\mathbf{R} = \mathbf{I}$ and $\mathbf{G}$ is an arbitrary diagonal matrix The assumption $\mathbf{R} = \mathbf{I}$ and the orthonormality of $\mathcal{T}$ implies that $\mathbf{R}_{\mathcal{T}} = \mathbf{I}$. Thus,

$$\sigma_{x_i}^2 = \sigma_{x_{T,i}}^2 = 1, \quad \text{for } i = 0, 1, \ldots, N-1. \tag{14.63}$$

Using (14.63) in (14.57) and (14.55), we get

$$\Gamma_{\mathrm{I}} = \sum_{i=0}^{N-1} \sigma_{\varepsilon_{o,i}} \tag{14.64}$$

and

$$\Gamma_{\mathcal{T}} = \sum_{i=0}^{N-1} \sigma_{\varepsilon_{T,o,i}}, \tag{14.65}$$

respectively. On the other hand, noting that $\mathbf{G}$ is a diagonal matrix consisting of the elements $\sigma_{\varepsilon_{o,0}}^2$, $\sigma_{\varepsilon_{o,1}}^2, \ldots, \sigma_{\varepsilon_{o,N-1}}^2$, the diagonal elements of $\mathbf{G}_{\mathcal{T}} = \mathcal{T}\mathbf{G}\mathcal{T}^{\mathrm{T}}$ are $\sigma_{\varepsilon_{T,o,0}}^2$, $\sigma_{\varepsilon_{T,o,1}}^2, \ldots, \sigma_{\varepsilon_{T,o,N-1}}^2$, and using the above lemma, we find that

$$\Gamma_{\mathrm{I}} \leq \Gamma_{\mathcal{T}}. \tag{14.66}$$

Using (14.66) in (14.54) and (14.56), we find that in the present case

$$\xi_{\mathrm{ex,o}}(\mathrm{LMS}) \leq \xi_{\mathrm{ex,o}}(\mathrm{TDLMS}). \tag{14.67}$$

That is, *when $\mathbf{R} = \mathbf{I}$ and $\mathbf{G}$ is diagonal, and the optimum step-size parameters, the $\mu_{o,i}$s, are used, there is no transformation that can improve the tracking behaviour of the LMS algorithm.*

Case 2: $\mathbf{R} = \mathbf{I}$ and $\mathbf{G}$ is arbitrary Let $\mathcal{T} = \mathcal{T}_{\mathrm{o}}$ be the orthonormal transform that results in a diagonal matrix $\mathbf{G}_{\mathcal{T}_{\mathrm{o}}} = \mathcal{T}_{\mathrm{o}}\mathbf{G}\mathcal{T}_{\mathrm{o}}^{\mathrm{T}}$. Using $\mathcal{T} = \mathcal{T}_{\mathrm{o}}$ leads to a TDLMS algorithm in which $\mathbf{R}_{\mathcal{T}_{\mathrm{o}}} = \mathcal{T}_{\mathrm{o}}\mathbf{R}\mathcal{T}_{\mathrm{o}}^{\mathrm{T}} = \mathbf{I}$ (since $\mathbf{R} = \mathbf{I}$), and $\mathbf{G}_{\mathcal{T}_{\mathrm{o}}}$ is diagonal. This is similar to Case 1 above. Hence, for the same reason as in Case 1, we can argue that *the choice of $\mathcal{T} = \mathcal{T}_o$ results in a TDLMS algorithm with optimum tracking behaviour.*

Case 3: $\mathbf{G} = \sigma_{\varepsilon_o}^2 \mathbf{I}$ and $\mathbf{R}$ is arbitrary Following the same line of reasoning as in Case 2, we find that here the optimum transform, $\mathcal{T}_{\mathrm{o}}$, which result in a TDLMS algorithm with the best tracking behaviour is the one that results in a diagonal $\mathbf{R}_{\mathcal{T}_{\mathrm{o}}} = \mathcal{T}_{\mathrm{o}}\mathbf{R}\mathcal{T}_{\mathrm{o}}^{\mathrm{T}}$. That is, here *the optimum transform for achieving best tracking is the KLT.*

14.7 VSLMS: An Algorithm with Optimum Tracking Behaviour

The variable step-size LMS (VSLMS) algorithm was introduced in Chapter 6, based on an intuitive understanding of the behaviour of the LMS algorithm. In this section we

present a formal derivation[2] of the VSLMS algorithm as an adaptive filtering scheme with optimal tracking behaviour.

14.7.1 Derivation of the VSLMS algorithm

From the results presented in Section 14.3 we observe that in a time-varying environment the steady-state MSE of an adaptive filter varies with the step-size parameters. Moreover, a study of the excess MSE, ξ_{ex}, shows that it is a convex function of the step-size parameters, the μ_is, when these vary over a range that does not result in instability (see equation (14.26)). This implies that the MSE

$$\xi = \mathrm{E}[e^2(n)] = \xi_{\min} + \xi_{ex}$$

is also a convex function of the step-size parameters. With this concept in mind, we may suggest the following gradient search method for finding the optimum step-size parameters, the $\mu_{o,i}$s, of the LMS algorithm:

$$\mu_i(n) = \mu_i(n-1) - \rho \frac{\partial \xi}{\partial \mu_i(n-1)}, \tag{14.68}$$

where ρ is a small positive adaptation parameter.

In analogy with the LMS algorithm, the stochastic version of the gradient recursion (14.68) is

$$\mu_i(n) = \mu_i(n-1) - \rho \frac{\partial e^2(n)}{\partial \mu_i(n-1)}. \tag{14.69}$$

Recalling that

$$\begin{aligned} e(n) &= d(n) - \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n) \\ &= d(n) - \sum_{l=0}^{N-1} w_l(n)x_l(n), \end{aligned} \tag{14.70}$$

we obtain

$$\begin{aligned} \frac{\partial e^2(n)}{\partial \mu_i(n-1)} &= 2e(n)\frac{\partial e(n)}{\partial \mu_i(n-1)} \\ &= -2e(n)x_i(n)\frac{\partial w_i(n)}{\partial \mu_i(n-1)}, \end{aligned} \tag{14.71}$$

where we have noted that in the summation $\sum_l w_l(n)x_l(n)$ only $w_i(n)$ varies with $\mu_i(n-1)$. The tap weight $w_i(n)$ is related to $\mu_i(n-1)$ according to the LMS recursion

$$w_i(n) = w_i(n-1) + 2\mu_i(n-1)e(n-1)x_i(n-1). \tag{14.72}$$

[2] The derivation presented here first appeared in Mathews and Xie (1990).

Substituting (14.72) in (14.71), and defining

$$g_i(n) = -2e(n)x_i(n), \tag{14.73}$$

we get

$$\frac{\partial e^2(n)}{\partial \mu_i(n-1)} = -g_i(n)g_i(n-1). \tag{14.74}$$

Finally, substituting (14.74) in (14.69) we obtain

$$\mu_i(n) = \mu_i(n-1) + \rho g_i(n)g_i(n-1), \tag{14.75}$$

for $i = 0, 1, \ldots, N-1$. This is the recursion (6.121) that was introduced in Chapter 6.

From (14.75) we note that the step-size parameters, the $\mu_i(n)$s, settle near their steady-state values when

$$\mathrm{E}[g_i(n)g_i(n-1)] = 0, \quad \text{for } i = 0, 1, \ldots, N-1. \tag{14.76}$$

Accordingly, a rigorous proof of the optimality of the VSLMS algorithm may be given by solving (14.76) for a set of unknown step-size parameters and showing that its solution matches the optimum parameters as given in (14.30). In fact, some researchers have proved this to be true for some specific cases (Mathews and Xie, 1993, and Farhang-Boroujeny, 1994). Here, we ignore such derivations because they are lengthy and a general solution turns out to be difficult to derive. Instead, we rely on simulations to verify the optimality of the VSLMS algorithm (see Section 14.7.4).

14.7.2 Variations and extensions

Sign update equation

Recall from Chapter 6 that the stochastic gradient terms, $g_i(n)$ and $g_i(n-1)$, in (14.75) may be replaced by their respective signs to obtain

$$\begin{aligned}\mu_i(n) &= \mu_i(n-1) + \rho \mathrm{sign}[g_i(n)] \cdot \mathrm{sign}[g_i(n-1)] \\ &= \mu_i(n-1) + \rho \mathrm{sign}[g_i(n)g_i(n-1)].\end{aligned} \tag{14.77}$$

This may be referred to as the *sign* update equation, in analogy with the LMS sign algorithm (see Section 6.5).

Multiplicative vs. linear increments

Other variations in the step-size update equations that have been proposed in the literature are (Farhang-Boroujeny, 1994)

$$\mu_i(n) = (1 + \rho g_i(n)g_i(n-1))\mu_i(n-1) \tag{14.78}$$

and its sign update version

$$\mu_i(n) = (1 + \rho \text{ sign}[g_i(n)g_i(n-1)])\mu_i(n-1). \tag{14.79}$$

For easy reference, we refer to (14.75) and (14.77) as *step-size update equations with linear increments*, and (14.78) and (14.79) as *step-size update equations with multiplicative increments*. We will make some comments on the performance of the linear and multiplicative increments later in Section 14.7.4.

Clearly, for a small ρ, (14.78) reaches it steady state when (14.76) is satisfied. This shows that both (14.75) and (14.78) converge to the same set of step-size parameters. Similarly, (14.77) and (14.79) converge to the same set of step-size parameters. Furthermore, *when* $g_i(n)g_i(n-1)$ *has a symmetrical distribution around its mean* (a case likely to happen, at least approximately, in most of applications) *and* ρ *is small, all of these step-size update equations converge to the same set of parameters.*

VSLMS algorithm with a common step-size parameter

In many applications, to keep the complexity of the filter low, we are often interested in using a common step-size parameter, $\mu(n)$, for all the filter taps. Following a similar line of derivations as in (14.68)–(14.75) the following recursion can be easily derived (Mathews and Xie, 1993):

$$\mu(n) = \mu(n-1) + \rho e(n)e(n-1)\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n-1). \tag{14.80}$$

The sign version of this recursion may be given as

$$\mu(n) = \mu(n-1) + \rho \text{sign}[e(n)e(n-1)\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n-1)]. \tag{14.81}$$

These are recursions with linear increments. Extension of these recursions to those with multiplicative increments is straightforward.

VSLMS algorithm for the complex-valued case

For filters with complex-valued input, following a similar line of derivation as that which led to (14.75) results in the following recursion (see Problem P14.7):

$$\mu_i(n) = \mu_i(n-1) + \rho(g_{i,\mathrm{R}}(n)g_{i,\mathrm{R}}(n-1) + g_{i,\mathrm{I}}(n)g_{i,\mathrm{I}}(n-1)). \tag{14.82}$$

Here, the subscripts R and I refer to the real and imaginary parts of $g_i(n)$ and $g_i(n-1)$, and

$$g_i(n) = -2e^*(n)x_i(n). \tag{14.83}$$

The sign version of (14.82) is

$$\mu_i(n) = \mu_i(n-1) + \rho(\text{sign}[g_{i,\mathrm{R}}(n)g_{i,\mathrm{R}}(n-1)] + \text{sign}[g_{i,\mathrm{I}}(n)g_{i,\mathrm{I}}(n-1)]). \tag{14.84}$$

Extensions of these results to update equations with multiplicative increments and also to the case where a common step-size parameter is used for all the filter taps are straightforward.

The final point to be noted here is that *the step-size parameters should always be limited to a range that satisfies the stability requirement of the LMS algorithm.* Equation (14.36) specifies the condition that must be satisfied by the step-size parameters to guarantee stability. However, how this is implemented in actual practice to limit the step-size parameters can vary. In Section 14.7.4 we discuss a possible way of limiting the step-size parameters.

14.7.3 Normalization of the parameter ρ

In the update equations (14.75) and (14.78), and similar equations for the complex-valued case, the step-size increments are proportional to the size of $g_i(n)g_i(n-1)$. This might be inappropriate when signal levels vary significantly with time. This results in fast and slow variations in the step-size parameters depending on the level of the input signal, $x_i(n)$, and also the output error, $e(n)$. To keep a more uniform control (variation) of the step-size parameters which does not depend on signal levels, we adopt a step-normalization technique similar to the one used in the LMS algorithm. Here, the adaptation parameter ρ is replaced by $\rho_i(n)$ which is obtained according to the equation

$$\rho_i(n) = \frac{\rho_o}{\hat{\sigma}_g^2(n) + \psi}, \tag{14.85}$$

where ρ_o is an unnormalized parameter common to all taps, $\hat{\sigma}_g^2(n)$ is an estimate of $E[|g_i(n)|^2]$ that may be obtained through the recursion

$$\hat{\sigma}_g^2(n) = \beta\hat{\sigma}_g^2(n-1) + (1-\beta)|g_i(n)|^2, \tag{14.86}$$

where β is a forgetting factor close to but less than one, and ψ is a positive constant which prevents possible instability of the algorithm when $\hat{\sigma}_g^2(n)$ is small.

In the case of (14.80), the normalization is done with respect to $E[e^2(n)\mathbf{x}^T(n)\mathbf{x}(n)]$. This can also be estimated through a time-averaging recursion similar to (14.86).

14.7.4 Computer simulations

In this section we present some simulation results to illustrate the optimal tracking behaviour of the VSLMS algorithm. As an example, we consider the application of the VSLMS algorithm in the identification of a multipath communication channel.[3] The channel is assumed to have two distinct paths with a continuous-time impulse response

$$h_t(t_o) = a_1(t_o)p(t - \tau_1(t_o)) + a_2(t_o)p(t - \tau_2(t_o)), \tag{14.87}$$

[3] The simulation results presented here are simplified versions of those presented by the author in Farhang-Boroujeny (1994). Here, we consider the case where all variables are real-valued. In Farhang-Boroujeny (1994) all the variables are assumed to be complex-valued. However, the conclusions derived from the results of this section as well as those in Farhang-Boroujeny (1994) are similar.

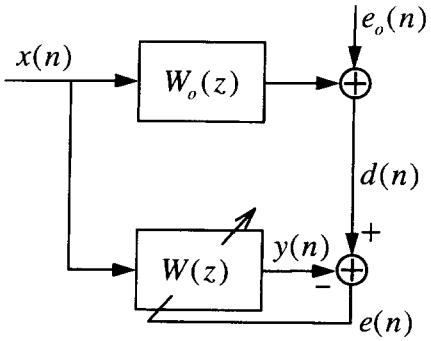


Figure 14.2 Modelling of a multipath communication channel

where t_o is the time at which the channel response is given (measured), $\tau_1(t_o)$ and $\tau_2(t_o)$ are the path delays, $a_1(t_o)$ and $a_2(t_o)$ are the path gains, t is the continuous time variable, $p(t)$ is the raised-cosine pulse with 50% roll-off factor given by

$$p(t) = \frac{\sin(\pi t/T_s)}{\pi t/T_s} \cdot \frac{\cos(\pi t/2T_s)}{1-(t/T_s)^2}, \tag{14.88}$$

and T_s is the symbol interval. As explicitly indicated in (14.87), the path delays and gains

An adaptive filter is used to track these variations.

For typical values of f_d that allow the VSLMS algorithm to follow variations in the channel, the variations in $a_1(nT_s)$ and $a_2(nT_s)$ very closely approximate a random walk model similar to the one used to develop the analytical results of the present chapter (Eweda, 1994); see also Problem P14.8. This approximate realization of a random walk prevents an indefinite increase in the path gains which would otherwise happen if we had used the random walk model of Section 14.1.

The following parameters are used in all the simulations that follow. The channel length, N, is set equal to 16. The same length is also assumed for the channel model (adaptive filter). The tails of the raised cosine pulses associated with the two paths that lie outside the range set by the channel length are truncated. A fade rate of $f_d = f_s/2400$ is assumed. In the implementation of the sign update equations with multiplicative increments, we choose $\rho = 0.002$. For the conventional update equations (14.75) and (14.78), the parameter ρ is normalized, as discussed in Section 14.7.3. The following parameters are used:

- for (14.75), $\beta = 0.95$, $\psi = 0.001$, and $\rho_o = 0.0002$;
- for (14.78), $\beta = 0.95$, $\psi = 0.001$, and $\rho_o = 0.002$.

The data sequence, $s(n)$, at the channel and adaptive filter input is a binary zero-mean white random process, taking values ± 1. The channel noise, $e_o(n)$, is a zero-mean white Gaussian process with variance $\sigma_{e_o}^2 = 0.02$. This choice of $\sigma_{e_o}^2$ results in an average signal-to-noise ratio of 20 dB at the channel output.

The step-size parameters are checked at the end of every iteration of the algorithm and limited to stay within a range that satisfies (14.36). When (14.36) is not satisfied, all the step-size parameters are scaled down by the same factor such that $\mathrm{tr}[\boldsymbol{\mu}\mathbf{R}]$ reduces to its upper bound 1/3. The step-size parameters are also hard limited to the minimum value of 0.001.

Next, we present a number of results comparing the relative performance of various implementations of the step-size adaptation in the present application. These results also serve to show the optimal tracking behaviour of the VSLMS algorithm.

Figure 14.3 presents a typical result comparing the performance of the update equations (14.75) and (14.78), i.e. the recursions with linear and multiplicative increments, respectively. These and the subsequent results of this section are based on single simulation runs, i.e. no ensemble averaging is used. However, time-averaging with a moving rectangular window is used to smooth the plots. We note that adaptation based on multiplicative increments results in lower steady-state MSE, i.e. a superior tracking behaviour. This may be explained by noting that the variation in step-size parameters

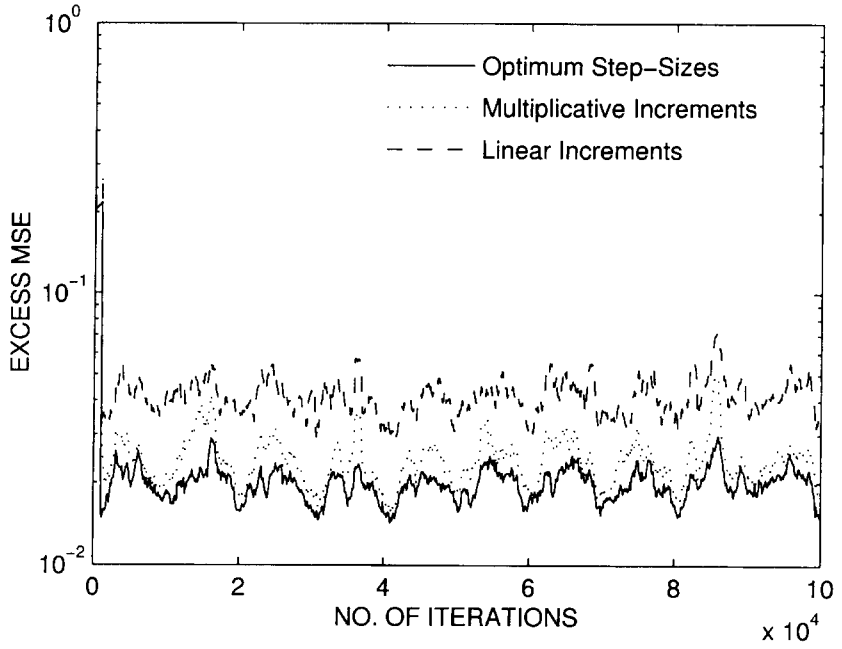


Figure 14.3 A typical simulation result comparing the VSLMS algorithm with linear and multiplicative increments

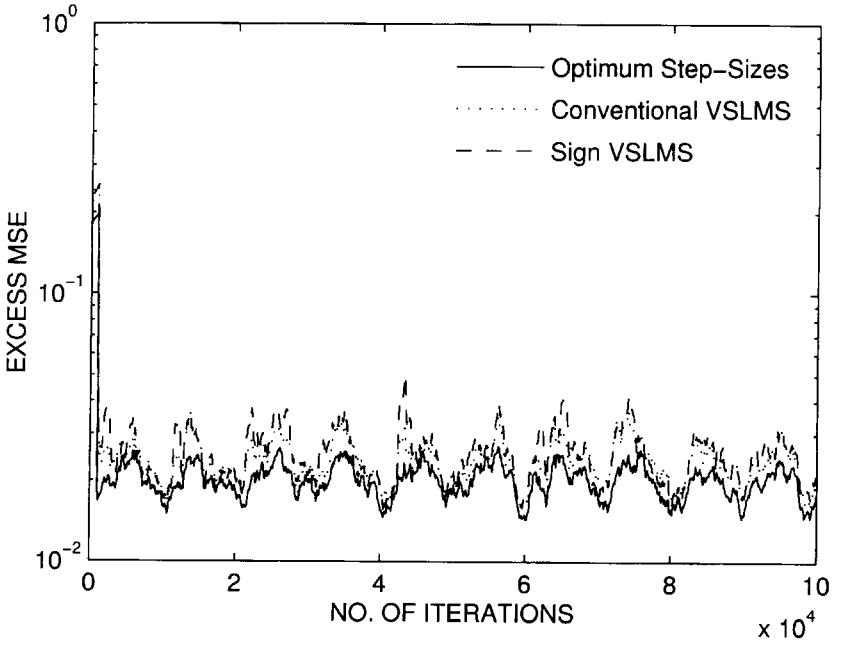
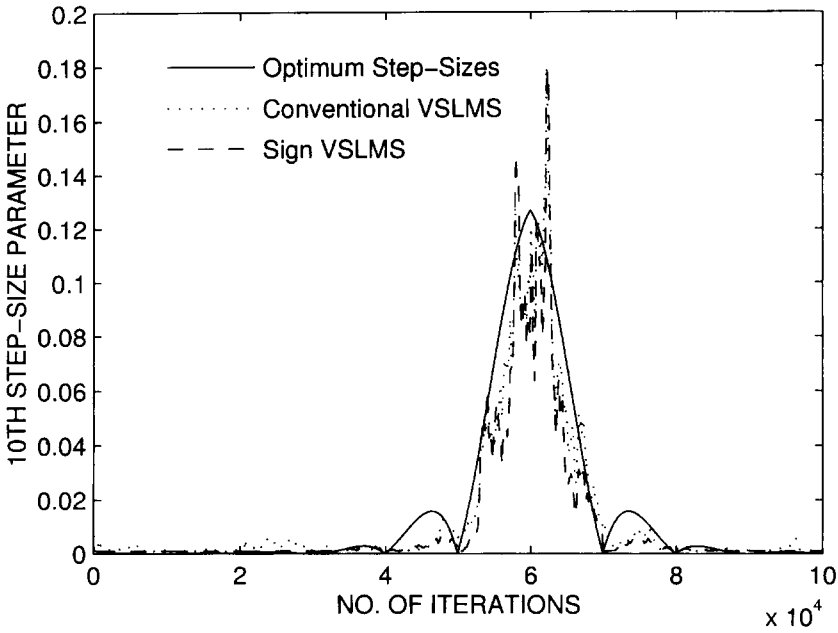


Figure 14.4 A typical simulation result comparing the LMS algorithm with the optimum step-size parameters and the VSLMS algorithm



optimum step-size parameters given by (14.30)

tracking behaviour of the VSLMS algorithm, as was predicted earlier in this section. We also note that there is very little difference between the behaviour of the recursion (14.78) and its sign counterpart, (14.79).

Figure 14.5 presents the results showing how the VSLMS algorithm tracks variations

Thus, for $k = 1$ and 2, we get

$$a_k((n+1)T_s) - a_k(nT_s) = -(1-\alpha)a_k(nT_s) + \sqrt{1-\alpha^2}\nu_k(n+1)$$
$$\approx \sqrt{1-\alpha^2}\nu_k(n+1), \tag{14.94}$$

where the latter approximation is (statistically) justified by noting that $a_k(nT_s)$ and $\nu_k(n+1)$ are two random variables with the same variance. On the other hand, from (14.2) we get $\varepsilon_o(n) = \mathbf{w}_o(n+1) - \mathbf{w}_o(n)$, or

$$\varepsilon_{o,i}(n) = w_{o,i}(n+1) - w_{o,i}(n). \tag{14.95}$$

Next, substituting (14.90) in (14.95), and assuming that the path delays $\tau_1(nT_s)$ and $\tau_2(nT_s)$ vary very slowly in time so that their variations over the span of the channel length (NT_s s) can be ignored we obtain, using (14.94),

$$\varepsilon_{o,i}(n) = \sqrt{1-\alpha^2}[\nu_1(n+1)p(iT_s - \tau_1(nT_s)) + \nu_2(n+1)p(iT_s - \tau_2(nT_s))], \tag{14.96}$$

for $i = 0, 1, \ldots, N-1$. Using (14.96) and recalling that $\nu_1(n+1)$ and $\nu_2(n+1)$ are unit-variance independent processes we obtain

$$\sigma^2_{\varepsilon_{o,i}}(n) = (1-\alpha^2)(p^2(iT_s - \tau_1(nT_s)) + p^2(iT_s - \tau_2(nT_s))) \tag{14.97}$$

or

$$\sigma_{\varepsilon_{o,i}}(n) = \sqrt{(1-\alpha^2)(p^2(iT_s - \tau_1(nT_s)) + p^2(iT_s - \tau_2(nT_s)))}. \tag{14.98}$$

14.8 The RLS Algorithm with a Variable Forgetting Factor

In this section we extend the idea of the VSLMS algorithm to propose an RLS algorithm with a variable forgetting factor. To this end, we recall from the previous chapter that in the steady state the RLS algorithm is approximately equivalent to the LMS–Newton algorithm. In particular, when the filter input is stationary, in the steady state, the RLS recursion is approximately equivalent to the recursion

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + (1-\lambda(n))\hat{e}_{n-1}(n)\mathbf{R}^{-1}\mathbf{x}(n) \tag{14.99}$$

(see (12.90)). Note that here we have added the time index n to the forgetting factor $\lambda(n)$ to emphasize that it may vary with time.

Starting with (14.99) and following the same line of derivations as those used to derive the VSLMS algorithm, the following update equation is obtained for the adaptive adjustment of $\lambda(n)$:

$$\lambda(n) = \lambda(n-1) - \rho z(n), \tag{14.100}$$

where

$$z(n) = (\hat{e}_{n-1}(n)\mathbf{x}(n))^T(\hat{e}_{n-2}(n-1)\mathbf{R}^{-1}\mathbf{x}(n-1)). \tag{14.101}$$

A more robust update equation for the adaptation of $\lambda(n)$ is obtained by defining

$$\beta(n) = 1 - \lambda(n) \tag{14.102}$$

and noting that (14.100) can equivalently be written as

$$\beta(n) = \beta(n-1) + \rho z(n). \tag{14.103}$$

Moreover, from our experience with the VSLMS algorithm, we may suggest using multiplicative increments instead of linear increments, and also replacing $z(n)$ by its sign, to obtain the following recursion:

$$\beta(n) = (1 + \rho \operatorname{sign}[z(n)])\beta(n-1). \tag{14.104}$$

To get a more easily usable expression for $z(n)$, from Chapter 12 we recall that in the steady state

$$\boldsymbol{\Psi}_\lambda(n) \approx \frac{1}{1-\lambda(n)}\mathbf{R}. \tag{14.105}$$

Table 14.2 Summary of the RLS algorithm with a variable forgetting factor

Input:	Tap-weight vector estimate, $\hat{\mathbf{w}}(n-1)$, Input vector, $\mathbf{x}(n)$, desired output, $d(n)$, Forgetting factor, $\lambda(n-1)$, gain vector $\mathbf{k}(n-1)$, and the matrix $\boldsymbol{\Psi}_\lambda^{-1}(n-1)$.
Output:	Filter output, $\hat{y}_{n-1}(n)$, Tap-weight vector update, $\hat{\mathbf{w}}(n)$, Forgetting factor $\lambda(n)$, and the updated matrix $\boldsymbol{\Psi}_\lambda^{-1}(n)$.

1. Computation of the gain vector:
$$\mathbf{u}(n) = \boldsymbol{\Psi}_\lambda^{-1}(n-1)\mathbf{x}(n)$$
$$\mathbf{k}(n) = \frac{1}{\lambda(n-1) + \mathbf{x}^{\mathrm{T}}(n)\mathbf{u}(n)}\mathbf{u}(n)$$
2. Filtering:
$$\hat{y}_{n-1}(n) = \hat{\mathbf{w}}^{\mathrm{T}}(n-1)\mathbf{x}(n)$$
3. Error estimation:
$$\hat{e}_{n-1}(n) = d(n) - \hat{y}_{n-1}(n)$$
4. Tap-weight vector adaptation:
$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\hat{e}_{n-1}(n)$$
5. $\boldsymbol{\Psi}_\lambda^{-1}(n)$ update:
$$\boldsymbol{\Psi}_\lambda^{-1}(n) = \mathrm{Tri}\{\lambda^{-1}(n-1)(\boldsymbol{\Psi}_\lambda^{-1}(n-1) - \mathbf{k}(n)\mathbf{u}^{\mathrm{T}}(n))\}$$
6. $\lambda(n)$ update:
$$\beta(n-1) = 1 - \lambda(n-1)$$
$$\beta(n) = \{1 + \rho \operatorname{sign}[\hat{e}_{n-1}(n)\hat{e}_{n-2}(n-1)] \times \operatorname{sign}[\mathbf{x}^{\mathrm{T}}(n)\mathbf{k}(n-1)]\}\beta(n-1)$$
$$\lambda(n) = 1 - \beta(n)$$
if $\lambda(n) > \lambda^+$, $\lambda(n) = \lambda^+$
if $\lambda(n) < \lambda^-$, $\lambda(n) = \lambda^-$

Rearranging (14.105) and replacing n by $n-1$ we obtain

$$\mathbf{R}^{-1} \approx \frac{1}{1-\lambda(n-1)} \boldsymbol{\Psi}_{\lambda}^{-1}(n-1). \tag{14.106}$$

Substituting (14.106) in (14.101) and using the definition (12.48) we obtain

$$\begin{aligned} z(n) &= \hat{e}_{n-1}(n)\hat{e}_{n-2}(n-1)\frac{\mathbf{x}^{\mathrm{T}}(n)\boldsymbol{\Psi}_{\lambda}^{-1}(n-1)\mathbf{x}(n-1)}{1-\lambda(n-1)} \\ &= \hat{e}_{n-1}(n)\hat{e}_{n-2}(n-1)\frac{\mathbf{x}^{\mathrm{T}}(n)\mathbf{k}(n-1)}{1-\lambda(n-1)}, \end{aligned} \tag{14.107}$$

where $\mathbf{k}(n)$ is the gain vector of the RLS algorithm. Taking the sign of $z(n)$ and recalling that $1-\lambda(n)$ is positive, since $\lambda(n) < 1$, we get

$$\operatorname{sign}[z(n)] = \operatorname{sign}[\hat{e}_{n-1}(n)\hat{e}_{n-2}(n-1)] \times \operatorname{sign}[\mathbf{x}^{\mathrm{T}}(n)\mathbf{k}(n-1)]. \tag{14.108}$$

Using the above results, Table 14.2 presents a summary of the implementation of the RLS algorithm with a variable forgetting factor – compare this with Table 12.2. Note that after every iteration the forgetting factor, $\lambda(n)$, is checked and limited to some pre-selected values, λ^{+} and λ^{-}.

14.9 Summary

In this chapter we studied the tracking behaviour of various adaptive filtering algorithms in the context of a system modelling problem. We introduced and analysed a generalized formulation of the LMS algorithm which could cover most of the algorithms discussed in previous chapters. The general conclusion derived from the analysis is that convergence and tracking are two different phenomena, and hence should be treated separately. We found that the algorithms that were previously introduced to speed up the convergence of adaptive filters do not necessarily have superior tracking behaviour. We presented cases where the conventional LMS algorithm, which has the slowest convergence behaviour, has better tracking behaviour than those with more complicated structures, such as the TDLMS or even the RLS algorithm.

We also considered the variable step-size LMS (VSLMS) algorithm (of Chapter 6) as an adaptive filtering scheme with optimum tracl ing. The optimal tracking behaviour of the VSLMS algorithm was confirmed through computer simulations. The idea of the VSLMS algorithm was also extended to the RLS algorithm by introducing a similar adaptive mechanism for controlling its forgetting factor (memory length).

Most of the present literature on the tracking behaviour of adaptive filters and also our discussion in this chapter is limited to the case where the adaptive filter input is a stationary process.[4] Only the desired output of the filter has been assumed to be non-stationary. We

[4] An exception is the work of Macchi and Bershad (1991) who compared the tracking performance of the LMS and RLS algorithms in recovering a chirped sinusoid. This is an example where the adaptive filter input is non-stationary.

may thus say that the treatment of the problem of tracking in the present literature is rather immature and much more work needs to be done on this very important topic.

Problems

P14.1 In the derivation of (14.19), we assumed that $\text{tr}[\boldsymbol{\mu}\mathbf{R}\mathbf{K}\mathbf{R}] \ll \text{tr}[\boldsymbol{\mu}\mathbf{R}]\xi_{\text{ex}}$. There, we remarked that on average this assumption becomes more accurate (valid) as the filter length, N, increases. In this problem we examine the validity of the assumption for a few specific cases.

(i) Show that in the case where $\mathbf{R} = \mathbf{I}$ ($\mathbf{I}$ is the identity matrix) and $\boldsymbol{\mu} = \mu\mathbf{I}$, i.e. a single scalar step-size parameter is used for all the taps,

$$\frac{\text{tr}[\boldsymbol{\mu}\mathbf{R}\mathbf{K}\mathbf{R}]}{\text{tr}[\boldsymbol{\mu}\mathbf{R}]\xi_{\text{ex}}} = \frac{1}{N}. \tag{P14.1-1}$$

(ii) Consider the case where $\mathbf{R}$ is diagonal and the elements of $\boldsymbol{\mu}$ are chosen according to (14.4). Show that in this case also (P14.1-1) holds.

(iii) Consider the case where $\boldsymbol{\mu} = \mu\mathbf{I}$, but $\mathbf{R}$ and $\mathbf{G}$ are arbitrary correlation matrices. Use the decomposition $\mathbf{R} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{\text{T}}$ (equation (4.19) of Chapter 4), to show that

$$\text{tr}[\boldsymbol{\mu}\mathbf{R}\mathbf{K}\mathbf{R}] = \mu \sum_{i=0}^{N-1} \lambda_i^2 k'_{ii}$$

and

$$\text{tr}[\boldsymbol{\mu}\mathbf{R}]\xi_{\text{ex}} = \mu\left(\sum_{i=0}^{N-1} \lambda_i\right)\left(\sum_{i=0}^{N-1} \lambda_i k'_{ii}\right),$$

where λ_i and k'_{ii} are the ith diagonal elements of the matrices $\boldsymbol{\Lambda}$ and $\mathbf{K}' = \mathbf{Q}^{\text{T}}\mathbf{K}Q$, respectively. Then, study the ratio

$$\frac{\text{tr}[\boldsymbol{\mu}\mathbf{R}\mathbf{K}\mathbf{R}]}{\text{tr}[\boldsymbol{\mu}\mathbf{R}]\xi_{\text{ex}}}$$

and find how the distribution of the λ_is and k'_{ii}s affect this ratio.

P14.2 In the case of the conventional LMS algorithm, i.e. where $\boldsymbol{\mu} = \mu\mathbf{I}$ with μ being a scalar, show that

(i) $$\xi_{\text{ex}} = \frac{1}{1 - \mu\text{tr}[\mathbf{R}]}(\mu\sigma_{e_o}^2\text{tr}[\mathbf{R}] + \tfrac{1}{4}\mu^{-1}\text{tr}[\mathbf{G}]).$$

(ii) Assuming that $\mu\text{tr}[\mathbf{R}] \ll 1$ for the range of interest of μ, show that the optimum value of μ that minimizes ξ_{ex} is

$$\mu_o \approx \frac{1}{2\sigma_{e_o}}\sqrt{\frac{\text{tr}[\mathbf{G}]}{\text{tr}[\mathbf{R}]}}.$$

(iii) Show that when $\mu = \mu_o$, the noise and lag misadjustments of the LMS algorithm are equal.
(iv) Show that the minimum value of ξ_{ex} is given by

$$\xi_{ex,o} \approx \sigma_{e_o} \sqrt{\text{tr}[\mathbf{R}]\text{tr}[\mathbf{G}]}.$$

P14.3 A shortcoming of the result of the previous problem is that when σ_{e_o} is very small (i.e. the plant noise is very small) the calculated optimum step-size parameter, μ_o, may become excessively large, resulting in an unstable LMS algorithm. Recalculate μ_o and $\xi_{ex,o}$ without imposing the condition $\mu\text{tr}[\mathbf{R}] \ll 1$ and show that the results are

$$\mu_o = \frac{\sqrt{\text{tr}[\mathbf{G}]/\text{tr}[\mathbf{R}]}}{\sqrt{\text{tr}[\mathbf{G}]\text{tr}[\mathbf{R}]} + \sqrt{\text{tr}[\mathbf{G}]\text{tr}[\mathbf{R}] + 4\sigma_{e_o}^2}}$$

and

$$\xi_{ex,o} = \frac{\left(\sqrt{\text{tr}[\mathbf{G}]\text{tr}[\mathbf{R}]} + \sqrt{\text{tr}[\mathbf{G}]\text{tr}[\mathbf{R}] + 4\sigma_{e_o}^2}\right)\sqrt{\text{tr}[\mathbf{G}]\text{tr}[\mathbf{R}]}}{2}.$$

Simplify these results when the plant noise is zero.

P14.4 Show that the solution of equations (14.27) is (14.28).

P14.5 Give a detailed derivation of (14.29).

P14.6 Give a detailed derivation of (14.80).

P14.7 This problem aims at giving a derivation of the VSLMS algorithm for the complex-valued case.
(i) Show that (14.82) can also be written as

$$\mu_i(n) = \mu_i(n-1) + \rho\Re[g_i(n)g_i^*(n-1)], \qquad \text{(P14.7-1)}$$

where $\Re[x]$ denotes real-part of x.
(ii) Following a similar line of derivation to those in Section 14.7.1, give a detailed derivation of (P14.7-1).

P14.8 Consider the Markovian process, $w(n)$, generated through the recursive equation

$$w(n) = \alpha w(n-1) + \nu(n),$$

where α is a parameter in the range of -1 to $+1$ and $\nu(n)$ is a stationary white noise.
(i) Define the sequence $\varepsilon(n) = w(n) - w(n-1)$ and show that when $k \neq 0$

$$\frac{\phi_{\varepsilon\varepsilon}(k)}{\phi_{\varepsilon\varepsilon}(0)} = \frac{\alpha - 1}{\alpha + 1}\alpha^{|k|-1}\sigma_\nu^2,$$

where $\phi_{\varepsilon\varepsilon}(k)$ is the autocorrelation function of $\varepsilon(n)$.

(ii) Use the result of part (i) to argue that the results presented in this chapter are also valid (within a good approximation) when the tap weights of the plant in the modelling problem of Figure 14.1 vary according to a Markovian model with a parameter α smaller, but close to one.

P14.9 Consider the LMS–Newton recursion

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu\mathbf{R}^{-1}e(n)\mathbf{x}(n)$$

when applied for tracking in the modelling problem of Section 14.1.

(i) Show that

$$\xi_{ex} = \frac{1}{1-\mu N}\left(\mu\sigma_{e_o}^2 N + \frac{\mu^{-1}}{4}\text{tr}[\mathbf{RG}]\right).$$

(ii) Let μ_o denote the optimum value of μ which results in minimum ξ_{ex}. Assuming that the plant changes very slowly so that $\mu_o N \ll 1$, show that

$$\mu_o = \frac{1}{2\sigma_{e_o}}\sqrt{\text{tr}[\mathbf{RG}]}.$$

(iii) Obtain an approximate expression for $\xi_{ex,o}$, i.e. the minimum value of ξ_{ex}, and compare that with your results in part (iv) of Problem P14.2.

(iv) Repeat parts (ii) and (iii) for the case where the condition $\mu_o N \ll 1$ does not hold.

P14.10 Suggest a variable step-size implementation of the LMS–Newton algorithm of Table 11.7.

P14.11 Give a detailed derivation of (14.100).

Simulation-Oriented Problems

P14.12 Consider a two-tap modelling problem with the parameters as specified in Case 3 of Section 14.5. Develop a simulation program for implementation of the LMS and TDLMS algorithms in this case. By running your program confirm that the predictions made through the numerical results of Table 14.1 are consistent with simulations. That is, the choice of $\theta = \pi/8$ results in the least steady-state MSE, and $\theta = \pi/4$ results in the maximum steady-state MSE.

Tip: To generate a random vector $\mathbf{x}(n)$ with a correlation matrix $\mathbf{R}$, you may proceed as follows. First find a square matrix $\mathbf{L}$ that satisfies $\mathbf{R} = \mathbf{L}^T\mathbf{L}$. You may then generate $\mathbf{x}(n)$ according to the equation $\mathbf{x}(n) = \mathbf{L}^T\mathbf{u}(n)$, where $\mathbf{u}(n)$ is a random vector with the correlation matrix equal to the identity matrix. If you are using MATLAB, a convenient way of obtaining a square matrix $\mathbf{L}$ that satisfies $\mathbf{R} = \mathbf{L}^T\mathbf{L}$ is by using the function 'chol' which finds Cholesky factorization of $\mathbf{R}$. The same method can be used to generate a random vector $\varepsilon_o(n)$ with a correlation matrix $\mathbf{G}$.

P14.13 The MATLAB simulation programs used to generate the results of Figures 14.3–14.5 are available on an accompanying diskette. They are called 'gtf14_3.m' and 'gtf14_4.m'. Experiment with these programs and confirm the results of Figures 14.3–14.5. Try also other variations of the simulation parameters to gain a better understanding of the behaviour of various implementations of the VSLMS algorithm.

P14.14 Develop a simulation program to study the convergence behaviour of the VSLMS algorithm in the channel modelling application that was discussed in Section 14.7.4, when a common step-size parameter is used for all taps. Compare your results with those of Figures 14.3–14.5, and discuss your observations.

P14.15 Develop a simulation program to study the convergence behaviour of the RLS algorithm with a variable forgetting factor (Table 14.2) in the channel modelling application that was discussed in Section 14.7.4. Compare your results with those of Figures 14.3–14.5 and also your results in Problem P14.14. Discuss your observations.

Appendix I: List of MATLAB Programs

Script Programs

1.	modelling.m	System modelling: conventional LMS algorithm (Chapter 6).
2.	equalizer.m	Channel equalization: conventional LMS algorithm (Chapter 6).
3.	lenhncr.m	Line enhancer: conventional LMS algorithm (Chapter 6).
4.	bformer.m	Narrow-band beamformer for a two element array: conventional LMS algorithm (Chapter 6).
5.	blk_mdlg.m	System modelling: Block LMS algorithm (Chapter 8).
6.	gtf10_9a.m	IIR line enhancer algorithm/1 (Chapter 10). This program, as its name suggests, has been used to generate the results of Figure 10.9a,
8.	gtf10_9b.m	IIR line enhancer algorithm 2 (Chapter 10). This program, as its name suggests, has been used to generate the results of Figure 10.9b.
9.	gtf10_11.m	Cascaded IIR line enhancer algorithm 2 (Chapter 10). This program, as its name suggests, has been used to generate the results of Figure 10.11.
10.	iirdsgn.m	IIR equalizer design for magnetic recording channels (Chapter 10).
11.	ltc_mdlg.m	System modelling: lattice structure with LMS adaptation (Chapter 11).
12.	ar_m_al1.m	System modelling: LMS–Newton algorithm 1 (Chapter 11).
13.	ar_m_al2.m	System modelling: LMS–Newton algorithm 2 (Chapter 11).
14.	rlsI.m	System modelling: Standard RLS algorithm (Version I) (Chapter 12).
15.	gtf14_3.m	Channel modelling: VSLMS algorithm (Chapter 14). This program, as its name suggests, has been used to generate the results of Figure 14.3.
16.	gtf14_4.m	Channel modelling: VSLMS algorithm (Chapter 14). This program, as its name suggests, has been used to generate the results of Figure 14.4.

Function Programs

1.	corlnm2.m	This function obtains the correlation matrix of a moving average process generated by passing a white noise process through a FIR filter.
2.	dftm.m	This function produces the discrete Fourier transform (DFT) matrix of a specified dimension.
3.	rdftm.m	This function produces the real-DFT matrix of a specified dimension.
4.	dctm.m	This function produces the discrete cosine transform (DCT) matrix of a specified dimension.
5.	dstm.m	This function produces the discrete sine transform (DST) matrix of a specified dimension.
6.	dhtm.m	This function produces the discrete Hartley transform (DHT) matrix of a specified dimension.
7.	whtm.m	This function produces the Walsh Hadamard transform (WHT) matrix of a specified dimension.
6.	eigenfir.m	FIR filter design using the design technique discussed in Section 9.7.1 (Chapter 9)
8.	lorentz.m	Lorentzian pulse generator. This function is called by iirdsgn.m.
9.	leqlzr.m	This function calculate the coefficient of a linear fractionally tap-spaced equalizer, based on related channel information. This function is called by iirdsgn.m.
10.	lsfft.m	Least-squares fit: finds the least-squares IIR fit of a FIR filter. This function is called by iirdsgn.m.
11.	ljpe.m	Lattice Joint Process Estimator. This function program follows Table 11.5. It is called by ltc_mdlg.m.
12.	rcos50.m	This function gives samples of a raised cosine pulse shape with a rolloff factor of 50%. It is called by gtf14_3.m and gtf14_4.m.

References

Ahmed, N., D. Hush, G. R. Elliott and R. J. Fogler (1984). 'Detection of multiple sinusoids using an adaptive cascaded structure,' in *Proc. ICASSP'84*, pp. 21.3.1–21.3.4.

Alexander, S. T. (1986a). *Adaptive Signal Processing: Theory and Applications*. Springer-Verlag, New York.

Alexander, S. T. (1986b). Fast adaptive filters: a geometrical approach, *IEEE ASSP Mag.*, **3**, no. 4, 18–28.

Amano, F., H. P. Meana, A. de Luca and G. Duchen (1995). 'A multirate acoustic echo canceller structure,' *IEEE Trans. Commun.*, **43**, no. 7, 2172–2176.

Anderson, B. D. O. and J. B. Moore (1979). *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, NJ.

Ardalan, S. H. (1987). 'Floating-point error analysis of recursive least-squares and least-mean-square adaptive filters,' *IEEE Trans. Circuits and Syst.*, **CAS-33**, no. 12, 1192–1208.

Ardalan, S. H. and S. T. Alexander (1987). 'Fixed-point roundoff error analysis of the exponentially windowed RLS algorithm for time-varying systems,' *IEEE Trans. Acoust. Speech and Signal Process.*, **ASSP-35**, no. 6, 770–783.

Asharif, M. R., F. Amano, S. Unagami and K. Muramo (1986a). 'A new structure of echo canceller based on frequency bin adaptive filtering (FBAF),' *DSP Symp.*, Japan, pp. 165–169.

Asharif, M. R., T. Takebayashi, T. Chugo and K. Murano (1986b). 'Frequency domain noise canceller: frequency-bin adaptive filtering (FBAF),' in *Proceedings of ICASSP'86*, April 7–11, Tokyo, Japan, pp. 41.22.1–41.22.4.

Asharif, M. R. and F. Amano (1994). 'Acoustic echo-canceler using the FBAF algorithm,' *Trans. Commun.*, **42**, no. 12, 3090–3094.

Ashihara, K., K. Nishikawa and H. Kiya (1995). 'Improvement of convergence speed for subband adaptive digital filters using the multirate repeating method,' in *Proc. IEEE ICASSP'95*, Detroit, MI, May, vol. 2, pp. 989–992.

Åström, K. J. and B. Wittenmark (1989). *Adaptive Control*. Adison-Wesley, Reading, Massachusetts.

Benallal, A. and A. Gilloire (1988). 'A new method to stabilize fast RLS algorithms based on a first-order model of the propagation of numerical errors,' in *Proc. ICASSP'88 Conf.*, vol. 3, pp. 1373–1376.

Benveniste, A. (1987). 'Design of adaptive algorithms for the tracking of time-varying systems,' *Int. J. Adaptive Control Signal Process.*, **1**, no. 1, 3–29.

Benveniste, A. and G. Ruget (1982). 'A measure of the tracking capability of recursive stochastic algorithms with constant gains,' *IEEE Trans. Autom. Control*, **AC-27**, 639–649.

Bergland, G. D. (1968). 'A fast Fourier transform algorithm for real-valued series,' *Commun. ACM*, **11**, no. 10, 703–710.

Bergmans, J. W. M. (1996). *Digital Baseband Transmission and Recording*. Kluwer Academic Publisher, Netherland.

Bershad, N. J. (1986). 'Analysis of the normalized LMS algorithm with Gaussian inputs,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-34**, no. 4, 793–806.

Bershad, N. J. and O. M. Macchi (1991). 'Adaptive recovery of a chirped sinusoid in noise, Part 2: performance of the LMS algorithm,' *IEEE Trans. Signal Process.*, **39**, no. 3, 595 602.

Botto, J.-L. and G. V. Moustakides (1989). 'Stabilizing the fast Kalman Algorithms,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-37**, no. 9, 1342–1348.

Brandwood, D. H. (1983). 'A complex gradient operator and its application in adaptive array theory,' *IEE Proc.*, **130**, parts F and H, no. 1, 11–16.

Bruun, G. (1978). 'z-Transform DFT filters and FFT's,' *IEEE Trans. Acoust., Speech, Signal Processing*, **ASSP-26**, no. 1, pp. 56–63.

Capon, J. (1969). 'High-resolution frequency-wavenumber spectrum analysis,' *Proc. IEEE*, **57**, no. 8, 1408–1419.

Caraiscos, C. and B. Liu (1984). 'A round-off analysis of the LMS adaptive algorithm,' *IEEE Trans. Acoust. Speech and Signal Process.*, **ASSP-32**, no. 1, 34–41.

Chen, J., H. Bes, J. Vandewalle and P. Janssens (1988). 'A new structure for sub-band acoustic echo canceller,' in *Proc. IEEE ICASSP'88*, New York, April, pp. 2574–2577.

Cho, N. I. and S. U. Lee (1993). 'On the adaptive lattice notch filter for the detection of sinusoids,' *IEEE Trans. Circuits and Syst. II: Analog and Digital Signal Process.*, **40**, no. 7, 405–416.

Cioffi, J. M. (1987a). 'Limited-precision effects in adaptive filtering,' *IEEE Trans. Circuits and Syst.*, **CAS-34**, no. 7, 821–833.

Cioffi, J. M. (1987b). 'A fast QR/frequency-domain RLS adaptive filter,' in *Proc. ICASSP-87 Conf.*, vol. 1, pp. 407–410.

Cioffi, J. M. (1990a). 'A fast echo canceller initialization method for the CCITT V.32 modem,' *IEEE Trans. Commun.*, **38**, no. 5, 629–638.

Cioffi, J. M. (1990b). 'The fast adaptive rotors RLS algorithm,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-38**, no. 4, 631–651.

Cioffi, J. M. and T. Kailath (1984). 'Fast recursive least-squares transversal filters for adaptive filtering,' *IEEE Trans. Acoustic, Speech and Signal Process.*, **ASSP-32**, no. 2, 304–337.

Cioffi, J. M. and T. Kailath (1985). 'Windowed fast transversal filters adaptive algorithms with normalization,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-33**, no. 3, 607–625.

Claasen, T. A. C. M. and W. F. G. Mecklenbräuker (1981). 'Comparison of the convergence of two algorithms for adaptive FIR digital filters,' *IEEE Trans. Acoustic, Speech and Signal Process.*, **ASSP-29**, no. 3, 670–678.

Clark, G. A., S. K. Mitra and S. R. Parker (1981). 'Block implementation of adaptive digital filters,' *IEEE Transactions on Circuits and Syst.*, **CAS-28**, no. 6, 584–592.

Clark, G. A., S. R. Parker and S. K. Mitra (1983). 'A unified approach to time and frequency domain realization of FIR adaptive digital filters,' *IEEE Trans. Acoustic, Speech and Signal Process.*, **ASSP-31**, no. 5, 1073–1083.

Clark, A. P. and S. F. Hau (1984). 'Adaptive adjustment of receiver for distorted digital signals,' *IEE Proceedings*, **131**, part F, 526–536.

Clark, A. P. and S. Hariharan (1989). 'Adaptive channel estimator for an HF radio link,' *IEEE Trans. Commun.*, **COM-37**, no. 9, 918–926.

Crochiere R. E. and L. R. Rabiner (1983). *Multirate Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Cupo L. R. and R. D. Gitlin (1989). 'Adaptive carrier recovery systems for digital data communications receivers,' *IEEE Journal on Selected Areas in Commun.*, **7**, no. 9, 1328–1339.

David, R. D., S. D. Stearns, G. R. Elliott and D. M. Etter (1983). 'IIR algorithm for adaptive line enhancement,' in *Proc. ICASSP'83*, pp. 17–20.

Davila, C. E. (1990). 'A stochastic Newton algorithm with data-adaptive step size,' *IEEE Trans. Acoustic, Speech and Signal Process.*, **ASSP-38**, no. 10, 1796–1798.

De Courville, M. and P. Duhamel (1995). 'Adaptive filtering in subbands using a weighted criterion,' in *Proc. IEEE ICASSP'95*, Detroit, MI, May, vol. 2, pp. 985–988.

De Léon, II, P. L. and D. M. Etter (1995). 'Experimental results with increased bandwidth analysis filters in oversampled, subband acoustic echo cancellers,' *IEEE Signal Process. Lett.*, **2**, no. 1, 1–3.

Dembo, A. and J. Salz (1990). 'On the least squares tap adjustment algorithm in adaptive digital echo cancellers,' *IEEE Trans. Commun.*, **COM-38**, no. 5, 622–628.

Demoment, G. and R. Reynaud (1985). 'Fast minimum variance deconvolution,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-33**, no. 4, 1324-1326.

Douglas, S. C. (1997). 'Performance comparison of two implementations of the leaky LMS adaptive filter,' *IEEE Trans. Signal Process.*, **45**, no. 8, 2125–2129.

Duttweiler, D. L. (1982). 'Adaptive filter performance with nonlinearities in the correlation multiplier,' *IEEE Trans. Acoustic, Speech and Signal Process.*, **ASSP-30**, no. 4, 578–586.

Egelmeers, G. P. M. and P. C. W. Sommen (1996). 'A new method for efficient convolution in frequency domain by nonuniform partitioning for adaptive filtering,' *IEEE Trans. Signal Process.*, **44**, no. 12, 3123–3129.

Eleftheriou, E. and D. D. Falconer (1986).'Tracking properties and steady-state performance of RLS adaptive filter algorithms,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-34**, no. 5, 1097–1109.

Eleftheriou, E. and D. D. Falconer (1987). 'Adaptive equalization techniques for HF channels,' *IEEE J. Selected Areas in Commun.*, **SAC-5**, 238–247.

Elliott, D. F. and K. R. Rao (1982). *Fast Transforms: Algorithms, Analysis, Applications.* Academic Press, New York.

Elliott, S. J. and B. Rafaely (1997). 'Rapid frequency-domain adaptation of causal FIR filters,' *IEEE Signal Process. Lett.*, **4**, no. 12, 337–339.

Eneman, K. and M. Moonen (1997). 'A relation between subband and frequency-domain adaptive filtering,' *IEEE Int. Conf. on Digital Signal Process.*, **1**, 25–28.

Ersoy, O. K. (1997). *Fourier-Related Transforms, Fast Algorithms and Applications.* Prentice-Hall PTR, Upper Saddle River, NJ.

Esterman, P. and A. Kaelin (1994). 'Analysis of the transient behavior of unconstrained frequency domain adaptive filters,' *IEEE Int. Symp. on Circuits and Syst.*, **2**, 21–24.

Eweda, E. (1990a). 'Analysis and design of a signed regressor LMS algorithm for stationary and nonstationary adaptive filtering with correlated Gaussian data,' *IEEE Trans. Circuits and Syst.*, **CAS-37**, no. 11, 1367–1374.

Eweda, E. (1990b). 'Optimum step size of sign algorithm for nonstationary adaptive filtering,' *IEEE Trans. Acoustic, Speech and Signal Process.*, **ASSP-38**, no. 11, 1897–1901.

Eweda, E. (1994). 'Comparison of RLS, LMS, and sign algorithms for tracking randomly time-varying channels,' *IEEE Trans. Signal Process.*, **42**, no. 11, 2937–2944.

Eweda, E. (1997). 'Tracking analysis of sign-sign algorithm for nonstationary adaptive filtering with Gaussian data,' *IEEE Trans. Signal Process.*, **45**, no. 5, 1375–1378.

Eweda, E. and O. Macchi (1985). 'Tracking error bounds of adaptive nonstationary filtering,' *Automatica*, **21**, 293–302.

Eweda, E. and O. Macchi (1987). 'Convergence of the RLS and LMS adaptive filters,' *IEEE Transactions on Circuits and Syst.*, **34**, no. 7, 799–803.

Falconer, D. D. and L. Ljung (1978). 'Application of fast Kalman estimation to adaptive equalization,' *IEEE Trans. Commun.*, **COM-26**, no. 10, 1439–1446.

Farden, D. C. (1981). 'Tracking properties of adaptive signal processing algorithms,' *IEEE Trans. Acoustic, Speech and Signal Process.*, **ASSP-29**, no. 3, 439–446.

Farhang-Boroujeny, B. (1993). 'Application of orthonormal transforms to implementation of quasi-LMS/Newton algorithm,' *IEEE Trans. Signal Process.*, **41**, no. 3, 1400–1405.

Farhang-Boroujeny, B. (1994). 'Variable-step-size LMS algorithm: new developments and experiments,' *IEE Proc. – Vis. Image and Signal Process.*, **141**, no. 5, 311–317.

Farhang-Boroujeny, B. (1996a). 'Performance of LMS-based adaptive filters in tracking a time-varying plant,' *IEEE Trans. Signal Process.*, **44**, no. 11, 2868–2871.

Farhang-Boroujeny, B. (1996b). 'Analysis and efficient implementation of partitioned block LMS adaptive filters,' *IEEE Trans. Signal Process.*, **44**, no. 11, 2865–2868.

Farhang-Boroujeny, B. (1996c). 'Channel equalization via channel identification: algorithms and simulation results for rapidly fading HF channel,' *IEEE Trans. Commun.*, **44**, no. 11, 1409–1412.

Farhang-Boroujeny, B. (1997a). 'An IIR adaptive line enhancer with controlled bandwidth,' *IEEE Trans. Signal Process.*, **45**, no. 2, 477–481.

Farhang-Boroujeny, B. (1997b). 'Fast LMS/Newton algorithms based on autoregressive modeling and their application to acoustic echo cancellation,' *IEEE Trans. Signal Process.*, **45**, no. 8, 1987–2000.

Farhang-Boroujeny, B. and S. Gazor (1991). 'Performance analysis of transform domain normalized LMS Algorithm,' in *Proc. ICASSP'91 Conf.*, Toronto, Canada, pp. 2133–2136.

Farhang-Boroujeny, B. and S. Gazor (1992). 'Selection of orthonormal transforms for improving the performance of the transform domain normalized LMS Algorithm,' *IEE Proceedings*, part F, **139**, no. 5, 327–335.

Farhang-Boroujeny, B. and Y. C. Lim (1992). 'A comment on the computational complexity of sliding FFT,' *IEEE Trans. Circuits and Syst.*, **39**, 875–876.

Farhang-Boroujeny, B. and S. Gazor (1994). 'Generalized sliding FFT and its application to implementation of block LMS adaptive filters,' *IEEE Trans. Signal Process.*, **42**, no. 3, 532–538.

Farhang-Boroujeny, B. and Z. Wang (1995). 'A modified subband adaptive filtering for acoustic echo cancellation,' in *Proc. Int. Conf. Signal Process. App. & Tech.*, Boston, MA, Oct., pp. 74–78.

Farhang-Boroujeny, B., Y. Lee and C. C. Ko (1996). 'Sliding transforms for efficient implementation of transform domain adaptive filters,' *Signal Process.*, **52**, 83–96.

Farhang-Boroujeny, B. and Z. Wang (1997). 'Adaptive filtering in subbands: design issues and experimental results for acoustic echo cancellation,' *Signal Process.*, **61**, 213–223.

Fechtel, S. A. and H. Meyr (1991). 'An investigation of channel estimation and equalization techniques for moderately rapid fading HF-channels,' *ICC'91 Conference Record*, Sheraton-Denver Technological Center, Denver, June 23–26, vol. 2, pp. 25.2.1–25.2.5.

Fechtel, S. A. and H. Meyr (1992). 'Optimal feedforward estimation of frequency-selective fading radio channels using statistical channel information,' *ICC'92 Conference Record*, Chicago, IL, June 14–18, vol. 2, pp. 677–681.

Ferrara, E. R. (1980). 'Fast implementation of LMS adaptive filters,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-28**, no. 6, 474–475.

Ferrara, E. R. and B. Widrow (1981). 'The time-sequenced adaptive filter,' *IEEE Trans. Circuits and Syst.*, **CAS-28**, 519–523.

Fertner, A. (1997). 'Frequency-domain echo canceller with phase adjustment,' *IEEE Trans. Circuits and Syst. II: Analog and Digital Signal Process.*, **44**, no. 10, 835–841.

Feuer, A. and E. Weinstein (1985). 'Convergence analysis of LMS filters with uncorrelated Gaussian data,' *IEEE Trans. Acoustic, Speech and Signal Process.*, **ASSP-33**, no. 1, 222–230.

Frost III, O. L. (1972). 'An algorithm for linearly constrained adaptive array processing,' *Proc. IEEE*, **60**, no. 8, 926–935.

Furukawa, I. (1984). 'A design of canceller of broad band acoustic echo,' *Int. Teleconference Symposium*, Tokyo, pp. 1/8–8/8.

Garayannis, G., D. Manolakis and N. Kalouptsidis (1983). 'A fast sequential algorithm for least squares filtering and prediction,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-31**, no. 6, 1394–1402.

Gardner, W. A. (1987). 'Nonstationary learning characteristics of the LMS algorithm,' *IEEE Trans. Circuits and Syst.*, **CAS-34**, no. 10, 1199–1207.

Gazor, S. and B. Farhang-Boroujeny (1992). 'Quantization effects in transform domain normalized LMS algorithm,' *IEEE Trans. Circuits and Syst. II: Analog and Digital Signal Process.*, **39**, no. 1, 1–7.

Gilloire, A. and M. Vetterli (1992). 'Adaptive filtering in subbands with critical sampling: analysis, experiments, and application to acoustic echo cancellation,' *IEEE Trans. Signal Process.*, **40**, no. 8, 1862–1875.

Gitlin, R. D. and S. B. Weinstein (1981). 'Fractionally-spaced equalization: an improved digital transversal equalizer,' *Bell Syst. Tech. J.*, **60**, no. 2, 275–296.

Gitlin, R. D., Hayes, J. F. and S. B. Weinstein (1992). *Data Communications Principles*. Plenum Press, New York.

Glentis, G. O. and N. Kalouptsidis (1992a). 'Efficient order recursive algorithms for multichannel least-squares filtering,' *IEEE Trans. Signal Process.*, **40**, no. 7, 1354–1374.

Glentis, G. O. and N. Kalouptsidis (1992b). 'Fast adaptive algorithms for multichannel filtering and system identification,' *IEEE Trans. Signal Process.*, **40**, no. 10, 2433–2458.

Golub, G. H. and C. F. Van Loan (1989). *Matrix Computations*, 2nd edn. The John Hopkins University Press, Baltimore.

Goodwin, G. C. and K. S. Sin (1984). *Adaptive Filtering, Prediction and Control*. Prentice-Hall, Englewood Cliffs, NJ.

Gray, R. M. (1972). 'On the asymptotic eigenvalue distribution of toeplitz matrices,' *IEEE Trans. Inform. Theory*, **IT-18**, no. 6, 725–730.

Griffiths, L. J. (1969). 'A simple adaptive algorithm for real-time processing in antenna arrays,' *Proc. IEEE*, **57**, no. 10, 1696–1704.

Griffiths, L. J. (1978). 'An adaptive lattice structure for noise-cancelling applications,' *ICASSP Conf.*, Tulsa, OK, pp. 87–90.

Griffiths, L. J. and C. W. Jim (1982). 'An alternative approach to linearly constrained adaptive beamforming,' *IEEE Trans. Antennas Propag.*, **AP-30**, no. 1, 27–34.

Hajivandi, M. and W. A. Gardner (1990). 'Measures of tracking performance for the LMS algorithm,' *IEEE Trans. Acoustics, Speech and Signal Process.*, **ASSP-38**, no. 11, 1953–1958.

Harris, R. W., D. M. Chabries and F. A. Bishop (1986). 'A variable step (VS) adaptive filter algorithm,' *IEEE Trans. Acoustics, Speech and Signal Process.*, **ASSP-34**, no. 2, 309–316.

Hassibi, B., A. H. Sayed and T. Kailath (1996). 'H^∞ optimality of the LMS algorithm,' *IEEE Trans. Signal Process.*, **44**, no. 2, 267–280.

Haykin, S. (1991). *Adaptive Filter Theory*, 2nd edn. Prentice-Hall, Englewood Cliffs, NJ.

Haykin, S. (1996). *Adaptive Filter Theory*, 3rd edn. Prentice-Hall, Upper Saddle River, NJ.

Hirsch, D. and W. J. Wolf (1970). 'A simple adaptive equalizer for efficient data transmission,' *IEEE Trans. Commun. Tech.*, **COM-18**, no. 1, pp. 5–12.

Honig, M. L. and D. G. Messerschmitt (1981). 'Convergence properties of an adaptive digital lattice filter,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-29**, no. 3, 642–653.

Honig, M. L. and D. G. Messerschmitt (1984). *Adaptive filters: structures, algorithms, and applications*. Kluwer Academic, Boston, MA.

Horowitz, L. L. and K. D. Senne (1981). 'Performance advantage of complex LMS for controlling narrow-band adaptive arrays,' *IEEE Trans. Acoustic, Speech and and Signal Process.*, **ASSP-29**, no. 3, 722–736.

Houacine, A. (1991). 'Regularized fast recursive least squares algorithms for adaptive filtering,' *IEEE Trans. Signal Process.*, **39**, no. 4, 860–871.

Householder, A. S. (1964). *Theory of Matrices in Numerical Analysis*. Blaisdell, New York.

Hsu, F. M. (1982). 'Square root Kalman filtering for high-speed data received over fading dispersive HF channels,' *IEEE Trans. Inform. Theory*, **IT-28**, no. 5, 753–763.

Hush, D. R., N. Ahmed, R. David and S. D. Stearns (1986). 'An adaptive IIR structure for sinusoidal enhancement, frequency estimation, and detection,' *IEEE Trans. Acoustic, Speech and Signal Process.*, **ASSP-34**, no. 6, 1380–1390.

Itakura, F. and S. Saito (1971). 'Digital filtering techniques for speech analysis and synthesis,' *Proc. 7th Int. Conf. on Acoust.*, vol. 3, paper 25C-1, pp. 261–264.

ITU-T Recommendation G.167 (03/93), *Acoustic Echo Controllers*. ITU, 1993.

Iyer, U., M. Nayeri and H. Ochi (1994). 'A poly-phase structure for system identification and adaptive filtering,' in *Proc. IEEE ICASSP'94*, Adelaide, South Australia, vol. III, pp. 433–436.

Jaggi, S. and A. B. Martinez (1990). 'Upper and lower bounds of the misadjustment in the LMS algorithm,' *IEEE Trans. Acoustic, Speech and Signal Process.*, **ASSP-38**, no. 1, 164–166.

Jain, A. K. (1976). 'A fast Karhunen-Loeve transform for a class of random processes,' *IEEE Trans. Commun.*, **COM-24**, 1023–1029.

Jayant, N. S. and P. Noll (1984). *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Prentice-Hall, Englewood Cliffs, NJ.

Jeyendran, B. and V. U. Reddy (1990). 'Recursive system identification in the presence of burst disturbance', *Signal Process.*, **20**, 227–245.

Jim, C. W. (1977). A comparison of two LMS constrained optimal array structures, *Proc. IEEE*, **65**, no. 12, 1730–1731.

Johnson, D. H. and D. E. Dudgeon (1993). *Array Signal Processing: Concepts and Techniques*. Prentice-Hall, Englewood Cliffs, NJ.

Kalouptsidis, N. and S. Theodoridis (Eds.) (1993). *Adaptive System Identification and Signal Processing Algorithms*. Prentice-Hall, U.K.

Karaboyas, S. and N. Kalouptsidis (1991). 'Efficient adaptive algorithms for ARX identification,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-39**, no. 3, pp. 571–582.

Karaboyas, S., N. Kalouptsidis and C. Caroubalos (1989). 'Highly parallel multichannel LS algorithms and application to decision feedback equalizers,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-37**, no. 9, 1380–1396.

Kay, S. (1988). *Advanced Topics in Signal Processing*, J. S. Lim and A. V. Oppenheim (Eds.). Prentice-Hall, Englewood Cliffs, NJ., Chapter 2.

Kellermann, W. (1984). 'Kompensation akustischer echos in frequenzteilbändern,' in *Aachener Kolloquium*, Aachen, FRG, pp. 322–325 (in German).

Kellermann, W. (1985). 'Kompensation akustischer echos in frequenzteilbändern,' in *Frequenz*, **39**, no. 7/8, 209–215.

Kellermann, W. (1988). 'Analysis and design of multirate systems for cancellation of acoustical echoes,' in *Proc. IEEE ICASSP'88*, New York, pp. 2570–2573.

Kuo, S. M. and D. R. Morgan (1996). *Active Noise Control Systems, Algorithms and DSP Implementations*. John Wiley & Sons, New York.

Kushner, H. J. (1984). *Approximation and Weak Convergence Methods for Random Processes with Applications to Stochastic Systems Theory*, MIT Press, Cambridge, MA.

Kwong, C. P. (1986). 'Dual sign algorithm for adaptive filtering,' *IEEE Trans. Commun.*, **COM-34**, no. 12, 1272–1275.

Lee, D. T., M. Morf and B. Friedlander (1981). 'Recursive least square ladder filter algorithms,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-29**, no. 3, 627–641.

Lee, E. A. and D. G. Messerschmitt (1994). *Digital Communication*, 2nd edn., Kluwer, Boston.

Lee, J. C. and C. K. Un (1984). 'A reduced structure of the frequency domain block LMS adaptive digital filter,' *Proc. IEEE*, **72**, no. 12, 1816–1818.

Lee, J. C. and C. K. Un (1986). 'Performance of transform-domain LMS adaptive digital filters,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-34**, no. 3, 499–510.

Lee, J. C. and C. K. Un (1989). 'Performance analysis of frequency domain block LMS adaptive digital filters,' *IEEE Trans. Circuits and Syst.*, **36**, no. 2, 173–189.

Lev-Ari, H. (1987). 'Modular architecture for adaptive multichannel lattice algorithms,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-35**, no. 4, 543–552.

Lev-Ari, H., T. Kailath and J. Cioffi (1984). 'Least-squares adaptive lattice and transversal filters: a unified geometric theory,' *IEEE, Trans. Inf. Theory*, **IT-30**, no. 2, 222–236.

Levinson, N. (1946). 'The Wiener r.m.s. (root-mean-square) error criterion in filter design and prediction,' *J. Math. Phys.*, **25**, 261–278.

Lewis, P. S. (1990). 'QR-based algorithms for multichannel adaptive least squares lattice filters,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-38**, no. 3, 421–432.

Li, G. (1997). 'A stable and efficient adaptive notch filter for direct frequency estimation,' *IEEE Trans. Signal Process.*, **45**, no. 8, 2001–2009.

Li, X. and W. K. Jenkins (1996). 'The comparison of the constrained and unconstrained frequency-domain block-LMS adaptive algorithms,' *IEEE Trans. Signal Process.*, **44**, no. 7, 1813–1816.

Lim, Y. C. and B. Farhang-Boroujeny (1992). 'Fast filter bank (FFB),' *IEEE Trans. Circuits and Syst. II: Analog and Digital Signal Process.*, **39**, no. 5, 316–318.

Lin, J., J. G. Proakis, F. Ling and H. Lev-Ari (1995). 'Optimal tracking of time-varying channels: a frequency domain approach for known and new algorithms,' *IEEE Journal Selected Area in Commun.*, **13**, no. 1, 141–154.

Ling, F. (1991). 'Givens rotation based least squares lattice and related algorithms,' *IEEE Trans. Signal Process.*, **39**, no. 7, 1541–1551.

Ling, F. (1993). *Adaptive System Identification and Signal Processing Algorithms*, N. Kalouptsidis and S. Theodoridis (Eds.). Prentice-Hall, U.K., Chapter 6.

Ling, F. and J. G. Proakis (1984a). 'Nonstationary learning characteristics of least squares adaptive estimation algorithms,' in *Proc. ICASSP'84*, San Diego, Calif., pp. 3.7.1–3.7.4.

Ling, F. and J. G. Proakis (1984b). 'A generalized multichannel least squares lattice algorithm based on sequential processing stages,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-32**, no. 2, 381–389.

Ling, F. and J. G. Proakis (1985). 'Adaptive lattice decision-feedback equalizers – their performance and application to time-variant multipath channels,' *IEEE Trans. Commun.*, **COM-33**, no. 4, 348–356.

Ling, F., D. G. Manolakis and J. G. Proakis (1986a). 'Flexible, numerically robust array processing algorithm and its relationship to the Givens transformation,' *Proc. IEEE Int. Conf. on ASSP*, Tokyo, Japan, April.

Ling, F., D. G. Manolakis and J. G. Proakis (1986b). 'A recursive modified Gram-Schmidt algorithm for least-squares estimation,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-34**, no. 4, 829–836.

Ling, F., D. G. Monolakis and J. G. Proakis (1986c). 'Numerically robust least-squares lattice ladder algorithms with direct updating of the reflection coefficients,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-34**, no. 4, 837–845.

Liu, J. C. and T. P. Lin (1988). 'Running DHT and real-time DHT analyzer,' *Electron. Lett.*, **24**, no. 12, 762–763.

Ljung, L. and T. Söderström (1983). *Theory and Practice of Recursive Identification*. MIT Press, Cambridge, MA.

Ljung, S. and L. Ljung (1985). 'Error propagation properties of recursive least-squares adaptation algorithms,' *Automatica*, **21**, no. 2, 157–167.

Ljung, L., M. Morf and D. Falconer (1978). 'Fast calculation of gain matrices for recursive estimation schemes,' *Int. J. Control*, **27**, 1–19.

Long, G., D. Shwed and D. D. Falconer (1987). 'Study of a pole-zero adaptive echo canceller,' *IEEE Transactions on Circuits and Syst.*, **34**, no. 7, 765–769.

Long, G., F. Ling and J. G. Proakis (1989). 'The LMS algorithm with delayed coefficient adaptation,' *IEEE Trans. Acoustics, Speech and Signal Process.*, **ASSP-37**, no. 9, 1397–1405.

Long, G. and F. Ling (1993). 'Fast initialization of data-deriven Nyquist in-band echo cancellers,' *IEEE Trans. Commun.*, **41**, no. 6, 893–904.

Macchi, O. (1986). 'Optimization of adaptive identification for time-varying filters,' *IEEE Trans. Automatic Control*, **AC-31**, no. 3, 283–287.

Macchi, O. (1995). *Adaptive Processing, the Least Mean Squares Approach with Applications in Transmission*. John Wiley & Sons, Chichester, U.K.

Macchi, O. M. and N. J. Bershad (1991). 'Adaptive recovery of a chirped sinusoid in noise, Part 1: performance of the RLS algorithm,' *IEEE Trans. Signal Process.*, **ASSP-39**, no. 5, 583–594.

Makhoul, J. (1975). 'Linear prediction: a tutorial review,' *Proc. IEEE*, **63**, no. 4, 561–580.

Makhoul, J. (1978). 'A class of all-zero lattice digital filters: properties and application,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-26**, no. 4, 304–314.

Makhoul, J. and R. Viswanathan (1978). 'Adaptive lattice methods for linear prediction,' in *Proc. ICASSP-78 Conf.*, pp. 87–90.

Mavridis, P. P. and G. V. Moustakides (1996). 'Simplified Newton-type adaptive estimation algorithms,' *IEEE Trans. Signal Process.*, **44**, no. 8, 1932–1940.

Mansour, D. and A. H. Gray, Jr (1982). 'Unconstrained frequency-domain adaptive filter,' *IEEE Trans. Acoust. Speech and Signal Process.*, **ASSP-30**, no. 5, 726–734.

Markel, J. D. (1971). 'FFT pruning,' *IEEE Trans. Audio Electroacoustics*, **AU-19**, no. 4, 305–311.

Marshall, D. F., W. K. Jenkins and J. J. Murphy (1989). 'The use of orthogonal transforms for improving performance of adaptive filters,' *IEEE Trans. Circuits and Syst.*, **CAS-36**, no. 4, 474–483.

Marshall, D. F. and W. K. Jenkins (1992). 'A fast quasi-Newton adaptive filtering algorithm,' *IEEE Trans. Signal Process.*, **40**, no. 7, 1652–1662.

Mathew, G., B. Farhang-Boroujeny and R. W. Wood (1997). 'Design of multilevel decision feedback equalizers,' *IEEE Trans. Magnetics*, **33**, no. 6, 4528–4542.

Mathew, G., V. U. Reddy and S. Dasgupta (1995). 'Adaptive estimation of eigensubspace,' *IEEE Trans. Signal Process.*, **43**, no. 2, 401–411.

Mathews, V. J. and S. H. Cho (1987). 'Improved convergence analysis of stochastic gradient adaptive filters using the sign algorithm,' *IEEE Trans. Acoustic, Speech and Signal Process.*, **ASSP-35**, no. 4, 450–454.

Mathews, V. J. and Z. Xie (1990). 'Stochastic gradient adaptive filters with gradient adaptive step sizes,' in *Proc. ICASSP'90, Conf.*, Albuquerque, NM, pp. 1385–1388.

Mathews, V. J. and Z. Xie (1993). 'A stochastic gradient adaptive filter with gradient adaptive step size,' *IEEE Trans. Signal Process.*, **SP-41**, no. 6, 2075–2087.

Mayyas, K. and T. Aboulnasr (1997). 'Leaky LMS algorithm: MSE analysis for Gaussian data,' *IEEE Trans. Signal Process.*, **45**, no. 4, 927–934.

Mazo, J. E. (1979). 'On the independence theory of equalizer convergence,' *Bell Syst. Tech. J.*, **58**, no. 5, 963–993.

McLaughlin, H. J. (1996). 'System and method for an efficiently constrained frequency-domain adaptive filter,' US Patent, no. 5,526,426, Date of Patent: June 11, 1996.

Mikhael, W. B., F. H. Wu, L. G. Kazovsky, G. S. Kang and L. J. Fransen (1986). 'Adaptive filters with individual adaptation of parameters,' *IEEE Trans. Circuits and Syst.*, **CAS-33**, no. 7, 677–686.

Montazeri, M. and P. Duhamel (1995). 'A set of algorithms linking NLMS and block RLS algorithms,' *IEEE Trans. Signal Process.*, **43**, no. 2, 444–453.

Moonen, M. and J. Vandewalle (1990). 'Recursive least square with stabilized inverse factorization,' *IEEE Trans. Signal Process.*, **21**, no. 1, 1–15.

Morf, M. and T. Kailath (1975). 'Square-root algorithms for least-squares estimation,' *IEEE Trans. Autom. Control*, **AC-20**, no. 4, 487–497.

Morf, M., B. Dickinson, T. Kailath and A. Vieira (1977). 'Efficient solution of covariance equations for linear prediction,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-25**, 423–433.

Morgan, D. R. (1995). 'Slow asymptotic convergence of LMS acoustic echo cancellers,' *IEEE Trans. Speech Audio Process.*, **3**, no. 2, 126–136.

Morgan, D. R. and J. C. Thi (1995). 'A delayless subband adaptive filter architecture,' *IEEE Trans. Signal Process.*, **43**, no. 8, 1819–1830.

Morgan, D. R. and S. G. Kratzer (1996). 'On a class of computationally efficient, rapidly converging, generalized NLMS algorithms,' *IEEE Signal Process. Lett.*, **3**, no. 8, 245–247.

Moulines, E., O. A. Amrane and Y. Grenier (1995). 'The generalized multidelay adaptive filter: structure and convergence analysis,' *IEEE Trans. Signal Process.*, **43**, no. 1, 14–28.

Moustakides, G. V. (1997). 'Study of the transient phase of the forgetting factor RLS,' *IEEE Trans. Signal Process.*, **45**, no. 10, 2468–2476.

Moustakides, G. V. and S. Theodoridis (1991). 'Fast Newton transversal filters – a new class of adaptive estimation algorithms,' *IEEE Trans. Signal Process.*, **39**, no. 10, 2184–2193.

Mueller, K. H. (1973). 'A new approach to optimum pulse shaping in sampled systems using time-domain filtering,' *Bell Syst. Tech. J.*, **52**, no. 5, 723–729.

Murthy, N. R. and M. N. S. Swamy (1992). 'On the computation of running discrete cosine and sine transforms,' *IEEE Trans. Signal Process.*, **40**, no. 6, 1430–1437.

Narayan, S. S. and A. M. Peterson (1981). Frequency domain LMS algorithm,' *Proc. IEEE*, **69**, no. 1, 124–126.

Narayan, S. S., A. M. Peterson and M. J. Narasimha (1983). 'Transform domain LMS algorithm,' *IEEE Trans. Acoustics, Speech and Signal Process.*, **ASSP-31**, no. 3, 609–615.

Nayebi, K., T. P. Barnwell III and M. J. T. Smith (1994). 'Low delay FIR filter banks: design and evaluation,' *IEEE Trans. Signal Process.*, **42**, no. 1, 24–31.

Nitzberg, R. (1985). 'Application of the normalized LMS algorithm to MSLC,' *IEEE Trans. Aerospace and Electronic Syst.*, **AES-21**, no. 1, 79–91.

Ogunfunmi, A. O. and A. M. Peterson (1992). 'On the implementation of the frequency-domain LMS adaptive filter,' *IEEE Trans. Circuits and Syst. II: Analog and Digital Signal Process.*, **39**, no. 5, 318–322.

Oppenheim, A. V. and R. W. Schafer (1975). *Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Oppenheim, A. V. and R. W. Schafer (1989). *Discrete-Time Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Oppenheim, A. V., A. S. Willsky and I. T. Young (1983). *Signals and Systems*. Prentice-Hall, Englewood Cliffs, NJ.

Papoulis, A. (1991). *Probability, Random Variables, and Stochastic Processes*, 3rd edn. McGraw-Hill, Singapore.

Perkins, F. A. and D. D. McRae (1982). 'A high performance HF modem,' (Harris Corp., April 1982), presented at the *Int. Defense Electron.* Expo, Hanover, West Germany, May.

Petraglia, M. R. and S. K. Mitra (1991). 'Generalized fast convolution implementations of adaptive filters,' *IEEE Symposium Circuits and Syst.*, **5**, 2916–2919.

Petraglia, M. R. and S. K. Mitra (1993). 'Adaptive FIR filter structure based on the generalized subband decomposition of FIR filters,' *IEEE Trans. Circuits and Syst. II: Analog and Digital Signal Process.*, **40**, no. 6, 354–362.

Petillon, T., A. Gilloire and S. Theodoridis (1994). 'The fast Newton transversal filter: an efficient scheme for acoustic echo cancellation in mobile radio,' *IEEE Trans. Signal Process.*, **42**, no. 3, 509–518.

Picchi, G. and G. Prati (1994). 'Self-orthogonalizing adaptive equalization in the discrete frequency domain,' *IEEE Trans. Commun.*, **COM-32**, no. 4, 371 379.

Proakis, J. G. (1995). *Digital Communications*, 3rd edn. McGraw-Hill, New York.

Proakis, J. G., C. Rader, F. Ling and C. Nikias (1992). *Advanced Digital Signal Processing*. Macmillan, New York.

Proudler, I. K., J. G. McWhirter and T. J. Shepherd (1990). 'The QRD-based least square lattice algorithm: some computer simulation using finite wordlengths,' *Proc. IEEE Int. Symp. on Circuits and Systems*, New Orleans, LA, May, pp. 258–261.

Proudler, I. K., J. G. McWhirter and T. J. Shepherd (1991). 'Computationally efficient QR decomposition approach to least squares adaptive filtering,' *IEE Proc.*, part F, **138**, no. 4, 341–353.

Qureshi, S. U. H. (1985). 'Adaptive equalization,' *Proc. IEEE*, **73**, no. 9, 1349–1387.

Rabiner, L. R. and B. Gold (1975). *Theory and Application of Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Rabiner, L. R. and R. W. Schafer (1978). *Digital Processing of Speech Signals*. Prentice-Hall, Englewood Cliffs, NJ.

Reddi, S. S. (1984). 'A time-domain adaptive algorithm for rapid convergence,' *Proc. IEEE*, **72**, no. 4, 533 535.

Reddy, V. U., B. Egardt and T. Kailath (1981). 'Optimized lattice-form adaptive line enhancer for a sinusoidal signal in broad-band noise,' *IEEE Trans. Circuits and Syst.*, **CAS-28**, no. 6, 542–550.

Reddy, V. U., A. Paulraj and T. Kailath (1987). 'Performance analysis of the optimum beamformer in the presence of correlated sources and its behavior under spatial smoothing,' *IEEE Trans. Acoust. Speech and Signal Process.*, **ASSP-35**, no. 7, 927–936.

Regalia, P. A. (1991). 'An improved lattice-based adaptive IIR notch filter,' *IEEE Trans. Signal Process.*, **39**, no. 9, 2124–2128.

Regalia, P. A. and M. G. Bellanger (1991). 'On the duality between fast QR methods and lattice methods in least squares adaptive filtering,' *IEEE Trans. Signal Process.*, **39**, no. 4, 879–891.

Rockafellar R. T. (1970). *Convex Analysis*. Princeton University Press, Princeton, NJ.

Samson, C. and V. U. Reddy (1983). 'Fixed-point error analysis of normalized ladder algorithm,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-31**, no. 5, 1177–1191.

Satorius, E. H. and S. T. Alexander (1979). 'Channel equalization using adaptive lattice algorithms,' *IEEE Trans. Commun.*, **COM-27**, no. 6, 899–905.

Satorius, E. H. and J. D. Pack (1981). 'Application of least-squares lattice algorithms to adaptive equalization,' *IEEE Trans. Commun.*, **COM-29**, no. 2, 136 142.

Schobben, D. W. E., G. P. M. Egelmeers and P. C. W. Sommen (1997). 'Efficient realization of the block frequency domain adaptive filter,' in *Proc. ICASSP'97, Conf.*, vol. 3, pp. 2257–2260.

Sethares, W. A., I. M. Y. Mareels, B. D. O. Anderson, C. R. Johnson, Jr and R. R. Bitmead (1988). 'Excitation conditions for signed regressor least mean squares adaptation,' *IEEE Trans. Circuits and Syst.*, **CAS-35**, no. 6, pp. 613–624.

Sharma, R., W. A. Sethares and J. A. Bucklew (1996). 'Asymptotic analysis of stochastic gradient-based adaptive filtering algorithms with general cost functions,' *IEEE Trans. Signal Process.*, **44**, no. 9, 2186–2194.

Shynk, J. J. (1989). 'Adaptive IIR filtering,' *IEEE ASSP Mag.*, **5**, April, 4–21.

Shynk, J. J. (1992). 'Frequency-domain and multirate adaptive filtering,' *IEEE Signal Process. Mag.*, **9**, no. 1, 14–37.

Sidhu, G. S. and T. Kailath (1974). 'Development of new estimation algorithms by innovations analysis and shift-invariance properties,' *IEEE Trans. Inf. Theory*, **IT-20**, no. 6, 759–762.

Slock, D. T. M. (1991). 'Fractionally-spaced subband and multiresolution adaptive filters,' in *Proc. IEEE ICASSP'91*, pp. 3693–3696.

Slock, D. T. M. (1993). 'On the convergence behaviour of the LMS and the normalized LMS algorithms,' *IEEE Trans. on Signal Process.*, **41**, no. 9, 2811–2825.

Slock, D. T. M. and T. Kailath (1988). 'Numerically stable fast recursive least-squares transversal filters,' in *Proc. ICASSP-88 Conf.*, vol. 3, pp. 1365–1368.

Slock, D. T. M. and T. Kailath (1991). 'Numerically stable fast transversal filters for recursive least-squares adaptive filtering,' *IEEE Trans. Signal Process.*, **39**, no. 1, 92–114.

Slock, D. T. M. and T. Kailath (1992). 'A modular multichannel multiexperiment fast transversal filter RLS algorithm,' *Signal Process.*, **28**, 25–45.

Slock, D. T. M., L. Chisci, H. Lev-Ari and T. Kailath (1992). 'Modular and numerically stable fast transversal filters for multichannel and multiexperiment RLS,' *IEEE Trans. Signal Process*, **ASSP-40**, no. 4, 784–802.

Slock, D. T. M. and T. Kailath (1993). *Adaptive System Identification and Signal Processing Algorithms*, N. Kalouptsidis and S. Theodoridis (Eds.). Prentice-Hall, U.K., Chapter 5.

Solo, V. (1992). 'The error variance of LMS with time-varying weights,' *IEEE, Trans. Signal Process.*, **40**, no. 4, 803–813.

Somayazulu, V. S., S. K. Mitra and J. J. Shynk (1989). 'Adaptive line enhancement using multirate techniques,' in *Proc. IEEE ICASSP'89, Conf.*, Glasgow, Scotland, May, pp. 928–931.

Sommen, P. (1988). 'On the convergence properties of a partitioned block frequency domain adaptive filter (PBFDAF),' *Proc. EUSIPCO*, pp. 1401–1404.

Sommen, P. C. W. (1989). 'Partitioned frequency domain adaptive filters,' in *Proc. Asilomar Conf. Signals, Syst. and Computers*, Pacific Grove, CA, pp. 677–681.

Sommen, P. C. W. and E. de Wilde (1992). 'Equal convergence conditions for normal and partitioned-frequency domain adaptive filters,' in *Proc. ICASSP'92, Conf.*, vol. IV, pp. 69–72.

Sommen, P. C. W., P. J. Van Gerwen, H. J. Kotmans and A. J. E. M. Janssen (1987). 'Convergence analysis of a frequency domain adaptive filter with exponential power averaging and generalized window function,' *IEEE Trans. Circuits and Syst.*, **34**, no. 7, 788–798.

Sondhi, M. M. and W. Kellermann (1992). 'Adaptive echo cancellation for speech signals,' in *Advances in Speech Signal Processing*, S. Furui and M. M. Sondhi (Eds.). Marcel Dekker, New York, Chapter 11, pp. 327–356.

Soo, J. S. and K. K. Pang (1987). 'A new structure for block FIR adaptive digital filters,' in *IREECON. Int. Dig. Papers*, Sydney, Australia, pp. 364–367.

Soo, J. S. and K. K. Pang (1990). 'Multidelay block frequency domain adaptive filter,' *IEEE Trans. Acoust. Speech and Signal Process.*, **ASSP-38**, no. 2, 373–376.

Soumekh, M. (1994). *Fourier Array Imaging*. Prentice-Hall, Englewood Cliffs, NJ.

Stasinski, R. (1990). 'Adaptive Filters in Domains of Adaptive Transforms,' in *Proc. of Singapore ICCS'90, Conf.*, Singapore, November 5–9, pp. 18.2.1–18.2.5.

Stearns, S. D. (1981). 'Error surfaces of recursive adaptive filters', *IEEE Trans. Circuits and Syst.*, **CAS-28**, no. 6, 603–606.

Stein, S. (1987). 'Fading channels issues in system engineering,' *IEEE J. Selected Areas in Commun.*, **SAC-5**, 68–89, invited paper.

Strang, G. (1980). *Linear Algebra and Its Applications*, 2nd edn., Academic Press, New York.

Tarrab, M. and A. Feuer (1988). 'Convergence and performance analysis of the normalized LMS algorithm with uncorrelated Gaussian data,' *IEEE Trans. Infor. Theory*, **IT-34**, no. 4, 680–691.

Tanrikulu, O., B. Baykal, A. G. Constantinides and J. A. Chambers (1997). 'Residual echo signal in critically sampled subband acoustic echo cancellers based on IIR and FIR filter banks,' *IEEE Trans. Signal Process.*, **45**, no. 4, 901–912.

Tummala, M and S. R. Parker (1987). 'A new efficient adaptive cascade lattice structure,' *IEEE Trans. Circuits and Syst.*, **34**, no. 7, 707–711.

Vaidyanathan, P. P. (1993). *Multirate Systems and Filter Banks*. Prentice-Hall, Englewood Cliffs, NJ.

Vaidyanathan, P. P. and T. Q. Nguyen (1987). 'Eigenfilters: a new approach to least-squares FIR filter design and applications including Nyquist filters,' *IEEE Trans. Circuits and Syst.*, **CAS-34**, no. 1, 11–23.

van den Bos, A. (1994). 'Complex gradient and Hessian,' *IEE Proc. – Vis. Image Signal Process.*, **141**, no. 6, 380–382.

Verhaegen, M. H. (1989). 'Improved understanding of the loss-of-symmetry phenomenon in the conventional Kalman filter,' *IEEE Trans. Autom. Control*, **AC-34**, no. 3, 331–333.

Verhaegen, M. H. (1989). 'Round-off error propagation in four generally-applicable, recursive least-squares estimation schemes,' *Automatica*, **25**, no. 3, 437–444.

Verhoeckx, N. A. M. and T. A. C. M. Claasen (1984). 'Some considerations on the design of adaptive digital filters equipped with the sign algorithm,' *IEEE Trans. Acoustic, Speech and Signal Process.*, **ASSP-32**, 258–266.

von Zitzewitz, A. (1990). 'Considerations on acoustic echo cancelling based on realtime experiments,' in *Proc. IEEE EUSIPCO'90 V European Signal Process. Conference*, Barcelona, Spain, September, pp. 1987–1990.

Wang, Z. (1996). *Adaptive Filtering in Subband*. M.Eng. Thesis, Department of Electrical Engineering, National University of Singapore.

Ward, C. R., P. J. Hargrave and J. G. McWhirter (1986). 'A novel algorithm and architecture for adaptive digital beamforming,' *IEEE Trans. Antennas Propag.*, **AP-34**, no. 3, 338–346.

Wei, P. C., J. R. Zeidler and W. H. Ku (1997). 'Adaptive recovery of a chirped signal using the RLS algorithm,' *IEEE Trans. Signal Process.*, **45**, no. 2, 363–376.

Weiss, A. and D. Mitra (1979). 'Digital adaptive filters: Conditions for convergence, rates of convergence, effects of noise and errors arising from the implementation,' *IEEE Trans. Inf. Theory*, **IT-25**, no. 6, 637–652.

Widrow, B. and M. E. Hoff, Jr (1960). 'Adaptive switching circuits,' *IRE WESCON Conv. Rec.*, part 4, pp. 96–104.

Widrow, B. and S. D. Stearns (1985). *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Widrow, B. and E. Walach (1984). 'On the statistical efficiency of the LMS algorithm with nonstationary inputs,' *IEEE Trans. Inform. Theory*, **IT-30**, no. 2, 211–221.

Widrow, B., P. Baudrenghien, M. Vetterli and P. F. Titchener (1987). 'Fundamental relations between the LMS algorithm and the DFT,' *IEEE Trans. Circuits and Syst.*, **CAS-34**, 814–820.

Widrow, B., J. R. Glover, Jr, J. M. McCool, J. Kaunitz, C. S., Williams, R. H. Hearn, J. R. Zeidler, E. Dong, Jr and R. C. Goodlin (1975). 'Adaptive noise cancelling: principles and applications,' *Proc. IEEE*, **63**, no. 12, 1692–1716.

Widrow B., P. E. Mantey, L. J. Griffiths and B. B. Goode (1967). 'Adaptive antenna systems,' *Proc. IEEE*, **55**, no. 12, 2143–2159.

Widrow, B., McCool, J. and Ball, M (1975). 'The complex LMS algorithm,' *Proc. IEEE*, **63**, no. 4, 719–720.

Widrow, B., J. M. McCool, M. G. Larimore and C. R. Johnson, Jr (1976). 'Stationary and nonstationary learning characteristics of the LMS adaptive filter,' *Proc. IEEE*, **64**, no. 8, 1151–1162.

Yang, B. (1994). 'A note on error propagation analysis of recursive least-squares algorithms,' *IEEE Trans. Signal Process*, **42**, no. 12, 3523–3525.

Yasukawa H. and S. Shimada (1987). 'Acoustic echo canceler with high speech quality,' in *Proc. IEEE ICASSP'87*, Dallas, TX, pp. 2125–2128.

Yasukawa, H. and S. Shimada (1993). 'An acoustic echo canceler using subband sampling and decorrelation methods,' *IEEE Trans. Signal Process.*, **41**, no. 2, 926–930.

Yon, C. H. and C. K. Un (1992). 'Normalised frequency-domain adaptive filter based on optimum block algorithm,' *IEE Electron. Lett.*, **28**, no. 1, 11–12.

Yon, C. H. and C. K. Un (1994). 'Fast multidelay block transform-domain adaptive filters based on a two-dimensional optimum block algorithm,' *IEEE Trans. Circuits and Syst. II: Analog and Digital Signal Process.*, **41**, no. 5, 337–345.

Yuen, S., K. Abend and R. S. Berkowitz (1988). 'A recursive least-squares algorithm for multiple inputs and outputs and a cylindrical systolic implementation,' *IEEE Trans. Acoust., Speech and Signal Process.*, **ASSP-36**, no. 12, 1917–1923.

Index

Adaptive Filters

Theory and Applications

B. Farhang-Boroujeny
National University of Singapore

This enlightened engineering approach to the study of adaptive filters employs MATLAB® computer simulations to clarify theoretical results. A highly accessible text, *Adaptive Filters* elucidates the concept of convergence and provides many application examples. The comprehensive coverage includes the theory of Wiener filters, eigenanalysis, the complete family of LMS-based algorithms, recursive least-squares and a new treatment of tracking.

Features include:

- Accompanying diskette containing the MATLAB programs used throughout the book and providing an insight into adaptive filtering concepts
- End-of-chapter exercises designed to extend results developed in the text and to sharpen the reader's skill in theoretical development
- MATLAB-based simulation problems which will enhance understanding of the behaviour of different adaptive algorithms
- Thorough treatment of transform domain, frequency domain and subband adaptive filters
- Section on eigenanalysis presenting the essential mathematics for the study of filters

A valuable student resource and an essential technical reference for signal processing engineers in industry, *Adaptive Filters* presents a broad subject overview with emphasis on new developments and popular applications.

ISBN 0-471-98337-3

9 780471 983378

JOHN WILEY & SONS

hichester · New York · Weinheim · Brisbane · Singapore · Toronto

Farhang-Boroujeny

Adaptive Filt