ADAPTIVE FILTERING

Algorithms and Practical Implementation

THE KLUWER INTERNATIONAL SERIES
IN ENGINEERING AND COMPUTER SCIENCE

ADAPTIVE FILTERING

Algorithms and Practical Implementation

by

Paulo Sergio Ramirez Diniz
Federal University of Rio de Janeiro

Springer Science+Business Media, LLC

ISBN 978-1-4613-4660-9 ISBN 978-1-4419-8660-3 (eBook)
DOI 10.1007/978-1-4419-8660-3

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available
from the Library of Congress.

Originally published by Kluwer Academic Publishers in 1997
Softcover reprint of the hardcover 1st edition 1997

Printed on acid-free paper.

CONTENTS

PREFACE

The field of *Digital Signal Processing* has developed so fast in the last two decades that it can be found in the graduate and undergraduate programs of most universities. This development is related to the growing available technologies for implementing digital signal processing algorithms. The tremendous growth of development in the digital signal processing area has turned some of its specialized areas into fields themselves. If accurate information of the signals to be processed is available, the designer can easily choose the most appropriate algorithm to process the signal. When dealing with signals whose statistical properties are unknown, fixed algorithms do not process these signals efficiently. The solution is to use an adaptive filter that automatically changes its characteristics by optimizing the internal parameters. The adaptive filtering algorithms are essential in many statistical signal processing applications.

Although the field of adaptive signal processing has been subject of research for over three decades, it was in the eighties that a major growth occurred in research and applications. Two main reasons can be credited to this growth, the availability of implementation tools and the appearance of early textbooks exposing the subject in an organized form. Presently, there is still a lot of activities going on in the area of adaptive filtering. In spite of that, the theoretical development in the linear-adaptive-filtering area reached a maturity that justifies a text treating the various methods in a unified way, emphasizing the algorithms that work well in practical implementation. This text concentrates on studying on-line algorithms, those whose adaptation occurs whenever a new sample of the environment signals is available. The so-called block algorithms, those whose adaptation occurs when a new block of data is available, are not directly presented here in our view this subject requires a book for itself. Besides, block algorithms require implementation resources that are distinct of the on-line algorithms. The theory of nonlinear adaptive filters based on high-order

statistics is probably the most important complement to the subject treated in this book. Although this subject is not treated here, the understanding of the material presented is fundamental for studying this still growing field.

The idea of writing this book started while teaching the adaptive signal processing course at the graduate school of the Federal University of Rio de Janeiro (UFRJ). The request of the students to cover as many algorithms as possible made me think how to organize this subject such that not much time is lost in adapting notations and derivations related to different algorithms. Another common question was which algorithms really work in a finite-precision implementation. These issues made me believe that a new text on this subject could be written with these objectives in mind. Also, considering that most graduate and undergraduate programs include a single adaptive filtering course, this book should not be lengthy. Another objective to seek is to provide an easy access to the working algorithms for the practicing engineer.

It was not until I spent a sabbatical year and a half at University of Victoria, Canada, that this project actually started. In the leisure hours, I slowly started this project. Parts of the early chapters of this book were used in short courses on adaptive signal processing taught in different institutions, namely: Helsinki University of Technology, Espoo, Finland; University Menendez Pelayo in Seville, Spain; and at the Victoria Micronet Center, University of Victoria, Canada. The remaining parts of the book were written based on notes of the graduate course in adaptive signal processing taught at COPPE (the graduate engineering school of UFRJ).

The philosophy of the presentation is to expose the material with a solid theoretical foundation, while avoiding straightforward derivations and repetition. The idea was to keep the text with a manageable size, without sacrificing clarity and without omitting important subjects. Another objective is to bring the reader up to the point where implementation can be tried and research can begin. A number of references are included in the end of the chapters in order to aid the reader to proceed on learning the subject.

It is assumed the reader has previous background on the basic principles of digital signal processing and stochastic processes, including: discrete-time Fourier- and $\mathcal{Z}$-transforms, finite impulse response (FIR) and infinite impulse response (IIR) digital filter realizations, random variables and processes, first- and second-order statistics, moments, and filtering of random signals. Assuming that the reader has this background, I believe the book is self contained.

Chapter 1 introduces the basic concepts of adaptive filtering and sets a general framework that all the methods presented in the following chapters fall under. A brief introduction to the typical application of adaptive filtering is also presented.

In Chapter 2, the basic concepts of discrete-time stochastic processes are reviewed with special emphasis to the results that are useful to analyze the behavior of adaptive filtering algorithms. In addition, the Wiener filter is presented, establishing the optimum linear filter that can be sought in stationary environments. The concept of mean-square error surface is then introduced, another useful tool to analyze adaptive filters. The classical Newton and steepest-descent algorithms are briefly introduced. Since the use of these algorithms would require a complete knowledge of the stochastic environment, the adaptive filtering algorithms introduced in the following chapters come into play. Practical applications of the adaptive filtering algorithms are revisited in more detail at the end of Chapter 2.

Chapter 3 presents the analysis of the LMS algorithm in some depth. Several aspects are discussed, such as convergence behavior in stationary and nonstationary environments, and quantization effects in fixed- and floating-point arithmetics.

Chapter 4 deals with some algorithms that are in a sense related to the LMS algorithm. In particular, the algorithms introduced are the quantized-error algorithms, the LMS-Newton algorithm, the transform-domain algorithm, and the normalized LMS algorithm. Some properties of these algorithms are also discussed in Chapter 4.

Chapter 5 introduces the conventional recursive least-squares (RLS) algorithm. This algorithm minimizes a deterministic objective function, differing in this sense from the LMS-based algorithms. Following the same pattern of presentation of Chapter 3, several aspects of the conventional RLS algorithm are discussed, such as convergence behavior in stationary and nonstationary environments, and quantization effects in fixed- and floating-point arithmetics. The results presented, except for the quantization effects, are also valid to the RLS algorithms presented in the following chapters.

In Chapter 6, a family of fast RLS algorithms based on the FIR lattice realization is introduced. These algorithms represent an interesting alternative to the computationally complex conventional RLS algorithm. In particular, the unnormalized, the normalized and the error-feedback algorithms are presented.

Chapter 7 deals with the fast transversal RLS algorithms, which are very attractive due to their low computational complexity. However, these algorithms are known to face stability problems in practical implementation. As a consequence, special attention is given to the stabilized fast transversal RLS algorithm.

Chapter 8 is devoted to a family of RLS algorithms based on the QR decomposition. The conventional and two fast versions of the QR-based algorithms are presented in this chapter.

Chapter 9 addresses the subject of adaptive filters using IIR digital filter realizations. The chapter includes a discussion of how to compute the gradient and how to derive the adaptive algorithms. The cascade, the parallel, and the lattice realizations are presented as interesting alternative to the direct-form realization for the IIR adaptive filter. The characteristics of the mean-square error surface, for the IIR adaptive filtering case, are also discussed in this chapter. Algorithms based on alternative error formulations, such as the equation-error and Steiglitz-McBride methods are also introduced.

I decided to use some standard examples to present a number of simulation results, in order to test and compare different algorithms. This way a lot of repetition was avoided while allowing the reader to easily compare the performance of the algorithms.

Acknowledgments

The support and understanding of the Department of Electronic Engineering of the School of Engineering (undergraduate school) of UFRJ and of the Program of Electrical Engineering of COPPE have been fundamental to complete this work.

I was lucky enough to have contact with a number of creative professors and researchers that by taking their time to discuss technical matters with me, raised many interesting questions and provided me with enthusiasm to write this book. In that sense, I would like to thank Prof. Pan Agathoklis, University of Victoria; Dr. T. I. Laakso, Helsinki University of Technology; Prof. W. S. Lu, University of Victoria; Dr. H. S. Malvar, Picturetel; Prof. M. R. Petraglia, UFRJ; Prof. J. M. T. Romano, State University of Campinas; Dr. S. Sunder, Analog Devices; Prof. E. Sanchez Sinencio, Texas A&M University.

My M.Sc. supervisor, my friend and colleague, Prof. L. P. Calôba has been a source of inspiration and encouragement not only for this work but for my entire career. Prof. A. Antoniou, my Ph.D. supervisor, has also been an invaluable friend and advisor, I learned a lot by writing papers with him. Having these guys as Professors was great.

The superior students that attend engineering at UFRJ are for sure another source of inspiration. In particular, I have been lucky to attract good and dedicated graduate students, who have participated in the research related to adaptive filtering. They are R. G. Alves, J. Apolinário, Jr., Prof. L. W. P. Biscainho, M. L. R. Campos, Prof. J. E. Cousseau, F. Gil, S. L. Netto, T. C. Macedo, Jr., M. G. Siqueira, S. Subramanian (University of Victoria). Most of them took time from their Ph.D. work to read parts of the manuscript and providing me with invaluable suggestions. Some parts of this book have influence of my interactions with these students.

I am particularly grateful to Prof. J. E. Cousseau and to Mr. L. W. P. Biscainho, for their support in producing some of the examples of the book.

Mr. L. W. P. Biscainho, Dr. M. L. R. Campos, and Dr. S. L. Netto also read every inch of the manuscript and provided numerous suggestions for improvements.

I am most grateful to Profs. E. B. da Silva and M. R. Petraglia, both from UFRJ, for their critical inputs on parts of the manuscript.

I am also thankful to Prof. I. Hartimo, Helsinki University of Technology; Prof J. L. Huertas, University of Seville; Prof. A. Antoniou, University of Victoria; and Prof. J. E. Cousseau, Universidad Nacional del Sur for giving me the opportunity to teach at the institutions they work for.

The support of Catherine Chang, Prof. J. E. Cousseau, F. Gil, and Dr. S. Sunder for solving my problems with the editor is also deeply appreciated.

The financial support of the Brazilian research councils CNPq and CAPES was fundamental for the completion of this book.

My parents provided me with the moral and educational support needed to pursue any project, including this one. My mother's patience, love and understanding seems to be endless.

My brother Fernando always says yes, what else do I want?

My family deserves special thanks. My daughters Paula and Luiza have been extremely understanding, and always forgive daddy for taking their leisure time. They are wonderful kids. My wife Mariza deserves my deepest gratitude for her endless love, support, and friendship. She always does her best to provide me with the conditions to develop this and other projects.

Prof. Paulo S. R. Diniz

Niterói, Brazil

To: My Parents,
Mariza,
Paula,
and Luiza.

1

INTRODUCTION TO ADAPTIVE FILTERING

1.1 INTRODUCTION

In this section we define the kind of signal processing systems that will be treated in this text.

In the last thirty years significant contributions have been made in the signal processing field. The advances in digital circuit design have been the key technological development that sparkled a growing interest in the field of digital signal processing. The resulting digital signal processing systems are attractive due to their reliability, accuracy, small physical sizes, and flexibility.

One example of a digital signal processing system is called *filtering*. Filtering is a signal processing operation whose objective is to process a signal in order to manipulate the information contained in the signal. In other words, a filter is a device that maps its input signal in another output signal facilitating the extraction of the desired information contained in the input signal. A digital filter is the one that processes discrete-time signals represented in digital format. For time-invariant filters the internal parameters and the structure of the filter are fixed, and if the filter is linear the output signal is a linear function of the input signal. Once prescribed specifications are given, the design of time-invariant linear filters entails three basic steps, namely: the approximation of the specifications by a rational transfer function, the choice of an appropriate structure defining the algorithm, and the choice of the form of implementation for the algorithm.

An adaptive filter is required when either the fixed specifications are unknown or the specifications cannot be satisfied by time-invariant filters. Strictly speaking

P. S. R. Diniz, *Adaptive Filtering*

an adaptive filter is a nonlinear filter since its characteristics are dependent on the input signal and consequently the homogeneity and additivity conditions are not satisfied. However, if we freeze the filter parameters at a given instant of time, the adaptive filter considered in this text is linear in the sense that its output signal is a linear function of its input signal.

The adaptive filters are time-varying since their parameters are continually changing in order to meet a performance requirement. In this sense, we can interpret an adaptive filter as a filter that performs the approximation step on-line. Usually the definition of the performance criterion requires the existence of a reference signal that is usually hidden in the approximation step of fixed-filter design. This discussion brings the feeling that in the design of fixed (nonadaptive) filters a complete characterization of the input and reference signals is required in order to design the most appropriate filter that meets a prescribed performance. Unfortunately, this is not the usual situation encountered in practice, where the environment is not well defined. The signals that compose the environment are the input and the reference signals, and in cases where any of them is not well defined, the design procedure is to model the signals and subsequently design the filter. This procedure could be costly and difficult to implement on-line. The solution to this problem is to employ an adaptive filter that performs on-line updating of its parameters through a rather simple algorithm, using only the information available in the environment. In other words, the adaptive filter performs a data-driven approximation step.

The subject of this book is adaptive filtering, which concerns the choice of structures and algorithms for a filter that has its parameters (or coefficients) adapted, in order to improve a prescribed performance criterion. The coefficient updating is performed using the information available at a given time.

The development of digital very large scale integration (VLSI) technology allowed the widespread use of adaptive signal processing techniques in a large number of applications. This is the reason why in this book only discrete-time implementations of adaptive filters are considered. Obviously, we assume that continuous-time signals taken from the real world are properly sampled, i.e., they are represented by discrete-time signals with sampling rate higher than twice their highest frequency. Basically, it is assumed that when generating a discrete-time signal by sampling a continuous-time signal, the Nyquist or sampling theorem is satisfied [1]-[8].

1.2 ADAPTIVE SIGNAL PROCESSING

As previously discussed, the design of digital filters with fixed coefficients requires well defined prescribed specifications. However, there are situations where the specifications are not available, or are time varying. The solution in these cases is to employ a digital filter with adaptive coefficients known as adaptive filters [9]-[15].

Since no specifications are available, the adaptive algorithm that determines the updating of the filter coefficients, requires extra information that is usually given in the form of a signal. This signal is in general called a desired or reference signal, whose choice is normally a tricky task that depends on the application.

Adaptive filters are considered nonlinear systems, therefore their behavior analysis is more complicated than for fixed filters. On the other hand, because the adaptive filters are self designing filters, from the practitioner's point of view their design can be considered less involved than in the case of digital filters with fixed coefficients.

The general set up of an adaptive filtering environment is illustrated in Fig. 1.1, where k is the iteration number, $x(k)$ denotes the input signal, $y(k)$ is the adaptive filter output signal, and $d(k)$ defines the desired signal. The error signal $e(k)$ is calculated as $d(k) - y(k)$. The error signal is then used to form a performance (or objective) function that is required by the adaptation algorithm in order to determine the appropriate updating of the filter coefficients. The minimization of the objective function implies that the adaptive filter output signal is matching the desired signal in some sense.

The complete specification of an adaptive system, as shown in Fig. 1.1, consists of three items:

1) **Application**: The type of application is defined by the choice of the signals acquired from the environment to be the input and desired-output signals. The number of different applications in which adaptive techniques are being successfully used has increased enormously during the last decade. Some examples are echo cancellation, equalization of dispersive channels, system identification, signal enhancement, adaptive beamforming, noise cancelling, and control [13]-[17]. The study of different applications is not the main scope of this book. However, some applications are briefly considered.

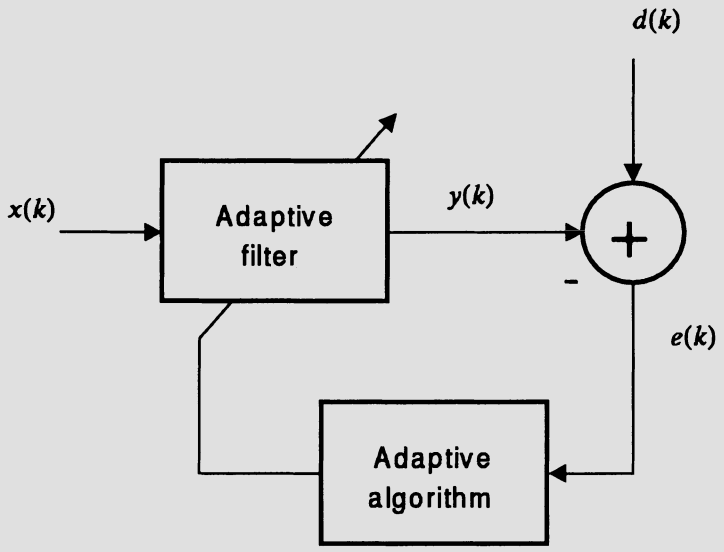


Figure 1.1 General adaptive filter configuration.

2) **Adaptive Filter Structure**: The adaptive filter can be implemented in a number of different structures or realizations. The choice of the structure can influence the computational complexity (amount of arithmetic operations per iteration) of the process and also the necessary number of iterations to achieve a desired performance level. Basically, there are two major classes of adaptive digital filter realizations, distinguished by the form of the impulse response, namely the finite-duration impulse response (FIR) filter and the infinite-duration impulse response (IIR) filters. FIR filters are usually implemented with nonrecursive structures, whereas IIR filters utilize recursive realizations.

- Adaptive FIR filter realizations: The most widely used adaptive FIR filter structure is the transversal filter, also called tapped delay line, that implements an all-zero transfer function with a canonic direct form realization without feedback. For this realization, the output signal $y(k)$ is a linear combination of the filter coefficients, that yields a quadratic mean-square error (MSE $= E[|e(k)|^2]$) function with a unique optimal solution. Other alternative adaptive FIR realizations are also used in order to obtain improvements as compared to the transversal filter structure, in terms of computational complexity, speed of convergence, and finite wordlength properties as will be seen later in the book.

- Adaptive IIR filter realizations: The most widely used realization of adaptive IIR filters is the canonic direct-form realization [4], due to its simple implementation and analysis. However, there are some inherent problems related to recursive adaptive filters which are structure dependent, such as pole-stability monitoring requirement and slow speed of convergence. To address these problems, different realizations were proposed attempting to overcome the limitations of the direct form structure. Among these alternative structures, the cascade, the lattice, and the parallel realizations are considered because of their unique features as will be discussed in Chapter 9.

3) **Algorithm**: The algorithm is the procedure used to adjust the adaptive filter coefficients in order to minimize a prescribed criterion. The algorithm is determined by defining the search method (or minimization algorithm), the objective function, and the error signal nature. The choice of the algorithm determines several crucial aspects of the overall adaptive process, such as existence of sub-optimal solutions, biased optimal solution, and computational complexity.

1.3 INTRODUCTION TO ADAPTIVE ALGORITHMS

The basic objective of the adaptive filter is to set its parameters, $\boldsymbol{\theta}(k)$, in such way that its output tries to minimize a meaningful objective function involving the reference signal. Usually, the objective function F is a function of the input, the reference, and adaptive filter output signals, i.e., $F = F[\boldsymbol{x}(k), d(k), y(k)]$. A consistent definition of the objective function must satisfy the following properties:

- Non-negativity: $F[\boldsymbol{x}(k), d(k), y(k)] \geq 0, \forall y(k), \boldsymbol{x}(k)$, and $d(k)$;
- Optimality: $F[\boldsymbol{x}(k), d(k), d(k)] = 0$.

One may understand that in an adaptive process, the adaptive algorithm attempts to minimize the function F, in such a way that $y(k)$ approximates $d(k)$, and as a consequence, $\boldsymbol{\theta}(k)$ converges to $\boldsymbol{\theta}_o$, where $\boldsymbol{\theta}_o$ is the optimum set of coefficients that leads to the minimization of the objective function.

Another way to interpret the objective function is to consider it a direct function of a generic error signal $e(k)$, which in turn is a function of the signals $x(k)$, $y(k)$, and $d(k)$, i.e., $F = F[e(k)] = F[e(x(k), y(k), d(k))]$. Using this framework, we can consider that an adaptive algorithm is composed of three basic items: definition of the minimization algorithm, definition of the objective function form, and definition of the error signal. These issues are discussed below

1) **Definition of the minimization algorithm for the function** F: This item is the main subject of Optimization Theory [19], and it essentially affects the speed of convergence and computational complexity of the adaptive process. The most commonly used optimization methods in the adaptive signal processing field are:

- Newton's method: This method seeks the minimum of a second-order approximation of the objective function using an iterative updating formula for the parameter vector given by

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) - \mu \mathbf{H}_{\boldsymbol{\theta}}^{-1}\{F[e(k)]\}\nabla_{\boldsymbol{\theta}}\{F[e(k)]\} \tag{1.1}$$

  where μ is a factor that controls the step size of the algorithm, i.e., it determines how fast the parameter vector will be changed. The matrix of second derivatives of $F[e(k)]$, $\mathbf{H}_{\boldsymbol{\theta}}\{F[e(k)]\}$ is the Hessian matrix of the objective function, and $\nabla_{\boldsymbol{\theta}}\{F[e(k)]\}$ is the gradient of the objective function with respect to the adaptive filter coefficients;

- *Quasi*-Newton methods: This class of algorithms is a simplified version of the method described above, as it attempts to minimize the objective function using a recursively calculated estimate of the inverse of the Hessian matrix, i.e.,

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) - \mu \mathbf{P}(k)\nabla_{\boldsymbol{\theta}}\{F[e(k)]\} \tag{1.2}$$

  where $\mathbf{P}(k)$ is an estimate of $\mathbf{H}_{\boldsymbol{\theta}}^{-1}\{F[e(k)]\}$, such that

$$\lim_{k\to\infty} \mathbf{P}(k) = \mathbf{H}_{\boldsymbol{\theta}}^{-1}\{F[e(k)]\}$$

  A usual way to calculate the inverse of the Hessian estimate is through the matrix inversion lemma (see, for example [18] and some chapters to come). Also, the gradient vector is usually replaced by a computationally efficient estimate.

- Steepest-descent method: This type of algorithm searches the objective function minimum point following the opposite direction of the gradient

vector of this function. Consequently, the updating equation assumes the form

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) - \mu \nabla_{\boldsymbol{\theta}} \{F[e(k)]\} \tag{1.3}$$

Here and in the open literature, the steepest-descent method is often also referred to as gradient method.

In general, gradient methods are easier to implement, but on the other hand, the Newton method usually requires a smaller number of iterations to reach a neighborhood of the minimum point. In many cases, *Quasi*-Newton methods can be considered a good compromise between the computational efficiency of the gradient methods and the fast convergence of the Newton method. However, the *Quasi*-Newton algorithms are susceptible to instability problems due to the recursive form used to generate the estimate of the inverse Hessian matrix. A detailed study of the most widely used minimization algorithms can be found in [19].

It should be pointed out that with any minimization method, the convergence factor μ controls the stability, speed of convergence, and some characteristics of residual error of the overall adaptive process. Usually, an appropriate choice of this parameter requires a reasonable amount of knowledge of the specific adaptive problem of interest. Consequently, there is no general solution to accomplish this task. In practice, computational simulations play an important role and are, in fact, the most used tool to address the problem.

2) **Definition of the objective function** $F[e(k)]$: There are many ways to define an objective function that satisfies the optimality and non-negativity properties formerly described. This definition affects the complexity of the gradient vector and the Hessian matrix calculation. Using the algorithm's computational complexity as a criterion, we can list the following forms for the objective function as the most commonly used in the derivation of an adaptive algorithm:

- Mean-Square Error (MSE): $F[e(k)] = E[|e(k)|^2]$;
- Least Squares (LS): $F[e(k)] = \frac{1}{k+1} \sum_{i=0}^{k} |e(k-i)|^2$;
- Weighted Least Squares (WLS): $F[e(k)] = \sum_{i=0}^{k} \lambda^i |e(k-i)|^2$, λ is a constant smaller than 1;
- Instantaneous Squared Value (ISV): $F[e(k)] = |e(k)|^2$.

The MSE, in a strict sense, is only of theoretical value, since it requires an infinite amount of information to be measured. In practice, this ideal objective function can be approximated by the other three listed. The LS, WLS, and ISV functions differ in the implementation complexity and in the convergence behavior characteristics; in general, the ISV is easier to implement but presents noisy convergence properties, since it represents a greatly simplified objective function. The LS is convenient to be used in stationary environment, whereas the WLS is useful in applications where the environment is slowly varying.

3) **Definition of the error signal** $e(k)$: The choice of the error signal is crucial for the algorithm definition, since it can affect several characteristics of the overall algorithm including computational complexity, speed of convergence, robustness, and most importantly for the IIR adaptive filtering case, the occurrence of biased and multiple solutions.

The minimization algorithm, the objective function, and the error signal as presented give us a structured and simple way to interpret, analyze, and study an adaptive algorithm. In fact, almost all known adaptive algorithms can be visualized in this form, or in a slight variation of this organization. In the remaining parts of this book, using this framework, we present the principles of adaptive algorithms. It may be observed that the minimization algorithm and the objective function affect the convergence speed of the adaptive process. An important step in the definition of an adaptive algorithm is the choice of the error signal, since this task exercises direct influence in many aspects of the overall convergence process.

1.4 APPLICATIONS

In this section, we discuss some possible choices for the input and desired signals and how these choices are related to the applications. Some of the classical applications of adaptive filtering are system identification, channel equalization, signal enhancement, and prediction.

In the system identification application, the desired signal is the output of the unknown system when excited by a broadband signal, in most cases a white-noise signal. The broadband signal is also used as input for the adaptive filter as illustrated in Fig. 1.2. When the output MSE is minimized, the adaptive filter represents a model for the unknown system.

The channel equalization scheme consists of applying the originally transmitted signal distorted by the channel as the input signal to an adaptive filter, whereas the desired signal is a delayed version of the original signal as depicted in Fig. 1.3. This delayed version of the input signal is in general available at the receiver in a form of standard training signal. The minimization of the MSE indicates that the adaptive filter represents an inverse model (equalizer) of the channel.

In the signal enhancement case, a signal $x(k)$ is corrupted by noise $n_1(k)$, and a signal $n_2(k)$ correlated to the noise is available (measurable). If $n_2(k)$ is used as an input to the adaptive filter with the signal corrupted by noise playing the role of the desired signal, after convergence the output error will be an enhanced version of the signal. Fig. 1.4 illustrates a typical signal enhancement setup.

Finally, in the prediction case the desired signal is a forward (or eventually a backward) version of the adaptive filter input signal as shown in Fig. 1.5. After convergence, the adaptive filter represents a model for the input signal, and can be used as a predictor model for the input signal.

Further details regarding the applications discussed here will be given in the following chapters.

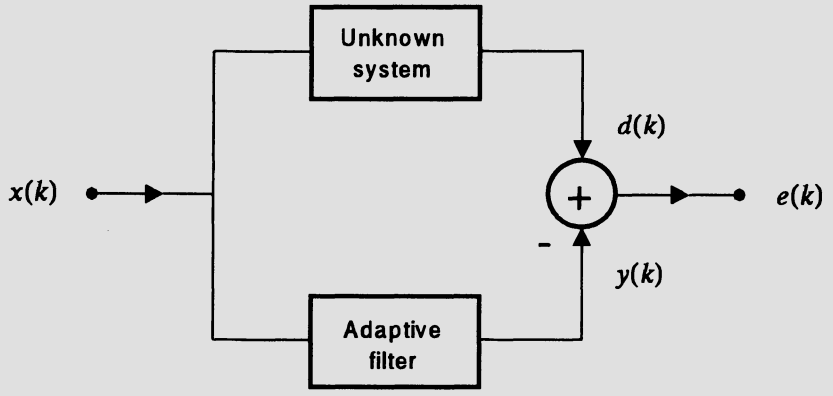


Figure 1.2 System identification.

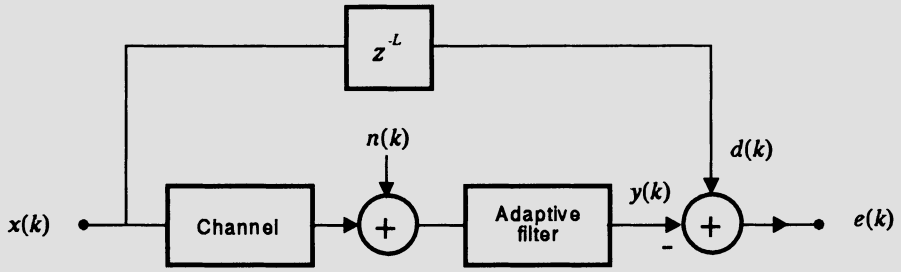


Figure 1.3 Channel equalization.

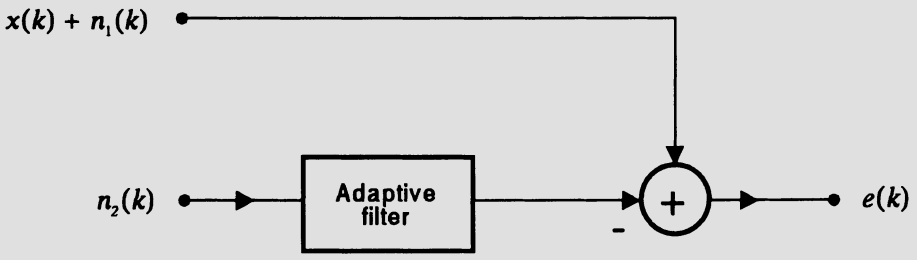


Figure 1.4 Signal enhancement ($n_1(k)$ and $n_2(k)$ are noise signals correlated to each other).

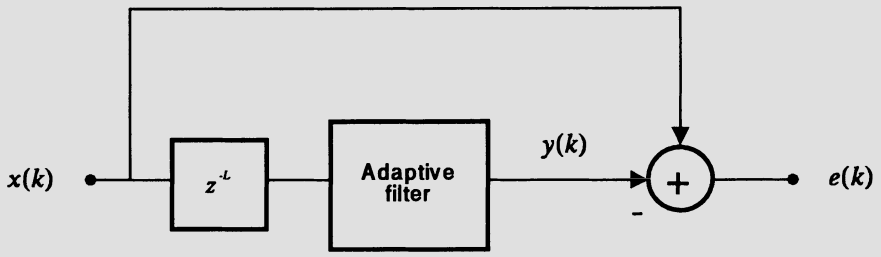


Figure 1.5 Signal prediction.

Example 1.1

Before concluding this chapter we present a simple example in order to illustrate how an adaptive filter can be useful in solving problems that lie in the general framework represented by Fig. 1.1. We chose the signal enhancement application illustrated in Fig. 1.4.

In this example the reference (or desired) signal consists of a discrete-time triangular waveform corrupted by a colored noise. Fig. 1.6 shows the desired signal. The adaptive filter input signal is a white noise correlated with the the noise signal that corrupted the triangular waveform. In Fig. 1.7 is shown the input signal.

The coefficients of the adaptive filter are adjusted in order to keep the average value of the output error as small as possible. As can be noticed in Fig. 1.8, as the number of iterations increase the error signal resembles the discrete-time triangular waveform shown in the same figure (dashed curve).

□

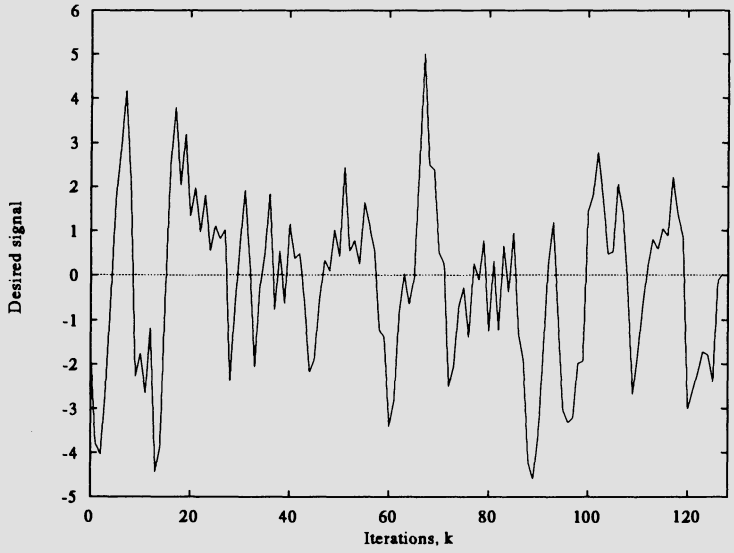


Figure 1.6 Desired signal.

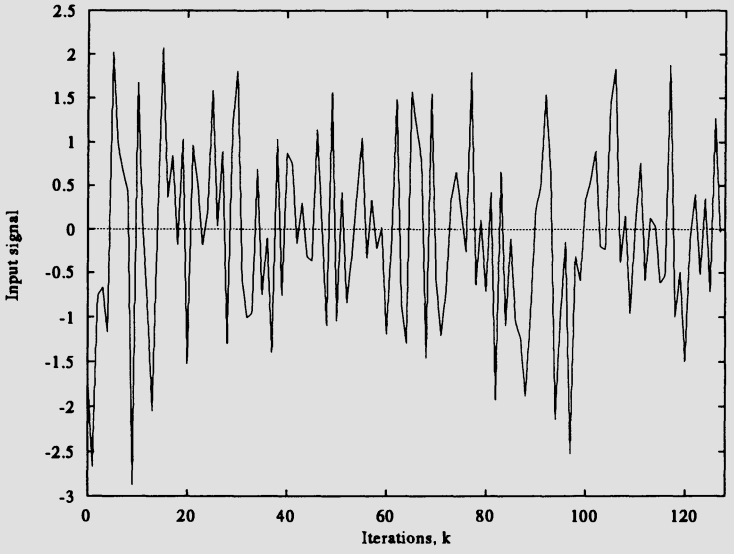


Figure 1.7 Input signal.

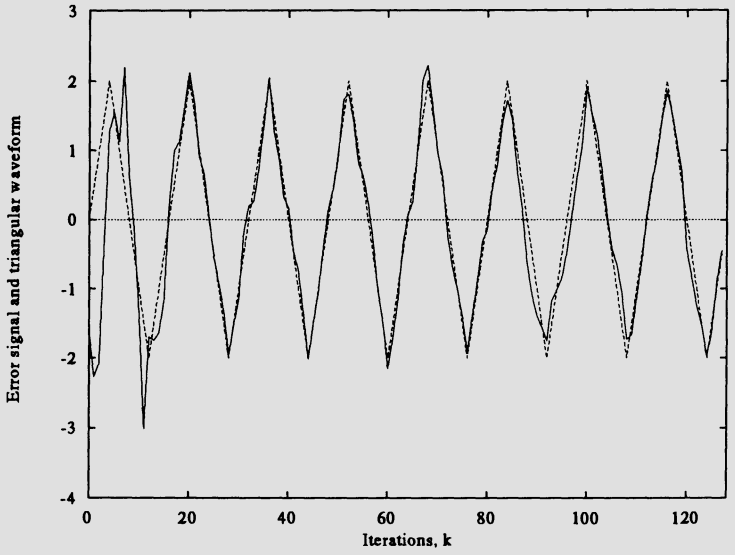


Figure 1.8 Error signal and triangular waveform.

References

1. A. Papoulis, *Signal Analysis*, McGraw Hill, New York, NY, 1977.

2. A. V. Oppenheim, A. S. Willsky, and I. T. Young, *Signals and Systems*, Prentice Hall, Englewood Cliffs, NJ, 1983.

3. A. V. Oppenheim and R. W. Schaffer, *Discrete-Time Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1989.

4. A. Antoniou, *Digital Filters: Analysis, Design, and Applications*, McGraw Hill, New York, NY, 2nd edition, 1992.

5. L. B. Jackson, *Digital Filters and Signal Processing*, Kluwer Academic Publishers, Norwell, MA, 3rd edition, 1996.

6. R. A. Roberts and C. T. Mullis, *Digital Signal Processing*, Addison-Wesley, Reading, MA, 1987.

7. J. G. Proakis and D. G. Manolakis, *Digital Signal Processing*, Macmillan, New York, NY, 2nd edition, 1992.

8. M. Bellanger, *Digital Processing of Signals*, John Wiley & Sons, New York, NY, 1984.

9. M. L. Honig and D. G. Messerschmitt, *Adaptive Filters: Structures, algorithms, and Applications*, Kluwer Academic Publishers, Boston, MA, 1984.

10. S. T. Alexander, *Adaptive Signal Processing*, Springer Verlag, New York, NY, 1986.

11. M. Bellanger, *Adaptive Digital Filters and Signal Analysis*, Marcel Dekker, Inc., New York, NY, 1987.

12. P. Strobach, *Linear Prediction Theory*, Springer Verlag, New York, NY, 1990.

13. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1985.

14. J. R. Treichler, C. R. Johnson, Jr., and M. G. Larimore, *Theory and Design of Adaptive Filters*, John Wiley & Sons, New York, NY, 1987.

15. S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, NJ, 2nd edition, 1991.

16. L. R. Rabiner and R. W. Schaffer, *Digital Processing of Speech Signals*, Prentice Hall, Englewood Cliffs, NJ, 1978.

17. D. H. Johnson and D. E. Dudgeon, *Array Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1993.

18. T. Kailath, *Linear Systems*, Prentice Hall, Englewood Cliffs, NJ, 1980.

19. D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison Wesley, Reading, MA, 1973.

2

FUNDAMENTALS OF ADAPTIVE FILTERING

2.1 INTRODUCTION

This chapter includes a brief review of deterministic and random signal representations. Due to the extent of those subjects our review is limited to the concepts that are directly relevant to adaptive filtering. The properties of the correlation matrix of the input signal vector are investigated in some detail, since they play a key role in the statistical analysis of the adaptive filtering algorithms.

The Wiener solution that represents the minimum mean-square error (MSE) solution of discrete-time filters realized through a linear combiner is also introduced. This solution depends on the input signal correlation matrix as well as on the the cross-correlation between the elements of the input signal vector and the reference signal. The values of these correlations form the parameters of the MSE surface, which is a quadratic function of the adaptive filter coefficients. Motivated by the importance of the properties of the MSE surface, we analyze them using some results related to the input signal correlation matrix.

In practice the parameters that determine the MSE surface shape are not available. What is left is to directly or indirectly estimate these parameters using the available data and to develop adaptive algorithms that use these estimates to search the MSE surface such that the adaptive filter coefficients converge to the Wiener solution in some sense. The starting point to obtain an estimation procedure is to investigate the convenience of using the classical searching methods of optimization theory [1]-[2] to adaptive filtering. The Newton and steepest-descent algorithms are investigated as possible searching methods for adaptive filtering. Although both methods are not directly applicable to practical ad-

P. S. R. Diniz, *Adaptive Filtering*

aptive filtering, smart reflections inspired on them led to practical algorithms such as the least-mean-square (LMS) [3]-[4] and Newton-based algorithms. The Newton and steepest-descent algorithms are introduced in this chapter, whereas the LMS algorithm is treated in the next chapter.

Also, in the present chapter, the main applications of adaptive filters are revisited and discussed in greater detail.

2.2 SIGNAL REPRESENTATION

In this section we briefly review some concepts related to deterministic and random discrete-time signals. Only specific results essential to the understanding of adaptive filtering are reviewed. For further details on signals and digital signal processing we refer to [5]-[12].

2.2.1 Deterministic Signals

A deterministic discrete-time signal is characterized by a defined mathematical function of the time index k [1], with $k = 0, \pm 1, \pm 2, \pm 3, \ldots$. An example of a deterministic signal (or sequence) is

$$x(k) = e^{-\alpha k} \cos(\omega k) + u(k) \tag{2.1}$$

where $u(k)$ is the unit step sequence.

The response of a linear time-invariant filter to an input $x(k)$ is given by the convolution summation, as follows [6]:

$$\begin{aligned} y(k) &= x(k) * h(k) = \sum_{n=-\infty}^{\infty} x(n)h(k-n) \\ &= \sum_{n=-\infty}^{\infty} h(n)x(k-n) = h(k) * x(k) \end{aligned} \tag{2.2}$$

where $h(k)$ is the impulse response of the filter.

[1] The index k can also denote space in some applications.

The $\mathcal{Z}$-transform of a given sequence $x(k)$ is defined as

$$\mathcal{Z}\{x(k)\} = X(z) = \sum_{k=-\infty}^{\infty} x(k)z^{-k} \tag{2.3}$$

If the $\mathcal{Z}$-transform is defined for a given region of the $\mathcal{Z}$-plane, in other words the summation above converges in that region, the convolution operation can be replaced by a product of the $\mathcal{Z}$-transforms as follows [6]:

$$Y(z) = H(z)\,X(z) \tag{2.4}$$

where $Y(z)$, $X(z)$, and $H(z)$ are the $\mathcal{Z}$-transforms of $y(k)$, $x(k)$, and $h(k)$, respectively. Considering only waveforms that start at an instant $k \geq 0$ and and have finite power, their $\mathcal{Z}$-transforms will always be defined outside the unit circle.

For finite-energy waveforms it is convenient to use the discrete-time Fourier transform defined as

$$\mathcal{F}\{x(k)\} = X(e^{j\omega}) = \sum_{k=-\infty}^{\infty} x(k)e^{-j\omega k} \tag{2.5}$$

Although the discrete-time Fourier transform does not exist for a signal with infinite energy, however if the signal has finite-power, a generalized discrete-time Fourier transform exists and is largely used for deterministic signals [15].

2.2.2 Random Signals

A random variable X is a function that assigns a number to every outcome of a given experiment denoted by ϱ. A stochastic process is a rule to describe the time evolution of the random variable depending on ϱ, therefore it is a function of two variables $\mathrm{X}(k, \varrho)$. The set of all experimental outcomes, i.e., the ensemble, is the domain of ϱ. We denote $x(k)$ as a sample of the given process with ϱ fixed, where in this case if k is also fixed, $x(k)$ is a number. When any statistical operator is applied to $x(k)$ it is implied that k is fixed and ϱ is variable. In this book $x(k)$ represents a random signal.

Random signals do not have a precise description of their waveforms. What is possible is to characterize them via measured statistics or through a probabilistic model. For random signals the first- and second-order statistics are sufficient

most of the time for characterization of the stochastic process. The first- and second-order statistics are also convenient for measurements. In addition, the effect on these statistics caused by linear filtering can be easily accounted for.

Let's consider for the time being that the random signals are real. The expected value, or mean value, of the process is defined by

$$m_x(k) = E[x(k)] \tag{2.6}$$

The definition of the expected value is expressed as

$$E[x(k)] = \int_{-\infty}^{\infty} y p_{x(k)}(y) dy \tag{2.7}$$

where $p_{x(k)}(y)$ is the probability density function (pdf) of $x(k)$ at the point y. In order to interpret the pdf we need to define the distribution function of a random variable as

$$P_{x(k)}(y) = \textit{probability of } x(k) \textit{ being smaller or equal to } y$$

or

$$P_{x(k)}(y) = \int_{-\infty}^{y} p_{x(k)}(z) dz \tag{2.8}$$

The derivative of the distribution function is the pdf

$$p_{x(k)}(y) = \frac{dP_{x(k)}(y)}{dy} \tag{2.9}$$

The autocorrelation function of the process $x(k)$ is defined by

$$r_x(k,l) = E[x(k)x(l)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} yz p_{x(k),x(l)}(y,z) dy dz \tag{2.10}$$

where $p_{x(k),x(l)}(y,z)$ is the joint probability density of the random variables $x(k)$ and $x(l)$ defined as

$$p_{x(k),x(l)}(y,z) = \frac{\partial^2 P_{x(k),x(l)}(y,z)}{\partial y \partial z} \tag{2.11}$$

where

$$P_{x(k),x(l)}(y,z) = \textit{probability of } \{x(k) \leq y \textit{ and } x(l) \leq z\}$$

The autocovariance function is defined as

$$\sigma_x^2(k,l) = E[(x(k) - m_x(k))(x(l) - m_x(l))] = r_x(k,l) - m_x(k)m_x(l) \quad (2.12)$$

where the second equality follows from the definitions of mean value and autocorrelation. For $k = l$, $\sigma_x^2(k,l) = \sigma_x^2(k)$ which is the variance of $x(k)$.

The most important specific example of probability density function is the Gaussian density function, also known as normal density function [13]-[14]. The Gaussian pdf is defined by

$$p_{x(k)}(y) = \frac{1}{\sqrt{2\pi\sigma_x^2(k)}} e^{-\frac{(y-m_x(k))^2}{2\sigma_x^2(k)}} \quad (2.13)$$

where $m_x(k)$ and $\sigma_x^2(k)$ are the mean and variance of $x(k)$, respectively.

One justification for the importance of the Gaussian distribution is the central limit theorem. Given a random variable x composed by the sum of n independent random variables x_i as follows:

$$x = \sum_{i=1}^{n} x_i \quad (2.14)$$

the central limit theorem states that under certain general conditions, the probability density function of x approaches a Gaussian density function for large n. The mean and variance of x are given respectively by

$$m_x = \sum_{i=1}^{n} m_{x_i} \quad (2.15)$$

$$\sigma_x^2 = \sum_{i=1}^{n} \sigma_{x_i}^2 \quad (2.16)$$

Considering that the values of the mean and variance of y can grow, define

$$x' = \frac{x - m_x}{\sigma_x} \quad (2.17)$$

In this case, for $n \to \infty$ it follows that

$$p_{x'}(y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} \quad (2.18)$$

In a number of situations we require the calculation of conditional distributions, where the probability of a certain event to occur is calculated assuming that another event B has occurred. In this case, we define

$$P_{x(k)}(y|B) = \frac{P(\{x(k) \leq y\} \cap B)}{P(B)} \tag{2.19}$$
$$\triangleq \textit{probability of } x(k) \leq y \textit{ assuming } B \textit{ has occurred}$$

This joint event consists of all outcomes $\varrho \in B$ such that $\mathrm{X}(\varrho) \leq y$. The definition of the conditional mean is given by

$$m_{x|B}(k) = E[x(k)|B] = \int_{-\infty}^{\infty} y p_{x(k)}(y|B) dy \tag{2.20}$$

where $p_{x(k)}(y|B)$ is the pdf of $x(k)$ conditioned on B.

The conditional variance is defined as

$$\sigma^2_{x|B}(k) = E[(x(k) - m_{x|B}(k))^2|B] = \int_{-\infty}^{\infty} (y - m_{x|B}(k))^2 p_{x(k)}(y|B) dy \tag{2.21}$$

There are processes such that the mean and autocorrelation functions are shift (or time) invariant, i.e.,

$$m_x(k - i) = m_x(k) = E[x(k)] = m_x \tag{2.22}$$

$$r_x(k, i) = E[x(k - j)x(i - j)] = r_x(k - i) = r_x(l) \tag{2.23}$$

and as a consequence

$$\sigma^2_x(l) = r_x(l) - m^2_x \tag{2.24}$$

These processes are said to be wide-sense stationary (WSS). If the nth-order statistics of a process is shift invariant, the process is said to be nth-order stationary. Also if the process is nth-order stationary for any value of n the process is stationary in strict sense.

Two processes are considered jointly WSS if and only if any linear combination of them is also WSS. This is the same as:

$$y(k) = k_1\, x_1(k) + k_2\, x_2(k) \tag{2.25}$$

must be WSS, for any constants k_1 and k_2, if $x_1(k)$ and $x_2(k)$ are jointly WSS. This property implies that both $x_1(k)$ and $x_2(k)$ have shift-invariant means and autocorrelations and that their cross-correlation is also shift invariant.

For complex signals where $x(k) = x_r(k) + jx_i(k)$, $y = y_r + jy_i$, and $z = z_r + jz_i$, we have the following definition of the expected value

$$E[x(k)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} y p_{x_r(k),x_i(k)}(y_r, y_i) dy_r dy_i \tag{2.26}$$

where $p_{x_r(k),x_i(k)}(y_r, y_i)$ is the joint probability density function (pdf) of $x_r(k)$ and $x_i(k)$.

The autocorrelation function of the complex random signal $x(k)$ is defined by

$$\begin{aligned} r_x(k,l) &= E[x(k)x^*(l)] \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} yz^* p_{x_r(k),x_i(k),x_r(l),x_i(l)}(y_r, y_i, z_r, z_i) dy_r dy_i dz_r dz_i \end{aligned} \tag{2.27}$$

where $*$ denotes complex conjugate, since we assume for now that we are dealing with complex signals, and $p_{x_r(k),x_i(k),x_r(l),x_i(l)}(y_r, y_i, z_r, z_i)$ is the joint probability density function of the random variables $x(k)$ and $x(l)$.

For complex signals the autocovariance function is defined as

$$\sigma_x^2(k,l) = E[(x(k) - m_x(k))(x(l) - m_x(l))^*] = r_x(k,l) - m_x(k)m_x^*(l) \tag{2.28}$$

Autoregressive Moving Average Process

The process resulting from the output of a system described by a general linear difference equation given by

$$y(k) = \sum_{j=0}^{M} b_j x(k-j) + \sum_{i=1}^{N} a_i y(k-i) \tag{2.29}$$

where $x(k)$ is a white noise, is called autoregressive moving average (ARMA) process. The coefficients a_i and b_j are the parameters of the ARMA process. The output signal $y(k)$ is also said to be a colored noise since the autocorrelation function of $y(k)$ is nonzero for a lag different from zero, i.e., $r(l) \neq 0$ for some $l \neq 0$.

For the special case where $b_j = 0$ for $j = 1, 2, \ldots, M$ the resulting process is called autoregressive (AR) process. The terminology means that the process depends on the present value of the input signal and on a linear combination

of past samples of the process. This indicates the presence of a feedback of the output signal.

For the special case where $a_i = 0$ for $i = 1, 2, \ldots, N$ the process is identified as a moving average (MA) process. This terminology indicates that the process depends on a linear combination of the present and past samples of the input signal. In summary, an ARMA process can be generated by applying a white noise to the input of a digital filter with poles and zeros, whereas for the AR and MA cases the digital filter is an all-pole and all-zero filter, respectively.

Markov Process

A stochastic process is called a Markov process if its past has no influence in the future if the present is specified [13], [15]. In other words, the present behavior of the process depends only on the most recent past, all behavior previous to the most recent past is not required. A first-order AR process is a first-order Markov process, whereas an Nth-order AR process is considered an Nth-order Markov process. Take as an example the sequence

$$y(k) = ay(k-1) + n(k) \tag{2.30}$$

where $n(k)$ is a white noise process. The process represented by $y(k)$ is determined by $y(k-1)$ and $n(k)$, and no information before the instant $k-1$ is required. We conclude that $y(k)$ represents a Markov process. In the previous example, if $a = 1$ and $y(-1) = 0$ the signal $y(k)$, for $k \geq 0$, is a sum of white noise samples, usually called random walk sequence.

Formally, an mth-order Markov process satisfies the following condition: for all $k \geq 0$, and for a fixed m, it follows that

$$\begin{aligned} &P_{x(k)}(y|x(k-1), x(k-2), \ldots, x(0)) \\ &= P_{x(k)}(y|x(k-1), x(k-2), \ldots, x(k-m)) \end{aligned} \tag{2.31}$$

Wold Decomposition

Another important result related to any wide-sense stationary process $x(k)$ is the Wold decomposition, which states that $x(k)$ can be decomposed as

$$x(k) = x_r(k) + x_p(k) \tag{2.32}$$

where $x_r(k)$ is a regular process that is equivalent to the response of a stable, linear, time-invariant, and causal filter to a white noise [15], and $x_p(k)$ is a

perfectly predictable (deterministic or singular) process. Also, $x_p(k)$ and $x_r(k)$ are orthogonal processes, i.e., $E[x_r(k)x_p(k)] = 0$. The key factor here is that the regular process can be modeled through a stable autoregressive model [19] with a stable and causal inverse. The importance of Wold decomposition lies on the observation that a WSS process can in part be represented by an AR process of adequate order, with the remaining part consisting of a perfectly predictable process. Obviously the perfectly predictable process part of $x(k)$ also admits an AR model with zero excitation.

Power Spectral Density

Stochastic signals that are wide-sense stationary are persistent and therefore are not finite-energy signals. On the other hand, they have finite-power such that the generalized discrete-time Fourier transform can be applied to them. When the generalized discrete-time Fourier transform is applied to a WSS process it leads to a random function of the frequency [15]. The autocorrelation functions of most practical stationary processes have discrete-time Fourier transform. Therefore, the discrete-time Fourier transform of the autocorrelation function of a stationary random process can be very useful in many situations. This transform, called power spectral density, is defined as

$$R_x(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r_x(l)e^{-j\omega l} = \mathcal{F}[r_x(l)] \tag{2.33}$$

where $r_x(l)$ is the autocorrelation of the process represented by $x(k)$. The inverse discrete-time Fourier transform allows us to recover $r_x(l)$ from $R_x(e^{j\omega})$ by employing the relation

$$r_x(l) = \frac{1}{2\pi}\int_{-\pi}^{\pi} R_x(e^{j\omega})e^{j\omega\, l}d\omega = \mathcal{F}^{-1}[R_x(e^{j\omega})] \tag{2.34}$$

It should be mentioned that $R_x(e^{j\omega})$ is a deterministic function of ω, and can be interpreted as the energy of the random process at a given frequency in the ensemble, i.e., considering the average outcome of all possible realizations of the process. In particular, the mean squared value of the process represented by $x(k)$ is given by

$$r_x(0) = \frac{1}{2\pi}\int_{-\pi}^{\pi} R_x(e^{j\omega})d\omega \tag{2.35}$$

If the random signal representing any single realization of a stationary process is applied as input to a linear and time-invariant filter with impulse response

$h(k)$, the following equalities are valid and can be easily verified:

$$y(k) = \sum_{n=-\infty}^{\infty} x(n)h(k-n) = x(k) * h(k) \tag{2.36}$$

$$r_y(l) = r_x(l) * r_h(l) \tag{2.37}$$

$$R_y(e^{j\omega}) = R_x(e^{j\omega})|H(e^{j\omega})|^2 \tag{2.38}$$

$$r_{yx}(l) = r_x(l) * h(l) = E[x(l)y^*(l)] \tag{2.39}$$

$$R_{yx}(e^{j\omega}) = R_x(e^{j\omega})H(e^{j\omega}) \tag{2.40}$$

where $r_h(l) = h(l) * h(-l)$, $R_y(e^{j\omega})$ is the power spectral density of the output signal, $r_{yx}(k)$ is the cross-correlation of $x(k)$ and $y(k)$, and $R_{yx}(e^{j\omega})$ is the cross-power spectral density.

The main feature of the spectral density function is to allow a simple analysis of the average behavior of WSS random signals processed with linear time-invariant systems. As an illustration, suppose a white noise is applied as input to a lowpass filter with impulse response $h(k)$ and sharp cutoff at a given frequency ω_l. The autocorrelation function of the output signal $y(k)$ will not be a single impulse, it will be $h(k) * h(-k)$. Therefore, the signal $y(k)$ will look like a band-limited random signal, in this case, a slow-varying noise. Some properties of the function $R_x(e^{j\omega})$ of a discrete-time and stationary stochastic process are worth mentioning. The power spectrum density is a periodic function of ω, with period 2π, as can be verified from its definition. Also, since for stationary and complex random process we have $r_x(-l) = r_x^*(l)$, $R_x(e^{j\omega})$ is real. Despite of the usefulness of the power spectrum density function in dealing with WSS processes, it will not be widely used in this book since usually the filters considered here are time varying. However, it should be noted its important role in areas such as spectrum estimation [20]-[21].

If the $\mathcal{Z}$-transforms of the autocorrelation and cross-correlation functions exist, we can generalize the definition of power spectral density. In particular, the definition of equation (2.33) corresponds to the following relation

$$\mathcal{Z}[r_x(k)] = R_x(z) = \sum_{k=-\infty}^{\infty} r_x(k)z^{-k} \tag{2.41}$$

As discussed before, if the random signal representing any single realization of a stationary process is applied as input to a linear and time-invariant filter with impulse response $h(k)$, the following equalities are valid:

$$R_y(z) = R_x(z)H(z)H(z^{-1}) \tag{2.42}$$

and

$$R_{yx}(z) = R_x(z)H(z) \tag{2.43}$$

where $H(z) = \mathcal{Z}[h(l)]$. If we wish to calculate the cross-correlation of $y(k)$ and $x(k)$, namely $r_{yx}(0)$, we can use the inverse $\mathcal{Z}$-transform formula as follows:

$$\begin{aligned} E[y(k)x^*(k)] &= \frac{1}{2\pi j}\oint R_{yx}(z)\frac{dz}{z} \\ &= \frac{1}{2\pi j}\oint H(z)R_x(z)\frac{dz}{z} \end{aligned} \tag{2.44}$$

where the integration path is a counterclockwise closed contour in the region of convergence of $R_{yx}(z)$. The contour integral equation above is usually solved through the Cauchy's residue theorem [7].

2.2.3 Ergodicity

In the probabilistic approach, the statistical parameters of the real data are obtained through ensemble averages (or expected values). The estimation of any parameter of the stochastic process can be obtained by averaging a large number of realizations of the given process at each instant of time. However, in many applications only a few or even a single sample of the process is available. In these situations, we need to find out in which cases the statistical parameters of the process can be estimated by using time average of a single sample (or ensemble member) of the process. This is obviously not possible if the desired parameter is time varying. The equivalence between the ensemble average and time average is called ergodicity [13], [15].

The time average of a given stationary process represented by $x(k)$ is calculated by

$$\hat{m}_{x_N} = \frac{1}{2N+1}\sum_{k=-N}^{N} x(k) \tag{2.45}$$

If

$$\sigma^2_{\hat{m}_{x_N}} = \lim_{N\to\infty} E\{|\hat{m}_{x_N} - m_x|^2\} = 0$$

the process is said to be mean-ergodic in the mean-square sense. Therefore, the mean-ergodic process has time average that approximates the ensemble average as $N \to \infty$. Obviously, $\hat{m}_{x_N}$ is an unbiased estimate of m_x since

$$E[\hat{m}_{x_N}] = \frac{1}{2N+1}\sum_{k=-N}^{N} E[x(n)] = m_x \tag{2.46}$$

Therefore, the process will be considered *ergodic* if the variance of $\hat{m}_{x_N}$ tends to zero ($\sigma^2_{\hat{m}_{x_N}} \to 0$) when $N \to \infty$. The variance $\sigma^2_{\hat{m}_{x_N}}$ can be expressed after some manipulations as

$$\sigma^2_{\hat{m}_{x_N}} = \frac{1}{2N+1} \sum_{l=-2N}^{2N} \sigma_x^2(l) \left(1 - \frac{|l|}{2N+1}\right) \tag{2.47}$$

where $\sigma_x^2(l)$ is the autocovariance of the stochastic process $x(k)$. The variance of $\hat{m}_{x_N}$ tends to zero if and only if

$$\lim_{N\to\infty} \frac{1}{N} \sum_{l=0}^{N} \sigma_x^2(l) \to 0$$

The condition above is necessary and sufficient to guarantee that the process is mean-ergodic.

The ergodicity concept can be extended to higher order statistics. In particular, for second-order statistics we can define the process

$$x_l(k) = x(k+l)x^*(k) \tag{2.48}$$

where the mean of this process corresponds to the autocorrelation of $x(k)$, i.e., $r_x(l)$. Mean-ergodicity of $x_l(k)$ implies mean-square ergodicity of the autocorrelation of $x(k)$.

The time average of $x_l(k)$ is given by

$$\hat{m}_{x_{l,N}} = \frac{1}{2N+1} \sum_{k=-N}^{N} x_l(k) \tag{2.49}$$

that is an unbiased estimate of $r_x(l)$. If the variance of $\hat{m}_{x_{l,N}}$ tends to zero as N tends to infinity, the process $x(k)$ is said to be mean-square ergodic of the autocorrelation, i.e.,

$$\lim_{N\to\infty} E\{|\hat{m}_{x_{l,N}} - r_x(l)|^2\} = 0 \tag{2.50}$$

The condition above is satisfied if and only if

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N} E\{x(k+l)x^*(k)x(k+l+i)x^*(k+i)\} - r_x^2(l) = 0 \tag{2.51}$$

where it is assumed that $x(n)$ has stationary fourth-order moments. The concept of ergodicity can be extended to nonstationary processes [15], however, that is beyond the scope of this book.

2.3 THE CORRELATION MATRIX

Usually, adaptive filters utilize the available input signals at instant k in their updating equations. These inputs are the elements of the input signal vector denoted by

$$\mathbf{x}(k) = [x_0(k)x_1(k)\ldots x_N(k)]^T$$

As will be noted, the characteristics of the correlation matrix $\mathbf{R} = E[\mathbf{x}(k)\mathbf{x}^H(k)]$ plays a key role in the understanding of properties of most adaptive filtering algorithms. As a consequence, it is important to examine the main properties of the matrix $\mathbf{R}$. Some properties of the correlation matrix comes from the statistical nature of the adaptive filtering problem, whereas other properties derive from the linear algebra theory.

For a given input vector, the correlation matrix is given by

$$\begin{aligned}\mathbf{R} &= \begin{bmatrix} E[|x_0(k)|^2] & E[x_0(k)x_1^*(k)] & \cdots & E[x_0(k)x_N^*(k)] \\ E[x_1(k)x_0^*(k)] & E[|x_1(k)|^2] & \cdots & E[x_1(k)x_N^*(k)] \\ \vdots & \vdots & \ddots & \vdots \\ E[x_N(k)x_0^*(k)] & E[x_N(k)x_1^*(k)] & \cdots & E[|x_N(k)|^2] \end{bmatrix} \\ &= E[\mathbf{x}(k)\mathbf{x}^H(k)] \end{aligned} \tag{2.52}$$

where $\mathbf{x}^H(k)$ is the Hermitian transposition of $\mathbf{x}(k)$, that means transposition followed by complex conjugation or vice versa.

The main properties of the $\mathbf{R}$ matrix are listed below:

1. The matrix $\mathbf{R}$ is positive semidefinite.

   Proof:

   Given an arbitrary complex weight vector $\mathbf{w}$, we can form a signal given by

   $$y(k) = \mathbf{w}^H\mathbf{x}(k)$$

   The magnitude squared of $y(k)$ is

   $$y(k)y^*(k) = |y(k)|^2 = \mathbf{w}^H\mathbf{x}(k)\mathbf{x}^H(k)\mathbf{w} \geq 0$$

   The mean-square (MS) value of $y(k)$ is then given by

   $$MS[y(k)] = E[|y(k)|^2] = \mathbf{w}^H E[\mathbf{x}(k)\mathbf{x}^H(k)]\mathbf{w} = \mathbf{w}^H\mathbf{R}\mathbf{w} \geq 0$$

Therefore, the matrix $\mathbf{R}$ is positive semidefinite.

□

Usually, the matrix $\mathbf{R}$ is positive definite, unless the signals that compose the input vector are linearly dependent. Linear dependent signals are rarely found in practice.

2. For WSS processes $x_i(k)$, the matrix $\mathbf{R}$ is Hermitian, i.e.,

$$\mathbf{R} = \mathbf{R}^H \tag{2.53}$$

Proof:

$$\mathbf{R}^H = E[(\mathbf{x}(k)\mathbf{x}^H(k))^H] = E[\mathbf{x}(k)\mathbf{x}^H(k)] = \mathbf{R}$$

□

3. A matrix is Toeplitz if the elements of the main diagonal and of any secondary diagonal are equal. When the input signal vector is composed of delayed versions of the same signal taken from a WSS process, matrix $\mathbf{R}$ is Toeplitz.

   Proof:

   For the delayed signal input vector, with $x(k)$ WSS, matrix $\mathbf{R}$ has the following form

$$\mathbf{R} = \begin{bmatrix} r_x(0) & r_x(1) & \cdots & r_x(N) \\ r_x(-1) & r_x(0) & \cdots & r_x(N-1) \\ \vdots & \vdots & \ddots & \vdots \\ r_x(-N) & r_x(-N+1) & \cdots & r_x(0) \end{bmatrix} \tag{2.54}$$

   By examining the right-hand side of the equation above, we can easily conclude that $\mathbf{R}$ is Toeplitz.

□

Note that $r_x^*(i) = r_x(-i)$, what also follows from the fact that the matrix $\mathbf{R}$ is Hermitian.

If matrix $\mathbf{R}$ given by equation (2.54) is nonsingular for a given N, the input signal is said to be *persistently exciting* of order $N+1$. This means that the power spectral density $R_x(e^{j\omega})$ is different from zero at least at $N+1$ points

in the interval $0 < \omega \leq 2\pi$. It also means that a nontrivial Nth-order FIR filter (with at least one nonzero coefficient) cannot filter $x(k)$ to zero. Note that a nontrivial filter, with $x(k)$ as input, would require at least $N+1$ zeros in order to generate an output with all samples equal to zero. The absence of persistence of excitation implies the misbehavior of some adaptive algorithms [16], [17]. The definition of persistence of excitation is not unique, and it is algorithm dependent (see the book by Johnson [16] for further details).

From now on in this section, we discuss some properties of the correlation matrix related to its eigenvalues and eigenvectors. A number λ is an eigenvalue of the matrix $\mathbf{R}$, with a corresponding eigenvector $\mathbf{q}$, if and only if

$$det(\mathbf{R} - \lambda\mathbf{I}) = 0 \tag{2.55}$$

and

$$\mathbf{R}\mathbf{q} = \lambda\mathbf{q} \tag{2.56}$$

where $\mathbf{I}$ is the $(N+1)$ by $(N+1)$ identity matrix. Equation (2.55) is called characteristic equation of $\mathbf{R}$, and has $(N+1)$ solutions for λ. We denote the $(N+1)$ eigenvalues of $\mathbf{R}$ by $\lambda_0, \lambda_1, \ldots, \lambda_N$. Note also that for every value of λ, the vector $\mathbf{q} = \mathbf{0}$ satisfies equation (2.56), however we consider only those particular values of λ that are linked to a nonzero eigenvector $\mathbf{q}$.

Some important properties related to the eigenvalues and eigenvectors of $\mathbf{R}$, that will be useful in the following chapters, are listed below.

1. The eigenvalues of $\mathbf{R}^m$ are λ_i^m, for $i = 0, 1, 2, \ldots, N$.

   Proof:

   By premultiplying equation (2.56) by $\mathbf{R}^{m-1}$, we obtain

$$\begin{aligned} \mathbf{R}^{m-1}\mathbf{R}\mathbf{q}_i &= \mathbf{R}^{m-1}\lambda_i\mathbf{q}_i = \lambda_i\mathbf{R}^{m-2}\mathbf{R}\mathbf{q}_i \\ &= \lambda_i\mathbf{R}^{m-2}\lambda_i\mathbf{q}_i = \lambda_i^2\mathbf{R}^{m-3}\mathbf{R}\mathbf{q}_i \\ &= \cdots = \lambda_i^m\mathbf{q}_i \end{aligned} \tag{2.57}$$

   □

2. Suppose $\mathbf{R}$ has $N+1$ linearly independent eigenvectors $\mathbf{q}_i$; then if we form a matrix $\mathbf{Q}$ with columns consisting of the $\mathbf{q}_i$'s, it follows that

$$\mathbf{Q}^{-1}\mathbf{R}\mathbf{Q} = \begin{bmatrix} \lambda_0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & & \vdots \\ \vdots & 0 & \cdots & \vdots \\ \vdots & \vdots & & 0 \\ 0 & 0 & \cdots & \lambda_N \end{bmatrix} = \mathbf{\Lambda} \tag{2.58}$$

Proof:

$$\begin{aligned} \mathbf{R}\mathbf{Q} &= \mathbf{R}[\mathbf{q}_0\, \mathbf{q}_1 \cdots \mathbf{q}_N] = [\lambda_0\mathbf{q}_0\, \lambda_1\mathbf{q}_1 \cdots \lambda_N\mathbf{q}_N] \\ &= \mathbf{Q}\begin{bmatrix} \lambda_0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & & \vdots \\ \vdots & 0 & \cdots & \vdots \\ \vdots & \vdots & & 0 \\ 0 & 0 & \cdots & \lambda_N \end{bmatrix} = \mathbf{Q}\mathbf{\Lambda} \end{aligned}$$

Therefore, since $\mathbf{Q}$ is invertible because the $\mathbf{q}_i$'s are linearly independent, we can show that

$$\mathbf{Q}^{-1}\mathbf{R}\mathbf{Q} = \mathbf{\Lambda}$$

□

3. The nonzero eigenvectors $\mathbf{q}_0, \mathbf{q}_1, \ldots \mathbf{q}_N$ that correspond to different eigenvalues are linearly independent.

Proof:

If we form a linear combination of the eigenvectors such that

$$a_0\mathbf{q}_0 + a_1\mathbf{q}_1 + \cdots + a_N\mathbf{q}_N = \mathbf{0} \tag{2.59}$$

By multiplying the equation above by $\mathbf{R}$ we have

$$a_0\mathbf{R}\mathbf{q}_0 + a_1\mathbf{R}\mathbf{q}_1 + \cdots + a_N\mathbf{R}\mathbf{q}_N = a_0\lambda_0\mathbf{q}_0 + a_1\lambda_1\mathbf{q}_1 + \cdots + a_N\lambda_N\mathbf{q}_N \tag{2.60}$$

Now by multiplying equation (2.59) by λ_N and subtracting the result from equation (2.60), we obtain

$$a_0(\lambda_0 - \lambda_N)\mathbf{q}_0 + a_1(\lambda_1 - \lambda_N)\mathbf{q}_1 + \cdots + a_{N-1}(\lambda_{N-1} - \lambda_N)\mathbf{q}_{N-1} = \mathbf{0}$$

By repeating the steps above, i.e., multiplying the equation above by $\mathbf{R}$ in one instance and by λ_{N-1} on the other instance, and subtracting the results, it yields

$$\begin{aligned} &a_0(\lambda_0 - \lambda_N)(\lambda_0 - \lambda_{N-1})\mathbf{q}_0 + a_1(\lambda_1 - \lambda_N)(\lambda_1 - \lambda_{N-1})\mathbf{q}_1 \\ &+ \cdots + a_{N-2}(\lambda_{N-2} - \lambda_{N-1})\mathbf{q}_{N-2} = \mathbf{0} \end{aligned}$$

By repeating the same steps above several times, we end up with

$$a_0(\lambda_0 - \lambda_N)(\lambda_0 - \lambda_{N-1}) \cdots (\lambda_0 - \lambda_1)\mathbf{q}_0 = \mathbf{0}$$

Since we assumed $\lambda_0 \neq \lambda_1$, $\lambda_0 \neq \lambda_2$, ... $\lambda_0 \neq \lambda_N$, and $\mathbf{q}_0$ was assumed nonzero, then $a_0 = 0$.

The same line of thought can be used to show that $a_0 = a_1 = a_2 = \cdots = a_N = 0$ is the only solution for equation (2.59). Therefore, the eigenvectors corresponding to different eigenvalues are linearly independent.

□

Not all matrices are diagonalizable. A matrix of order $(N+1)$ is diagonalizable if it possesses $(N+1)$ linearly independent eigenvectors. A matrix with repeated eigenvalues can be diagonalized or not, depending on the linear dependency of the eigenvectors. A nondiagonalizable matrix is called defective [18].

4. Since $\mathbf{R}$ is a Hermitian matrix, i.e., $\mathbf{R}^H = \mathbf{R}$, its eigenvalues are real and equal to or greater than zero.

   Proof:

   First note that given an arbitrary complex vector $\mathbf{w}$, $(\mathbf{w}^H\mathbf{R}\mathbf{w})^H = \mathbf{w}^H\mathbf{R}^H(\mathbf{w}^H)^H = \mathbf{w}^H\mathbf{R}\mathbf{w}$. Therefore, $\mathbf{w}^H\mathbf{R}\mathbf{w}$ is a real number. Assume now that λ_i is an eigenvalue of $\mathbf{R}$ corresponding to the eigenvector $\mathbf{q}_i$, i.e., $\mathbf{R}\mathbf{q}_i = \lambda_i\mathbf{q}_i$. By premultiplying this equation by $\mathbf{q}_i^H$, it follows that

$$\mathbf{q}_i^H\mathbf{R}\mathbf{q}_i = \lambda_i\mathbf{q}_i^H\mathbf{q}_i = \lambda_i\|\mathbf{q}_i\|^2$$

   where the operation $\|\mathbf{a}\|^2 = |a_0|^2 + |a_1|^2 + \cdots + |a_N|^2$ is the Euclidean norm squared of the vector $\mathbf{a}$, that is always real. Since the term on the left hand

is also real and $\mathbf{R}$ is positive semidefinite, we can conclude that λ_i is real and nonnegative.

□

Note that $\mathbf{Q}$ is not unique since each $\mathbf{q}_i$ can be multiplied by an arbitrary nonzero constant, and the resulting vector continues to be an eigenvector. For practical reasons, we consider only normalized eigenvectors having length one, that is

$$\mathbf{q}_i^H \mathbf{q}_i = 1 \quad \text{for } i = 0, 1, \ldots, N \tag{2.61}$$

5. If $\mathbf{R}$ is a Hermitian matrix with different eigenvalues, the eigenvectors are orthogonal to each other. As a consequence, there is a diagonalizing matrix $\mathbf{Q}$ that is unitary, i.e., $\mathbf{Q}^H \mathbf{Q} = \mathbf{I}$.

   Proof:

   Given two eigenvalues λ_i and λ_j, it follows that

$$\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i$$

   and

$$\mathbf{R}\mathbf{q}_j = \lambda_j \mathbf{q}_j \tag{2.62}$$

   Using the fact that $\mathbf{R}$ is Hermitian and that λ_i and λ_j are real then

$$\mathbf{q}_i^H \mathbf{R} = \lambda_i \mathbf{q}_i^H$$

   and by multiplying this equation on the right by $\mathbf{q}_j$, we get

$$\mathbf{q}_i^H \mathbf{R}\mathbf{q}_j = \lambda_i \mathbf{q}_i^H \mathbf{q}_j$$

   Now by premultiplying equation (2.62) by $\mathbf{q}_i^H$, it follows that

$$\mathbf{q}_i^H \mathbf{R}\mathbf{q}_j = \lambda_j \mathbf{q}_i^H \mathbf{q}_j$$

   Therefore,

$$\lambda_i \mathbf{q}_i^H \mathbf{q}_j = \lambda_j \mathbf{q}_i^H \mathbf{q}_j$$

   Since $\lambda_i \neq \lambda_j$, it can be concluded that

$$\mathbf{q}_i^H \mathbf{q}_j = 0 \quad for\ i \neq j$$

   If we form matrix $\mathbf{Q}$ with normalized eigenvectors, matrix $\mathbf{Q}$ is a unitary matrix. □

An important result is that any Hermitian matrix $\mathbf{R}$ can be diagonalized by a suitable unitary matrix $\mathbf{Q}$, even if the eigenvalues of $\mathbf{R}$ are not distinct. The proof is omitted here and can be found in [18]. Therefore, for Hermitian matrices with repeated eigenvalues it is always possible to find a complete set of orthonormal eigenvectors.

A useful form to decompose a Hermitian matrix that results from the last property is

$$\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^H = \sum_{i=0}^{N} \lambda_i \mathbf{q}_i \mathbf{q}_i^H \tag{2.63}$$

that is known as *spectral decomposition*. From this decomposition, one can easily derive the following relation

$$\mathbf{w}^H\mathbf{R}\mathbf{w} = \sum_{i=0}^{N} \lambda_i \mathbf{w}^H \mathbf{q}_i \mathbf{q}_i^H \mathbf{w} = \sum_{i=0}^{N} \lambda_i |\mathbf{w}^H \mathbf{q}_i|^2 \tag{2.64}$$

In addition, since $\mathbf{q}_i = \lambda_i \mathbf{R}^{-1}\mathbf{q}_i$, the eigenvectors of a matrix and of its inverse coincide, whereas the eigenvalues are reciprocals of each other. As a consequence,

$$\mathbf{R}^{-1} = \sum_{i=0}^{N} \frac{1}{\lambda_i} \mathbf{q}_i \mathbf{q}_i^H \tag{2.65}$$

Another consequence of the unitary property of $\mathbf{Q}$ for Hermitian matrices is that any Hermitian matrix can be written in the form

$$\begin{aligned} \mathbf{R} &= \left[\sqrt{\lambda_0}\mathbf{q}_0 \; \sqrt{\lambda_1}\mathbf{q}_1 \cdots \sqrt{\lambda_N}\mathbf{q}_N\right] \begin{bmatrix} \sqrt{\lambda_0}\mathbf{q}_0^H \\ \sqrt{\lambda_1}\mathbf{q}_1^H \\ \vdots \\ \sqrt{\lambda_N}\mathbf{q}_N^H \end{bmatrix} \\ &= \mathbf{L}\mathbf{L}^H \end{aligned} \tag{2.66}$$

6. The sum of the eigenvalues of $\mathbf{R}$ is equal to the trace of $\mathbf{R}$, and the product of the eigenvalues of $\mathbf{R}$ is equal to the determinant of $\mathbf{R}$.

   Proof:

$$tr[\mathbf{Q}^{-1}\mathbf{R}\mathbf{Q}] = tr[\mathbf{\Lambda}]$$

where, $tr[\mathbf{A}] = \sum_{i=0}^{N} a_{ii}$. Since $tr[\mathbf{A}'\mathbf{A}] = tr[\mathbf{A}\mathbf{A}']$, we have

$$tr[\mathbf{Q}^{-1}\mathbf{R}\mathbf{Q}] = tr[\mathbf{R}\mathbf{Q}\mathbf{Q}^{-1}] = tr[\mathbf{R}\mathbf{I}] = tr[\mathbf{R}] = \sum_{i=0}^{N} \lambda_i$$

Also $det[\mathbf{Q}^{-1}\,\mathbf{R}\,\mathbf{Q}] = det[\mathbf{R}]\,det[\mathbf{Q}]\,det[\mathbf{Q}^{-1}] = det[\mathbf{R}] = det[\mathbf{\Lambda}] = \prod_{i=0}^{N} \lambda_i$

□

7. The Rayleigh's quotient defined as

$$\mathcal{R} = \frac{\mathbf{w}^H \mathbf{R} \mathbf{w}}{\mathbf{w}^H \mathbf{w}} \tag{2.67}$$

of a Hermitian matrix is bounded by the minimum and maximum eigenvalues, i.e.,

$$\lambda_{min} \leq \mathcal{R} \leq \lambda_{max} \tag{2.68}$$

where the minimum and maximum values are reached when the vector $\mathbf{w}$ is chosen to be the eigenvector corresponding to the minimum and maximum eigenvalues, respectively.

Proof:

Suppose $\mathbf{w} = \mathbf{Q}\mathbf{w}'$, where $\mathbf{Q}$ is the matrix that diagonalizes $\mathbf{R}$, then

$$\begin{aligned} \mathcal{R} &= \frac{\mathbf{w}'^H \mathbf{Q}^H \mathbf{R} \mathbf{Q} \mathbf{w}'}{\mathbf{w}'^H \mathbf{Q}^H \mathbf{Q} \mathbf{w}'} \\ &= \frac{\mathbf{w}'^H \mathbf{\Lambda} \mathbf{w}'}{\mathbf{w}'^H \mathbf{w}'} \\ &= \frac{\sum_{i=0}^{N} \lambda_i w_i'^2}{\sum_{i=0}^{N} w_i'^2} \end{aligned}$$

It is then easy to show that the minimum value for the equation above occurs when $w_i = 0$ for $i \neq j$ and λ_j is the smallest eigenvalue. Identically, the maximum value for $\mathcal{R}$ occurs when $w_i = 0$ for $i \neq l$, where λ_l is the largest eigenvalue.

□

There are several ways to define the norm of a matrix. In this book the norm of a matrix $\mathbf{R}$, denoted by $||\mathbf{R}||$, is defined by

$$||\mathbf{R}||^2 = \max_{\mathbf{w} \neq 0} \frac{||\mathbf{R}\mathbf{w}||^2}{||\mathbf{w}||^2} = \max_{\mathbf{w} \neq 0} \frac{\mathbf{w}^H \mathbf{R}^H \mathbf{R} \mathbf{w}}{\mathbf{w}^H \mathbf{w}} \tag{2.69}$$

Note that the norm of $\mathbf{R}$ is a measure of how a vector $\mathbf{w}$ grows in magnitude, when it is multiplied by $\mathbf{R}$.

When the matrix $\mathbf{R}$ is Hermitian, the norm of $\mathbf{R}$ is easily obtained by using the results of equations (2.57) and (2.68). The result is

$$||\mathbf{R}|| = \lambda_{max} \tag{2.70}$$

where λ_{max} is the maximum eigenvalue of $\mathbf{R}$.

A common problem that we encounter in adaptive filtering is the solution of a system of linear equations such as

$$\mathbf{Rw} = \mathbf{p} \tag{2.71}$$

In case there is an error in the vector $\mathbf{p}$, originated by quantization or estimation, how does it affect the solution of the system of linear equations? For a positive definite Hermitian matrix $\mathbf{R}$ it can be shown [18] that the relative error in the solution of the above linear system of equations is bounded by

$$\frac{\|\Delta\mathbf{w}\|}{\|\mathbf{w}\|} \leq \frac{\lambda_{max}}{\lambda_{min}} \frac{\|\Delta\mathbf{p}\|}{\|\mathbf{p}\|} \tag{2.72}$$

where λ_{max} and λ_{min} are the maximum and minimum values of the eigenvalues of $\mathbf{R}$, respectively. The ratio $\lambda_{max}/\lambda_{min}$ is called condition number of a matrix, that is

$$C = \frac{\lambda_{max}}{\lambda_{min}} = \|\mathbf{R}\|\|\mathbf{R}^{-1}\| \tag{2.73}$$

The value of C influences the convergence behavior of a number of adaptive filtering algorithms, as will be seen in the following chapters. Large values of C indicate that the matrix $\mathbf{R}$ is ill-conditioned and that errors introduced by the manipulation of $\mathbf{R}$ may be largely amplified. When $C = 1$ the matrix is perfectly conditioned. In case $\mathbf{R}$ represents the correlation matrix of the input signal of an adaptive filter, with the input vector composed by uncorrelated elements of a delay line (see Fig. 2.1.b, and the discussions around it), then $C = 1$.

Example 2.1

Suppose the input signal vector is composed by a delay line with a single input signal, i.e.,

$$\mathbf{x}(k) = [x(k)x(k-1)\ldots x(k-N)]^T$$

Given the following input signals:

(a)

$$x(k) = n(k)$$

(b)

$$x(k) = a\cos\omega_0 k + n(k)$$

(c)

$$x(k) = \sum_{i=0}^{M} b_i n(k-i)$$

(d)

$$x(k) = -a_1 x(k-1) + n(k)$$

(e)

$$x(k) = ae^{j(\omega_0 k + n(k))}$$

where $n(k)$ is a white noise with zero mean and variance σ_n^2.

Calculate the autocorrelation matrix $\mathbf{R}$ for $N = 3$.

Solution:

(a) In this case, we have that $E[x(k)x(k-l)] = \sigma_n^2 \delta(l)$, where $\delta(l)$ denotes an impulse sequence. Therefore,

$$\mathbf{R} = E[\mathbf{x}(k)\mathbf{x}^T(k)] = \sigma_n^2 \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

(b) In this example, $n(k)$ is zero mean and uncorrelated with the deterministic cosine. The autocorrelation function can then be expressed as

$$\begin{aligned} r(l) &= E[a^2 \cos(\omega_0 k)\cos(\omega_0 k - \omega_0 l) + n(k)n(k-l)] \\ &= a^2 E[\cos(\omega_0 k)\cos(\omega_0 k - \omega_0 l)] + \sigma_n^2 \delta(l) \\ &= \frac{a^2}{2}[\cos(\omega_0 l) - \cos(2\omega_0 k - \omega_0 l)] + \sigma_n^2 \delta(l) \end{aligned}$$

where $\delta(l)$ again denotes an impulse sequence.

(c) By exploring the fact that $n(k)$ is a white noise, we can perform the following simplifications:

$$\begin{aligned} r(l) &= E[x(k)x(k-l)] = E[\sum_{j=0}^{M-l}\sum_{i=0}^{M} b_i b_j n(k-i)n(k-l-j)] \\ &= \sum_{j=0}^{M-l} b_j b_{l+j} E[n^2(k-l-j)] = \sigma_n^2 \sum_{j=0}^{M} b_j b_{l+j} \\ &\quad 0 \le l + j \le M \end{aligned}$$

For $M = 3$, the correlation matrix has the following form

$$\mathbf{R} = \sigma_n^2 \begin{bmatrix} \sum_{i=0}^{3} b_i^2 & \sum_{i=0}^{2} b_i b_{i+1} & \sum_{i=0}^{1} b_i b_{i+2} & b_0 b_3 \\ \sum_{i=0}^{2} b_i b_{i+1} & \sum_{i=0}^{3} b_i^2 & \sum_{i=0}^{2} b_i b_{i+1} & \sum_{i=0}^{1} b_i b_{i+2} \\ \sum_{i=0}^{1} b_i b_{i+2} & \sum_{i=0}^{2} b_i b_{i+1} & \sum_{i=0}^{3} b_i^2 & \sum_{i=0}^{2} b_i b_{i+1} \\ b_0 b_3 & \sum_{i=0}^{1} b_i b_{i+2} & \sum_{i=0}^{2} b_i b_{i+1} & \sum_{i=0}^{3} b_i^2 \end{bmatrix}$$

(d) By solving the difference equation, we can obtain the correlation between $x(k)$ and $x(k-l)$, that is

$$x(k) = (-a_1)^l x(k-l) + \sum_{j=0}^{l-1} (-a_1)^j n(k-j)$$

Multiplying $x(k-l)$ on both sides of the equation above and taking the expected value of the result, we obtain

$$E[x(k)x(k-l)] = (-a_1)^l E[x^2(k-l)]$$

since $x(k-l)$ is independent of $n(k-j)$ for $j \leq l-1$.

For $l = 0$, just calculate $x^2(k)$ and apply the expectation operation to the result. The partial result is

$$E[x^2(k)] = a_1^2 E[x^2(k-1)] + E[n^2(k)]$$

therefore,

$$E[x^2(k)] = \frac{\sigma_n^2}{1 - a_1^2}$$

The elements of $\mathbf{R}$ are then given by

$$r(l) = \frac{(-a_1)^{|l|}}{1 - a_1^2} \sigma_n^2$$

(e) In this case, we are interested in calculating the autocorrelation of a complex sequence, that is

$$\begin{aligned} r(l) &= E[x(k)x^*(k-l)] \\ &= a^2 E[e^{-j(\omega l - n(k) + n(k-l))}] = a^2 e^{-j(\omega l)} \delta(l) \end{aligned}$$

where in the second equality it was considered the fact that we have two exponential functions with white noise as exponents. These exponentials are nonorthogonal only if $l = 0$.

□

In the remaining part of this chapter and in the following chapters, only real signals will be addressed, in order to keep the notation simple. The derivations of the adaptive filtering algorithms for complex signals are usually straightforward, and are left as exercises.

2.4 WIENER FILTER

One of the most widely used objective function in adaptive filtering is the mean-square error (MSE) defined as

$$F(e(k)) = \xi(k) = E[e^2(k)] = E[d^2(k) - 2d(k)y(k) + y^2(k)] \tag{2.74}$$

Suppose the adaptive filter consists of a linear combiner, i.e., the output signal is composed by a linear combination of signals coming from an array as depicted in Fig. 2.1.a. In this case,

$$y(k) = \sum_{i=0}^{N} w_i(k)x_i(k) = \mathbf{w}^T(k)\mathbf{x}(k) \tag{2.75}$$

where $\mathbf{x}(k) = [x_0(k)x_1(k)\ldots x_N(k)]^T$ and $\mathbf{w}(k) = [w_0(k)w_1(k)\ldots w_N(k)]^T$ are the input signal and the adaptive filter coefficient vectors, respectively.

In many applications, each element of the input signal vector consists of a delayed version of the same signal, that is: $x_0(k) = x(k), x_1(k) = x(k-1), \ldots, x_N(k) = x(k-N)$. Note that in this case signal $y(k)$ is the result of applying an FIR filter to the input signal $x(k)$. Since most of the analyses and algorithms presented in this book apply equally to the linear combiner and the FIR filter cases, we will consider the latter case throughout the rest of the book. The main reason for this decision is that the fast algorithms for the recursive least-squares solution, to be discussed in the forthcoming chapters, explore the fact that the input signal vector consists of the output of a delay line with a single input signal, and, as a consequence, are not applicable to the linear combiner case.

The most straightforward realization for the adaptive filter is through the direct form FIR structure as illustrated in Fig. 2.1.b, with the output given by

$$y(k) = \sum_{i=0}^{N} w_i(k)x(k-i) = \mathbf{w}^T(k)\mathbf{x}(k) \tag{2.76}$$

where $\mathbf{x}(k) = [x(k)x(k-1)\ldots x(k-N)]^T$, and $\mathbf{w}(k) = [w_0(k)w_1(k)\ldots w_N(k)]^T$ are the input and tap-weight vectors, respectively.

In both the linear combiner and FIR filter cases, the objective function can be rewritten as

$$\begin{aligned} E[e^2(k)] &= \xi(k) \\ &= E[d^2(k) - 2d(k)\mathbf{w}^T(k)\mathbf{x}(k) + \mathbf{w}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}(k)] \\ &= E[d^2(k)] - 2E[d(k)\mathbf{w}^T(k)\mathbf{x}(k)] + E[\mathbf{w}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}(k)] \end{aligned} \tag{2.77}$$

For a filter with fixed coefficients the MSE function is given by

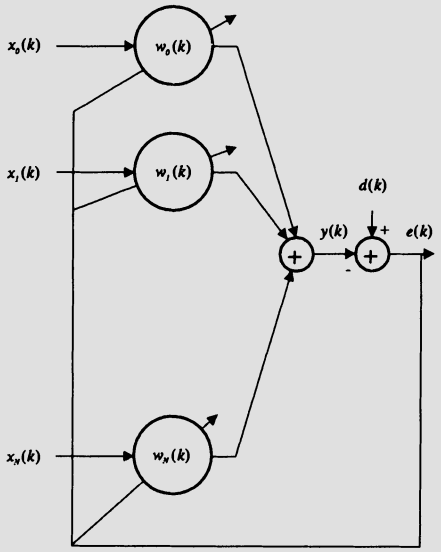
$$\begin{aligned} \xi &= E[d^2(k)] - 2\mathbf{w}^T E[d(k)\mathbf{x}(k)] + \mathbf{w}^T E[\mathbf{x}(k)\mathbf{x}^T(k)]\mathbf{w} \\ &= E[d^2(k)] - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w} \end{aligned} \tag{2.78}$$

where $\mathbf{p} = E[d(k)\mathbf{x}(k)]$ is the cross-correlation vector between the desired and input signals, and $\mathbf{R} = E[\mathbf{x}(k)\mathbf{x}^T(k)]$ is the input signal correlation matrix. As can be noted, the objective function ξ is a quadratic function of the tap-weight coefficients which would allow a straightforward solution for $\mathbf{w}$, if vector $\mathbf{p}$ and matrix $\mathbf{R}$ are known. Note that matrix $\mathbf{R}$ corresponds to the Hessian matrix of the objective function defined in the previous chapter.

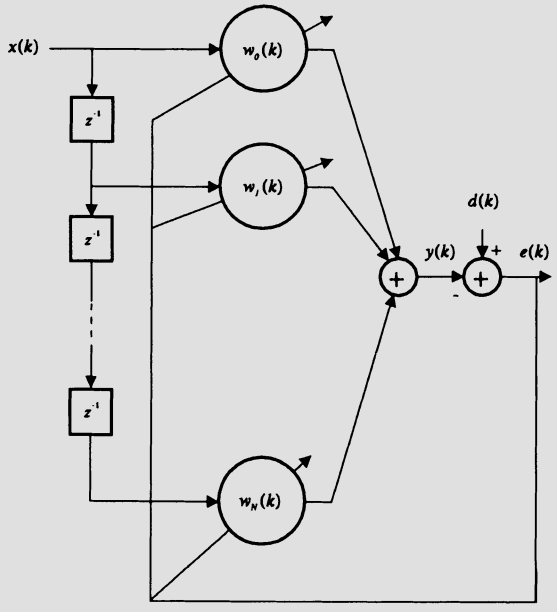
If the adaptive filter is implemented through an IIR filter, the objective function is a nonquadratic function of the filter parameters, turning the minimization problem much more difficult. Local minima are likely to exist, rendering some solutions obtained by gradient-based algorithms unacceptable. Despite its disadvantages, adaptive IIR filters are needed in a number of applications where the order of a suitable FIR filter is too high. Typical applications include data equalization in communication channels and cancellation of acoustic echo.

The gradient vector of the MSE function related to the filter tap-weight coefficients is given by

$$\begin{aligned} \mathbf{g}_{\mathbf{w}} &= \frac{\partial \xi}{\partial \mathbf{w}} = [\frac{\partial \xi}{\partial w_0}\frac{\partial \xi}{\partial w_1}\cdots\frac{\partial \xi}{\partial w_N}]^T \\ &= -2\mathbf{p} + 2\mathbf{R}\mathbf{w} \end{aligned} \tag{2.79}$$



(a)



(b)

Figure 2.1 (a) Linear combiner; (b) Adaptive FIR filter.

By equating the gradient vector to zero and assuming $\mathbf{R}$ is nonsingular, the optimal values for the tap-weight coefficients that minimizes the objective function can be evaluated as follows:

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p} \tag{2.80}$$

This solution is called the Wiener solution. Unfortunately, in practice, precise estimations of $\mathbf{R}$ and $\mathbf{p}$ are not available. When the input and the desired signals are ergodic, one is able to use time averages to estimate $\mathbf{R}$ and $\mathbf{p}$, what is implicitly performed by most adaptive algorithms.

If we replace the optimal solution for $\mathbf{w}$ in the MSE expression, we can calculate the minimum MSE provided by the Wiener solution:

$$\begin{aligned} \xi_{min} &= E[d^2(k)] - 2\mathbf{w}_o^T\mathbf{p} + \mathbf{w}_o^T\mathbf{R}\mathbf{R}^{-1}\mathbf{p} \\ &= E[d^2(k)] - \mathbf{w}_o^T\mathbf{p} \end{aligned} \tag{2.81}$$

The equation above indicates that the optimal set of parameters removes part of the power of the desired signal through the cross-correlation between $x(k)$ and $d(k)$, assuming both signals stationary. If the reference signal and the input signal are orthogonal, the optimal coefficients are equal to zero and the minimum MSE is $E[d^2(k)]$. This result is expected since nothing can be done with the parameters in order to minimize the MSE if the input signal carries no information about the desired signal. In this case, if any of the taps is nonzero, it would only increase the MSE.

An important property of the Wiener filter can be deduced if we analyze the gradient of the error surface at the optimal solution. The gradient vector can be expressed as follows:

$$\mathbf{g}_\mathbf{w} = \frac{\partial E[e^2(k)]}{\partial \mathbf{w}} = E[2e(k)\frac{\partial e(k)}{\partial \mathbf{w}}] = -E[2e(k)\mathbf{x}(k)] \tag{2.82}$$

With the coefficients set at their optimal values, i.e., at the Wiener solution, the gradient vector is equal to zero, implying that

$$E[e(k)\mathbf{x}(k)] = \mathbf{0} \tag{2.83}$$

or

$$E[e(k)x(k-i)] = 0 \tag{2.84}$$

for $i = 0, 1, \ldots, N$. This means that the error signal is orthogonal to the elements of the input signal vector. In case either the error or the input signal has zero mean, the orthogonality property implies that $e(k)$ and $x(k)$ are uncorrelated.

The orthogonality principle also applies to the correlation between the output signal $y(k)$ and the error $e(k)$, when the tap weights are given by $\mathbf{w} = \mathbf{w}_o$. By premultiplying the equation (2.83) by $\mathbf{w}_o^T$, the desired result follows, e.g.,

$$E[e(k)\mathbf{w}_o^T\mathbf{x}(k)] = E[e(k)y(k)] = 0 \tag{2.85}$$

2.5 MEAN-SQUARE ERROR SURFACE

The mean-square error is a quadratic function of the parameters $\mathbf{w}$. Assuming a given fixed $\mathbf{w}$, the MSE is not a function of time and can be expressed as

$$\xi = \sigma_d^2 - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w} \tag{2.86}$$

where σ_d^2 is the variance of $d(k)$ assuming it has zero-mean. The MSE is a quadratic function of the tap-weights forming a hyperparaboloid surface. The MSE surface is convex and has only positive values. For two weights, the surface is a paraboloid. Fig. 2.2 illustrates the MSE surface for a numerical example where $\mathbf{w}$ has two coefficients. If the MSE surface is intersected by a plane parallel to the $\mathbf{w}$ plane, placed at a level superior to ξ_{min}, the intersection consists of an ellipse representing equal MSE contours as depicted in Fig. 2.3. Note that in this figure we showed three distinct ellipses, corresponding to different levels of MSE. The ellipses of constant MSE are all concentric.

In order to understand the properties of the MSE surface, it is convenient to define a translated coefficient vector as follows:

$$\Delta\mathbf{w} = \mathbf{w} - \mathbf{w}_o \tag{2.87}$$

The MSE can be expressed as a function of $\Delta\mathbf{w}$ as follows:

$$\begin{aligned}
\xi &= \sigma_d^2 - \mathbf{w}_o^T\mathbf{p} + \mathbf{w}_o^T\mathbf{p} - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w} \\
&= \xi_{min} - \Delta\mathbf{w}^T\mathbf{p} - \mathbf{w}^T\mathbf{R}\mathbf{w}_o + \mathbf{w}^T\mathbf{R}\mathbf{w} \\
&= \xi_{min} - \Delta\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\Delta\mathbf{w} \\
&= \xi_{min} - \mathbf{w}_o^T\mathbf{R}\Delta\mathbf{w} + \mathbf{w}^T\mathbf{R}\Delta\mathbf{w} \\
&= \xi_{min} + \Delta\mathbf{w}^T\mathbf{R}\Delta\mathbf{w}
\end{aligned} \tag{2.88}$$

The corresponding error surface contours are depicted in Fig. 2.4.

By employing the diagonalized form of $\mathbf{R}$, the last equation can be rewritten as follows:

$$\xi = \xi_{min} + \Delta\mathbf{w}^T\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T\Delta\mathbf{w}$$

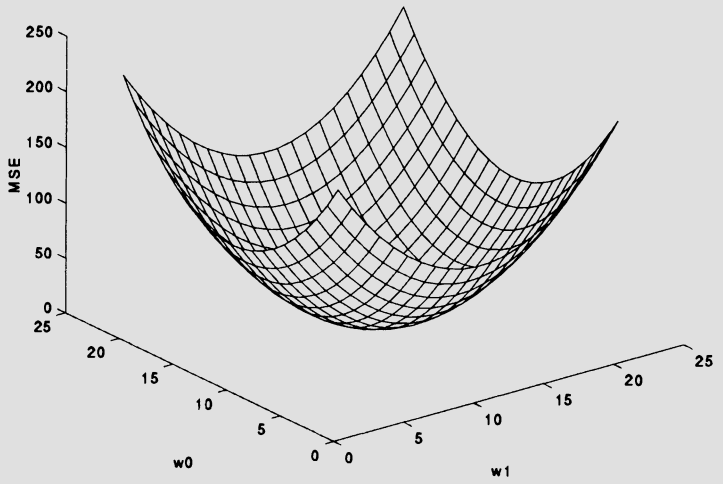


Figure 2.2 Mean-square error surface.

$$\begin{aligned} &= \xi_{min} + \mathbf{v}^T \mathbf{\Lambda} \mathbf{v} \\ &= \xi_{min} + \sum_{i=0}^{N} \lambda_i v_i^2 \end{aligned} \tag{2.89}$$

where $\mathbf{v} = \mathbf{Q}^T \Delta \mathbf{w}$ are the rotated parameters.

The above form for representing the MSE surface is an uncoupled form in the sense that each component of the gradient vector of the MSE with respect to the rotated parameters is a function of a single parameter, that is

$$\nabla_{\mathbf{v}} \xi = [2\lambda_0 v_0 \;\; 2\lambda_1 v_1 \;\; \ldots \;\; 2\lambda_N v_N]^T$$

This property means that if all v_i's are zero except one, the gradient direction coincides with the nonzero parameter axis. In other words, the rotated parameters represent the principal axes of the hyperellipse of constant MSE, as illustrated in Fig. 2.5. Note that since the rotated parameters are the result of the projection of the original parameter vector $\Delta \mathbf{w}$ on the eigenvectors $\mathbf{q}_i$ direction, it is straightforward to conclude that the eigenvectors represent the principal axes of the constant MSE hyperellipses.

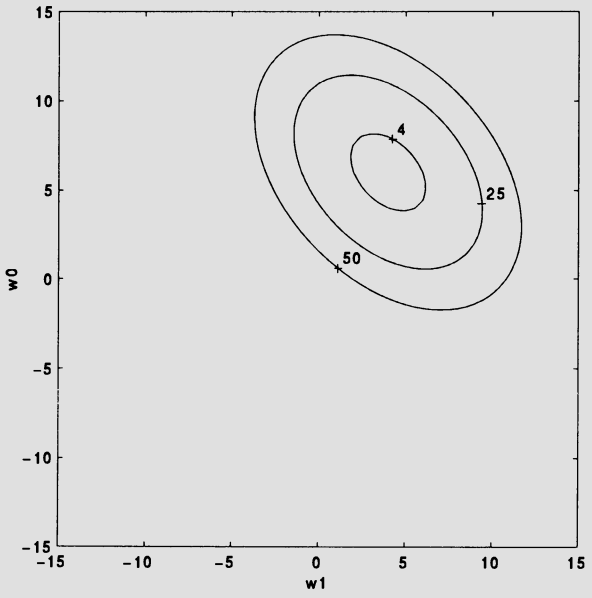


Figure 2.3 Contours of the MSE surface.

The matrix of second derivatives of ξ as related to the rotated parameters is $\mathbf{\Lambda}$. We can note that the gradient will be steeper in the principal axes corresponding to larger eigenvalues. This is the direction, in the two axes case, where the ellipse is narrow.

2.6 BIAS AND CONSISTENCY

The correct interpretation of the results obtained by the adaptive filtering algorithm requires the definitions of bias and consistency. An estimate is considered unbiased if the following condition is satisfied

$$E[\mathbf{w}(k)] = \mathbf{w}_o \tag{2.90}$$

The difference $E[\mathbf{w}(k)] - \mathbf{w}_o$ is called the bias in the parameter estimate.

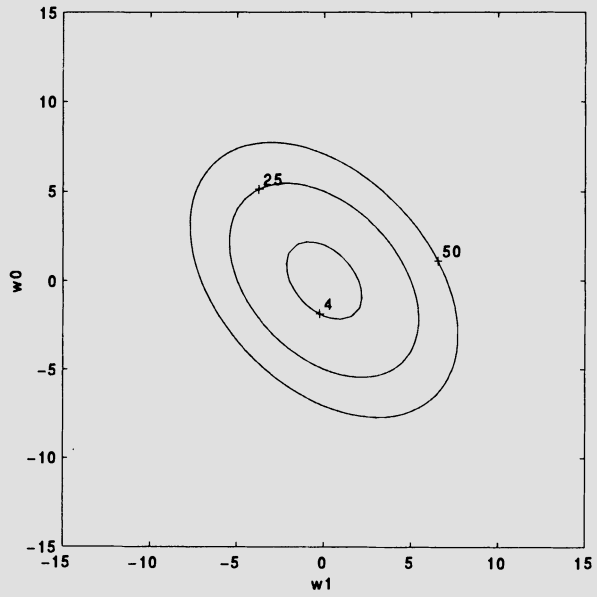


Figure 2.4 Translated contours of the MSE surface.

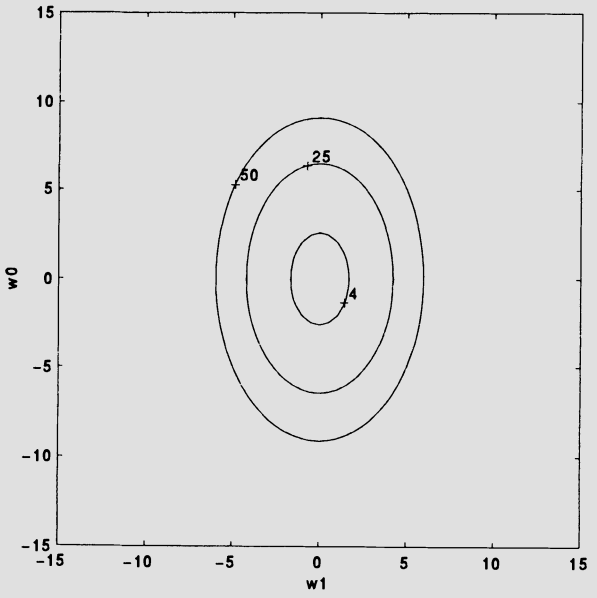


Figure 2.5 Rotated contours of the MSE surface.

An estimate is considered consistent if

$$\mathbf{w}(k) \rightarrow \mathbf{w}_o \ as \ k \rightarrow \infty \tag{2.91}$$

Note that since $\mathbf{w}(k)$ is a random variable, it is necessary to define in which sense the limit is taken. Usually, the limit with probability one is employed. In the case of identification, a system is considered identifiable if the given parameter estimates are consistent. For a more formal treatment on this subject refer to [17].

2.7 NEWTON ALGORITHM

In the context of the MSE minimization discussed in the previous section, the coefficient-vector updating using the Newton method is performed as follows:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu \mathbf{R}^{-1} \mathbf{g}_{\mathbf{w}}(k) \tag{2.92}$$

Assuming the true gradient and the matrix $\mathbf{R}$ are available, the coefficient-vector updating can be expressed as

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu \mathbf{R}^{-1}(-2\mathbf{p} + 2\mathbf{R}\mathbf{w}(k)) = (\mathbf{I} - 2\mu\mathbf{I})\mathbf{w}(k) + 2\mu\mathbf{w}_o \tag{2.93}$$

where if $\mu = 1/2$, the Wiener solution is reached in one step.

The Wiener solution can be approached using a Newton-like search algorithm, by updating the adaptive filter coefficients as follows:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu \hat{\mathbf{R}}^{-1}(k) \hat{\mathbf{g}}_{\mathbf{w}}(k) \tag{2.94}$$

where $\hat{\mathbf{R}}^{-1}(k)$ is an estimate of $\mathbf{R}^{-1}$ and $\hat{\mathbf{g}}_{\mathbf{w}}(k)$ is an estimate of $\mathbf{g}_{\mathbf{w}}$, both at instant k. The parameter μ is the convergence factor that regulates the convergence rate. Newton-based algorithms present, in general, fast convergence. However, the estimate of $\mathbf{R}^{-1}$ is computationally intensive and can become numerically unstable if special care is not taken. These factors made the steepest-descent-based algorithms more popular in adaptive filtering applications.

2.8 STEEPEST-DESCENT ALGORITHM

In order to get a practical feeling of a problem that is being solved using the steepest-descent algorithm, we assume that the optimal coefficient vector, i.e.,

the Wiener solution, is $\mathbf{w}_o$, and that the reference signal is not corrupted by measurement noise.

The main objective of the present section is to study the rate of convergence, the stability, and the steady-state behavior of an adaptive filter whose coefficients are updated through the steepest-descent algorithm. It is worth mentioning that the steepest-descent method can be considered an efficient gradient-type algorithm, in the sense that it works with the true gradient vector, and not with an estimate of it. Therefore, the performance of other gradient-type algorithms can at most be close to the performance of the steepest-descent algorithm. When the objective function is the MSE, the difficult task of obtaining the matrix $\mathbf{R}$ and the vector $\mathbf{p}$ impairs the steepest-descent algorithm from being useful in adaptive filtering applications. Its performance, however, serves as a comparison pattern for gradient-based algorithms.

The steepest-descent algorithm updates the coefficients in the following general form

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu \mathbf{g}_{\mathbf{w}}(k) \tag{2.95}$$

It is worth noting that several alternative gradient-based algorithms available replace $\mathbf{g}_{\mathbf{w}}(k)$ by an estimate $\hat{\mathbf{g}}_{\mathbf{w}}(k)$, and they differ in the way the gradient vector is estimated. The true gradient expression is given in equation (2.79) and, as can be noted, it depends on the vector $\mathbf{p}$ and the matrix $\mathbf{R}$, that are usually not available.

Substituting equation (2.79) in equation (2.95), we get

$$\mathbf{w}(k+1) = \mathbf{w}(k) - 2\mu\mathbf{R}\mathbf{w}(k) + 2\mu\mathbf{p} \tag{2.96}$$

Now some of the main properties related to the convergence behavior of the steepest-descent algorithm in stationary environment are described. First, an analysis is required to determine the influence of the convergence factor μ in the convergence behavior of the steepest-descent algorithm.

The error in the adaptive filter coefficients when compared to the Wiener solution is defined as

$$\Delta\mathbf{w}(k) = \mathbf{w}(k) - \mathbf{w}_o \tag{2.97}$$

The steepest-descent algorithm can then be described in an alternative way, that is:

$$\begin{aligned} \Delta\mathbf{w}(k+1) &= \Delta\mathbf{w}(k) - 2\mu(\mathbf{R}\mathbf{w}(k) - \mathbf{R}\mathbf{w}_o) \\ &= \Delta\mathbf{w}(k) - 2\mu\mathbf{R}\Delta\mathbf{w}(k) \\ &= (\mathbf{I} - 2\mu\mathbf{R})\,\Delta\mathbf{w}(k) \end{aligned} \tag{2.98}$$

where the relation $\mathbf{p} = \mathbf{R}\mathbf{w}_o$ (see equation (2.80)) was employed. It can be easily shown from the equation above that

$$\Delta\mathbf{w}(k+1) = (\mathbf{I} - 2\mu\mathbf{R})^{k+1}\Delta\mathbf{w}(0) \tag{2.99}$$

or

$$\mathbf{w}(k+1) = \mathbf{w}_o + (\mathbf{I} - 2\mu\mathbf{R})^{k+1}(\mathbf{w}(0) - \mathbf{w}_o) \tag{2.100}$$

The last equation premultiplied by $\mathbf{Q}^T$, where $\mathbf{Q}$ is the unitary matrix that diagonalizes $\mathbf{R}$ through a similarity transformation, yields

$$\begin{aligned}
\mathbf{Q}^T\Delta\mathbf{w}(k+1) &= (\mathbf{I} - 2\mu\mathbf{Q}^T\mathbf{R}\mathbf{Q})\mathbf{Q}^T\Delta\mathbf{w}(k) \\
&= \mathbf{v}(k+1) \\
&= (\mathbf{I} - 2\mu\mathbf{\Lambda})\mathbf{v}(k) \\
&= \begin{bmatrix} 1-2\mu\lambda_0 & 0 & \cdots & 0 \\ 0 & 1-2\mu\lambda_1 & & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & 1-2\mu\lambda_N \end{bmatrix} \mathbf{v}(k)
\end{aligned} \tag{2.101}$$

In the equation above $\mathbf{v}(k+1) = \mathbf{Q}^T\Delta\mathbf{w}(k+1)$ is the rotated coefficient-vector error. Using induction, equation (2.101) can be rewritten as

$$\begin{aligned}
\mathbf{v}(k+1) &= (\mathbf{I} - 2\mu\mathbf{\Lambda})^{k+1}\mathbf{v}(0) \\
&= \begin{bmatrix} (1-2\mu\lambda_0)^{k+1} & 0 & \cdots & 0 \\ 0 & (1-2\mu\lambda_1)^{k+1} & & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & (1-2\mu\lambda_N)^{k+1} \end{bmatrix} \mathbf{v}(0)
\end{aligned} \tag{2.102}$$

This equation shows that in order to guarantee the convergence of the coefficients, each element $1 - 2\mu\lambda_i$ must have an absolute value less than one. As a consequence, the convergence factor of the steepest-descent algorithm must be chosen in the range

$$0 < \mu < \frac{1}{\lambda_{max}} \tag{2.103}$$

where λ_{max} is the largest eigenvalue of $\mathbf{R}$. In this case, all the elements of the diagonal matrix in equation (2.101) tend to zero as $k \to \infty$, resulting in $\mathbf{v}(k+1) \to 0$ for large k.

The μ value in the range above guarantees that the coefficient vector approaches the optimum coefficient vector $\mathbf{w}_o$. It should be mentioned that if matrix $\mathbf{R}$ has large eigenvalue spread, the convergence speed of the coefficients will be primarily dependent on the value of the smallest eigenvalue. Note that the slowest decaying element in equation (2.101) is given by $(1 - 2\mu\lambda_{min})^{k+1}$.

The MSE presents a transient behavior during the adaptation process, that can be analyzed in a straightforward way if we employ the diagonalized version of $\mathbf{R}$. Recalling from equation (2.88) that

$$\xi(k) = \xi_{min} + \Delta\mathbf{w}^T(k)\mathbf{R}\Delta\mathbf{w}(k) \tag{2.104}$$

the MSE can then be simplified as follows:

$$\begin{aligned} \xi(k) &= \xi_{min} + \Delta\mathbf{w}^T(k)\mathbf{Q}\mathbf{\Lambda}\ \mathbf{Q}^T\Delta\mathbf{w}(k) \\ &= \xi_{min} + \mathbf{v}^T(k)\mathbf{\Lambda}\ \mathbf{v}(k) \\ &= \xi_{min} + \sum_{i=0}^{N} \lambda_i v_i^2(k) \end{aligned} \tag{2.105}$$

If we apply the result of equation (2.101) in equation (2.105), it can be shown that the following relation results

$$\begin{aligned} \xi(k) &= \xi_{min} + \mathbf{v}^T(k-1)(\mathbf{I} - 2\mu\mathbf{\Lambda})\mathbf{\Lambda}\ (\mathbf{I} - 2\mu\mathbf{\Lambda})\mathbf{v}(k-1) \\ &= \xi_{min} + \sum_{i=0}^{N} \lambda_i(1 - 2\mu\lambda_i)^{2k} v_i^2(0) \end{aligned} \tag{2.106}$$

The analyses presented in this section show that before the steepest-descent algorithm reaches the steady-state behavior, there is a transient period where the error is usually high and the coefficients are far from the Wiener solution. As can be seen from equation (2.101), in the case of the adaptive filter coefficients, the convergence will follow $(N+1)$ geometric decaying curves with ratios $r_{wi} = (1-2\mu\lambda_i)$. Each of these curves can be approximated by an exponential envelope with time constant τ_{wi} as follows [4]:

$$r_{wi} = e^{\frac{-1}{\tau_{wi}}} = 1 - \frac{1}{\tau_{wi}} + \frac{1}{2!\tau_{wi}^2} + \cdots \tag{2.107}$$

In general, r_{wi} is slightly smaller than one, specially in the cases of slowly decreasing modes that correspond to small values λ_i and μ. Therefore,

$$r_{wi} = (1 - 2\mu\lambda_i) \approx 1 - \frac{1}{\tau_{wi}} \tag{2.108}$$

then

$$\tau_{wi} \approx \frac{1}{2\mu\lambda_i}$$

for $i = 0, 1, \ldots, N$.

For the convergence of the MSE, the range of values of μ is the same to guarantee the convergence of the coefficients. In this case, due to the exponent $2k$ in equation (2.106), the geometric decaying curves have ratios given by $r_{ei} = (1 - 4\mu\lambda_i)$, that can be approximated by exponential envelopes with time constants given by

$$\tau_{ei} \approx \frac{1}{4\mu\lambda_i} \tag{2.109}$$

for $i = 0, 1, \ldots, N$. In the convergence of both the error and the coefficients, the time required for the convergence depends on the ratio of the eigenvalues of the input signal. Further discussions on convergence properties that apply to gradient-type algorithms can be found in Chapter 3.

Example 2.2

The matrix $\mathbf{R}$ and the vector $\mathbf{p}$ are known for a given experimental environment:

$$\mathbf{R} = \begin{bmatrix} 1 & 0.4045 \\ 0.4045 & 1 \end{bmatrix}$$

$$\mathbf{p} = [0 \ 0.2939]^T$$

$$E[d^2(k)] = 0.5$$

(a) Deduce the equation for the MSE.

(b) Choose a small value for μ, and starting the parameters at $[0 \ \ -2]^T$ plot the convergence path of the steepest-descent algorithm in the MSE surface.

(c) Repeat the previous item for the Newton algorithm.

Solution:

(a) The MSE function is given by

$$\begin{aligned}\xi &= E[d^2(k)] - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w} \\ &= \sigma_d^2 - 2[w_1 \ \ w_2]\begin{bmatrix} 0. \\ 0.2939 \end{bmatrix} + [w_1 \ \ w_2]\begin{bmatrix} 1 & 0.4045 \\ 0.4045 & 1 \end{bmatrix}\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}\end{aligned}$$

After performing the algebraic calculation we obtain the following result

$$\xi = 0.5 + w_1^2 + w_2^2 + 0.8090 w_1 w_2 - 0.5878 w_2$$

(b) The steepest-descent algorithm was applied to minimize the MSE using a convergence factor $\mu = 0.1/\lambda_{max}$, where $\lambda_{max} = 1.4045$. The convergence path of the algorithm in the MSE surface is depicted in Fig. 2.6. As can be noted, the path followed by the algorithm first approaches the main axis (eigenvector) corresponding to the smaller eigenvalue, and then follows toward the minimum in a direction increasingly aligned with this main axis.

(c) The Newton algorithm was also applied to minimize the MSE using a convergence factor $\mu = 0.1/\lambda_{max}$. The convergence path of the Newton algorithm in the MSE surface is depicted in Fig. 2.7. The Newton algorithm follows a straight path to the minimum.

□

2.9 APPLICATIONS REVISITED

In this section, we give a brief introduction to the typical applications where the adaptive filtering algorithms are required, including a discussion of where in the real world these applications are found. The main objective of this section is to illustrate how the adaptive filtering algorithms, in general, and the ones presented in the book, in particular, are applied to solve practical problems.

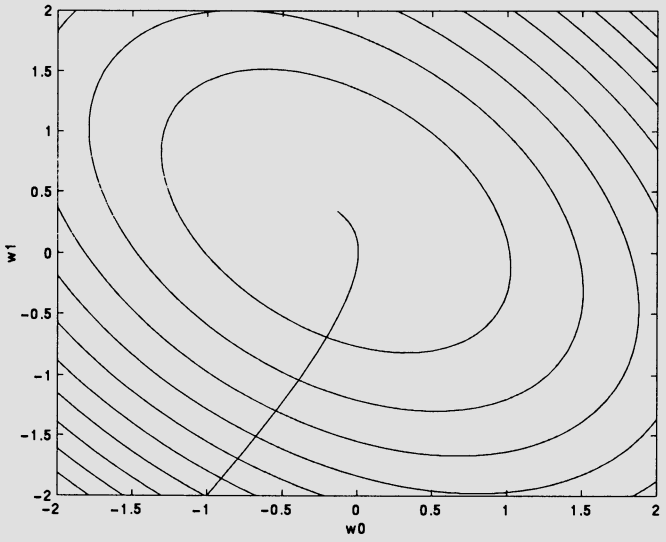


Figure 2.6 Convergence path of the steepest-descent algorithm.

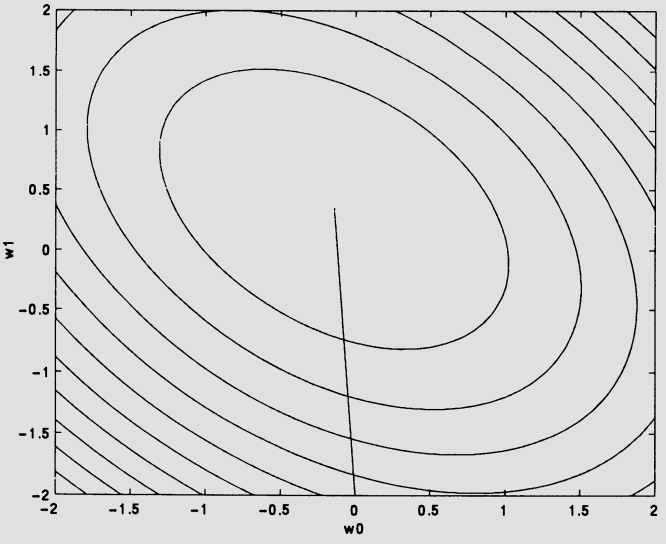


Figure 2.7 Convergence path of the Newton algorithm.

It should be noted that the detailed analysis of any particular application is beyond the scope of this book. Nevertheless, a number of specific references are given for the interested reader. The distinctive feature of each application is the way the adaptive filter input signal and the desired signal are chosen. Once these signals are determined, any known properties of them can be used to understand the expected behavior of the adaptive filter when attempting to minimize the chosen objective function (for example, the MSE, ξ).

2.9.1 System Identification

The typical set up of the system identification application is depicted in Fig. 2.8. A common input signal is applied to the unknown system and to the adaptive filter. Usually, the input signal is a wideband signal, in order to allow the adaptive filter to converge to a good model of the unknown system.

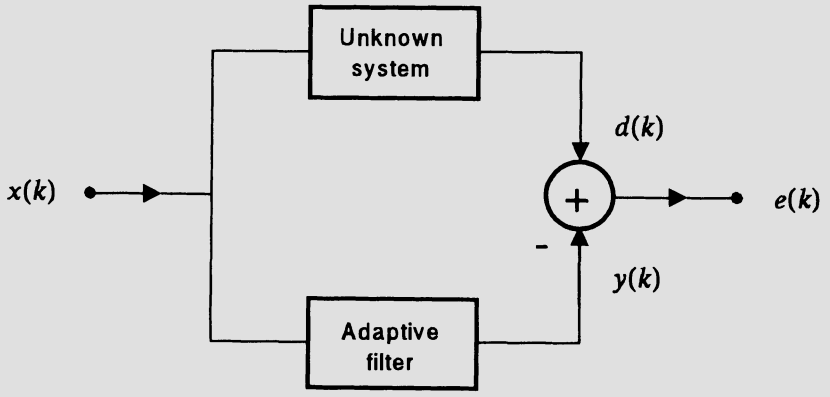


Figure 2.8 System identification.

Assume the unknown system has impulse response given by $h(k)$, for $k = 0, 1, 2, 3, \ldots, \infty$ and zero for $k < 0$. The error signal is then given by

$$\begin{aligned} e(k) &= d(k) - y(k) \\ &= \sum_{l=0}^{\infty} h(l)x(k-l) - \sum_{i=0}^{N} w_i(k)x(k-i) \end{aligned} \tag{2.110}$$

where $w_i(k)$ are the coefficients of the adaptive filter.

Assuming that $x(k)$ is a white noise, the MSE for a fixed $\mathbf{w}$ is given by

$$\begin{aligned}\xi &= E[(\mathbf{h}^T\mathbf{x}_\infty(k) - \mathbf{w}^T\mathbf{x}_{N+1}(k))^2] \\ &= E[\mathbf{h}^T\mathbf{x}_\infty(k)\mathbf{x}_\infty^T(k)\mathbf{h} - 2\mathbf{h}^T\mathbf{x}_\infty(k)\mathbf{x}_{N+1}^T(k)\mathbf{w} + \mathbf{w}^T\mathbf{x}_{N+1}(k)\mathbf{x}_{N+1}^T(k)\mathbf{w}] \\ &= \sigma_x^2\sum_{i=0}^{\infty} h^2(i) - 2\sigma_x^2\mathbf{h}^T\begin{bmatrix}\mathbf{I}_{N+1}\\ \mathbf{0}\end{bmatrix}\mathbf{w} + \mathbf{w}^T\mathbf{R}_{N+1}\mathbf{w} \qquad (2.111)\end{aligned}$$

where $\mathbf{x}_\infty(k)$ and $\mathbf{x}_{N+1}(k)$ are the input signal vector with infinite and finite lengths, respectively.

By calculating the derivative of ξ with respect to the coefficients of the adaptive filter, it follows that

$$\mathbf{w}_o = \mathbf{h}_{N+1} \qquad (2.112)$$

where

$$\mathbf{h}_{N+1}^T = \mathbf{h}^T\begin{bmatrix}\mathbf{I}_{N+1}\\ \mathbf{0}\end{bmatrix} \qquad (2.113)$$

If the input signal is a white noise, the best model for the unknown system is a system whose impulse response coincides with the $N+1$ first samples of the unknown system impulse response. In the cases where the impulse response of the unknown system is of finite length and the adaptive filter is of sufficient order (i.e., it has enough number of parameters), the MSE becomes zero if there is no measurement noise (or channel noise). In practical applications the measurement noise is unavoidable, and if it is uncorrelated with the input signal, the expected value of the adaptive filter coefficients will coincide with the unknown-system impulse response samples. The output error will of course be the measurement noise. We can observe that the measurement noise introduces a variance in the estimates of the unknown system parameters.

Some real world applications of the system identification scheme include modeling of multipath communication channels [30], control systems [23], seismic exploration [31], and cancellation of echo caused by hybrids in some communication systems [32]-[36], just to mention a few.

2.9.2 Signal Enhancement

In the signal enhancement application, the reference signal consists of a desired signal $x(k)$ that is corrupted by an additive noise $n_1(k)$. The input signal of

the adaptive filter is a noise signal $n_2(k)$ that is correlated with the interference signal $n_1(k)$, but uncorrelated with $x(k)$. Fig. 2.9 illustrates the configuration of the signal enhancement application. In practice, this configuration is found in acoustic echo cancellation for auditoriums [39], hearing aids, noise cancellation in hydrophones [38], cancelling of power line interference in electrocardiography [23], and in other applications. The cancelling of echo caused by the hybrid in some communication systems can also be considered a signal enhancement problem [23].

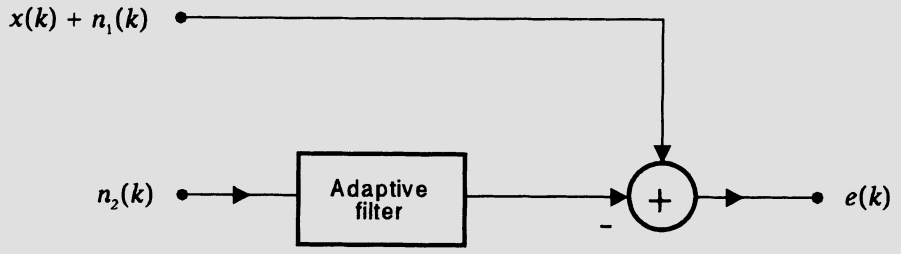


Figure 2.9 Signal enhancement ($n_1(k)$ and $n_2(k)$ are noise signals correlated to each other).

In this application, the error signal is given by

$$e(k) = x(k) + n_1(k) - \sum_{l=0}^{N} w_l n_2(k-l) = x(k) + n_1(k) - y(k) \tag{2.114}$$

The resulting MSE is then given by

$$E[e^2(k)] = E[x^2(k)] + E[(n_1(k) - y(k))^2] \tag{2.115}$$

where it was assumed that $x(k)$ is uncorrelated with $n_1(k)$ and $n_2(k)$. The equation above shows that if the adaptive filter, having $n_2(k)$ as the input signal, is able to perfectly predict the signal $n_1(k)$, the minimum MSE is given by

$$\xi_{min} = E[x^2(k)] \tag{2.116}$$

where the error signal, in this situation, is the desired signal $x(k)$.

The effectiveness of the signal enhancement scheme depends on the high correlation between $n_1(k)$ and $n_2(k)$. In some applications, it is useful to include a delay of L samples in the reference signal or in the input signal, such that their relative delay yields a maximum cross-correlation between $y(k)$ and $n_1(k)$, reducing the MSE. This delay provides a kind of synchronization between the signals involved. An example exploring this issue will be presented in the following chapters.

2.9.3 Signal Prediction

In the signal prediction application, the adaptive filter input consists of a delayed version of the desired signal as illustrated in Fig. 2.10. The MSE is given by

$$\xi = E[(x(k) - \mathbf{w}^T\mathbf{x}(k-L))^2] \tag{2.117}$$

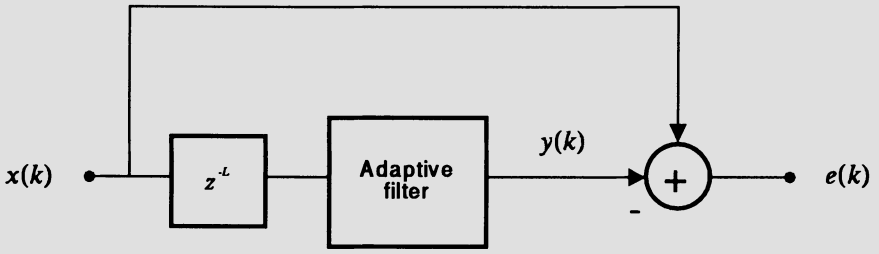


Figure 2.10 Signal prediction.

The minimization of the MSE leads to an FIR filter, whose coefficients are the elements of $\mathbf{w}$. This filter is able to predict the present sample of the input signal using as information old samples such as $x(k-L), x(k-L-1), \ldots, x(k-L-N)$. The resulting FIR filter can then be considered a model for the signal $x(k)$ when the MSE is small. The minimum MSE is given by

$$\xi_{min} = r(0) - \mathbf{w}_o^T \begin{bmatrix} r(L) \\ r(L+1) \\ \cdot \\ \cdot \\ \cdot \\ r(L+N) \end{bmatrix} \tag{2.118}$$

where $\mathbf{w}_o$ is the optimum predictor coefficient vector and $r(l) = E[x(k)x(k-l)]$ for a stationary process.

A typical predictor's application is in linear prediction coding of speech signals [37], where the predictor's task is to estimate the speech parameters. These parameters $\mathbf{w}$ are part of the coding information that is transmitted or stored along with other informations inherent to the speech characteristics, such as pitch period, among others.

The adaptive signal predictor is also used for adaptive line enhancement (ALE), where the input signal is a narrowband signal (predictable) added to a wideband

signal. After convergence the predictor output will be an enhanced version of the narrowband signal.

Yet another application of the signal predictor is the suppression of narrowband interference in a wideband signal. The input signal, in this case, has the same general characteristics of the ALE. However, we are now interested in removing the narrowband interferer. For such an application, the output signal of interest is the error signal [39].

2.9.4 Channel Equalization

As can be seen from Fig. 2.11, channel equalization or inverse filtering consists of estimating a transfer function to compensate for the linear distortion caused by the channel. From another point of view, the objective is to force a prescribed dynamic behavior for the cascade of the channel (unknown system) and the adaptive filter, determined by the input signal. The first interpretation is more appropriate in communications, where the information is transmitted through dispersive channels [29], [35]. The second interpretation is appropriate for control applications, where the inverse filtering scheme generates control signals to be used in the unknown system [23].

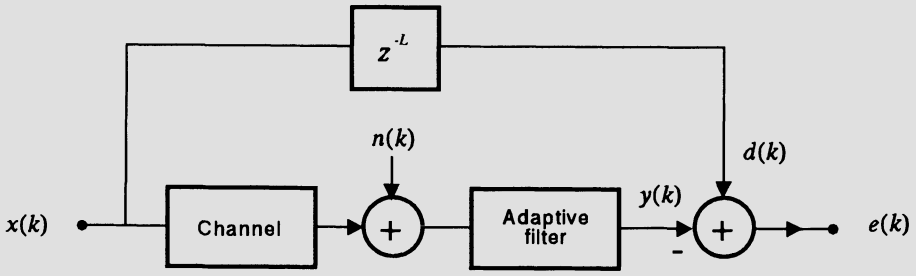


Figure 2.11 Channel equalization.

In the ideal situation, where $n(k) = 0$ and the equalizer has sufficient order, the error signal is zero if

$$W(z)H(z) = z^{-L} \tag{2.119}$$

where $W(z)$ and $H(z)$ are the equalizer and unknown system transfer functions, respectively. Therefore, the ideal equalizer has the following transfer function

$$W(z) = \frac{z^{-L}}{H(z)} \tag{2.120}$$

From the equation above, we can conclude that if $H(z)$ is an IIR transfer function with nontrivial numerator and denominator polynomials, $W(z)$ will also be IIR. If $H(z)$ is an all-pole model, $W(z)$ is FIR. If $H(z)$ is an all-zero model, $W(z)$ is an all-pole transfer function.

By applying the inverse $\mathcal{Z}$-transform to equation (2.119), we can conclude that the optimal equalizer impulse response convolved with the channel impulse response producing as a result an impulse. This means that for zero additional error in the channel, the output signal $y(k)$ restores $x(k-L)$ and, therefore, one can conclude that a deconvolution process took place.

The delay in the reference signal plays an important role in the equalization process. Without the delay, the desired signal is $x(k)$, whereas the signal $y(k)$ will be mainly influenced by old samples of the input signal, since the unknown system is usually causal. As a consequence, the equalizer should also perform the task of predicting $x(k)$ simultaneously with the main task of equalizing the channel. The introduction of a delay alleviates the prediction task, leaving the equalizer free to invert the channel response. A rule of thumb for choosing the delay was proposed and analyzed in [23], where it was conjectured that the best delay should be close to half the time span of the equalizer. In practice the reader should try different delays.

In the case the unknown system is not of minimum phase, i.e., its transfer function has zeros outside the unit circle of the $\mathcal{Z}$ plane, the optimum equalizer is either stable and noncausal, or unstable and causal. Both solutions are unacceptable. The noncausal stable solution could be better approximated by a causal FIR filter when the delay is included in the desired signal. The delay forces a time shift in the ideal impulse response of the equalizer, allowing the time span, where most of the energy is concentrated, to be in the *causal* region.

If channel noise signal is present and is uncorrelated with the channel's input signal, the error signal and $y(k)$ will be accordingly noisier. However, it should be noticed that the adaptive equalizer, in the process of reducing the MSE, disturbs the optimal solution by trying to reduce the effects of $n(k)$. Therefore, in a noisy environment the equalizer transfer function is not exactly the inverse of $H(z)$.

In practice, the noblest use of the adaptive equalizer is to compensate for the distortion caused by the transmission channel in a communication system. The main distortions caused by the channels are high attenuation and intersymbol interference (ISI). The ISI is generated when different frequency components of the transmitted signals arrive at different times at the receiver, a phenomenon

caused by the nonlinear group delay of the channel [29]. For example, in a digital communication system, the time-dispersive channel extends a transmitted symbol beyond the time interval allotted to it, interfering in the past and future symbols. Under severe ISI, when short symbol space is used, the number of symbols causing ISI is large.

The channel impulse response is a time spread sequence described by $h(k)$ with the received signal being given by

$$re(k+J) = x(k)h(J) + \sum_{l=-\infty,\, l\neq k}^{k+J} x(l)h(k+J-l) + n(k+J) \qquad (2.121)$$

where J denotes the channel time delay (including the sampler phase). The first term of the equation above corresponds to the desired information, the second term is the interference of the symbols sent before and after $x(k)$. The third term accounts for channel noise. Obviously only the neighboring symbols have significant influence in the second term of the equation above. The elements of the second term involving $x(l)$, for $l > k$, are called pre-cursor ISI since they are caused by components of the data signal that reach the receiver before their cursor. On the other hand, the elements involving $x(l)$, for $l < k$, are called post-cursor ISI.

In many situations, the ISI is reduced by employing an equalizer consisting of an adaptive FIR filter of appropriate length. The adaptive equalizer attempts to cancel the ISI in the presence of noise. In digital communication, a decision device is placed after the equalizer in order to identify the symbol at a given instant. The equalizer coefficients are updated in two distinct circumstances by employing different reference signals. During the equalizer training period, a previously chosen training signal is transmitted through the channel and a properly delayed version of this signal, that is prestored in the receiver end, is used as reference signal. The training signal is usually a pseudo-noise sequence long enough to allow the equalizer to compensate for the channel distortions. After convergence, the error between the adaptive filter output and the decision device output is utilized to update the coefficients. The resulting scheme is the decision-directed adaptive equalizer. It should be mentioned that in some applications no training period is available. Usually, in this case, the decision-directed error is used all the time.

A more general equalizer scheme is the decision-feedback equalizer (DFE) illustrated in Fig. 2.12. The DFE is widely used in situations where the channel distortion is severe [29], [40]. The basic idea is to feed back, via a second FIR filter, the decisions made by the decision device that is applied to the equalized

signal. Assuming the decisions were correct, we are actually feeding back the symbols $x(l)$, for $l < k$, of equation (2.121). The DFE is able to cancel the post-cursor ISI for a number of past symbols (depending on the order of the FIR feedback filter), leaving more freedom for the feedforward section to take care of the remaining terms of the ISI. Some known characteristics of the DFE are [29]:

- The signals that are fed back are symbols, being noise free and allowing computational savings.
- The noise enhancement is reduced, if compared with the feedforward-only equalizer.
- Short time recovery when incorrect decisions are made.
- Reduced sensitivity to sampling phase.

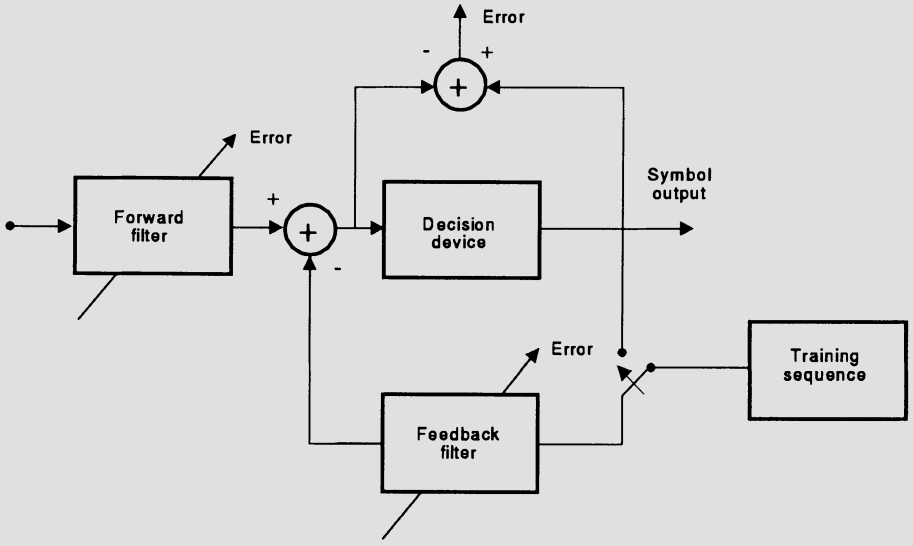


Figure 2.12 Decision-feedback equalizer.

2.9.5 Digital Communication System

For illustration, a general digital communication scheme over a channel consisting of a subscriber line (telephone line, for example) is shown in Fig. 2.13.

In either end the input signal is first coded and conditioned by a transmit filter. The filter shapes the pulse and limits in band the signal that is actually transmitted. The signal then crosses the hybrid to travel through a dual duplex channel. The hybrid is an impedance bridge used to transfer the transmit signal into the channel with minimal leakage to the near-end receiver. The imperfections of the hybrid cause echo that should be properly cancelled. In the channel, the signal is corrupted by white noise and crosstalk (leakage of signals being transmitted by other subscribers).

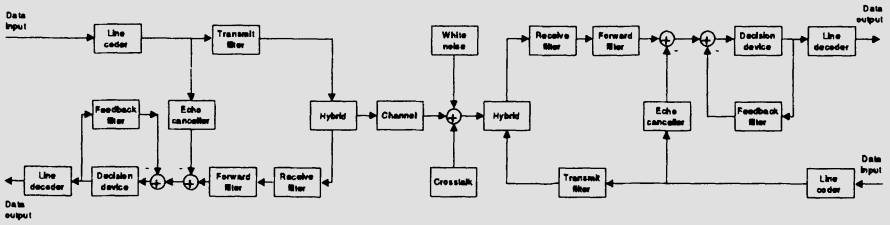


Figure 2.13 General digital communication transceiver.

After crossing the channel and the far-end hybrid, the signal is filtered by the receive filter that attenuates high-frequency noise and also acts as an antialiasing filter. Subsequently, we have a joint DFE and echo canceller, where the forward filter and echo canceller outputs are subtracted. The result after subtracting the decision feedback output is applied to the decision device. After passing through the decision device, the symbol is decoded.

Other schemes for data transmission in subscriber line exist [35]. The one shown here is for illustration purposes, having as special feature the joint equalizer and echo canceller strategy. The digital subscriber line (DSL) structure shown here has been used in integrated services digital network (ISDN) basic access, that allows a data rate of 144 Kbits/s [35]. Also, a similar scheme is employed in the high bit rate digital subscriber line (HDSL) [34], [41] that operates over short and conditioned loops [42], [43].

2.10 CONCLUDING REMARKS

In this chapter, we described some of the concepts underlying the adaptive filtering theory. The material presented here forms the basis to understand the behavior of most adaptive filtering algorithms in a practical implementation. The basic concept of the MSE surface searching algorithms was briefly reviewed,

serving as a starting point for the development of a number of practical adaptive filtering algorithms to be presented in the following chapters. The theory and practice of adaptive signal processing is also the main subject of some excellent books such as [22]-[28].

References

1. D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison Wesley, Reading, MA, 1973.

2. R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons, New York, NY, 2nd edition, 1990.

3. B. Widrow and M. E. Hoff, "Adaptive switching circuits," *WESCOM Conv. Rec.*, pt. 4, pp. 96-140, 1960.

4. B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson, Jr., "Stationary and nonstationary learning characteristics of the LMS adaptive filters," *Proceedings of the IEEE*, vol. 64, pp. 1151-1162, Aug. 1976.

5. A. Papoulis, *Signal Analysis*, McGraw Hill, New York, NY, 1977.

6. A. V. Oppenheim, A. S. Willsky, and I. T. Young, *Signals and Systems*, Prentice Hall, Englewood Cliffs, NJ, 1983.

7. A. V. Oppenheim and R. W. Schaffer, *Discrete-Time Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1989.

8. A. Antoniou, *Digital Filters: Analysis, Design, and Applications*, McGraw Hill, New York, NY, 2nd edition, 1992.

9. L. B. Jackson, *Digital Filters and Signal Processing*, Kluwer Academic Publishers, Norwell, MA, 3rd edition, 1996.

10. R. A. Roberts and C. T. Mullis, *Digital Signal Processing*, Addison-Wesley, Reading, MA, 1987.

11. J. G. Proakis and D. G. Manolakis, *Digital Signal Processing*, Macmillan Publishing Company, New York, NY, 2nd edition, 1992.

12. M. Bellanger, *Digital Processing of Signals*, John Wiley & Sons, New York, NY, 1984.

13. A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw Hill, New York, NY, 3rd edition, 1991.

14. P. Z. Peebles, Jr., *Probability, Random Variables, and Random Signal Principles*, McGraw Hill, New York, NY, 3rd edition, 1993.

15. W. A. Gardner, *Introduction to Random Processes*, McGraw Hill, New York, NY, Second edition, 1990.

16. C. R. Johnson, Jr., *Lectures on Adaptive Parameter Estimation*, Prentice Hall, Englewood Cliffs, NJ, 1988.

17. T. Söderström and P. Stoica, *System Identification*, Prentice Hall International, Hemel Hempstead, Hertfordshire, 1989.

18. G. Strang, *Linear Algebra and Its Applications*, Academic Press, New York, NY, 2nd Edition, 1980.

19. A. Papoulis, "Predictable processes and Wold's decomposition: A review," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-33, pp. 933-938, Aug. 1985.

20. S. M. Kay, *Modern Spectral Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1988.

21. S. L. Marple, Jr., *Digital Spectral Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1987.

22. M. L. Honig and D. G. Messerschmitt, *Adaptive Filters: Structures, Algorithms, and Applications*, Kluwer Academic Publishers, Boston, MA, 1984.

23. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1985.

24. S. T. Alexander, *Adaptive Signal Processing*, Springer Verlag, New York, NY, 1986.

25. J. R. Treichler, C. R. Johnson, Jr., and M. G. Larimore, *Theory and Design of Adaptive Filters*, John Wiley & Sons, New York, NY, 1987.

26. M. Bellanger, *Adaptive Digital Filters and Signal Analysis*, Marcel Dekker, Inc., New York, NY, 1987.

27. P. Strobach, *Linear Prediction Theory*, Springer Verlag, New York, NY, 1990.

28. S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, NJ, 2nd edition, 1991.

29. S. U. Qureshi, "Adaptive Equalization," *Proceedings of the IEEE*, vol. 73, pp. 1349-1387, Sept. 1985.

30. J. G. Proakis, *Digital Communication*, McGraw Hill, New York, NY, 2nd edition, 1989.

31. L. C. Wood and S. Treitel, "Seismic signal processing," *Proceedings of the IEEE*, vol. 63, pp. 649-661, Dec. 1975.

32. D. G. Messerschmitt, "Echo cancellation in speech and data transmission," *IEEE Journal on Selected Areas in Communications*, vol. SAC-2, pp. 283-296, March 1984.

33. M. L. Honig, "Echo cancellation of voiceband data signals using recursive least squares and stochastic gradient algorithms," *IEEE Trans. on Communications*, vol. COM-33, pp. 65-73, Jan. 1985.

34. S. Subramanian, D. J. Shpak, P. S. R. Diniz, and A. Antoniou, "The performance of adaptive filtering algorithms in a simulated HDSL environment," *Proc. IEEE Canadian Conf. Electrical and Computer Engineering*, Toronto, Canada, pp. TA 2.19.1-TA 2.19.5, Sept. 1992.

35. D. W. Lin, "Minimum mean-squared error echo cancellation and equalization for digital subscriber line transmission: Part I - theory and computation," *IEEE Trans. on Communications*, vol. 38, pp. 31-38, Jan. 1990.

36. D. W. Lin, "Minimum mean-squared error echo cancellation and equalization for digital subscriber line transmission: Part II - a simulation study," *IEEE Trans. on Communications*, vol. 38, pp. 39-45, Jan. 1990.

37. L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice Hall, Englewood Cliffs, NJ, 1978.

38. B. D. Van Veen and K. M. Buckley, "Beamforming: a versatile approach to spatial filtering," *IEEE Acoust., Speech, Signal Processing Magazine*, vol. 37, pp. 4-24, April 1988.

39. B. Widrow, J. R. Grover, Jr., J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearns, J. R. Zeidler, E. Dong, Jr., and R. C. Goodlin, "Adaptive noise cancelling: Principles and applications," *Proceedings of the IEEE*, vol. 63, pp. 1692-1716, Dec. 1975.

40. M. Abdulrahman and D. D. Falconer, "Cyclostationary crosstalk suppression by decision feedback equalization on digital subscriber line," *IEEE Journal on Selected Areas in Communications*, vol. 10, pp. 640-649, April 1992.

41. H. Samueli, B. Daneshrad, R. B. Joshi, B. C. Wong, and H. T. Nicholas, III, "A 64-tap CMOS echo canceller/decision feedback equalizer for 2B1Q HDSL transceiver," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 839-847, Aug. 1991.

42. J.-J. Werner, "The HDSL environment," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 785-800, Aug. 1991.

43. J. W. Leichleider, "High bit rate digital subscriber lines: A review of HDSL progress," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 769-784, Aug. 1991.

Problems

1. Supposing the input signal vector is composed by a delay line with a single input signal. Compute the correlation matrix for the following input signals:

   a)
   $$x(k) = \sin(\frac{\pi}{6}k) + \cos(\frac{\pi}{4}k) + n(k)$$

   b)
   $$x(k) = an_1(k)\cos(\omega_0 k) + n_2(k)$$

   c)
   $$x(k) = an_1(k)\sin(\omega_0 k + n_2(k))$$

   d)
   $$x(k) = -a_1 x(k-1) - a_2 x(k-2) + n(k)$$

   e)
   $$x(k) = \sum_{i=0}^{4} 0.25n(k-i)$$

   f)
   $$x(k) = an(k)e^{j\omega_0 k}$$

   In all cases, $n(k), n_1(k)$, and $n_2(k)$ are white noise with uniform distribution, with zero mean and with variances σ_n^2, $\sigma_{n_1}^2$, and $\sigma_{n_2}^2$, respectively. These random signals are considered independent.

2. For the correlation matrices given below, calculate their eigenvalues, eigenvectors, and the conditioning numbers.

a)

$$\mathbf{R} = \frac{1}{4}\begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

b)

$$\mathbf{R} = \begin{bmatrix} 1 & 0.95 & 0.9025 & 0.857375 \\ 0.95 & 1 & 0.95 & 0.9025 \\ 0.9025 & 0.95 & 1 & 0.95 \\ 0.857375 & 0.9025 & 0.95 & 1 \end{bmatrix}$$

c)

$$\mathbf{R} = 50\sigma_n^2 \begin{bmatrix} 1 & 0.9899 & 0.98 & 0.970 \\ 0.9899 & 1 & 0.9899 & 0.98 \\ 0.98 & 0.9899 & 1 & 0.9899 \\ 0.970 & 0.98 & 0.9899 & 1 \end{bmatrix}$$

d)

$$\mathbf{R} = \begin{bmatrix} 1 & 0.5 & 0.25 & 0.125 \\ 0.5 & 1 & 0.5 & 0.25 \\ 0.25 & 0.5 & 1 & 0.5 \\ 0.125 & 0.25 & 0.5 & 1 \end{bmatrix}$$

3. For the correlation matrix given below, calculate its eigenvalues, eigenvectors, and form the matrix $\mathbf{Q}$.

$$\mathbf{R} = \frac{1}{4}\begin{bmatrix} a_1 & a_2 \\ a_2 & a_1 \end{bmatrix}$$

4. Generate the ARMA processes $x(k)$ described below. Calculate the variance of the output signal and the autocorrelation for lags 1 and 2. In all cases, $n(k)$ is a white noise with variance 0.1.

a)

$$\begin{aligned} x(k) &= 1.9368x(k-1) - 0.9519x(k-2) + n(k) \\ &- 1.8894n(k-1) + n(k-2) \end{aligned}$$

b)

$$\begin{aligned} x(k) &= -1.9368x(k-1) - 0.9519x(k-2) \\ &+ n(k) + 1.8894n(k-1) + n(k-2) \end{aligned}$$

Hint: For white noise generation consult for example [13], [14].

5. Generate the AR processes $x(k)$ described below. Calculate the variance of the output signal and the autocorrelation for lags 1 and 2. In all cases, $n(k)$ is a white noise with variance 0.05.

a)

$$x(k) = -0.8987x(k-1) - 0.9018x(k-2) + n(k)$$

b)

$$x(k) = 0.057x(k-1) + 0.889x(k-2) + n(k)$$

6. Generate the MA processes $x(k)$ described below. Calculate the variance of the output signal and the autocovariance matrix. In all cases, $n(k)$ is a white noise with variance 1.

a)

$$\begin{aligned} x(k) &= 0.0935n(k) + 0.3027n(k-1) + 0.4n(k-2) \\ &+ 0.3027n(k-4) + 0.0935n(k-5) \end{aligned}$$

b)

$$x(k) = n(k) - n(k-1) + n(k-2) - n(k-4) + n(k-5)$$

c)

$$x(k) = n(k) + 2n(k-1) + 3n(k-2) + 2n(k-4) + n(k-5)$$

7. Show that a process generated by adding two AR processes is in general an ARMA process.

8. Determine if the following processes are mean ergodic:

a)

$$x(k) = an_1(k)\cos(\omega_0 k) + n_2(k)$$

b)

$$x(k) = an_1(k)\sin(\omega_0 k + n_2(k))$$

c)

$$x(k) = an(k)e^{2j\omega_0 k}$$

In all cases, $n(k), n_1(k)$, and $n_2(k)$ are white noise with uniform distribution, with zero mean and with variances σ_n^2, $\sigma_{n_1}^2$, and $\sigma_{n_2}^2$, respectively. These random signals are considered independent.

9. Suppose the matrix $\mathbf{R}$ and the vector $\mathbf{p}$ are known for a given experimental environment. Compute the Wiener solution for the following cases:

a)

$$\mathbf{R} = \frac{1}{4}\begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} \frac{1}{2} & \frac{3}{8} & \frac{2}{8} & \frac{1}{8} \end{bmatrix}^T$$

b)

$$\mathbf{R} = \begin{bmatrix} 1 & 0.8 & 0.64 & 0.512 \\ 0.8 & 1 & 0.8 & 0.64 \\ 0.64 & 0.8 & 1 & 0.8 \\ 0.512 & 0.64 & 0.8 & 1 \end{bmatrix}$$

$$\mathbf{p} = \frac{1}{4}[0.4096 \ 0.512 \ 0.64 \ 0.8]^T$$

c)

$$\mathbf{R} = \frac{1}{3}\begin{bmatrix} 3 & -2 & 1 \\ -2 & 3 & -2 \\ 1 & -2 & 3 \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} -2 & 1 & -\frac{1}{2} \end{bmatrix}^T$$

10. For the environments described in the previous problem, derive the updating formula for the steepest-descent method. Considering that the adaptive filter coefficients are initially zero, calculate their values for the ten first iterations.

11. Repeat the previous problem using the Newton method.

12. Calculate the spectral decomposition for the matrices $\mathbf{R}$ of problem 9.

13. Calculate the minimum MSE for the examples of problem 9 considering that the variance of the reference signal is given by σ_d^2.

14. Calculate the time constants of the MSE and of the coefficients for the examples of problem 9 considering that the steepest-descent algorithm was employed.

15. For the examples of problem 9, describe the equations for the MSE surface.

16. Using the spectral decomposition of a Hermitian matrix show that for N even

$$\mathbf{R}^{\frac{1}{N}} = \mathbf{Q}\mathbf{\Lambda}^{\frac{1}{N}}\mathbf{Q}^H = \sum_{i=0}^{N} \lambda_i^{\frac{1}{N}} \mathbf{q}_i \mathbf{q}_i^H$$

17. The gradient with respect to a complex parameter has not been defined. For our purposes the complex gradient vector can be defined as

$$\nabla_{\boldsymbol{\theta}}\{F(e(k))\} = \frac{\partial F(e(k))}{\partial re[\boldsymbol{\theta}(k)]} - j\frac{\partial F(e(k))}{\partial im[\boldsymbol{\theta}(k)]}$$

where $re[\cdot]$ and $im[\cdot]$ indicate real and imaginary parts of $[\cdot]$ respectively. Note that the partial derivatives are calculated for each element of $\boldsymbol{\theta}(k)$.

Derive the complex steepest-descent algorithm.

18. Derive the Newton algorithm for complex signals.

19. In a signal enhancement application, assume that $n_1(k) = n_2(k) * h(k)$, where $h(k)$ represents the impulse response of an unknown system. Also, assume that some small leakage of the signal $x(k)$, given by $h'(k) * x(k)$, is added to the adaptive filter input. Analyze the consequences of this phenomenon.

20. In the equalizer application, calculate the optimal equalizer transfer function when the channel noise is present.

3

THE LEAST-MEAN-SQUARE (LMS) ALGORITHM

3.1 INTRODUCTION

The least mean-square (LMS) is a search algorithm in which a simplification of the gradient vector computation is made possible by appropriately modifying the objective function [1]-[2]. The LMS algorithm, as well as others related to it, is widely used in various applications of adaptive filtering due to its computational simplicity [3]-[7]. The convergence characteristics of the LMS algorithm are examined in order to establish a range for the convergence factor that will guarantee stability. The convergence speed of the LMS is shown to be dependent of the eigenvalue spread of the input-signal correlation matrix [2]-[6]. In this chapter, several properties of the LMS algorithm are discussed including the misadjustment in stationary and nonstationary environments [2]-[9], tracking performance, and finite wordlength effects [10]-[12].

The LMS algorithm is by far the most widely used algorithm in adaptive filtering for several reasons. The main features that attracted the use of the LMS algorithm are low computational complexity, proof of convergence in stationary environment, unbiased convergence in the mean to the Wiener solution, and stable behavior when implemented with finite-precision arithmetic.

3.2 THE LMS ALGORITHM

In the previous chapter we derived the optimal solution for the parameters of the adaptive filter implemented through a linear combiner, which corresponds to the case of multiple input signals. This solution leads to the minimum mean-square

P. S. R. Diniz, *Adaptive Filtering*

error in estimating the reference signal $d(k)$. The optimal (Wiener) solution is given by

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p} \tag{3.1}$$

where $\mathbf{R} = E[\mathbf{x}(k)\mathbf{x}^T(k)]$ and $\mathbf{p} = E[d(k)\mathbf{x}(k)]$, assuming that $d(k)$ and $\mathbf{x}(k)$ are jointly wide-sense stationary.

If good estimates of matrix $\mathbf{R}$, denoted by $\hat{\mathbf{R}}(k)$, and of vector $\mathbf{p}$, denoted by $\hat{\mathbf{p}}(k)$, are available, a steepest-descent-based algorithm can be used to search the Wiener solution of equation (3.1) as follows:

$$\begin{aligned} \mathbf{w}(k+1) &= \mathbf{w}(k) - \mu\hat{\mathbf{g}}_{\mathbf{w}}(k) \\ &= \mathbf{w}(k) + 2\mu(\hat{\mathbf{p}}(k) - \hat{\mathbf{R}}(k)\mathbf{w}(k)) \end{aligned} \tag{3.2}$$

for $k = 0, 1, 2, \ldots$, where $\hat{\mathbf{g}}_{\mathbf{w}}(k)$ represents an estimate of the gradient vector of the objective function with respect to the filter coefficients.

One possible solution is to estimate the gradient vector by employing instantaneous estimates for $\mathbf{R}$ and $\mathbf{p}$ as follows:

$$\begin{aligned} \hat{\mathbf{R}}(k) &= \mathbf{x}(k)\mathbf{x}^T(k) \\ \hat{\mathbf{p}}(k) &= d(k)\mathbf{x}(k) \end{aligned} \tag{3.3}$$

The resulting gradient estimate is given by

$$\begin{aligned} \hat{\mathbf{g}}_{\mathbf{w}}(k) &= -2d(k)\mathbf{x}(k) + 2\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}(k) \\ &= 2\mathbf{x}(k)(-d(k) + \mathbf{x}^T(k)\mathbf{w}(k)) \\ &= -2e(k)\mathbf{x}(k) \end{aligned} \tag{3.4}$$

Note that if the objective function is replaced by the instantaneous square error $e^2(k)$, instead of the MSE, the gradient estimate above represents the true gradient vector since

$$\begin{aligned} \frac{\partial e^2(k)}{\partial \mathbf{w}} &= \left[2e(k)\frac{\partial e(k)}{\partial w_0(k)} \; 2e(k)\frac{\partial e(k)}{\partial w_1(k)} \; \cdots \; 2e(k)\frac{\partial e(k)}{\partial w_N(k)}\right]^T \\ &= -2e(k)\mathbf{x}(k) \\ &= \hat{\mathbf{g}}_{\mathbf{w}}(k) \end{aligned} \tag{3.5}$$

The resulting gradient-based algorithm is known, because it minimizes the mean of the squared error, as the least-mean-square (LMS) algorithm, whose updating equation is

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu e(k)\mathbf{x}(k) \tag{3.6}$$

Algorithm 3.1

LMS Algorithm

Initialization

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^T$

Do for $k \geq 0$

$e(k) = d(k) - \mathbf{x}^T(k)\mathbf{w}(k)$

$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu e(k)\mathbf{x}(k)$

□

where the convergence factor μ should be chosen in a range to guarantee convergence.

Fig. 3.1 depicts the realization of the LMS algorithm for a delay line input $\mathbf{x}(k)$. Typically, one iteration of the LMS requires $N + 2$ multiplications for the filter coefficient updating and $N + 1$ multiplications for the error generation. The detailed description of the LMS algorithm is shown in the table denoted Algorithm 3.1.

It should be noted that the initialization is not necessarily performed as described in Algorithm 3.1, where the coefficients of the adaptive filter were initialized with zeros. For example, if a rough idea of the optimal coefficient value is known, these values could be used to form $\mathbf{w}(0)$ leading to a reduction in the number of iterations required to reach the neighborhood of $\mathbf{w}_o$.

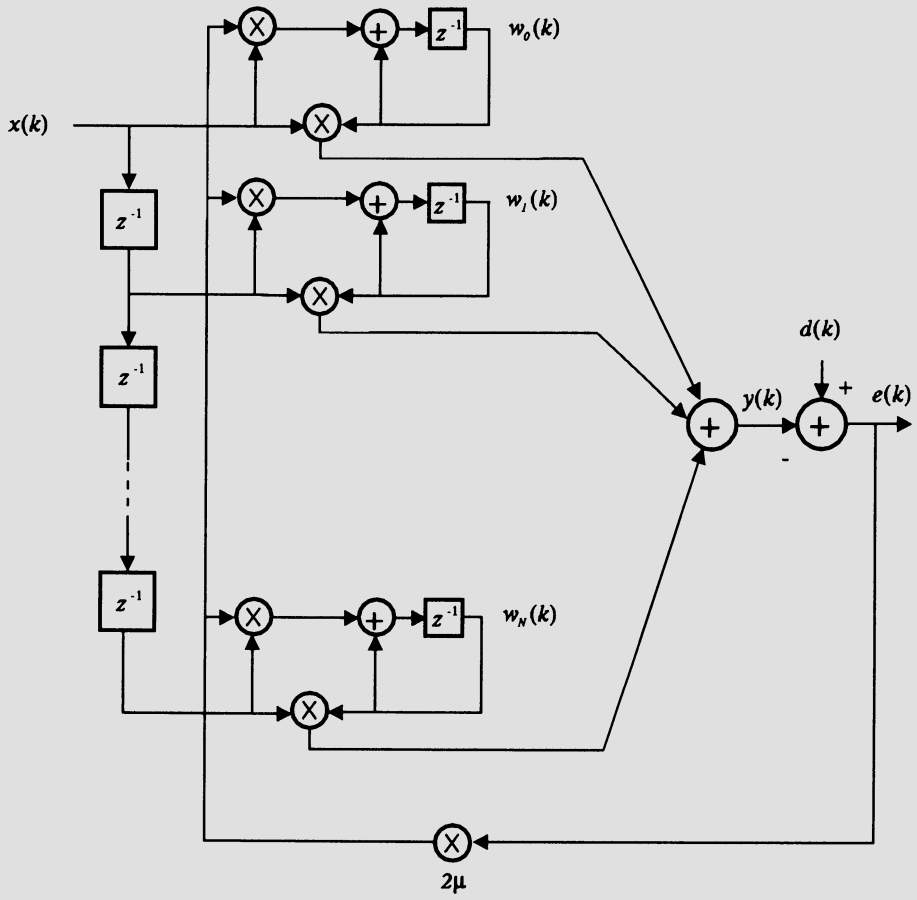


Figure 3.1 LMS adaptive FIR filter.

3.3 SOME PROPERTIES OF THE LMS ALGORITHM

In this section, the main properties related to the convergence behavior of the LMS algorithm in a stationary environment are described. The information contained here is essential to understand the influence of the convergence factor μ in various convergence aspects of the LMS algorithm.

3.3.1 Gradient Behavior

As shown in the previous chapter, see equation (2.79), the ideal gradient direction to perform a search on the MSE surface for the optimum coefficient vector solution is

$$\begin{aligned} \mathbf{g}_{\mathbf{w}}(k) &= 2\left(E\left[\mathbf{x}(k)\mathbf{x}^T(k)\right]\mathbf{w}(k) - E\left[d(k)\mathbf{x}(k)\right]\right) \\ &= 2(\mathbf{R}\mathbf{w}(k) - \mathbf{p}) \end{aligned} \tag{3.7}$$

In the LMS algorithm instantaneous estimates of $\mathbf{R}$ and $\mathbf{p}$ are used to determine the search direction, i.e.,

$$\hat{\mathbf{g}}_{\mathbf{w}}(k) = 2\left[\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}(k) - d(k)\mathbf{x}(k)\right] \tag{3.8}$$

As can be expected, the direction determined by equation (3.8) is quite different from that of equation (3.7). Therefore, by using the more computationally attractive gradient direction of the LMS algorithm, the convergence behavior is not the same as that of the steepest-descent algorithm.

In average, it can be said that the LMS gradient direction has the tendency to approach the ideal gradient direction since for a fixed coefficient vector $\mathbf{w}$

$$\begin{aligned} E[\hat{\mathbf{g}}_{\mathbf{w}}(k)] &= 2\left(E\left[\mathbf{x}(k)\mathbf{x}^T(k)\right]\mathbf{w} - E\left[d(k)\mathbf{x}(k)\right]\right) \\ &= \mathbf{g}_{\mathbf{w}} \end{aligned} \tag{3.9}$$

hence, vector $\hat{\mathbf{g}}_{\mathbf{w}}(k)$ can be interpreted as an unbiased instantaneous estimate of $\mathbf{g}_{\mathbf{w}}$. If for a fixed $\mathbf{w}$ vector $\hat{\mathbf{g}}_{\mathbf{w}}(k)$ is calculated for a large number of inputs and reference signals, the average direction tends to $\mathbf{g}_{\mathbf{w}}$, i.e.,

$$\lim_{M\to\infty} \frac{1}{M} \sum_{i=1}^{M} \hat{\mathbf{g}}_{\mathbf{w}}(k+i) \to \mathbf{g}_{\mathbf{w}} \tag{3.10}$$

3.3.2 Convergence Behavior of the Coefficient Vector

Assume that an unknown FIR filter with coefficient vector given by $\mathbf{w}_o$ is being identified by an adaptive FIR filter of the same order, employing the LMS algorithm. Measurement white noise $n(k)$ with zero mean and variance σ_n^2 is added to the output of the unknown system.

The error in the adaptive filter coefficients as related to the ideal coefficient vector $\mathbf{w}_o$, in each iteration, is described by the $N+1$ vector

$$\Delta\mathbf{w}(k) = \mathbf{w}(k) - \mathbf{w}_o \tag{3.11}$$

With this definition, the LMS algorithm can alternatively be described by

$$\begin{aligned}
\Delta\mathbf{w}(k+1) &= \Delta\mathbf{w}(k) + 2\mu e(k)\mathbf{x}(k) \\
&= \Delta\mathbf{w}(k) + 2\mu\mathbf{x}(k)\left(\mathbf{x}^T(k)\mathbf{w}_o + n(k) - \mathbf{x}^T(k)\mathbf{w}(k)\right) \\
&= \Delta\mathbf{w}(k) + 2\mu\mathbf{x}(k)\left(e_o(k) - \mathbf{x}^T(k)\Delta\mathbf{w}(k)\right) \\
&= \left(\mathbf{I} - 2\mu\mathbf{x}(k)\mathbf{x}^T(k)\right)\Delta\mathbf{w}(k) + 2\mu e_o(k)\mathbf{x}(k)
\end{aligned} \tag{3.12}$$

where $e_o(k)$ is the minimum output error given by

$$\begin{aligned}
e_o(k) &= d(k) - \mathbf{w}_o^T\mathbf{x}(k) \\
&= \mathbf{w}_o^T\mathbf{x}(k) + n(k) - \mathbf{w}_o^T\mathbf{x}(k) \\
&= n(k)
\end{aligned} \tag{3.13}$$

The expected error in the coefficient vector is then given by

$$E[\Delta\mathbf{w}(k+1)] = E[(\mathbf{I} - 2\mu\mathbf{x}(k)\mathbf{x}^T(k))\Delta\mathbf{w}(k)] + 2\mu E[e_o(k)\mathbf{x}(k)] \tag{3.14}$$

If it is assumed that the elements of $\mathbf{x}(k)$ are statistically independent of the elements of $\Delta\mathbf{w}(k)$ and $e_o(k)$, equation (3.14) can be simplified as follows:

$$\begin{aligned}
E[\Delta\mathbf{w}(k+1)] &= \left(\mathbf{I} - 2\mu E[\mathbf{x}(k)\mathbf{x}^T(k)]\right) E[\Delta\mathbf{w}(k)] \\
&= (\mathbf{I} - 2\mu\mathbf{R})E[\Delta\mathbf{w}(k)]
\end{aligned} \tag{3.15}$$

The first assumption is justified if we assume that the deviation in the parameters is dependent of previous input-signal vectors only, whereas the second assumption means that the error signal at the optimal solution is orthogonal to the elements of the input-signal vector. The expression above leads to

$$E[\Delta\mathbf{w}(k+1)] = (\mathbf{I} - 2\mu\mathbf{R})^{k+1}E[\Delta\mathbf{w}(0)] \tag{3.16}$$

Equation (3.15) premultiplied by $\mathbf{Q}^T$, where $\mathbf{Q}$ is the unitary matrix that diagonalizes $\mathbf{R}$ through a similarity transformation, yields

$$\begin{aligned}
E\left[\mathbf{Q}^T\Delta\mathbf{w}(k+1)\right] &= (\mathbf{I} - 2\mu\mathbf{Q}^T\mathbf{R}\mathbf{Q})E\left[\mathbf{Q}^T\mathbf{w}(k)\right] \\
&= E\left[\Delta\mathbf{w}'(k+1)\right]
\end{aligned}$$

$$
\begin{aligned}
&= (\mathbf{I} - 2\mu\mathbf{\Lambda})E\left[\Delta\mathbf{w}'(k)\right] \\
&= \begin{bmatrix} 1-2\mu\lambda_0 & 0 & \cdots & 0 \\ 0 & 1-2\mu\lambda_1 & & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & 1-2\mu\lambda_N \end{bmatrix} E\left[\Delta\mathbf{w}'(k)\right]
\end{aligned}
\tag{3.17}
$$

where $\Delta\mathbf{w}'(k+1) = \mathbf{Q}^T\Delta\mathbf{w}(k+1)$ is the rotated-coefficient error vector. The applied rotation yielded an equation where the driving matrix is diagonal, making easier to analyze the equation dynamic behavior. Alternatively, the above relation can be expressed as

$$
\begin{aligned}
E\left[\Delta\mathbf{w}'(k+1)\right] &= (\mathbf{I} - 2\mu\mathbf{\Lambda})^{k+1}E\left[\Delta\mathbf{w}'(0)\right] \\
&= \begin{bmatrix} (1-2\mu\lambda_0)^{k+1} & 0 & \cdots & 0 \\ 0 & (1-2\mu\lambda_1)^{k+1} & & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & (1-2\mu\lambda_N)^{k+1} \end{bmatrix} \\
&\quad \times\ E\left[\Delta\mathbf{w}'(0)\right]
\end{aligned}
\tag{3.18}
$$

This equation shows that in order to guarantee convergence of the coefficients in the mean, the convergence factor of the LMS algorithm must be chosen in the range

$$
0 < \mu < \frac{1}{\lambda_{max}} \tag{3.19}
$$

where λ_{max} is the largest eigenvalue of $\mathbf{R}$. Values of μ in the range guarantees that all elements of the diagonal matrix in the equation (3.18) tend to zero as $k \to \infty$. As a result $E[\Delta\mathbf{w}'(k+1)]$ tends to zero for large k.

The choice of μ as explained above ensures that the mean value of the coefficient vector approaches the optimum coefficient vector $\mathbf{w}_o$. It should be mentioned that if the matrix $\mathbf{R}$ has a large eigenvalue spread, it is advisable to choose a value for μ much smaller than the upper bound. As a result, the convergence speed of the coefficients will be primarily dependent on the value of the smallest eigenvalue, responsible for the slowest mode in equation (3.18).

The key assumption for the analysis above is the so-called independence theory [4], which considers all vectors $\mathbf{x}(i)$, for $i = 0, 1, \ldots, k$, statistically independent. This assumption allowed us to consider $\Delta\mathbf{w}(k)$ independent of $\mathbf{x}(k)\mathbf{x}^T(k)$ in equation (3.14). Such an assumption, despite not being rigorously valid es-

pecially when $\mathbf{x}(k)$ represents the elements of a delay line, leads to theoretical results that are in good agreement with the experimental results.

3.3.3 Coefficient-Error-Vector Covariance Matrix

In this subsection, we derive the expressions for the second-order statistics of the errors in the adaptive filter coefficients. Since the mean value of $\Delta\mathbf{w}(k)$ is zero, the covariance of the coefficient-error vector is defined as

$$\text{cov}[\Delta\mathbf{w}(k)] = E[\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k)] = E\left[(\mathbf{w}(k) - \mathbf{w}_o)(\mathbf{w}(k) - \mathbf{w}_o)^T\right] \quad (3.20)$$

By replacing the equation (3.12) in (3.20) it follows that

$$\begin{aligned}\text{cov}[\Delta\mathbf{w}(k+1)] &= E\left[(\mathbf{I} - 2\mu\mathbf{x}(k)\mathbf{x}^T(k))\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k)\ (\mathbf{I} - 2\mu\mathbf{x}(k)\mathbf{x}^T(k))^T\right. \\ &\quad +(\mathbf{I} - 2\mu\mathbf{x}(k)\mathbf{x}^T(k))2\mu e_o(k)\mathbf{x}^T(k) \\ &\quad +2\mu e_o(k)\mathbf{x}(k)\Delta\mathbf{w}^T(k)(\mathbf{I} - 2\mu\mathbf{x}(k)\mathbf{x}^T(k))^T \\ &\quad \left.+4\mu^2 e_o^2(k)\mathbf{x}(k)\mathbf{x}^T(k)\right] \end{aligned} \quad (3.21)$$

By considering $e_o(k)$ independent of $\Delta\mathbf{w}(k)$ and orthogonal to $\mathbf{x}(k)$, the second and third terms of the right-hand side of the above equation can be eliminated. The details of this simplification can be carried out by describing each element of the eliminated matrices explicitly. In this case,

$$\begin{aligned}\text{cov}[\Delta\mathbf{w}(k+1)] &= \text{cov}[\Delta\mathbf{w}(k)] + E[-2\mu\mathbf{x}(k)\mathbf{x}^T(k)\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k) \\ &\quad -2\mu\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k)\mathbf{x}(k)\mathbf{x}^T(k) \\ &\quad +4\mu^2\mathbf{x}(k)\mathbf{x}^T(k)\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k)\mathbf{x}(k)\mathbf{x}^T(k) \\ &\quad +4\mu^2 e_o^2(k)\mathbf{x}(k)\mathbf{x}^T(k)] \end{aligned} \quad (3.22)$$

In addition, assuming that $\Delta\mathbf{w}(k)$ and $\mathbf{x}(k)$ are independent, the equation (3.22) can be rewritten as

$$\begin{aligned}\text{cov}[\Delta\mathbf{w}(k+1)] &= \text{cov}[\Delta\mathbf{w}(k)] - 2\mu E[\mathbf{x}(k)\mathbf{x}^T(k)]E[\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k)] \\ &\quad -2\mu E[\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k)]E[\mathbf{x}(k)\mathbf{x}^T(k)] \\ &\quad +4\mu^2 E[\mathbf{x}(k)\mathbf{x}^T(k)\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)] \\ &\quad +4\mu^2 E[e_o^2(k)]E[\mathbf{x}(k)\mathbf{x}^T(k)] \\ &= \text{cov}[\Delta\mathbf{w}(k)] - 2\mu\mathbf{R}\ \text{cov}[\Delta\mathbf{w}(k)] \\ &\quad -2\mu\ \text{cov}[\Delta\mathbf{w}(k)]\mathbf{R} + 4\mu^2\mathbf{A} + 4\mu^2\sigma_n^2\mathbf{R} \end{aligned} \quad (3.23)$$

The calculation of $\mathbf{A} = E[\mathbf{x}(k)\mathbf{x}^T(k)\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)]$ involves fourth-order moments and the result can be obtained by expanding the matrix inside the operation $E[\cdot]$ as described in [4] and [13] for jointly Gaussian input-signal samples. The details are not shown here since similar derivation is shown in equation (3.92). The result is

$$\mathbf{A} = 2\mathbf{R}\ \mathrm{cov}[\Delta\mathbf{w}(k)]\ \mathbf{R} + \mathbf{R}\ \mathrm{tr}[\mathbf{R}\ \mathrm{cov}[\Delta\mathbf{w}(k)]] \tag{3.24}$$

where tr[·] denotes trace of [·]. Equation (3.23) is needed to calculate the excess of mean-square error caused by the noisy estimate of the gradient employed by the LMS algorithm. As can be noted, $\mathrm{cov}[\Delta\mathbf{w}(k+1)]$ does not tend to $\mathbf{0}$ as $k \to \infty$, due to the last term in equation (3.23) that provides an excitation in the dynamic matrix equation.

A more useful form for the equation (3.23) can be obtained by premultiplying and postmultiplying it by $\mathbf{Q}^T$ and $\mathbf{Q}$ respectively, yielding

$$\begin{aligned}
\mathbf{Q}^T\mathrm{cov}[\Delta\mathbf{w}(k+1)]\mathbf{Q} &= \mathbf{Q}^T\ \mathrm{cov}[\Delta\mathbf{w}(k)]\ \mathbf{Q} \\
&\quad -2\mu\mathbf{Q}^T\mathbf{R}\mathbf{Q}\mathbf{Q}^T\ \mathrm{cov}[\Delta\mathbf{w}(k)]\mathbf{Q} \\
&\quad -2\mu\mathbf{Q}^T\ \mathrm{cov}[\Delta\mathbf{w}(k)]\mathbf{Q}\mathbf{Q}^T\mathbf{R}\mathbf{Q} \\
&\quad +8\mu^2\mathbf{Q}^T\mathbf{R}\mathbf{Q}\mathbf{Q}^T\ \mathrm{cov}[\Delta\mathbf{w}(k)]\mathbf{Q}\mathbf{Q}^T\mathbf{R}\mathbf{Q} \\
&\quad +4\mu^2\mathbf{Q}^T\mathbf{R}\mathbf{Q}\mathbf{Q}^T\ \mathrm{tr}[\mathbf{R}\mathbf{Q}\mathbf{Q}^T\ \mathrm{cov}[\Delta\mathbf{w}(k)]]\mathbf{Q} \\
&\quad +4\mu^2\sigma_n^2\mathbf{Q}^T\mathbf{R}\mathbf{Q}
\end{aligned} \tag{3.25}$$

where we used the equality $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}$. Using the fact that $\mathbf{Q}^T\mathrm{tr}[\mathbf{B}]\mathbf{Q} = \mathrm{tr}[\mathbf{Q}^T\mathbf{B}\mathbf{Q}]$ for any $\mathbf{B}$,

$$\begin{aligned}
\mathrm{cov}[\Delta\mathbf{w}'(k+1)] &= \mathrm{cov}[\Delta\mathbf{w}'(k)] \\
&\quad -2\mu\mathbf{\Lambda}\ \mathrm{cov}[\Delta\mathbf{w}'(k)] - 2\mu\ \mathrm{cov}[\Delta\mathbf{w}'(k)]\mathbf{\Lambda} \\
&\quad +8\mu^2\mathbf{\Lambda}\ \mathrm{cov}[\Delta\mathbf{w}'(k)]\mathbf{\Lambda} \\
&\quad +4\mu^2\mathbf{\Lambda}\ \mathrm{tr}[\mathbf{\Lambda}\ \mathrm{cov}[\Delta\mathbf{w}'(k)]] + 4\mu^2\sigma_n^2\mathbf{\Lambda}
\end{aligned} \tag{3.26}$$

where $\mathrm{cov}[\Delta\mathbf{w}'(k)] = E[\mathbf{Q}^T\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k)\mathbf{Q}]$.

As will be shown in section 3.3.5, only the diagonal elements of $\mathrm{cov}[\Delta\mathbf{w}'(k)]$ contribute to the excess of MSE in the LMS algorithm. By defining $\mathbf{v}'(k)$ as a vector with elements consisting of the diagonal elements of $\mathrm{cov}[\Delta\mathbf{w}'(k)]$, and $\boldsymbol{\lambda}$ as a vector consisting of the eigenvalues of $\mathbf{R}$, the following relation can be derived from the equations above

$$\begin{aligned}
\mathbf{v}'(k+1) &= (\mathbf{I} - 4\mu\mathbf{\Lambda} + 8\mu^2\mathbf{\Lambda}^2 + 4\mu^2\boldsymbol{\lambda}\boldsymbol{\lambda}^T)\mathbf{v}'(k) + 4\mu^2\sigma_n^2\boldsymbol{\lambda} \\
&= \mathbf{B}\mathbf{v}'(k) + 4\mu^2\sigma_n^2\boldsymbol{\lambda}
\end{aligned} \tag{3.27}$$

where the elements of $\mathbf{B}$ are given by

$$b_{ij} = \begin{cases} 1 - 4\mu\lambda_i + 8\mu^2\lambda_i^2 + 4\mu^2\lambda_i^2 & \text{for } i = j \\ 4\mu^2\lambda_i\lambda_j & \text{for } i \neq j \end{cases} \tag{3.28}$$

The value of the convergence factor μ must be chosen in a range that guarantees the convergence of $\mathbf{v}'(k)$. A sufficient condition to guarantee convergence is to force the sum of the elements in any row of $\mathbf{B}$ to be kept in the range $-1 < \sum_{j=0}^{N} b_{ij} < 1$. Since

$$\sum_{j=0}^{N} b_{ij} = 1 - 4\mu\lambda_i + 8\mu^2\lambda_i^2 + 4\mu^2\lambda_i \sum_{j=0}^{N} \lambda_j \tag{3.29}$$

the critical values of μ are those which the equation above approaches 1, as for any μ the expression is always positive. This will occur only if the last three terms of equation (3.29) approach zero, that is

$$-4\mu\lambda_i + 8\mu^2\lambda_i^2 + 4\mu^2\lambda_i \sum_{j=0}^{N} \lambda_j \approx 0$$

After simple manipulation the stability condition obtained is

$$0 < \mu < \frac{1}{2\lambda_i + \sum_{j=0}^{N} \lambda_j} < \frac{1}{\sum_{j=0}^{N} \lambda_j} = \frac{1}{\mathrm{tr}[\mathbf{R}]} \tag{3.30}$$

where the last and simpler expression is more widely used in practice.

The obtained upper bound for the value of μ is important from the practical point of view, because it gives us an indication of the maximum value of μ that could be used in order to achieve convergence of the coefficients. However, the reader should be advised that the given upper bound is somewhat optimistic due to the approximations and assumptions made. In most cases, the value of μ should not be chosen close to the upper bound.

3.3.4 Behavior of the Error Signal

In this subsection, the mean value of the output error in the adaptive filter is calculated, considering that the unknown system model has infinite impulse

response and there is measurement noise. The error signal, when an additional zero-mean measurement noise is accounted for, is given by

$$e(k) = d'(k) - \mathbf{w}^T(k)\mathbf{x}(k) + n(k) \tag{3.31}$$

where $d'(k)$ is the desired signal without measurement noise. For a given known input vector $\mathbf{x}(k)$, the expected value of the error signal is

$$\begin{aligned} E[e(k)] &= E[d'(k)] - E[\mathbf{w}^T(k)\mathbf{x}(k)] + E[n(k)] \\ &= E[d'(k)] - \mathbf{w}_o^T\mathbf{x}(k) + E[n(k)] \end{aligned} \tag{3.32}$$

where $\mathbf{w}_o$ is the optimal solution, i.e., the Wiener solution for the coefficient vector. Note that the input-signal vector was assumed known in the above equation, in order to expose what can be expected if the adaptive filter converges to the optimal solution. If $d'(k)$ was generated through an infinite impulse response system, a residue error remains in the subtraction of the first two terms due to undermodeling, i.e.,

$$E[e(k)] = E\left[\sum_{i=N+1}^{\infty} \tilde{w}_{oi}x(k-i)\right] + E[n(k)] \tag{3.33}$$

where $\tilde{w}_{0i}$ are the coefficients of the process that generated the part of $d'(k)$ not identified by the adaptive filter. If the input signal and $n(k)$ have zero mean, then $E[e(k)] = 0$.

Now, the minimum MSE is calculated for undermodeling situations (adaptive FIR filter with insufficient number of parameters) and in the presence of additional noise. The minimum mean-square error that can be obtained, assuming the input signal is a white noise uncorrelated with the additional noise signal is given by

$$\begin{aligned} \xi_{min} &= E[e^2(k)]_{min} = \sum_{i=N+1}^{\infty} \tilde{w}_{oi}^2 E[x^2(k-i)] + E[n^2(k)] \\ &= \sum_{i=N+1}^{\infty} \tilde{w}_{oi}^2\sigma_x^2 + \sigma_n^2 \end{aligned} \tag{3.34}$$

This minimum error is achieved when it is assumed that the adaptive filter multiplier coefficients are frozen at their optimum values. In case the adaptive filter has sufficient order to model the process that generated $d(k)$, the minimum MSE that can be achieved is equal to the variance of the additional noise.

3.3.5 Excess of Mean-Square Error and Misadjustment

The result of the previous subsection assumes that the adaptive filter coefficients converge to their optimal values, but in practice this is not so. Although the coefficient vector in average converges to $\mathbf{w}_o$, the instantaneous deviation $\Delta\mathbf{w}(k) = \mathbf{w}(k) - \mathbf{w}_o$ generates an excess of MSE. The excess of MSE can be quantified as described in the present section. The output error at instant k is given by

$$\begin{aligned} e(k) &= d(k) - \mathbf{w}_o^T(k)\mathbf{x}(k) - \Delta\mathbf{w}^T(k)\mathbf{x}(k) \\ &= e_o(k) - \Delta\mathbf{w}^T(k)\mathbf{x}(k) \end{aligned} \tag{3.35}$$

then

$$e^2(k) = e_o^2(k) - 2e_o(k)\Delta\mathbf{w}^T(k)\mathbf{x}(k) + \Delta\mathbf{w}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\Delta\mathbf{w}(k) \tag{3.36}$$

The so-called independence theory assumes that the vectors $\mathbf{x}(k)$, for all k, are statistically independent, allowing a simple mathematical treatment for the LMS algorithm. As mentioned before, this assumption is in general not true, especially in the case where $\mathbf{x}(k)$ consists of the elements of a delay line. However, even in this case the use of the independence assumption is justified by the agreement between the analytical and the experimental results. With the independence assumption, $\Delta\mathbf{w}(k)$ can be considered independent of $\mathbf{x}(k)$, since only previous input vectors are involved in determining $\Delta\mathbf{w}(k)$. By using the assumption and applying the expected value operator to the equation (3.36), we have

$$\begin{aligned} \xi(k) &= E[e^2(k)] \\ &= \xi_{min} - 2E[\Delta\mathbf{w}^T(k)]E[e_o(k)\mathbf{x}(k)] \\ &\quad + E[\Delta\mathbf{w}^T(k)E[\mathbf{x}(k)\mathbf{x}^T(k)]\Delta\mathbf{w}(k)] \end{aligned} \tag{3.37}$$

Since $\mathbf{R} = E[\mathbf{x}(k)\mathbf{x}^T(k)]$ and by the orthogonality principle $E[e_o(k)\mathbf{x}(k)] = 0$, the equation above can be simplified as follows:

$$\xi(k) = \xi_{min} + E[\Delta\mathbf{w}^T(k)\mathbf{R}\Delta\mathbf{w}(k)] \tag{3.38}$$

The excess in the MSE is given by

$$\begin{aligned} \Delta\xi(k) &\triangleq \xi(k) - \xi_{min} = E[\Delta\mathbf{w}^T(k)\mathbf{R}\Delta\mathbf{w}(k)] \\ &= E[\mathrm{tr}(\mathbf{R}\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k))] \\ &= \mathrm{tr}[E[\mathbf{R}\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k)]] \end{aligned} \tag{3.39}$$

where in the second equality we used the property $\text{tr}[\mathbf{A} \cdot \mathbf{B}] = \text{tr}[\mathbf{B} \cdot \mathbf{A}]$. By using the fact that $\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$, the following relation results

$$\begin{aligned} \Delta\xi(k) &= \text{tr}[E[\mathbf{Q}\mathbf{Q}^T\mathbf{R}\mathbf{Q}\mathbf{Q}^T\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k)\mathbf{Q}\mathbf{Q}^T]] \\ &= \text{tr}[\mathbf{Q}\mathbf{\Lambda}\ \text{cov}(\Delta\mathbf{w}'(k))\mathbf{Q}^T] \end{aligned} \tag{3.40}$$

Therefore,

$$\Delta\xi(k) = \text{tr}[\mathbf{\Lambda}\ \text{cov}(\Delta\mathbf{w}'(k))] \tag{3.41}$$

From equation (3.27), it is easy to show that

$$\Delta\xi(k) = \sum_{i=0}^{N} \lambda_i v_i'(k) = \mathbf{\lambda}^T\mathbf{v}'(k) \tag{3.42}$$

Since

$$\begin{aligned} v_i'(k+1) &= (1 - 4\mu\lambda_i + 8\mu^2\lambda_i^2)v_i'(k) \\ &\quad +4\mu^2\lambda_i \sum_{j=0}^{N} \lambda_j v_j'(k) + 4\mu^2\sigma_n^2\lambda_i \end{aligned} \tag{3.43}$$

and $v_i'(k+1) \approx v_i'(k)$ for large k, we can apply a summation operation to the above equation in order to obtain

$$\begin{aligned} \sum_{j=0}^{N} \lambda_j v_j'(k) &= \frac{\mu\sigma_n^2 \sum_{j=0}^{N} \lambda_j + 2\mu \sum_{j=0}^{N} \lambda_j^2 v_j'(k)}{1 - \mu \sum_{j=0}^{N} \lambda_j} \\ &\approx \frac{\mu\sigma_n^2 \sum_{j=0}^{N} \lambda_j}{1 - \mu \sum_{j=0}^{N} \lambda_j} \\ &= \frac{\mu\sigma_n^2 \text{tr}[\mathbf{R}]}{1 - \mu\text{tr}[\mathbf{R}]} \end{aligned} \tag{3.44}$$

where the term $2\mu \sum_{j=0}^{N} \lambda_j^2 v_j'(k)$ was considered very small as compared to the remaining terms of the numerator. This assumption is not easily justifiable, but is valid for small values of μ.

The excess of mean-square error can then be expressed as

$$\xi_{exc} = \lim_{k\to\infty} \Delta\xi(k) \approx \frac{\mu\sigma_n^2 \text{tr}[\mathbf{R}]}{1 - \mu\text{tr}[\mathbf{R}]} \tag{3.45}$$

This equation, for very small μ, can be approximated by

$$\xi_{exc} \approx \mu\sigma_n^2 \text{tr}[\mathbf{R}] = \mu(N+1)\sigma_n^2\sigma_x^2 \tag{3.46}$$

where σ_x^2 is the input-signal variance and σ_n^2 is the additional-noise variance.

The misadjustment M, defined as the ratio between the ξ_{exc} and the minimum MSE, is a common parameter used to compare different adaptive signal processing algorithms. For the LMS algorithm the misadjustment is given by

$$M \triangleq \frac{\xi_{exc}}{\xi_{min}} = \frac{\mu \text{tr}[\mathbf{R}]}{1 - \mu \text{tr}[\mathbf{R}]} \tag{3.47}$$

3.3.6 Transient Behavior

Before the LMS algorithm reaches the steady-state behavior, a number of iterations are spent in the transient part. During this time, the adaptive filter coefficients and the output error change from their initial values to values close to that of the corresponding optimal solution.

In the case of the adaptive filter coefficients, the convergence in the mean will follow $(N+1)$ geometric decaying curves with ratios $r_{wi} = (1 - 2\mu\lambda_i)$. Each of these curves can be approximated by an exponential envelope with time constant τ_{wi} as follows (see equation (3.18)) [2]:

$$r_{wi} = e^{\frac{-1}{\tau_{wi}}} = 1 - \frac{1}{\tau_{wi}} + \frac{1}{2!\tau_{wi}^2} + \cdots \tag{3.48}$$

where for each iteration, the decay in the exponential envelope is equal to the decay in the original geometric curve. In general, r_{wi} is slightly smaller than one, especially in the case of the slowly decreasing modes corresponding to small λ_i and μ. Therefore,

$$r_{wi} = (1 - 2\mu\lambda_i) \approx 1 - \frac{1}{\tau_{wi}} \tag{3.49}$$

then

$$\tau_{wi} = \frac{1}{2\mu\lambda_i}$$

for $i = 0, 1, \ldots, N$. Note that in order to guarantee convergence of the tap coefficients in the mean, μ must be chosen in the range $0 < \mu < 1/\lambda_{max}$ (see equation (3.19)).

According to equation (3.30), for the convergence of the MSE the range of values for μ is $0 < \mu < 1/\text{tr}[\mathbf{R}]$, and the corresponding time constant can be calculated from matrix $\mathbf{B}$ in equation (3.27), by considering the terms in

μ^2 small as compared to the remaining terms in matrix $\mathbf{B}$. In this case, the geometric decaying curves have ratios given by $r_{ei} = (1 - 4\mu\lambda_i)$ that can be fitted to exponential envelopes with time constants given by

$$\tau_{ei} = \frac{1}{4\mu\lambda_i} \tag{3.50}$$

for $i = 0, 1, \ldots, N$. In the convergence of both the error and the coefficients, the time required for the convergence depends on the ratio of eigenvalues of the input signal correlation matrix. For example, if μ is chosen to be approximately $1/\lambda_{max}$ the corresponding time constant for the MSE is given by

$$\tau_{ei} \approx \frac{\lambda_{max}}{4\lambda_i} \leq \frac{\lambda_{max}}{4\lambda_{min}} \tag{3.51}$$

Since the mode with the highest time constant takes longer to reach convergence, the rate of convergence is determined by the slowest mode given by $\tau_{e_{max}} = \lambda_{max}/(4\lambda_{min})$. Suppose the convergence is considered achieved when the slowest mode provides an attenuation of 100, i.e.,

$$e^{\frac{-k}{\tau_{e_{max}}}} = 0.01$$

this requires the following number of iterations in order to reach convergence:

$$k \approx 4.6\frac{\lambda_{max}}{4\lambda_{min}}$$

The situation above is quite optimistic because μ was chosen to be high. As mentioned before, in practice we should choose the value of μ much smaller than the upper bound. For an eigenvalue spread approximating one, μ should be smaller than $1/(N+1)\lambda_{max}$. In this case, the LMS algorithm will require at least

$$k \approx 4.6\frac{(N+1)\lambda_{max}}{4\lambda_{min}}$$

iterations to achieve the convergence.

The analytical results presented in this section are valid for stationary environments. The LMS algorithm can also operate in the case of nonstationary environments, as shown in the following section.

3.4 LMS ALGORITHM BEHAVIOR IN NONSTATIONARY ENVIRONMENTS

In practical situations, the environment in which the adaptive filter is embedded may be nonstationary. In these cases, the input-signal autocorrelation matrix and/or the cross-correlation vector, denoted respectively by $\mathbf{R}(k)$ and $\mathbf{p}(k)$, are/is varying with time. Therefore, the optimal solution for the coefficient vector is also a time-varying vector given by $\mathbf{w}_o(k)$.

Since the optimal coefficient vector is not fixed, it is important to analyze if the LMS algorithm will be able to track changes in $\mathbf{w}_o(k)$. It is also of interest to learn how the tracking error in the coefficients given by $E[\mathbf{w}(k)] - \mathbf{w}_o(k)$ will affect the output MSE. It will be shown later that the excess of MSE caused by lag in the tracking of $\mathbf{w}_o(k)$ can be separated from the excess of MSE caused by the measurement noise, and therefore, without loss of generality, in the following analysis the additional noise will be considered zero.

The coefficient-vector updating in the LMS algorithm can be written in the following form

$$\begin{aligned} \mathbf{w}(k+1) &= \mathbf{w}(k) + 2\mu\mathbf{x}(k)e(k) \\ &= \mathbf{w}(k) + 2\mu\mathbf{x}(k)(d(k) - \mathbf{x}^T(k)\mathbf{w}(k)) \end{aligned} \tag{3.52}$$

Since

$$d(k) = \mathbf{x}^T(k)\mathbf{w}_o(k) \tag{3.53}$$

the coefficient updating can be expressed as follows:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu\mathbf{x}(k)(\mathbf{x}^T(k)\mathbf{w}_o(k) - \mathbf{x}^T(k)\mathbf{w}(k)) \tag{3.54}$$

Now assume that an ensemble of a nonstationary adaptive identification process has been built, where the input signal in each experiment is taken from the same stochastic process. The input signal is considered stationary and ergodic. The first assumption results in a fixed $\mathbf{R}$ matrix, and the nonstationarity is caused by the desired signal that is generated by applying the input signal to a time-varying system. With these assumptions, by using the expected value operation to the ensemble, with the coefficient updating in each experiment given by equation (3.54), and assuming that $\mathbf{w}(k)$ is independent of $\mathbf{x}(k)$ yields

$$\begin{aligned} E[\mathbf{w}(k+1)] &= E[\mathbf{w}(k)] + 2\mu E[\mathbf{x}(k)\mathbf{x}^T(k)]\mathbf{w}_o(k) \\ &\quad -2\mu E[\mathbf{x}(k)\mathbf{x}^T(k)]E[\mathbf{w}(k)] \\ &= E[\mathbf{w}(k)] + 2\mu\mathbf{R}(\mathbf{w}_o(k) - E[\mathbf{w}(k)]) \end{aligned} \tag{3.55}$$

If the lag in the coefficient vector is defined by

$$\mathbf{l}_{\mathbf{W}}(k) = E[\mathbf{w}(k)] - \mathbf{w}_o(k) \tag{3.56}$$

equation (3.55) can be rewritten as

$$\mathbf{l}_{\mathbf{W}}(k+1) = (\mathbf{I} - 2\mu\mathbf{R})\mathbf{l}_{\mathbf{W}}(k) - \mathbf{w}_o(k+1) + \mathbf{w}_o(k) \tag{3.57}$$

In order to simplify our analysis, we can premultiply the equation above by $\mathbf{Q}^T$, resulting in a decoupled set of equations given by

$$\mathbf{l}'_{\mathbf{W}}(k+1) = (\mathbf{I} - 2\mu\mathbf{\Lambda})\mathbf{l}'_{\mathbf{W}}(k) - \mathbf{w}'_o(k+1) + \mathbf{w}'_o(k) \tag{3.58}$$

where the vectors with superscript are the original vectors projected onto the transformed space. As can be noted, each element of the lag-error vector is determined by the following relation

$$l'_i(k+1) = (1 - 2\mu\lambda_i)l'_i(k) - w'_{oi}(k+1) + w'_{oi}(k) \tag{3.59}$$

where $l'_i(k)$ is the ith element of $\mathbf{l}'_{\mathbf{W}}(k)$. By properly interpreting the equation above, we can say that the lag is generated by applying the transformed instantaneous optimal coefficient to a first-order discrete-time filter denoted *lag filter*, i.e.,

$$L'_i(z) = -\frac{z-1}{z-1+2\mu\lambda_i} W'_{oi}(z) \tag{3.60}$$

The discrete-time filter transient response converges with a time constant of the exponential envelope given by

$$\tau_i = \frac{1}{2\mu\lambda_i} \tag{3.61}$$

which is of course different for each individual tap. Therefore, the tracking ability of the coefficients in the LMS algorithm is dependent on the eigenvalues of the input-signal correlation matrix.

The lag in the adaptive filter coefficients leads to an excess of mean-square error. In order to calculate the excess of MSE, suppose that each element of the optimal coefficient vector is modeled as a first-order Markov process. This nonstationary situation can be considered somewhat simplified as compared with some real practical situations. However, it allows a manageable mathematical analysis while retaining the essence of handling the more complicated cases. The first-order Markov process is described by

$$\mathbf{w}_o(k) = \lambda_{\mathbf{W}}\mathbf{w}_o(k-1) + \mathbf{n}_{\mathbf{W}}(k) \tag{3.62}$$

where $\mathbf{n_W}(k)$ is a vector whose elements are zero-mean Gaussian noise processes with variance $\sigma^2_{\mathbf{W}}$, and $\lambda_{\mathbf{W}} < 1$. Note that $(1 - 2\mu\lambda_i) < \lambda_{\mathbf{W}} < 1$, for $i = 0, 1, \ldots, N$, since the optimal coefficients values must vary slower than the filter tracking speed, i.e., $\frac{1}{2\mu\lambda_i} \ll \frac{1}{1-\lambda_{\mathbf{W}}}$. This model may not represent a real system when $\lambda_{\mathbf{W}} \to 1$, since the $\mathrm{cov}[\mathbf{w}_o(k)]$ will have unbounded elements if, for example, $\mathbf{n_W}(k)$ is not exactly zero mean. A more realistic model would include a factor $(1 - \lambda_{\mathbf{W}})^{\frac{p}{2}}$, for $p \geq 1$, multiplying $\mathbf{n_W}(k)$ in order to guarantee that $\mathrm{cov}[\mathbf{w}_o(k)]$ is bounded. In the following discussions, this case will not be considered since the corresponding results can be easily derived.

From equations (3.59) and (3.60), we can infer that the lag-error vector elements are generated by applying a first-order discrete-time system to the elements of the unknown system coefficient vector, both in the transformed space. On the other hand, the coefficients of the unknown system are generated by applying each element of the noise vector $\mathbf{n_W}(k)$ to a first-order all-pole filter, with the pole placed at $\lambda_{\mathbf{W}}$. For the unknown coefficient vector with the model above, the lag-error vector elements can be generated by applying the elements of the transformed noise vector $\mathbf{n'_W}(k) = \mathbf{Q}^T\mathbf{n_W}(k)$ to a discrete-time filter with transfer function

$$H(z) = \frac{-(z-1)z}{(z-1+2\mu\lambda_i)(z-\lambda_{\mathbf{W}})} \tag{3.63}$$

This transfer function consists of a cascade of the lag filter with the all-pole filter representing the first-order Markov process as illustrated in Fig. 3.2. Using the inverse $\mathcal{Z}$-transform, the variance of the elements of the vector $\mathbf{l'_W}(k)$ can then be calculated by

$$\begin{aligned} E[l_i'^2(k)] &= \frac{1}{2\pi_J}\oint H(z)H(z^{-1})\sigma^2_{\mathbf{W}}z^{-1}\,dz \\ &= \left(\frac{1}{1-\lambda_{\mathbf{W}}-2\mu\lambda_i}\right)^2 \\ &\quad \left[\frac{\mu\lambda_i}{1-\mu\lambda_i} + \frac{1-\lambda_{\mathbf{W}}}{1+\lambda_{\mathbf{W}}} + \frac{4(1-\lambda_{\mathbf{W}})\mu\lambda_i}{1-\lambda_{\mathbf{W}}+2\mu\lambda_{\mathbf{W}}\lambda_i}\right]\sigma^2_{\mathbf{W}} \end{aligned} \tag{3.64}$$

If $\lambda_{\mathbf{W}}$ is considered very close to 1, it is possible to simplify the equation above as

$$E[l_i'^2(k)] = \frac{\sigma^2_{\mathbf{W}}}{4\mu\lambda_i(1-\mu\lambda_i)} \tag{3.65}$$

Any error in the coefficient vector of the adaptive filter as compared to the optimal coefficient filter generates an excess of MSE (see equation 3.38). Since the lag is one source of error in the adaptive filter coefficients, then the excess

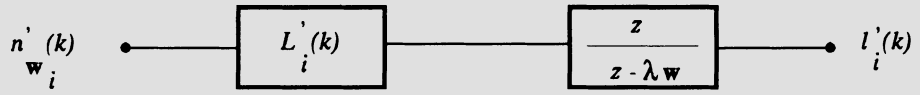


Figure 3.2 Lag model in nonstationary environment.

of MSE due to lag is given by

$$\begin{aligned}
\xi_{lag} &= E[\mathbf{l}_{\mathbf{W}}^T(k)\mathbf{R}\mathbf{l}_{\mathbf{W}}(k)] \\
&= E[\mathrm{tr}(\mathbf{R}\mathbf{l}_{\mathbf{W}}(k)\mathbf{l}_{\mathbf{W}}^T(k))] \\
&= \mathrm{tr}(\mathbf{R}E[\mathbf{l}_{\mathbf{W}}(k)\mathbf{l}_{\mathbf{W}}^T(k)]) \\
&= \mathrm{tr}(\mathbf{\Lambda}E[\mathbf{l}'_{\mathbf{W}}(k)\mathbf{l}'^T_{\mathbf{W}}(k)]) \\
&= \sum_{i=0}^{N} \lambda_i E[l_i'^2(k)] \\
&= \frac{\sigma_{\mathbf{W}}^2}{4\mu} \sum_{i=0}^{N} \frac{1}{1-\mu\lambda_i}
\end{aligned} \tag{3.66}$$

If μ is very small, the MSE due to lag tends to infinity indicating that the LMS algorithm, in this case, cannot track any change in the environment. On the other hand, for μ appropriately chosen the algorithm can track variations in the environment leading to an excess of MSE. This excess of MSE depends on the variance of the optimal coefficient disturbance and on the values of the input-signal autocorrelation matrix eigenvalues, as indicated in equation (3.66).

Now we analyze how the error due to lag interacts with the error generated by the noisy calculation of the gradient in the LMS algorithm. The overall error in the taps is given by

$$\Delta\mathbf{w}(k) = \mathbf{w}(k) - \mathbf{w}_o(k) = (\mathbf{w}(k) - E[\mathbf{w}(k)]) + (E[\mathbf{w}(k)] - \mathbf{w}_o(k)) \tag{3.67}$$

where the first error in the equation above is due to the additional noise and the second is the error due to lag. The overall excess of MSE can then be expressed as

$$\begin{aligned}
\xi_{total} &= E[(\mathbf{w}(k) - \mathbf{w}_o(k))^T\mathbf{R}(\mathbf{w}(k) - \mathbf{w}_o(k))] \\
&\approx E[(\mathbf{w}(k) - E[\mathbf{w}(k)])^T\mathbf{R}(\mathbf{w}(k) - E[\mathbf{w}(k)])] \\
&\quad + E[(E[\mathbf{w}(k)] - \mathbf{w}_o(k))^T\mathbf{R}(E[\mathbf{w}(k)] - \mathbf{w}_o(k))]
\end{aligned} \tag{3.68}$$

since $2E[(\mathbf{w}(k) - E[\mathbf{w}(k)])^T\mathbf{R}(E[\mathbf{w}(k)] - \mathbf{w}_o(k))] \approx 0$, if we consider the fact that $\mathbf{w}_o(k)$ is kept fixed in each experiment of the ensemble. As a consequence,

an estimate for the overall excess of MSE can be obtained by adding the results of equations (3.45) and (3.66), i.e.,

$$\xi_{total} \approx \frac{\mu\sigma_n^2 \mathrm{tr}[\mathbf{R}]}{1-\mu\mathrm{tr}[\mathbf{R}]} + \frac{\sigma_{\mathbf{w}}^2}{4\mu}\sum_{i=0}^{N}\frac{1}{1-\mu\lambda_i} \tag{3.69}$$

If small μ is employed, the above equation can be simplified as follows:

$$\xi_{total} \approx \mu\sigma_n^2 \mathrm{tr}[\mathbf{R}] + \frac{\sigma_{\mathbf{w}}^2}{4\mu}(N+1) \tag{3.70}$$

Differentiating the equation above with respect to μ and setting the result to zero yields an optimum value for μ given by

$$\mu_{opt} = \sqrt{\frac{(N+1)\sigma_{\mathbf{w}}^2}{4\sigma_n^2 \mathrm{tr}[\mathbf{R}]}} \tag{3.71}$$

The μ_{opt} is supposed to lead to the minimum excess of MSE. However, the user should bear in mind that the μ_{opt} can only be used if it satisfies stability conditions, and if its value can be considered small enough to validate equation (3.70). Also this value is optimum only when quantization effects are not taken into consideration, where for short wordlength implementation the best μ should be chosen following the guidelines given in the following section. It should also be mentioned that the study of the misadjustment due to nonstationarity of the environment is considerably more complicated when the input signal and the desired signal are simultaneously nonstationary [8], [10]-[15]. Therefore, the analysis presented here is only valid if the assumptions made are valid. However, the simplified analysis provides a good sample of the LMS algorithm behavior in a nonstationary environment and gives a general indication of what can be expected in more complicated situations.

3.5 QUANTIZATION EFFECTS

The results of the analysis of the previous sections are obtained assuming that the algorithm is implemented with infinite precision. However, the widespread use of adaptive filtering algorithms in real-time requires their implementation with short wordlength, in order to meet the speed requirements. When implemented with short-wordlength precision the LMS algorithm behavior can be very different from what is expected in infinite precision. In particular, when the convergence factor μ tends to zero it is expected that the minimum mean-square error is reached in steady state; however, due to quantization effects the

MSE tends to increase significantly if μ is reduced below a certain value. In fact, the algorithm can stop updating some filter coefficients if μ is not chosen appropriately. In this section, several aspects of the finite-wordlength effects in the LMS algorithm are discussed for the case of implementation in fixed- and floating-point arithmetics [16]-[18].

3.5.1 Error Description

All scalars and vector elements in the LMS algorithm will deviate from their correct values due to quantization effects. The error generated in any individual quantization is considered to be a zero-mean random variable that is independent of any other errors and quantities related to the adaptive filter algorithm. The variances of these errors depend on the type of quantization and arithmetic that will be employed in the algorithm implementation.

The errors in the quantities related to the LMS algorithm are defined by

$$n_e(k) = e(k) - e(k)_Q \tag{3.72}$$

$$\mathbf{n_w}(k) = \mathbf{w}(k) - \mathbf{w}(k)_Q \tag{3.73}$$

$$n_y(k) = y(k) - y(k)_Q \tag{3.74}$$

where the subscript Q denotes the quantized form of the given value or vector.

It is assumed that the input signal and desired signal suffer no quantization, so that only internal computation quantizations are taken into account. The effects of quantization in the input and desired signals can be easily taken into consideration separately from other quantization error sources. In the case of the desired signal, the quantization error can be added to the measurement noise, while for the input signal the basic effect at the output of the filter is an additional noise as will be discussed later.

The following relations describe the computational errors introduced in the LMS algorithm implemented with finite wordlength:

$$e(k)_Q = d(k) - \mathbf{x}^T(k)\mathbf{w}(k)_Q - n_e(k) \tag{3.75}$$

$$\mathbf{w}(k+1)_Q = \mathbf{w}(k)_Q + 2\mu e(k)_Q \mathbf{x}(k) - \mathbf{n_w}(k) \tag{3.76}$$

where $n_e(k)$ is the noise sequence due to quantization in the inner product $\mathbf{x}^T(k)\mathbf{w}(k)_Q$, the additional measurement noise $n(k)$ is included in $d(k)$, and $\mathbf{n_w}(k)$ is a noise vector generated by quantization in the product $2\mu e(k)_Q \mathbf{x}(k)$.

Algorithm 3.2

LMS Algorithm Including Quantization

Initialization

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^T$

Do for $k \geq 0$

$e(k)_Q = (d(k) - \mathbf{x}^T(k)\mathbf{w}(k)_Q)_Q$

$\mathbf{w}(k+1)_Q = (\mathbf{w}(k)_Q + 2\mu e(k)_Q \mathbf{x}(k))_Q$

□

The generation of quantization noise as described applies for fixed-point arithmetic, whereas for floating-point arithmetic the addition also introduces quantization error that should be included in $n_e(k)$ and $\mathbf{n_w}(k)$.

The objective now is to study the LMS algorithm behavior when internal computations are performed in finite precision. Algorithm 3.2 describes the LMS algorithm including quantization and with presence of additional noise.

Define

$$\Delta\mathbf{w}(k)_Q = \mathbf{w}(k)_Q - \mathbf{w}_o \tag{3.77}$$

where $\mathbf{w}_o$ is the optimal coefficient vector, and considering that

$$d(k) = \mathbf{x}^T(k)\mathbf{w}_o + n(k) \tag{3.78}$$

it then follows that

$$\begin{aligned} e(k)_Q &= (d(k) - \mathbf{x}^T(k)\mathbf{w}(k)_Q)_Q \\ &= -\mathbf{x}^T(k)\Delta\mathbf{w}(k)_Q - n_e(k) + n(k) \end{aligned} \tag{3.79}$$

and from equation (3.76)

$$\begin{aligned} \Delta\mathbf{w}(k+1)_Q &= \Delta\mathbf{w}(k)_Q + 2\mu\mathbf{x}(k)\left[-\mathbf{x}^T(k)\Delta\mathbf{w}(k)_Q - n_e(k) + n(k)\right] \\ &\quad -\mathbf{n_w}(k) \end{aligned} \tag{3.80}$$

This equation can be rewritten as

$$\Delta \mathbf{w}(k+1)_Q = \left[\mathbf{I} - 2\mu \mathbf{x}(k)\mathbf{x}^T(k)\right] \Delta \mathbf{w}(k)_Q + \mathbf{n}'_{\mathbf{W}}(k) \tag{3.81}$$

where

$$\mathbf{n}'_{\mathbf{W}}(k) = 2\mu \mathbf{x}(k)(n(k) - n_e(k)) - \mathbf{n}_{\mathbf{W}}(k) \tag{3.82}$$

For the sake of illustration and completeness, the solution of the equation (3.81) is

$$\begin{aligned} \Delta \mathbf{w}(k+1)_Q &= \prod_{i=0}^{k} \left[\mathbf{I} - 2\mu \mathbf{x}(i)\mathbf{x}^T(i)\right] \Delta \mathbf{w}(0)_Q \\ &\quad + \sum_{i=0}^{k} \left\{ \prod_{j=i+1}^{k} \left[\mathbf{I} - 2\mu \mathbf{x}(j)\mathbf{x}^T(j)\right] \mathbf{n}'_{\mathbf{W}}(i) \right\} \end{aligned} \tag{3.83}$$

where we define that for $j = k+1$ in the second product, $\prod_{j=k+1}^{k}[\cdot] = 1$.

3.5.2 Error Models for Fixed-Point Arithmetic

In the case of fixed-point arithmetic, with rounding assumed for quantization, the error after each product can be modeled as a zero-mean stochastic process, with variance given by [19]-[20]

$$\sigma^2 = \frac{2^{-2b}}{12} \tag{3.84}$$

where b is the number of bits after the sign bit. Here it is assumed that the number of bits after the sign bit for quantities representing signals and filter coefficients are different and given by b_d and b_c, respectively. It is also assumed that the internal signals are properly scaled, so that no overflow occurs during the computations and that the signal values lie between -1 and +1 all the time. The error signals consisting of the elements of $n_e(k)$ and $\mathbf{n}_{\mathbf{W}}(k)$ are all uncorrelated and independent of each other. The variance of $n_e(k)$ and the covariance of $\mathbf{n}_{\mathbf{W}}(k)$ are given by

$$E[n_e^2(k)] = \sigma_e^2 \tag{3.85}$$

$$E[\mathbf{n}_{\mathbf{W}}(k)\mathbf{n}_{\mathbf{W}}^T(k)] = \sigma_{\mathbf{W}}^2 \mathbf{I} \tag{3.86}$$

respectively. If distinction is made between data and coefficient wordlengths, the above mentioned variances are given by

$$\sigma_e^2 = \sigma_y^2 = \gamma \frac{2^{-2b_d}}{12} \tag{3.87}$$

$$\sigma_{\mathbf{w}}^2 = \gamma' \frac{2^{-2b_c}}{12} \tag{3.88}$$

where $\gamma' = \gamma = 1$ if the quantization is performed after addition, i.e., products are performed in full precision and only after all the additions in the inner product are finished is the quantization applied. For quantization after each product, $\gamma = N + 1$ where $N + 1$ is the number of partial products. Those not familiar with the results of the equations above should consult a basic digital signal processing textbook such as [19] or [20].

Note that $\sigma_{\mathbf{w}}^2$ depends on how the product $2\mu e(k)\mathbf{x}(k)$ is performed. In the equation above, it was assumed that the product was available in full precision, and then a quantization to b_c bits in the fractional part was performed, or equivalently, the product $2\mu e(k)$ in full precision was multiplied by $\mathbf{x}(k)$, and only in the last operation quantization was introduced. In case of quantization of partial results, the variance $\sigma_{\mathbf{w}}^2$ is increased due to the products of partial errors with the remaining product components.

3.5.3 Coefficient-Error-Vector Covariance Matrix

Obviously, internal quantization noise generated during the operation of the LMS algorithm affects its convergence behavior. In this subsection, we discuss the effects of the finite wordlength computations on the second-order statistics of the errors in the adaptive filter coefficients. First, we assume that the quantization noise $n_e(k)$ and the vector $\mathbf{n}_{\mathbf{W}}(k)$ are all independent of the data, of the filter coefficients, and of each other. Also, these quantization errors are all considered zero-mean stochastic processes. With these assumptions, the covariance of the error in the coefficient vector, defined by $E[\Delta\mathbf{w}(k)_Q \Delta\mathbf{w}^T(k)_Q]$, can be easily derived from equations (3.81) and (3.82):

$$\begin{aligned}
\text{cov}[\Delta\mathbf{w}(k+1)_Q] &= E[\Delta\mathbf{w}(k+1)_Q \Delta\mathbf{w}^T(k+1)_Q] \\
&= E\left\{\left[\mathbf{I} - 2\mu\mathbf{x}(k)\mathbf{x}^T(k)\right] \Delta\mathbf{w}(k)_Q \Delta\mathbf{w}^T(k)_Q \right. \\
&\quad \left. \left[\mathbf{I} - 2\mu\mathbf{x}(k)\mathbf{x}^T(k)\right]\right\} \\
&\quad + 4\mu^2 E[\mathbf{x}(k)\mathbf{x}^T(k)]E[n^2(k)] \\
&\quad + 4\mu^2 E[\mathbf{x}(k)\mathbf{x}^T(k)]E[n_e^2(k)] \\
&\quad + E[\mathbf{n}_{\mathbf{W}}(k)\mathbf{n}_{\mathbf{W}}^T(k)]
\end{aligned} \tag{3.89}$$

Each term in the right hand side of the above equation can be approximated in order to derive the solution for the overall equation. The only assumption made

is the independence between $\mathbf{x}(k)$ and $\Delta\mathbf{w}(k)_Q$ that is reasonably accurate in practice.

The first term in equation (3.89) can be expressed as

$$\begin{aligned}\mathbf{T}_1 &= \text{cov}[\Delta\mathbf{w}(k)_Q] - 2\mu\text{cov}[\Delta\mathbf{w}(k)_Q]E[\mathbf{x}(k)\mathbf{x}^T(k)] \\ &\quad -2\mu E[\mathbf{x}(k)\mathbf{x}^T(k)]\text{cov}[\Delta\mathbf{w}(k)_Q] \\ &\quad +4\mu^2 E\left\{\mathbf{x}(k)\mathbf{x}^T(k)\text{cov}[\Delta\mathbf{w}(k)_Q]\mathbf{x}(k)\mathbf{x}^T(k)\right\} \end{aligned} \tag{3.90}$$

The element (i,j) of the last term in the equation above is given by

$$\begin{aligned}&4\mu^2 E\left\{\mathbf{x}(k)\mathbf{x}^T(k)\text{cov}[\Delta\mathbf{w}(k)_Q]\mathbf{x}(k)\mathbf{x}^T(k)\right\}_{ij} \\ &= 4\mu^2 \sum_{m=0}^{N}\sum_{l=0}^{N} \text{cov}[\Delta\mathbf{w}(k)_Q]_{m,l} E[x_i(k)x_m(k)x_l(k)x_j(k)] \end{aligned} \tag{3.91}$$

where $x_i(k)$ represents the ith element of $\mathbf{x}(k)$. If it is assumed that the elements of the input-signal vector are jointly Gaussian, the following relation is valid

$$E[x_i(k)x_m(k)x_l(k)x_j(k)] = \mathbf{R}_{i,m}\mathbf{R}_{l,j} + \mathbf{R}_{m,l}\mathbf{R}_{i,j} + \mathbf{R}_{m,j}\mathbf{R}_{i,l} \tag{3.92}$$

where $\mathbf{R}_{ij}$ is the element (i,j) of $\mathbf{R}$. Replacing this expression in equation (3.91), it can be easily shown that

$$\begin{aligned}&\sum_{m=0}^{N}\sum_{l=0}^{N} \text{cov}[\Delta\mathbf{w}(k)_Q]_{m,l} E[x_i(k)x_m(k)x_l(k)x_j(k)] \\ &= 2\left\{\mathbf{R}\text{cov}[\Delta\mathbf{w}(k)_Q]\mathbf{R}\right\}_{i,j} + \mathbf{R}_{i,j}\text{tr}\left\{\mathbf{R}\text{cov}[\Delta\mathbf{w}(k)_Q]\right\} \end{aligned} \tag{3.93}$$

Using this result in the last term of $\mathbf{T}_1$, it follows that

$$\begin{aligned}\mathbf{T}_1 &= \text{cov}[\Delta\mathbf{w}(k)_Q] - 2\mu(\mathbf{R}\text{cov}[\Delta\mathbf{w}(k)_Q] + \text{cov}[\Delta\mathbf{w}(k)_Q]\mathbf{R}) \\ &\quad +4\mu^2\left(2\mathbf{R}\text{cov}[\Delta\mathbf{w}(k)_Q]\mathbf{R} + \mathbf{R}\text{tr}\left\{\mathbf{R}\text{cov}[\Delta\mathbf{w}(k)_Q]\right\}\right) \end{aligned} \tag{3.94}$$

Since the remaining terms in equation (3.89) are straightforward to calculate, replacing equation (3.94) in (3.89) yields

$$\begin{aligned}\text{cov}[\Delta\mathbf{w}(k+1)_Q] &= (\mathbf{I} - 2\mu\mathbf{R})\text{cov}[\Delta\mathbf{w}(k)_Q] - 2\mu\text{cov}[\Delta\mathbf{w}(k)_Q]\mathbf{R} \\ &\quad +4\mu^2\mathbf{R}\text{tr}\left\{\mathbf{R}\text{cov}[\Delta\mathbf{w}(k)_Q]\right\} + 8\mu^2\mathbf{R}\text{cov}[\Delta\mathbf{w}(k)_Q]\mathbf{R} \\ &\quad +4\mu^2(\sigma_n^2 + \sigma_e^2)\mathbf{R} + \sigma_{\mathbf{w}}^2\mathbf{I} \end{aligned} \tag{3.95}$$

Before reaching the steady state, the covariance of $\Delta\mathbf{w}(k+1)_Q$ presents a transient behavior that can be analyzed in the same form as equation (3.26).

It is worth mentioning that the condition for convergence of the coefficients given in equation (3.30) also guarantees the convergence of the equation above. In fact, equation (3.95) is almost the same as equation (3.26) except for the extra excitation terms σ_e^2 and $\sigma_{\mathbf{w}}^2$ that account for the quantization effects, and, therefore, the behavior of the coefficients in the LMS algorithm in finite precision must resemble its behavior in infinite precision, with the convergence curve shifted up in the finite precision case.

In most cases, the norm of $\mathbf{R}\text{cov}[\Delta\mathbf{w}(k)_Q]\mathbf{R}$ is much smaller than the norm of $\mathbf{R}\text{tr}\{\mathbf{R}\text{cov}[\Delta\mathbf{w}(k)_Q]\}$ and can be eliminated from equation (3.95). Now, by considering in equation (3.95) that in the steady state $\text{cov}[\Delta\mathbf{w}(k)_Q] \approx \text{cov}[\Delta\mathbf{w}(k+1)_Q]$ and applying the trace operation in both sides, it is easy to conclude that

$$\text{tr}\{\mathbf{R}\text{cov}[\Delta\mathbf{w}(k)_Q]\} = \frac{4\mu^2(\sigma_n^2+\sigma_e^2)\text{tr}[\mathbf{R}] + (N+1)\sigma_{\mathbf{w}}^2}{4\mu - 4\mu^2\text{tr}[\mathbf{R}]} \tag{3.96}$$

This expression will be useful to calculate the excess of MSE in the finite-precision implementation of the LMS algorithm.

If $x(k)$ is considered a Gaussian white noise with variance σ_x^2, it is possible to calculate the expected value of $||\Delta\mathbf{w}(k)_Q||^2$, defined as the trace of $\text{cov}[\Delta\mathbf{w}(k)_Q]$, from equations (3.95) and (3.96). The result is

$$E[||\Delta\mathbf{w}(k)_Q||^2] = \frac{\mu(\sigma_n^2+\sigma_e^2)(N+1)}{1-\mu(N+1)\sigma_x^2} + \frac{(N+1)\sigma_{\mathbf{w}}^2}{4\mu\sigma_x^2(1-\mu(N+1)\sigma_x^2)} \tag{3.97}$$

As can be noted, when μ is small, the noise in the calculation of the coefficients plays a major role in the overall error in the adaptive filter coefficients.

3.5.4 Algorithm Stop

The adaptive filter coefficients may stop updating due to limited wordlength employed in the internal computation. In particular, for the LMS algorithm, it will occur when

$$|2\mu e(k)_Q\mathbf{x}(k)|_i < 2^{-b_c-1} \tag{3.98}$$

where $|\cdot|_i$ denotes the modulus of the ith component of . The condition above can be stated in an equivalent form given by

$$4\mu^2(\sigma_e^2+\sigma_n^2)\sigma_x^2 < 4\mu^2E[e^2(k)_Q]E[x_i^2(k)] < \frac{2^{-2b_c}}{4} \tag{3.99}$$

where in the first inequality it was considered that the variances of all elements of $\mathbf{x}(k)$ are the same, and that $\sigma_e^2 + \sigma_n^2$ is a lower bound for $E[e^2(k)_Q]$ since the effect of misadjustment due to noise in the gradient is not considered. If μ is chosen such that

$$\mu > \frac{2^{-b_c}}{4\sigma_x\sqrt{\sigma_e^2 + \sigma_n^2}} \tag{3.100}$$

the algorithm will not stop before convergence is reached. If μ is small such that the convergence is not reached, the MSE at the output of the adaptive system will be totally determined by the quantization error. In this case, the quantization error is usually larger than the expected MSE in the infinite-precision implementation.

3.5.5 Mean-Square Error

The mean-square error of the conventional LMS algorithm in the presence of quantization noise is given by

$$\xi(k)_Q = E[e^2(k)_Q] \tag{3.101}$$

By recalling that $e(k)_Q$ can be expressed as

$$e(k)_Q = -\mathbf{x}^T(k)\Delta\mathbf{w}(k)_Q - n_e(k) + n(k) \tag{3.102}$$

it then follows that

$$\begin{aligned}
\xi(k)_Q &= E[\mathbf{x}^T(k)\Delta\mathbf{w}(k)_Q\mathbf{x}^T(k)\Delta\mathbf{w}(k)_Q] + \sigma_e^2 + \sigma_n^2 \\
&= E\left\{\text{tr}[\mathbf{x}(k)\mathbf{x}^T(k)\Delta\mathbf{w}(k)_Q\Delta\mathbf{w}^T(k)_Q]\right\} + \sigma_e^2 + \sigma_n^2 \\
&= \text{tr}\left\{\mathbf{R}\text{cov}[\Delta\mathbf{w}(k)_Q]\right\} + \sigma_e^2 + \sigma_n^2
\end{aligned} \tag{3.103}$$

If we replace the equation (3.96) in (3.103), the MSE of the adaptive system is given by

$$\begin{aligned}
\xi(k)_Q &= \frac{\mu(\sigma_n^2 + \sigma_e^2)\text{tr}[\mathbf{R}]}{1 - \mu\text{tr}[\mathbf{R}]} + \frac{(N+1)\sigma_{\mathbf{W}}^2}{4\mu(1 - \mu\text{tr}[\mathbf{R}])} + \sigma_e^2 + \sigma_n^2 \\
&= \frac{\sigma_e^2 + \sigma_n^2}{1 - \mu\text{tr}[\mathbf{R}]} + \frac{(N+1)\sigma_{\mathbf{W}}^2}{4\mu(1 - \mu\text{tr}[\mathbf{R}])}
\end{aligned} \tag{3.104}$$

This formula is valid as long as the algorithm does not stop updating the coefficients. However, the MSE tends to increase in a form similar to that determined in equation (3.104) when μ does not satisfy equation (3.100).

In case the input signal is also quantized, a noise with variance σ_i^2 is generated at the input, causing an increase in the MSE. This noise will have a direct gain to the output given by (see problem 18)

$$\sigma_i^2(||\mathbf{w}_o||^2 + \text{tr}\{\text{cov}[\Delta\mathbf{w}(k)_Q]\}) \approx \sigma_i^2||\mathbf{w}_o||^2$$

corresponding to the noise transfer function from the adaptive filter input to the output. However, the result of this term being fedback in the algorithm through the error signal generates an extra term in the MSE with the same gain as the measurement noise that is approximately given by

$$\frac{\mu\sigma_i^2\text{tr}[\mathbf{R}]}{1-\mu\text{tr}[\mathbf{R}]}||\mathbf{w}_o||^2$$

Therefore, the total contribution of the input signal quantization is

$$\xi_i \approx \frac{||\mathbf{w}_o||^2\sigma_i^2}{1-\mu\text{tr}[\mathbf{R}]} \tag{3.105}$$

where in the analysis above it was considered that the terms with $\sigma_i^2 \cdot \sigma_{\mathbf{w}}^2$, $\sigma_i^2 \cdot \sigma_e^2$, and $\sigma_i^2 \cdot \sigma_n^2$ are small enough to be neglected.

3.5.6 Floating-Point Arithmetic Implementation

A succinct analysis of the quantization effects in the LMS algorithm when implemented in floating-point arithmetic is presented in this section. Most of the derivations are given in the Appendix and follow closely the procedure of the fixed-point analysis.

In floating-point arithmetic, quantization errors occur after addition and multiplication operations. These errors are respectively modeled as follows [21]:

$$fl[a+b] = a+b-(a+b)n_a \tag{3.106}$$

$$fl[a\cdot b] = a\cdot b-(a\cdot b)n_p \tag{3.107}$$

where n_a and n_p are zero-mean random variables that are independent of any other errors. Their variances are respectively given by

$$\sigma_{n_p}^2 \approx 0.18\ \cdot 2^{-2b} \tag{3.108}$$

and

$$\sigma_{n_a}^2 < \sigma_{n_p}^2 \tag{3.109}$$

where b is the number of bits in the mantissa representation.

The quantized error and the quantized filter coefficients vector are given by

$$e(k)_Q = d(k) - \mathbf{x}^T(k)\mathbf{w}(k)_Q - n_e(k) \tag{3.110}$$

$$\mathbf{w}(k+1)_Q = \mathbf{w}(k)_Q + 2\mu\mathbf{x}(k)e(k)_Q - \mathbf{n}_\mathbf{W}(k) \tag{3.111}$$

where $n_e(k)$ and $\mathbf{n}_\mathbf{W}(k)$ represent computational errors, and their expressions are given in the Appendix. Since $\mathbf{n}_\mathbf{W}(k)$ is a zero-mean vector, it is shown in the Appendix that in the average $\mathbf{w}(k)_Q$ tends to $\mathbf{w}_o$. Also, it can be shown that

$$\begin{aligned}\Delta\mathbf{w}(k+1)_Q = {} & [\mathbf{I} - 2\mu\mathbf{x}(k)\mathbf{x}^T(k) + \mathbf{N}_{\Delta\mathbf{W}}(k)]\Delta\mathbf{w}(k) \\ & + \mathbf{N}'_a(k)\mathbf{w}_o + 2\mu\mathbf{x}(k)[n(k) - n_e(k)]\end{aligned} \tag{3.112}$$

where $\mathbf{N}_{\Delta\mathbf{W}}(k)$ combines several quantization-noise effects as discussed in the Appendix, and $\mathbf{N}'_a(k)$ is a diagonal noise matrix that models the noise generated in the vector addition required to update $\mathbf{w}(k+1)_Q$. The error matrix $\mathbf{N}_{\Delta\mathbf{W}}(k)$ can be considered negligible as compared to $(\mathbf{I} - 2\mu\mathbf{x}(k)\mathbf{x}^T(k))$ and therefore is eliminated in the analysis below.

By following a similar analysis used to derive equation (3.95) in the case of fixed-point arithmetic, we obtain

$$\begin{aligned}&\mathrm{tr}\,\{\mathbf{R}\mathrm{cov}[\Delta\mathbf{w}(k)_Q]\} \\ = {} & \frac{4\mu^2(\sigma_n^2 + \sigma_e^2)\mathrm{tr}[\mathbf{R}] + ||\mathbf{w}_o||^2\sigma_{n_a}^2 + \mathrm{tr}\{\mathrm{cov}[\Delta\mathbf{w}(k)]\}\sigma_{n_a}^2}{4\mu - 4\mu^2\mathrm{tr}[\mathbf{R}]}\end{aligned} \tag{3.113}$$

where it was considered that all noise sources in matrix $\mathbf{N}'_a(k)$ have the same variance given by $\sigma_{n_a}^2$.

If $x(k)$ is considered a Gaussian white noise with variance σ_x^2, it is straightforward to calculate $E[||\mathbf{w}(k)_Q||^2]$. The expression is given by

$$\begin{aligned}E[||\mathbf{w}(k)_Q||^2] = {} & \frac{\mu(\sigma_n^2 + \sigma_e^2)(N+1)}{1 - \mu(N+1)\sigma_x^2} \\ & + \frac{||\mathbf{w}_o||^2\sigma_{n_a}^2}{4\mu\sigma_x^2(1 - \mu(N+1)\sigma_x^2)} \\ & + \frac{\sigma_{n_a}^2\sigma_n^2(N+1)}{4\sigma_x^2(1 - \mu(N+1)\sigma_x^2)^2}\end{aligned} \tag{3.114}$$

where $\mathrm{tr}\{\mathrm{cov}[\Delta\mathbf{w}(k)]\}$ is given in the Appendix. For small values of μ, the quantization of addition in the updating of $\mathbf{w}(k)_Q$ may be the dominant source of error in the adaptive filter coefficients.

The MSE in the LMS algorithm implemented with floating-point arithmetic is then given by

$$\begin{aligned}\xi(k)_Q &= \mathrm{tr}\{\mathbf{R}\mathrm{cov}[\Delta\mathbf{w}(k)_Q]\} + \sigma_e^2 + \sigma_n^2 \\ &= \frac{(\sigma_n^2+\sigma_e^2)}{1-\mu\mathrm{tr}[\mathbf{R}]} + \frac{||\mathbf{w}_o||^2\sigma_{n_a}^2 + \mathrm{tr}\{\mathrm{cov}[\Delta\mathbf{w}(k)]\}\sigma_{n_a}^2}{4\mu(1-\mu\mathrm{tr}[\mathbf{R}])}\end{aligned} \tag{3.115}$$

For $\mu \ll \mathrm{tr}[\mathbf{R}]$, and again considering $x(k)$ a Gaussian white noise with variance σ_x^2, the equation above can be simplified as follows:

$$\xi(k)_Q = \sigma_n^2 + \sigma_e^2 + \frac{||\mathbf{w}_o||^2\sigma_{n_a}^2}{4\mu} + \frac{(N+1)\sigma_n^2\sigma_{n_a}^2}{4} \tag{3.116}$$

The ith coefficient of the adaptive filter will not be updated in floating-point implementation if

$$|2\mu e(k)_Q\mathbf{x}(k)|_i < 2^{-b_a-1}|\mathbf{w}(k)|_i \tag{3.117}$$

where $|\cdot|_i$ denotes the modulus of the ith component of $\cdot$, and b_a is the number of bits in the fractional part of the addition in the coefficient updating. In the steady state we can assume that $\sigma_n^2 + \sigma_e^2$ is a lower bound for $E[e^2(k)_Q]$ and equation (3.117) can be equivalently rewritten as

$$4\mu^2(\sigma_n^2+\sigma_e^2)\sigma_x^2 < 4\mu^2E[e^2(k)_Q]E[x_i^2(k)] < \frac{2^{-2b_a}}{4}w_{oi}^2 \tag{3.118}$$

The algorithm will not stop updating before the convergence is achieved if μ is chosen such that

$$\mu > \frac{2^{-b_a}}{4}\sqrt{\frac{w_{oi}^2}{(\sigma_n^2+\sigma_e^2)\sigma_x^2}} \tag{3.119}$$

In case μ does not satisfy the condition above, the MSE is determined by the quantization error.

3.6 EXAMPLES

In this section a number of examples are presented in order to illustrate the use of the LMS algorithm as well as to verify theoretical results presented in the previous sections.

3.6.1 Analytical Examples

Some analytical tools presented so far are employed to characterize two interesting types of adaptive filtering problems. The problems are also solved with the LMS algorithm.

Example 3.1

A Gaussian noise with unit variance colored by a filter with transfer function

$$H_{in}(z) = \frac{1}{z - 0.5}$$

is transmitted through a communication channel with model given by

$$H_c(z) = \frac{1}{z + 0.8}$$

and with the channel noise being Gaussian with variance $\sigma_n^2 = 0.1$.

Fig. 3.3 illustrates the experimental environment. Note that $x'(k)$ is generated by first applying a Gaussian noise with variance $\sigma_{in}^2 = 1$ to a filter with transfer function $H_{in}(z)$. The result then is applied to a communication channel with transfer function $H_c(z)$, and then a Gaussian channel noise with variance $\sigma_n^2 = 0.1$ is added. On the other hand, $d(k)$ is generated by applying the same Gaussian noise with variance $\sigma_{in}^2 = 1$ to the filter with transfer function $H_{in}(z)$, with the result delayed by L samples.

(a) Determine the best value for the delay L.

(b) Compute the Wiener solution.

(c) Choose an appropriate value for μ and plot the convergence path for the LMS algorithm on the MSE surface.

(d) Plot the learning curves of the MSE and the filter coefficients in a single run as well as for the average of 25 runs.

Solution:

(a) In order to determine L, we will examine the behavior of the cross-correlation between the adaptive filter input signal denoted by $x'(k)$ and the reference signal $d(k)$.

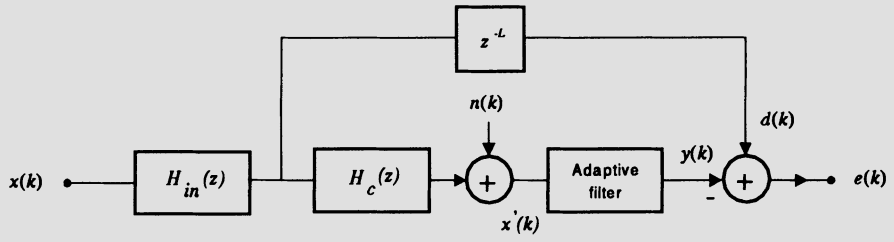


Figure 3.3 Channel equalization of Example 3.1.

The cross-correlation between $d(k)$ and $x'(k)$ is given by

$$\begin{aligned} p(i) &= E[d(k)x'(k-i)] \\ &= \frac{1}{2\pi j}\oint H_{in}(z)z^{-L}z^{i}H_{in}(z^{-1})H_c(z^{-1})\sigma_{in}^2\frac{dz}{z} \\ &= \frac{1}{2\pi j}\oint \frac{1}{z-0.5}z^{-L}z^{i}\frac{z}{1-0.5z}\frac{z}{1+0.8z}\sigma_{in}^2\frac{dz}{z} \end{aligned}$$

where the integration path is a counterclockwise closed contour corresponding to the unit circle.

The contour integral of the equation above can be solved through the Cauchy's residue theorem. For $L = 0$ and $L = 1$, the general solution is

$$p(0) = E[d(k)x'(k)] = \sigma_{in}^2[0.5^{-L+1}\frac{1}{0.75}\frac{1}{1.4}]$$

where in order to obtain $p(0)$, we computed the residue at the pole located at 0.5. The values of the cross-correlation for $L = 0$ and $L = 1$ are respectively

$$\begin{aligned} p(0) &= 0.47619 \\ p(0) &= 0.95238 \end{aligned}$$

For $L = 2$, we have that

$$p(0) = . \sigma_{in}^2[0.5^{-L+1}\frac{1}{0.75}\frac{1}{1.4} - 2] = -0.09522$$

where in this case we computed the residues at the poles located at 0.5 and at 0, respectively. For $L = 3$, we have

$$p(0) = \sigma_{in}^2[\frac{0.5^{-L+1}}{1.05} - 3.4] = -0.4095$$

From the analysis above, we see that the strongest correlation between $x'(k)$ and $d(k)$ occurs for $L = 1$. For this delay, the equalization is more effective. As a result, from the calculations above, we can obtain the elements of vector $\mathbf{p}$ as follows:

$$\mathbf{p} = \begin{bmatrix} p(0) \\ p(1) \end{bmatrix} = \begin{bmatrix} 0.9524 \\ 0.4762 \end{bmatrix}$$

Note that $p(1)$ for $L = 1$ is equal to $p(0)$ for $L = 0$.

The elements of the correlation matrix of the adaptive filter input signal are calculated as follows:

$$\begin{aligned} r(i) &= E[x'(k)x'(k-i)] \\ &= \frac{1}{2\pi j}\oint H_{in}(z)H_c(z)z^{-i}H_{in}(z^{-1})H_c(z^{-1})\sigma_{in}^2\frac{dz}{z} + \sigma_n^2\delta(i) \\ &= \frac{1}{2\pi j}\oint \frac{1}{z-0.5}\frac{1}{z+0.8}z^{-i}\frac{z}{1-0.5z}\frac{z}{1+0.8z}\sigma_{in}^2\frac{dz}{z} + \sigma_n^2\delta(i) \end{aligned}$$

where again the integration path is a counterclockwise closed contour corresponding to the unit circle, and $\delta(i)$ is the unitary impulse. Solving the contour integral equation, we obtain

$$\begin{aligned} r(0) &= E[x'^2(k)] \\ &= \sigma_{in}^2[\frac{1}{1.3}\frac{0.5}{0.75}\frac{1}{1.4} + \frac{-1}{1.3}\frac{-0.8}{1.4}\frac{1}{0.36}] + \sigma_n^2 = 1.6873 \end{aligned}$$

where the in order to obtain $r(0)$ we computed the residues at the poles located at 0.5 and -0.8, respectively. Similarly, we have that

$$\begin{aligned} r(1) &= E[x'(k)x'(k-1)] \\ &= \sigma_{in}^2[\frac{1}{1.3}\frac{1}{0.75}\frac{1}{1.4} + \frac{-1}{1.3}\frac{1}{1.4}\frac{1}{0.36}] = -0.7937 \end{aligned}$$

where again we computed the residues at the poles located at 0.5 and -0.8, respectively.

The correlation matrix of the adaptive filter input signal is given by

$$\mathbf{R} = \begin{bmatrix} 1.6873 & -0.7937 \\ -0.7937 & 1.6873 \end{bmatrix}$$

(b) The coefficients corresponding to the Wiener solution are given by

$$\begin{aligned}\mathbf{w}_o &= \mathbf{R}^{-1}\mathbf{p} \\ &= 0.45106 \begin{bmatrix} 1.6873 & 0.7937 \\ 0.7937 & 1.6873 \end{bmatrix} \begin{bmatrix} 0.9524 \\ 0.4762 \end{bmatrix} \\ &= \begin{bmatrix} 0.8953 \\ 0.7034 \end{bmatrix}\end{aligned}$$

where for calculating the inverse we utilized the result

$$\mathbf{R}^{-1} = \frac{1}{r_{11}r_{22} - r_{12}r_{21}} \begin{bmatrix} r_{22} & -r_{12} \\ -r_{21} & r_{11} \end{bmatrix}$$

where r_{ij} is the element of row i and column j of the matrix $\mathbf{R}$.

(c) The LMS algorithm was applied to minimize the MSE using a convergence factor $\mu = 1/40tr[\mathbf{R}]$, where $tr[\mathbf{R}] = 3.3746$. The value of μ was 0.0074. This small value of the convergence factor allows a smooth convergence path. The convergence path of the algorithm on the MSE surface is depicted in Fig. 3.4. As can be noted, the path followed by the LMS algorithm looks like a noisy steepest-descent path. It first approaches the main axis (eigenvector) corresponding to the smaller eigenvalue, and then follows toward the minimum in a direction increasingly aligned with this main axis.

(d) The learning curves of the MSE and the filter coefficients, in a single run are depicted in Fig. 3.5. The learning curves of the MSE and the filter coefficients, obtained by averaging the results of 25 runs, are depicted in Fig. 3.6. As can be noted, these curves are less noisy than in the single run case. □

The adaptive filtering problems discussed so far assumed that the signals taken from the environment were stochastic signals. Also, by assuming these signals were ergodic, we have shown that the adaptive filter is able to approach the Wiener solution by replacing the ensemble average by time averages. In conclusion, we can assume that the solution reached by the adaptive filter is based on time averages of the cross-correlations of the environment signals.

For example, if the environment signals are periodic deterministic signals, the optimal solution depends on the time average of the related cross-correlations computed over one period of the signals. Note that in this case, the solution obtained using an ensemble average would be time varying since we are dealing with a nonstationary problem. The following examples illustrate this issue.

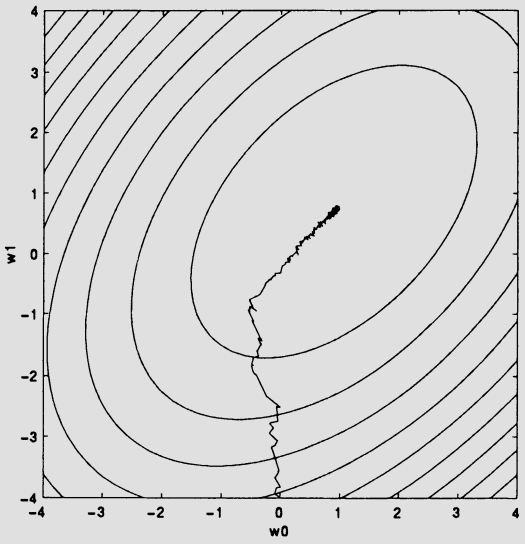


Figure 3.4 Convergence path on the MSE surface.

Example 3.2

Suppose in an adaptive filtering environment, the input signal consists of

$$x(k) = \cos(\omega_0 k)$$

The desired signal is given by

$$d(k) = \sin(\omega_0 k)$$

where $\omega_0 = \frac{2\pi}{M}$. In this case $M = 7$.

Compute the optimal solution for a first-order adaptive filter.

Solution:

In this example, the signals involved are deterministic and periodic. If the adaptive filter coefficients are fixed, the error is a periodic signal with period M. In this case, the objective function that will be minimized by the adaptive

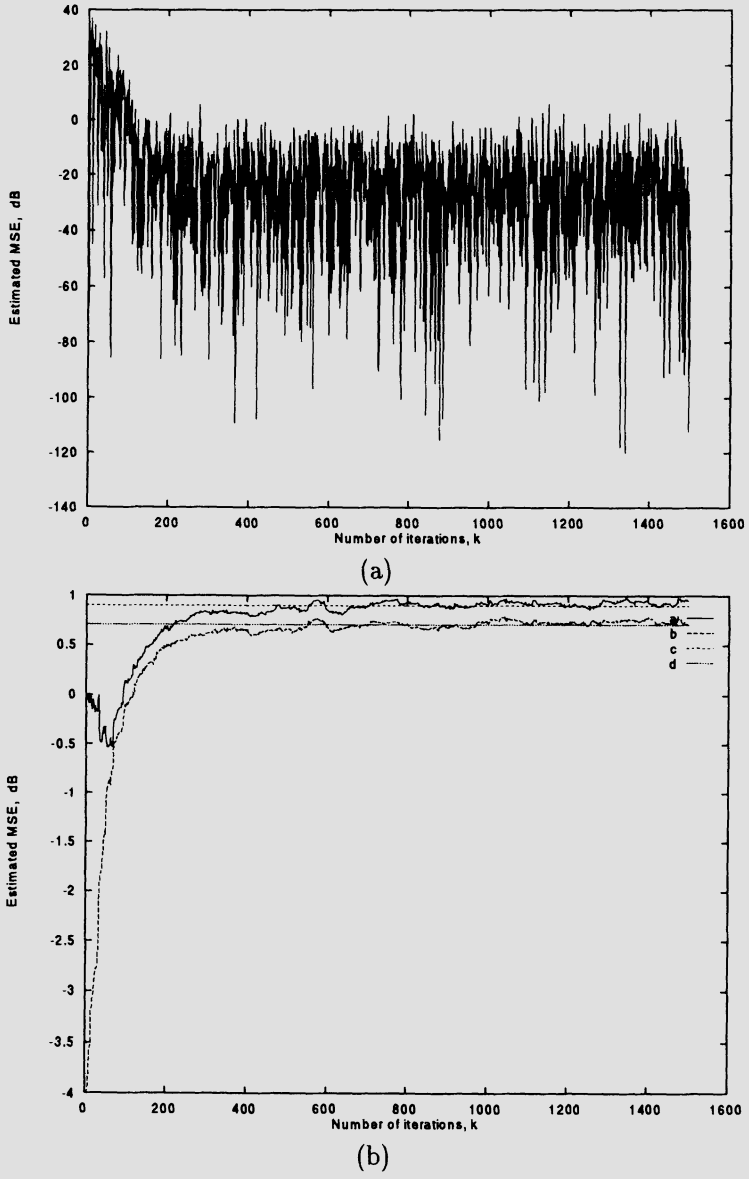


Figure 3.5 (a) Learning curve of the instantaneous squared error (b) Learning curves of the coefficients, a - first coefficient, b - second coefficient, c - optimal value for the first coefficient, d - optimal value of the second coefficient.

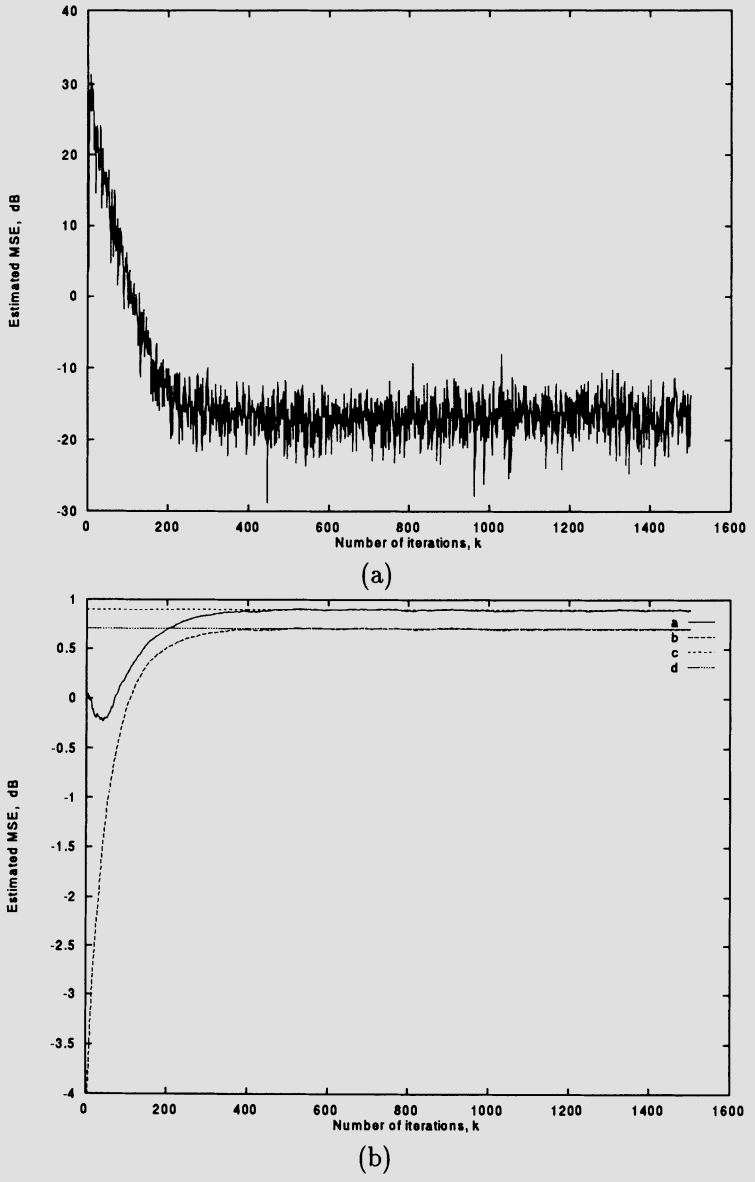


Figure 3.6 (a) Learning curve of the MSE (b) Learning curves of the coefficients. Average of 25 runs. a - first coefficient, b - second coefficient, c - optimal value of the first coefficient, d - optimal value of the second coefficient.

filter is the average value of the squared error defined by

$$\begin{aligned} \bar{E}[e^2(k)] &= \frac{1}{M}\sum_{m=0}^{M-1}[e^2(k-m)] \\ &= \bar{E}[d^2(k)] - 2\mathbf{w}^T\bar{\mathbf{p}} + \mathbf{w}^T\bar{\mathbf{R}}\mathbf{w} \end{aligned} \tag{3.120}$$

where

$$\bar{\mathbf{R}} = \begin{bmatrix} \bar{E}[\cos^2(\omega_0 k)] & \bar{E}[\cos(\omega_0 k)\cos(\omega_0(k-1)] \\ \bar{E}[\cos(\omega_0 k)\cos(\omega_0(k-1)] & \bar{E}[\cos^2(\omega_0 k)] \end{bmatrix}$$

$$\bar{\mathbf{p}} = [\bar{E}[\sin(\omega_0 k)\cos(\omega_0 k)] \quad \bar{E}[\sin(\omega_0 k)\cos(\omega_0 k-1)]]^T$$

The expression for the optimal coefficient vector can be easily derived.

$$\mathbf{w}_o = \bar{\mathbf{R}}^{-1}\bar{\mathbf{p}}$$

Now the above results are applied to the problem described. The elements of the vector $\bar{\mathbf{p}}$ are calculated as follows:

$$\begin{aligned} \bar{\mathbf{p}} &= \frac{1}{M}\sum_{m=0}^{M-1}\begin{bmatrix} d(k-m)x(k-m) \\ d(k-m)x(k-m-1) \end{bmatrix} \\ &= \frac{1}{M}\sum_{m=0}^{M-1}\begin{bmatrix} \sin(\omega_0(k-m))\cos(\omega_0(k-m)) \\ \sin(\omega_0(k-m))\cos(\omega_0(k-m-1)) \end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix} 0 \\ \sin(\omega_0) \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0.3909 \end{bmatrix} \end{aligned}$$

The elements of the correlation matrix of the adaptive filter input signal are calculated as follows:

$$\begin{aligned} \bar{r}(i) &= \bar{E}[x(k)x(k-i)] \\ &= \frac{1}{M}\sum_{m=0}^{M-1}[\cos(\omega_0(k-m))\cos(\omega_0(k-m-i))] \end{aligned}$$

where

$$\begin{aligned} \bar{r}(0) &= \bar{E}[\cos^2(\omega_0(k))] = 0.5 \\ \bar{r}(1) &= \bar{E}[\cos(\omega_0(k))\cos(\omega_0(k-1))] = 0.3117 \end{aligned}$$

The correlation matrix of the adaptive filter input signal is given by

$$\bar{\mathbf{R}} = \begin{bmatrix} 0.5 & 0.3117 \\ 0.3117 & 0.5 \end{bmatrix}$$

The coefficients corresponding to the optimal solution are given by

$$\begin{aligned} \bar{\mathbf{w}}_o &= \bar{\mathbf{R}}^{-1}\bar{\mathbf{p}} \\ &= \begin{bmatrix} -0.7972 \\ 1.2788 \end{bmatrix} \end{aligned}$$

□

Example 3.3

(a) Assume the input and desired signals are deterministic and periodic with period M. Study the LMS algorithm behavior.

(b) Choose an appropriate value for μ in the previous example and plot the convergence path for the LMS algorithm on the average error surface.

Solution:

(a) It is convenient at this point to recall the coefficient updating of the LMS algorithm

$$\begin{aligned} \mathbf{w}(k+1) &= \mathbf{w}(k) + 2\mu\mathbf{x}(k)e(k) \\ &= \mathbf{w}(k) + 2\mu\mathbf{x}(k)\left[d(k) - \mathbf{x}^T(k)\mathbf{w}(k)\right] \end{aligned}$$

This equation can be rewritten as

$$\mathbf{w}(k+1) = \left[\mathbf{I} - 2\mu\mathbf{x}(k)\mathbf{x}^T(k)\right]\mathbf{w}(k) + 2\mu d(k)\mathbf{x}(k) \qquad (3.121)$$

The solution of the equation (3.121), as a function of the initial values of the adaptive filter coefficients, is given by

$$\begin{aligned}\mathbf{w}(k+1) &= \prod_{i=0}^{k}\left[\mathbf{I}-2\mu\mathbf{x}(i)\mathbf{x}^T(i)\right]\mathbf{w}(0) \\ &\quad +\sum_{i=0}^{k}\left\{\prod_{j=i+1}^{k}\left[\mathbf{I}-2\mu\mathbf{x}(j)\mathbf{x}^T(j)\right]2\mu d(i)\mathbf{x}(i)\right\} \qquad (3.122)\end{aligned}$$

where we define that $\prod_{j=k+1}^{k}[\cdot]=1$ for the second product.

Assuming the value of the convergence factor μ is small enough to guarantee that the LMS algorithm will converge, the first term on the righthand side of the equation above will vanish as $k \to \infty$. The resulting expression for the coefficient vector is given by

$$\mathbf{w}(k+1) = \sum_{i=0}^{k}\left\{\prod_{j=i+1}^{k}\left[\mathbf{I}-2\mu\mathbf{x}(j)\mathbf{x}^T(j)\right]2\mu d(i)\mathbf{x}(i)\right\}$$

The analysis of the above solution is not straightforward. Following an alternative path based on averaging the results in a period M, we can reach conclusive results.

Let us define the average value of the adaptive filter parameters as follows:

$$\overline{\mathbf{w}(k+1)} = \frac{1}{M}\sum_{m=0}^{M-1}\mathbf{w}(k+1-m)$$

Similar definition can be applied to the remaining parameters of the algorithm.

Considering that the signals are deterministic and periodic, we can apply the average operation to equation (3.121). The resulting equation is

$$\begin{aligned}\overline{\mathbf{w}(k+1)} &= \frac{1}{M}\sum_{m=0}^{M-1}\left[\mathbf{I}-2\mu\mathbf{x}(k-m)\mathbf{x}^T(k-m)\right]\mathbf{w}(k-m) \\ &\quad + \frac{1}{M}\sum_{m=0}^{M-1}2\mu d(k-m)\mathbf{x}(k-m)\end{aligned}$$

$$= \overline{[\mathbf{I} - 2\mu\mathbf{x}(k)\mathbf{x}^T(k)]\,\mathbf{w}(k)} + 2\mu\overline{d(k)\mathbf{x}(k)} \tag{3.123}$$

For large k and small μ, it is expected that the parameters converge to the neighborhood of the optimal solution. In this case, we can consider that $\overline{\mathbf{w}(k+1)} \approx \overline{\mathbf{w}(k)}$ and that the following approximation is valid

$$\overline{\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}(k)} \approx \overline{\mathbf{x}(k)\mathbf{x}^T(k)}\ \ \overline{\mathbf{w}(k)}$$

since the parameters after convergence wander around the optimal solution. Using these approximations in (3.123), the average values of the parameters in the LMS algorithm for periodic signals are given by

$$\overline{\mathbf{w}(k)} \approx \overline{\mathbf{x}(k)\mathbf{x}^T(k)}^{-1}\overline{d(k)\mathbf{x}(k)} = \bar{\mathbf{R}}^{-1}\bar{\mathbf{p}}$$

(b) The LMS algorithm was applied to minimize the squared error of the problem described in Example 3.2 using a convergence factor $\mu = 1/100tr[\bar{\mathbf{R}}]$, where $tr[\bar{\mathbf{R}}] = 1$. The value of μ was 0.01. The convergence path of the algorithm on the MSE surface is depicted in Fig. 3.7. As can be verified, the parameters generated by the LMS algorithm approach the optimal solution.

□

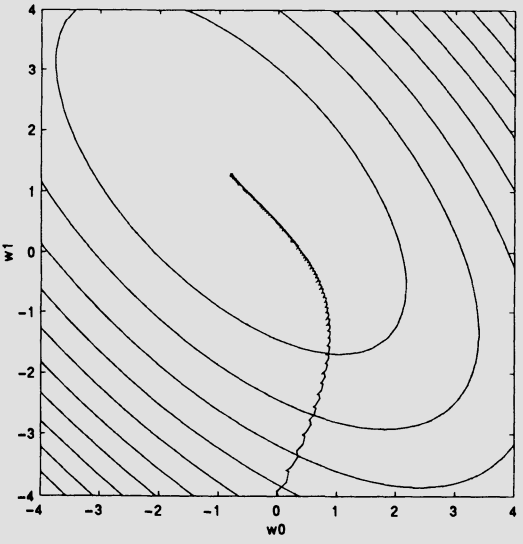


Figure 3.7 Convergence path on the MSE surface.

3.6.2 System Identification Simulations

In this subsection, a system identification problem is described and solved by using the LMS algorithm. In the following chapters the same problem will be solved using other algorithms presented in the book. For the FIR adaptive filters the following identification problem is posed:

Example 3.4

An adaptive filtering algorithm is used to identify a system with impulse response given below.

$$\mathbf{h} = [0.1\ 0.3\ 0.0\ -0.2\ -0.4\ -0.7\ -0.4\ -0.2]^T$$

Consider three cases for the input signal: colored noises with variance $\sigma_x^2 = 1$ and eigenvalue spread of their correlation matrix equal to 1.0, 20, and 80, respectively. The measurement noise is a Gaussian white noise uncorrelated with the input and with variance $\sigma_n^2 = 10^{-4}$. The adaptive filter has 8 coefficients.

(a) Run the algorithm and comment on the convergence behavior in each case.

(b) Measure the misadjustment in each example and compare with the theoretical results where appropriate.

(c) Considering that fixed-point arithmetic is used, run the algorithm for a set of experiments and calculate the expected values for $||\Delta\mathbf{w}(k)_Q||^2$ and $\xi(k)_Q$ for the following case:

Additional noise: white noise with variance	$\sigma_n^2 = 0.0015$
Coefficient wordlength:	$b_c = 16$ bits
Signal wordlength:	$b_d = 16$ bits
Input signal: Gaussian white noise with variance	$\sigma_x^2 = 1.0$

(d) Repeat the previous experiment for the following cases
$b_c = 12$ bits, $b_d = 12$ bits.
$b_c = 10$ bits, $b_d = 10$ bits.

(e) Suppose the unknown system is a time-varying system whose coefficients are first-order Markov processes with $\lambda_{\mathbf{W}} = 0.99$ and $\sigma^2_{\mathbf{W}} = 0.0015$. The initial time-varying-system multiplier coefficients are the ones described above. The input signal is a Gaussian white noise with variance $\sigma_x^2 = 1.0$, and the measurement noise is also a Gaussian noise independent of the input signal and of the elements of $\mathbf{n_W}(k)$, with variance $\sigma_n^2 = 0.01$. Simulate the experiment described, measure the total excess of MSE, and compare to the calculated results.

Solution:

The colored input signal was generated by applying a Gaussian noise, with variance σ_v^2, to a first-order filter with transfer function

$$H(z) = \frac{z}{z-a}$$

As can be shown from the example 2.1.d, the input-signal correlation matrix in this case is given by

$$\mathbf{R} = \frac{\sigma_v^2}{1-a^2}\begin{bmatrix} 1 & a & \cdots & a^7 \\ a & 1 & \cdots & a^6 \\ \vdots & \vdots & \ddots & \vdots \\ a^7 & a^6 & \cdots & 1 \end{bmatrix}$$

The proper choice of the value of a, in order to obtain the desired eigenvalue spread, is not a straightforward task. Some guidelines are now discussed. For example, if the adaptive filter was of first order, the matrix $\mathbf{R}$ would be two by two with eigenvalues

$$\lambda_{max} = \frac{\sigma_v^2}{1-a^2}(1+a)$$

and

$$\lambda_{min} = \frac{\sigma_v^2}{1-a^2}(1-a)$$

respectively. In this case, the choice of a is straightforward.

For a very large order adaptive filter, the eigenvalue spread approaches

$$\frac{\lambda_{max}}{\lambda_{min}} \approx \frac{|H_{max}(e^{j\omega})|^2}{|H_{min}(e^{j\omega})|^2} = \{\frac{1+a}{1-a}\}^2$$

where the details to reach this result can be found in Haykin [6].

Using the relations above as guidelines, we reached the correct values of a. These values are $a = 0.6894$ and $a = 0.8702$ for eigenvalue spreads of 20 and 80, respectively.

Since the variance of the input signal should be unitary, the variance of the Gaussian noise that produces $x(k)$ should be given by

$$\sigma_v^2 = 1 - a^2$$

For the LMS algorithm, we first calculated the upper bound for μ (μ_{max}) to guarantee the algorithm stability, and ran the algorithm for μ_{max}, $\mu_{max}/5$, and $\mu_{max}/10$.

In this example, the LMS algorithm did not converge for $\mu = \mu_{max} \approx 0.1$. The convergence behavior for $\mu_{max}/5$ and $\mu_{max}/10$ are illustrated through the learning curves depicted in Fig. 3.8, where in this case the eigenvalue spread was 1. Each curve was obtained by averaging the results of 200 independent runs. As can be noticed the reduction of the convergence factor leads to a reduction in the convergence speed. Also note that for $\mu = 0.02$ the estimated MSE was plotted only for the first 400 iterations, enough to display the convergence behavior. In all examples the tap coefficients were initialized with zero. Fig. 3.9 illustrates the learning curves for the various eigenvalue spreads, where in each case the convergence factor was $\mu_{max}/5$. As expected the convergence rate is reduced for a high eigenvalue spread.

The misadjustment was measured and compared with the results obtained from the following relation

$$M = \frac{\mu(N+1)\sigma_x^2}{1 - \mu(N+1)\sigma_x^2}$$

Also, for the present problem we calculated the time constants τ_{wi} and τ_{ei}, and the expected number of iterations to achieve convergence using the relations

$$\tau_{wi} \approx \frac{1}{2\mu\lambda_i}$$

$$\tau_{ei} \approx \frac{1}{4\mu\lambda_i}$$

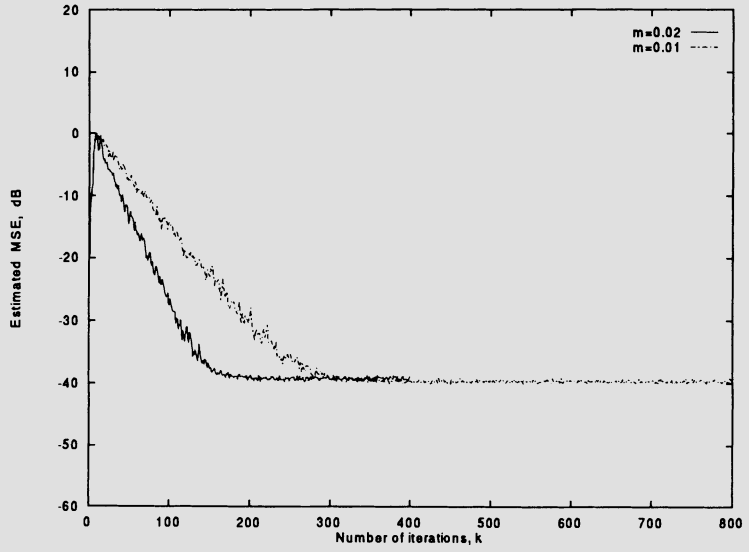


Figure 3.8 Learning curves for the LMS algorithm with convergence factors $\mu_{max}/5$ and $\mu_{max}/10$.

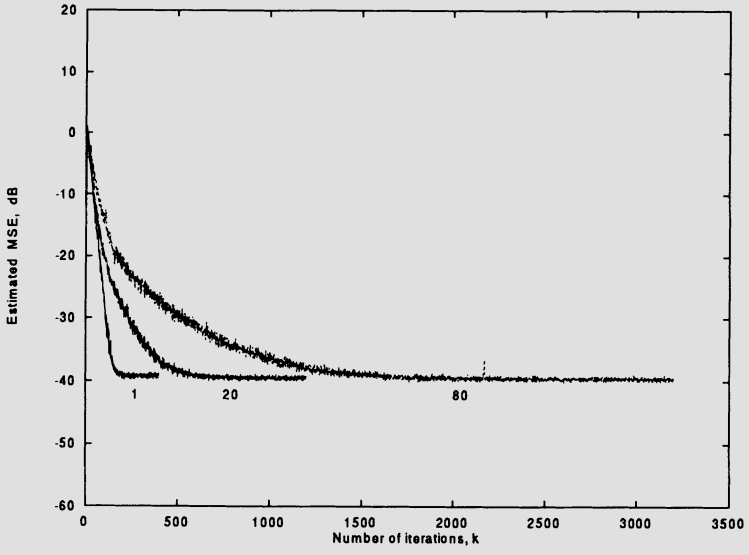


Figure 3.9 Learning curves for the LMS algorithm for eigenvalue spreads: 1, 20, and 80.

$$k \approx \tau_{e_{max}} \ln(100)$$

Table 3.1 illustrates the obtained results. As can be noted the analytical results agree with the experimental results, especially those related to the misadjustment. The analytical results related to the convergence time are optimistic as compared with the measured results. This discrepancy are mainly due to the approximations in the analysis.

Table 3.1 Evaluation of the LMS Algorithm

		Misadjustment		$\tau_{e_{max}}$	$\tau_{w_{max}}$	Iterations
μ	$\frac{\lambda_{max}}{\lambda_{min}}$	Experiment	Theory			
0.02	1	0.2027	0.1905	12.5	25	15
0.01280	20	0.1298	0.1141	102.5	205	473
0.01024	80	0.1045	0.0892	338.9	677.5	1561
0.01	1	0.0881	0.0870	25	50	29
0.006401	20	0.0581	0.0540	205	410	944
0.005119	80	0.0495	0.0427	677.5	1355	3121

The LMS algorithm was implemented employing fixed-point arithmetic using 16, 12, and 10 bits for data and coefficient wordlengths. The chosen value of μ was 0.01. The learning curves for the MSE are depicted in Fig. 3.10. Fig. 3.11 depicts the evolution of $||\Delta\mathbf{w}(k)_Q||^2$ with the number of iterations. The experimental results show that the algorithm still works for such limited precision. In Table 3.2, we present a summary of the results obtained from simulation experiments and a comparison with the results predicted by the theory. The experimental results were obtained by averaging the results of 200 independent runs. The relations employed to calculate the theoretical results shown in Table 3.2 are repeated here for convenience:

$$E[||\Delta\mathbf{w}(k)_Q||^2] = \frac{\mu(\sigma_n^2 + \sigma_e^2)(N+1)}{1-\mu(N+1)\sigma_x^2} + \frac{(N+1)\sigma_{\mathbf{w}}^2}{4\mu\sigma_x^2(1-\mu(N+1)\sigma_x^2)}$$

$$\xi(k)_Q = \frac{\sigma_e^2 + \sigma_n^2}{1-\mu(N+1)\sigma_x^2} + \frac{(N+1)\sigma_{\mathbf{w}}^2}{4\mu(1-\mu(N+1)\sigma_x^2)}$$

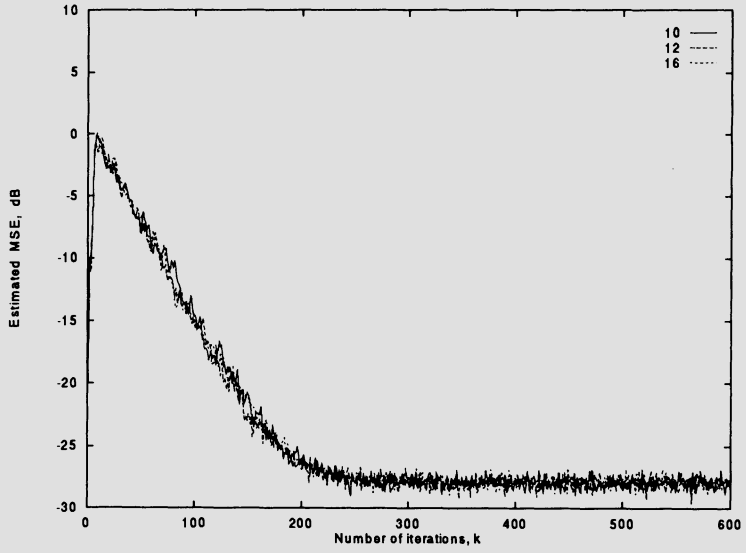


Figure 3.10 Learning curves for the LMS algorithm implemented with fixed-point arithmetic and with $\mu = 0.01$.

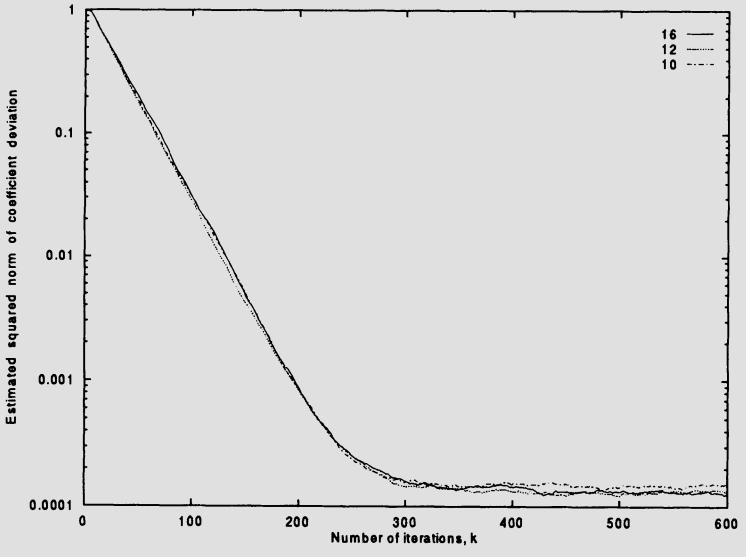


Figure 3.11 Estimate of $||\Delta \mathbf{w}(k)_Q||^2$ for the LMS algorithm implemented with fixed-point arithmetic and with $\mu = 0.01$.

Table 3.2 Results of the Finite Precision Implementation of the LMS Algorithm

No. of bits	$\xi(k)_Q$		$E[\|\|\Delta\mathbf{w}(k)_Q\|\|^2]$	
	Experiment	Theory	Experiment	Theory
16	$1.629\ 10^{-3}$	$1.630\ 10^{-3}$	$1.316\ 10^{-4}$	$1.304\ 10^{-4}$
12	$1.632\ 10^{-3}$	$1.631\ 10^{-3}$	$1.309\ 10^{-4}$	$1.315\ 10^{-4}$
10	$1.663\ 10^{-3}$	$1.648\ 10^{-3}$	$1.465\ 10^{-4}$	$1.477\ 10^{-4}$

The results of Table 3.2 confirm that the finite-precision implementation analysis presented is accurate.

The performance of the LMS algorithm was also tested in the nonstationary environment described above. The excess of MSE was measured and depicted in Fig. 3.12. For this example μ_{opt} was found to be greater than μ_{max}. The value of μ used in the example was 0.05. The excess of MSE in steady state predicted by the relation

$$\xi_{total} \approx \frac{\mu\sigma_n^2 \mathrm{tr}[\mathbf{R}]}{1-\mu\mathrm{tr}[\mathbf{R}]} + \frac{\sigma_{\mathbf{w}}^2}{4\mu}\sum_{i=0}^{N}\frac{1}{1-\mu\lambda_i}$$

was 0.124, whereas the measured excess of MSE in steady state was 0.118. Once more the results obtained from the analysis are accurate.

□

3.6.3 Channel Equalization Simulations

In this subsection an equalization example is described. This example will be used as pattern for comparison of several algorithms presented in this book.

Example 3.5

Perform the equalization of a channel with the following impulse response

$$h(k) = 0.1\ (0.5^k)$$

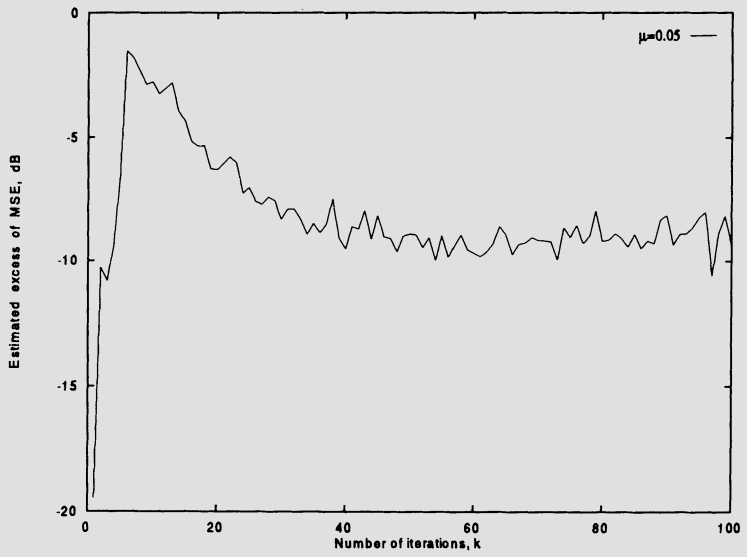


Figure 3.12 The excess of MSE of the LMS algorithm in nonstationary environment, $\mu = 0.05$.

for $k = 0, 1, \ldots 8$. Use a known training signal that consists of independent binary samples (-1,1). An additional Gaussian white noise with variance $10^{-2.5}$ is present at the channel output.

(a) Find the impulse response of an equalizer with 50 coefficients.

(b) Convolve the equalizer impulse response at a given iteration after convergence, with the channel impulse response and comment the result.

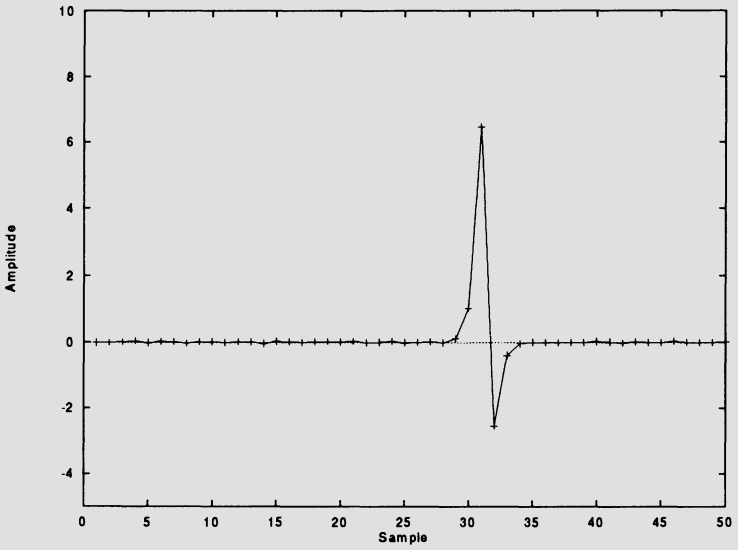


Figure 3.13 Equalizer impulse response; LMS algorithm.

Solution:

We have applied the LMS algorithm to solve the equalization problem. We used $\mu_{max}/5$ for the value of the convergence factor. In order to obtain μ_{max}, the values of $\lambda_{max} = 0.04275$ and $\sigma_x^2 = 0.01650$ were measured and applied in equation (3.30). The resulting value of μ was 0.2197.

The appropriate value of L was found to be $round(\frac{9+50}{2}) = 30$. The impulse response of the resulting equalizer is shown in Fig. 3.13. By convolving this response with the channel impulse response, we obtain the result depicted in Fig. 3.14 that clearly approximates an impulse. The measured MSE was 0.3492.

□

3.7 CONCLUDING REMARKS

In this chapter, we studied the LMS adaptive algorithm that is certainly the most popular among the adaptive filtering algorithms. The attractiveness of the LMS algorithm is due to its simplicity and accessible analysis under ideal-

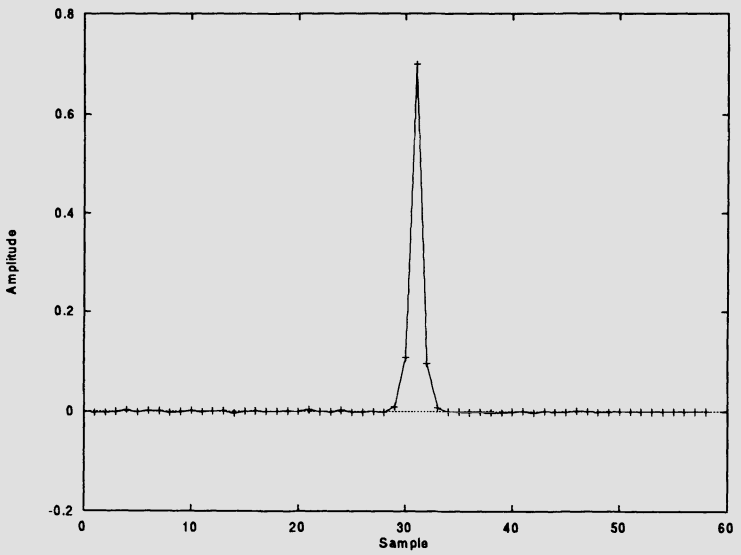


Figure 3.14 Convolution result; LMS algorithm.

ized conditions. As demonstrated in the present chapter, the noisy estimate of the gradient that is used in the LMS algorithm is the main source of loss in performance for stationary environments. Further discussions on the convergence behavior of the LMS algorithm have been reported in the open literature, see for example [22]-[24].

For nonstationary environments we showed how the algorithm behaves assuming the optimal parameter can be modeled as first-order Markov process. The analysis allowed us to determine the conditions for adequate tracking and acceptable excess of MSE. Further analysis can be found in [25].

The quantization effects on the behavior of the LMS algorithm was also presented. The algorithm is fairly robust against quantization errors, and this is for sure one of the reasons for its choice in a number of practical applications [26]-[29].

A number of simulation examples with the LMS algorithm was presented in this chapter. The simulations included examples in system identification and equalization. Also a number of theoretical results derived in the present chapter were verified, such as the excess of MSE in stationary and nonstationary environments, the finite-precision analysis etc.

Appendix

In this appendix we derive the expressions for the quantization errors generated in the implementation of the LMS algorithm using floating-point arithmetic.

The error in the output error computation is given by

$$
\begin{aligned}
n_e(k) \approx & -n_a(k)[d(k) - \mathbf{x}^T(k)\mathbf{w}(k)_Q] \\
& + \mathbf{x}^T(k) \begin{bmatrix} n_{p_0}(k) & 0 & 0 & \cdots & 0 \\ 0 & n_{p_1}(k) & \cdots & \cdots & 0 \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & & & n_{p_N}(k) \end{bmatrix} \mathbf{w}(k)_Q \\
& - [n_{a_1}(k)\ n_{a_2}(k)\ \cdots\ n_{a_N}(k)] \begin{bmatrix} \sum_{i=0}^{1} x(k-i)w_i(k)_Q \\ \sum_{i=0}^{2} x(k-i)w_i(k)_Q \\ \vdots \\ \sum_{i=0}^{N} x(k-i)w_i(k)_Q \end{bmatrix} \\
= & -n_a(k)e(k)_Q - \mathbf{x}^T(k)\mathbf{N}_p(k)\mathbf{w}(k-1)_Q - \mathbf{n}_a(k)\mathbf{s}_i(k)
\end{aligned}
$$

where $n_{p_i}(k)$ accounts for the noise generated in the products $x(k-i)w_i(k)_Q$ and $n_{a_i}(k)$ accounts for the noise generated in the additions of the product $\mathbf{x}^T(k)\mathbf{w}(k)$. Note that the error terms of second- and higher-order have been neglected.

Using similar assumptions one can show that

$$
\begin{aligned}
\mathbf{n}_{\mathbf{w}}(k) = & -2\mu n'_p(k)e(k)_Q\mathbf{x}(k) - 2\mu\mathbf{N}''_p(k)e(k)_Q\mathbf{x}(k) \\
& -\mathbf{N}'_a(k)[\mathbf{w}(k)_Q + 2\mu e(k)_Q\mathbf{x}(k)]
\end{aligned} \tag{3.124}
$$

where

$$
N''_p(k) = \begin{bmatrix} n''_{p_0}(k) & 0 & \cdots & 0 \\ 0 & n''_{p_1}(k) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & n''_{p_N}(k) \end{bmatrix}
$$

$$N'_a(k) = \begin{bmatrix} n'_{a_0}(k) & 0 & \cdots & 0 \\ 0 & n'_{a_1}(k) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & n'_{a_N}(k) \end{bmatrix}$$

and $n'_p(k)$ accounts for the quantization of the product 2μ by $e(k)_Q$, considering that 2μ is already available. Matrix $\mathbf{N}''_p(k)$ models the quantization in the product of $2\mu e(k)_Q$ by $\mathbf{x}(k)$, while $\mathbf{N}'_a(k)$ models the error in the vector addition used to generate $\mathbf{w}(k+1)_Q$.

If we substitute the expression for $e(k)_Q$ in equation (3.79) in $\mathbf{n_w}(k)$ given in (3.124), and use the result in equation (3.82), it can be easily shown that

$$\begin{aligned} \Delta\mathbf{w}(k+1)_Q &= [\mathbf{I} - 2\mu\mathbf{x}(k)\mathbf{x}^T(k)]\Delta\mathbf{w}(k)_Q \\ &\quad +2\mu\mathbf{x}(k)(n(k) - n_e(k)) - \mathbf{n_w}(k) \\ &\approx [\mathbf{I} - 2\mu\mathbf{x}(k)\mathbf{x}^T(k) + 2\mu n'_p(k)\mathbf{x}(k)\mathbf{x}^T(k) \\ &\quad +2\mu\mathbf{N}''_p(k)\mathbf{x}(k)\mathbf{x}^T(k) + 2\mu\mathbf{N}'_a(k)\mathbf{x}(k)\mathbf{x}^T(k) \\ &\quad +\mathbf{N}'_a(k)]\Delta\mathbf{w}(k)_Q + \mathbf{N}'_a(k)\mathbf{w}_o + 2\mu\mathbf{x}(k)(n(k) - n_e(k)) \end{aligned}$$

where the terms corresponding to products of quantization errors were considered small enough to be neglected.

Finally, the variance of the error noise can be derived as follows:

$$\begin{aligned} \sigma_e^2 &= \sigma_{n_a}^2\xi(k)_Q + \sigma_{n_p}^2\sum_{i=0}^{N}\mathbf{R}_{i,i}\text{cov}[\mathbf{w}(k+1)_Q]_{i,i} \\ &\quad +\sigma_{n_a}^2\left\{E[(\sum_{i=0}^{1}x(k-i)w_i(k)_Q)^2] + E[(\sum_{i=0}^{2}x(k-i)w_i(k)_Q)^2]\right. \\ &\quad \left. + \cdots + E[(\sum_{i=0}^{N}x(k-i)w_i(k)_Q)^2]\right\} \end{aligned}$$

where $\sigma'^2_{n_{a_i}}$ was considered equal to $\sigma^2_{n_a}$, and $[\cdot]_{i,i}$ means diagonal elements of $[\cdot]$. The second term can be further simplified as follows:

$$\begin{aligned} \text{tr}\{\mathbf{R}\text{cov}[\mathbf{w}(k+1)_Q]\} &\approx \sum_{i=0}^{N}\mathbf{R}_{i,i}w_{oi}^2 + \mathbf{R}_{i,i}\text{cov}[\Delta\mathbf{w}(k+1)]_{i,i} \\ &\quad +\text{first- and higher-order terms} \cdots \end{aligned}$$

Since this term is multiplied by $\sigma_{n_p}^2$, any first- and higher-order terms can be neglected. The first term of σ_e^2 is also small in the steady state. The last term can be rewritten as

$$\begin{aligned}\sigma_{n_a}^2 & \left\{ E[(\sum_{i=0}^{1} x(k-i)w_{oi})^2] + E[(\sum_{i=0}^{2} x(k-i)w_{oi})^2] + \cdots \right. \\ & \left. + E[(\sum_{i=0}^{N} x(k-i)w_{oi})^2] \right\} \\ = \quad & \sigma_{n_a}^2 \left\{ \sum_{j=1}^{N} \sum_{i=0}^{j} \mathbf{R}_{i,i} \text{cov}[\Delta \mathbf{w}(k+1)]_{i,i} \right\}\end{aligned}$$

where terms of order higher than one were neglected, $x(k)$ was considered uncorrelated to $\Delta \mathbf{w}(k+1)$, and cov$[\Delta \mathbf{w}(k+1)]$ was considered a diagonal matrix. Actually, if $x(k)$ is considered a zero-mean Gaussian noise, from equation (3.23) it can be shown that

$$\begin{aligned}\text{cov}[\Delta \mathbf{w}(k)] & \approx \mu \sigma_n^2 \mathbf{I} + \frac{\mu^2 (N+1) \sigma_x^2 \sigma_n^2 \mathbf{I}}{1 - \mu(N+1)\sigma_x^2} \\ & = \frac{\mu \sigma_n^2 \mathbf{I}}{1 - \mu(N+1)\sigma_x^2}\end{aligned}$$

Since this term will be multiplied by $\sigma_{n_a}^2$ and $\sigma_{n_p}^2$, it can also be disregarded. In conclusion,

$$\sigma_e^2 \approx \sigma_{n_a}^2 \{ E[\sum_{j=1}^{N} (\sum_{i=0}^{j} x(k-i)w_{oi})^2] \} + \sigma_{n_p}^2 \sum_{i=0}^{N} \mathbf{R}_{i,i} w_{oi}^2$$

This equation can be further simplified when $x(k)$ is as described above and $\sigma_{n_a}^2 = \sigma_{n_p}^2 = \sigma_d^2$

$$\begin{aligned}\sigma_e^2 & \approx \sigma_d^2 [\sum_{i=1}^{N} (N-i+2) \mathbf{R}_{i,i} w_{oi}^2 - \mathbf{R}_{1,1} w_{o1}^2] \\ & = \sigma_d^2 \sigma_x^2 [\sum_{i=1}^{N} (N-i+2) w_{oi}^2 - w_{o1}^2]\end{aligned}$$

References

1. B. Widrow and M. E. Hoff, "Adaptive switching circuits," *WESCOM Conv. Rec.*, pt. 4, pp. 96-140, 1960.

2. B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson, Jr., "Stationary and nonstationary learning characteristics of the LMS adaptive filters," *Proceedings of the IEEE*, vol. 64, pp. 1151-1162, Aug. 1976.

3. G. Ungerboeck, "Theory on the speed of convergence in adaptive equalizers for digital communication," *IBM Journal on Research and Development*, vol. 16, pp. 546-555, Nov. 1972.

4. J. E. Mazo, "On the independence theory of equalizer convergence," *The Bell System Technical Journal*, vol. 58, pp. 963-993, May 1979.

5. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1985.

6. S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, NJ, 2nd edition, 1991.

7. M. G. Bellanger, *Adaptive Digital Filters and Signal Analysis*, Marcel Dekker Inc., NY, 1987.

8. D. C. Farden, "Tracking properties of adaptive signal processing algorithms," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-29, pp. 439-446, June 1981.

9. B. Widrow and E. Walach, "On the statistical efficiency of the LMS algorithm with nonstationary inputs," *IEEE Trans. on Information Theory*, vol. IT-30, pp. 211-221, March 1984.

10. O. Macchi, "Optimization of adaptive identification for time varying filters," *IEEE Trans. on Automatic Control*, vol. AC-31, pp. 283-287, March 1986.

11. A. Benveniste, "Design of adaptive algorithms for the tracking of time varying systems," *Int. J. Adaptive Control and Signal Processing*, vol. 1, pp. 3-29, Jan. 1987.

12. W. A. Gardner, "Nonstationary learning characteristics of the LMS algorithm," *IEEE Trans. on Circuits and Systems*, vol. CAS-34, pp. 1199-1207, Oct. 1987.

13. A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw Hill, New York, NY, 3rd edition, 1991.

14. V. Solo, "The limiting behavior of LMS," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol-37, pp. 1909-1922, Dec. 1989.

15. N. J. Bershad and O. M. Macchi, "Adaptive recovery of a chirped sinusoid in noise, Part 2: Performance of the LMS algorithm," *IEEE Trans. on Signal Processing*, vol. 39, pp. 595-602, March 1991.

16. M. Andrews and R. Fitch, "Finite wordlength arithmetic computational error effects on the LMS adaptive weights," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, pp. 628-631, May 1977.

17. C. Caraiscos and B. Liu, "A roundoff error analysis of the LMS adaptive algorithm," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-32, pp. 34-41, Feb. 1984.

18. S. T. Alexander, "Transient weight misadjustment properties for the finite precision LMS algorithm," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-35, pp. 1250-1258, Sept. 1987.

19. A. V. Oppenheim and R. W. Schaffer, *Discrete-Time Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1989.

20. A. Antoniou, *Digital Filters: Analysis, Design, and Applications*, McGraw Hill, New York, NY, 2nd edition, 1992.

21. A. B. Spirad and D. L. Snyder, "Quantization errors in floating-point arithmetic," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-26, pp. 456-464, Oct. 1983.

22. A. Feuer and E. Weinstein, "Convergence analysis of LMS filters with uncorrelated Gaussian data," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-33, pp. 222-230, Jan. 1985.

23. D. T. Slock, "On the convergence behavior of the LMS and normalized LMS algorithms," *IEEE Trans. on Signal Processing*, vol-40, pp. 2811-2825, Sept. 1993.

24. W. A. Sethares, D. A. Lawrence, C. R. Johnson, Jr., and R. R. Bitmead, "Parameter drift in LMS adaptive filters," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-34, pp. 868-878, Aug. 1986.

25. V. Solo, "The error variance of LMS with time varying weights," *IEEE Trans. on Signal Processing*, vol-40, pp. 803-813, April 1992.

26. S. U. Qureshi, "Adaptive Equalization," *Proceedings of the IEEE*, vol-73, pp. 1349-1387, Sept. 1985.

27. M. L. Honig, "Echo cancellation of voiceband data signals using recursive least squares and stochastic gradient algorithms," *IEEE Trans. on Communications*, vol. COM-33, pp. 65-73, Jan. 1985.

28. S. Subramanian, D. J. Shpak, P. S. R. Diniz, and A. Antoniou, "The performance of adaptive filtering algorithms in a simulated HDSL environment," *Proc. IEEE Canadian Conf. Electrical and Computer Engineering*, Toronto, pp. TA 2.19.1-TA 2.19.5, Sept. 1992.

29. C. S. Modlin and J. M. Cioffi, "A fast decision feedback LMS algorithm using multiple step sizes," *Proc. IEEE SUPERCOMM Inter. Conf. on Communications*, New Orleans, pp. 1201-1205, May 1994.

Problems

1. The LMS algorithm is used to predict the signal $x(k) = \cos(\pi k/3)$ using a second-order FIR filter with the first tap fixed at 1, by minimizing the MSE of $y(k)$. Calculate an appropriate μ, the output signal, and the filter coefficients for the first 10 iterations. Start with $\mathbf{w}^T(0) = [1\ 0\ 0]$.

2. The signal

$$x(k) = -0.85x(k-1) + n(k)$$

is applied to a first-order predictor, where $n(k)$ is a Gaussian white noise with variance $\sigma_n^2 = 0.3$.

(a) Compute the Wiener solution.

(b) Choose an appropriate value for μ and plot the convergence path for the LMS algorithm on the MSE error surface.

(c) Plot the learning curves for the MSE and the filter coefficients in a single run as well as for the average of 25 runs.

3. Use the LMS algorithm to identify a system with the transfer function given below. The input signal is a uniformly distributed white noise with variance $\sigma_x^2 = 1$, and the measurement noise is a Gaussian white noise uncorrelated with the input with variance $\sigma_n^2 = 10^{-3}$. The adaptive filter has 12 coefficients.

$$H(z) = \frac{1 - z^{-12}}{1 - z^{-1}}$$

(a) Calculate the upper bound for μ (μ_{max}) to guarantee the algorithm stability.
(b) Run the algorithm for $\mu_{max}/2$, $\mu_{max}/10$, and $\mu_{max}/50$. Comment on the convergence behavior in each case.
(c) Measure the misadjustment in each example and compare with the results obtained by the equation (3.47).
(d) Plot the obtained FIR filter frequency response in any iteration after convergence is achieved and compare with the unknown system.

4. Repeat the previous problem using an adaptive filter with 8 coefficients and interpret the results.

5. Repeat problem 2 in case the input signal is a uniformly distributed white noise with variance $\sigma_x^2 = 0.5$ filtered by an all-pole filter given by

$$H(z) = \frac{z}{z - 0.9}$$

6. Perform the equalization of a channel with the following impulse response

$$h(k) = ku(k) - (2k - 9)u(k - 5) + (k - 9)u(k - 10)$$

Using a known training signal that consists of a binary (-1,1) random signal, generated by applying a white noise to a hard limiter (the output is 1 for positive input samples and -1 for negative). An additional Gaussian white noise with variance 10^{-2} is present at the channel output.
(a) Apply the LMS with an appropriate μ and find the impulse response of an equalizer with 100 coefficients.
(b) Convolve one of the equalizer's impulse response after convergence with the channel impulse response and comment on the result.

7. Under the assumption that the elements of $\mathbf{x}(k)$ are jointly Gaussian vector, show that equation (3.24) is valid.

8. In a system identification problem the input signal is generated by an autoregressive process given by

$$x(k) = -1.2x(k - 1) - 0.81x(k - 2) + n_x(k)$$

where $n_x(k)$ is a zero-mean Gaussian white noise with variance such that $\sigma_x^2 = 1$. The unknown system is described by

$$H(z) = 1 + 0.9z^{-1} + 0.1z^{-2} + 0.2z^{-3}$$

The adaptive filter is also a third-order FIR filter, and the additional noise is a zero-mean Gaussian noise with variance $\sigma_n^2 = 0.04$. Using the LMS

algorithm:
(a) Choose an appropriate μ, run an ensemble of 20 experiments, and plot the average learning curve.
(b) Plot the curve obtained using the equations (3.38), (3.42), and (3.43), and compare the results.
(c) Compare the measured and theoretical value for the misadjustment.
(d) Calculate the time constants τ_{wi} and τ_{ei}, and the expected number of iterations to achieve convergence.

9. In a nonstationary environment the optimal coefficient vector is described by

$$\mathbf{w}_o(k) = -\lambda_1 \mathbf{w}_o(k-1) - \lambda_2 \mathbf{w}_o(k-2) + \mathbf{n}_{\mathbf{W}}(k)$$

where $\mathbf{n}_{\mathbf{W}}(k)$ is a vector which has the elements that are zero-mean Gaussian processes with variance $\sigma^2_{\mathbf{W}}$. Calculate the elements of the lag-error vector.

10. Repeat the previous problem for

$$\mathbf{w}_o(k) = \lambda_w \mathbf{w}_o(k-1) + (1-\lambda_w)\mathbf{n}_{\mathbf{W}}(k)$$

11. The LMS algorithm was applied to identify a 7th-order time-varying unknown system whose coefficients are a first-order Markov process with $\lambda_{\mathbf{W}} = 0.999$ and $\sigma^2_{\mathbf{W}} = 0.001$. The initial time-varying-system multiplier coefficients are

$$\mathbf{w}_o^T =$$

$$[0.03490 \ -0.011 \ -0.06864 \ 0.22391 \ 0.55686 \ 0.35798 \ -0.0239 \ -0.07594]$$

The input signal is a Gaussian white noise with variance $\sigma_x^2 = 0.7$, and the measurement noise is also a Gaussian noise independent of the input signal and of the elements of $\mathbf{n}_{\mathbf{W}}(k)$, with variance $\sigma_n^2 = 0.01$.
(a) For $\mu = 0.05$, compute the excess of MSE.
(b) Repeat (a) for $\mu = 0.01$.
(c) Compute μ_{opt} and comment if it can be used.

12. Simulate the experiment described in problem 11, measure the excess of MSE, and compare to the calculated results.

13. Reduce the value of $\lambda_{\mathbf{W}}$ to 0.97 in the problem 11, simulate, and comment on the results.

14. Suppose a 15th-order filter FIR digital filter with multiplier coefficients given below was identified through an adaptive FIR filter of the same order using the LMS algorithm.

(a) Considering that fixed-point arithmetic was used, compute the expected values for $||\Delta \mathbf{w}(k)_Q||$ and $\xi(k)_Q$, and the probable number of iterations before the algorithm stops updating, for the following case:

Additional noise: white noise with variance	$\sigma_n^2 = 0.0015$
Coefficient wordlength:	$b_c = 16$ bits
Signal wordlength:	$b_d = 16$ bits
Input signal: Gaussian white noise with variance	$\sigma_x^2 = 0.7$
	$\mu = 0.01$

Hint: Utilize the formulas for the time constant in the LMS algorithm and equation (3.99).
(b) Simulate the experiment and plot the learning curves for the finite- and infinite-precision implementations.
(c) Compare the simulated results with those obtained through the closed form formulas.

$\mathbf{w}_o^T$ = [0.0219360 0.0015786 − 0.0602449 − 0.0118907 0.1375379 0.0574545 − 0.3216703 − 0.5287203 − 0.2957797 0.0002043 0.290670 −0.0353349 − 0.068210 0.0026067 0.0010333 − 0.0143593]

15. Repeat the problem above for the following cases
(a) $\sigma_n^2 = 0.01$, $b_c = 12$ bits, $b_d = 12$ bits, $\sigma_x^2 = 0.7$, $\mu = 2.0\ 10^{-3}$.
(b) $\sigma_n^2 = 0.1$, $b_c = 10$ bits, $b_d = 10$ bits, $\sigma_x^2 = 0.8$, $\mu = 1.0\ 10^{-4}$.
(c) $\sigma_n^2 = 0.05$, $b_c = 14$ bits, $b_d = 14$ bits, $\sigma_x^2 = 0.8$, $\mu = 2.0\ 10^{-3}$.

16. Find the optimal value of μ (μ_o) that minimizes the excess of MSE given in equation (3.104), and compute the expected values of $||\Delta \mathbf{w}(k)_Q||$ and $\xi(k)_Q$ for the examples described problem 15.

17. Repeat problem 14 in the case the input signal is a first-order Markov process with $\lambda_{\mathbf{x}} = 0.95$.

18. Include the input-signal quantizer noise in the analysis of the cov$[\Delta \mathbf{w}(k+1)_Q]$ and show that its effect in the MSE is given by equation (3.105).

19. A digital channel model can be represented by the following impulse response:

$$[-0.001\ -0.002\ 0.002\ 0.2\ 0.6\ 0.76\ 0.9\ 0.78\ 0.67\ 0.58$$
$$0.45\ 0.3\ 0.2\ 0.12\ 0.06\ 0\ -0.2\ -1\ -2\ -1\ 0\ 0.1]$$

The channel is corrupted by a Gaussian noise with power spectrum given by

$$|S(e^{j\omega})|^2 = \kappa' \omega^{3/2}$$

where $\kappa' = 10^{-1.5}$. The training signal consists of independent binary samples (-1,1).

Design an FIR equalizer for this problem and use the LMS algorithm. Use a filter of order 50 and plot the learning curves.

20. For the previous problem, using the maximum of 51 adaptive filter coefficients, implement a DFE equalizer and compare the results with those obtained with the FIR filter. Again use the LMS algorithm.

21. Implement with fixed-point arithmetic the DFE equalizer of problem 20, using the LMS algorithm with 12 bits of wordlength for data and coefficients.

4

LMS-BASED ALGORITHMS

4.1 INTRODUCTION

There are a number of algorithms for adaptive filters which are derived from the conventional LMS algorithm discussed in the previous chapter. The objective of the alternative LMS-based algorithms is either to reduce computational complexity or convergence time. In this chapter, four LMS-based algorithms are presented and analyzed, namely, the quantized-error algorithms [1]-[10], the frequency-domain (or transform-domain) LMS algorithm [11]-[13], the normalized LMS algorithm [14], and the LMS-Newton algorithm [15]-[16]. Several algorithms that are related to the main algorithms presented in this chapter are also briefly discussed.

The quantized-error algorithms reduce the computational complexity of the LMS algorithms by representing the error signal with short wordlength or by a simple power-of-two number.

The convergence speed in the LMS-Newton algorithm is independent of the eigenvalue spread of the input signal correlation matrix. This improvement is achieved by using an estimate of the inverse of the input signal correlation matrix leading to a substantial increase in the computational complexity.

The normalized LMS algorithm utilizes a variable convergence factor that minimizes the instantaneous error. Such a convergence factor usually reduces the convergence time but increases the misadjustment.

In the frequency-domain algorithm, a transform is applied to the input signal in order to allow the reduction of the eigenvalue spread of the transformed

signal correlation matrix as compared to the eigenvalue spread of the input signal correlation matrix. The LMS algorithm applied to the better conditioned transformed signal achieves faster convergence.

4.2 QUANTIZED-ERROR ALGORITHMS

The computational complexity of the LMS algorithm is mainly due to multiplications performed in the coefficient updating and in the calculation of the adaptive filter output. In applications where the adaptive filters are required to operate in high speed, such as echo cancellation and channel equalization, it is important to minimize hardware complexity.

A first step to simplify the LMS algorithm is to apply quantization to the error signal, generating the quantized-error algorithm which updates the filter coefficients according to

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu Q[e(k)]\mathbf{x}(k) \tag{4.1}$$

where $Q[.]$ represents a quantization operation. The quantization function is discrete valued, bounded, and nondecreasing. The type of quantization identifies the quantized-error algorithm.

If the convergence factor μ is a power-of-two number, the coefficient updating can be implemented with simple multiplications, basically consisting of bit shifts and additions. In a number of applications, such as the echo cancellation in full-duplex data transmission [2] and equalization of channels with binary data [3], the input signal $x(k)$ is a binary signal, i.e., assumes values +1 and -1. In this case, the adaptive filter can be implemented without any intricate multiplication.

The quantization of the error actually implies a modification in the objective function that is minimized, denoted by $F[e(k)]$. In a general gradient-type algorithm coefficient updating is performed by

$$\begin{aligned} \mathbf{w}(k+1) &= \mathbf{w}(k) - \mu \frac{\partial F[e(k)]}{\partial \mathbf{w}(k)} \\ &= \mathbf{w}(k) - \mu \frac{\partial F[e(k)]}{\partial e(k)} \frac{\partial e(k)}{\partial \mathbf{w}(k)} \end{aligned} \tag{4.2}$$

For a linear combiner the equation above can be rewritten as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \frac{\partial F[e(k)]}{\partial e(k)} \mathbf{x}(k) \tag{4.3}$$

Therefore, the objective function that is minimized in the quantized-error algorithms is such that

$$\frac{\partial F[e(k)]}{\partial e(k)} = 2Q[e(k)] \tag{4.4}$$

where $F[e(k)]$ is obtained by integrating $2Q[e(k)]$ with respect to $e(k)$. Note that the chain rule applied in equation (4.3) is not valid at the points of discontinuity of $Q[.]$ where $F[e(k)]$ is not differentiable [6].

The performances of the quantized-error and LMS algorithms are obviously different. The analyses of some widely used quantized-error algorithms are presented in the following subsections.

4.2.1 Sign-Error Algorithm

The simplest form for the quantization function is the sign (*sgn*) function defined by

$$\text{sgn}[b] = \begin{cases} 1, & b > 0 \\ 0, & b = 0 \\ -1, & b < 0 \end{cases} \tag{4.5}$$

The sign-error algorithm utilizes the sign function as the error quantizer, where the coefficient vector updating is performed by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu \, \text{sgn}[e(k)] \, \mathbf{x}(k) \tag{4.6}$$

Fig. 4.1 illustrates the realization of the sign-error algorithm for a delay line input $\mathbf{x}(k)$. If μ is a power-of-two number, one iteration of the sign-error algorithm requires $N+1$ multiplications for the error generation. The total number of additions is $2N+2$. The detailed description of the sign-error algorithm is shown in Algorithm 4.1. Obviously, the vectors $\mathbf{x}(0)$ and $\mathbf{w}(0)$ can be initialized in a different way from that described in the algorithm.

The objective function that is minimized by the sign-error algorithm is the modulus of the error multiplied by two, i.e.,

$$F[e(k)] = 2|e(k)| \tag{4.7}$$

Note that the factor two is included only to present the sign-error and LMS algorithms in a unified form. Obviously, in real implementation this factor can be merged with convergence factor μ.

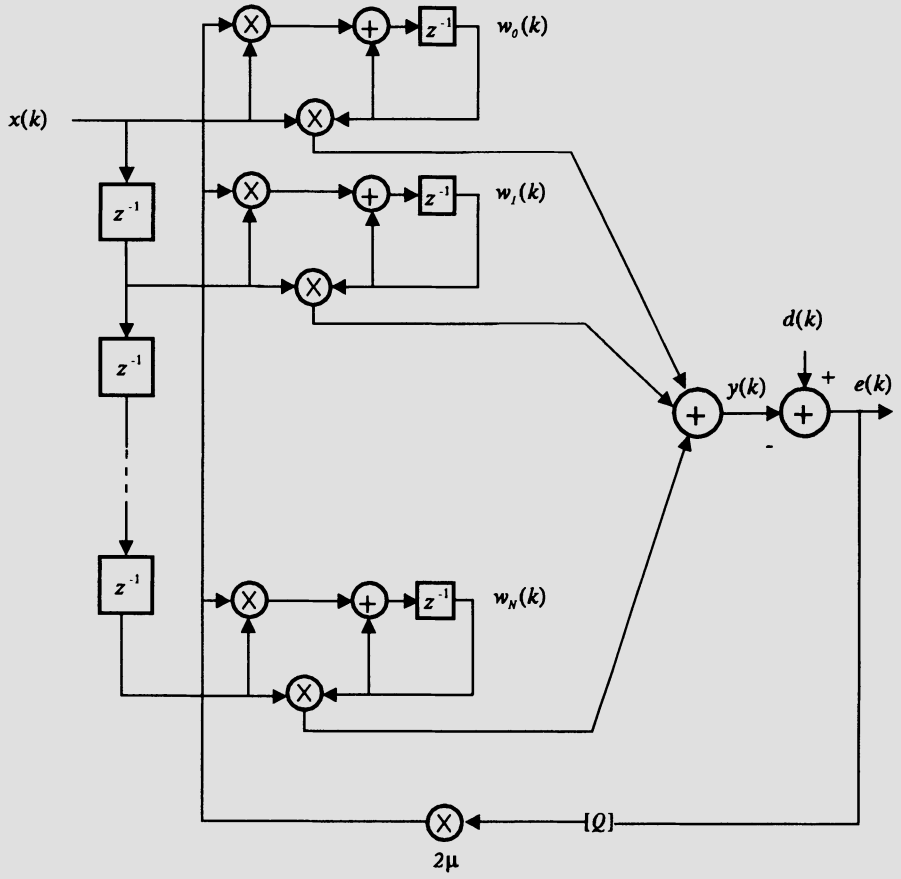


Figure 4.1 Sign-error adaptive FIR filter: $Q[e(k)] = sgn[e(k)]$.

Some of the properties related to the convergence behavior of the sign-error algorithm in a stationary environment are described, following the same procedure used in the previous chapter for the LMS algorithm.

Steady-State Behavior of the Coefficient Vector

The sign-error algorithm can be alternatively described by

$$\Delta \mathbf{w}(k+1) = \Delta \mathbf{w}(k) + 2\mu \ \text{sgn}[e(k)] \ \mathbf{x}(k) \tag{4.8}$$

Algorithm 4.1

Sign-Error Algorithm

Initialization
$\mathbf{x}(0) = \mathbf{w}(0) = [0 \ldots 0]^T$
Do for $k \geq 0$
$e(k) = d(k) - \mathbf{x}^T(k)\mathbf{w}(k)$
$\rho = \text{sgn}[e(k)]$
$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu\rho\mathbf{x}(k)$

□

where $\Delta\mathbf{w}(k) = \mathbf{w}(k) - \mathbf{w}_o$. The expected value of the coefficient-error vector is then given by

$$E[\Delta\mathbf{w}(k+1)] = E[\Delta\mathbf{w}(k)] + 2\mu E[\text{sgn}[e(k)]\ \mathbf{x}(k)] \tag{4.9}$$

The importance of the probability density function of the measurement noise $n(k)$ on the convergence of the sign-error algorithm is a noteworthy characteristic. This is due to the fact that $E\{\text{sgn}[e(k)]\ \mathbf{x}(k)\} = E\{\text{sgn}[-\Delta\mathbf{w}^T(k)\mathbf{x}(k) + n(k)]\mathbf{x}(k)\}$, where the result of the sign operation is highly dependent on the probability density function of $n(k)$. In [1], the authors present a convergence analysis of the output MSE, i.e., $E[e^2(k)]$, for different distributions of the additional noise, such as Gaussian, uniform, and binary distributions.

A closer examination of equation (4.8) indicates that even if the error signal becomes very small, the adaptive filter coefficients will be continually updated due to the sign function applied to the error signal. Therefore, in a situation where the adaptive filter has a sufficient number of coefficients to model the desired signal, and there is no additional noise, $\Delta\mathbf{w}(k)$ will not converge to zero. In this case, $\mathbf{w}(k)$ will be convergent to a balloon centered at $\mathbf{w}_o$, when μ is appropriately chosen. The mean absolute value of $e(k)$ is also convergent to a balloon centered around zero, that means $|e(k)|$ remains smaller than the balloon radius r [6].

Recalling that the desired signal without measurement noise is denoted as $d'(k)$, if it is considered that $d'(k)$ and the elements of $\mathbf{x}(k)$ are zero mean and jointly Gaussian and that the additional noise $n(k)$ is also zero mean, Gaussian, and independent of $\mathbf{x}(k)$ and $d'(k)$, the error signal will also be a zero-mean Gaussian signal conditioned on $\Delta\mathbf{w}(k)$. In this case, using the results of the Price theorem described in [18] and in Papoulis [19], the following result is valid

$$E\{\text{sgn}[e(k)]\ \mathbf{x}(k)\} \approx \sqrt{\frac{2}{\pi\xi(k)}} E[\mathbf{x}(k)e(k)] \tag{4.10}$$

where $\xi(k)$ is the variance of $e(k)$ assuming the error has zero mean. The approximation above is valid for small values of μ. For large μ, $e(k)$ is dependent on $\Delta\mathbf{w}(k)$ and conditional expected value on $\Delta\mathbf{w}(k)$ should be used instead [3]-[5].

By applying (4.10) in (4.9) and by replacing $e(k)$ by $e_o(k) + \Delta\mathbf{w}^T(k)\mathbf{x}(k)$, it follows that

$$\begin{aligned} E[\Delta\mathbf{w}(k+1)] &= \left\{\mathbf{I} - 2\mu\sqrt{\frac{2}{\pi\xi(k)}} E[\mathbf{x}(k)\mathbf{x}^T(k)]\right\} E[\Delta\mathbf{w}(k)] \\ &+ 2\mu\sqrt{\frac{2}{\pi\xi(k)}}\ E[e_o(k)\mathbf{x}(k)] \end{aligned} \tag{4.11}$$

From the orthogonality principle we know that $E[e_o(k)\mathbf{x}(k)] = \mathbf{0}$, so that the last element of the above equation is zero. Therefore,

$$E[\Delta\mathbf{w}(k+1)] = \left[\mathbf{I} - 2\mu\sqrt{\frac{2}{\pi\xi(k)}}\mathbf{R}\right]\ E[\Delta\mathbf{w}(k)] \tag{4.12}$$

Following the same steps for the analysis of $E[\Delta\mathbf{w}(k)]$ in the traditional LMS algorithm, it can be shown that the coefficients of the adaptive filter implemented with the sign-error algorithm converge in the mean if the convergence factor is chosen in the range

$$0 < \mu < \frac{1}{\lambda_{max}}\sqrt{\frac{\pi\xi(k)}{2}} \tag{4.13}$$

where λ_{max} is the largest eigenvalue of $\mathbf{R}$. It should be mentioned that in case $\frac{\lambda_{max}}{\lambda_{min}}$ is large, the convergence speed of the coefficients depends on the value of λ_{min} that is related to the slowest mode in equation (4.12). This conclusion can be drawn by following the same steps of the convergence analysis of the LMS algorithm, where by applying a transformation to the equation (4.12) we obtain an equation similar to (3.17).

A more practical range for μ, avoiding the use of eigenvalue, is given by

$$0 < \mu < \frac{1}{tr[\mathbf{R}]}\sqrt{\frac{\pi\xi(k)}{2}} \tag{4.14}$$

Note that the upper bound for the value of μ requires the knowledge of the MSE, i.e., $\xi(k)$.

Coefficient-Error-Vector Covariance Matrix

The covariance of the coefficient-error vector defined as

$$\text{cov}[\Delta\mathbf{w}(k)] = E\left[(\mathbf{w}(k) - \mathbf{w}_o)\,(\mathbf{w}(k) - \mathbf{w}_o)^T\right] \tag{4.15}$$

is calculated by replacing (4.8) in (4.15) following the same steps used in the LMS algorithm. The resulting difference equation for $\text{cov}[\Delta\mathbf{w}(k)]$ is given by

$$\begin{aligned}\text{cov}[\Delta\mathbf{w}(k+1)] &= \text{cov}[\Delta\mathbf{w}(k)] + 2\mu E[\text{sgn}[e(k)]\mathbf{x}(k)\Delta\mathbf{w}^T(k)] \\ &\quad +2\mu E[\text{sgn}[e(k)]\Delta\mathbf{w}^T(k)\mathbf{x}(k)] + 4\mu^2\mathbf{R}\end{aligned} \tag{4.16}$$

The first term with expected value operation in the equation above can be expressed as

$$\begin{aligned}& E[\text{sgn}[e(k)]\mathbf{x}(k)\Delta\mathbf{w}^T(k)] \\ =\;& E[\text{sgn}[e_o(k) - \Delta\mathbf{w}^T(k)\mathbf{x}(k)]\mathbf{x}(k)\Delta\mathbf{w}^T(k)] \\ =\;& E\{E[\text{sgn}[e_o(k) - \Delta\mathbf{w}^T(k)\mathbf{x}(k)]\mathbf{x}(k)|\Delta\mathbf{w}(k)]\Delta\mathbf{w}^T(k)\}\end{aligned}$$

where $E[a|\Delta\mathbf{w}(k)]$ is the expected value of a conditioned on the value of $\Delta\mathbf{w}(k)$. In the first equality, $e(k)$ was replaced by the relation $d(k) - \mathbf{w}^T(k)\mathbf{x}(k) - \Delta\mathbf{w}_o^T\mathbf{x}(k) + \Delta\mathbf{w}_o^T\mathbf{x}(k) = e_o(k) - \Delta\mathbf{w}^T(k)\mathbf{x}(k)$. In the second equality, the concept of conditioned expected value was applied.

Using the Price theorem and considering that the minimum output error $e_o(k)$ is zero-mean and uncorrelated with $\mathbf{x}(k)$, the following approximations result

$$\begin{aligned}& E\{E[\text{sgn}[e_o(k) - \Delta\mathbf{w}^T(k)\mathbf{x}(k)]\mathbf{x}(k)|\Delta\mathbf{w}(k)]\Delta\mathbf{w}^T(k)\} \\ \approx\;& E\left\{\sqrt{\frac{2}{\pi\xi(k)}}E[e_o(k)\mathbf{x}(k) - \mathbf{x}(k)\mathbf{x}^T(k)\Delta\mathbf{w}(k)|\Delta\mathbf{w}(k)]\Delta\mathbf{w}^T(k)\right\} \\ \approx\;& -E\left\{\sqrt{\frac{2}{\pi\xi(k)}}\mathbf{R}\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k)\right\}\end{aligned}$$

$$= -\sqrt{\frac{2}{\pi\xi(k)}}\mathbf{R}\text{cov}[\Delta\mathbf{w}(k)] \tag{4.17}$$

Following similar steps to derive the equation above, the second term with the expected value operation in (4.16) can be approximated as

$$E\left[\text{sgn}[e(k)]\Delta\mathbf{w}^T(k)x(k)\right] \approx -\sqrt{\frac{2}{\pi\xi(k)}}\text{cov}[\Delta\mathbf{w}(k)]\mathbf{R} \tag{4.18}$$

Substituting equations (4.17) and (4.18) in equation (4.16), we can calculate the vector $\mathbf{v}'(k)$ consisting of diagonal elements of $\text{cov}[\Delta\mathbf{w}'(k)]$, using the same steps employed in the LMS case (see equation (3.26)). The resulting dynamic equation for $\mathbf{v}'(k)$ is given by

$$\mathbf{v}'(k+1) = \left(\mathbf{I} - 4\mu\sqrt{\frac{2}{\pi\xi(k)}}\,\mathbf{\Lambda}\right)\,\mathbf{v}'(k) + 4\mu^2\mathbf{\Lambda} \tag{4.19}$$

The value of μ must be chosen in a range that guarantees the convergence of $\mathbf{v}'(k)$, which is given by

$$0 < \mu < \frac{1}{2\lambda_{max}}\sqrt{\frac{\pi\xi(k)}{2}} \tag{4.20}$$

A more severe and practical range for μ is

$$0 < \mu < \frac{1}{2tr[\mathbf{R}]}\sqrt{\frac{\pi\xi(k)}{2}} \tag{4.21}$$

For $k \to \infty$ each element of $\mathbf{v}'(k)$ tends to

$$v_i(\infty) = \mu\sqrt{\frac{\pi\xi(\infty)}{2}} \tag{4.22}$$

Excess of Mean-Square Error and Misadjustment

The excess of MSE can be expressed as a function of the elements of $\mathbf{v}'(k)$ by

$$\Delta\xi(k) = \sum_{i=0}^{N}\lambda_i v_i(k) = \mathbf{\lambda}^T\mathbf{v}'(k) \tag{4.23}$$

Substituting (4.22) in (4.23) yields

$$\begin{aligned} \xi_{exc} &= \mu \sum_{i=0}^{N} \lambda_i \sqrt{\frac{\pi \xi(k)}{2}}, k \to \infty \\ &= \mu \sum_{i=0}^{N} \lambda_i \sqrt{\pi \frac{\xi_{min} + \xi_{exc}}{2}} \end{aligned} \tag{4.24}$$

since $\lim_{k \to \infty} \xi(k) = \xi_{min} + \xi_{exc}$. Therefore,

$$\xi_{exc}^2 = \mu^2 \left(\sum_{i=0}^{N} \lambda_i \right)^2 \left(\frac{\pi \xi_{min}}{2} + \frac{\pi \xi_{exc}}{2} \right) \tag{4.25}$$

There are two solutions for ξ_{exc}^2 in the equation above, where only the positive one is valid. The meaningful solution for ξ_{exc}, when μ is small, is approximately given by

$$\begin{aligned} \xi_{exc} &\approx \mu \sqrt{\frac{\pi \xi_{min}}{2}} \sum_{i=0}^{N} \lambda_i \\ &= \mu \sqrt{\frac{\pi \xi_{min}}{2}} \, tr[\mathbf{R}] \end{aligned} \tag{4.26}$$

By comparing the excess of MSE predicted by the equation above with the corresponding equation for the LMS, it can be concluded that both can generate the same excess of MSE if μ in the sign-error algorithm is chosen such that

$$\mu = \mu_{LMS} \sqrt{\frac{2}{\pi} \xi_{min}} \tag{4.27}$$

The misadjustment in the sign-error algorithm is

$$M = \mu \sqrt{\frac{\pi}{2 \xi_{min}}} \, tr[\mathbf{R}] \tag{4.28}$$

Equation (4.26) would leave the impression that if there is no additional noise and there is sufficient parameters in the adaptive filter, the output MSE would converge to zero. However, when $\xi(k)$ becomes small, $||E[\Delta \mathbf{w}(k+1)]||$ in equation (4.11) can increase, since the condition of equation (4.13) will not be satisfied. This is the situation where the parameters reach the convergence balloon. In this case, from (4.8) we can conclude that

$$||\Delta \mathbf{w}(k+1)||^2 - ||\Delta \mathbf{w}(k)||^2 = -4\mu \, \text{sgn}[e(k)] \, e(k) + 4\mu^2 ||\mathbf{x}(k)||^2 \tag{4.29}$$

from where it is easy to show that a decrease in the norm of $\Delta\mathbf{w}(k)$ is obtained only when

$$|e(k)| > \mu||\mathbf{x}(k)||^2 \tag{4.30}$$

For no additional noise, first transpose the vectors in equation (4.8) and post-multiply each side by $\mathbf{x}(k)$. Next, squaring the resulting equation and applying the expected value operation on each side, the obtained result is

$$E[e^2(k+1)] = E[e^2(k)] - 4\mu E[|e(k)|\ ||\mathbf{x}(k)||^2] + 4\mu^2 E[||\mathbf{x}(k)||^4] \tag{4.31}$$

After convergence $E[e^2(k+1)] \approx E[e^2(k)]$. Also, considering that

$$E[|e(k)|\ ||\mathbf{x}(k)||^2] \approx E[|e(k)|\ ]E[||\mathbf{x}(k)||^2]$$

and

$$\frac{E[||\mathbf{x}(k)||^4]}{E[||\mathbf{x}(k)||^2]} \approx E[||\mathbf{x}(k)||^2]$$

we conclude that

$$E[|e(k)|] \approx \mu E[||x(k)||^2], k \to \infty \tag{4.32}$$

For a zero-mean Gaussian $e(k)$, the following approximation is valid

$$E[|e(k)|] \approx \sqrt{\frac{2}{\pi}}\sigma_e(k), k \to \infty \tag{4.33}$$

therefore, the expected variance of $e(k)$ is

$$\sigma_e^2(k) \approx \frac{\pi}{2}\mu^2\ tr^2[\mathbf{R}], k \to \infty \tag{4.34}$$

where we used the relation $tr[\mathbf{R}] = E[||\mathbf{x}(k)||^2]$. This relation gives an estimate of the variance of the output error when no additional noise exists. As can be noted, unlike the LMS algorithm, there is an excess of MSE in the sign-error algorithm caused by the nonlinear device, even when $\sigma_n^2 = 0$.

If frequently $n(k)$ has large absolute values as compared to $-\Delta\mathbf{w}^T(k)\mathbf{x}(k)$, then for most iterations $\text{sgn}[e(k)] = \text{sgn}[n(k)]$. As a result, the sign-error algorithm is fully controlled by the additional noise. In this case, the algorithm does not converge.

Transient Behavior

The ratios r_{w_i} of the geometric decaying convergence curves of the coefficients in the sign-error algorithm can be easily derived from equation (4.18) by employing

an identical analysis of the transient behavior for the LMS algorithm. The ratios are given by

$$r_{w_i} = \left(1 - 2\mu\sqrt{\frac{2}{\pi\xi(k)}}\lambda_i\right) \tag{4.35}$$

for $i = 0, 1, \ldots, N$. If μ is chosen as suggested in equation (4.27), in order to reach the same excess of MSE of the LMS algorithm, then

$$r_{w_i} = \left(1 - \frac{4}{\pi}\mu_{LMS}\sqrt{\frac{\xi_{min}}{\xi(k)}}\ \lambda_i\right) \tag{4.36}$$

By recalling that r_{w_i} for the LMS algorithm is $(1 - 2\mu_{LMS}\lambda_i)$, since $\frac{2}{\pi}\sqrt{\frac{\xi_{min}}{\xi(k)}} < 1$, it is concluded that the sign-error algorithm is slower than the LMS for the same excess of MSE.

Example 4.1

Suppose in an adaptive filtering environment that the input signal consists of

$$x(k) = e^{j\omega_0 k} + n(k)$$

and that the desired signal is given by

$$d(k) = e^{j\omega_0(k-1)}$$

where $n(k)$ is a uniformly distributed white noise with variance $\sigma_n^2 = 0.1$ and $\omega_0 = \frac{2\pi}{M}$. In this case $M = 8$.

Compute the input-signal correlation matrix for a first-order adaptive filter. Calculate the value of μ_{max} for the sign-error algorithm.

Solution:

The input-signal correlation matrix for this example can be easily calculated as shown below:

$$\mathbf{R} = \begin{bmatrix} 1 + \sigma_n^2 & e^{j\omega_0} \\ e^{-j\omega_0} & 1 + \sigma_n^2 \end{bmatrix}$$

Since in this case $tr[\mathbf{R}] = 2.2$ and $\xi_{min} = 0.1$, we have

$$\xi_{exc} \approx \mu\sqrt{\frac{\pi\xi_{min}}{2}}\ tr[\mathbf{R}] = 0.87\mu$$

The range of values of the convergence factor is given by

$$0 < \mu < \frac{1}{2tr[\mathbf{R}]}\sqrt{\frac{\pi(\xi_{min} + \xi_{exc})}{2}}$$

From the expression above it is straightforward to calculate the upper bound for the convergence factor that is given by

$$\mu_{max} \approx 0.132$$

□

4.2.2 Dual-Sign Algorithm

The dual-sign algorithm attempts to perform large corrections to the coefficient vector when the modulus of the error signal is larger than a prescribed level. The basic motivation to use the dual-sign algorithm is to avoid the slow convergence inherent to the sign-error algorithm that is caused by replacing $e(k)$ by $\text{sgn}[e(k)]$ when $|e(k)|$ is large.

The quantization function for the dual-sign algorithm is given by

$$\text{ds}[a] = \begin{cases} \gamma\ \text{sgn}[a], & |a| > \rho \\ \text{sgn}[a], & |a| \leq \rho \end{cases} \tag{4.37}$$

where $\gamma > 1$ is a power of two. The dual-sign algorithm utilizes the function described above as the error quantizer, and the coefficient updating is performed as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu\ \text{ds}[e(k)]\mathbf{x}(k) \tag{4.38}$$

The objective function that is minimized by the sign-error algorithm is given by

$$F[e(k)] = \begin{cases} 2\gamma|e(k)| - 2\rho(\gamma - 1), & |e(k)| > \rho \\ 2|e(k)|, & |e(k)| \leq \rho \end{cases} \tag{4.39}$$

where the constant $2\rho(\gamma - 1)$ was included in the objective function to make it continuous. Obviously the gradient of $F[e(k)]$ with respect to the filter coefficients is $2\mu\ \text{ds}[e(k)]\mathbf{x}(k)$ except at points where $\text{ds}[e(k)]$ is nondifferentiable [6].

The same analysis procedure used for the sign-error algorithm can be applied to the dual-sign algorithm except for the fact that the quantization function

is now different. The alternative quantization leads to particular expectations of nonlinear functions whose solutions are not presented here. The interested reader should refer to the work of Mathews [7]. The choice of γ and ρ determine the convergence behavior of the dual-sign algorithm [7]; typically, a large γ tends to increase both convergence speed and excess of MSE. A large ρ tends to reduce both the convergence speed and the excess of MSE. If $\lim_{k\to\infty} \xi(k) \ll \rho^2$, the excess of MSE of the dual-sign algorithm is approximately equal to the one given by (4.26) for the sign-error algorithm [7], since in this case $|e(k)|$ is usually much smaller than ρ. For a given MSE in steady state, the dual-sign algorithm is expected to converge faster than the sign-error algorithm.

4.2.3 Power-of-Two Error Algorithm

The power-of-two error algorithm applies to the error signal a quantization defined by

$$\text{pe}[b] = \begin{cases} \text{sgn}[b], & |b| \geq 1 \\ 2^{\text{int}[log_2|b|]}\,\text{sgn}[b], & 2^{-b_d+1} \leq |b| < 1 \\ \tau\text{sgn}[b], & |b| < 2^{-b_d+1} \end{cases} \tag{4.40}$$

where int[.] indicates integer part of [.], b_d is the data wordlength excluding the sign bit, and τ is usually 0 or 2^{-b_d}.

The coefficient updating for the power-of-two error algorithm is given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu\ \text{pe}[e(k)]\mathbf{x}(k) \tag{4.41}$$

For $\tau = 2^{-b_d}$, the additional noise and the convergence factor can be arbitrarily small and the algorithm will not stop updating. For $\tau = 0$, when $|e(k)| < 2^{-b_d+1}$ the algorithm reaches the so-called *dead zone*, where the algorithm stops updating if $|e(k)|$ is smaller than 2^{-b_d+1} most of the time [4], [8].

A simplified and somewhat accurate analysis of this algorithm can be performed by approximating the function pe$[e(k)]$ by a straight line passing through the center of each quantization step. In this case, the quantizer characteristics can be approximated by pe$[e(k)] \approx \frac{2}{3}e(k)$ as illustrated in Fig. 4.2. Using this approximation, the algorithm analysis can be performed exactly in the same way as the LMS algorithm. The results for the power-of-two error algorithm can be obtained from the results for the LMS algorithm, by replacing μ by $\frac{2}{3}\mu$. It should be mentioned that such results are only approximate, and more accurate ones can be found in [8].

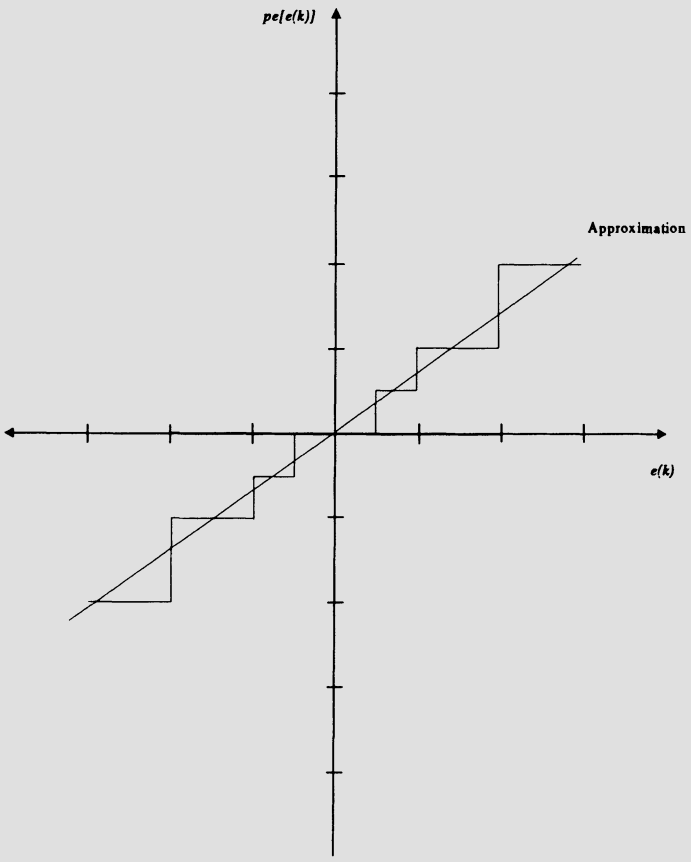


Figure 4.2 Transfer characteristic of a quantizer with 3 bits and $\tau = 0$.

4.2.4 Sign-Data Algorithm

The algorithms discussed in this subsection cannot be considered as quantized error algorithms, but since they were proposed with similar motivation we decided to introduce them here. An alternative way to simplify the computational burden of the LMS algorithm is to apply quantization to the data vector $\mathbf{x}(k)$. One possible quantization scheme is to apply the sign function to the input signals, giving rise to the sign-data algorithm whose coefficient updating is performed as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu e(k)\ \mathrm{sgn}[\mathbf{x}(k)] \tag{4.42}$$

where the sign operation is applied to each element of the input vector.

The quantization of the data vector can lead to a decrease in the convergence speed, and possible divergence. In the LMS algorithm, the average gradient direction follows the true gradient direction (or steepest descent direction), whereas in the sign-data algorithm only a discrete set of directions can be followed. The limitation in the gradient direction followed by the sign-data algorithm may cause updates that result in frequent increase in the squared error, leading to instability. Therefore, it is relatively easy to find inputs that would lead to the convergence of the LMS algorithm and to the divergence of the sign-data algorithm [6], [9]. It should be mentioned, however, that the sign-data algorithm is stable for Gaussian inputs, and, as such, has been found useful in certain applications.

Another related algorithm is the sign-sign algorithm that has very simple implementation. The coefficient updating in this case is given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu \, \text{sgn}[e(k)] \, \text{sgn}[\mathbf{x}(k)] \tag{4.43}$$

The sign-sign algorithm also presents the limitations related to the quantized-data algorithm.

4.3 THE LMS-NEWTON ALGORITHM

In this section, the LMS-Newton algorithm incorporating estimates of the second-order statistics of the environment signals is introduced. The objective of the algorithm is to avoid the slow convergence of the LMS algorithm when the input signal is highly correlated. The improvement in the convergence rate is achieved at the expense of an increased computational complexity.

Nonrecursive realization of the adaptive filter leads to a MSE surface that is a quadratic function of the filter coefficients. For the direct form FIR structure, the MSE can be described by

$$\begin{aligned} \xi(k+1) &= \xi(k) + \mathbf{g_w}^T(k)\,(\mathbf{w}(k+1) - \mathbf{w}(k)) \\ &\quad + (\mathbf{w}(k+1) - \mathbf{w}(k))^T \mathbf{R}\,(\mathbf{w}(k+1) - \mathbf{w}(k)) \end{aligned} \tag{4.44}$$

$\xi(k)$ represents the MSE when the adaptive filter coefficients are fixed at $\mathbf{w}(k)$ and $\mathbf{g_w}(k) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(k)$ is the gradient vector of the MSE surface as related to the filter coefficients at $\mathbf{w}(k)$. The MSE is minimized at the instant $k+1$ if

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \frac{1}{2}\mathbf{R}^{-1}\mathbf{g_w}(k) \tag{4.45}$$

This equation is the updating formula of the Newton method. Note that in the ideal case, where matrix $\mathbf{R}$ and gradient vector $\mathbf{g}_{\mathbf{w}}(k)$ are known precisely, $\mathbf{w}(k+1) = \mathbf{R}^{-1}\mathbf{p} = \mathbf{w}_o$. Therefore, the Newton method converges to the optimal solution in a single iteration, as expected for a quadratic objective function.

In practice, only estimates of the autocorrelation matrix $\mathbf{R}$ and of the gradient vector are available. These estimates can be applied to the Newton updating formula in order to derive a Newton-like method given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu\hat{\mathbf{R}}^{-1}(k)\hat{\mathbf{g}}_{\mathbf{w}}(k) \tag{4.46}$$

The convergence factor μ is introduced so that the algorithm can be protected from divergence, originated by the use of noisy estimates of $\mathbf{R}$ and $\mathbf{g}_{\mathbf{w}}(k)$.

For stationary input signals, an unbiased estimate of $\mathbf{R}$ is

$$\begin{aligned} \hat{\mathbf{R}}(k) &= \frac{1}{k+1}\sum_{i=0}^{k}\mathbf{x}(i)\mathbf{x}^T(i) \\ &= \frac{k}{k+1}\hat{\mathbf{R}}(k-1) + \frac{1}{k+1}\mathbf{x}(k)\mathbf{x}^T(k) \end{aligned} \tag{4.47}$$

since

$$\begin{aligned} E[\hat{\mathbf{R}}(k)] &= \frac{1}{k+1}\sum_{i=0}^{k}E[\mathbf{x}(i)\mathbf{x}^T(i)] \\ &= \mathbf{R} \end{aligned} \tag{4.48}$$

However, this is not a practical estimate for $\mathbf{R}$, since for large k any change on the input signal statistics would be disregarded due to the infinite memory of the estimation algorithm.

Another form to estimate the autocorrelation matrix can be generated by employing a weighted summation as follows:

$$\begin{aligned} \hat{\mathbf{R}}(k) &= \alpha\mathbf{x}(k)\mathbf{x}^T(k) + (1-\alpha)\hat{\mathbf{R}}(k-1) \\ &= \alpha\mathbf{x}(k)\mathbf{x}^T(k) + \alpha\sum_{i=0}^{k-1}(1-\alpha)^{k-i}\mathbf{x}(i)\mathbf{x}^T(i) \end{aligned} \tag{4.49}$$

where in practice, α is a small factor chosen in the range $0 < \alpha \leq 0.1$. This range of values of α allows a good balance between the present and past input signal information. By taking the expected value on both sides of the equation

above and assuming that $k \to \infty$, it follows that

$$\begin{aligned} E[\hat{\mathbf{R}}(k)] &= \alpha \sum_{i=0}^{k} (1-\alpha)^{k-i} E[\mathbf{x}(i)\mathbf{x}^T(i)] \\ &= \mathbf{R} \qquad k \to \infty \end{aligned} \tag{4.50}$$

Therefore, the estimate of $\mathbf{R}$ of equation (4.49) is unbiased.

In order to avoid inverting $\hat{\mathbf{R}}(k)$, which is required by the Newton-like algorithm, we can use the so-called matrix inversion lemma given by

$$[\mathbf{A} + \mathbf{BCD}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{DA}^{-1}\mathbf{B} + \mathbf{C}^{-1}]^{-1}\mathbf{DA}^{-1} \tag{4.51}$$

where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ are matrices of appropriate dimensions, and $\mathbf{A}$ and $\mathbf{C}$ are nonsingular. The relation above can be proved by simply showing that the result of premultiplying the expression in the right hand side by $\mathbf{A} + \mathbf{BCD}$ is the identity matrix (see problem (19)). If we choose $\mathbf{A} = (1-\alpha)\,\hat{\mathbf{R}}(k-1)$, $\mathbf{B} = \mathbf{D}^T = \mathbf{x}(k)$, and $\mathbf{C} = \alpha$, it can be shown that

$$\hat{\mathbf{R}}^{-1}(k) = \frac{1}{1-\alpha}\left[\hat{\mathbf{R}}^{-1}(k-1) - \frac{\hat{\mathbf{R}}^{-1}(k-1)\mathbf{x}(k)\mathbf{x}^T(k)\hat{\mathbf{R}}^{-1}(k-1)}{\frac{1-\alpha}{\alpha} + \mathbf{x}^T(k)\hat{\mathbf{R}}^{-1}(k-1)\mathbf{x}(k)}\right] \tag{4.52}$$

The resulting equation to calculate $\hat{\mathbf{R}}^{-1}(k)$ is less complex to update (of order N^2 multiplications) than the direct inversion of $\hat{\mathbf{R}}(k)$ at every iteration (of order N^3 multiplications).

If the estimate for the gradient vector used in the LMS algorithm is applied in equation (4.46), the LMS-Newton algorithm results with the following coefficient updating formula

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\,\mu\, e(k)\,\hat{\mathbf{R}}^{-1}(k)\mathbf{x}(k) \tag{4.53}$$

The complete LMS-Newton algorithm is outlined in Algorithm 4.2. It should be noticed that alternative initialization procedures to the one presented in Algorithm 4.2 are possible.

As previously mentioned, the LMS gradient direction has the tendency to approach the ideal gradient direction. Similarly, the vector resulting from the

Algorithm 4.2

LMS-Newton Algorithm

Initialization

$\hat{\mathbf{R}}^{-1}(-1) = \delta \mathbf{I}$ (δ a small positive constant)

$\mathbf{w}(0) = \mathbf{x}(-1) = [0...0]^T$

Do for $k \geq 0$

$e(k) = d(k) - \mathbf{x}^T(k)\mathbf{w}(k)$

$$\hat{\mathbf{R}}^{-1}(k) = \frac{1}{1-\alpha}\left[\hat{\mathbf{R}}^{-1}(k-1) - \frac{\hat{\mathbf{R}}^{-1}(k-1)\mathbf{x}(k)\mathbf{x}^T(k)\hat{\mathbf{R}}^{-1}(k-1)}{\frac{1-\alpha}{\alpha} + \mathbf{x}^T(k)\hat{\mathbf{R}}^{-1}(k-1)\mathbf{x}(k)}\right]$$

$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\,\mu\, e(k)\,\hat{\mathbf{R}}^{-1}(k)\mathbf{x}(k)$

□

multiplication of $\hat{\mathbf{R}}^{-1}(k)$ to the LMS gradient direction tends to approach the Newton direction. Therefore, we can conclude that the LMS-Newton algorithm converges in a more straightforward path to the minimum of the MSE surface. It can also be shown that the convergence characteristics of the algorithm is independent of the eigenvalue spread of $\mathbf{R}$.

The LMS-Newton algorithm is mathematically identical to the recursive least-squares (RLS) algorithm if the forgetting factor (λ) in the latter is chosen such that $2\mu = \alpha = 1 - \lambda$ [32]. Since a complete discussion of the RLS algorithm is given later, no further discussion of the LMS-Newton algorithm is included here.

4.4 THE NORMALIZED LMS ALGORITHM

If one wishes to increase the convergence speed of the LMS algorithm without using estimates of the input signal correlation matrix, a variable convergence factor is a natural solution. The normalized LMS algorithm usually converges

faster than the LMS algorithm, since it utilizes a variable convergence factor aiming the minimization of the instantaneous output error.

The updating equation of the LMS algorithm can employ a variable convergence factor μ_k in order to improve the convergence rate. In this case, the updating formula is expressed as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu_k e(k)\mathbf{x}(k) = \mathbf{w}(k) + \Delta\mathbf{w}'(k) \tag{4.54}$$

where μ_k must be chosen with the objective of achieving a faster convergence. A possible strategy is to reduce the instantaneous squared error as much as possible. The motivation behind this strategy is that the instantaneous squared error is a good and simple estimate of the MSE.

The instantaneous squared error is given by

$$e^2(k) = d^2(k) + \mathbf{w}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}(k) - 2d(k)\mathbf{w}^T(k)\mathbf{x}(k) \tag{4.55}$$

If a change given by $\mathbf{w}'(k) = \mathbf{w}(k) + \Delta\mathbf{w}'(k)$ is performed in the weight vector, the corresponding squared error can be shown to be

$$\begin{aligned} e'^2(k) &= e^2(k) + 2\Delta\mathbf{w}'^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}(k) \\ &+ \Delta\mathbf{w}'^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\Delta\mathbf{w}'(k) \\ &- 2d(k)\Delta\mathbf{w}'^T(k)\mathbf{x}(k) \end{aligned} \tag{4.56}$$

It then follows that

$$\begin{aligned} \Delta e^2(k) &\triangleq e'^2(k) - e^2(k) \\ &= -2\Delta\mathbf{w}'^T(k)\mathbf{x}(k)e(k) + \Delta\mathbf{w}'^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\Delta\mathbf{w}'(k) \end{aligned} \tag{4.57}$$

In order to increase the convergence rate, the objective is to maximize the squared error reduction by appropriately choosing μ_k.

By replacing $\Delta\mathbf{w}'(k)$ in the equation above it follows that

$$\Delta e^2(k) = -4\mu_k e^2(k)\mathbf{x}^T(k)\mathbf{x}(k) + 4\mu_k^2 e^2(k)[\mathbf{x}^T(k)\mathbf{x}(k)]^2 \tag{4.58}$$

The value of μ_k such that $\frac{\partial \Delta e^2(k)}{\partial \mu_k} = 0$ is given by

$$\mu_k = \frac{1}{2\mathbf{x}^T(k)\mathbf{x}(k)} \tag{4.59}$$

Algorithm 4.3

The Normalized LMS Algorithm

Initialization

$\mathbf{x}(0) = \hat{\mathbf{w}}(0) = [0...0]^T$

choose μ_n in the range $0 < \mu_n \leq 2$

γ = small constant

Do for $k \geq 0$

$e(k) = d(k) - \mathbf{x}^T(k)\mathbf{w}(k)$

$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu_n}{\gamma + \mathbf{x}^T(k)\mathbf{x}(k)}\, e(k)\, \mathbf{x}(k)$

□

This value of μ_k leads to a negative value of $\Delta e^2(k)$, and, therefore, it corresponds to a minimum point of $\Delta e^2(k)$.

Using this variable convergence factor, the updating equation for the LMS algorithm is then given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{e(k)\mathbf{x}(k)}{\mathbf{x}^T(k)\mathbf{x}(k)} \tag{4.60}$$

Usually a fixed convergence factor μ_n is introduced in the updating formula in order to control the misadjustment, since all the derivations are based on instantaneous values of the squared errors and not on the MSE. Also a parameter γ should be included, in order to avoid large step sizes when $\mathbf{x}^T(k)\mathbf{x}(k)$ becomes small. The coefficient updating equation is then given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu_n}{\gamma + \mathbf{x}^T(k)\mathbf{x}(k)}\, e(k)\, \mathbf{x}(k) \tag{4.61}$$

The resulting algorithm is called the normalized LMS algorithm, and is summarized in Algorithm 4.3.

The range of values of μ_n to guarantee stability can be easily derived by first considering that $E[\mathbf{x}^T(k)\mathbf{x}(k)] = tr[\mathbf{R}]$ and that

$$E\left[\frac{e(k)\mathbf{x}(k)}{\mathbf{x}^T(k)\mathbf{x}(k)}\right] \approx \frac{E[e(k)\mathbf{x}(k)]}{E[\mathbf{x}^T(k)\mathbf{x}(k)]}$$

Next, consider that the average value of the convergence factor actually applied to the LMS direction $2e(k)\mathbf{x}(k)$ is $\frac{\mu_n}{2\,tr[\mathbf{R}]}$. Finally, by comparing the updating formula of the standard LMS algorithm with that of the normalized LMS algorithm, the desired upper bound result follows:

$$0 < \mu = \frac{\mu_n}{2\,tr[\mathbf{R}]} < \frac{1}{tr[\mathbf{R}]} \tag{4.62}$$

or $0 < \mu_n < 2$.

4.5 THE TRANSFORM-DOMAIN LMS ALGORITHM

The transform-domain algorithm is another technique to increase the convergence speed of the LMS algorithm when the input signal is highly correlated. The basic idea behind this methodology is to modify the input signal to be applied to the adaptive filter such that the conditioning number of the corresponding correlation matrix is improved.

In the transform-domain LMS algorithm, the input signal vector $\mathbf{x}(k)$ is transformed in a more convenient vector $\mathbf{s}(k)$, by applying an orthonormal (or unitary) transform [10]-[12], i.e.,

$$\mathbf{s}(k) = \mathbf{T}\mathbf{x}(k) \tag{4.63}$$

where $\mathbf{T}\mathbf{T}^T = \mathbf{I}$. The MSE surface related to the direct form implementation of the FIR adaptive filter can be described by

$$\xi(k) = \xi_{min} + \Delta\mathbf{w}^T(k)\mathbf{R}\Delta\mathbf{w}(k) \tag{4.64}$$

where $\Delta\mathbf{w}(k) = \mathbf{w}(k) - \mathbf{w}_o$. In the transform-domain case, the MSE surface becomes

$$\begin{aligned}\xi(k) &= \xi_{min} + \Delta\hat{\mathbf{w}}^T(k)E[\mathbf{s}(k)\mathbf{s}^T(k)]\Delta\hat{\mathbf{w}}(k) \\ &= \xi_{min} + \Delta\hat{\mathbf{w}}^T(k)\mathbf{T}\mathbf{R}\mathbf{T}^T\Delta\hat{\mathbf{w}}(k)\end{aligned} \tag{4.65}$$

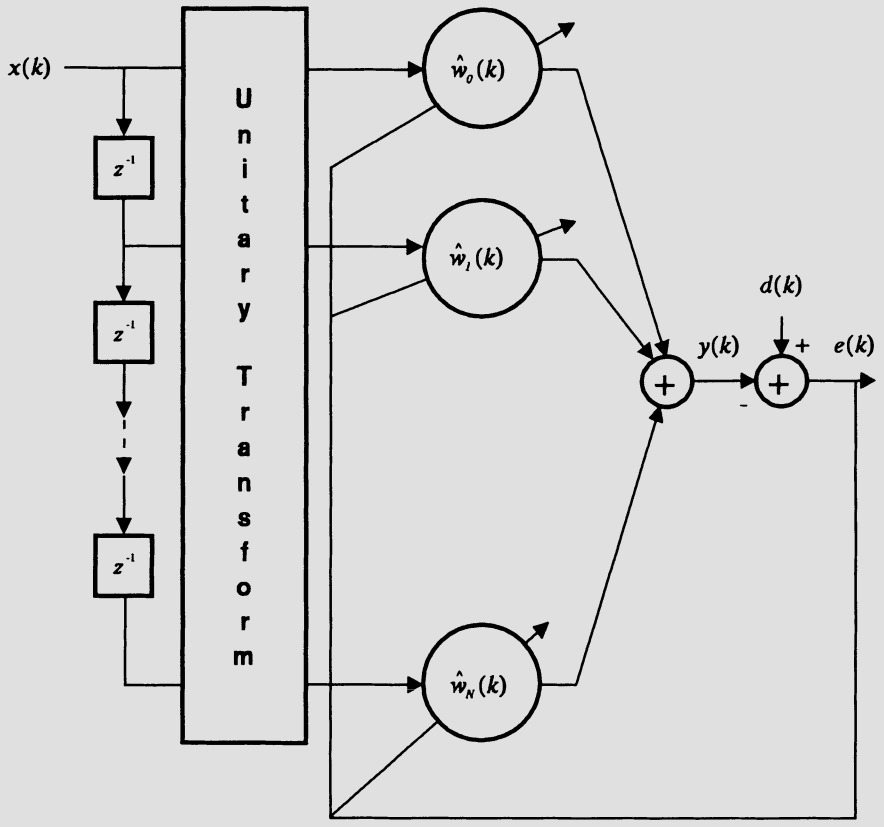
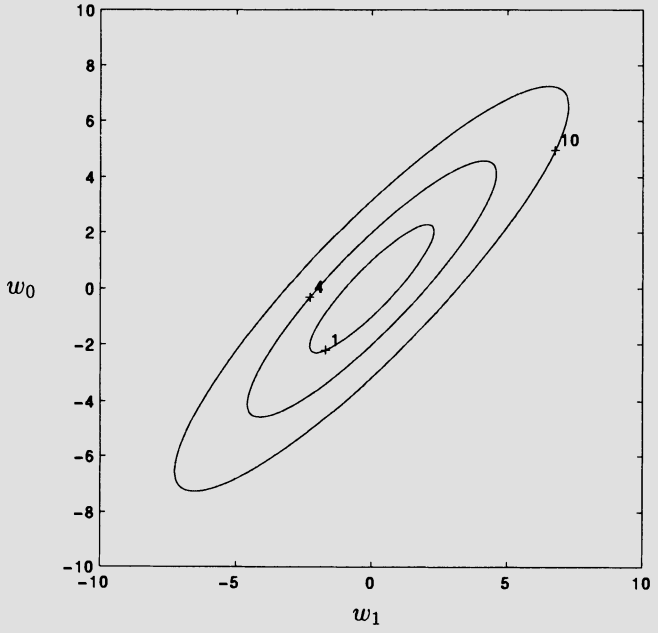


Figure 4.3 Transform-domain adaptive filter.

where $\hat{\mathbf{w}}(k)$ represents the adaptive coefficients of the transform-domain filter. Fig. 4.3 depicts the transform-domain adaptive filter.

The effect of applying the transformation matrix $\mathbf{T}$ to the input signal is to rotate the error surface as illustrated in the numerical examples of Figs. 4.4 and 4.5. It can be noticed that the eccentricity of the MSE surface remains unchanged by the application of the transformation, and, therefore, the eigenvalue spread is unaffected by the transformation. As a consequence, no improvement in the convergence rate is expected to occur. However, if in addition each element of the transform output is power normalized, the distance between the points where the equal-error contours meet the coefficient axes and the origin are equalized. As a result, a reduction in the eigenvalue spread is expected, specially when the coefficient axes are almost aligned with the principal axes of the ellipses.

Fig. 4.6 illustrates the effect of power normalization. The perfect alignment and power normalization means that the error surface will become a hyperparaboloid spheric, with the eigenvalue spread becoming equal to one. Alternatively, it means that the transform was able to turn the elements of the vector $\mathbf{s}(k)$ uncorrelated. Fig. 4.7 shows another error surface which after properly rotated and normalized is transformed in the error surface of Fig. 4.8.



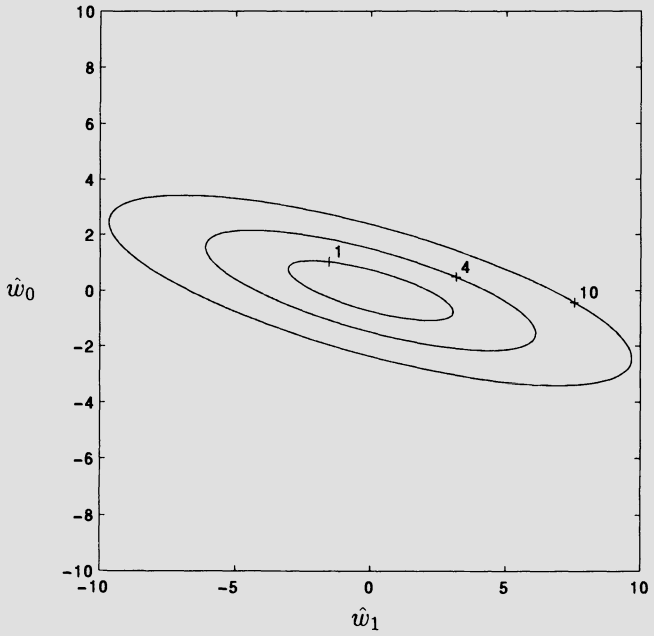
$$\mathbf{R} = \begin{bmatrix} 1 & -0.9 \\ -0.9 & 1 \end{bmatrix}$$

Figure 4.4 Contours of the original MSE surface.

The autocorrelation matrix related to the transform-domain filter is given by

$$\mathbf{R}_s = \mathbf{T}\mathbf{R}\mathbf{T}^T \tag{4.66}$$

therefore if the elements of $\mathbf{s}(k)$ are uncorrelated, matrix $\mathbf{R}_s$ is diagonal, meaning that the application of the transformation matrix was able to diagonalize the



$$\mathbf{T} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \quad \theta = 60°$$

Figure 4.5 Rotated contours of the MSE surface.

autocorrelation matrix $\mathbf{R}$. It can then be concluded that $\mathbf{T}^T$, in this case, corresponds to a matrix whose columns consist of the orthonormal eigenvectors of $\mathbf{R}$. The resulting transformation matrix corresponds to the Karhunen-Loève Transform (KLT)[17].

The normalization of $\mathbf{s}(k)$ and subsequent application of the LMS algorithm would lead to a transform-domain algorithm with the limitation that the solution would be independent of the input signal power. An alternative solution, without this limitation, is to apply the normalized LMS algorithm to update the coefficients of the transform-domain algorithm. We can give an interpretation for the good performance of this solution. Assuming the transform was

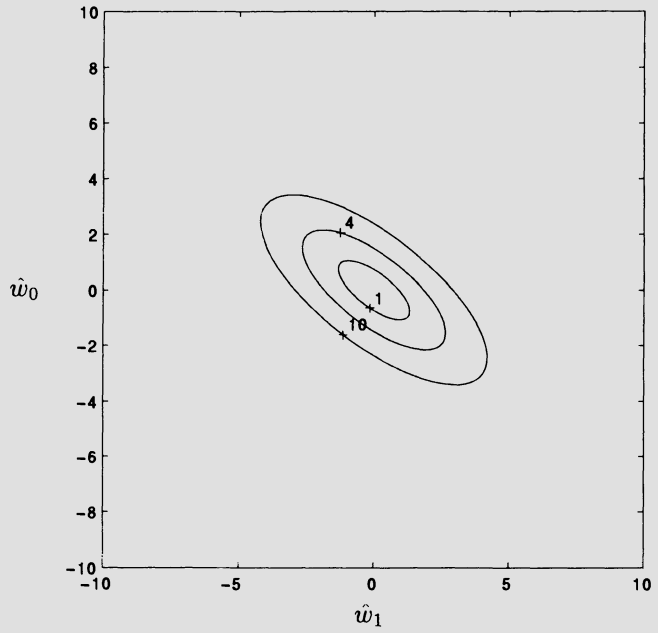


Figure 4.6 Contours of the power normalized MSE surface.

efficient in the rotation of the MSE surface, the variable convergence factor is large in the update of the coefficients corresponding to low signal power. On the other hand, the convergence factor is small if the corresponding transform output power is high. Specifically, the signals $s_i(k)$ are normalized by their power denoted by $\sigma_i^2(k)$ only when applied in the updating formula. The coefficient update equation in this case is

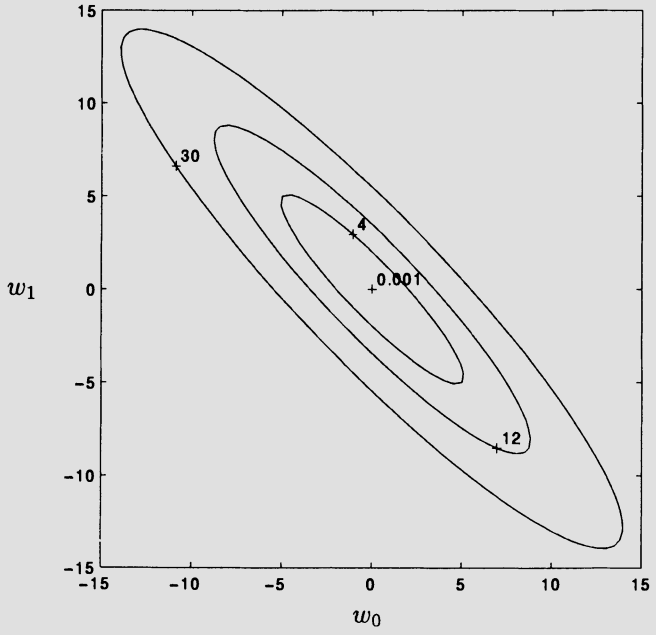
$$\hat{w}_i(k+1) = \hat{w}_i(k) + \frac{2\mu}{\gamma + \sigma_i^2(k)} e(k) s_i(k) \tag{4.67}$$

where $\sigma_i^2(k) = \alpha s_i^2(k) + (1-\alpha)\sigma_i^2(k-1)$, α is a small factor chosen in the range $0 < \alpha \leq 0.1$, and γ is also a small constant to avoid that the second term of the update equation becomes too large when $\sigma_i^2(k)$ is small.

In matrix form the updating equation above can be rewritten as

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + 2\mu e(k) \boldsymbol{\sigma}^{-2}(k) \mathbf{s}(k) \tag{4.68}$$

where $\boldsymbol{\sigma}^{-2}(k)$ is a diagonal matrix containing as elements the inverse of the power estimates of the elements of $\mathbf{s}(k)$ added to γ.



$$\mathbf{R} = \begin{bmatrix} 1 & 0.92 \\ 0.92 & 1 \end{bmatrix}$$

Figure 4.7 Contours of the original MSE surface.

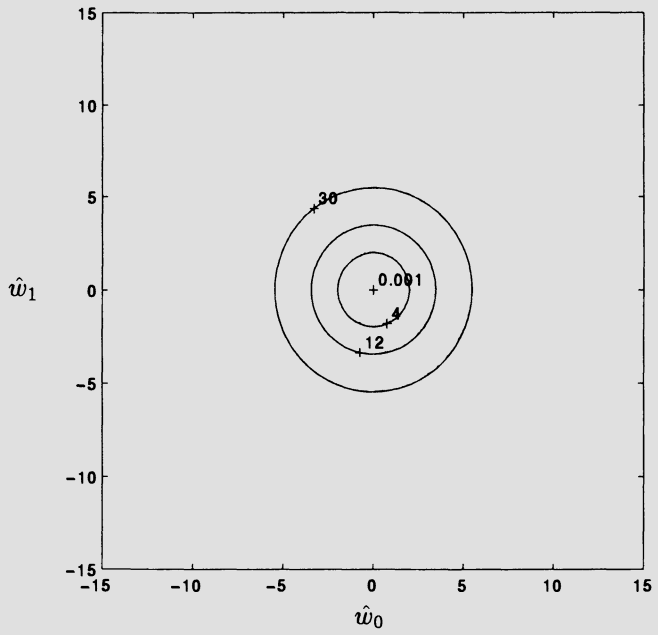
It can be easily shown that if μ is chosen appropriately, the adaptive filter coefficients converge to

$$\hat{\mathbf{w}}_o = \mathbf{R}_s^{-1}\mathbf{p}_s \tag{4.69}$$

where $\mathbf{R}_s = \mathbf{T}\mathbf{R}\mathbf{T}^T$ and $\mathbf{p}_s = \mathbf{T}\mathbf{p}$. As a consequence, the optimum coefficient vector is

$$\hat{\mathbf{w}}_o = (\mathbf{T}\mathbf{R}\mathbf{T}^T)^{-1}\mathbf{T}\mathbf{p} = \mathbf{T}\mathbf{R}^{-1}\mathbf{p} = \mathbf{T}\mathbf{w}_o \tag{4.70}$$

The convergence speed of the coefficient vector $\hat{\mathbf{w}}(k)$ is determined by the eigenvalue spread of $\sigma^{-2}(k)\mathbf{R}_s$.



$$\mathbf{TR} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Figure 4.8 Contours of the rotated and power normalized MSE surface.

The requirements on the transformation matrix are that it should be invertible. If the matrix $\mathbf{T}$ is not square (number of columns larger than rows), the space spanned by the polynomials formed with the rows of $\mathbf{T}$ will be of dimension $N+1$, but these polynomials are of order larger than N. This subspace does not contain the complete space of polynomials of order N. In general, except for very specific desired signals, the entire space of Nth-order polynomials would be required. For an invertible matrix $\mathbf{T}$ there is one-to-one correspondence between the solutions obtained by the LMS and transform-domain LMS algorithms. Although the transformation matrix is not required to be unitary, it appears that no advantages are obtained by using nonunitary transform [12].

The best unitary transform for the transform-domain adaptive filter is the KLT. However, the KLT is a function of the input signal, it cannot be efficiently computed in real time. An alternative is to choose a unitary transform that is close to the KLT of the particular input signal. By close is meant that both transforms perform nearly the same rotation of the MSE surface. In any situation, the choice of an appropriate transform is not an easy task. Some guidelines can be given, such as: i) Since the KLT of a real signal is real, the chosen transform should be real for real input signals; ii) For speech signals the discrete-time cosine transform (DCT) is a good approximation for the KLT [17]; iii) Transforms with fast algorithms should be given special attention.

A number of real transforms such as DCT, discrete-time Hartley transform, and others, are available [17]. Most of them have fast algorithms or can be implemented in recursive frequency-domain format. In particular, the outputs of the DCT are given by

$$s_0(k) = \frac{1}{\sqrt{N+1}} \sum_{l=0}^{N} x(k-l) \tag{4.71}$$

and

$$s_i(k) = \sqrt{\frac{2}{N+1}} \sum_{l=0}^{N} x(k-l) \cos\left[\pi i \frac{(2l+1)}{2(N+1)}\right] \tag{4.72}$$

From Fig. 4.3, we observe that the delay line and the unitary transform form a single-input and multiple-output preprocessing filter. In case the unitary transform is the DCT, the transfer function from the input to the outputs of the DCT preprocessing filter can be described in a recursive format as follows:

$$T_i(z) = \frac{k_0}{N+1} \cos\tau \frac{[z^{N+1} - (-1)^i](z-1)}{z^N[z^2 - (2\cos 2\tau)z + 1]} \tag{4.73}$$

where

$$k_0 = \begin{cases} \sqrt{2} & if \quad i = 0 \\ 2 & if \quad i = 1, ..., N \end{cases}$$

and $\tau = \frac{i}{2(N+1)}$.

The derivation details are not given here, since they are beyond the scope of this text.

For complex input signals, the discrete-time Fourier transform (DFT) is a natural choice due to its efficient implementations.

Algorithm 4.4

The Transform-Domain LMS Algorithm

Initialization

$\mathbf{x}(0) = \hat{\mathbf{w}}(0) = [0...0]^T$

γ = small constant

$0 < \alpha \leq 0.1$

Do for each $x(k)$ and $d(k)$ given for $k \geq 0$

$\mathbf{s}(k) = \mathbf{T}\mathbf{x}(k)$

$e(k) = d(k) - \mathbf{s}^T(k)\hat{\mathbf{w}}(k)$

$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + 2\,\mu\, e(k)\, \boldsymbol{\sigma}^{-2}(k)\mathbf{s}(k)$

□

Although no general procedure is available to choose the best transform when the input signal is not known *a priori*, the decorrelation performed by the transform, followed by the power normalization, is sufficient to reduce the eigenvalue spread for a broad class (not all) of input signals. Therefore, the transform-domain LMS filters are expected to converge faster than the standard LMS algorithm in most applications [12].

The complete transform-domain LMS algorithm is outlined on Algorithm 4.4.

Example 4.2

Repeat the equalization problem of example 3.1 of the previous chapter using the transform-domain algorithm.

(a) Compute the Wiener solution.

(b) Choose an appropriate value for μ and plot the convergence path for the transform-domain algorithm on the MSE surface.

Solution:

(a) In this example, the correlation matrix of the adaptive filter input signal is given by

$$\mathbf{R} = \begin{bmatrix} 1.6873 & -0.7937 \\ -0.7937 & 1.6873 \end{bmatrix}$$

and the cross-correlation vector $\mathbf{p}$ is

$$\mathbf{p} = \begin{bmatrix} 0.9524 \\ 0.4762 \end{bmatrix}$$

For square matrix $\mathbf{R}$ of dimension 2, the transformation matrix corresponding to the cosine transform is given by

$$\mathbf{T} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$$

For this filter order, the transformation matrix above coincides with the KLT.

The coefficients corresponding to the Wiener solution of the transform-domain filter are given by

$$\begin{aligned} \hat{\mathbf{w}}_o &= (\mathbf{TRT}^T)^{-1}\mathbf{Tp} \\ &= \begin{bmatrix} \frac{1}{0.8936} & 0 \\ 0 & \frac{1}{2.4810} \end{bmatrix} \begin{bmatrix} 1.0102 \\ 0.3367 \end{bmatrix} \\ &= \begin{bmatrix} 1.1305 \\ 0.1357 \end{bmatrix} \end{aligned}$$

(b) The transform-domain algorithm was applied to minimize the MSE using a small convergence factor $\mu = 1/300$, in order to obtain a smoothly converging curve. The convergence path of the algorithm in the MSE surface is depicted in Fig. 4.9. As can be noted, the transformation aligned the coefficient axes with the main axes of the ellipses belonging to the error surface. The reader should notice that the algorithm follows an almost straight path to the minimum and that the effect of the eigenvalue spread is compensated by the power normalization. The convergence in this case is faster than for the LMS case.

□

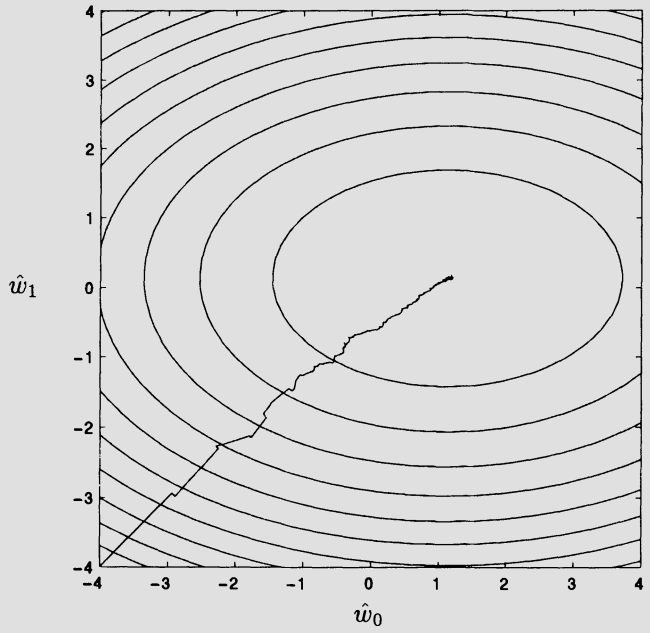


Figure 4.9 Convergence path of the transform-domain adaptive filter.

From the transform-domain algorithm point of view, we can consider that the LMS-Newton algorithm attempts to utilize an estimate of the KLT through $\hat{\mathbf{R}}^{-1}(k)$. On the other hand, the normalized LMS algorithm utilizes an identity transform with an instantaneous estimate of the input signal power given by $\mathbf{x}^T(k)\mathbf{x}(k)$.

4.6 SIMULATION EXAMPLES

In this section, some adaptive filtering problems are described and solved by using some of the algorithms presented in this chapter.

Example 4.3: Transform-Domain Algorithms

Use the transform-domain LMS algorithm to identify the system described in the example 3.4 of the previous chapter. The transform is the DCT.

Solution:

All the results presented here for the transform-domain LMS algorithm were obtained by averaging the results of 200 independent runs.

We run the algorithm with a value of $\mu = 0.01$, with $\alpha = 0.05$ and $\gamma = 10^{-6}$. With this value of μ, the misadjustment of the transform-domain is about the same as that of the LMS algorithm with $\mu = 0.02$. In Fig. 4.10, the learning curves for the eigenvalue spreads 20 and 80 are illustrated. First note that the convergence speed is about the same for different eigenvalue spreads, showing the effectiveness of the rotation performed by the transform in this case. If we compare these curves with those of Fig. 3.9 for the LMS algorithm, we conclude that the transform-domain algorithm has better performance than the LMS algorithm for high eigenvalue spread. For an eigenvalue spread equal to 20, the transform-domain algorithm required around 200 iterations to achieve convergence, whereas the LMS required at least 500 iterations. This improvement is achieved without increasing the misadjustment as can be verified by comparing the results of Tables 3.1 and 4.1.

The reader should bear in mind that the improvements in convergence of the transform-domain algorithm can be achieved only if the transformation is effective. In this example, since the input signal was colored using a first-order all-pole filter, the cosine transform is known to be effective because it approximates the KLT.

The finite-precision implementation of the transform-domain LMS algorithm presents similar performance to that of the LMS algorithm, as can be verified by comparing the results of Tables 3.2 and 4.2. An eigenvalue spread of one was used in this example. The value of μ was 0.01, while the remaining parameter values were $\gamma = 2^{-b_d}$ and $\alpha = 0.05$. The value of μ in this case was chosen the same as for the LMS algorithm.

□

Table 4.1 Evaluation of the Transform-Domain LMS Algorithm

$\frac{\lambda_{max}}{\lambda_{min}}$	Misadjustment
1	0.2027
20	0.2037
80	0.2093

Table 4.2 Results of the Finite-Precision Implementation of the Transform-Domain LMS Algorithm

	$\xi(k)_Q$	$E[\|\|\Delta\mathbf{w}(k)_Q\|\|^2]$
No of bits	Experiment	Experiment
16	$1.627\ 10^{-3}$	$1.313\ 10^{-4}$
12	$1.640\ 10^{-3}$	$1.409\ 10^{-4}$
10	$1.648\ 10^{-3}$	$1.536\ 10^{-4}$

4.6.1 Signal Enhancement Simulation

In this subsection a signal enhancement simulation environment is described. This example will also be employed in some of the following chapters.

In a signal enhancement problem, the reference signal is

$$r(k) = \sin(0.2\pi k) + n_r(k)$$

where $n_r(k)$ is a zero-mean Gaussian white noise with variance $\sigma^2_{n_r} = 10$. The input signal is given by $n_r(k)$ passed through a filter with the following transfer function

$$H(z) = \frac{0.4}{z^2 - 1.36z + 0.79}$$

The adaptive filter is a 20th-order FIR filter. In all examples, a delay $L = 10$ was applied to the reference signal.

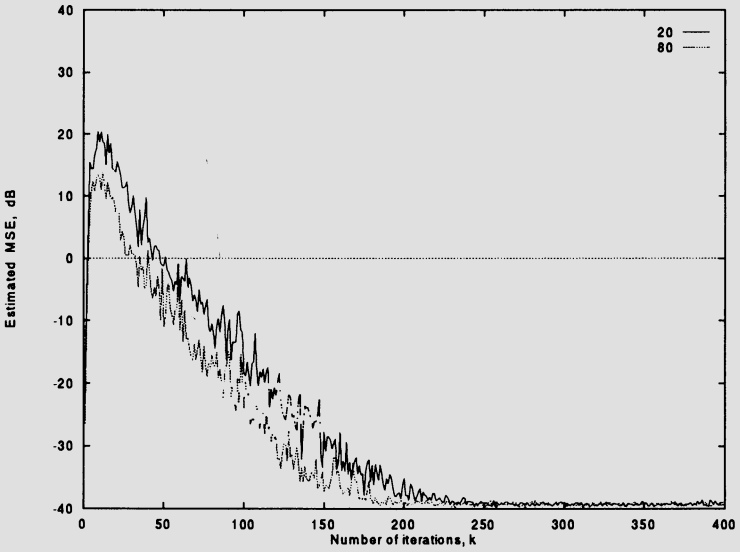


Figure 4.10 Learning curves for the transform-domain LMS algorithm for eigenvalue spreads: 20 and 80.

Example 4.4: LMS-Based Algorithms

Using the sign-error, power-of-two error with $b_d = 12$, and normalized LMS algorithms:
(a) Choose an appropriate μ in each case and run an ensemble of 50 experiments. Plot the average learning curve.
(b) Plot the output errors and comment on the results.

Solution:

The maximum value of μ for the LMS algorithm in this example is 0.005. The values of μ for the sign-error and power-of-two LMS algorithms were chosen 0.001. The coefficients of the adaptive filter were initialized with zero. For the normalized LMS algorithm $\mu_n = 0.4$ and $\gamma = 10^{-6}$ were used. Fig. 4.11 depicts the learning curves for the three algorithms. The results show that the sign-error and power-of-two error algorithms present a similar convergence speed, whereas the normalized LMS was faster to converge. The reader should notice that the MSE after convergence is not small since we are dealing with an example where the signal to noise ratio is low.

The DFT with 128 points of the input signal is shown in Fig. 4.12 where the presence of the sinusoid cannot be noted. In the same figure the DFT of the error and the error signal itself for the experiment using the normalized LMS algorithm are shown. In the cases of DFT, the result presented is the magnitude of the DFT outputs. As can be verified, the output error tends to produce a signal with the same period of the sinusoid after convergence and the DFT shows clearly the presence of the sinusoid. The other two algorithms lead to similar results.

□

4.6.2 Signal Prediction Simulation

In this subsection a signal prediction simulation environment is described. This example will also be used in some of the following chapters.

In a prediction problem the input signal is

$$x(k) = -\sqrt{2}\ \sin(0.2\pi k) + \sqrt{2}\ \sin(0.05\pi k) + n_x(k)$$

where $n_x(k)$ is a zero-mean Gaussian white noise with variance $\sigma^2_{n_x} = 1$. The adaptive filter is a fourth-order FIR filter.
(a) Run an ensemble of 50 experiments and plot the average learning curve.
(b) Determine the zeros of the resulting FIR filter and comment on the results.

Example 4.5: LMS-Based Algorithms

We solved the problem above using the sign-error, power-of-two error with $b_d = 12$, and normalized LMS algorithms.

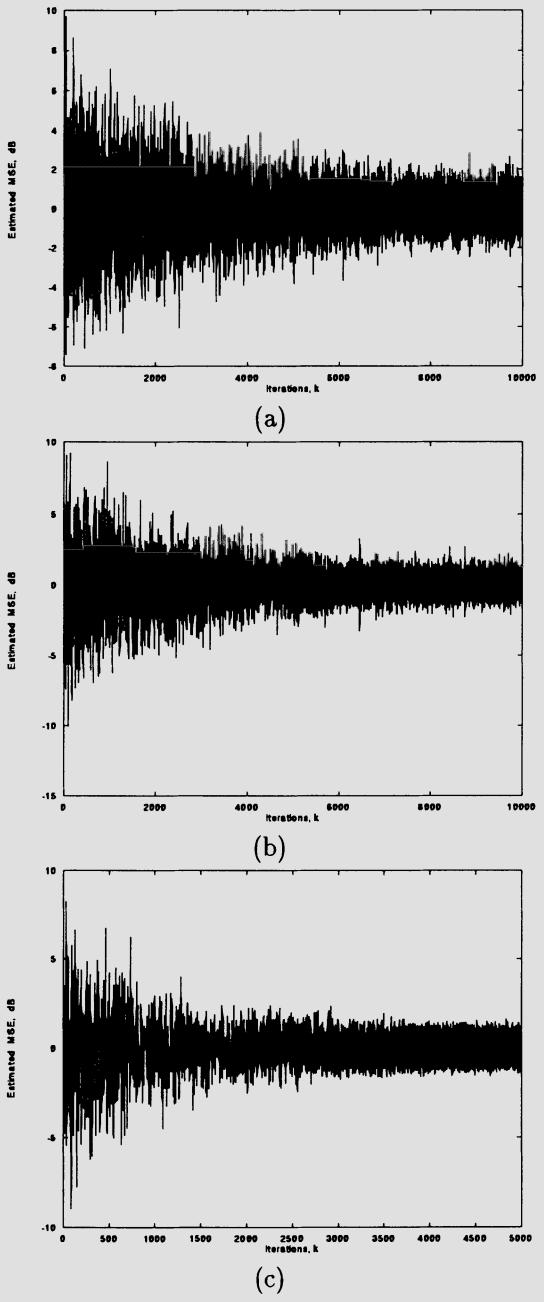


Figure 4.11 Learning curves for the (a) Sign-error, (b) Power-of-two, and (c) Normalized LMS algorithms.

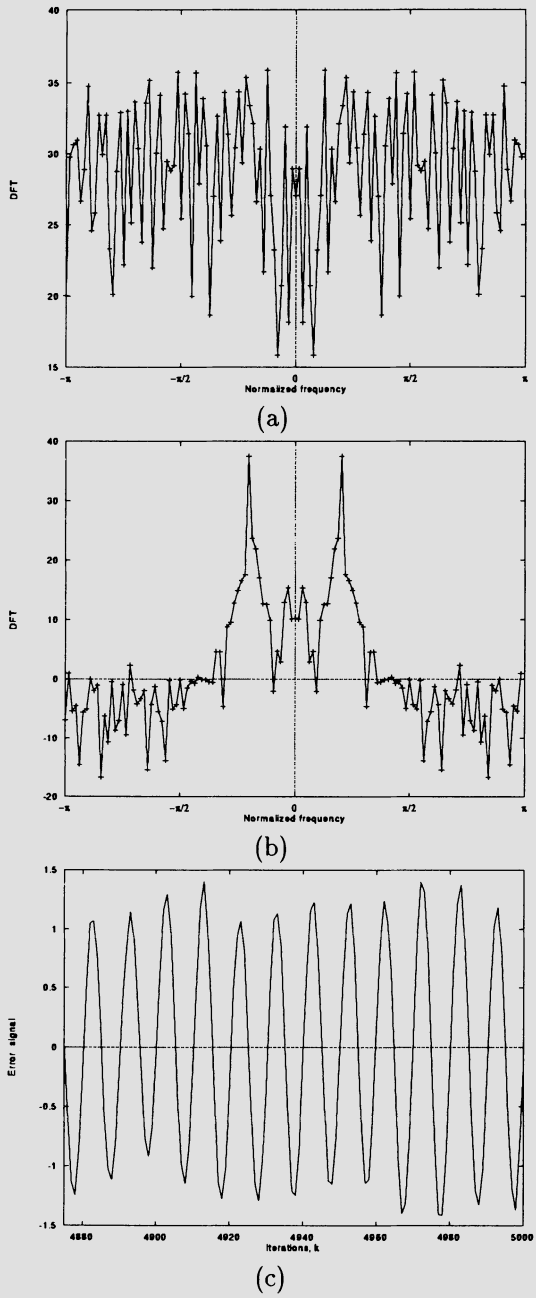


Figure 4.12 (a) DFT of the input signal, (b) DFT of the error signal, (c) The output error for the normalized LMS algorithm.

Solution:

In the first step, each algorithm was tested in order to determine experimentally the maximum value of μ in which the convergence was achieved. The choice of the convergence factor was $\mu_{max}/5$ for each algorithm. The values of μ for the sign-error and power-of-two LMS algorithms were chosen 0.0028 and 0.0044, respectively. For the normalized LMS algorithm, $\mu_n = 0.4$ and $\gamma = 10^{-6}$ were used. The coefficients of the adaptive filter were initialized with zero. In all cases, we noticed a strong attenuation of the predictor response around the frequencies of the two sinusoids. See, for example, the response depicted in Fig. 4.13 obtained by running the power-of-two LMS algorithm. The learning curves for the three algorithms are depicted in Fig. 4.14. The zeros of the transfer function from the input to the output error were calculated for the power-of-two algorithm:

$$-0.3939;\ -0.2351 \pm j0.3876;\ -0.6766 \pm j0.3422$$

Notice that the predictor tends to place its zeros at low frequencies, in order to attenuate the two low-frequency sinusoids.

In the experiments, we noticed that for a given additional noise, smaller convergence factor leads to higher attenuation at the sinusoid frequencies. This is an expected result since the excess of the MSE is smaller. Another observation is that the attenuation also grows as the signal to noise ratio is reduced, again due to the smaller MSE.

□

4.7 CONCLUDING REMARKS

In this chapter, a number of adaptive filtering algorithms were presented derived from the LMS algorithm. There were two basic directions followed in the derivation of the algorithms: one direction is to search for simpler algorithms from the computational point of view, and the other is to sophisticate the LMS algorithm looking for improvements in performance. The simplified algorithms lead to low-power, low-complexity and/or high-speed integrated circuit implementations [20], at a cost of increasing the misadjustment and/or of losing convergence speed among other things [21]. The simplified algorithms discussed

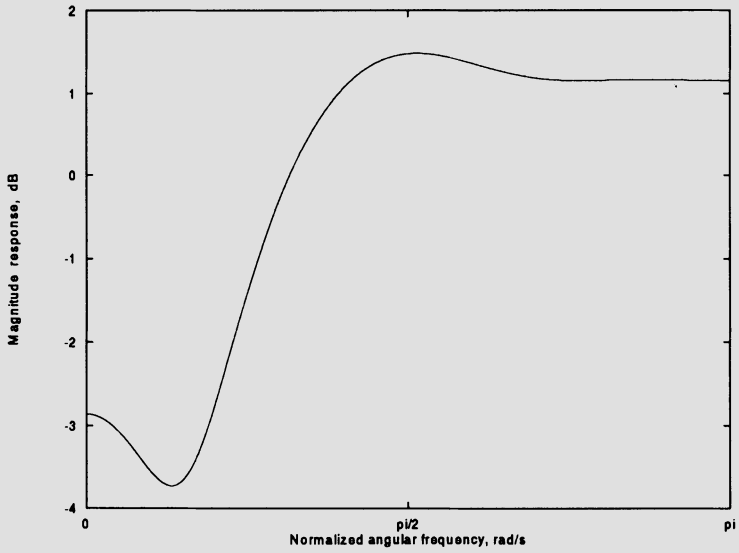


Figure 4.13 Magnitude response of the FIR adaptive filter at a given iteration after convergence using the power-of-two LMS algorithm.

here are the quantized-error algorithms. There are several other algorithms related to the LMS algorithm that were not presented here, and the interested reader can refer, for example, to [22]-[23].

We also introduced the LMS-Newton algorithm, whose performance is independent of the eigenvalue spread of the input-signal correlation matrix. This algorithm is related to the RLS algorithm which will be discussed in the following chapter, although some distinctive features exist between them [32]. Newton-type algorithms with reduced computational complexity are also known [33]-[34], and the main characteristic of this class of algorithms is to reduce the computation involving the inverse of the estimate of $\mathbf{R}$.

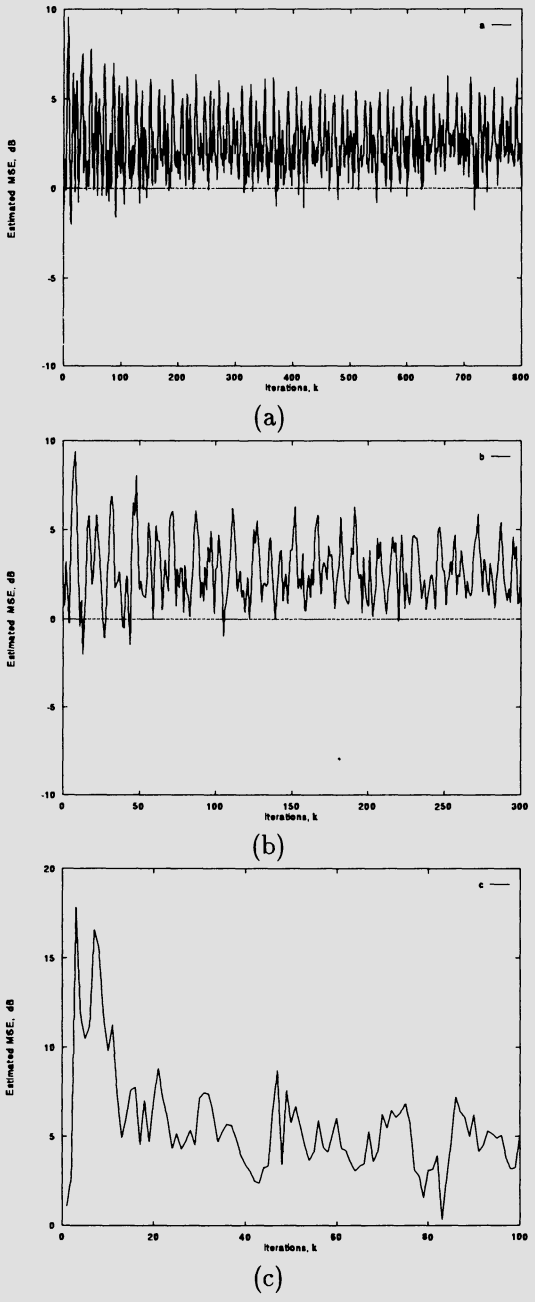


Figure 4.14 Learning curves for the (a) Sign-error, (b) Power-of-two, and (c) Normalized LMS algorithms.

In the normalized LMS algorithm, the straightforward objective was to find the step size that minimizes the instantaneous output error. There are many papers dealing with the analysis [24]-[26] and applications [27] of the normalized LMS algorithm. The idea of using variable step size in the LMS and normalized LMS algorithms can lead to a number of interesting algorithms [28]-[30], that in some cases are very efficient in tracking nonstationary environments [31].

The transform-domain algorithm aimed to reduce the eigenvalue spread of the input signal correlation matrix. Several frequency-domain adaptive algorithms, which are related in some sense to the transform-domain algorithm, have been investigated in the recent years [35]. Such algorithms exploit the whitening property associated with the normalized transform-domain algorithm, and most of them update the coefficients at a rate lower than the input sampling rate. One of the resulting structures, presented in [36], can be interpreted as a direct generalization of the transform-domain LMS algorithm and is called generalized adaptive subband decomposition structure. Such structure consists of a small-size fixed transform, which is applied to the input sequence, followed by sparse adaptive subfilters which are updated at the input rate. In high-order adaptive filtering problems, the use of this structure with appropriately chosen transform-size and sparsity factor can lead to significant convergence rate improvement for colored input signals when compared to the standard LMS algorithm. The convergence rate improvement is achieved without the need for large transform sizes.

Frequency-domain adaptive filtering algorithms which employ block processing in order to reduce the computational complexity associated with high-order adaptive filters have also been suggested [37]. Such algorithms utilize FFTs to implement convolutions (for filtering) and correlations (for coefficient updating). More general block algorithms, in which the block size can be smaller than the order of the adaptive filter, have also been investigated [38].

Other adaptive filtering structures based on multirate techniques have been suggested in [39]-[43]. In such structures, the input signal is decomposed in subbands by an analysis filter bank, and the resulting signals are downsampled and filtered by adaptive filters. Each of these adaptive filters has order smaller than the equivalent full-band adaptive filter (by a factor of approximately the downsampling factor). Several adaptive subband structures have been suggested. One early approach used pseudo-QMF banks with overlapping filter banks and critical subsampling [39], resulting in the appearance of undesirable aliased components in the output, which caused severe degradation. A second approach used QMF banks with critical subsampling [40] and, in order to avoid aliasing problems resulting from the channel modifications, suggested the use

of additional adaptive cross terms between the subbands. These cross terms, however, increase the computational complexity and reduce the convergence rate of the adaptive algorithm. Another approach used an auxiliary, non-decimated channel [41], which also resulted in increased complexity and was shown to be useful particularly for the adaptive line enhancement applications. A last approach [42]-[43] employed a reduction in the sampling rate of the filtered signals by a factor smaller than the critical subsampling factor (or number of bands) to avoid aliasing problems, with a consequent increase in the computational complexity. In all of the subband structures described above, the convergence rate can be improved for colored input signals by using a normalized gradient algorithm in the update of the coefficients of each of the subband filters.

Several simulation examples involving the LMS-based algorithms were presented in this chapter. These examples aid the reader to understand what are the main practical characteristics of the LMS-based algorithms.

References

1. T. A. C. M. Claasen and W. F. G. Mecklenbräuker, "Comparison of the convergence of two algorithms for adaptive FIR filters," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-29, pp. 670-678, June 1981.

2. N. A. M. Verhoeckx and T. A. C. M. Claasen, "Some considerations on the design of adaptive digital filters equipped with the sign algorithm," *IEEE Trans. on Communications*, vol. COM-32, pp. 258-266, March 1984.

3. N. J. Bershad, "Comments on 'Comparison of the convergence of two algorithms for adaptive FIR digital filters'," *IEEE Trans. on Acoust., Speech, and Signal Processing* , vol. ASSP-33, pp. 1604-1606, Dec. 1985.

4. P. Xue and B. Liu, "Adaptive equalizer using finite-bit power-of-two quantizer," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-34, pp. 1603-1611, Dec. 1986.

5. V. J. Mathews and S. H. Cho, "Improved convergence analysis of stochastic gradient adaptive filters using the sign algorithm," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-35, pp. 450-454, April 1987.

6. W. A. Sethares and C. R. Johnson, Jr., "A comparison of two quantized state adaptive algorithms," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-37, pp. 138-143, Jan. 1989.

7. V. J. Mathews, "Performance analysis of adaptive filters equipped with dual sign algorithm," *IEEE Trans. on Signal Processing*, vol. 39, pp. 85-91, Jan. 1991.

8. E. Eweda, "Convergence analysis and design of an adaptive filter with finite-bit power-of-two quantizer error," *IEEE Trans. on Circuits and Systems II : Analog and Digital Signal Processing*, vol. 39, pp. 113-115, Feb. 1992.

9. W. A. Sethares, I. M. X. Mareels, B. D. O. Anderson, C. R. Johnson, Jr., and R. R. Bitmead, "Excitation conditions for signed regressor least mean square adaptation," *IEEE Trans. on Circuits and Systems*, vol. 35, pp. 613-624, June 1988.

10. S. H. Cho and V. J. Mathews, "Tracking analysis of the sign algorithm in nonstationary environments," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. 38, pp. 2046-2057, Dec. 1990.

11. S. S. Narayan, A. M. Peterson, and M. J. Narasimha, "Transform domain LMS algorithm," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-31, pp. 609-615, June 1983.

12. D. F. Marshall, W. K. Jenkins and J. J. Murphy, "The use of orthogonal transform for improving performance of adaptive filters, " *IEEE Trans. on Circuits and Systems*, vol. 36, pp. 474-484, April 1989.

13. J. C. Lee and C. K. Un, "Performance of transform-domain LMS adaptive digital filters," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-34, pp. 499-510, June 1986.

14. F. F. Yassa, "Optimality in the choice of convergence factor for gradient based adaptive algorithms," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-35, pp. 48-59, Jan. 1987.

15. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1985.

16. P. S. R. Diniz and L. W. Biscainho, "Optimal variable step size for the LMS/Newton algorithm with application to subband adaptive filtering," *IEEE Trans. on Signal Processing*, vol. SP-40, pp. 2825-2829, Nov. 1992.

17. N. S. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*, Prentice Hall, Englewood Cliffs, NJ, 1984.

18. R. Price, "A useful theorem for nonlinear devices having Gaussian inputs," *IRE Trans. on Information Theory*, vol. IT-4, pp. 69-72, June 1958.

19. A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 3rd edition, McGraw-Hill, New York, NY, 1991.

20. H. Samueli, B. Daneshrad, R. B. Joshi, B. C. Wong, and H. T. Nicholas, III, "A 64-tap CMOS echo canceller/decision feedback equalizer for 2B1Q HDSL ," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 839-847, Aug. 1991.

21. C. R. Johnson, Jr., *Lectures on Adaptive Parameter Estimation*, Prentice Hall, Englewood Cliffs, NJ, 1988.

22. S. Roy and J. J. Shynk, "Analysis of the data-reusing LMS algorithm ," *Proc. Midwest Symposium on Circuits and Systems*, Urbana, IL, pp. 1127-1130, Aug. 1989.

23. J. J. Shynk and S. Roy "Analysis of the momentum updating LMS algorithm ," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. 38, pp. 2088-2098, Dec. 1990.

24. D. T. Slock, "On the convergence behavior of the LMS and normalized LMS algorithms," *IEEE Trans. on Signal Processing*, vol-40, pp. 2811-2825, Sept. 1993.

25. N. J. Bershad, "Analysis of the normalized LMS algorithm with Gaussian inputs," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-34, pp. 793-806, Aug. 1986.

26. M. Tarrab and A. Feuer, "Convergence and performance analysis of the normalized LMS algorithm with uncorrelated Gaussian data," *IEEE Trans. on Information Theory*, vol. IT-34, pp. 680-691, July 1988.

27. J. F. Doherty, "An adaptive algorithm for stable decision-feedback filtering," *IEEE Trans. on Circuits and Systems–II: Analog and Digital Signal Processing*, vol. 40, pp. 1-8, Jan. 1993.

28. W. B. Mikhael, F. H. Fu, L. G. Kazovsky, G. S. Kang, and L. J. Fransen, "Adaptive filter with individual adaptation of parameters ," *IEEE Trans. on Circuits and Systems* , vol. 33, pp. 677-686, July 1986.

29. R. W. Harris, D. M. Chabries, and F. A. Bishop, "A variable step (VS) adaptive filter algorithm," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-34, pp. 309-316, April 1986.

30. C. S. Modlin and J. M. Cioffi, "A fast decision feedback LMS algorithm using multiple step sizes," *Proc. IEEE Inter. Conf. on Communications*, New Orleans, pp. 1201-1205, May 1994.

31. S. D. Peters and A. Antoniou, "Environment estimation for enhanced NLMS adaptation ," *Proc. IEEE Pac. Rim Conf. on Comm., Comp. and Sig. Proc.*, Victoria, Canada, pp. 342-345, May 1993.

32. P. S. R. Diniz, M. L. R. de Campos, and A. Antoniou, "Analysis of LMS-Newton adaptive filtering algorithms with variable convergence factor," *IEEE Trans. on Signal Processing*, vol. 43, pp. 617-627, March 1995.

33. D. F. Marshall and W. K. Jenkins, "A fast quasi-Newton adaptive filtering algorithm," *IEEE Trans. on Signal Processing*, vol-40, pp. 1652-1662, July 1993.

34. G. V. Moustakides and S. Theodoridis, "Fast Newton transversal filters - A new class of adaptive estimation algorithm," *IEEE Trans. on Signal Processing*, vol-39, pp. 2184-2193, Oct. 1991.

35. J. J. Shynk, "Frequency-domain and multirate adaptive filtering," *IEEE Signal Processing Magazine*, pp. 15-37, Jan. 1992.

36. M. R. Petraglia and S. K. Mitra,"Adaptive FIR filter structure based on the generalized subband decomposition of FIR filters," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 40, pp. 354-362, June 1993.

37. G. A. Clark, S. R. Parker, and S. K. Mitra, "A unified approach to time- and frequency-domain realization of FIR adaptive digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1073-1083, Oct. 1983.

38. P. C. Sommen, "On the convergence properties of a partitioned block frequency domain adaptive filter (PBFDAF)," *Proc. European Signal Processing Conf.*, pp. 201-203, Barcelona, Spain, Sept. 1990.

39. A. Gilloire, "Experiments with sub-band acoustic echo cancellers for teleconferencing," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 2141-2144, Dallas, TX, April 1987.

40. A. Gilloire and M. Vetterli, "Adaptive filtering in subbands with critical sampling: analysis, experiments, and application to acoustic echo cancellation," *IEEE Trans. Signal Processing*, vol. 40, pp. 1862-1875, Aug. 1992.

41. V. S. Somayazulu, S. K. Mitra, and J. J. Shynk, "Adaptive line enhancement using multirate techniques," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 928-931, Glasgow, Scotland, April 1989.

42. W. Kellermann, "Analysis and design of multirate systems for cancellation of acoustical echoes," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 2570-2573, New York, NY, April 1988.

43. M. R. Petraglia and S. K. Mitra, "Performance analysis of adaptive filter structures based on subband decomposition," *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 60-63, Chicago, IL, May 1993.

Problems

1. From equation (4.16) derive the difference equation for $\mathbf{v}'(k)$ given by equation (4.19).

2. Prove the validity of equation (4.27).

3. The sign-error algorithm is used to predict the signal $x(k) = \sin(\pi k/3)$ using a second-order FIR filter with the first tap fixed at 1, by minimizing the MSE of $y^2(k)$. This is an alternative way to interpret how the predictor works. Calculate an appropriate μ, the output signal $y(k)$, and the filter coefficients for the first 10 iterations. Start with $\mathbf{w}^T(0) = [1\ 0\ 0]$.

4. Use the sign-error algorithm to identify a system with the transfer function given below. The input signal is a uniformly distributed white noise with variance $\sigma_x^2 = 1$, and the measurement noise is a Gaussian white noise uncorrelated with the input with variance $\sigma_n^2 = 10^{-3}$. The adaptive filter has 12 coefficients.

$$H(z) = \frac{1 - z^{-12}}{1 + z^{-1}}$$

(a) Calculate the upper bound for μ (μ_{max}) to guarantee the algorithm stability.
(b) Run the algorithm for $\mu_{max}/2$, $\mu_{max}/5$, and $\mu_{max}/10$. Comment on the convergence behavior in each case.
(c) Measure the misadjustment in each example and compare with the results obtained by the equation (4.28).
(d) Plot the obtained FIR filter frequency response in any iteration after convergence is achieved and compare with the unknown system.

5. Repeat the previous problem using an adaptive filter with 8 coefficients and interpret the results.

6. Repeat problem 4 when the input signal is a uniformly distributed white noise with variance $\sigma_x^2 = 0.5$, filtered by an all-pole filter given by

$$H(z) = \frac{z}{z - 0.9}$$

7. In problem 4, consider that the additional noise has the following variances (a) $\sigma_n^2 = 0$, (b) $\sigma_n^2 = 1$. Comment on the results obtained in each case.

8. Perform the equalization of a channel with the following impulse response

$$h(k) = ku(k) - (2k - 9)u(k - 5) + (k - 9)u(k - 10)$$

using a known training signal consisting of a binary (-1,1) random signal. An additional Gaussian white noise with variance 10^{-2} is present at the channel output.
(a) Apply the sign-error with an appropriate μ and find the impulse response of an equalizer with 15 coefficients.
(b) Convolve the equalizer impulse response at an iteration after convergence, with the channel impulse response and comment on the result.

9. In a system identification problem, the input signal is generated by an autoregressive process given by

$$x(k) = -1.2x(k - 1) - 0.81x(k - 2) + n_x(k)$$

where $n_x(k)$ is a zero-mean Gaussian white noise with variance such that $\sigma_x^2 = 1$. The unknown system is described by

$$H(z) = 1 + 0.9z^{-1} + 0.1z^{-2} + 0.2z^{-3}$$

The adaptive filter is also a third-order FIR filter. Using the sign-error algorithm:
(a) Choose an appropriate μ, run an ensemble of 20 experiments, and plot the average learning curve.
(b) Measure the excess of MSE and compare the results with the theoretical value.
(c) Compare the measured and theoretical value for the misadjustment.

10. In the previous problem, calculate the time constant τ_{wi} and the expected number of iterations to achieve convergence.

11. The sign-error algorithm was applied to identify a 7th-order time-varying unknown system whose coefficients are first-order Markov processes with $\lambda_{\mathbf{W}} = 0.999$ and $\sigma^2_{\mathbf{W}} = 0.001$. The initial time-varying system multiplier coefficients are

$$\mathbf{w}_o^T = [0.03490 \; -0.011 \; -0.06864 \; 0.22391 \; 0.55686 \; 0.35798 \; -0.0239 \; -0.07594]$$

The input signal is a Gaussian white noise with variance $\sigma_x^2 = 0.7$, and the measurement noise is also a Gaussian noise independent of the input signal and of the elements of $\mathbf{n_W}(k)$, with variance $\sigma_n^2 = 0.01$.
For $\mu = 0.01$, simulate the experiment described and measure the excess MSE.

12. Reduce the value of $\lambda_{\mathbf{W}}$ to 0.95 in the problem 11, simulate, and comment on the results.

13. Suppose a 15th-order filter FIR digital filter with multiplier coefficients given below, was identified through an adaptive FIR filter of the same order using the sign-error algorithm. Use fixed-point arithmetic and run simulations for the following case.

Additional noise: white noise with variance	$\sigma_n^2 = 0.0015$
Coefficient wordlength:	$b_c = 16$ bits
Signal wordlength:	$b_d = 16$ bits
Input signal: Gaussian white noise with variance	$\sigma_x^2 = 0.7$
	$\mu = 0.01$

$\mathbf{w}_o^T$ = [0.0219360 0.0015786 − 0.0602449 − 0.0118907 0.1375379 0.0574545 − 0.3216703 − 0.5287203 − 0.2957797 0.0002043 0.290670 −0.0353349 − 0.068210 0.0026067 0.0010333 − 0.0143593]

Plot the learning curves of the estimates of $E[||\Delta\mathbf{w}(k)_Q||]$ and $\xi(k)_Q$ obtained through 25 independent runs, for the finite- and infinite-precision implementations.

14. Repeat the problem above for the following cases
(a) $\sigma_n^2 = 0.01$, $b_c = 12$ bits, $b_d = 12$ bits, $\sigma_x^2 = 0.7$, $\mu = 10^{-4}$.
(b) $\sigma_n^2 = 0.1$, $b_c = 10$ bits, $b_d = 10$ bits, $\sigma_x^2 = 0.8$, $\mu = 2.0 \; 10^{-5}$.
(c) $\sigma_n^2 = 0.05$, $b_c = 14$ bits, $b_d = 16$ bits, $\sigma_x^2 = 0.8$, $\mu = 3.5 \; 10^{-4}$.

15. Repeat problem 13 in the case the input signal is a first-order Markov process with $\lambda_{\mathbf{X}} = 0.95$.

16. Repeat problem 4 for the dual-sign algorithm given $\gamma = 16$ and $\rho = 1$, and comment on the results.

17. Repeat problem 4 for the power-of-two error algorithm given $b_d = 6$ and $\tau = 2^{-b_d+1}$, and comment on the results.

18. Repeat problem 4 for the sign-data and sign-sign algorithms and compare the results.

19. Show the validity of the matrix inversion lemma defined in equation (4.51).

20. For the setup described in problem 6, choose an appropriate μ and run the LMS-Newton algorithm.
    (a) Measure the misadjustment.
    (b) Plot the frequency response of the FIR filter obtained after convergence is achieved and compare with the unknown system.

21. Repeat problem 6 using the normalized LMS algorithm.

22. Repeat problem 6 using the transform-domain LMS algorithm with DCT. Compare the results with those obtained with the standard LMS algorithm.

23. For the input signal described in problem 6, derive the autocorrelation matrix of order one (2×2). Apply the DCT and the normalization to $\mathbf{R}$ in order to generate $\hat{\mathbf{R}} = \boldsymbol{\sigma}^{-2}\mathbf{T}\mathbf{R}\mathbf{T}^T$. Compare the eigenvalue spreads of $\mathbf{R}$ and $\hat{\mathbf{R}}$.

24. Repeat the previous problem for $\mathbf{R}$ with dimension 3 by 3.

5

CONVENTIONAL RLS ADAPTIVE FILTER

5.1 INTRODUCTION

Least-squares algorithms aim at the minimization of the sum of the squares of the difference between the desired signal and the model filter output [1]-[2]. When new samples of the incoming signals are received at every iteration, the solution for the least-squares problem can be computed in recursive form resulting in the recursive least-squares (RLS) algorithms. The conventional version of these algorithms will be the topic of the present chapter.

The RLS algorithms are known to pursue fast convergence even when the eigenvalue spread of the input signal correlation matrix is large. These algorithms have excellent performance when working in time-varying environments. All these advantages come with the cost of an increased computational complexity and some stability problems, which are not as critical in LMS-based algorithms [3]-[4].

Several properties related to the RLS algorithms are discussed including misadjustment, tracking behavior, and finite-wordlength effects [3]-[10].

5.2 THE RECURSIVE LEAST-SQUARES ALGORITHM

The objective here is to choose the coefficients of the adaptive filter such that the output signal $y(k)$, during the period of observation, will match the desired

P. S. R. Diniz, *Adaptive Filtering*

signal as closely as possible in the least-squares sense. The minimization process requires the information of the input signal available so far. Also, the objective function we seek to minimize is deterministic.

The generic FIR adaptive filter realized in the direct form is shown in Fig 5.1. The input-signal information vector at a given instant k is given by

$$\mathbf{x}(k) = [x(k)\ x(k-1)\ \ldots x(k-N)]^T \tag{5.1}$$

where N is the order of the filter. The coefficients $w_j(k)$, for $j = 0, 1, \ldots, N$, are adapted aiming at the minimization of a given objective function. In the case of least-squares algorithms, the objective function is deterministic and is given by

$$\begin{aligned} \xi^d(k) &= \sum_{i=0}^{k} \lambda^{k-i} e^2(i) \\ &= \sum_{i=0}^{k} \lambda^{k-i} \left[d(i) - \mathbf{x}^T(i)\mathbf{w}(k)\right]^2 \end{aligned} \tag{5.2}$$

where $e(i)$ is the output error at instant i and $\mathbf{w}(k) = [w_o(k)\ w_1(k)\ \ldots, w_N(k)]^T$ is the adaptive filter coefficient vector. The parameter λ is an exponential weighting factor that should be chosen in the range $0 \ll \lambda \leq 1$. This parameter is also called forgetting factor since the information of the distant past has an increasingly negligible effect on the coefficient updating.

As can be noted, each error consists of the difference between the desired signal and the filter output, using the most recent coefficients $\mathbf{w}(k)$. By differentiating $\xi^d(k)$ with respect to $\mathbf{w}(k)$, it follows that

$$\frac{\partial \xi^d(k)}{\partial \mathbf{w}(k)} = -2 \sum_{i=0}^{k} \lambda^{k-i} \mathbf{x}(i)[d(i) - \mathbf{x}^T(i)\mathbf{w}(k)] \tag{5.3}$$

By equating the result to zero, it is possible to find the optimal vector $\mathbf{w}(k)$ that minimizes the least-squares error, through the following relation:

$$-\sum_{i=0}^{k} \lambda^{k-i} \mathbf{x}(i)\mathbf{x}^T(i)\mathbf{w}(k) + \sum_{i=0}^{k} \lambda^{k-i} \mathbf{x}(i)d(i) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

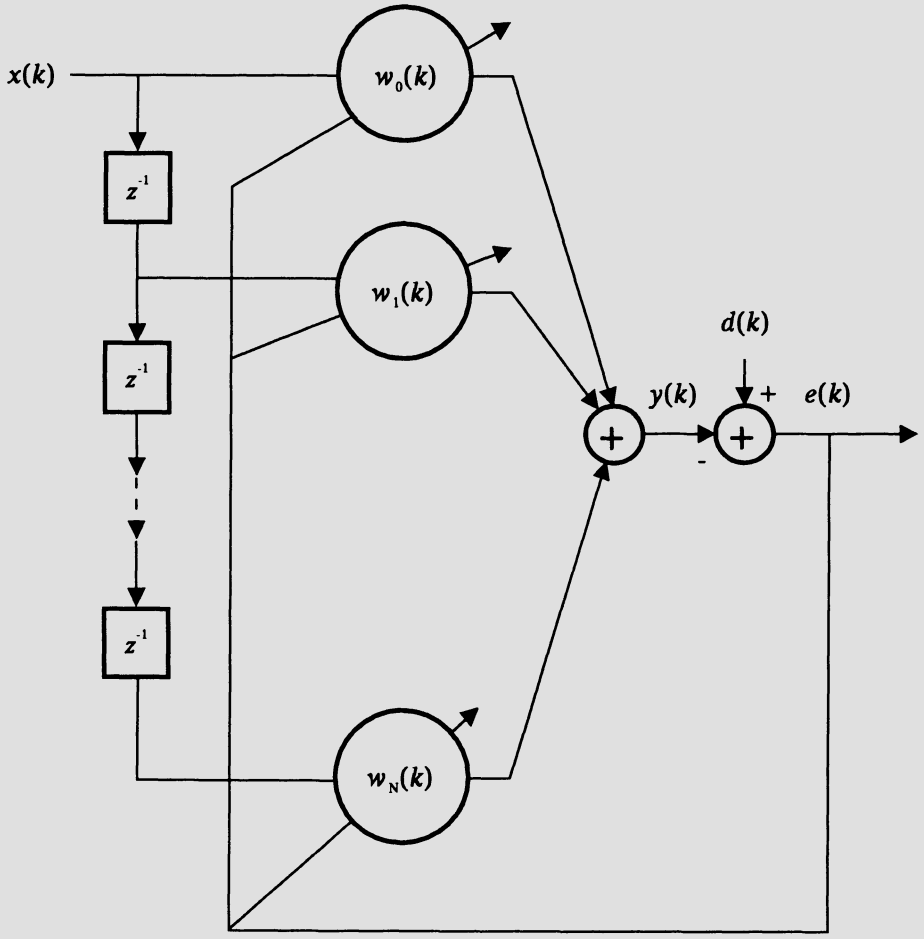


Figure 5.1 Adaptive FIR filter.

The resulting expression for the optimal coefficient vector $\mathbf{w}(k)$ is given by

$$\begin{aligned} \mathbf{w}(k) &= \left[\sum_{i=0}^{k} \lambda^{k-i}\mathbf{x}(i)\mathbf{x}^T(i)\right]^{-1} \sum_{i=0}^{k} \lambda^{k-i}\mathbf{x}(i)d(i) \\ &= \mathbf{R}_D^{-1}(k)\mathbf{p}_D(k) \end{aligned} \tag{5.4}$$

where $\mathbf{R}_D(k)$ and $\mathbf{p}_D(k)$ are called the deterministic correlation matrix of the input signal and the deterministic cross-correlation vector between the input and desired signals, respectively.

In equation (5.4) it was assumed that $\mathbf{R}_D(k)$ is nonsingular. However if $\mathbf{R}_D(k)$ is singular a generalized inverse [1] should be used instead in order to obtain a solution for $\mathbf{w}(k)$ that minimizes $\xi^d(k)$. Since we are assuming that in most practical applications the input signal has persistence of excitation, the cases requiring generalized inverse are not discussed here. It should be mentioned that if the input signal is considered to be zero for $k < 0$ then $\mathbf{R}_D(k)$ will always be singular for $k < N$, i.e., during the initialization period. During this period, the optimal value of the coefficients can be calculated for example by the backsubstitution algorithm to be presented in subsection 8.2.1.

The straightforward computation of the inverse of $\mathbf{R}_D(k)$ results in an algorithm with computational complexity of $O[N^3]$. In the conventional RLS algorithm the computation of the inverse matrix is avoided through the use of the matrix inversion lemma [1], first presented in the previous chapter for the LMS-Newton algorithm. Using the matrix inversion lemma, see equation (4.51), the inverse of the deterministic correlation matrix can then be calculated in the following form

$$\mathbf{S}_D(k) = \mathbf{R}_D^{-1}(k) = \frac{1}{\lambda}\left[\mathbf{S}_D(k-1) - \frac{\mathbf{S}_D(k-1)\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{S}_D(k-1)}{\lambda + \mathbf{x}^T(k)\mathbf{S}_D(k-1)\mathbf{x}(k)}\right] \tag{5.5}$$

The complete conventional RLS algorithm is described in Algorithm 5.1.

An alternative way to describe the conventional RLS algorithm can be obtained if equation (5.4) is rewritten in the following form

$$\left[\sum_{i=0}^{k} \lambda^{k-i}\mathbf{x}(i)\mathbf{x}^T(i)\right]\mathbf{w}(k) = \lambda\left[\sum_{i=0}^{k-1} \lambda^{k-i-1}\mathbf{x}(i)d(i)\right] + \mathbf{x}(k)d(k) \tag{5.6}$$

By considering that $\mathbf{R}_D(k-1)\mathbf{w}(k-1) = \mathbf{p}_D(k-1)$, it follows that

$$\begin{aligned}\left[\sum_{i=0}^{k} \lambda^{k-i}\mathbf{x}(i)\mathbf{x}^T(i)\right]\mathbf{w}(k) &= \lambda\mathbf{p}_D(k-1) + \mathbf{x}(k)d(k) \\ &= \lambda\mathbf{R}_D(k-1)\mathbf{w}(k-1) + \mathbf{x}(k)d(k) \\ &= \left[\sum_{i=0}^{k} \lambda^{k-i}\mathbf{x}(i)\mathbf{x}^T(i) - \mathbf{x}(k)\mathbf{x}^T(k)\right]\mathbf{w}(k-1) \\ &\quad + \mathbf{x}(k)d(k)\end{aligned} \tag{5.7}$$

where in the last equality the matrix $\mathbf{x}(k)\mathbf{x}^T(k)$ was added and subtracted inside square bracket on the right side of equation (5.7). Now, define the *a priori* error

Algorithm 5.1

Conventional RLS Algorithm

Initialization

$\mathbf{S}_D(-1) = \delta\mathbf{I}$

where δ can be the inverse of the input-signal power estimate

$\mathbf{p}_D(-1) = \mathbf{x}(-1) = [0\ 0\ \ldots\ 0]^T$

Do for $k \geq 0$:

$\mathbf{S}_D(k) = \frac{1}{\lambda}[\mathbf{S}_D(k-1) - \frac{\mathbf{S}_D(k-1)\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{S}_D(k-1)}{\lambda+\mathbf{x}^T(k)\mathbf{S}_D(k-1)\mathbf{x}(k)}]$

$\mathbf{p}_D(k) = \lambda\mathbf{p}_D(k-1) + d(k)\mathbf{x}(k)$

$\mathbf{w}(k) = \mathbf{S}_D(k)\mathbf{p}_D(k)$

If necessary compute

$y(k) = \mathbf{w}^T(k)\mathbf{x}(k)$

$e(k) = d(k) - y(k)$

□

as

$$e'(k) = d(k) - \mathbf{x}^T(k)\mathbf{w}(k-1) \tag{5.8}$$

By expressing $d(k)$ as a function of the *a priori* error and replacing the result in equation (5.7), after few manipulations, it can be shown that

$$\mathbf{w}(k) = \mathbf{w}(k-1) + e'(k)\mathbf{S}_D(k)\mathbf{x}(k) \tag{5.9}$$

With equation (5.9), it is straightforward to generate an alternative conventional RLS algorithm as shown in Algorithm 5.2.

In Algorithm 5.2, $\boldsymbol{\Psi}(k)$ is an auxiliary vector required to reduce the computational burden. Further reduction in the number of divisions is possible if it is used an additional auxiliary vector defined as $\boldsymbol{\phi}(k) = \frac{\boldsymbol{\Psi}(k)}{\lambda+\boldsymbol{\Psi}^T(k)\mathbf{x}(k)}$. This vector can be used to update $\mathbf{S}_D(k)$ as follows:

$$\mathbf{S}_D(k) = \frac{1}{\lambda}\left[\mathbf{S}_D(k-1) - \boldsymbol{\Psi}(k)\boldsymbol{\phi}^T(k)\right] \tag{5.10}$$

As will be discussed, the relation above can lead to stability problems in the RLS algorithm.

Algorithm 5.2

Alternative RLS Algorithm

Initialization

$\mathbf{S}_D(-1) = \delta\mathbf{I}$

where δ can be the inverse of an estimate of the input signal power

$\mathbf{x}(-1) = \mathbf{w}(-1) = [0\ 0\ \ldots\ 0]^T$

Do for $k \geq 0$

$e'(k) = d(k) - \mathbf{x}^T(k)\mathbf{w}(k-1)$

$\boldsymbol{\Psi}(k) = \mathbf{S}_D(k-1)\mathbf{x}(k)$

$\mathbf{S}_D(k) = \frac{1}{\lambda}[\mathbf{S}_D(k-1) - \frac{\boldsymbol{\Psi}(k)\boldsymbol{\Psi}^T(k)}{\lambda + \boldsymbol{\Psi}^T(k)\mathbf{x}(k)}]$

$\mathbf{w}(k) = \mathbf{w}(k-1) + e'(k)\mathbf{S}_D(k)\mathbf{x}(k)$

If necessary compute

$y(k) = \mathbf{w}^T(k)\mathbf{x}(k)$

$e(k) = d(k) - y(k)$

□

5.3 PROPERTIES OF THE LEAST-SQUARES SOLUTION

In this section, some properties related to the least-squares solution are discussed in order to give some insight to the algorithm behavior in several situations to be discussed later on.

5.3.1 Orthogonality Principle

Define the matrices $\mathbf{X}(k)$ and $\mathbf{d}(k)$ that contain all the information about the input signal vector $\mathbf{x}(k)$ and the desired signal vector $d(k)$ as follows:

$$\mathbf{X}(k) = \begin{bmatrix} x(k) & \lambda^{1/2}x(k-1) & \cdots & \lambda^{(k-1)/2}x(1) & \lambda^{k/2}x(0) \\ x(k-1) & \lambda^{1/2}x(k-2) & \cdots & \lambda^{(k-1)/2}x(0) & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ x(k-N) & \lambda^{1/2}x(k-N-1) & \cdots & 0 & 0 \end{bmatrix}$$

$$= [\mathbf{x}(k)\ \lambda^{1/2}\mathbf{x}(k-1)\ \cdots\ \lambda^{k/2}\mathbf{x}(0)] \tag{5.11}$$

$$\mathbf{d}(k) = [d(k)\ \lambda^{1/2}d(k-1)\ \cdots\ \lambda^{k/2}d(0)]^T \tag{5.12}$$

where $\mathbf{X}(k)$ is $(N+1)\times(k+1)$ and $\mathbf{d}(k)$ is $(k+1)\times(1)$.

By using the matrices defined above it is possible to replace the least-squares solution of equation (5.4) by the following relation

$$\mathbf{X}(k)\mathbf{X}^T(k)\mathbf{w}(k) = \mathbf{X}(k)\mathbf{d}(k) \tag{5.13}$$

The product $\mathbf{X}^T(k)\mathbf{w}(k)$ forms a vector including all the adaptive filter outputs when the coefficients are given by $\mathbf{w}(k)$. This vector corresponds to an estimate of $\mathbf{d}(k)$. Hence defining

$$\mathbf{y}(k) = \mathbf{X}^T(k)\mathbf{w}(k) = [y(k)\ \lambda^{1/2}y(k-1)\ \cdots\ \lambda^{k/2}y(0)]^T \tag{5.14}$$

it follows from equation (5.13) that

$$\mathbf{X}(k)\mathbf{X}^T(k)\mathbf{w}(k) - \mathbf{X}(k)\mathbf{d}(k) = \mathbf{X}(k)[\mathbf{y}(k) - \mathbf{d}(k)] = \mathbf{0} \tag{5.15}$$

This relation means that the weighted-error vector given by

$$\mathbf{e}(k) = \begin{bmatrix} e(k) \\ \lambda^{1/2}e(k-1) \\ \vdots \\ \lambda^{k/2}e(0) \end{bmatrix} = \mathbf{y}(k) - \mathbf{d}(k) \tag{5.16}$$

is in the null space of $\mathbf{X}(k)$, i.e., the weighted-error vector is orthogonal to all row vectors of $\mathbf{X}(k)$. This justifies the fact that (5.13) is often called normal equation. A geometrical interpretation can easily be given for a least-squares problem solution with a single coefficient filter.

Example 5.1

Suppose that $\lambda = 1$ and that the following signals are involved in the least-squares problem

$$\mathbf{d}(1) = \begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix} \quad \mathbf{X}(1) = [1 \quad -2]$$

The optimal coefficient is given by

$$[1 \quad -2]\begin{bmatrix} 1 \\ -2 \end{bmatrix} \; ; \; \mathbf{w}(1) = [1 \quad -2]\begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix}$$

After performing the calculations the result is

$$\mathbf{w}(1) = -\frac{1}{2}$$

The output of the adaptive filter with coefficient given by $\mathbf{w}(1)$ is

$$\mathbf{y}(1) = \begin{bmatrix} -\frac{1}{2} \\ 1 \end{bmatrix}$$

Note that

$$\mathbf{X}(1)[\mathbf{y}(1) - \mathbf{d}(1)] = [1 \quad -2]\begin{bmatrix} -1 \\ -0.5 \end{bmatrix} = 0$$

Fig. 5.2 illustrates the fact that $\mathbf{y}(1)$ is the projection of $\mathbf{d}(1)$ in the $\mathbf{X}(1)$ direction. In the general case we can say that the vector $\mathbf{y}(k)$ is the projection of $\mathbf{d}(k)$ onto the subspace spanned by the rows of the $\mathbf{X}(k)$.

□

5.3.2 Relation Between Least-Squares and Wiener Solutions

When $\lambda = 1$ the matrix $\frac{1}{k+1}\mathbf{R}_D(k)$ for large k is a consistent estimate of the input signal autocorrelation matrix $\mathbf{R}$, if the process from which the input signal was taken is ergodic. The same observation is valid for the vector $\frac{1}{k+1}\mathbf{p}_D(k)$ related to $\mathbf{p}$ if the desired signal is also ergodic. In this case,

$$\mathbf{R} = \lim_{k\to\infty} \frac{1}{k+1} \sum_{i=0}^{k} \mathbf{x}(i)\mathbf{x}^T(i) = \lim_{k\to\infty} \frac{1}{k+1}\mathbf{R}_D(k) \tag{5.17}$$

and

$$\mathbf{p} = \lim_{k\to\infty} \frac{1}{k+1} \sum_{i=0}^{k} \mathbf{x}(i)d(i) = \lim_{k\to\infty} \frac{1}{k+1}\mathbf{p}_D(k) \tag{5.18}$$

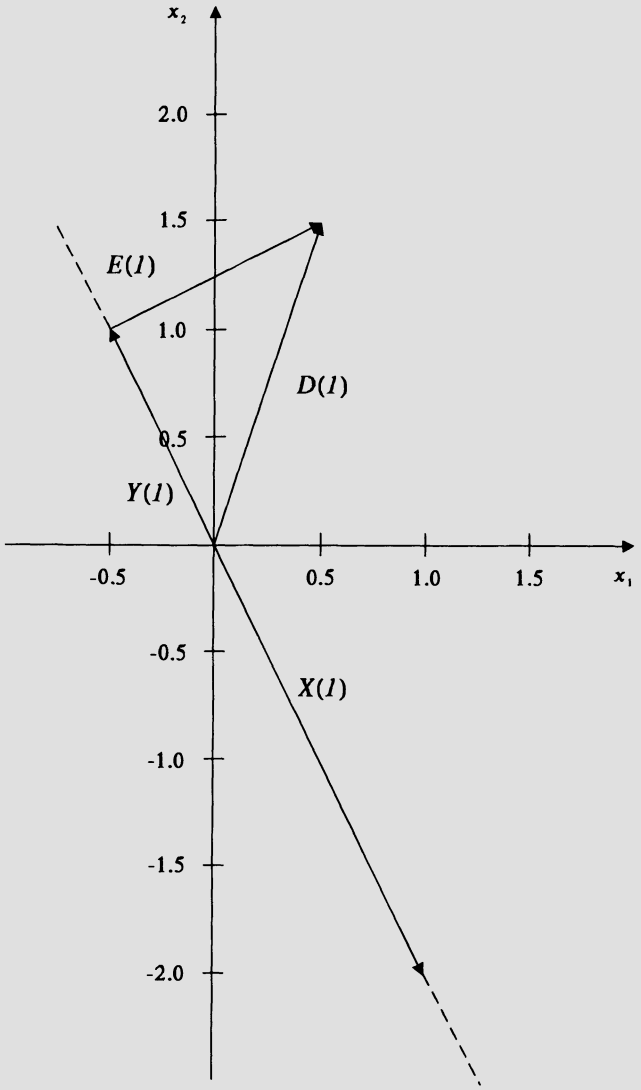


Figure 5.2 Geometric interpretation of least-square solution.

It can then be shown that

$$\mathbf{w}(k) = \mathbf{R}_D^{-1}(k)\mathbf{p}_D(k) = \mathbf{R}^{-1}\mathbf{p} = \mathbf{w}_o \tag{5.19}$$

when k tends to infinity. This result indicates that the least-squares solution tends to the Wiener solution if the signals involved are ergodic and stationary.

The stationarity requirement is due to the fact that the estimate of $\mathbf{R}$ given by equation (5.17) is not sensitive to any changes in $\mathbf{R}$ for large values of k. If the input signal is nonstationary $\mathbf{R}_D(k)$ is a biased estimate for $\mathbf{R}$. Note that in this case $\mathbf{R}$ is time-varying.

5.3.3 Influence of the Deterministic Autocorrelation Initialization

The initialization of $\mathbf{S}_D(-1) = \delta\mathbf{I}$ causes a bias in the coefficients estimated by the adaptive filter. Suppose that the initial value given to $\mathbf{R}_D(k)$ is taken into account in the actual RLS solution as follows:

$$\begin{aligned}\sum_{i=-1}^{k} \lambda^{k-i}\mathbf{x}(i)\mathbf{x}^T(i)\mathbf{w}(k) &= [\sum_{i=0}^{k} \lambda^{k-i}\mathbf{x}(i)\mathbf{x}^T(i) + \frac{\lambda^{k+1}}{\delta}\mathbf{I}]\mathbf{w}(k) \\ &= \mathbf{p}_D(k) \end{aligned} \tag{5.20}$$

By recognizing that the deterministic autocorrelation matrix leading to an unbiased solution does not include the initialization matrix, we now examine the influence of this matrix. By multiplying $\mathbf{S}_D(k) = \mathbf{R}_D^{-1}(k)$ on both sides of (5.20), and by considering $k \to \infty$, it can be concluded that

$$\mathbf{w}(k) + \frac{\lambda^{k+1}}{\delta}\mathbf{S}_D(k)\mathbf{w}(k) = \mathbf{w}_o \tag{5.21}$$

The bias caused by the initialization of $\mathbf{S}_D(k)$ is approximately

$$\mathbf{w}(k) - \mathbf{w}_o \approx -\frac{\lambda^{k+1}}{\delta}\mathbf{S}_D(k)\mathbf{w}_o \tag{5.22}$$

For $\lambda < 1$ it is straightforward to conclude that the bias tends to zero as k tends to infinity. On the other hand, when $\lambda = 1$ the elements of $\mathbf{S}_D(k)$ get smaller when the number of iterations increase, as a consequence this matrix approaches a null matrix for large k.

The RLS algorithm would reach the optimum solution for the coefficients after $N + 1$ iterations if no measurement noise is present and the influence of the initialization matrix $\mathbf{S}_D(-1)$ is negligible at this point. This result follows from the fact that after $N + 1$ iterations the input signal vector has enough information to allow the adaptive algorithm to identify the coefficients of the unknown system. In other words, enough information means the tap delay line is filled with information of the input signal.

5.3.4 Steady-State Behavior of the Coefficient Vector

In order to understand better the steady-state behavior of the adaptive filter coefficients, suppose that an FIR filter with coefficients given by $\mathbf{w}_o$ is being identified by an adaptive FIR filter of the same order employing an LS algorithm. Also assume that a measurement noise signal $n(k)$ is added to the desired signal before the error signal is calculated as follows:

$$d(k) = \mathbf{w}_o^T \mathbf{x}(k) + n(k) \tag{5.23}$$

where the additional noise is considered to be a white noise with zero mean and variance given by σ_n^2.

Given the adaptive-filter input vectors $\mathbf{x}(k)$, for $k = 0, 1, \ldots$, we are interested in calculating the average values of the adaptive filter coefficients $w_i(k)$, for $i = 0, 1, \ldots, N$. The desired result is the following equality valid for $k \geq N$.

$$\begin{aligned} E[\mathbf{w}(k)] &= E\left\{\left[\mathbf{X}(k)\mathbf{X}^T(k)\right]^{-1}\mathbf{X}(k)\mathbf{d}(k)\right\} \\ &= E\left\{\left[\mathbf{X}(k)\mathbf{X}^T(k)\right]^{-1}\mathbf{X}(k)(\mathbf{X}^T(k)\mathbf{w}_o + \mathbf{n}(k))\right\} \\ &= E\left\{\left[\mathbf{X}(k)\mathbf{X}^T(k)\right]^{-1}\mathbf{X}(k)\mathbf{X}^T(k)\mathbf{w}_o\right\} = \mathbf{w}_o \end{aligned} \tag{5.24}$$

where $\mathbf{n}(k) = [n(k)\ \lambda^{1/2}n(k-1)\ \lambda n(k-2)\ \ldots\ \lambda^{k/2}n(0)]^T$ is the noise vector, whose elements were considered orthogonal to the input signal. The equation above shows that the estimate given by the LS algorithm is an unbiased estimate when $\lambda \leq 1$.

A more accurate analysis reveals the behavior of the adaptive filter coefficients during the transient period. The error in the filter coefficients can be described by the following $(N+1) \times 1$ vector

$$\Delta\mathbf{w}(k) = \mathbf{w}(k) - \mathbf{w}_o \tag{5.25}$$

It follows from equation (5.7) that

$$\mathbf{R}_D(k)\mathbf{w}(k) = \lambda\mathbf{R}_D(k-1)\mathbf{w}(k-1) + \mathbf{x}(k)d(k) \tag{5.26}$$

Defining the minimum output error as

$$e_o(k) = d(k) - \mathbf{x}^T(k)\mathbf{w}_o \tag{5.27}$$

and replacing $d(k)$ in equation (5.26), it can be easily deduced that

$$\mathbf{R}_D(k)\Delta\mathbf{w}(k) = \lambda\mathbf{R}_D(k-1)\Delta\mathbf{w}(k-1) + \mathbf{x}(k)e_o(k) \tag{5.28}$$

where it was used the following straightforward relation

$$\mathbf{R}_D(k) = \lambda\mathbf{R}_D(k-1) + \mathbf{x}(k)\mathbf{x}^T(k) \tag{5.29}$$

The solution of equation (5.28) is given by

$$\Delta\mathbf{w}(k) = \lambda^{k+1}\mathbf{S}_D(k)\mathbf{R}_D(-1)\Delta\mathbf{w}(-1) + \mathbf{S}_D(k)\sum_{i=0}^{k}\lambda^{k-i}\mathbf{x}(i)e_o(i) \tag{5.30}$$

By replacing $\mathbf{R}_D(-1)$ by $\frac{1}{\delta}\,\mathbf{I}$ and taking the expected value of the resulting equation, it follows that

$$E[\Delta\mathbf{w}(k)] = \frac{\lambda^{k+1}}{\delta}E[\mathbf{S}_D(k)]\Delta\mathbf{w}(-1)$$
$$+E[\mathbf{S}_D(k)\sum_{i=0}^{k}\lambda^{k-i}\mathbf{x}(i)e_o(i)] \tag{5.31}$$

Since $\mathbf{S}_D(k)$ is dependent on all past input signal vectors, becoming relatively invariant when the number of iterations increase, the contribution of any individual $\mathbf{x}(i)$ can be considered negligible. Also, due to the orthogonality principle, $e_o(i)$ can also be considered uncorrelated to all elements of $\mathbf{x}(i)$. This means that the last vector in equation (5.31) cannot have large element values. On the other hand, the first vector in equation (5.31) can have large element values only during the initial convergence, since as $k \to \infty, \lambda^{k+1} \to 0$ and $\mathbf{S}_D(k)$ is expected to have a nonincreasing behavior, i.e., $\mathbf{R}_D(k)$ is assumed to remain positive definite as $k \to \infty$ and the input signal power does not become too small. The above discussion leads to the conclusion that the adaptive filter coefficients tend to the optimal values in $\mathbf{w}_o$ almost independently from the eigenvalue spread of the input signal correlation matrix.

If we consider the spectral decomposition of the matrix $E[\mathbf{S}_D(k)]$ (see equation (2.65)), the dependency on the eigenvalues of $\mathbf{R}$ can be easily accounted for in

the simple case of $\lambda = 1$. Applying the expected value operator to the relation of equation (5.17), we can infer that

$$E[\mathbf{S}_D(k)] \approx \frac{\mathbf{R}^{-1}}{(k+1)} \tag{5.32}$$

for large k. Now consider the slowest decaying mode of the spectral decomposition of $E[\mathbf{S}_D(k)]$ given by

$$\mathbf{S}_{D_{max}} = \frac{\mathbf{q}_{min}\mathbf{q}_{min}^T}{\lambda_{min}(k+1)} \tag{5.33}$$

where λ_{min} is the smallest eigenvalue of $\mathbf{R}$ and $\mathbf{q}_{min}$ is the corresponding eigenvector. Applying this result to equation (5.31), with $\lambda = 1$, we can conclude that the value of the minimum eigenvalue affects the convergence of the filter coefficients only in the first few iterations, because the term $k+1$ in the denominator reduces the values of the elements of $\mathbf{S}_{D_{max}}$.

Further interesting properties of the coefficients generated by the LS algorithm are:

- The estimated coefficients are the best linear unbiased solution to the identification problem [1], in the sense that no other unbiased solution generated by alternative approaches has lower variance.
- If the additive noise is normally distributed the LS solution reaches the Cramer-Rao lower bound, resulting in a minimum-variance unbiased solution [1]. The Cramer-Rao lower bound establishes a lower bound to the coefficient-error-vector covariance matrix for any unbiased estimator of the optimal parameter vector $\mathbf{w}_o$.

5.3.5 Coefficient-Error-Vector Covariance Matrix

So far, we have shown that the estimation parameters in the vector $\mathbf{w}(k)$ converge in average to their optimal value of the vector $\mathbf{w}_o$. However, it is essential to analyze the coefficient-error-vector covariance matrix in order to determine how good is the obtained solution, in the sense that we are measuring how far the parameters wander around the optimal solution.

Using the same convergence assumption of the last section it will be shown here that the coefficient-error-vector covariance matrix is given by

$$cov[\Delta\mathbf{w}(k)] = E\left[(\mathbf{w}(k) - \mathbf{w}_o)(\mathbf{w}(k) - \mathbf{w}_o)^T\right] = \sigma_n^2 \mathbf{S}_D(k) \tag{5.34}$$

for $\lambda = 1$.

Proof

First note that by using equations (5.4) and (5.13), the following relations are verified

$$\begin{aligned}
\mathbf{w}(k) - \mathbf{w}_o &= \mathbf{S}_D(k)\mathbf{p}_D(k) - \mathbf{S}_D(k)\mathbf{S}_D^{-1}(k)\mathbf{w}_o \\
&= \left[\mathbf{X}(k)\mathbf{X}^T(k)\right]^{-1}\mathbf{X}(k)\left[\mathbf{d}(k) - \mathbf{X}^T(k)\mathbf{w}_o\right] \\
&= \left[\mathbf{X}(k)\mathbf{X}^T(k)\right]^{-1}\mathbf{X}(k)\mathbf{n}(k)
\end{aligned}$$

where $\mathbf{n}(k) = [n(k)\ \lambda^{1/2}n(k-1)\ \lambda n(k-2)\ \ldots\ \lambda^{k/2}n(0)]^T$.

Applying the last equation to the covariance of the coefficient-error-vector it follows that

$$\begin{aligned}
cov[\Delta\mathbf{w}(k)] &= \left[\mathbf{X}(k)\mathbf{X}^T(k)\right]^{-1}\mathbf{X}(k)E[\mathbf{n}(k)\mathbf{n}^T(k)]\mathbf{X}^T(k)\left[\mathbf{X}(k)\mathbf{X}^T(k)\right]^{-1} \\
&= \sigma_n^2\mathbf{S}_D(k)\mathbf{X}(k)\Lambda\mathbf{X}^T(k)\mathbf{S}_D(k)
\end{aligned}$$

where $\mathbf{X}(k)$ was considered to be known and

$$\Lambda = \begin{bmatrix} 1 & & & \\ & \lambda & & \mathbf{0} \\ & & \lambda^2 & \\ & \mathbf{0} & & \ddots \\ & & & & \lambda^k \end{bmatrix}$$

where for $\lambda = 1$, $\Lambda = \mathbf{I}$. In this case,

$$\begin{aligned}
cov[\Delta\mathbf{w}(k)] &= \sigma_n^2\mathbf{S}_D(k)\mathbf{X}(k)\mathbf{X}^T(k)\mathbf{S}_D(k) \\
&= \sigma_n^2\mathbf{S}_D(k)\mathbf{R}_D(k)\mathbf{S}_D(k) \\
&= \sigma_n^2\mathbf{S}_D(k)
\end{aligned}$$

□

Therefore, when $\lambda = 1$, the coefficient-error-vector covariance matrix tends to decrease its norm as time progresses since $\mathbf{S}_D(k)$ is also norm decreasing. The variance of the additional noise $n(k)$ influences directly the norm of the covariance matrix.

5.3.6 Behavior of the Error Signal

It is important to understand how the error signal behaves in the RLS algorithm. When a measurement noise is present in the adaptive filtering process, the error signal is given by

$$e(k) = d'(k) - \mathbf{w}^T(k)\mathbf{x}(k) + n(k) \tag{5.35}$$

where $d'(k)$ is the desired signal without measurement noise.

Again if the input signal is considered known (conditional expectation), then

$$\begin{aligned} E[e(k)] &= E[d'(k)] - E[\mathbf{w}^T(k)]\mathbf{x}(k) + E[n(k)] \\ &= E[d'(k)] - \mathbf{w}_o^T\mathbf{x}(k) + E[n(k)] \\ &= E[n(k)] \end{aligned} \tag{5.36}$$

assuming that the adaptive filter order is sufficient to model perfectly the desired signal.

From equation (5.36) it can be concluded that if the noise signal has zero mean then

$$E[e(k)] = 0$$

It is also important to access the minimum mean value of the squared error that is reachable using an RLS algorithm. The minimum mean-square error (MSE) in the presence of external uncorrelated noise is given by

$$\xi_{min} = E[e^2(k)] = E[n^2(k)] = \sigma_n^2 \tag{5.37}$$

where it is assumed that the adaptive-filter multiplier coefficients were frozen at their optimum values and that the number of coefficients of the adaptive filter is sufficient to model the desired signal. It should be noted, however, that if the additive noise is correlated with the input and the desired signal, a more complicated expression for the MSE results, accounting for the referred correlation.

Example 5.2

Repeat the equalization problem of Example 3.1 of Chapter 3 using the RLS algorithm.

(a) Using $\lambda = 0.99$, run the algorithm and save the matrix $\mathbf{S}_D(k)$ at the iteration 500 and compare with the the inverse of the input signal correlation matrix.

(b) Plot the convergence path for the RLS algorithm on the MSE surface.

Solution:

(a) The inverse of matrix $\mathbf{R}$, as computed in the Example 3.1, is given by

$$\mathbf{R}^{-1} = 0.45106 \begin{bmatrix} 1.6873 & 0.7937 \\ 0.7937 & 1.6873 \end{bmatrix} = \begin{bmatrix} 0.7611 & 0.3580 \\ 0.3580 & 0.7611 \end{bmatrix}$$

The initialization matrix $\mathbf{S}_D(-1)$ was a diagonal matrix with the diagonal elements equal to 0.1. The matrix $\mathbf{S}_D(k)$ at the $500th$ iteration, obtained by averaging the results of 30 experiments, was

$$\mathbf{S}_D(500) = \begin{bmatrix} 0.0078 & 0.0037 \\ 0.0037 & 0.0078 \end{bmatrix}$$

Also, the obtained values of the deterministic cross-correlation vector was

$$\mathbf{p}_D(500) = \begin{bmatrix} 95.05 \\ 46.21 \end{bmatrix}$$

Now if we divide each element of the matrix $\mathbf{R}^{-1}$ by

$$\frac{1 - \lambda^{k+1}}{1 - \lambda} = 99.34$$

The resulting matrix is

$$\frac{1}{99.34}\mathbf{R}^{-1} = \begin{bmatrix} 0.0077 & 0.0036 \\ 0.0036 & 0.0077 \end{bmatrix}$$

as can be noted the values of the elements of the above matrix are close to the average values of the corresponding elements of matrix $\mathbf{S}_D(500)$.

Similarly, if we multiply the cross-correlation vector $\mathbf{p}$ by 99.34, the result is

$$99.34\mathbf{p} = \begin{bmatrix} 94.61 \\ 47.31 \end{bmatrix}$$

The values of the elements of this vector are also close to the corresponding elements of $\mathbf{p}_D(500)$.

(b) The convergence path of the RLS algorithm on the MSE surface is depicted in Fig. 5.3. The reader should notice that the RLS algorithm approaches the minimum using large steps when the coefficients of the adaptive filter are far away from the optimum solution.

□

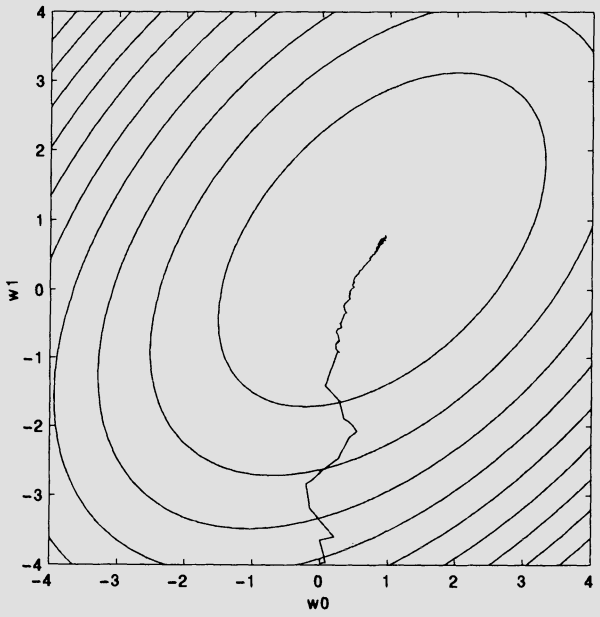


Figure 5.3 Convergence path of the RLS adaptive filter.

5.3.7 Excess of Mean-Square Error and Misadjustment

In a practical implementation of the recursive least-squares algorithm, the best estimation for the unknown parameter vector is given by $\mathbf{w}(k)$, whose expected value is $\mathbf{w}_o$. However, there is always an excess of MSE at the output caused by the error in the coefficient estimation, namely $\Delta\mathbf{w}(k) = \mathbf{w}(k) - \mathbf{w}_o$. The mean-square error for a given coefficient vector (see equation (3.38))

$$\begin{aligned} \xi(k) &= \xi_{min} + (\mathbf{w}(k) - \mathbf{w}_o)^T \mathbf{R}(\mathbf{w}(k) - \mathbf{w}_o) \\ &= \sigma_n^2 + \Delta\mathbf{w}^T(k)\mathbf{R}\Delta\mathbf{w}(k) \end{aligned} \tag{5.38}$$

Now considering that $\Delta w_j(k)$ for $j = 0, 1, \ldots, N$ are random variables with zero mean and independent of $\mathbf{x}(k)$, then by employing equations (5.34) and (5.18), the ensemble average of the MSE can be calculated as follows

$$\begin{aligned} E[\xi(k)] &= \sigma_n^2 + E[\Delta\mathbf{w}^T(k)\mathbf{R}\Delta\mathbf{w}(k)] \\ &= \sigma_n^2 + E[tr(\mathbf{R}\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k))] \\ &= \sigma_n^2 + tr(\mathbf{R}E[\Delta\mathbf{w}(k)\Delta\mathbf{w}^T(k)]) \end{aligned}$$

On a number of occasions it is interesting to consider the analysis for $\lambda = 1$ separated from that for $\lambda < 1$.

Excess of MSE for $\lambda = 1$

From the results of equations (5.34) and (5.17) we can infer that

$$\begin{aligned} E[\xi(k)] &= \sigma_n^2 + \sigma_n^2 tr(\mathbf{R}\mathbf{S}_D(k)) \\ &= \sigma_n^2(1 + tr(\mathbf{R}\frac{\mathbf{R}^{-1}}{k+1})) \; for \; k \to \infty \\ &= \sigma_n^2(1 + \frac{N+1}{k+1}) \end{aligned}$$

for $\lambda = 1$. As can be noted the minimum MSE can be reached only after the algorithm has operated in a number of samples larger than the filter order.

Excess of MSE for $\lambda < 1$

Again assuming that the mean-square error surface is quadratic as considered in equation (5.38), the expected excess in the MSE is then defined by

$$\Delta\xi(k) = E[\Delta\mathbf{w}^T(k)\mathbf{R}\Delta\mathbf{w}(k)] \tag{5.39}$$

The objective now is to calculate and analyze the excess of MSE when $\lambda < 1$. From equation (5.28) one can easily show that

$$\Delta\mathbf{w}(k) = \lambda\mathbf{S}_D(k)\mathbf{R}_D(k-1)\Delta\mathbf{w}(k-1) + \mathbf{R}_D^{-1}(k)\mathbf{x}(k)e_o(k) \tag{5.40}$$

By applying equation (5.40) to (5.39), it follows that

$$E[\Delta\mathbf{w}^T(k)\mathbf{R}\Delta\mathbf{w}(k)] = \rho_1 + \rho_2 + \rho_3 + \rho_4 \tag{5.41}$$

where

$$\begin{aligned}
\rho_1 &= \lambda^2 E[\Delta\mathbf{w}^T(k-1)\mathbf{R}_D(k-1)\mathbf{S}_D(k)\mathbf{R}\mathbf{S}_D(k)\mathbf{R}_D(k-1)\Delta\mathbf{w}(k-1)] \\
\rho_2 &= \lambda E[\Delta\mathbf{w}^T(k-1)\mathbf{R}_D(k-1)\mathbf{S}_D(k)\mathbf{R}\mathbf{S}_D(k)\mathbf{x}(k)e_o(k)] \\
\rho_3 &= \lambda E[\mathbf{x}^T(k)\mathbf{S}_D(k)\mathbf{R}\mathbf{S}_D(k)\mathbf{R}_D(k-1)\Delta\mathbf{w}(k-1)e_o(k)] \\
\rho_4 &= E[\mathbf{x}^T(k)\mathbf{S}_D(k)\mathbf{R}\mathbf{S}_D(k)\mathbf{x}(k)e_o^2(k)]
\end{aligned}$$

Now each term in equation (5.41) will be evaluated separately.

1- Evaluation of ρ_1

First note that as $k \to \infty$, it can be assumed that $\mathbf{R}_D(k) \approx \mathbf{R}_D(k-1)$, then

$$\rho_1 \approx \lambda^2 E[\Delta\mathbf{w}^T(k-1)\mathbf{R}\Delta\mathbf{w}(k-1)] \tag{5.42}$$

2- Evaluation of ρ_2

Since each element of $\mathbf{R}_D(k)$ is given by

$$r_{d,ij}(k) = \sum_{l=0}^{k} \lambda^{k-l} x(l-i)x(l-j) \tag{5.43}$$

for $0 \leq i,\ j < N$. Therefore,

$$E[r_{d,ij}(k)] = \sum_{l=0}^{k} \lambda^{k-l} E[x(l-i)x(l-j)]$$

If $x(k)$ is stationary, $r(i-j) = E[x(l-i)x(l-j)]$ is independent of the value l, then

$$r_{d,ij}(k) = r(i-j)\frac{1-\lambda^{k+1}}{1-\lambda} \approx \frac{r(i-j)}{1-\lambda} \tag{5.44}$$

Equation (5.44) allows the conclusion that

$$E[\mathbf{R}_D(k)] \approx \frac{1}{1-\lambda}E[\mathbf{x}(k)\mathbf{x}^T(k)] = \frac{1}{1-\lambda}\mathbf{R} \tag{5.45}$$

In each step, it can be considered that

$$\mathbf{R}_D(k) = \frac{1}{1-\lambda}\mathbf{R} + \Delta\mathbf{R}(k) \tag{5.46}$$

where $\Delta\mathbf{R}(k)$ is a symmetric error matrix with zero-mean stochastic entries that are independent of the input signal. From equations (5.45) and (5.46), it can be concluded that

$$\mathbf{S}_D(k)\mathbf{R} \approx (1-\lambda)[\mathbf{I} - (1-\lambda)\mathbf{R}^{-1}\Delta\mathbf{R}(k)] \tag{5.47}$$

where in the last relation $\mathbf{S}_D(k)\Delta\mathbf{R}(k)$ was considered approximately equal to $(1-\lambda)\mathbf{R}^{-1}\Delta\mathbf{R}(k)$, by disregarding second-order errors.

In the long run, it is known that $E[\mathbf{S}_D(k)\mathbf{R}] = (1-\lambda)\mathbf{I}$, that means the second term inside the square bracket in equation (5.47) is a measure of the perturbation caused by $\Delta\mathbf{R}(k)$ in the product $\mathbf{S}_D(k)\mathbf{R}$. Denoting the perturbation by $\Delta\mathbf{I}(k)$, it can be concluded that

$$\begin{aligned} \rho_2 &\approx \lambda(1-\lambda)E[\Delta\mathbf{w}^T(k-1)(\mathbf{I}-\Delta\mathbf{I}(k))\mathbf{x}(k)e(k)] \\ &\approx \lambda(1-\lambda)E[\Delta\mathbf{w}^T(k-1)]E[\mathbf{x}(k)e_o(k)] = 0 \end{aligned} \tag{5.48}$$

where it was considered that $\Delta\mathbf{w}^T(k)$ is independent of $\mathbf{x}(k)$ and $e_o(k)$, $\Delta\mathbf{I}(k)$ was also considered an independent error matrix with zero mean, and finally it was used the fact that $\mathbf{x}(k)$ and $e_o(k)$ are orthogonal.

3- Following a similar approach it can be shown that $\rho_3 = 0$.

4- Evaluation of ρ_4

$$\begin{aligned} \rho_4 &= E[\mathbf{x}^T(k)\mathbf{S}_D(k)\mathbf{R}\mathbf{S}_D(k)\mathbf{R}\mathbf{R}^{-1}\mathbf{x}(k)e_o^2(k)] \\ &\approx (1-\lambda)^2 E[\mathbf{x}^T(k)(\mathbf{I}-\Delta\mathbf{I}(k))^2\mathbf{R}^{-1}\mathbf{x}(k)]\xi_{min} \end{aligned} \tag{5.49}$$

where equation (5.47) was used and $e_o(k)$ was considered independent of $\mathbf{x}(k)$ and $\Delta\mathbf{I}(k)$. By using the property that

$$E[\mathbf{x}^T(k)(\mathbf{I}-\Delta\mathbf{I}(k))^2\mathbf{R}^{-1}\mathbf{x}(k)] = trE[(\mathbf{I}-\Delta\mathbf{I}(k))^2\mathbf{R}^{-1}\mathbf{x}(k)\mathbf{x}^T(k)]$$

and recalling that $\Delta\mathbf{I}(k)$ has zero mean and is independent of $\mathbf{x}(k)$, then equation (5.49) is simplified to

$$\rho_4 = (1-\lambda)^2 tr\{\mathbf{I}+E[\Delta\mathbf{I}^2(k)]\}\xi_{min} \tag{5.50}$$

where $tr[\cdot]$ means trace of $[\cdot]$.

By using equations (5.42), (5.48), and (5.50), it follows that

$$\begin{aligned} E[\Delta\mathbf{w}^T(k)\mathbf{R}\Delta\mathbf{w}(k)] &= \lambda^2 E[\Delta\mathbf{w}^T(k-1)\mathbf{R}\Delta\mathbf{w}(k-1)] \\ &\quad +(1-\lambda)^2 tr\{\mathbf{I}+E[\Delta\mathbf{I}^2(k)]\}\xi_{min} \end{aligned} \tag{5.51}$$

Asymptotically, the solution of the equation above is

$$\xi_{exc} = \frac{1-\lambda}{1+\lambda} tr\{\mathbf{I}+E[\Delta\mathbf{I}^2(k)]\}\xi_{min} \tag{5.52}$$

Note that the term given by $E[\Delta\mathbf{I}^2(k)]$ is not easy to estimate and is dependent on fourth-order statistics of the input signal. However, in specific situations, it is possible to compute an approximate estimate for this matrix. In steady state, it can be considered for white noise input signal that only the diagonal elements of $\mathbf{R}$ and $\Delta\mathbf{R}$ are important to the generation of excess of MSE. Even when the input signal is not white, this diagonal dominance can be considered a reasonable approximation in most of the cases. From the definition of $\Delta\mathbf{I}(k)$,

$$E[\Delta\mathbf{I}_{ii}^2(k)] = (1-\lambda)\frac{E[\Delta r_{ii}^2(k)]}{[\sigma_x^2]^2} \tag{5.53}$$

where σ_x^2 is variance of $x(k)$. By calculating $\Delta\mathbf{R}(k)-\lambda\Delta\mathbf{R}(k-1)$ using equation (5.46), we show that

$$\Delta r_{ii}(k) = \lambda\Delta r_{ii}(k-1) + x(k-i)x(k-i) - r_{ii} \tag{5.54}$$

Squaring the equation above, applying the expectation operation, and using the independence between $\Delta r_{ii}(k)$ and $x(k)$, it follows that

$$\begin{aligned} E[\Delta r_{ii}^2(k)] &= \lambda^2 E[\Delta r_{ii}^2(k-1)] \\ &\quad +E[(x(k-i)x(k-i)-r_{ii})^2] \end{aligned} \tag{5.55}$$

Therefore, asymptotically

$$E[\Delta r_{ii}^2(k)] = \frac{1}{1-\lambda^2}\sigma_{x^2(k-i)}^2 = \frac{1}{1-\lambda^2}\sigma_{x^2}^2 \tag{5.56}$$

By substituting equation (5.56) in (5.53), it becomes

$$E[\Delta \mathbf{I}_{ii}^2] = \frac{1-\lambda}{1+\lambda}\frac{\sigma_{x^2}^2}{\sigma_x^2} = \frac{1-\lambda}{1+\lambda}\mathcal{K} \tag{5.57}$$

where $\mathcal{K} = \frac{\sigma_{x^2}^2}{\sigma_x^2}$ is dependent on input signal statistics. For Gaussian signals, $\mathcal{K} = 2$ [9].

Returning to our main objective, the excess of MSE can then be described as

$$\xi_{exc} = (N+1)\frac{1-\lambda}{1+\lambda}(1+\frac{1-\lambda}{1+\lambda}\mathcal{K})\xi_{min} \tag{5.58}$$

If λ is approximately one and $\mathcal{K}$ is not very large then

$$\xi_{exc} = (N+1)\frac{1-\lambda}{1+\lambda}\xi_{min} \tag{5.59}$$

this expression can be reached through a simpler analysis [8]. However, the more complete derivation shown here can give more insight to the interpretation of the results obtained by using the RLS algorithm, mainly when λ is not very close to one.

The misadjustment formula can easily be deduced from equation (5.58)

$$M = \frac{\xi_{exc}}{\xi_{min}} = (N+1)\frac{1-\lambda}{1+\lambda}(1+\frac{1-\lambda}{1+\lambda}\mathcal{K}) \tag{5.60}$$

As can be noted the decrease of λ from one brings a fourth-order statistics term into the picture by increasing the misadjustment. Then, the fast adaptation of the RLS algorithm, that corresponds to smaller λ, brings a noisier steady-state response. Therefore, when working in a stationary environment the best choice for λ would be one, if the excess of MSE in the steady state is unacceptable. However, other problems such as instability due to quantization noise are prone to occur in this case.

5.4 BEHAVIOR IN NONSTATIONARY ENVIRONMENTS

In cases where the input signal and/or the desired signal are nonstationary, the optimal values of the coefficients are time variant and described by $\mathbf{w}_o(k)$. That means the autocorrelation matrix $\mathbf{R}(k)$ and/or the cross-correlation vector $\mathbf{p}(k)$ are time variant. For example, typically in a system identification application the autocorrelation matrix $\mathbf{R}(k)$ is time invariant while the cross-correlation matrix $\mathbf{p}(k)$ is time variant, because in this case the designer can choose the input signal. On the other hand, in equalization, prediction, and signal enhancement applications both the input and the desired signal are nonstationary leading to time-varying matrices $\mathbf{R}(k)$ and $\mathbf{p}(k)$.

The objective in the present section is to analyze how close the RLS algorithm is able to track the time-varying solution $\mathbf{w}_o(k)$. Also, it is of interest to learn how the tracking error in $\mathbf{w}(k)$ affects the output MSE [9]. Here, the effects of the measurement noise are not considered, since only the nonstationary effects are desired. Also, both effects on the MSE can be added since, in general, they are independent.

Recall from equations (5.8) and (5.9) that

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mathbf{S}_D(k)\mathbf{x}(k)(d(k) - \mathbf{x}^T(k)\mathbf{w}(k-1)) \tag{5.61}$$

and

$$d(k) = \mathbf{x}^T(k)\mathbf{w}_o(k-1) + e'_o(k) \tag{5.62}$$

The error signal $e'_o(k)$ is the minimum error at iteration k being generated by the nonstationarity of the environment. One can replace equation (5.62) in (5.61) in order to obtain the following relation

$$\begin{aligned} \mathbf{w}(k) &= \mathbf{w}(k-1) + \mathbf{S}_D(k)\mathbf{x}(k)\mathbf{x}^T(k)[\mathbf{w}_o(k-1) - \mathbf{w}(k-1)] \\ &\quad + \mathbf{S}_D(k)\mathbf{x}(k)e'_o(k) \end{aligned} \tag{5.63}$$

By taking the expected value of equation (5.63), considering that $\mathbf{x}(k)$ and $e'_o(k)$ are approximately orthogonal, and that $\mathbf{w}(k-1)$ is independent of $\mathbf{x}(k)$, then

$$\begin{aligned} E[\mathbf{w}(k)] &= E[\mathbf{w}(k-1)] \\ &\quad + E[\mathbf{S}_D(k)\mathbf{x}(k)\mathbf{x}^T(k)]\{\mathbf{w}_o(k-1) - E[\mathbf{w}(k-1)]\} \end{aligned} \tag{5.64}$$

It is now needed to compute $E[\mathbf{S}_D(k)\mathbf{x}(k)\mathbf{x}^T(k)]$ in the case of nonstationary input signal. From equations (5.43) and (5.45), one can show that

$$\mathbf{R}_D(k) = \sum_{l=0}^{k} \lambda^{k-l}\mathbf{R}(l) + \Delta\mathbf{R}(k) \tag{5.65}$$

since $E[\mathbf{R}_D(k)] = \sum_{l=0}^{k}\lambda^{k-l}\mathbf{R}(l)$. The matrix $\Delta\mathbf{R}(k)$ is again considered a symmetric error matrix with zero-mean stochastic entries that are independent of the input signal.

If the environment is considered to be varying in a slower pace than the memory of the adaptive RLS algorithm then

$$\mathbf{R}_D(k) \approx \frac{1}{1-\lambda}\mathbf{R}(k) + \Delta\mathbf{R}(k) \tag{5.66}$$

Considering that $(1-\lambda)||\mathbf{R}^{-1}(k)\Delta\mathbf{R}(k)|| < 1$ and using the same procedure to deduce equation (5.47), we obtain

$$\mathbf{S}_D(k) \approx (1-\lambda)\mathbf{R}^{-1}(k) - (1-\lambda)^2\mathbf{R}^{-1}(k)\Delta\mathbf{R}(k)\mathbf{R}^{-1}(k) \tag{5.67}$$

it then follows that

$$\begin{aligned} & E[\mathbf{w}(k)] = E[\mathbf{w}(k-1)] + \{(1-\lambda)E[\mathbf{R}^{-1}(k)\mathbf{x}(k)\mathbf{x}^T(k)] \\ - \quad & (1-\lambda)^2 E[\mathbf{R}^{-1}(k)\Delta\mathbf{R}(k)\mathbf{R}^{-1}(k)\mathbf{x}(k)\mathbf{x}^T(k)]\}\,[\mathbf{w}_o(k-1) - E[\mathbf{w}(k-1)]] \\ \approx \quad & E[\mathbf{w}(k-1)] + (1-\lambda)\,[\mathbf{w}_o(k-1) - E[\mathbf{w}(k-1)]] \end{aligned} \tag{5.68}$$

where it was considered that $\Delta\mathbf{R}(k)$ is independent of $\mathbf{x}(k)$ and has zero expected value.

Now defining the lag-error vector in the coefficients as

$$\mathbf{l}_{\mathbf{w}}(k) = E[\mathbf{w}(k)] - \mathbf{w}_o(k) \tag{5.69}$$

From equation (5.68) it can be concluded that

$$\mathbf{l}_{\mathbf{w}}(k) = \lambda\mathbf{l}_{\mathbf{w}}(k-1) - \mathbf{w}_o(k) + \mathbf{w}_o(k-1) \tag{5.70}$$

Equation (5.70) is equivalent to say that the lag is generated by applying the optimal instantaneous value $\mathbf{w}_o(k)$ through a first-order discrete-time filter as follows:

$$L_{\mathbf{w}i}(z) = -\frac{z-1}{z-\lambda}W_{oi}(z) \tag{5.71}$$

The discrete-time filter transient response converges with a time constant given by

$$\tau = \frac{1}{1-\lambda} \tag{5.72}$$

The time constant is of course the same for each individual coefficient. Note that the tracking ability of the coefficients in the RLS algorithm is independent of the input-signal correlation-matrix eigenvalues.

The lag in the coefficients leads to an excess of MSE. In order to calculate the MSE suppose that the optimal coefficients values are first-order Markov processes described by

$$\mathbf{w}_o(k) = \lambda_{\mathbf{W}}\mathbf{w}_o(k-1) + \mathbf{n}_{\mathbf{W}}(k) \tag{5.73}$$

where $\mathbf{n}_{\mathbf{W}}(k)$ is a vector whose elements are zero-mean Gaussian noise processes with variance $\sigma^2_{\mathbf{W}}$, and $\lambda_{\mathbf{W}} < 1$. Note that $\lambda < \lambda_{\mathbf{W}} < 1$, since the optimal coefficients values must vary slower than the filter tracking speed, that means $\frac{1}{1-\lambda} \ll \frac{1}{1-\lambda_{\mathbf{W}}}$.

The excess of MSE due to lag is then given by (see the derivations around equation (3.38))

$$\begin{aligned} \xi_{lag} &= E[\mathbf{l}^T_{\mathbf{W}}(k)\mathbf{R}\mathbf{l}_{\mathbf{W}}(k)] \\ &= E[tr(\mathbf{R}\mathbf{l}_{\mathbf{W}}(k)\mathbf{l}^T_{\mathbf{W}}(k))] \\ &= tr(\mathbf{R}E[\mathbf{l}_{\mathbf{W}}(k)\mathbf{l}^T_{\mathbf{W}}(k)]) \end{aligned} \tag{5.74}$$

For $\lambda_{\mathbf{W}}$ not close to one, it is a bit more complicated to deduce the excess of MSE due to lag than for $\lambda_{\mathbf{W}} \approx 1$. However, the effort is worth it because the resulting expression is more accurate. From equation (5.71), we can see that the lag-error vector elements are generated by applying a first-order discrete-time system to the elements of the unknown system coefficient vector. On the other hand, the coefficients of the unknown system are generated by applying each element of the noise vector $\mathbf{n}_{\mathbf{W}}(k)$ to a first-order all-pole filter, with the pole placed at $\lambda_{\mathbf{W}}$. For the unknown coefficient vector with the model above, the lag-error vector elements can be generated by applying the elements of the noise vector $\mathbf{n}_{\mathbf{W}}(k)$ to a discrete-time filter with transfer function

$$H(z) = \frac{-(z-1)z}{(z-\lambda)(z-\lambda_{\mathbf{W}})} \tag{5.75}$$

This transfer function consists of a cascade of the lag filter with the all-pole filter representing the first-order Markov process. The solution for the variance

of the lag terms l_i can be computed through the inverse $\mathcal{Z}$-transform as follows:

$$E[l_i^2(k)] = \frac{1}{2\pi j} \oint H(z)H(z^{-1})\sigma_{\mathbf{W}}^2 z^{-1}\, dz$$

The integral above can be solved using the residue theorem as previously shown in the LMS case.

Using the solution for the variance of the lag terms for values of $\lambda_{\mathbf{W}} < 1$, it can be easily shown that

$$\xi_{lag} \approx \frac{(N+1)\sigma_{\mathbf{W}}^2\sigma_x^2}{\lambda_{\mathbf{W}}(1+\lambda^2) - \lambda(1+\lambda_{\mathbf{W}}^2)} \left(\frac{1-\lambda}{1+\lambda} - \frac{1-\lambda_{\mathbf{W}}}{1+\lambda_{\mathbf{W}}}\right) \tag{5.76}$$

If $\lambda = 1$ and $\lambda_{\mathbf{W}} \approx 1$, the MSE due to lag tends to infinity indicating that the RLS algorithm in this case cannot track any change in the environment. On the other hand, for $\lambda < 1$ the algorithm can track variations in the environment leading to an excess of MSE that depends on the variance of the optimal coefficient disturbance and on the input signal variance.

For $\lambda_{\mathbf{W}} \approx 1$ and $\lambda \approx 1$, it is possible to rewrite equation (5.76) as

$$\xi_{lag} \approx (N+1)\frac{\sigma_{\mathbf{W}}^2}{2(1-\lambda)}\sigma_x^2 \tag{5.77}$$

The total excess of MSE accounting for the lag and finite memory is given by

$$\xi_{total} \approx (N+1)\left[\frac{1-\lambda}{1+\lambda}\xi_{min} + \frac{\sigma_{\mathbf{W}}^2\sigma_x^2}{2(1-\lambda)}\right] \tag{5.78}$$

By differentiating the above equation with respect to λ and setting the result to zero, an optimum value for λ can be found that yields minimum excess of MSE.

$$\lambda_{opt} = \frac{1 - \frac{\sigma_{\mathbf{W}}\sigma_x}{2\sigma_n}}{1 + \frac{\sigma_{\mathbf{W}}\sigma_x}{2\sigma_n}} \tag{5.79}$$

In the equation above $\sigma_n = \sqrt{\xi_{min}}$. Note that the optimal value of λ does not depend on the adaptive filter order N, and can be used when it falls in an acceptable range of values for λ. Also this value is optimum only when quantization effects are not important and the first-order Markov model is a good approximation for the nonstationarity of the desired signal.

5.5 QUANTIZATION EFFECTS

When implemented with finite-precision arithmetic, the conventional RLS algorithm behavior can differ significantly from what is expected under infinite precision. A series of inconvenient effects can show up in the practical implementation of the conventional RLS algorithm, such as divergence and freezing in the updating of the adaptive filter coefficients. In this section, several aspects of the finite-wordlength effects in the RLS algorithm are discussed for the cases of implementation with fixed- and floating-point arithmetic [3]-[6], [11]-[14].

5.5.1 Error Descriptions

All the elements of matrices and vectors in the RLS algorithm will deviate from their correct values due to quantization effects. The error generated in any individual quantization is considered to be a zero-mean random variable that is independent of any other error and quantities related to the adaptive filter algorithm. The variances of these errors depend on the type of quantization and arithmetic that will be applied in the algorithm implementation.

The errors in the quantities related to the conventional RLS algorithm are defined by

$$n_{e'}(k) = e'(k) - e'(k)_Q \tag{5.80}$$

$$\mathbf{n}_{\Psi}(k) = \mathbf{S}_D(k-1)_Q\mathbf{x}(k) - [\mathbf{S}_D(k-1)_Q\mathbf{x}(k)]_Q \tag{5.81}$$

$$\mathbf{N}_{\mathbf{S}_D}(k) = \mathbf{S}_D(k) - \mathbf{S}_D(k)_Q \tag{5.82}$$

$$\mathbf{n}_{\mathbf{w}}(k) = \mathbf{w}(k) - \mathbf{w}(k)_Q \tag{5.83}$$

$$n_y(k) = y(k) - y(k)_Q \tag{5.84}$$

$$n_e(k) = e(k) - e(k)_Q \tag{5.85}$$

where the subscript Q denotes the quantized form of the given matrix, vector, or scalar.

It is assumed that the input signal and desired signal suffer no quantization, so only quantizations of internal computations are taken into account. With the definitions above, the following relations describe the computational error in some quantities of interest related to the RLS algorithm:

$$e'(k)_Q = d(k) - \mathbf{x}^T(k)\mathbf{w}(k-1)_Q - n_{e'}(k) \tag{5.86}$$

$$\mathbf{w}(k)_Q = \mathbf{w}(k-1)_Q + \mathbf{S}_D(k)_Q\mathbf{x}(k)e'(k)_Q - \mathbf{n}_{\mathbf{w}}(k) \tag{5.87}$$

where $n_{e'}(k)$ is the noise sequence due to quantization in the inner product $\mathbf{x}^T(k)\mathbf{w}(k-1)_Q$ and $\mathbf{n_w}(k)$ is a noise vector due to quantization in the product $\mathbf{S}_D(k)_Q\mathbf{x}(k)e'(k)_Q$.

The development here is intended to study the algorithm behavior when the internal signals, vectors, and matrices are available in quantized form as happens in a practical implementation. This means that, for example in Algorithm 5.2, all the information needed from the previous time interval $(k-1)$ to update the adaptive filter in the instant k is available in quantized form.

Now we can proceed with the analysis of the deviation in the coefficient vector generated by the quantization error. By defining

$$\Delta\mathbf{w}(k)_Q = \mathbf{w}(k)_Q - \mathbf{w}_o \tag{5.88}$$

and considering that

$$d(k) = \mathbf{x}^T(k)\mathbf{w}_o + n(k)$$

then it follows that

$$e'(k)_Q = -\mathbf{x}^T(k)\Delta\mathbf{w}(k-1)_Q - n_{e'}(k) + n(k) \tag{5.89}$$

and

$$\begin{aligned}\Delta\mathbf{w}(k)_Q &= \Delta\mathbf{w}(k-1)_Q \\ &\quad +\mathbf{S}_D(k)_Q\mathbf{x}(k)[-\mathbf{x}^T(k)\Delta\mathbf{w}(k-1)_Q - n_{e'}(k) + n(k)] \\ &\quad -\mathbf{n_w}(k)\end{aligned} \tag{5.90}$$

Equation (5.90) can be rewritten as follows:

$$\Delta\mathbf{w}(k)_Q = [\mathbf{I} - \mathbf{S}_D(k)_Q\mathbf{x}(k)\mathbf{x}^T(k)]\Delta\mathbf{w}(k-1)_Q + \mathbf{n'_w}(k) \tag{5.91}$$

where

$$\mathbf{n'_w}(k) = \mathbf{S}_D(k)_Q\mathbf{x}(k)[n(k) - n_{e'}(k)] - \mathbf{n_w}(k) \tag{5.92}$$

The solution of equation (5.91) can be calculated as

$$\begin{aligned}\Delta\mathbf{w}(k)_Q &= \prod_{i=0}^{k}\left[\mathbf{I} - \mathbf{S}_D(i)_Q\mathbf{x}(i)\mathbf{x}^T(i)\right]\Delta\mathbf{w}(-1)_Q \\ &\quad +\sum_{i=0}^{k}\left\{\prod_{j=i+1}^{k}\left[\mathbf{I} - \mathbf{S}_D(j)_Q\mathbf{x}(j)\mathbf{x}^T(j)\right]\right\}\mathbf{n'_w}(i)\end{aligned} \tag{5.93}$$

Algorithm 5.3

RLS algorithm including quantization

Initialization

$\mathbf{S}_D(-1) = \delta\mathbf{I}$

where δ can be the inverse of an estimate of the input signal power.

$\mathbf{x}(-1) = \mathbf{w}(-1) = [0\ 0\ \ldots\ 0]^T$

Do for $k \geq 0$

$e'(k)_Q = d'(k) - \mathbf{x}^T(k)\mathbf{w}(k-1)_Q - n_{e'}(k) + n(k)$

$\mathbf{\Psi}(k)_Q = \mathbf{S}_D(k-1)_Q\mathbf{x}(k) - \mathbf{n}_{\mathbf{\Psi}}(k)$

$\mathbf{S}_D(k)_Q = \frac{1}{\lambda}\left[\mathbf{S}_D(k-1)_Q - \frac{\mathbf{\Psi}(k)_Q\mathbf{\Psi}^T(k)_Q}{\lambda+\mathbf{\Psi}^T(k)_Q\mathbf{x}(k)}\right] - \mathbf{N}_{\mathbf{S}_D}(k)$

$\mathbf{w}(k)_Q = \mathbf{w}(k-1)_Q + e'(k)_Q\mathbf{S}_D(k)_Q\mathbf{x}(k) - \mathbf{n}_{\mathbf{w}}(k)$

If necessary compute

$y(k)_Q = \mathbf{w}^T(k)_Q\mathbf{x}(k) - n_y(k)$

$e(k)_Q = d(k) - y_Q(k)$

□

where in the last term of the equation above for $i = k$, we consider that

$$\prod_{j=k+1}^{k} [\cdot] = 1$$

Now, if we rewrite Algorithm 5.2 taking into account that any calculation in the present updating generates quantization noise, we obtain Algorithm 5.3 that describes the RLS algorithm with quantization and additional noise taken into account. Notice that Algorithm 5.3 is not a new algorithm.

5.5.2 Error Models for Fixed-Point Arithmetic

In the case of fixed-point arithmetic, with rounding assumed for quantization, the error after each product can be modeled as a zero-mean stochastic process, with variance given by [15]

$$\sigma^2 = \frac{2^{-2b}}{12} \tag{5.94}$$

where b is the number of bits after the sign bit. Here it is assumed that the number of bits after the sign bit for quantities representing signals and filter coefficients are different and given by b_d and b_c, respectively. It is also assumed that the internal signals are properly scaled, so that no overflow occurs during the computations, and that the signal values are between -1 and $+1$. If in addition independence between errors is assumed, each element in equations (5.80) to (5.85) is in average zero. The respective covariance matrices are given by

$$E[n_{e'}^2(k)] = E[n_e^2(k)] = \sigma_e^2 \tag{5.95}$$

$$E[\mathbf{N}_{\mathbf{S}_D}(k)\mathbf{N}^T{}_{\mathbf{S}_D}(k)] = \sigma_{\mathbf{S}_D}^2\mathbf{I} \tag{5.96}$$

$$E[\mathbf{n}_{\mathbf{W}}(k)\mathbf{n}_{\mathbf{W}}^T(k)] = \sigma_{\mathbf{W}}^2\mathbf{I} \tag{5.97}$$

$$E[\mathbf{\Psi}(k)\mathbf{\Psi}^T(k)] = \sigma_{\mathbf{\Psi}}^2\mathbf{I} \tag{5.98}$$

$$E[n_y^2(k)] = \sigma_y^2 \tag{5.99}$$

If distinction is made between data and coefficient wordlengths, the noise variances of data and coefficients are respectively given by

$$\sigma_e^2 = \sigma_y^2 = \gamma\frac{2^{-2b_d}}{12} \tag{5.100}$$

$$\sigma_{\mathbf{W}}^2 = \gamma'\frac{2^{-2b_c}}{12} \tag{5.101}$$

where $\gamma' = \gamma = 1$ if the quantization is performed after addition, i.e., the products are performed in full precision and the quantization is applied only after all the additions in the inner product are finished. For quantization after each product, then $\gamma = N + 1$ and $\gamma' = N + 2$, since each quantization in the partial product generates an independent noise, and the number of products in the error computation is $N + 1$ whereas in the coefficient computation it is $N + 2$.

As an illustration, it is shown how to calculate the value of the variance $\sigma^2_{\mathbf{S}_D}$ when making some simple assumptions. The value of $\sigma^2_{\mathbf{S}_D}$ depends on how the computations to generate $\mathbf{S}_D(k)$ are performed. Assume the multiplications and divisions are performed with the same wordlength and that the needed divisions are performed once, followed by the corresponding scalar matrix product. Also, assuming the inner product quantizations are performed after the addition, each element of the matrix $\mathbf{S}_D(k)_Q$ requires five multiplications considering that $1/\lambda$ is prestored. The diagonal elements of equation (5.96) consist of $N+1$ noise autocorrelations, each with variance $5\sigma^2_\psi$. The desired result is then given by

$$\sigma^2_{\mathbf{S}_D} = 5(N+1)\sigma^2_\psi \tag{5.102}$$

where σ^2_ψ is the variance of each multiplication error.

5.5.3 Coefficient-Error-Vector Covariance Matrix

Assume that the quantization signals $n_{e'}(k), n(k)$, and the vector $\mathbf{n_w}(k)$ are all independent of the data, filter coefficients, and of each other. Also, assuming that these errors are all zero-mean stochastic processes, the covariance matrix of the coefficient-error vector given by $E[\Delta\mathbf{w}(k)_Q\Delta\mathbf{w}^T(k)_Q]$ can be easily derived from equations (5.91) and (5.92)

$$\begin{aligned} cov[\Delta\mathbf{w}(k)_Q] &= E[\Delta\mathbf{w}(k)_Q\Delta\mathbf{w}^T(k)_Q] \\ &= E\{[\mathbf{I}-\mathbf{S}_D(k)_Q\mathbf{x}(k)\mathbf{x}^T(k)]\Delta\mathbf{w}(k-1)_Q\Delta\mathbf{w}^T(k-1)_Q \\ &\quad [\mathbf{I}-\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{S}_D(k)_Q]\} \\ &\quad +E[\mathbf{S}_D(k)_Q\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{S}_D(k)_Q]E[n^2(k)] \\ &\quad +E[\mathbf{S}_D(k)_Q\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{S}_D(k)_Q]E[n^2_{e'}(k)] \\ &\quad +E[\mathbf{n_w}(k)\mathbf{n}^T_{\mathbf{w}}(k)] \end{aligned} \tag{5.103}$$

The above equation can be approximated in the steady state, where each term in the right-hand side will be considered separately. It should be noted that during the derivations it is implicitly assumed that the algorithm follows closely the behavior of its infinite precision counterpart. This assumption can always be considered as true if the wordlengths used are sufficiently long. However, under short wordlength implementation this assumption might not be true as will be discussed later on.

Term 1:

The elements of $\Delta\mathbf{w}(k-1)_Q$ can be considered independent of $\mathbf{S}_D(k)_Q$ and $\mathbf{x}(k)$. In this case, the first term in equation (5.103) can be expressed as

$$\begin{aligned}\mathbf{T}_1 &= cov[\Delta\mathbf{w}(k-1)_Q] - cov[\Delta\mathbf{w}(k-1)_Q]E[\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{S}_D(k)_Q] \\ &\quad -E[\mathbf{S}_D(k)_Q\mathbf{x}(k)\mathbf{x}^T(k)]cov[\Delta\mathbf{w}(k-1)_Q] \\ &\quad +E\{\mathbf{S}_D(k)_Q\mathbf{x}(k)\mathbf{x}^T(k)cov[\Delta\mathbf{w}(k-1)_Q]\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{S}_D(k)_Q\}\end{aligned} \tag{5.104}$$

If it is recalled that $\mathbf{S}_D(k)_Q$ is the unquantized $\mathbf{S}_D(k)$ matrix disturbed by a noise matrix that is uncorrelated to the input signal vector, then in order to compute the second and third terms of $\mathbf{T}_1$ it suffices to calculate

$$E[\mathbf{S}_D(k)\mathbf{x}(k)\mathbf{x}^T(k)] \approx E\left[\mathbf{S}_D(k)\right] E\left[\mathbf{x}(k)\mathbf{x}^T(k)\right] \tag{5.105}$$

where the approximation is justified by the fact that $\mathbf{S}_D(k)$ is slowly varying as compared to $\mathbf{x}(k)$ when $\lambda \to 1$. Using equation (5.44) it follows that

$$E\left[\mathbf{S}_D(k)\mathbf{x}(k)\mathbf{x}^T(k)\right] \approx \frac{1-\lambda}{1-\lambda^{k+1}}\mathbf{I} \tag{5.106}$$

Now we need to use stronger assumptions for $\mathbf{S}_D(k)$ than those considered in the equation above. If the matrix $E[\mathbf{S}_D(k)_Q]$ is assumed to be approximately constant for large k (see the discussions around equation (5.43)), the last term in $\mathbf{T}_1$ can be approximated by

$$\begin{aligned}&E\{\mathbf{S}_D(k)_Q\mathbf{x}(k)\mathbf{x}^T(k)cov[\Delta\mathbf{w}(k+1)_Q]\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{S}_D(k)_Q\} \\ &\approx E[\mathbf{S}_D(k)_Q]E\{\mathbf{x}(k)\mathbf{x}^T(k)cov[\Delta\mathbf{w}(k-1)_Q]\mathbf{x}(k)\mathbf{x}^T(k)]\}E[\mathbf{S}_D(k)_Q]\end{aligned} \tag{5.107}$$

If it is further assumed that the elements of the input signal vector are jointly Gaussian, then each element of the middle term in the last equation can be given by

$$\begin{aligned}&E\{\mathbf{x}(k)\mathbf{x}^T(k)cov[\Delta\mathbf{w}(k-1)_Q]\mathbf{x}(k)\mathbf{x}^T(k)\}_{i,j} \\ &= \sum_{m=0}^{N}\sum_{l=0}^{N} cov[\Delta\mathbf{w}(k-1)_Q]_{ml}E[x_i(k)x_m(k)x_l(k)x_j(k)] \\ &= 2\{\mathbf{R}cov[\Delta\mathbf{w}(k-1)_Q]\mathbf{R}\}_{i,j} + [\mathbf{R}]_{i,j}tr\{\mathbf{R}cov[\Delta\mathbf{w}(k-1)_Q]\}\end{aligned} \tag{5.108}$$

where $[\cdot]_{i,j}$ denotes the ith, jth element of the matrix $[\cdot]$. It then follows that

$$\begin{aligned} & E\{\mathbf{x}(k)\mathbf{x}^T(k)cov[\Delta\mathbf{w}(k-1)_Q]\mathbf{x}(k)\mathbf{x}^T(k)\} \\ = \; & 2\mathbf{R}cov[\Delta\mathbf{w}(k-1)_Q]\mathbf{R} + \mathbf{R}tr\{\mathbf{R}cov[\Delta\mathbf{w}(k-1)_Q]\} \end{aligned} \tag{5.109}$$

The last term of $\mathbf{T}_1$ in equation (5.104) , after simplified, yields

$$\begin{aligned} & 2\left(\frac{1-\lambda}{1-\lambda^{k+1}}\right)^2 cov[\Delta\mathbf{w}(k-1)_Q] \\ & + \left(\frac{1-\lambda}{1-\lambda^{k+1}}\right)^2 tr\{\mathbf{R}cov[\Delta\mathbf{w}(k-1)_Q]\}\mathbf{R}^{-1} \\ & + E\{\mathbf{N}_{\mathbf{S}_D}(k)\mathbf{x}(k)\mathbf{x}^T(k)cov[\Delta\mathbf{w}(k-1)_Q]\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{N}_{\mathbf{S}_D}(k)\} \end{aligned} \tag{5.110}$$

After a few manipulations, it can be shown that the third term in the equation above is nondiagonal because $\mathbf{N}_{\mathbf{S}_D}(k)$ is symmetric for the RLS algorithm described in Algorithm 5.3. On the other hand, if the matrix $\mathbf{R}$ is diagonal dominant, that is in general the case, the third term of (5.110) becomes approximately diagonal and given by

$$\mathbf{T}_S(k) \approx (N+1)\sigma^2_{\mathbf{S}_D}\sigma_x^4 tr\{cov[\Delta\mathbf{w}(k-1)_Q]\}\mathbf{I} \tag{5.111}$$

where σ_x^2 is the variance of the input signal. This term, which is proportional to a quantization noise variance, can actually be neglected in the analysis, since it has in general much smaller norm then the remaining terms in $\mathbf{T}_1$.

Terms 2 and 3:

Using the same arguments applied before, such as $\mathbf{S}_D(k)$ is almost fixed as $\lambda \to 1$, then the main result required to calculate the terms 2 and 3 of equation (5.103) is approximately given by

$$\begin{aligned} & E[\mathbf{S}_D(k)_Q\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{S}_D(k)_Q] \\ \approx \; & E[\mathbf{S}_D(k)]\mathbf{R}E[\mathbf{S}_D(k)] + E[\mathbf{N}_{\mathbf{S}_D}(k)\mathbf{R}\mathbf{N}_{\mathbf{S}_D}(k)] \\ \approx \; & \left(\frac{1-\lambda}{1-\lambda^{k+1}}\right)^2 \mathbf{R}^{-1} \end{aligned} \tag{5.112}$$

where the term $E[\mathbf{N}_{\mathbf{S}_D}(k)\mathbf{R}\mathbf{N}_{\mathbf{S}_D}(k)]$ can be neglected because it is in general much smaller than the remaining term. In addition, it will be multiplied by a small variance when equation (5.112) is replaced back in equation (5.103).

From equations (5.103), (5.107), (5.112), (5.95), (5.97), and (5.101) it follows that

$$\begin{aligned} cov[\Delta\mathbf{w}(k)_Q] &= [1-2\left(\frac{1-\lambda}{1-\lambda^{k+1}}\right)+2\left(\frac{1-\lambda}{1-\lambda^{k+1}}\right)^2]cov[\Delta\mathbf{w}(k-1)_Q] \\ &+\left(\frac{1-\lambda}{1-\lambda^{k+1}}\right)^2 tr\{\mathbf{R}cov[\Delta\mathbf{w}(k-1)_Q]\}\mathbf{R}^{-1} \\ &+\left(\frac{1-\lambda}{1-\lambda^{k+1}}\right)^2 (\sigma_n^2+\sigma_e^2)\mathbf{R}^{-1}+\sigma_{\mathbf{w}}^2\mathbf{I} \end{aligned} \tag{5.113}$$

Now, by considering in equation (5.113) that in the steady state $cov[\Delta\mathbf{w}(k)_Q] \approx cov[\Delta\mathbf{w}(k-1)_Q]$, multiplying the resulting expression by $\mathbf{R}$, and calculating the trace of the final equation, it can be shown that

$$tr\{\mathbf{R}cov[\Delta\mathbf{w}(k-1)_Q]\} \approx \frac{(1-\lambda)^2(N+1)(\sigma_n^2+\sigma_e^2)+\sigma_{\mathbf{w}}^2 tr(\mathbf{R})}{(1-\lambda)[2\lambda-(1-\lambda)(N+1)]} \tag{5.114}$$

where it was considered that $\lambda^{k+1} \to 0$. Replacing the equation (5.114) in (5.113), and computing the steady-state solution the following equation results

$$\begin{aligned} cov[\Delta\mathbf{w}(k)_Q] &\approx \frac{(1-\lambda)(\sigma_n^2+\sigma_e^2)}{2\lambda-(1-\lambda)(N+1)}\mathbf{R}^{-1} \\ &+\frac{(1-\lambda)tr(\mathbf{R})\mathbf{R}^{-1}+[2\lambda-(1-\lambda)(N+1)]\mathbf{I}}{2(1-\lambda)\lambda[2\lambda-(1-\lambda)(N+1)]}\sigma_{\mathbf{w}}^2 \end{aligned} \tag{5.115}$$

Finally, if the trace of the above equation is calculated considering that $x(k)$ is a Gaussian noise with variance σ_x^2, and that $2\lambda \gg (1-\lambda)(N+1)$ for $\lambda \to 1$, the resulting expected value of $||\Delta\mathbf{w}(k)_Q||^2$ is

$$E[||\Delta\mathbf{w}(k)_Q||^2] \approx \frac{(1-\lambda)(N+1)}{2\lambda}\frac{\sigma_n^2+\sigma_e^2}{\sigma_x^2}+\frac{(N+1)\sigma_{\mathbf{w}}^2}{2\lambda(1-\lambda)} \tag{5.116}$$

As can be noted if the value of λ is very close to one, the square errors in the tap coefficients tend to increase and to become more dependent of the tap coefficient wordlengths. On the other hand, if λ is not close to one, in general for fast tracking purposes, the effects of the additive noise and data wordlength become more disturbing to the coefficient square errors. The optimum value for λ, as far as quantization effects are concerned, can be derived by calculating the derivative of $E||\Delta\mathbf{w}(k)_Q||^2]$ with respect to λ and setting the result to zero

$$\lambda_o = 1-\frac{\sigma_{\mathbf{w}}\sigma_x}{\sqrt{\sigma_n^2+\sigma_e^2}} \tag{5.117}$$

By noting that $\frac{1-\lambda}{1-\lambda^{k+1}}$ should be replaced by $\frac{1}{k+1}$ when $\lambda = 1$, it can be shown from equation (5.113) that the algorithm tends to diverge when $\lambda = 1$, since in this case $||cov[\Delta \mathbf{w}(k)_Q]||$ is growing with k.

5.5.4 Algorithm Stop

In some cases the adaptive filter tap coefficients may stop adapting due to quantization effects. In particular, the conventional RLS algorithm will freeze when the coefficient updating term is not representable with the available wordlength. This occurs when its modulus is smaller than half the value of the least significant bit, i.e.,

$$|e'(k)_Q \mathbf{S}_D(k)_Q \mathbf{x}(k)|_i < 2^{-b_c-1} \tag{5.118}$$

where $|\ |_i$ denotes the modulus of the ith component. Equivalently it can be concluded that updating will be stopped if

$$\begin{aligned} & E(e'(k)_Q^2) E[|\mathbf{S}_D(k)_Q \mathbf{x}(k) \mathbf{x}^T(k) \mathbf{S}_D(k)_Q|_{ii}] \\ \approx & \left(\frac{1-\lambda}{1-\lambda^{k+1}}\right)^2 \frac{\sigma_e^2 + \sigma_n^2}{\sigma_x^2} < 2^{-2b_c-2} \end{aligned} \tag{5.119}$$

where $x(k)$ was considered a Gaussian white noise with variance σ_x^2, and the following approximation was made $E[e'(k)_Q^2] \approx \sigma_e^2 + \sigma_n^2$.

For a given coefficient wordlength b_c the algorithm can always be kept updating if

$$\lambda < 1 - 2^{-b_c-1} \frac{\sigma_x}{\sqrt{\sigma_e^2 + \sigma_n^2}} \tag{5.120}$$

On the other hand, if the condition above is not satisfied, it can be expected that the algorithm will stop updating in

$$k \approx \frac{\sqrt{\sigma_e^2 + \sigma_n^2}}{\sigma_x} 2^{b_c+1} - 1 \tag{5.121}$$

iterations for $\lambda = 1$, and

$$k \approx \frac{\ln[(\lambda - 1)\frac{\sqrt{\sigma_e^2+\sigma_n^2}}{\sigma_x} 2^{b_c+1} + 1]}{\ln \lambda} - 1 \tag{5.122}$$

iterations for $\lambda < 1$.

In the case $\lambda = 1$ the algorithm always stops updating. If σ_n^2 and b_c are not large, any steady-state analysis for the RLS algorithm when $\lambda = 1$ does not apply, since the algorithm stops prematurely. Because of that, the norm of the covariance of $\Delta\mathbf{w}(k)_Q$ does not become unbounded.

5.5.5 Mean-Square Error

The MSE in the conventional RLS algorithm in the presence of quantization noise is given by

$$\xi(k)_Q = E[e^2(k)_Q] \tag{5.123}$$

By recalling that $e(k)_Q$ can be expressed as

$$e(k)_Q = -\mathbf{x}^T(k)\Delta\mathbf{w}(k)_Q - n_e(k) + n(k) \tag{5.124}$$

it then follows that

$$\begin{aligned} \xi(k)_Q &= E[\mathbf{x}^T(k)\Delta\mathbf{w}(k)_Q\mathbf{x}^T(k)\Delta\mathbf{w}(k)_Q] + \sigma_e^2 + \sigma_n^2 \\ &= E\{tr[\mathbf{x}(k)\mathbf{x}^T(k)\Delta\mathbf{w}(k)_Q\Delta\mathbf{w}^T(k)_Q]\} + \sigma_e^2 + \sigma_n^2 \\ &= tr\{\mathbf{R}cov[\Delta\mathbf{w}(k)_Q]\} + \sigma_e^2 + \sigma_n^2 \end{aligned} \tag{5.125}$$

By replacing equation (5.114) in (5.125), it can be concluded that

$$\xi(k)_Q = \frac{(1-\lambda)^2(N+1)(\sigma_n^2+\sigma_e^2) + \sigma_{\mathbf{w}}^2 tr\mathbf{R}}{(1-\lambda)[2\lambda-(1-\lambda)(N+1)]} + \sigma_n^2 + \sigma_e^2 \tag{5.126}$$

If it is again assumed that $x(k)$ is a Gaussian noise with variance σ_x^2 and that $2\lambda \gg (1-\lambda)(N+1)$ for $\lambda \to 1$, the MSE expression can be simplified to

$$\xi(k)_Q \approx \sigma_n^2 + \sigma_e^2 + \frac{(N+1)\sigma_{\mathbf{w}}^2\sigma_x^2}{2\lambda(1-\lambda)} \tag{5.127}$$

5.5.6 Fixed-Point Implementation Issues

The implementation of the conventional RLS algorithm in fixed-point arithmetic must consider the possibility of occurrence of overflow and underflow during the computations. In general, some scaling must be performed in certain quantities of the RLS algorithm to avoid undesired behavior due to overflow and underflow. The scaling procedure must be applied in almost all computations required in the conventional RLS algorithm [6], increasing the computational complexity and/or implementation control by a large amount. A possible solution is to leave

enough room in the integer and fractional parts of the number representation, in order to avoid frequent overflows and underflows and also avoid the use of cumbersome scaling strategies. In other words, a fixed-point implementation does require a reasonable number of bits to represent each quantity.

The error propagation analysis can be performed by studying the behavior of the difference between each quantity of the algorithm calculated in infinite precision and finite precision. This analysis allows the detection of divergence of the algorithm due to quantization error accumulation. The error propagation analysis for the conventional RLS algorithm reveals divergence behavior linked to the fact that $\mathbf{S}_D(k)$ loses the positive definiteness property [6]. The main factors contributing to divergence are:

- Large maximum eigenvalue in the matrix $\mathbf{R}$ that amplifies some terms in propagation error of the $\mathbf{S}_D(k)$ matrix. In this case, $\mathbf{S}_D(k)$ might have a small minimum eigenvalue, being as consequence "almost" singular.
- A small number of bits used in the calculations increases the roundoff noise contributing to divergence.
- The forgetting factor when small, turns the memory of the algorithm short, making the matrix $\mathbf{S}_D(k)$ deviate from its expected steady-state value and more likely to lose the positive definiteness property.

Despite these facts, the conventional RLS algorithm can be implemented without possibility of divergence if some special quantization strategies for the internal computations are used [6]. These quantization strategies, along with adaptive scaling strategies, must be used when implementing the conventional RLS algorithm in fixed-point arithmetic with short wordlength.

5.5.7 Floating-Point Arithmetic Implementation

In this section, a succinct analysis of the quantization effects in the conventional RLS algorithm when implemented in floating-point arithmetic is presented. Most of the derivations are given in the appendix and follow closely the procedure of the fixed-point analysis.

In floating-point arithmetic, quantization errors are injected after multiplication and addition operations, and are modeled as follows [16]:

$$fl[a+b] = a+b-(a+b)n_a \tag{5.128}$$

$$fl(a\cdot b) = a\cdot b - a\cdot b\cdot n_p \tag{5.129}$$

where n_a and n_p are zero-mean random variables that are independent of any other errors. Their variances are given by

$$\sigma^2_{n_p} \approx 0.18\ 2^{-2b} \tag{5.130}$$

and

$$\sigma^2_{n_a} < \sigma^2_{n_p} \tag{5.131}$$

where b is the number of bits in the mantissa representation.

The quantized error and the quantized coefficient vector are given by

$$e'(k)_Q = d'(k) - \mathbf{x}^T(k)\mathbf{w}(k-1)_Q - n_{e'}(k) + n(k) \tag{5.132}$$

$$\mathbf{w}(k)_Q = \mathbf{w}(k-1)_Q + \mathbf{S}_D(k)_Q\mathbf{x}(k)e'(k)_Q - \mathbf{n}_\mathbf{W}(k) \tag{5.133}$$

where $n_{e'}(k)$ and $\mathbf{n}_\mathbf{W}(k)$ represent computational errors and their expressions are given in the appendix. Since $\mathbf{n}_\mathbf{W}(k)$ is a zero-mean vector, it is shown in the appendix that in average $\mathbf{w}(k)_Q$ tends to $\mathbf{w}_o$. Also, it can be shown that

$$\begin{aligned}\Delta\mathbf{w}(k)_Q = {} & [\mathbf{I} - \mathbf{S}_D(k)_Q\mathbf{x}(k)\mathbf{x}^T(k) + \mathbf{N}_{\Delta\mathbf{W}}(k)]\Delta\mathbf{w}(k-1) \\ & + \mathbf{N'}_a(k)\mathbf{w}_o + \mathbf{S}_D(k)_Q\mathbf{x}(k)[n(k) - n_{e'}(k)]\end{aligned} \tag{5.134}$$

where $\mathbf{N}_{\Delta\mathbf{W}}(k)$ combines several quantization noise effects as discussed in the appendix and $\mathbf{N'}_a(k)$ is a diagonal noise matrix that models the noise generated in the vector addition required to update $\mathbf{w}(k)_Q$.

The covariance matrix of $\Delta\mathbf{w}(k)_Q$ can be calculated through the same procedure previously used in the fixed-point case, resulting in

$$\begin{aligned}cov[\Delta\mathbf{w}(k)_Q] \approx {} & \frac{(1-\lambda)(\sigma_n^2+\sigma_e^2)\mathbf{R}^{-1}}{2\lambda-(1-\lambda)(N+1)} \\ & + \frac{(1-\lambda)\mathbf{R}^{-1}tr\{\mathbf{R}diag[w_{oi}^2]\} + [2\lambda-(1-\lambda)(N+1)]diag[w_{oi}^2]}{2(1-\lambda)\lambda[2\lambda-(1-\lambda)(N+1)]}\sigma^2_{n'_a}\end{aligned} \tag{5.135}$$

where $\mathbf{N}_{\mathbf{S}_D}(k)$ of equation (5.82) and $\mathbf{N}_{\Delta\mathbf{W}}(k)$ were considered negligible as compared to the remaining matrices multiplying $\Delta\mathbf{w}(k-1)$ in equation (5.134),

and $\sigma^2_{n'_a}$ is given by equation (5.131). The term $diag[w^2_{oi}]$ represents a diagonal matrix formed with the squared elements of $\mathbf{w}_o$.

The expected value of $||\Delta\mathbf{w}(k)_Q||^2$ in the floating-point case is approximately given by

$$E[||\Delta\mathbf{w}(k)_Q||^2] \approx \frac{(1-\lambda)(N+1)}{2\lambda}\frac{\sigma_n^2+\sigma_{e'}^2}{\sigma_x^2} + \frac{1}{2\lambda(1-\lambda)}||\mathbf{w}_o||^2\sigma^2_{n'_a} \quad (5.136)$$

where it was considered that $x(k)$ is a Gaussian noise with variance σ_x^2 and that $2\lambda \gg (1-\lambda)(N+1)$ for $\lambda \to 1$. If the value of λ is very close to one, the squared errors in the tap coefficients tend to increase. Notice that the second term on the right-hand side of the equation above turns these errors more dependent on the precision of the vector addition of the taps updating. For λ not very close to one, the effects of the additive noise and data wordlength become more pronounced. In floating-point implementation, the optimal value of λ as far as quantization effects are concerned is given by

$$\lambda_o = 1 - \frac{\sigma_{n'_a}\sigma_x}{\sqrt{\sigma_n^2+\sigma_{e'}^2}}||\mathbf{w}_o|| \quad (5.137)$$

where this relation was obtained by calculating the derivative of equation (5.136) with respect to λ, and equalizing the result to zero in order to reach the value of λ that minimizes the $E[||\Delta\mathbf{w}(k)_Q||^2]$. For $\lambda = 1$, like in the fixed-point case, $||cov[\Delta\mathbf{w}(k)_Q]||$ is also a growing function that can make the conventional RLS algorithm diverge.

The algorithm may stop updating if

$$|e'(k)_Q\mathbf{S}_D(k)\mathbf{x}(k)|_i < 2^{-b_c-1}w_i(k) \quad (5.138)$$

where $|\ |_i$ is the modulus of the ith component and b_c is the number of bits in the mantissa of the coefficients representation. Following the same procedure to derive equation (5.119), we can infer that the updating will be stopped if

$$\left(\frac{1-\lambda}{1-\lambda^{k+1}}\right)^2 \frac{\sigma_{e'}^2+\sigma_n^2}{\sigma_x^2} < 2^{-2b_c-2}|w_{oi}|^2 \quad (5.139)$$

where w_{oi} is the ith element of $\mathbf{w}_o$.

The updating can be continued indefinitely if

$$\lambda < 1 - 2^{-b_c-1}\frac{\sigma_x|w_{oi}|}{\sqrt{\sigma_e^2+\sigma_n^2}} \quad (5.140)$$

In the case λ does not satisfy the condition above, the algorithm will stop updating the ith tap in approximately

$$k = \frac{\sqrt{\sigma_e^2 + \sigma_n^2}}{\sigma_x |w_{oi}|} - 1 \tag{5.141}$$

iterations for $\lambda = 1$, and

$$k \approx \frac{\ln[(\lambda - 1)\frac{\sqrt{\sigma_{e'}^2 + \sigma_n^2}}{\sigma_x |w_{oi}|} 2^{-b_c - 1} + 1]}{\ln \lambda} - 1 \tag{5.142}$$

iterations for $\lambda < 1$.

Following the same procedure as in the fixed-point implementation, it can be shown that the MSE in the floating-point case is given by

$$\begin{aligned} \xi(k)_Q &= tr\{\mathbf{R}cov[\Delta \mathbf{w}(k)_Q]\} + \sigma_e^2 + \sigma_n^2 \\ &\approx \frac{(1-\lambda)^2(N+1)(\sigma_n^2 + \sigma_e^2) + \sigma_{n_a'}^2 tr\{\mathbf{R}diag[w_{oi}^2]\}}{(1-\lambda)[2\lambda - (1-\lambda)(N+1)]} + \sigma_e^2 + \sigma_n^2 \end{aligned} \tag{5.143}$$

where σ_e^2 was considered equal to $\sigma_{e'}^2$. If $x(k)$ is a Gaussian noise with variance σ_x^2 and $2\lambda \gg (1-\lambda)(N+1)$ for $\lambda \to 1$, the MSE can be approximated by

$$\xi(k)_Q \approx \sigma_n^2 + \sigma_e^2 + \frac{||\mathbf{w}_o||^2 \sigma_{n_a'}^2 \sigma_x^2}{2\lambda(1-\lambda)} \tag{5.144}$$

Note that σ_e^2 has a somewhat complicated expression that is given in the appendix.

Finally, it should be mentioned that in floating-point implementations the matrix $\mathbf{S}_D(k)$ can also lose its positive definite property [13]. In [6], it was mentioned that if no interactions between errors is considered, preserving the symmetry of $\mathbf{S}_D(k)$ is enough to keep it positive definite. However, interactions between errors do exist in practice, so the conventional RLS algorithm can become unstable in floating-point implementations unless some special quantization procedures are employed in the actual implementation. An alternative is to use numerically stable RLS algorithms that will be discussed later on.

5.6 SIMULATION EXAMPLES

In this section, some adaptive filtering problems described in the last two chapters are solved using the conventional RLS algorithm presented in this chapter.

Example 5.3: System Identification Simulations

The conventional RLS algorithm was employed in the identification of the system described in the subsection (3.6.2). The forgetting factor was chosen $\lambda = 0.99$.

Solution:

In the first test, we address the sensitivity of the RLS algorithm to the eigenvalue spread of the input signal correlation matrix. The measured simulation results were obtained by ensemble averaging 200 independent runs. The learning curves of the mean-squared *a priori* error are depicted in Fig. 5.4, for different values of the eigenvalue spread. Also, the measured misadjustment in each example is given in Table 5.1. From these results, we conclude that the RLS algorithm is insensitive to the eigenvalue spread. It is worth mentioning at this point that the convergence speed of the RLS algorithm is affected by the choice of λ, since a smaller value of λ leads to faster convergence while increasing the misadjustment in stationary environment. In Table 5.1, the misadjustment predicted by theory calculated using the relation repeated below is given. As can be seen from this table the analytical results agree with those obtained through simulations.

$$M = (N+1)\frac{1-\lambda}{1+\lambda}(1+\frac{1-\lambda}{1+\lambda}\mathcal{K})$$

Table 5.1 Evaluation of the RLS Algorithm

$\frac{\lambda_{max}}{\lambda_{min}}$	Misadjustment	
	Experiment	Theory
1	0.04211	0.04020
20	0.04211	0.04020
80	0.04547	0.04020

The conventional RLS algorithm was implemented with finite-precision arithmetic, using fixed-point representation with 16, 12, and 10 bits respectively.

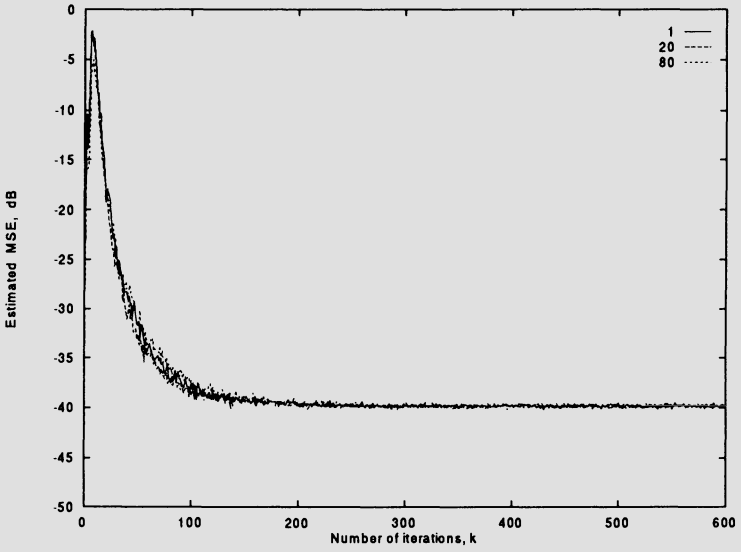


Figure 5.4 Learning curves for RLS algorithm for eigenvalue spreads: 1, 20, and 80; $\lambda = 0.99$.

The results presented were measured before any sign of instability was noticed. Table 5.2 summarizes the results of the finite-precision implementation of the conventional RLS algorithm. Note that in most cases there is a close agreement between the measurement results and those predicted by the equations given below.

$$E[||\Delta \mathbf{w}(k)_Q||^2] \approx \frac{(1-\lambda)(N+1)}{2\lambda} \frac{\sigma_n^2 + \sigma_e^2}{\sigma_x^2} + \frac{(N+1)\sigma_{\mathbf{w}}^2}{2\lambda(1-\lambda)}$$

$$\xi(k)_Q \approx \sigma_n^2 + \sigma_e^2 + \frac{(N+1)\sigma_{\mathbf{w}}^2 \sigma_x^2}{2\lambda(1-\lambda)}$$

For the simulations with 12 and 10 bits, the discrepancy between the measured and theoretical estimates of $E[||\Delta \mathbf{w}(k)_Q||^2]$ are caused by the freezing of some coefficients.

If the results presented here are compared with the results presented in Table 3.2 for the LMS, we notice that both the LMS and the RLS algorithms performed well in the finite-precision implementation. The reader should bear in mind that the conventional RLS algorithm requires an expensive strategy to keep the deterministic correlation matrix positive definite.

Table 5.2 Results of the Finite Precision Implementation of the RLS Algorithm

No. of bits	$\xi(k)_Q$		$E[\|\|\Delta\mathbf{w}(k)_Q\|\|^2]$	
	Experiment	Theory	Experiment	Theory
16	$1.566\ 10^{-3}$	$1.500\ 10^{-3}$	$6.013\ 10^{-5}$	$6.061\ 10^{-5}$
12	$1.522\ 10^{-3}$	$1.502\ 10^{-3}$	$3.128\ 10^{-5}$	$6.261\ 10^{-5}$
10	$1.566\ 10^{-3}$	$1.532\ 10^{-3}$	$6.979\ 10^{-5}$	$9.272\ 10^{-5}$

The simulations related to the experiment described for nonstationary environments were also performed. From the simulations we measured the total excess of MSE, and then compared the results to those obtained with the expression below.

$$\begin{aligned}\xi_{exc} \approx\ & (N+1)\frac{1-\lambda}{1+\lambda}(1+\frac{1-\lambda}{1+\lambda}\mathcal{K})\xi_{\min} \\ & +\frac{(N+1)\sigma_{\mathbf{w}}^2\sigma_x^2}{\lambda_{\mathbf{w}}(1+\lambda^2)-\lambda(1+\lambda_{\mathbf{w}}^2)}(\frac{1-\lambda}{1+\lambda}-\frac{1-\lambda_{\mathbf{w}}}{1+\lambda_{\mathbf{w}}})\end{aligned}$$

An attempt to use the optimal value of λ was made. The predicted optimal value however was too small and as a consequence $\lambda = 0.99$ was used. The measured excess of MSE was 0.0254, whereas the theoretical value predicted by the equation above was 0.0418. Note that the theoretical result is not as accurate as all the previous cases discussed so far, due to a number of approximations used in the analysis. However, the equation above provides a good indication of what is expected in the practical implementation. By choosing a smaller value for λ a better tracking performance is obtained, with the equation above is not as accurate.

□

Example 5.4: Signal Enhancement Simulations

We solved the same signal enhancement problem described in the subsection (4.6.1) with the conventional RLS and LMS algorithms.

Solution:

For the LMS algorithm, the convergence factor was chosen $\mu_{max}/5$. The resulting value for μ in the LMS case is 0.001, whereas $\lambda = 1.0$ was used for the RLS. The learning curves for the algorithms are shown in Fig. 5.5, where we can verify the faster convergence of the RLS algorithm. By plotting the output errors after convergence, we noted the large variance of the MSE for both algorithms. This result is due to the small signal to noise ratio, in this case. Fig. 5.6 depicts the output error and its DFT with 128 points for the RLS algorithm. In both cases, we can clearly detect the presence of the sinusoid.

□

5.7 CONCLUDING REMARKS

In this chapter, we introduced the conventional RLS algorithm and discussed various aspects related to its performance behavior. Much of the results obtained herein through mathematical analysis are valid for the whole class of RLS algorithms to be presented in the following chapters, except of course the finite-precision analysis since that depends on the form the internal calculations of each algorithm are performed. The analysis presented here is far from being complete. However, the main aspects of the conventional RLS have been addressed, such as: convergence behavior, tracking capabilities, and finite-wordlength effects. The interested reader should consult [17]-[19] for some further results.

From the analysis presented, one can conclude that the computational complexity and the stability in finite-precision implementations are two aspects to be concerned. When the elements of the input signal vector consist of delayed versions of the same signal, it is possible to derive a number of fast RLS algorithms whose computational complexity is of order N per output sample. Several different classes of these algorithms are presented in the following chapters. In all cases, their stability conditions in finite-precision implementation are briefly discussed.

For the general case where the elements of the input signal vector have different origins the QR-RLS algorithm is a good alternative to the conventional RLS algorithm. The stability of the QR-RLS algorithm can be easily guaranteed.

The conventional RLS algorithm is fully tested in a number of simulation results included in this chapter. These examples were meant to verify the theoretical

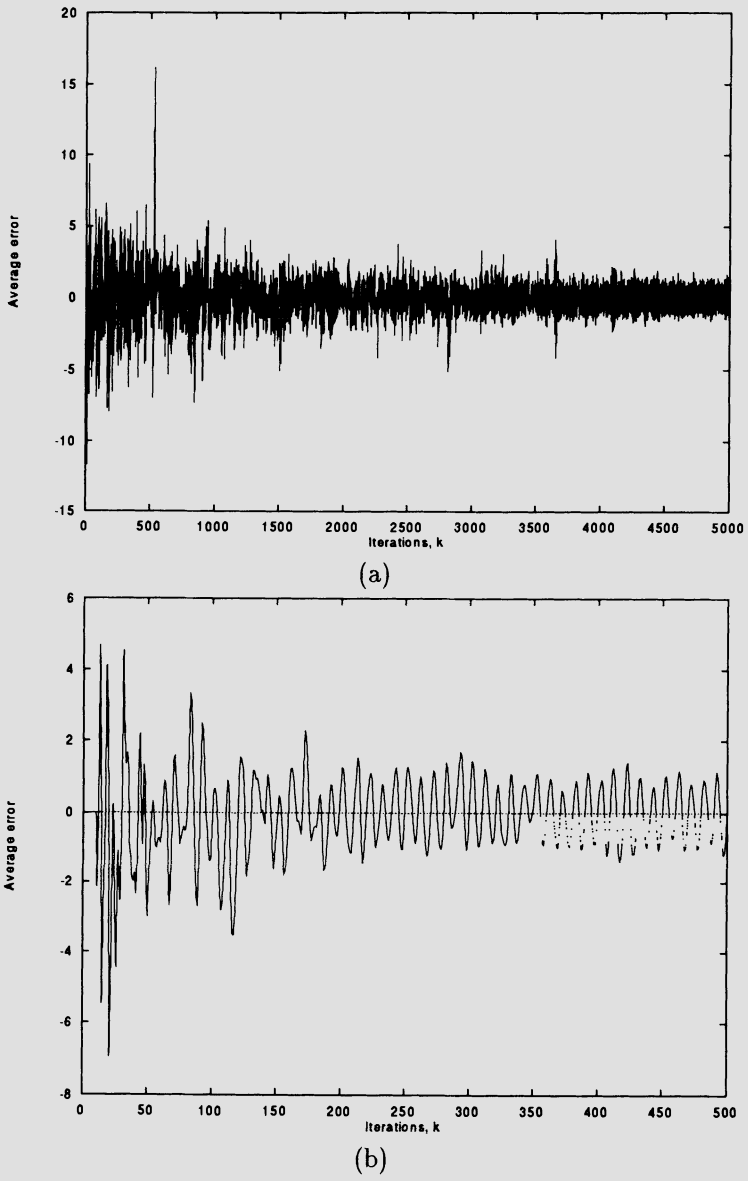


Figure 5.5 Learning curves for the (a) LMS and (b) RLS algorithms.

results discussed in the present chapter and to compare the RLS algorithm with the LMS algorithm.

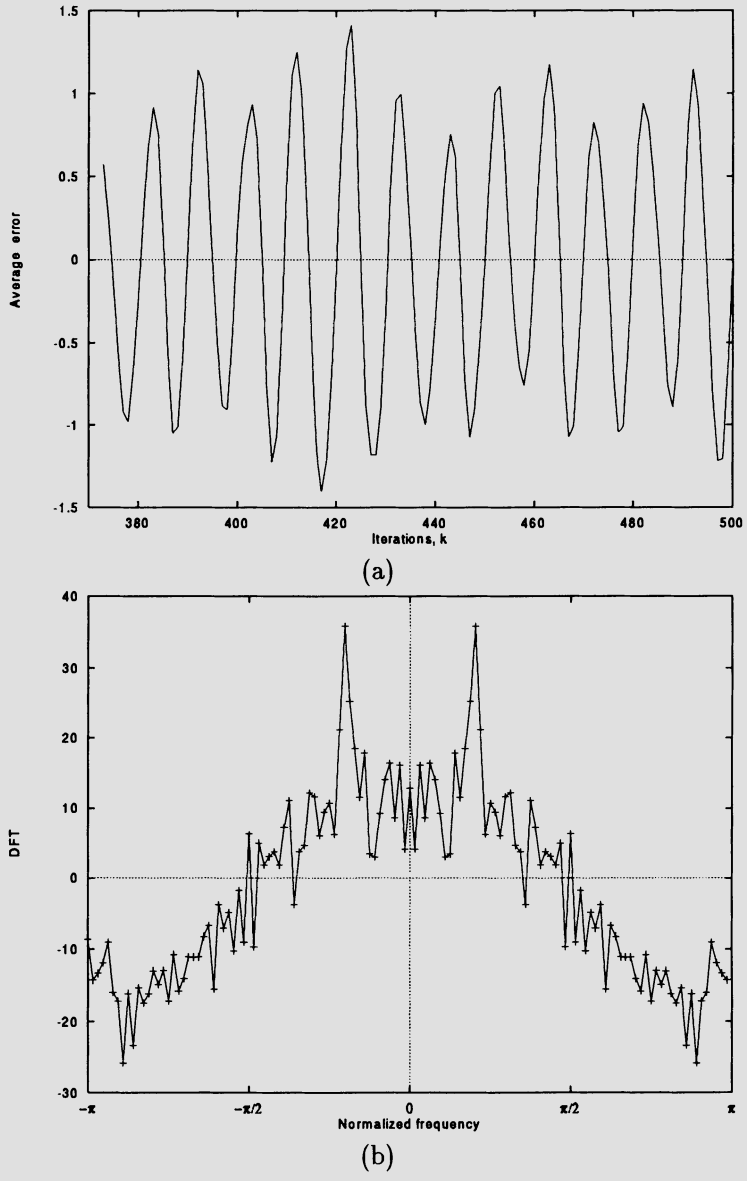


Figure 5.6 (a) Output error for the RLS algorithm and (b) DFT of the output error.

Appendix

The error in the *a priori* output error computation is given by

$$
\begin{aligned}
n_{e'}(k) \approx\ & -n_a(k)[d(k) - \mathbf{x}^T(k)\mathbf{w}(k-1)_Q] \\
& -\mathbf{x}^T(k)\begin{bmatrix} n_{p_o}(k) & 0 & 0 & \cdots & 0 \\ 0 & n_{p_1}(k) & \cdots & \cdots & 0 \\ \vdots & & \ddots & & \\ 0 & 0 & \cdots & \cdots & n_{p_N}(k) \end{bmatrix}\mathbf{w}(k-1)_Q \\
& -[n_{a_1}(k)n_{a_2}(k)\ldots n_{a_N}(k)]\begin{bmatrix} \sum_{i=0}^{1} x(k-i)w_i(k-1)_Q \\ \sum_{i=0}^{2} x(k-i)w_i(k-1)_Q \\ \vdots \\ \sum_{i=0}^{N} x(k-i)w_i(k-1)_Q \end{bmatrix} \\
=\ & -n_a(k)e'(k)_Q - \mathbf{x}^T(k)\mathbf{N}_p(k)\mathbf{w}(k-1)_Q - \mathbf{n}_a(k)\mathbf{s}_i(k)
\end{aligned}
$$

where $n_{p_i}(k)$ accounts for the noise generated in the products $x(k-i)w_i(k-1)_Q$ and $n_{a_i}(k)$ accounts for the noise generated in the additions of the product $\mathbf{x}^T(k)\mathbf{w}(k-1)$. Please note that the error terms of second- and higher-order have been neglected.

Using similar assumptions one can show that

$$
\begin{aligned}
&\mathbf{n}_{\mathbf{W}}(k) = -\{\mathbf{n}_{Sx}(k)e'(k)_Q + \mathbf{S}_D(k)_Q\mathbf{N'}_p(k)\mathbf{x}(k)e'(k)_Q \\
&+\mathbf{N''}_p(k)\mathbf{S}_D(k)_Q\mathbf{x}(k)e'(k)_Q + \mathbf{N'}_a(k)[\mathbf{w}(k-1) + \mathbf{S}_D(k)_Q\mathbf{x}(k)e'(k)_Q]\}
\end{aligned}
$$

where

$$
\mathbf{n}_{Sx}(k) = \begin{bmatrix} \sum_{j=1}^{N} n'_{a_{1,j}}(k)\sum_{i=0}^{j}\mathbf{S}_{D_{1,i}}(k)_Q x(k-i) \\ \vdots \\ \sum_{j=1}^{N} n'_{a_{N+1,j}}(k)\sum_{i=0}^{j}\mathbf{S}_{D_{N+1,i}}(k)_Q x(k-i) \end{bmatrix}
$$

$$\mathbf{N'}_a(k) = \begin{bmatrix} n'_{a_o}(k) & 0 & \cdots & 0 \\ 0 & n'_{a_1}(k) & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & n'_{a_N}(k) \end{bmatrix}$$

$$\mathbf{N'}_p(k) = \begin{bmatrix} n'_{p_o}(k) & 0 & \cdots & 0 \\ 0 & n'_{p_1}(k) & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & n'_{p_N}(k) \end{bmatrix}$$

$$\mathbf{N''}_p(k) = \begin{bmatrix} n''_{p_{1,1}}(k) & n''_{p_{1,2}}(k) & \cdots & n''_{p_{1,N+1}}(k) \\ n''_{p_{2,1}}(k) & n''_{p_{2,2}}(k) & & \vdots \\ \vdots & & \ddots & \vdots \\ n''_{p_{N+1,1}}(k) & \cdots & \cdots & n''_{p_{N+1,N+1}}(k) \end{bmatrix}$$

The vector $\mathbf{n}_{Sx}(k)$ is due to the quantization of additions in the matrix product $\mathbf{S}_D(k)\mathbf{x}(k)$, while the matrix $\mathbf{N''}_p(k)$ accounts for product quantizations in the same operation. The matrix $\mathbf{N'}_a(k)$ models the error in the vector addition to generate $\mathbf{w}(k)_Q$, while $\mathbf{N'}_p(k)$ models the quantization in the product of $e'(k)$ by $\mathbf{S}_D(k)_Q\mathbf{x}(k)$.

By replacing $d'(k)$ by $\mathbf{x}^T(k)\mathbf{w}_o$ in the expression of $e'(k)_Q$ given in equation (5.86), it follows that

$$e'(k)_Q = -\mathbf{x}^T(k)\Delta\mathbf{w}(k-1)_Q - n'_e(k) + n(k)$$

By using in the equation above the expression of $\mathbf{w}(k)_Q$ of equation (5.87), subtracting $\mathbf{w}_o$ in each side of the equation, and neglecting the second- and higher-order errors, after some manipulations the following equality results

$$\begin{aligned}\Delta\mathbf{w}(k)_Q = [&\mathbf{I} - \mathbf{S}_D(k)_Q\mathbf{x}(k)\mathbf{x}^T(k) + \mathbf{n}_{Sx}\mathbf{x}^T(k) + \mathbf{S}_D(k)_Q\mathbf{N'}_p(k)\mathbf{x}(k)\mathbf{x}^T(k) \\ &+\mathbf{N''}_p(k)\mathbf{S}_D(k)_Q\mathbf{x}(k)\mathbf{x}^T(k) + \mathbf{N'}_a(k)\mathbf{S}_D(k)_Q\mathbf{x}(k)\mathbf{x}^T(k) + \mathbf{N'}_a(k)] \\ &\Delta\mathbf{w}(k-1)_Q + \mathbf{N'}_a(k)\mathbf{w}_o + \mathbf{S}_D(k)_Q\mathbf{x}(k)[n(k) - n'_e(k)]\end{aligned}$$

Since all the noise components in the above equation have zero mean, in average the tap coefficients will converge to their optimal values because the same dynamic equation describes the evolution of $\Delta\mathbf{w}(k)$ and $\Delta\mathbf{w}(k)_Q$.

Finally, the variance of the *a priori* error noise can be derived as follows:

$$\begin{aligned}\sigma_{e'}^2 &= \sigma_e^2 = \sigma_{n_a}^2 \xi(k)_Q + \sigma_{n_p}^2 \sum_{i=0}^{N} \mathbf{R}_{i,i} cov[\mathbf{w}(k)_Q]_{i,i} \\ &+\sigma_{n_a}^2 \left\{ E\left[\left(\sum_{i=0}^{1} x(k-i) w_i(k-1)_Q\right)^2\right] \right. \\ &+E\left[\left(\sum_{i=0}^{2} x(k-i) w_i(k-1)_Q\right)^2\right] \\ &\left. +\cdots + E\left[\left(\sum_{i=0}^{N} x(k-i) w_i(k-1)_Q\right)^2\right] \right\}\end{aligned}$$

where $\sigma_{n'_{ai}}^2 = \sigma_{n_a}^2$ was used and $[\,]_{i,i}$ means diagonal elements of $[\,]$. The second term can be further simplified as follows:

$$\begin{aligned}tr\{\mathbf{R} cov[\mathbf{w}(k)_Q]\} &\approx \sum_{i=0}^{N} \mathbf{R}_{i,i} w_{oi}^2 + \sum_{i=0}^{N} \mathbf{R}_{i,i} cov[\Delta \mathbf{w}(k)]_{i,i} \\ &+ first - \text{ and higher} - order \text{ terms } \cdots\end{aligned}$$

Since this term is multiplied by $\sigma_{n_p}^2$, any first- and higher-order terms can be neglected. The first term of σ_e^2 is also small in the steady state. The last term can be rewritten as

$$\begin{aligned}&\sigma_{n_a}^2 \left\{ E\left[\left(\sum_{i=0}^{1} x(k-i) w_{oi}\right)^2\right] + E\left[\left(\sum_{i=0}^{2} x(k-i) w_{oi}\right)^2\right] + \cdots \right. \\ &\left. +E\left[\left(\sum_{i=0}^{N} x(k-i) w_{oi}\right)^2\right] \right\} = \sigma_{n_a}^2 \left\{ \sum_{j=1}^{N} \sum_{i=0}^{j} \mathbf{R}_{i,i} [cov(\Delta \mathbf{w}(k))]_{i,i} \right\}\end{aligned}$$

where terms of order higher than one were neglected, $x(k)$ was considered uncorrelated to $\Delta \mathbf{w}(k)$, and $\text{cov}(\Delta \mathbf{w}(k))$ was considered a diagonal matrix. Actually, if $x(k)$ is considered a zero-mean Gaussian noise from the proof of equation (5.34) and equation (5.44), it can be shown that

$$cov[\Delta \mathbf{w}(k)] \approx \frac{\sigma_n^2}{\sigma_x^2} \mathbf{I}$$

Since this term will be multiplied by $\sigma_{n_a}^2$ and $\sigma_{n_p}^2$, it can also be disregarded. In conclusion

$$\sigma_e^2 \approx \sigma_{n_a}^2 \{E[\sum_{j=1}^{N}(\sum_{i=0}^{j} x(k-i)w_{oi})^2]\} + \sigma_{n_p}^2 \sum_{i=0}^{N} \mathbf{R}_{i,i} w_{oi}^2$$

This equation can be simplified further when $x(k)$ is as described above and $\sigma_{na}^2 = \sigma_{n_p}^2 = \sigma_d^2$

$$\begin{aligned} \sigma_e^2 &\approx \sigma_d^2 \left[\sum_{i=1}^{N}(N-i+2)\mathbf{R}_{i,i} w_{oi}^2 - \mathbf{R}_{1,1} w_{o1}^2\right] \\ &= \sigma_d^2 \sigma_x^2 \left[\sum_{i=1}^{N}(N-i+2) w_{oi}^2 - w_{o1}^2\right] \end{aligned}$$

References

1. G. C. Goodwin and R. L. Payne, *Dynamic System Identification: Experiment Design and Data Analysis*, Academic Press, New York, NY, 1977.

2. S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, NJ, 2nd edition, 1991.

3. S. H. Ardalan, "Floating-point analysis of recursive least-squares and least-mean squares adaptive filters," *IEEE Trans. on Circuits and Systems*, vol. CAS-33, pp. 1192-1208, Dec. 1986.

4. J. M. Cioffi, "Limited precision effects in adaptive filtering," *IEEE Trans. on Circuits and Systems*, vol. CAS-34, pp. 821-833, July 1987.

5. S. H. Ardalan and S. T. Alexander, "Fixed-point roundoff error analysis of the exponentially windowed RLS algorithm for time-varying systems," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-35, pp. 770-783, June 1983.

6. G. E. Bottomley and S. T. Alexander, "A novel approach for stabilizing recursive least squares filters," *IEEE Trans. on Signal Processing*, vol. 39, pp. 1770-1779, Aug. 1991.

7. R. S. Medaugh and L. J. Griffiths, "A comparison of two linear predictors," *Proc. IEEE Intern. Conf. on Acoust., Speech, Signal Processing*, Atlanta, GA, pp. 293-296, April 1981.

8. F. Ling and J. G. Proakis, "Nonstationary learning characteristics of least squares adaptive estimation algorithms," *Proc. IEEE Intern. Conf. on Acoust., Speech, Signal Processing*, San Diego, CA, pp. 30.3.1.-30.3.4, March 1984.

9. E. Eleftheriou and D. D. Falconer, "Tracking properties and steady-state performance of RLS adaptive filter algorithms," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-34, pp. 1097-1110, Oct. 1986.

10. J. M. Cioffi and T. Kailath, "Fast recursive-least-squares transversal filters for adaptive filtering," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-32, pp. 304- 337, April 1984.

11. F. Ling and J. G. Proakis, "Numerical accuracy and stability: two problems of adaptive estimation algorithms caused by round-off error," *Proc. IEEE Intern. Conf. on Acoust., Speech, Signal Processing*, San Diego, CA, pp. 30.3.1-30.3.4, March 1984.

12. G. Kubin, "Stabilization of the RLS algorithm in the absence of persistent excitation," *Proc. IEEE Intern. Conf. on Acoust., Speech, Signal Processing*, NY, pp. 1369-1372, Apr. 1988.

13. M. H. Verhaegen, "Round-off error propagation in four generally applicable, recursive, least squares estimation schemes," *Automatica*, vol. 25, pp. 437-444, Mar. 1989.

14. T. Adali and S. H. Ardalan, "Steady state and convergence characteristics of the fixed-point RLS algorithm," *Proc. IEEE Intern. Symp. on Circuits Systems,* New Orleans, LA, pp. 788-791, May 1990.

15. A. Antoniou, *Digital Filters: Analysis, Design and Applications*, McGraw Hill, New York, NY, 2nd edition, 1992.

16. A. B. Spirad and D. L. Snyder, "Quantization errors in floating point arithmetic," *IEEE Trans. on Acoust., Speech, and Signal Processing,* vol. ASSP-26, pp. 456-463, Oct. 1978.

17. S. Ardalan, "On the sensitivity of transversal RLS algorithms to random perturbations in the filter coefficients," *IEEE Trans. on Acoust., Speech, and Signal Processing,* vol. 36, pp. 1781-1783, Nov. 1988.

18. C. R. Johnson, Jr., *Lectures on Adaptive Parameter Estimation*, Prentice Hall, Englewood Cliffs, NJ, 1988.

19. O. M. Macchi and N. J. Bershad, "Adaptive recovery of a chirped sinusoid in noise, Part 1: performance of the RLS algorithm," *IEEE Trans. on Signal Processing*, vol. 39, pp. 583-594, March 1991.

Problems

1. The RLS algorithm is used to predict the signal $x(k) = \cos\frac{\pi k}{3}$ using a second-order FIR filter with the first tap fixed at 1. Given $\lambda = 0.98$, calculate the output signal $y(k)$ and the tap coefficients for the first 10 iterations. Note that we aim the minimization of $E[y^2(k)]$.

   Start with $\mathbf{w}^T(0) = [1\ 0\ 0]$ and $\delta = 100$.

2. Show that the solution in equation (5.4) is a minimum point.

3. Show that $\mathbf{S}_D(k)$ approaches a null matrix for large k, when $\lambda = 1$.

4. Suppose that the measurement noise $n(k)$ is a random signal with zero-mean and the probability density with normal distribution. In a sufficient-order identification of an FIR system with optimal coefficients given by $\mathbf{w}_o$, show that the least-squares solution with $\lambda = 1$ is also normally distributed with mean $\mathbf{w}_o$ and covariance $\mathbf{S}_D(k)\sigma_n^2$.

5. Prove that equation (5.37) is valid. What is the result when $n(k)$ has zero mean and is correlated to the input signal $x(k)$?

   Hint: You can use the relation $E[e^2(k)] = E[e(k)]^2 + \sigma^2[e(k)]$. where $\sigma^2[\cdot]$ means variance of $[\cdot]$.

6. Consider that the additive noise $n(k)$ is uncorrelated with the input and the desired signals and is also a nonwhite noise with autocorrelation matrix $\mathbf{R}_n$. Determine the transfer function of a prewhitening filter that applied to $d'(k) + n(k)$ and $x(k)$ generates the optimum least-squares solution $\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p}$ for $k \to \infty$.

7. Show that if the additive noise is uncorrelated with $d'(k)$ and $x(k)$, and nonwhite, the least-squares algorithm will converge asymptotically to the optimal solution.

8. In problem 4, when $n(k)$ is correlated to $x(k)$, is $\mathbf{w}_o$ still the optimal solution? If not, what is the optimal solution?

9. Show that in the RLS algorithm the following relation is true

$$\xi^d(k) = \lambda\xi^d(k-1) + e(k)e'(k)$$

   where $e'(k)$ is the *a priori* error as defined in equation (5.8).

10. Prove the validity of the approximation in equation (5.67).

11. Show that for an input signal with diagonal dominant correlation matrix $\mathbf{R}$ the following approximation related to equations (5.107) and (5.111) is valid.

$$E\{\mathbf{N}_{\mathbf{S}_D}(k)\mathbf{x}(k)\mathbf{x}^T(k)cov[\Delta\mathbf{w}(k-1)_Q]\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{N}_{\mathbf{S}_D}(k)\}$$
$$\approx \sigma^2_{\mathbf{S}_D}\sigma^2_x cov[\Delta\mathbf{w}(k-1)_Q]$$

12. Derive the equations (5.114), (5.115), and (5.116).

13. The conventional RLS algorithm was applied to identify a 7th-order time-varying unknown system whose coefficients are first-order Markov processes with $\lambda_{\mathbf{W}} = 0.999$ and $\sigma^2_{\mathbf{W}} = 0.033$. The initial time-varying system multiplier coefficients are

$$\mathbf{w}_o^T =$$

$$[0.03490 - 0.01100 - 0.068640.223910.556860.35798 - 0.02390 - 0.07594]$$

The input signal is a Gaussian white noise with variance $\sigma^2_x = 1$ and the measurement noise is also a Gaussian noise independent of the input signal and of the elements of $\mathbf{n}_{\mathbf{W}}(k)$, with variance $\sigma^2_n = 0.01$.

(a) For $\lambda = 0.97$, compute the excess of MSE.

(b) Repeat (a) for $\lambda = \lambda_{opt}$

(c) Simulate the experiment described, measure the excess of MSE, and compare to the calculated results.

14. Reduce the value of $\lambda_{\mathbf{W}}$ to 0.97 in the problem 13, simulate, and comment on the results.

15. Suppose a 15th-order FIR digital filter with multiplier coefficients given below was identified through an adaptive FIR filter of the same order using the conventional RLS algorithm. Consider that fixed-point arithmetic was used.

Additional noise : white noise with variance	$\sigma^2_n = 0.0015$
Coefficient wordlength:	$b_c = 16$ bits
Signal wordlength:	$b_d = 16$ bits
Input signal: Gaussian white noise with variance	$\sigma^2_x = 0.7$
	$\lambda = \lambda_o$

$\mathbf{w}_o^T$ = [0.0219360 0.0015786 -0.0602449 -0.0118907 0.1375379 0.0574545 -0.3216703 -0.5287203 -0.2957797 0.0002043 0.290670 -0.0353349 -0.0068210 0.0026067 0.0010333 - 0.0143593]

(a) Compute the expected values for $||\Delta \mathbf{w}(k)_Q||$ and $\xi(k)_Q$ for the following case.

(b) Simulate the identification example described and compare the simulated results with those obtained through the closed form formulas.

(c) Plot the learning curves for the finite- and infinite-precision implementations. Also, plot $||\Delta \mathbf{w}(k)||^2$ versus k in both cases.

16. Repeat the problem above for the following cases

(a) $\sigma_n^2 = 0.01, b_c = 9$ bits, $b_d = 9$ bits, $\sigma_x^2 = 0.7, \lambda = \lambda_o$.

(b) $\sigma_n^2 = 0.1, b_c = 10$ bits, $b_d = 10$ bits, $\sigma_x^2 = 0.8, \lambda = \lambda_o$.

(c) $\sigma_n^2 = 0.05, b_c = 8$ bits, $b_d = 16$ bits, $\sigma_x^2 = 0.8, \lambda = \lambda_o$.

17. In the problem 16 above, compute (do not simulate) $E[||\Delta \mathbf{w}(k)_Q||^2], \xi(k)_Q$, and the probable number of iterations before the algorithm stop updating for $\lambda = 1, \lambda = 0.980, \lambda = 0.960$, and $\lambda = \lambda_o$.

18. Repeat problem 15 in the case the input signal is a first-order Markov process with $\lambda_{\mathbf{x}} = 0.95$.

19. A digital channel model can be represented by the following impulse response:

$$[-0.001 \ -0.002 \ 0.002 \ 0.2 \ 0.6 \ 0.76 \ 0.9 \ 0.78 \ 0.67 \ 0.58$$
$$0.45 \ 0.3 \ 0.2 \ 0.12 \ 0.06 \ 0 \ -0.2 \ -1 \ -2 \ -1 \ 0 \ 0.1]$$

The channel is corrupted by a Gaussian noise with power spectrum given by

$$|S(e^{j\omega})|^2 = \kappa' \omega^{3/2}$$

where $\kappa' = 10^{-1.5}$. The training signal consists of independent binary samples (-1,1).

Design an FIR equalizer for this problem and use the RLS algorithm. Use a filter of order 50 and plot the learning curve.

20. For the previous problem, using the maximum of 51 adaptive filter coefficients, implement a DFE equalizer and compare the results with those obtained with the FIR equalizer. Again use the RLS algorithm.

6

ADAPTIVE LATTICE-BASED RLS ALGORITHMS

6.1 INTRODUCTION

There are a large number of algorithms that solve the least-squares problem in a recursive form. In particular, the algorithms based on the lattice realization are very attractive because they allow modular implementation and require a reduced number of arithmetic operations (of order N) [1]-[7]. As a consequence, the lattice recursive least-squares (LRLS) algorithms are considered fast implementations of the RLS problem.

The LRLS algorithms are derived by solving the forward and backward linear prediction problems simultaneously. The lattice-based formulation provides the prediction and the general adaptive filter (joint-process estimation) solutions of all intermediate orders from 1 to N simultaneously. Consequently, the order of the adaptive filter can be increased and decreased without affecting the lower order solutions. This property allows the user to activate or inactivate sections of the lattice realization in real time according to performance requirements.

Unlike the RLS algorithm previously discussed which requires only time-recursive equations, the lattice RLS algorithms use time-update and order-update equations.

The performance of the LRLS algorithms when implemented with infinite-precision arithmetic is identical to any other RLS algorithm. However, in finite-precision implementation each algorithm will perform differently.

In this chapter, several forms of the LRLS algorithm are presented. First, the standard LRLS algorithm based on *a posteriori* errors is presented, followed by

P. S. R. Diniz, *Adaptive Filtering*

the normalized version. The algorithms with error feedback are also derived. Then, we develop the LRLS algorithm based on *a priori* errors.

6.2 RECURSIVE LEAST-SQUARES PREDICTION

The solutions of the RLS forward and backward prediction problems are essential to derive the order-updating equations inherent to the LRLS algorithms. In both cases, the results are derived following the same derivation procedure of the conventional RLS algorithm, since the only distinct feature of the prediction problems is the definition of the reference signal $d(k)$. For example, in the forward prediction case we have $d(k) = x(k)$ whereas the input signal vector has the sample $x(k-1)$ as the most recent data. For the backward prediction case $d(k) = x(k-i-1)$, where the index i defines the sample in the past which we wish to predict, and the input signal vector has $x(k)$ as the most recent data. In this section, these solutions are studied and the results show how informations can be exchanged between the forward and backward predictor solutions.

6.2.1 Forward Prediction Problem

The objective of the forward prediction is to predict a future sample of a given input sequence using the currently available information of the sequence. For example, one can try to predict the value of $x(k)$ using past samples $x(k-1)$, $x(k-2)\ldots$, through an FIR prediction filter with $i+1$ coefficients as follows:

$$y_f(k, i+1) = \mathbf{w}_f^T(k, i+1)\mathbf{x}(k-1, i+1) \tag{6.1}$$

where $y_f(k, i+1)$ is the predictor output signal,

$$\mathbf{w}_f(k, i+1) = [w_{f0}(k)\ w_{f1}(k)\ \ldots\ w_{fi}(k)]^T$$

is the FIR forward prediction coefficient vector, and

$$\mathbf{x}(k-1, i+1) = [x(k-1)\ x(k-2)\ \ldots\ x(k-i-1)]^T$$

is the available input signal vector. The second variable included in vectors of equation (6.1) is to indicate the vector dimension, since it is required in the order-updating equations of the LRLS algorithm. This second variable will be included where needed in the present chapter.

The instantaneous *a posteriori* forward prediction error is given by

$$e_f(k, i+1) = x(k) - \mathbf{w}_f^T(k, i+1)\mathbf{x}(k-1, i+1) \tag{6.2}$$

For the RLS formulation of the forward prediction problem, define the weighted forward prediction error vector as follows:

$$\mathbf{e}_f(k, i+1) = \hat{\mathbf{x}}(k) - \mathbf{X}^T(k-1, i+1)\mathbf{w}_f(k, i+1) \tag{6.3}$$

where

$$\hat{\mathbf{x}}(k) = [x(k)\ \lambda^{1/2}x(k-1)\ \lambda x(k-2)\ \ldots\ \lambda^{k/2}x(0)]^T$$

$$\mathbf{e}_f(k,i+1)=[e_f(k,i+1)\ \lambda^{1/2}e_f(k-1,i+1)\ \lambda e_f(k-2,i+1)\ \ldots\ \lambda^{k/2}e_f(0,i+1)]^T$$

and

$$\mathbf{X}_{(k-1,i+1)}=\begin{bmatrix} x(k-1) & \lambda^{1/2}x(k-2) & \cdots & \lambda^{(k-2)/2}x(1) & \lambda^{(k-1)/2}x(0) & 0 \\ x(k-2) & \lambda^{1/2}x(k-3) & \cdots & \lambda^{(k-2)/2}x(0) & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ x(k-i-1) & \lambda^{1/2}x(k-i-2) & \cdots & 0 & 0 & 0 \end{bmatrix}$$

It is straightforward to show that $\mathbf{e}_f(k, i+1)$ can be rewritten as

$$\mathbf{e}_f(k, i+1) = \mathbf{X}^T(k, i+2)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k, i+1) \end{bmatrix} \tag{6.4}$$

The objective function that we want to minimize in the least-squares sense is the forward prediction error given by

$$\begin{aligned} \xi_f^d(k, i+1) &= \mathbf{e}_f^T(k, i+1)\mathbf{e}_f(k, i+1) \\ &= \sum_{l=0}^{k} \lambda^{k-l} e_f^2(l, i+1) \\ &= \sum_{l=0}^{k} \lambda^{k-l}[x(l) - \mathbf{x}^T(l-1, i+1)\mathbf{w}_f(k, i+1)]^2 \end{aligned} \tag{6.5}$$

By differentiating $\xi_f^d(k, i+1)$ with respect to $\mathbf{w}_f(k, i+1)$ and equating the result to zero, we can find the optimum coefficient vector that minimizes the objective function, namely,

$$\begin{aligned}
\mathbf{w}_f(k, i+1) &= \left[\sum_{l=0}^{k} \lambda^{k-l} \mathbf{x}(l-1, i+1)\mathbf{x}^T(l-1, i+1)\right]^{-1} \\
&\quad \cdot \sum_{l=0}^{k} \lambda^{k-l} \mathbf{x}(l-1, i+1)x(l) \\
&= [\mathbf{X}(k-1, i+1)\mathbf{X}^T(k-1, i+1)]^{-1}\mathbf{X}(k-1, i+1)\hat{\mathbf{x}}(k) \\
&= \mathbf{R}_{Df}^{-1}(k-1, i+1)\mathbf{p}_{Df}(k, i+1) \qquad (6.6)
\end{aligned}$$

where $\mathbf{R}_{Df}(k-1, i+1)$ is equal to the deterministic correlation matrix $\mathbf{R}_D(k-1)$ of order $i+1$, and $\mathbf{p}_{Df}(k, i+1)$ is the deterministic cross-correlation vector between $x(l)$ and $\mathbf{x}(l-1, i+1)$.

The exponentially weighted sum of squared errors can be written as (see equation (6.5)):

$$\begin{aligned}
\xi_f^d(k, i+1) &= \sum_{l=0}^{k} \lambda^{k-l}[x^2(l) - 2x(l)\mathbf{x}^T(l-1, i+1)\mathbf{w}_f(k, i+1) \\
&\quad + (\mathbf{x}^T(l-1, i+1)\mathbf{w}_f(k, i+1))^2] \\
&= \sum_{l=0}^{k} \lambda^{k-l}[x^2(l) - x(l)\mathbf{x}^T(l-1, i+1)\mathbf{w}_f(k, i+1)] \\
&\quad + \sum_{l=0}^{k} \lambda^{k-l}[-x(l) + \mathbf{x}^T(l-1, i+1)\mathbf{w}_f(k, i+1)]\mathbf{x}^T(l-1, i+1)\mathbf{w}_f(k, i+1) \\
&= \sum_{l=0}^{k} \lambda^{k-l} x(l)[x(l) - \mathbf{x}^T(l-1, i+1)\mathbf{w}_f(k, i+1)] \\
&\quad + \Big[\textstyle\sum_{l=0}^{k} -\lambda^{k-l} x(l)\mathbf{x}^T(l-1, i+1) \\
&\quad + \mathbf{w}_f^T(k, i+1) \textstyle\sum_{l=0}^{k} \lambda^{k-l}\mathbf{x}(l-1, i+1)\mathbf{x}^T(l-1, i+1)\Big]\mathbf{w}_f(k, i+1)
\end{aligned} \qquad (6.7)$$

If we replace equation (6.6) in the second term of the last relation above, it can be shown by using the fact that $\mathbf{R}_D(k-1)$ is symmetric that this term is zero.

Therefore, the minimum value of $\xi_f^d(k,i+1)$ [1] is given by

$$\begin{aligned}\xi_{f_{min}}^d(k,i+1) &= \sum_{l=0}^{k}\lambda^{k-l}x(l)[x(l)-\mathbf{x}^T(l-1,i+1)\mathbf{w}_f(k,i+1)] \\ &= \sum_{l=0}^{k}\lambda^{k-l}x^2(l)-\mathbf{p}_{Df}^T(k,i+1)\mathbf{w}_f(k,i+1) \\ &= \sigma_f^2(k)-\mathbf{w}_f^T(k,i+1)\mathbf{p}_{Df}(k,i+1) \end{aligned} \tag{6.8}$$

By combining equation (6.6) for $\mathbf{w}_f(k,i)$ and (6.8) for $\xi_{f_{min}}^d(k,i+1)$ the following matrix equation can be obtained

$$\begin{bmatrix} \sigma_f^2(k) & \mathbf{p}_{Df}^T(k,i+1) \\ \mathbf{p}_{Df}(k,i+1) & \mathbf{R}_{Df}(k-1,i+1) \end{bmatrix}\begin{bmatrix} 1 \\ -\mathbf{w}_f(k,i+1) \end{bmatrix} = \begin{bmatrix} \xi_{f_{min}}^d(k,i+1) \\ \mathbf{0} \end{bmatrix} \tag{6.9}$$

Since $\sigma_f^2(k)=\sum_{l=0}^{k}\lambda^{k-l}x^2(l)$ and $\mathbf{p}_{Df}(k,i+1)=\sum_{l=0}^{k}\lambda^{k-l}\mathbf{x}(l-1,i+1)x(l)$, it is easy to conclude that the left most term of equation (6.9) can be rewritten as

$$\begin{aligned}&\begin{bmatrix} \sum_{l=0}^{k}\lambda^{k-l}x^2(l) & \sum_{l=0}^{k}\lambda^{k-l}\mathbf{x}^T(l-1,i+1)x(l) \\ \sum_{l=0}^{k}\lambda^{k-l}\mathbf{x}(l-1,i+1)x(l) & \sum_{l=0}^{k}\lambda^{k-l}\mathbf{x}(l-1,i+1)\mathbf{x}^T(l-1,i+1) \end{bmatrix} \\ &= \sum_{l=0}^{k}\lambda^{k-l}\begin{bmatrix} x(l) \\ \mathbf{x}(l-1,i+1) \end{bmatrix}[x(l)\ \mathbf{x}^T(l-1,i+1)] \\ &= \mathbf{R}_D(k,i+2) \end{aligned} \tag{6.10}$$

Therefore,

$$\mathbf{R}_D(k,i+2)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k,i+1) \end{bmatrix} = \begin{bmatrix} \xi_{f_{min}}^d(k,i+1) \\ \mathbf{0} \end{bmatrix}$$

where $\mathbf{R}_D(k,i+2)$ corresponds to $\mathbf{R}_D(k)$ used in the previous chapter with dimension $i+2$. The equation above relates the deterministic correlation matrix of order $i+2$ to the minimum least-squares forward prediction error. The appropriate partitioning of matrix $\mathbf{R}_D(k,i+2)$ enables the derivation of the order-updating equation for the predictor tap coefficients, as will be discussed later.

[1] Notice that no special notation was previously used for the minimum value of the RLS objective function, however, when deriving the lattice algorithms this definition is necessary.

6.2.2 Backward Prediction Problem

The objective of the backward predictor is to generate an estimate of a past sample of a given input sequence using the currently available information of the sequence. For example, sample $x(k-i-1)$ can be estimated from $\mathbf{x}(k, i+1)$, through an FIR backward prediction filter with $i+1$ coefficients as follows:

$$y_b(k, i+1) = \mathbf{w}_b^T(k, i+1)\mathbf{x}(k, i+1) \tag{6.11}$$

where $y_b(k, i+1)$ is the backward predictor output signal, and

$$\mathbf{w}_b^T(k, i+1) = [w_{b0}(k) w_{b1}(k) \ \ldots \ w_{bi}(k)]^T$$

is the FIR backward prediction coefficient vector.

The instantaneous *a posteriori* backward prediction error is given by

$$e_b(k, i+1) = x(k-i-1) - \mathbf{w}_b^T(k, i+1)\mathbf{x}(k, i+1) \tag{6.12}$$

The weighted backward prediction error vector is defined as follows:

$$\mathbf{e}_b(k, i+1) = \hat{\mathbf{x}}(k-i-1) - \mathbf{X}^T(k, i+1)\mathbf{w}_b(k, i+1) \tag{6.13}$$

where

$$\hat{\mathbf{x}}(k-i-1) = [x(k-i-1)\, \lambda^{1/2} x(k-i-2) \ \ldots \ \lambda^{(k-i-1)/2} x(0)\, 0 \ldots 0]^T$$

$$\mathbf{e}_b(k, i+1) = [e_b(k, i+1)\, \lambda^{1/2} e_b(k-1, i+1) \ \ldots \ \lambda^{k/2} e_b(0, i+1)]^T$$

and

$$\mathbf{X}(k, i+1) = \begin{bmatrix} x(k) & \lambda^{1/2}x(k-1) & \cdots & \lambda^{(k-1)/2}x(1) & \lambda^{(k)/2}x(0) \\ x(k-1) & \lambda^{1/2}x(k-2) & \cdots & \lambda^{(k-2)/2}x(0) & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ x(k-i) & \lambda^{1/2}x(k-i-1) & \cdots & 0 \cdots & 0 \end{bmatrix}$$

The error vector can be rewritten as

$$\mathbf{e}_b(k, i+1) = \mathbf{X}^T(k, i+2) \begin{bmatrix} -\mathbf{w}_b(k, i+1) \\ 1 \end{bmatrix} \tag{6.14}$$

The objective function to be minimized in the backward prediction problem is given by

$$\begin{aligned}\xi_b^d(k,i+1) &= \mathbf{e}_b^T(k,i+1)\mathbf{e}_b(k,i+1) \\ &= \sum_{l=0}^{k}\lambda^{k-l}e_b^2(l,i+1) \\ &= \sum_{l=0}^{k}\lambda^{k-l}[x(l-i-1)-\mathbf{x}^T(l,i+1)\mathbf{w}_b(k,i+1)]^2 \quad (6.15)\end{aligned}$$

The optimal solution for the coefficient vector is

$$\begin{aligned}\mathbf{w}_b(k,i+1) &= \left[\sum_{l=0}^{k}\lambda^{k-l}\mathbf{x}(l,i+1)\mathbf{x}^T(l,i+1)\right]^{-1} \\ &\quad \cdot\sum_{l=0}^{k}\lambda^{k-l}\mathbf{x}(l,i+1)x(l-i-1) \\ &= [\mathbf{X}(k,i+1)\mathbf{X}^T(k,i+1)]^{-1}\mathbf{X}(k,i+1)\hat{\mathbf{x}}(k-i-1) \\ &= \mathbf{R}_{Db}^{-1}(k,i+1)\mathbf{p}_{Db}(k,i+1) \quad (6.16)\end{aligned}$$

where $\mathbf{R}_{Db}(k,i+1)$ is equal to the deterministic correlation matrix $\mathbf{R}_D(k)$ of order $i+1$, and $\mathbf{p}_{Db}(k,i+1)$ is the deterministic cross-correlation vector between $x(l-i-1)$ and $\mathbf{x}(l,i+1)$.

Using the same procedure to derive the minimum least-squares solution in the RLS problem, it can be shown that the minimum value of $\xi_b^d(k)$ is given by

$$\begin{aligned}\xi_{b_{min}}^d(k,i+1) &= \sum_{l=0}^{k}\lambda^{k-l}x(l-i-1)[x(l-i-1)-\mathbf{x}^T(l,i+1)\mathbf{w}_b(k,i+1)] \\ &= \sum_{l=0}^{k}\lambda^{k-l}x^2(l-i-1)-\mathbf{p}_{Db}^T(k,i+1)\mathbf{w}_b(k,i+1) \\ &= \sigma_b^2(k)-\mathbf{w}_b^T(k,i+1)\mathbf{p}_{Db}(k,i+1) \quad (6.17)\end{aligned}$$

By combining equations (6.16) and (6.17), the following matrix equation results

$$\begin{bmatrix} \mathbf{R}_{Db}(k,i+1) & \mathbf{p}_{Db}(k,i+1) \\ \mathbf{p}_{Db}^T(k,i+1) & \sigma_b^2(k) \end{bmatrix}\begin{bmatrix} -\mathbf{w}_b(k,i+1) \\ 1 \end{bmatrix}$$

$$
\begin{aligned}
&= \begin{bmatrix} \sum_{l=0}^{k} \lambda^{k-l}\mathbf{x}(l,i+1)\mathbf{x}^T(l,i+1) & \sum_{l=0}^{k} \lambda^{k-l}\mathbf{x}(l,i+1)x(l-i-1) \\ \sum_{l=0}^{k} \lambda^{k-l}\mathbf{x}^T(l,i+1)x(l-i-1) & \sum_{l=0}^{k} \lambda^{k-l}x^2(l-i-1) \end{bmatrix} \\
&\quad \cdot \begin{bmatrix} -\mathbf{w}_b(k,i+1) \\ 1 \end{bmatrix} \\
&= \mathbf{R}_{D(k,i+2)} \begin{bmatrix} -\mathbf{w}_b(k,i+1) \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{0} \\ \xi^d_{b_{min}}(k,i+1) \end{bmatrix} \qquad (6.18)
\end{aligned}
$$

where $\mathbf{R}_D(k,i+2)$ is equal to $\mathbf{R}_D(k)$ of dimension $i+2$. The equation above relates the deterministic correlation matrix of order $i+1$ to the minimum least-squares backward prediction error. The equation is important in the derivation of the order-updating equation for the backward predictor tap coefficients. This issue is discussed in the following section.

6.3 ORDER-UPDATING EQUATIONS

The objective of this section is to derive the order-updating equations for the forward and backward prediction errors. These equations are the starting point to generate the lattice realization.

6.3.1 A New Parameter $\delta(k,i)$

Using the results of equations (6.9) and (6.10), and the decomposition of $\mathbf{R}_D(k,i+2)$ given in equation (6.18), we can show that

$$
\begin{aligned}
\mathbf{R}_D(k,i+2) \begin{bmatrix} 1 \\ -\mathbf{w}_f(k,i) \\ 0 \end{bmatrix} &= \begin{bmatrix} \mathbf{R}_D(k,i+1) & \mathbf{p}_{Db}(k,i+1) \\ \mathbf{p}^T_{Db}(k,i+1) & \sigma^2_b(k) \end{bmatrix} \\
&\quad \cdot \begin{bmatrix} 1 \\ -\mathbf{w}_f(k,i) \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \xi^d_{f_{min}}(k,i) \\ \mathbf{0} \\ \mathbf{p}^T_{Db}(k,i+1) \begin{bmatrix} 1 \\ -\mathbf{w}_f(k,i) \end{bmatrix} \end{bmatrix}
\end{aligned}
$$

$$= \begin{bmatrix} \xi^d_{f_{min}}(k,i) \\ \mathbf{0} \\ \delta_f(k,i) \end{bmatrix} \tag{6.19}$$

where relation (6.9) was employed in the second equality,

$$\begin{aligned} \delta_f(k,i) &= \sum_{l=0}^{k} \lambda^{k-l} x(l)x(l-i-1) \\ &\quad - \sum_{l=0}^{k} \lambda^{k-l} x(l-i-1)\mathbf{x}^T(l-1,i)\mathbf{w}_f(k,i) \\ &= \sum_{l=0}^{k} \lambda^{k-l} x(l)x(l-i-1) - \sum_{l=0}^{k} \lambda^{k-l} x(l-i-1)y_f(l,i) \\ &= \sum_{l=0}^{k} \lambda^{k-l} e_f(l,i)x(l-i-1) \end{aligned}$$

and $y_f(l,i) = \mathbf{x}^T(l-1,i)\mathbf{w}_f(k,i)$ is the output of a forward prediction filter of order $i-1$. Note that the parameter $\delta_f(k,i)$ can be interpreted as the deterministic cross-correlation between the forward prediction error $e_f(l,i)$ with the coefficients fixed at $\mathbf{w}_f(k,i)$ and the desired signal of the backward predictor filter $x(l-i-1)$.

Similarly, using the results of equations (6.17) and (6.18) it can be shown that

$$\begin{aligned} \mathbf{R}_D(k,i+2) \begin{bmatrix} 0 \\ -\mathbf{w}_b(k-1,i) \\ 1 \end{bmatrix} &= \begin{bmatrix} \sigma_f^2(k) & \mathbf{p}_{Df}^T(k,i+1) \\ \mathbf{p}_{Df}(k,i+1) & \mathbf{R}_D(k-1,i+1) \end{bmatrix} \\ &\quad \cdot \begin{bmatrix} 0 \\ -\mathbf{w}_b(k-1,i) \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{p}_{Df}^T(k,i+1) \begin{bmatrix} -\mathbf{w}_b(k-1,i) \\ 1 \end{bmatrix} \\ \mathbf{0} \\ \xi^d_{b_{min}}(k-1,i) \end{bmatrix} \\ &= \begin{bmatrix} \delta_b(k,i) \\ \mathbf{0} \\ \xi^d_{b_{min}}(k-1,i) \end{bmatrix} \end{aligned} \tag{6.20}$$

where in the second equality we applied the result of equation (6.18), and

$$\begin{aligned}\delta_b(k,i) &= \sum_{l=0}^{k} \lambda^{k-l} x(l-i-1)x(l) - \sum_{l=0}^{k} \lambda^{k-l} x(l)\mathbf{x}^T(l-1,i)\mathbf{w}_b(k-1,i) \\ &= \sum_{l=0}^{k} \lambda^{k-l} x(l-i-1)x(l) - \sum_{l=0}^{k} \lambda^{k-l} x(l) y_b(l-1,i) \\ &= \sum_{l=0}^{k} \lambda^{k-l} e_b(l-1,i)x(l)\end{aligned}$$

where $y_b(l-1,i) = \mathbf{x}^T(l-1,i)\mathbf{w}_b(k-1,i)$ is the output of a backward prediction filter of order $i-1$ with input data of instant $l-1$, when the coefficients of the predictor are $\mathbf{w}_b(k-1,i)$. The parameter $\delta_b(k,i)$ can be interpreted as the deterministic cross-correlation between the backward prediction error $e_b(l-1,i)$ and the desired signal of the forward predictor filter $x(l)$.

In equations (6.19) and (6.20) two new parameters were defined, namely $\delta_f(k,i)$ and $\delta_b(k,i)$. In the following derivations we will show that these parameters are equal. If $\mathbf{R}_D(k,i+2)$ is premultiplied by $[0 \;\; -\mathbf{w}_b^T(k-1,i) \;\; 1]$ and postmultiplied by $[1 \;\; -\mathbf{w}_f(k,i) \;\; 0]^T$, it can be shown that

$$[0 \;\; -\mathbf{w}_b^T(k-1,i) \;\; 1]\, \mathbf{R}_D(k,i+2) \begin{bmatrix} 1 \\ -\mathbf{w}_f(k,i) \\ 0 \end{bmatrix} = \delta_f(k,i) \quad (6.21)$$

By transposing the first and last terms of equation (6.20) the following relation is obtained

$$[0 \;\; -\mathbf{w}_b^T(k-1,i) \;\; 1]\, \mathbf{R}_D(k,i+2) = [\delta_b(k,i) \;\; \mathbf{0}^T \;\; \xi^d_{b_{min}}(k-1,i)] \quad (6.22)$$

By substituting this result in equation (6.21) it follows that

$$[\delta_b(k,i) \;\; \mathbf{0}^T \;\; \xi^d_{b_{min}}(k-1,i)] \begin{bmatrix} 1 \\ -\mathbf{w}_f(k,i) \\ 0 \end{bmatrix} = \delta_f(k,i) \quad (6.23)$$

Therefore, from equation (6.23) we conclude that

$$\delta_f(k,i) = \delta_b(k,i) = \delta(k,i) \quad (6.24)$$

In conclusion, the deterministic cross-correlations between $e_f(l,i)$ and $x(l-i-1)$ and between $e_b(l-1,i)$ and $x(l)$ are equal.

6.3.2 Order Updating of $\xi^d_{b_{min}}(k,i)$ and $\mathbf{w}_b(k,i)$

The order updating of the minimum LS error and the tap coefficients for the backward predictor can be deduced by multiplying equation (6.19) by the scalar $\delta(k,i)/\xi^d_{f_{min}}(k,i)$, i.e.,

$$\frac{\delta(k,i)}{\xi^d_{f_{min}}(k,i)}\mathbf{R}_D(k,i+2)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k,i) \\ 0 \end{bmatrix} = \begin{bmatrix} \delta(k,i) \\ \mathbf{0} \\ \frac{\delta^2(k,i)}{\xi^d_{f_{min}}(k,i)} \end{bmatrix} \tag{6.25}$$

Subtracting equation (6.20) from this result yields

$$\mathbf{R}_D(k,i+2)\begin{bmatrix} \frac{\delta(k,i)}{\xi^d_{f_{min}}(k,i)} \\ -\mathbf{w}_f(k,i)\frac{\delta(k,i)}{\xi^d_{f_{min}}(k,i)} + \mathbf{w}_b(k-1,i) \\ -1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -\xi^d_{b_{min}}(k-1,i) + \frac{\delta^2(k,i)}{\xi^d_{f_{min}}(k,i)} \end{bmatrix} \tag{6.26}$$

Comparing equations (6.18) and (6.26) we conclude that

$$\xi^d_{b_{min}}(k,i+1) = \xi^d_{b_{min}}(k-1,i) - \frac{\delta^2(k,i)}{\xi^d_{f_{min}}(k,i)} \tag{6.27}$$

and

$$\mathbf{w}_b(k,i+1) = \begin{bmatrix} 0 \\ \mathbf{w}_b(k-1,i) \end{bmatrix} - \frac{\delta(k,i)}{\xi^d_{f_{min}}(k,i)}\begin{bmatrix} -1 \\ \mathbf{w}_f(k,i) \end{bmatrix} \tag{6.28}$$

6.3.3 Order Updating of $\xi^d_{f_{min}}(k,i)$ and $\mathbf{w}_f(k,i)$

Similarly, by multiplying equation (6.20) by $\delta(k,i)/\xi^d_{b_{min}}(k-1,i)$, we get

$$\frac{\delta(k,i)}{\xi^d_{b_{min}}(k-1,i)}\mathbf{R}_D(k,i+2)\begin{bmatrix} 0 \\ -\mathbf{w}_b(k-1,i) \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\delta^2(k,i)}{\xi^d_{b_{min}}(k-1,i)} \\ \mathbf{0} \\ \delta(k,i) \end{bmatrix} \tag{6.29}$$

Subtracting equation (6.29) from (6.19), it follows that

$$\mathbf{R}_D(k,i+2)\begin{bmatrix} 1 \\ \frac{\delta(k,i)}{\xi^d_{b_{min}}(k-1,i)}\mathbf{w}_b(k-1,i)-\mathbf{w}_f(k,i) \\ -\frac{\delta(k,i)}{\xi^d_{b_{min}}(k-1,i)} \end{bmatrix} = \begin{bmatrix} \xi^d_{f_{min}}(k,i)-\frac{\delta^2(k,i)}{\xi^d_{b_{min}}(k-1,i)} \\ \mathbf{0} \end{bmatrix} \tag{6.30}$$

Comparing this equation with (6.9) we conclude that

$$\xi^d_{f_{min}}(k,i+1)=\xi^d_{f_{min}}(k,i)-\frac{\delta^2(k,i)}{\xi^d_{b_{min}}(k-1,i)} \tag{6.31}$$

and

$$\mathbf{w}_f(k,i+1) = \begin{bmatrix} \mathbf{w}_f(k,i) \\ 0 \end{bmatrix} - \frac{\delta(k,i)}{\xi^d_{b_{min}}(k-1,i)}\begin{bmatrix} \mathbf{w}_b(k-1,i) \\ -1 \end{bmatrix} \tag{6.32}$$

6.3.4 Order Updating of Prediction Errors

The order updating of the *a posteriori* forward and backward prediction errors can now be derived as follows:

$$\begin{aligned} e_f(k,i+1) &= \mathbf{x}^T(k,i+2)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k,i+1) \end{bmatrix} \\ &= \mathbf{x}^T(k,i+2)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k,i) \\ 0 \end{bmatrix} \\ &\quad + \frac{\delta(k,i)}{\xi^d_{b_{min}}(k-1,i)}\mathbf{x}^T(k,i+2)\begin{bmatrix} 0 \\ \mathbf{w}_b(k-1,i) \\ -1 \end{bmatrix} \\ &= e_f(k,i)-\kappa_f(k,i)e_b(k-1,i) \end{aligned} \tag{6.33}$$

where in the second equality we employed the order-updating equation for the forward prediction coefficients (6.32). The coefficient $\kappa_f(k,i)=\frac{\delta(k,i)}{\xi^d_{b_{min}}(k-1,i)}$ is the so-called forward reflection coefficient.

The order updating of the *a posteriori* backward prediction error is obtained by using equation (6.28) as follows:

$$\begin{aligned} e_b(k,i+1) &= \mathbf{x}^T(k,i+2)\begin{bmatrix} -\mathbf{w}_b(k,i+1) \\ 1 \end{bmatrix} \\ &= \mathbf{x}^T(k,i+2)\begin{bmatrix} 0 \\ -\mathbf{w}_b(k-1,i) \\ 1 \end{bmatrix} \\ &\quad + \frac{\delta(k,i)}{\xi^d_{f_{min}}(k,i)}\mathbf{x}^T(k,i+2)\begin{bmatrix} -1 \\ \mathbf{w}_f(k,i) \\ 0 \end{bmatrix} \\ &= e_b(k-1,i) - \kappa_b(k,i)e_f(k,i) \end{aligned} \tag{6.34}$$

where in the second equality we employed the order-updating equation for the backward prediction coefficients (6.28). The coefficient $\kappa_b(k,i) = \frac{\delta(k,i)}{\xi^d_{f_{min}}(k,i)}$ is the backward reflection coefficient.

The equations (6.33) and (6.34) above can be implemented with a lattice section as illustrated in Fig. 6.1(a). An order-increasing lattice-based forward and backward predictor can be constructed as illustrated in Fig. 6.1(b). The coefficients $\kappa_b(k,i)$ and $\kappa_f(k,i)$ are often called reflection coefficients of the lattice realization.

In the first section of the lattice, the forward and backward prediction errors are equal to the input signal itself since no prediction is performed before the first lattice section, therefore

$$e_b(k,0) = e_f(k,0) = x(k) \tag{6.35}$$

consequently

$$\xi^d_{f_{min}}(k,0) = \xi^d_{b_{min}}(k,0) = \sum_{l=0}^{k} \lambda^{k-l}x^2(l) = x^2(k) + \lambda\xi^d_{f_{min}}(k-1,0) \tag{6.36}$$

A closer look at equations (6.9) and (6.18) leads to the conclusion that the backward and forward predictors utilize the same information matrix $\mathbf{R}_D(k,i+2)$. This result was key to derive the expressions for the *a posteriori* forward and backward prediction errors of (6.33) and (6.34). A hidden importance of these expressions is that they can be shown to be independent of the predictor

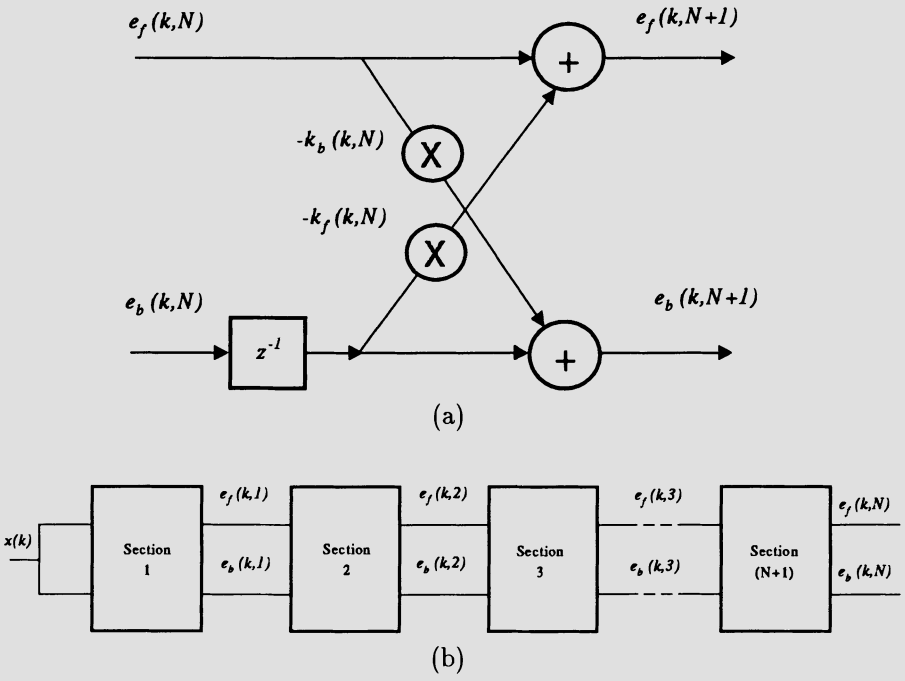


Figure 6.1 Least-squares lattice-based predictor.

tap coefficients. This result will be proved in the following section, where it will be presented an updating formula for $\delta(k,i)$ that is not directly dependent of $\mathbf{w}_f(k,i)$ and $\mathbf{w}_b(k-1,i)$.

Now that all order-updating equations are available, it is necessary to derive the time-updating equations to allow the adaptation of the lattice predictor coefficients.

6.4 TIME-UPDATING EQUATIONS

The time-updating equations are required to deal with the new incoming data that becomes available. Recall that in this text we are studying adaptive filtering algorithms utilizing the new incoming data as soon as they become available. In this section the time-updating equations for the internal quantities of the lattice algorithm are derived.

6.4.1 Time Updating for Prediction Coefficients

From equation (6.6), the time updating of the forward prediction filter coefficients is given by

$$\begin{aligned} \mathbf{w}_f(k,i) &= \mathbf{S}_D(k-1,i)\mathbf{p}_{Df}(k,i) \\ &= \mathbf{R}_D^{-1}(k-1,i)\mathbf{p}_{Df}(k,i) \end{aligned} \tag{6.37}$$

where this is the standard expression for the computation of the optimal coefficient vector leading to the minimization of the LS objective function, adapted to the forward prediction case.

The updating formula of $\mathbf{S}_D(k,i)$ based on the matrix inversion lemma derived in the previous chapter (see Algorithm 5.2) for the conventional RLS algorithm can be used in equation (6.37). The resulting equation is given by

$$\begin{aligned} \mathbf{w}_f(k,i) &= \tfrac{1}{\lambda}\left[\mathbf{S}_D(k-2,i) - \frac{\boldsymbol{\Psi}(k-1,i)\boldsymbol{\Psi}^T(k-1,i)}{\lambda + \boldsymbol{\Psi}^T(k-1,i)\mathbf{X}(k-1,i)}\right]\mathbf{p}_{Df}(k,i) \\ &= \tfrac{1}{\lambda}\left[\mathbf{S}_D(k-2,i) - \frac{\boldsymbol{\Psi}(k-1,i)\mathbf{X}^T(k-1,i)\mathbf{S}_D(k-2,i)}{\lambda + \boldsymbol{\Psi}^T(k-1,i)\mathbf{X}(k-1,i)}\right] \\ &\quad \cdot \left[\lambda \mathbf{p}_{Df}(k-1,i) + x(k)\mathbf{x}(k-1,i)\right] \\ &= \mathbf{w}_f(k-1,i) - \frac{\boldsymbol{\Psi}(k-1,i)\mathbf{X}^T(k-1,i)\mathbf{W}_f(k-1,i)}{\lambda + \boldsymbol{\Psi}^T(k-1,i)\mathbf{X}(k-1,i)} + \frac{x(k)}{\lambda}\mathbf{c} \end{aligned} \tag{6.38}$$

where in the second equality we have applied the time-recursive updating formula of $\mathbf{p}_{Df}(k,i)$, and in the second term of the last expression we have replaced $\mathbf{S}_D(k-2,i)\mathbf{p}_{Df}(k-1,i)$ by $\mathbf{w}_f(k-1,i)$. Vector $\mathbf{c}$ is given by

$$\begin{aligned} \mathbf{c} &= \mathbf{S}_D(k-2,i)\mathbf{x}(k-1,i) - \frac{\boldsymbol{\Psi}(k-1,i)\mathbf{x}^T(k-1,i)\mathbf{S}_D(k-2,i)\mathbf{x}(k-1,i)}{\lambda + \boldsymbol{\Psi}^T(k-1,i)\mathbf{x}(k-1,i)} \\ &= \frac{\lambda \mathbf{S}_D(k-2,i)\mathbf{x}(k-1,i)}{\lambda + \boldsymbol{\Psi}^T(k-1,i)\mathbf{x}(k-1,i)} \end{aligned}$$

Also, it is convenient at this point to recall that $\boldsymbol{\Psi}(k-1,i) = \mathbf{S}_D(k-2,i)\mathbf{x}(k-1,i)$.

The last term in equation (6.38) can be simplified if we apply the following definition

$$\phi(k-1,i) = \frac{\boldsymbol{\Psi}(k-1,i)}{\lambda + \boldsymbol{\Psi}^T(k-1,i)\mathbf{x}(k-1,i)} \tag{6.39}$$

where $\phi(k-1,i)$ is redefined here, including now the order index i. Using this definition in the second and third terms of the last expression of equation (6.38), it can be shown that

$$\begin{aligned}\mathbf{w}_f(k,i) &= \mathbf{w}_f(k-1,i) + \phi(k-1,i)[x(k) - \mathbf{w}_f^T(k-1,i)\mathbf{x}(k-1,i)] \\ &= \mathbf{w}_f(k-1,i) + \phi(k-1,i)e'_f(k,i) \qquad (6.40)\end{aligned}$$

where $e'_f(k,i)$ is the so-called *a priori* forward prediction error of a predictor of order $i-1$, because it utilizes the tap coefficients of the previous instant $k-1$.

Following similar steps to derive equation (6.40), we can show that the time updating for the backward predictor filter is given by

$$\begin{aligned}\mathbf{w}_b(k,i) &= \frac{1}{\lambda}\left[\mathbf{S}_D(k-1,i) - \frac{\boldsymbol{\Psi}(k,i)\boldsymbol{\Psi}^T(k,i)}{\lambda + \boldsymbol{\Psi}^T(k,i)\mathbf{x}(k,i)}\right][\lambda \mathbf{p}_{Db}(k-1,i) + \mathbf{x}(k,i)x(k-i)] \\ &= \mathbf{w}_b(k-1,i) - \phi(k,i)\mathbf{x}^T(k,i)\mathbf{w}_b(k-1,i) + \phi(k,i)x(k-i) \\ &= \mathbf{w}_b(k-1,i) + \phi(k,i)e'_b(k,i) \qquad (6.41)\end{aligned}$$

where $e'_b(k,i)$ is the *a priori* backward prediction error of a predictor of order $i-1$.

6.4.2 Time Updating for $\delta(k,i)$

From the computational point of view, it would be interesting to compute the prediction errors without explicitly using the predictor's tap coefficients. In order to achieve this goal a time-updating expression for $\delta(k,i)$ is derived. A byproduct of this derivation is the introduction of a new parameter, namely $\gamma(k,i)$ that is shown to be a conversion factor between *a priori* and *a posteriori* errors.

Recall from (6.19) the definition of parameter $\delta(k,i)$

$$\delta(k,i) = \mathbf{p}_{Db}^T(k,i+1)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k,i) \end{bmatrix} \qquad (6.42)$$

where $\mathbf{p}_{Db}(k,i+1)$ can be expressed in a recursive form as

$$\begin{aligned}\mathbf{p}_{Db}(k,i+1) &= \sum_{l=0}^{k} \lambda^{k-l}\mathbf{x}(l,i+1)x(l-i-1) \\ &= \mathbf{x}(k,i+1)x(k-i-1) + \lambda \mathbf{p}_{Db}(k-1,i+1) \qquad (6.43)\end{aligned}$$

Substituting equations (6.40) and (6.43) in (6.42), it follows that

$$\begin{aligned}
\delta(k,i) &= [x(k-i-1)\mathbf{x}^T(k,i+1)+\lambda\mathbf{p}_{Db}^T(k-1,i+1)]\begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,i)-\boldsymbol{\phi}(k-1,i)e_f'(k,i) \end{bmatrix} \\
&= \lambda\delta(k-1,i)+\lambda\mathbf{p}_{Db}^T(k-1,i+1)\begin{bmatrix} 0 \\ -\boldsymbol{\phi}(k-1,i)e_f'(k,i) \end{bmatrix} \\
&\quad +x(k-i-1)\mathbf{x}^T(k,i+1)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,i) \end{bmatrix} \\
&\quad +x(k-i-1)\mathbf{x}^T(k,i+1)\begin{bmatrix} 0 \\ -\boldsymbol{\phi}(k-1,i)e_f'(k,i) \end{bmatrix} \qquad (6.44)
\end{aligned}$$

where the equality of (6.42) for the order index $i-1$ was used to obtain the first term of the last equality.

We now derive two relations which are essential to obtain a time-updating equation for $\delta(k,i)$. The resulting equation is efficient from the computational point of view. From the definitions of $\boldsymbol{\phi}(k-1,i)$ and $\boldsymbol{\Psi}(k-1,i)$, see equation (6.39) and the comments after equation (6.38) respectively, it can be shown that the following relation is valid:

$$\begin{aligned}
\mathbf{p}_{Db}^T(k-1,i+1)\begin{bmatrix} 0 \\ \boldsymbol{\phi}(k-1,i) \end{bmatrix} &= \mathbf{p}_{Db}^T(k-2,i)\boldsymbol{\phi}(k-1,i) \\
&= \frac{\mathbf{p}_{Db}^T(k-2,i)\boldsymbol{\Psi}(k-1,i)}{\lambda+\boldsymbol{\Psi}^T(k-1,i)\mathbf{x}(k-1,i)} \\
&= \frac{\mathbf{p}_{Db}^T(k-2,i)\mathbf{S}_D(k-2,i)\mathbf{x}(k-1,i)}{\lambda+\boldsymbol{\Psi}^T(k-1,i)\mathbf{x}(k-1,i)} \\
&= \frac{\mathbf{w}_b^T(k-2,i)\mathbf{x}(k-1,i)}{\lambda+\boldsymbol{\Psi}^T(k-1,i)\mathbf{x}(k-1,i)} \\
&= -\frac{e_b'(k-1,i)-x(k-i-1)}{\lambda+\boldsymbol{\Psi}^T(k-1,i)\mathbf{x}(k-1,i)} \qquad (6.45)
\end{aligned}$$

Also, again using equation (6.39) it is easy to obtain the relation

$$\begin{aligned}
\mathbf{x}^T(k,i+1)\begin{bmatrix} 0 \\ \boldsymbol{\phi}(k-1,i) \end{bmatrix} &= \frac{\mathbf{x}^T(k-1,i)\mathbf{S}_D(k-2,i)\mathbf{x}(k-1,i)}{\lambda+\boldsymbol{\Psi}^T(k-1,i)\mathbf{x}(k-1,i)} \\
&= \frac{\boldsymbol{\Psi}^T(k-1,i)\mathbf{x}(k-1,i)}{\lambda+\boldsymbol{\Psi}^T(k-1,i)\mathbf{x}(k-1,i)} \qquad (6.46)
\end{aligned}$$

If we recall that the *a priori* forward prediction error can be computed in the following form

$$\mathbf{x}^T(k,i+1)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,i) \end{bmatrix} = e'_f(k,i)$$

and by substituting equations (6.45) and (6.46) in equation (6.44), after some straightforward manipulations we obtain the following time-updating equation for $\delta(k,i)$

$$\begin{aligned} \delta(k,i) &= \lambda\delta(k-1,i) + \frac{\lambda e'_b(k-1,i)e'_f(k,i)}{\lambda + \mathbf{\Psi}^T(k-1,i)\mathbf{x}(k-1,i)} \\ &= \lambda\delta(k-1,i) + \gamma(k-1,i)e'_b(k-1,i)e'_f(k,i) \end{aligned} \tag{6.47}$$

where

$$\begin{aligned} \gamma(k-1,i) &= \frac{\lambda}{\lambda + \mathbf{\Psi}^T(k-1,i)\mathbf{x}(k-1,i)} \\ &= 1 - \boldsymbol{\phi}^T(k-1,i)\mathbf{x}(k-1,i) \end{aligned} \tag{6.48}$$

and the last relation follows from the definition of $\boldsymbol{\phi}(k-1,i)$. Parameter $\gamma(k-1,i)$ is key to relate the *a posteriori* and *a priori* prediction errors, as will be following exposed.

Since, the first lattice-based algorithm we want to derive is based on *a posteriori* errors, the relationship between the *a priori* and *a posteriori* errors is now derived. The *a posteriori* forward prediction error is related to the *a priori* forward prediction error as follows:

$$\begin{aligned} e_f(k,i) &= x(k) - \mathbf{w}_f^T(k,i)\mathbf{x}(k-1,i) \\ &= x(k) - \mathbf{w}_f^T(k-1,i)\mathbf{x}(k-1,i) - \boldsymbol{\phi}^T(k-1,i)\mathbf{x}(k-1,i)e'_f(k,i) \\ &= e'_f(k,i)[1 - \boldsymbol{\phi}^T(k-1,i)\mathbf{x}(k-1,i)] \\ &= e'_f(k,i)\gamma(k-1,i) \end{aligned} \tag{6.49}$$

Similarly, the relationship between *a posteriori* and *a priori* backward prediction errors can be expressed as

$$\begin{aligned} e_b(k,i) &= x(k-i-1) - \mathbf{w}_b^T(k,i)\mathbf{x}(k,i) \\ &= x(k-i-1) - \mathbf{w}_b^T(k-1,i)\mathbf{x}(k,i) - \boldsymbol{\phi}^T(k,i)\mathbf{x}(k,i)e'_b(k,i) \\ &= e'_b(k,i)[1 - \boldsymbol{\phi}^T(k,i)\mathbf{x}(k,i)] \\ &= e'_b(k,i)\gamma(k,i) \end{aligned} \tag{6.50}$$

Parameter $\gamma(k,i)$ is often called conversion factor between *a priori* and *a posteriori* errors.

Using equations (6.49) and (6.50), equation (6.47) can be expressed as

$$\delta(k,i) = \lambda\delta(k-1,i) + \frac{e_b(k-1,i)e_f(k,i)}{\gamma(k-1,i)} \tag{6.51}$$

As a rule each variable of the lattice-based algorithms requires an order-updating equation. Therefore, an order-updating equation for $\gamma(k,i)$ is necessary. This is the objective of the derivations in the following subsection.

6.4.3 Order Updating for $\gamma(k,i)$

Variable $\gamma(k-1,i)$ is defined by

$$\gamma(k-1,i) \;=\; 1 - \boldsymbol{\phi}^T(k-1,i)\mathbf{x}(k-1,i)$$

where $\boldsymbol{\phi}(k-1,i) = \mathbf{S}_D(k-1,i)\mathbf{x}(k-1,i)$. By multiplying the expression of $\boldsymbol{\phi}(k-1,i)$ by $\mathbf{R}_D(k-1,i)$ on both sides, we obtain the following relation.

$$\mathbf{R}_D(k-1,i)\boldsymbol{\phi}(k-1,i) \;=\; \mathbf{x}(k-1,i) \tag{6.52}$$

With this equation, we will be able to derive an order-updating equation for $\boldsymbol{\phi}(k-1,i)$, with the aid of an appropriate partitioning of $\mathbf{R}_D(k-1,i)$.

By partitioning matrix $\mathbf{R}_D(k-1,i)$ as in equation (6.19), it follows that

$$\begin{aligned}
&\mathbf{R}_D(k-1,i)\begin{bmatrix} \boldsymbol{\phi}(k-1,i-1) \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{R}_D(k-1,i-1) & \mathbf{p}_{Db}(k-1,i-1) \\ \mathbf{p}_{Db}^T(k-1,i-1) & \sigma_b^2(k-1) \end{bmatrix}\begin{bmatrix} \boldsymbol{\phi}(k-1,i-1) \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{R}_{Db}(k-1,i-1)\boldsymbol{\phi}(k-1,i-1) \\ \mathbf{p}_{Db}^T(k-1,i-1)\boldsymbol{\phi}(k-1,i-1) \end{bmatrix}
\end{aligned}$$

We can proceed by replacing $\boldsymbol{\phi}(k-1,i-1)$ using equation (6.52) in the last element of the vector above, that is

$$\mathbf{R}_D(k-1,i)\begin{bmatrix} \boldsymbol{\phi}(k-1,i-1) \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{R}_{Db}(k-1,i-1)\boldsymbol{\phi}(k-1,i-1) \\ \mathbf{p}_{Db}^T(k-1,i-1)\mathbf{S}_{Db}(k-1,i-1)\mathbf{x}(k-1,i-1) \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{R}_{Db}(k-1,i-1)\boldsymbol{\phi}(k-1,i-1) \\ \mathbf{w}_b^T(k-1,i-1)\mathbf{x}(k-1,i-1) \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{x}(k-1,i-1) \\ x(k-i) - e_b(k-1,i-1) \end{bmatrix}$$

$$= \mathbf{x}(k-1,i) - \begin{bmatrix} \mathbf{0} \\ e_b(k-1,i-1) \end{bmatrix} \tag{6.53}$$

By multiplying the above equation by $\mathbf{S}_D(k-1,i)$, we have

$$\begin{bmatrix} \boldsymbol{\phi}(k-1,i-1) \\ 0 \end{bmatrix} = \boldsymbol{\phi}(k-1,i) - \mathbf{S}_D(k-1,i) \begin{bmatrix} \mathbf{0} \\ e_b(k-1,i-1) \end{bmatrix} \tag{6.54}$$

Applying the relation above in the definition of the conversion factor, we deduce that

$$\begin{aligned} \gamma(k-1,i) &= 1 - \boldsymbol{\phi}^T(k-1,i)\mathbf{x}(k-1,i) \\ &= \gamma(k-1,i-1) - [\mathbf{0}^T\, e_b(k-1,i)]^T \mathbf{S}_D(k-1,i)\mathbf{x}(k-1,i) \end{aligned} \tag{6.55}$$

This equation can be modified to a more useful form by using the following result:

$$\begin{aligned} \mathbf{S}_{D(k-1,i)} &= \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{S}_{D(k-2,i-1)} \end{bmatrix} \\ &\quad + \frac{1}{\xi_{f_{min}}^d(k-1,i-1)} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,i-1) \end{bmatrix} \begin{bmatrix} 1 & -\mathbf{w}_f^T(k-1,i-1) \end{bmatrix} \end{aligned} \tag{6.56}$$

Proof:

Since

$$\begin{aligned} \mathbf{R}_D(k-1,i) &= \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{R}_D(k-2,i-1) \end{bmatrix} \\ &\quad + \begin{bmatrix} \sigma_f^2(k-1) & \mathbf{p}_{Df}^T(k-2,i-1) \\ \mathbf{p}_{Df}(k-2,i-1) & \mathbf{0}_{i-1,i-1} \end{bmatrix} \end{aligned} \tag{6.57}$$

By assuming equation (6.56) is valid and using (6.57), it is easily seen that

$$\begin{aligned}
&\mathbf{R}_D(k-1,i)\mathbf{S}_D(k-1,i) \\
&= \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{I}_{i-1,i-1} \end{bmatrix} + \begin{bmatrix} 0 & \mathbf{p}_{Df}^T(k-2,i-1)\mathbf{S}_D(k-2,i-1) \\ \mathbf{0} & \mathbf{0}^T \end{bmatrix} \\
&+ \frac{1}{\xi^d_{f_{min}}(k-1,i-1)} \mathbf{R}_D(k-1,i) \begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,i-1) \end{bmatrix} [1 \ -\mathbf{w}_f^T(k-1,i-1)] \\
&= \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{I}_{i-1,i-1} \end{bmatrix} + \begin{bmatrix} 0 & \mathbf{w}_f^T(k-1,i-1) \\ \mathbf{0} & \mathbf{0}_{i-2,i-2} \end{bmatrix} \\
&+ \frac{1}{\xi^d_{f_{min}}(k-1,i-1)} \begin{bmatrix} \xi^d_{f_{min}}(k-1,i-1) \\ \mathbf{0} \end{bmatrix} [1 \ -\mathbf{w}_f^T(k-1,i-1)] \\
&= \begin{bmatrix} 0 & \mathbf{w}_f^T(k-1,i-1) \\ \mathbf{0} & \mathbf{I}_{i-1,i-1} \end{bmatrix} + \begin{bmatrix} 1 & -\mathbf{w}_f^T(k-1,i-1) \\ \mathbf{0} & \mathbf{0}_{i-1,i} \end{bmatrix} = \mathbf{I}_{i,i}
\end{aligned}$$

□

By applying equation (6.56) in (6.55), the following expression for $\gamma(k,i+1)$ is easily derived

$$\begin{aligned}
\gamma(k,i+1) &= 1 - \boldsymbol{\phi}^T(k,i+1)\mathbf{x}(k,i+1) \\
&= \gamma(k-1,i) - \frac{e_f^2(k,i)}{\xi^d_{f_{min}}(k,i)}
\end{aligned} \tag{6.58}$$

Following a similar way to derive (6.56), it can also be shown that

$$\begin{aligned}
\mathbf{S}_D(k-1,i) &= \begin{bmatrix} \mathbf{S}_D(k-1,i-1) & \mathbf{0}_{i-1} \\ \mathbf{0}_{i-1}^T & 0 \end{bmatrix} \\
&+ \frac{1}{\xi^d_{b_{min}}(k-1,i-1)} \begin{bmatrix} -\mathbf{w}_b(k-1,i-1) \\ 1 \end{bmatrix} \left[-\mathbf{w}_b^T(k-1,i-1)\ 1\right]
\end{aligned} \tag{6.59}$$

Now by replacing equation above in equation (6.55) we can show that

$$\begin{aligned}
\gamma(k-1,i) &= \gamma(k-1,i-1) \\
&\quad - \frac{e_b(k-1,i-1)}{\xi^d_{b_{min}}(k-1,i-1)} \left[-\mathbf{w}_b^T(k-1,i-1)\ 1\right] \mathbf{x}(k-1,i) \\
&= \gamma(k-1,i-1) - \frac{e_b^2(k-1,i-1)}{\xi^d_{b_{min}}(k-1,i-1)}
\end{aligned} \tag{6.60}$$

The last equation completes the set of relations required to solve the backward and forward prediction problems. In the following section, the modeling of a reference signal (joint-processor estimation) is discussed.

6.5 JOINT-PROCESS ESTIMATION

In the previous sections, we considered only the forward and backward prediction problems exploring some common features in their solutions. In a more general situation the interest is to predict the behavior of one process represented by $d(k)$ through measurements of a related process contained in $\mathbf{x}(k-1,i)$. Therefore, it is important to derive an adaptive lattice-based realization to match a desired signal $d(k)$ through the minimization of the weighted squared error function given by

$$\begin{aligned}\xi^d(k,i+1) &= \sum_{l=0}^{k} \lambda^{k-l} e^2(l,i+1) \\ &= \sum_{l=0}^{k} \lambda^{k-l} [d(l) - \mathbf{w}^T(k,i+1)\mathbf{x}(l,i+1)]^2 \end{aligned} \tag{6.61}$$

where $y(k,i+1) = \mathbf{w}^T(k,i+1)\mathbf{x}(k,i+1)$ is the adaptive filter output signal, and $e(l,i+1)$ is the *a posteriori* error at a given instant l if the adaptive filter coefficients were fixed at $\mathbf{w}(k,i+1)$. The minimization procedure of $\xi^d(k,i+1)$ is often called joint-process estimation.

The prediction lattice realization generates the forward and backward prediction errors and requires some feedforward coefficients to allow the minimization of $\xi^d(k,i+1)$. In fact, the lattice predictor part in this case works as a signal processing building block which improves the quality of the signals (in the sense of reducing the eigenvalue spread of the autocorrelation matrix) that are inputs to the output taps. The question is where the taps should be placed. We give some statistical arguments for this choice here. First, we repeat, for convenience, the expression of the backward prediction error

$$e_b(k,i+1) = \mathbf{x}^T(k,i+2) \begin{bmatrix} -\mathbf{w}_b(k,i+1) \\ 1 \end{bmatrix}$$

From the orthogonality property of the RLS algorithm, for $k \to \infty$, we can infer that

$$E[e_b(k,i+1)x(k-l)] = 0$$

for $l = 0, 1, \ldots, i+1$. From this equation, it is easy to show that

$$E[e_b(k, i+1)\mathbf{x}^T(k, i+2)] = \mathbf{0}^T$$

If we postmultiply the equation above by $[-\mathbf{w}_b(k,i)\ 1\ 0]^T$, the following result is obtained

$$E\left\{ e_b(k,i+1)\mathbf{x}^T(k,i+2) \begin{bmatrix} -\mathbf{w}_b(k,i) \\ 1 \\ 0 \end{bmatrix} \right\} = E[e_b(k,i+1)e_b(k,i)] = 0$$

This result shows that backward prediction errors of consecutive orders are uncorrelated. Using similar arguments one can show that $E[e_b(k,i+1)e_b(k,l)] = 0$, for $l = 0, 1, \ldots, i$.

In problem 4, it is also shown that backward prediction errors are uncorrelated with each other in the sense of time averaging, as a consequence they should be naturally chosen as inputs to the output taps. The objective function can now be written as

$$\begin{aligned} \xi^d(k,i+1) &= \sum_{l=0}^{k} \lambda^{k-l} e^2(l, i+1) \\ &= \sum_{l=0}^{k} \lambda^{k-l} [d(l) - \hat{\mathbf{e}}_b^T(k,i+1)\mathbf{v}(l,i+1)]^2 \end{aligned} \tag{6.62}$$

where $\hat{\mathbf{e}}_b^T(k,i+1) = [e_b(k,0)\, e_b(k,1) \ldots e_b(k,i)]$ is the backward prediction error vector and $\mathbf{v}^T(k,i+1) = [v_0(k)\ v_1(k)\ \ldots\ v_i(k)]$ is the feedforward coefficient vector.

The main objective of the present section is to derive a time-updating formula for the output tap coefficients. From equations (6.61) and (6.62), it is obvious that the lattice realization generates the optimal estimation by using a parameterization different from that related to the direct form realization. We can derive the updating equations for the elements of the forward coefficient vector using the order-updating equation for the tap coefficients of the direct form realization. Employing equation (6.59), the equivalent optimal solution with the direct form realization can be expressed as

$$\begin{aligned} \mathbf{w}(k,i+1) &= \mathbf{S}_D(k,i+1)\mathbf{p}_D(k,i+1) \\ &= \begin{bmatrix} \mathbf{S}_D(k,i) & \mathbf{0}_i \\ \mathbf{0}_i^T & 0 \end{bmatrix} \mathbf{p}_D(k,i+1) \end{aligned}$$

$$+\frac{1}{\xi^d_{b_{min}}(k,i)}\begin{bmatrix} -\mathbf{w}_b(k,i) \\ 1 \end{bmatrix}[-\mathbf{w}_b^T(k,i)\ 1]\mathbf{p}_D(k,i+1)$$

$$= \begin{bmatrix} \mathbf{w}(k,i) \\ 0 \end{bmatrix} + \frac{\delta_D(k,i)}{\xi^d_{b_{min}}(k,i)}\begin{bmatrix} -\mathbf{w}_b(k,i) \\ 1 \end{bmatrix} \tag{6.63}$$

where

$$\delta_D(k,i) = [-\mathbf{w}_b^T(k,i)\ 1]\mathbf{p}_D(k,i+1)$$

and

$$\mathbf{p}_D(k,i+1) = \sum_{l=0}^{k} \lambda^{k-l}\mathbf{x}(l,i+1)d(l)$$

Since

$$\mathbf{p}_D(k,i+1) = \lambda\mathbf{p}_D(k-1,i+1) + d(k)\mathbf{x}(k,i+1)$$

and

$$\mathbf{w}_b(k,i) = \mathbf{w}_b(k-1,i) + \phi(k,i)e'_b(k,i)$$

and following the same steps to deduce the time update of $\delta(k,i)$ in equation (6.47), we can show that

$$\delta_D(k,i) = \lambda\delta_D(k-1,i) + \frac{e(k,i)e_b(k,i)}{\gamma(k,i)} \tag{6.64}$$

By calculating the output signal of the joint-process estimator using the order-updating equation (6.63) for the direct form realization, we can show that

$$\mathbf{w}^T(k,i+1)\mathbf{x}(k,i+1) = [\mathbf{w}^T(k,i)\ 0]\mathbf{x}(k,i+1) + \frac{\delta_D(k,i)}{\xi^d_{b_{min}}(k,i)}[-\mathbf{w}_b^T(k,i)\ 1]\mathbf{x}(k,i+1) \tag{6.65}$$

This equation can be rewritten as follows:

$$y(k,i+1) = y(k,i) + \frac{\delta_D(k,i)}{\xi^d_{b_{min}}(k,i)}e_b(k,i) \tag{6.66}$$

where it can now be noticed that the joint-predictor output $y(k,i+1)$ is a function of the backward prediction error $e_b(k,i)$. This was the motivation for using in equation (6.63) the decomposition of $\mathbf{S}_D(k,i+1)$ given in (6.59).

The feedforward multiplier coefficients can be identified as

$$v_i(k) = \frac{\delta_D(k,i)}{\xi^d_{b_{min}}(k,i)} \tag{6.67}$$

and the *a posteriori* output error of the adaptive filter of order i from 1 to N are obtained simultaneously, where

$$e(k,i+1) = e(k,i) - v_i(k)e_b(k,i) \tag{6.68}$$

The result above was derived by subtracting $d(k)$ on both sides of equation (6.66). The resulting lattice realization is depicted in Fig 6.2.

We have now available all the relations required to generate the lattice recursive least-squares adaptive filtering algorithm based on *a posteriori* estimation errors. The algorithm is described in Algorithm 6.1.

6.6 TIME RECURSIONS OF THE LEAST-SQUARES ERROR

In this section, we provide a set of relations for the time updating of the minimum LS error of the prediction problems. These relations allow the derivation of two important equations involving the ratio of conversion factor of consecutive order prediction problems, namely $\frac{\gamma(k-1,i+1)}{\gamma(k-1,i)}$ and $\frac{\gamma(k,i+1)}{\gamma(k-1,i)}$. The results provided in this section are required for the derivation of some alternative lattice algorithms such as the error feedback, as well as for the fast RLS algorithms of the following chapter.

By replacing each term in the definition of the minimum weighted least-squares error for the backward prediction problem by their time updating equation, we have (see equations (6.16), (6.17))

$$\begin{aligned}
\xi^d_{b_{min}}(k,i) &= \sigma_b^2(k) - \mathbf{w}_b^T(k,i)\mathbf{p}_{Db}(k,i) \\
&= \sigma_b^2(k) - [\mathbf{w}_b^T(k-1,i) + e'_b(k,i)\boldsymbol{\phi}^T(k,i)][\lambda \mathbf{p}_{Db}(k-1,i) + x(k-i)\mathbf{x}(k,i)] \\
&= \sigma_b^2(k) - \lambda \mathbf{w}_b^T(k-1,i)\mathbf{p}_{Db}(k-1,i) - x(k-i)\mathbf{w}_b^T(k-1,i)\mathbf{x}(k,i) \\
&\quad - \lambda e'_b(k,i)\boldsymbol{\phi}^T(k,i)\mathbf{p}_{Db}(k-1,i) - e'_b(k,i)\boldsymbol{\phi}^T(k,i)\mathbf{x}(k,i)x(k-i) \\
&= x^2(k-i) + \lambda\sigma_b^2(k-1) - \lambda \mathbf{w}_b^T(k-1,i)\mathbf{p}_{Db}(k-1,i) - x(k-i)\mathbf{w}_b^T(k-1,i)\mathbf{x}(k,i) \\
&\quad - \lambda e'_b(k,i)\boldsymbol{\phi}^T(k,i)\mathbf{p}_{Db}(k-1,i) - e'_b(k,i)\boldsymbol{\phi}^T(k,i)\mathbf{x}(k,i)x(k-i)
\end{aligned} \tag{6.69}$$

Algorithm 6.1

Lattice RLS Algorithm Based on *A Posteriori* Errors

Initialization

Do for $i = 0, 1 \ldots, N$
$\delta(-1,i) = \delta_D(-1,i) = 0$ (assuming $x(k) = 0$ for $k < 0$)
$\xi^d_{b_{min}}(-1,i) = \xi^d_{f_{min}}(-1,i) = \epsilon$ (a small positive constant)
$\gamma(-1,i) = 1$
$e_b(-1,i) = 0$
End

Do for $k \geq 0$
$\gamma(k,0) = 1$
$e_b(k,0) = e_f(k,0) = x(k)$ (6.35)
$\xi^d_{b_{min}}(k,0) = \xi^d_{f_{min}}(k,0) = x^2(k) + \lambda\xi^d_{f_{min}}(k-1,0)$ (6.36)
$e(k,0) = d(k)$

For each $k \geq 0$, do for $i = 0, 1 \ldots, N$

$\delta(k,i) = \lambda\delta(k-1,i) + \frac{e_b(k-1,i)e_f(k,i)}{\gamma(k-1,i)}$ (6.51)
$\gamma(k,i+1) = \gamma(k,i) - \frac{e_b^2(k,i)}{\xi^d_{b_{min}}(k,i)}$ (6.60)
$\kappa_b(k,i) = \frac{\delta(k,i)}{\xi^d_{f_{min}}(k,i)}$
$\kappa_f(k,i) = \frac{\delta(k,i)}{\xi^d_{b_{min}}(k-1,i)}$
$e_b(k,i+1) = e_b(k-1,i) - \kappa_b(k,i)e_f(k,i)$ (6.34)
$e_f(k,i+1) = e_f(k,i) - \kappa_f(k,i)e_b(k-1,i)$ (6.33)
$\xi^d_{b_{min}}(k,i+1) = \xi^d_{b_{min}}(k-1,i) - \frac{\delta^2(k,i)}{\xi^d_{f_{min}}(k,i)}$ (6.27)
$\xi^d_{f_{min}}(k,i+1) = \xi^d_{f_{min}}(k,i) - \frac{\delta^2(k,i)}{\xi^d_{b_{min}}(k-1,i)}$ (6.31)

Feedforward Filtering

$\delta_D(k,i) = \lambda\delta_D(k-1,i) + \frac{e(k,i)e_b(k,i)}{\gamma(k,i)}$ (6.64)
$v_i(k) = \frac{\delta_D(k,i)}{\xi^d_{b_{min}}(k,i)}$ (6.67)
$e(k,i+1) = e(k,i) - v_i(k)e_b(k,i)$ (6.68)
End
End

□

By combining the second and third terms we get

$$\lambda[\sigma_b^2(k-1) - \mathbf{w}_b^T(k-1,i)\mathbf{p}_{Db}(k-1,i)] = \lambda\xi_{b_{min}}^d(k-1,i)$$

Similarly, by combining the first, fourth and sixth terms we obtain

$$\begin{aligned} x(k-i)[x(k-i) - \mathbf{w}_b^T(k-1,i)\mathbf{x}(k,i) - e_b'(k,i)\boldsymbol{\phi}^T(k,i)\mathbf{x}(k,i)] \\ = x(k-i)[e_b'(k,i) - e_b'(k,i)\boldsymbol{\phi}^T(k,i)\mathbf{x}(k,i)] \\ = x(k-i)e_b'(k,i)[1 - \boldsymbol{\phi}^T(k,i)\mathbf{x}(k,i)] \end{aligned}$$

Now by applying these results in equation (6.69), we can show that

$$\begin{aligned} \xi_{b_{min}}^d(k,i) &= \lambda\xi_{b_{min}}^d(k-1,i) + x(k-i)e_b'(k,i)[1 - \boldsymbol{\phi}^T(k,i)\mathbf{x}(k,i)] \\ &\quad -\lambda e_b'(k,i)\boldsymbol{\phi}^T(k,i)\mathbf{p}_{Db}(k-1,i) \\ &= \lambda\xi_{b_{min}}^d(k-1,i) + x(k-i)e_b'(k,i) \\ &\quad -e_b'(k,i)\boldsymbol{\phi}^T(k,i)[x(k-i)\mathbf{x}(k,i) + \lambda\mathbf{p}_{Db}(k-1,i)] \end{aligned}$$

If we apply the definition of $\boldsymbol{\phi}(k,i)$ and the equation (6.16) for the backward prediction problem we obtain

$$\begin{aligned} \xi_{b_{min}}^d(k,i) &= \lambda\xi_{b_{min}}^d(k-1,i) + x(k-i)e_b'(k,i) - e_b'(k,i)\boldsymbol{\phi}^T(k,i)\mathbf{p}_{Db}(k,i) \\ &= \lambda\xi_{b_{min}}^d(k-1,i) + x(k-i)e_b'(k,i) \\ &\quad -e_b'(k,i)\mathbf{x}^T(k,i)\mathbf{S}_D(k-1,i)\mathbf{p}_{Db}(k,i) \\ &= \lambda\xi_{b_{min}}^d(k-1,i) + e_b'(k,i)[x(k-i) - \mathbf{w}_b^T(k,i)\mathbf{x}(k,i)] \\ &= \lambda\xi_{b_{min}}^d(k-1,i) + e_b'(k,i)e_b(k,i) \\ &= \lambda\xi_{b_{min}}^d(k-1,i) + \frac{e_b^2(k,i)}{\gamma(k,i)} \end{aligned} \tag{6.70}$$

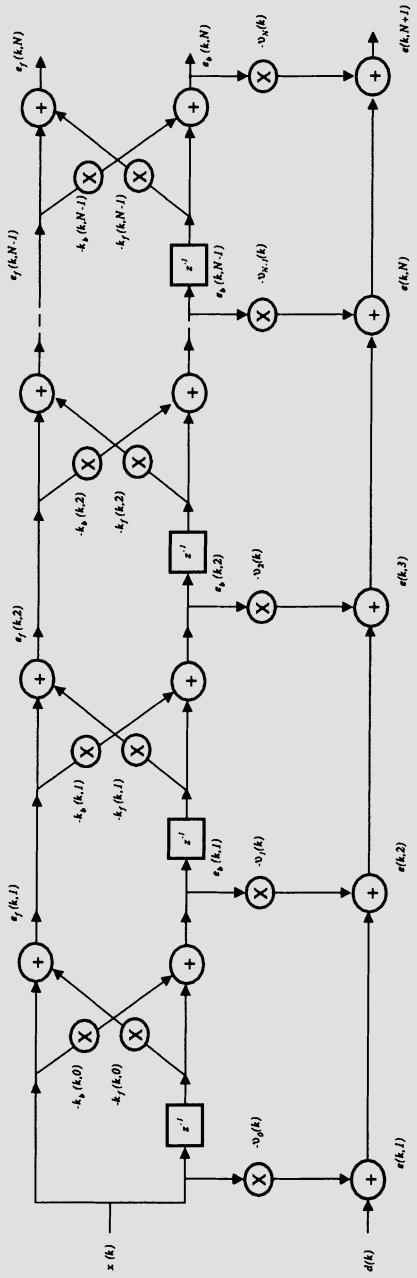


Figure 6.2 Joint-process estimation lattice realization.

Following similar steps to prove the equation above, we can show that

$$\xi^d_{f_{min}}(k,i) = \lambda\xi^d_{f_{min}}(k-1,i) + \frac{e_f^2(k,i)}{\gamma(k-1,i)} \tag{6.71}$$

From the last two equations, we can easily infer the following relations that are useful to derive alternative lattice-based algorithms, namely the normalized and error-feedback algorithms.

$$\begin{aligned}\frac{\lambda\xi^d_{b_{min}}(k-2,i)}{\xi^d_{b_{min}}(k-1,i)} &= 1 - \frac{e_b^2(k-1,i)}{\gamma(k-1,i)\xi^d_{b_{min}}(k-1,i)} \\ &= \frac{\gamma(k-1,i+1)}{\gamma(k-1,i)}\end{aligned} \tag{6.72}$$

and

$$\begin{aligned}\frac{\lambda\xi^d_{f_{min}}(k-1,i)}{\xi^d_{f_{min}}(k,i)} &= 1 - \frac{e_f^2(k,i)}{\gamma(k-1,i)\xi^d_{f_{min}}(k,i)} \\ &= \frac{\gamma(k,i+1)}{\gamma(k-1,i)}\end{aligned} \tag{6.73}$$

where equations (6.60) and (6.58) were used in the derivation of the right-hand-side expressions of the equations above, respectively.

6.7 NORMALIZED LATTICE RLS ALGORITHM

An alternative form of the lattice RLS algorithm can be obtained by applying a wise normalization to the internal variables of the algorithm, keeping their magnitude bounded by one. The normalized lattice is specially suitable for fixed-point arithmetic implementation. Also, this algorithm requires fewer recursions and variables than the unnormalized lattices, i.e., only three equations per prediction section per time sample.

6.7.1 Basic Order Recursions

A natural way to normalize the backward and forward prediction errors is to divide them by the square root of the corresponding weighted least-squares error. However, it will be shown that a wiser strategy leads to a reduction in the number of recursions. At the same time, we must think of a way to normalize variable $\delta(k,i)$. In the process of normalizing $e_f(k,i), e_b(k,i)$, and $\delta(k,i)$, we can reduce the number of equations by eliminating the conversion variable $\gamma(k,i+1)$. Note that $\gamma(k,i+1)$ is originally normalized. These goals can be reached if the normalization of $\delta(k,i)$ is performed by

$$\overline{\delta}(k,i) = \frac{\delta(k,i)}{\sqrt{\xi^d_{f_{min}}(k,i)\xi^d_{b_{min}}(k-1,i)}} \tag{6.74}$$

By noting that the conversion variable $\gamma(k-1,i)$ appears dividing the product $e_f(k,i)e_b(k-1,i)$ in the time-updating formula (6.51), we can easily devise a way to perform the normalization of the prediction errors leading to its elimination. The appropriate normalization of the forward and backward estimation errors are respectively performed as follows:

$$\overline{e}_f(k,i) = \frac{e_f(k,i)}{\sqrt{\gamma(k-1,i)\xi^d_{f_{min}}(k,i)}} \tag{6.75}$$

$$\overline{e}_b(k,i) = \frac{e_b(k,i)}{\sqrt{\gamma(k,i)\xi^d_{b_{min}}(k,i)}} \tag{6.76}$$

where the terms $\sqrt{\xi^d_{f_{min}}(k,i)}$ and $\sqrt{\xi^d_{b_{min}}(k,i)}$ perform the power normalization whereas $\sqrt{\gamma(k-1,i)}$ and $\sqrt{\gamma(k,i)}$ perform the so-called angle normalization, since $\gamma(k,i)$ is related to the angle between the spaces spanned by $\mathbf{x}(k-1,i)$ and $\mathbf{x}(k,i)$.

From the equation above and (6.51), we can show that

$$\begin{aligned}\overline{\delta}(k,i)\sqrt{\xi^d_{f_{min}}(k,i)\xi^d_{b_{min}}(k-1,i)} &= \lambda\overline{\delta}(k-1,i)\sqrt{\xi^d_{f_{min}}(k-1,i)\xi^d_{b_{min}}(k-2,i)} \\ &\quad +\overline{e}_b(k-1,i)\overline{e}_f(k,i)\sqrt{\xi^d_{f_{min}}(k,i)\xi^d_{b_{min}}(k-1,i)}\end{aligned} \tag{6.77}$$

Therefore,

$$\overline{\delta}(k,i) = \lambda\overline{\delta}(k-1,i)\sqrt{\frac{\xi^d_{f_{min}}(k-1,i)\xi^d_{b_{min}}(k-2,i)}{\xi^d_{f_{min}}(k,i)\xi^d_{b_{min}}(k-1,i)}} + \overline{e}_b(k-1,i)\overline{e}_f(k,i) \tag{6.78}$$

We show now that the term under the square root in the equation above can be expressed in terms of the normalized errors. That is

$$\begin{aligned}\frac{\lambda\xi^d_{b_{min}}(k-2,i)}{\xi^d_{b_{min}}(k-1,i)} &= \frac{\gamma(k-1,i+1)}{\gamma(k-1,i)} \\ &= 1 - \frac{e^2_b(k-1,i)}{\gamma(k-1,i)\xi^d_{b_{min}}(k-1,i)} \\ &= 1 - \overline{e}^2_b(k-1,i)\end{aligned} \tag{6.79}$$

and

$$\begin{aligned}\frac{\lambda\xi^d_{f_{min}}(k-1,i)}{\xi^d_{f_{min}}(k,i)} &= \frac{\gamma(k,i+1)}{\gamma(k-1,i)} \\ &= 1 - \frac{e^2_f(k,i)}{\gamma(k-1,i)\xi^d_{f_{min}}(k,i)} \\ &= 1 - \overline{e}^2_f(k,i)\end{aligned} \tag{6.80}$$

Applying the last two equations in (6.78), we can easily show that

$$\overline{\delta}(k,i) = \overline{\delta}(k-1,i)\sqrt{(1-\overline{e}^2_b(k-1,i))(1-\overline{e}^2_f(k,i))} + \overline{e}_b(k-1,i)\overline{e}_f(k,i) \tag{6.81}$$

Following a similar procedure to derive the time-updating equation for $\overline{\delta}(k,i)$, one can derive the order-updating equation of the normalized forward and backward prediction errors. In the case of the forward prediction error the following order-updating relation results

$$\overline{e}_f(k,i+1) = \left[\overline{e}_f(k,i) - \overline{\delta}(k,i)\overline{e}_b(k-1,i)\right]\sqrt{\frac{\xi^d_{f_{min}}(k,i)}{\xi^d_{f_{min}}(k,i+1)}}\sqrt{\frac{\gamma(k-1,i)}{\gamma(k-1,i+1)}} \tag{6.82}$$

Here again, we can express the terms under the square roots in terms of the normalized variables. Using equations (6.31), (6.74), and (6.80), it can be easily shown that

$$\bar{e}_f(k,i+1) = \frac{\bar{e}_f(k,i) - \bar{\delta}(k,i)\bar{e}_b(k-1,i)}{\sqrt{1-\bar{\delta}^2(k,i)}\sqrt{1-\bar{e}_b^2(k-1,i)}} \tag{6.83}$$

If the same steps to derive $\bar{e}_f(k,i+1)$ are followed, we can derive the order-updating equation for the backward prediction error

$$\bar{e}_b(k,i+1) = \frac{\bar{e}_b(k-1,i) - \bar{\delta}(k,i)\bar{e}_f(k,i)}{\sqrt{1-\bar{\delta}^2(k,i)}\sqrt{1-\bar{e}_f^2(k,i)}} \tag{6.84}$$

6.7.2 Feedforward Filtering

The procedure to generate the joint-processor estimator is repeated here, but using normalized variables. Define

$$\bar{\delta}_D(k,i) = \frac{\delta_D(k,i)}{\sqrt{\xi_{min}^d(k,i)\xi_{b_{min}}^d(k,i)}} \tag{6.85}$$

and

$$\bar{e}(k,i) = \frac{e(k,i)}{\sqrt{\gamma(k,i)\xi_{min}^d(k,i)}} \tag{6.86}$$

Using a similar approach to derive equation (6.31), one can show that

$$\xi_{min}^d(k,i+1) = \xi_{min}^d(k,i) - \frac{\delta_D^2(k,i)}{\xi_{b_{min}}^d(k,i)} \tag{6.87}$$

The same procedure used to derive the order-updating equations for the normalized prediction errors and the parameter $\bar{\delta}(k,i)$ can be followed to derive the equivalent parameters in the joint-process estimation case. For the *a posteriori* output error the following equation results

$$\bar{e}(k,i+1) = \sqrt{\frac{\gamma(k,i)}{\gamma(k,i+1)}}\sqrt{\frac{\xi_{min}^d(k,i)}{\xi_{min}^d(k,i+1)}}\left[\bar{e}(k,i) - \bar{\delta}_D(k,i)\bar{e}_b(k,i)\right]$$

$$= \frac{1}{\sqrt{1-\overline{e}_b^2(k,i)}} \frac{1}{\sqrt{1-\overline{\delta}_D^2(k,i)}} \left[\overline{e}(k,i) - \overline{\delta}_D(k,i)\overline{e}_b(k,i)\right] \tag{6.88}$$

The order-updating equation of $\overline{\delta}_D(k,i)$ is

$$\begin{aligned} \overline{\delta}_D(k,i) &= \sqrt{\frac{\lambda^2 \xi_{min}^d(k-1,i)\xi_{b_{min}}^d(k-1,i)}{\xi_{min}^d(k,i)\xi_{b_{min}}^d(k,i)}}\overline{\delta}_D(k-1,i) + \overline{e}(k,i)\overline{e}_b(k,i) \\ &= \sqrt{(1-\overline{e}_b^2(k,i))(1-\overline{e}^2(k,i))}\overline{\delta}_D(k-1,i) + \overline{e}(k,i)\overline{e}_b(k,i) \end{aligned} \tag{6.89}$$

where it was used the fact that

$$\frac{\lambda \xi_{min}^d(k-1,i)}{\xi_{min}^d(k,i)} = 1 - \overline{e}^2(k,i) \tag{6.90}$$

The normalized lattice RLS algorithm based on *a posteriori* errors is described in Algorithm 6.2.

Notice that in the updating formulas of the normalized errors, the terms involving the square root operation could be conveniently implemented through separate multiplier coefficients, namely $\eta_f(k,i)$, $\eta_b(k,i)$, and $\eta_D(k,i)$. In this way, one can perform the order updating by calculating the numerator first and proceeding with a single multiplication. These coefficients are given by

$$\eta_f(k,i+1) = \frac{1}{\sqrt{1-\overline{\delta}^2(k,i)}\sqrt{1-\overline{e}_b^2(k-1,i)}} \tag{6.91}$$

$$\eta_b(k,i+1) = \frac{1}{\sqrt{1-\overline{\delta}^2(k,i)}\sqrt{1-\overline{e}_f^2(k,i)}} \tag{6.92}$$

$$\eta_D(k,i+1) = \frac{1}{\sqrt{1-\overline{e}_b^2(k,i)}\sqrt{1-\overline{\delta}_D^2(k,i)}} \tag{6.93}$$

Algorithm 6.2

Normalized *A Posteriori* Error LRLS Algorithm

Initialization

Do for $i = 0, 1 \ldots, N$
$\bar{\delta}(-1, i) = 0$ (assuming $x(k) = d(k) = 0$ for $k < 0$)
$\bar{\delta}_D(-1, i) = 0$
$\bar{e}_b(-1, i) = 0$
End

$\sigma_x^2(-1) = \lambda\sigma_d^2(-1) = \epsilon$ (ϵ small positive constant)

Do for $k \geq 0$
$\sigma_x^2(k) = \lambda\sigma_x^2(k-1) + x^2(k)$ (Input signal energy)
$\sigma_d^2(k) = \lambda\sigma_d^2(k-1) + d^2(k)$ (Reference signal energy)
$\bar{e}_b(k, 0) = \bar{e}_f(k, 0) = x(k)/\sigma_x(k)$
$\bar{e}(k, 0) = d(k)/\sigma_d(k)$

For each $k \geq 0$, do for $i = 0, 1 \ldots, N$

$$\bar{\delta}(k, i) = \bar{\delta}(k-1, i)\sqrt{(1 - \bar{e}_b^2(k-1, i))(1 - \bar{e}_f^2(k, i))} + \bar{e}_b(k-1, i)\bar{e}_f(k, i) \tag{6.81}$$

$$\bar{e}_b(k, i+1) = \frac{\bar{e}_b(k-1, i) - \bar{\delta}(k, i)\bar{e}_f(k, i)}{\sqrt{(1 - \bar{\delta}^2(k, i))(1 - \bar{e}_f^2(k, i))}} \tag{6.84}$$

$$\bar{e}_f(k, i+1) = \frac{\bar{e}_f(k, i) - \bar{\delta}(k, i)\bar{e}_b(k-1, i)}{\sqrt{(1 - \bar{\delta}^2(k, i))(1 - \bar{e}_b^2(k-1, i))}} \tag{6.83}$$

Feedforward Filter

$$\bar{\delta}_D(k, i) = \bar{\delta}_D(k-1, i)\sqrt{(1 - \bar{e}_b^2(k, i))(1 - \bar{e}^2(k, i))} + \bar{e}(k, i)\bar{e}_b(k, i) \tag{6.89}$$

$$\bar{e}(k, i+1) = \frac{1}{\sqrt{(1 - \bar{e}_b^2(k, i))(1 - \bar{\delta}_D^2(k, i))}} \left[\bar{e}(k, i) - \bar{\delta}_D(k, i)\bar{e}_b(k, i)\right] \tag{6.88}$$

End

End

□

With these multipliers it is straightforward to obtain the structure for the joint-processor estimator depicted in Fig. 6.3.

The unique feature of the normalized lattice algorithm is the reduced number of equations and variables, at the expense of employing a number of square root operations. These operations can be costly to be implemented in most types of hardware architectures. Another interesting feature of the normalized lattice algorithm is that the forgetting factor λ does not appear in the internal updating equations, it appears only in the calculation of the energy of the input and reference signals. This property may be advantageous from the computational point of view in situations where there is a need to vary the value of λ.

6.8 ERROR-FEEDBACK LATTICE RLS ALGORITHM

The reflection coefficients of the lattice algorithm were so far updated in an indirect way, without time recursions. This section describes an alternative form to update the reflection coefficients using time updating. These updating equations are recursive in nature and are often called direct updating, since the updating equations used for $\kappa_b(k,i)$ and $\kappa_f(k,i)$ in Algorithm 6.1 are dependent exclusively of quantities other than past reflection coefficients. Algorithms employing the recursive time updating are called error-feedback lattice RLS algorithms. These algorithms have better numerical properties than their indirect updating counterparts [3].

6.8.1 Recursive Formulas for the Reflection Coefficients

The derivation of a direct updating equation for $\kappa_f(k,i)$ starts by replacing $\delta(k,i)$ by its time-updating equation (6.51)

$$\begin{aligned}\kappa_f(k,i) &= \frac{\delta(k,i)}{\xi^d_{b_{min}}(k-1,i)} \\ &= \frac{\lambda\delta(k-1,i)}{\xi^d_{b_{min}}(k-1,i)} + \frac{e_b(k-1,i)e_f(k,i)}{\gamma(k-1,i)\xi^d_{b_{min}}(k-1,i)}\end{aligned}$$

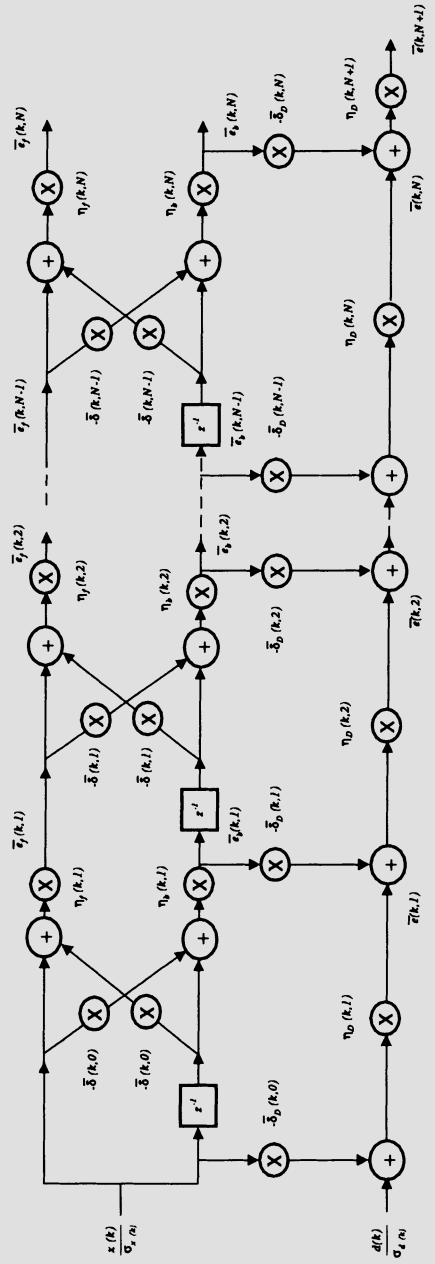


Figure 6.3 Joint-process estimation normalized lattice realization.

By multiplying and dividing the first term by $\xi^d_{b_{min}}(k-2,i)$ and next using equation (6.72) in the first and second terms, we obtain

$$\begin{aligned}
\kappa_f(k,i) &= \frac{\delta(k-1,i)}{\xi^d_{b_{min}}(k-2,i)} \frac{\lambda\xi^d_{b_{min}}(k-2,i)}{\xi^d_{b_{min}}(k-1,i)} + \frac{e_b(k-1,i)e_f(k,i)}{\gamma(k-1,i)\xi^d_{b_{min}}(k-1,i)} \\
&= \kappa_f(k-1,i)\frac{\gamma(k-1,i+1)}{\gamma(k-1,i)} + \frac{e_b(k-1,i)e_f(k,i)\gamma(k-1,i+1)}{\gamma^2(k-1,i)\lambda\xi^d_{b_{min}}(k-2,i)} \\
&= \frac{\gamma(k-1,i+1)}{\gamma(k-1,i)}\left[\kappa_f(k-1,i) + \frac{e_b(k-1,i)e_f(k,i)}{\gamma(k-1,i)\lambda\xi^d_{b_{min}}(k-2,i)}\right] \qquad (6.94)
\end{aligned}$$

Similarly, using equations (6.51) and (6.73) it is straightforward to show that

$$\kappa_b(k,i) = \frac{\gamma(k,i+1)}{\gamma(k-1,i)}\left[\kappa_b(k-1,i) + \frac{e_b(k-1,i)e_f(k,i)}{\gamma(k-1,i)\lambda\xi^d_{f_{min}}(k-1,i)}\right] \qquad (6.95)$$

The feedforward coefficients can also be time updated in a recursive form, by appropriately combining equations (6.64), (6.67), and (6.72). The time-recursive updating equation for $v_i(k)$ is

$$v_i(k) = \frac{\gamma(k,i+1)}{\gamma(k,i)}\left[v_i(k-1) + \frac{e(k,i)e_b(k,i)}{\gamma(k,i)\lambda\xi^d_{b_{min}}(k-1,i)}\right] \qquad (6.96)$$

The error-feedback LRLS algorithm described in Algorithm 6.3 employs the equations (6.94), (6.95), and (6.96). This algorithm is directly derived from Algorithm 6.1.

Alternative *a posteriori* LRLS algorithms can be obtained if we replace equations (6.27) and (6.31) by (6.70) and (6.72) in Algorithms 6.1 and 6.3, respectively. These modifications as well as others possible do not change the behavior of the LRLS algorithm when implemented with infinite precision (long wordlength). Differences exist in computational complexity and in the effects of quantization error propagation.

Algorithm 6.3

Error-Feedback LRLS Algorithm Based on *A Posteriori* Errors

Initialization

Do for $i = 0, 1 \ldots, N$

$\kappa_b(-1,i) = \kappa_f(-1,i) = v_i(-1) = \delta(-1,i) = 0, \gamma(-1,i) = 1$

$\xi^d_{b_{min}}(-2,i) = \xi^d_{b_{min}}(-1,i) = \xi^d_{f_{min}}(-1,i) = \epsilon$ (a small positive constant)

$e_b(-1,i) = 0$

End

Do for $k \geq 0$

$\gamma(k,0) = 1$

$e_b(k,0) = e_f(k,0) = x(k)$ (6.35)

$\xi^d_{f_{min}}(k,0) = \xi^d_{b_{min}}(k,0) = x^2(k) + \lambda \xi^d_{f_{min}}(k-1,0)$ (6.36)

$e(k,0) = d(k)$

For each $k \geq 0$, do for $i = 0, 1 \ldots, N$

$$\delta(k,i) = \lambda \delta(k-1,i) + \frac{e_b(k-1,i)e_f(k,i)}{\gamma(k-1,i)} \quad (6.51)$$

$$\gamma(k,i+1) = \gamma(k,i) - \frac{e_b^2(k,i)}{\xi^d_{b_{min}}(k,i)} \quad (6.60)$$

$$\kappa_f(k,i) = \frac{\gamma(k-1,i+1)}{\gamma(k-1,i)} \left[\kappa_f(k-1,i) + \frac{e_b(k-1,i)e_f(k,i)}{\gamma(k-1,i)\lambda \xi^d_{b_{min}}(k-2,i)} \right] \quad (6.94)$$

$$\kappa_b(k,i) = \frac{\gamma(k,i+1)}{\gamma(k-1,i)} \left[\kappa_b(k-1,i) + \frac{e_b(k-1,i)e_f(k,i)}{\gamma(k-1,i)\lambda \xi^d_{f_{min}}(k-1,i)} \right] \quad (6.95)$$

$$e_b(k,i+1) = e_b(k-1,i) - \kappa_b(k,i)e_f(k,i) \quad (6.34)$$

$$e_f(k,i+1) = e_f(k,i) - \kappa_f(k,i)e_b(k-1,i) \quad (6.33)$$

$$\xi^d_{f_{min}}(k,i+1) = \xi^d_{f_{min}}(k,i) - \frac{\delta^2(k,i)}{\xi^d_{b_{min}}(k-1,i)} \quad (6.31)$$

$$\xi^d_{b_{min}}(k,i+1) = \xi^d_{b_{min}}(k-1,i) - \frac{\delta^2(k,i)}{\xi^d_{f_{min}}(k,i)} \quad (6.27)$$

Feedforward Filtering

$$v_i(k) = \frac{\gamma(k,i+1)}{\gamma(k,i)} \left[v_i(k-1) + \frac{e(k,i)e_b(k,i)}{\gamma(k,i)\lambda \xi^d_{b_{min}}(k-1,i)} \right] \quad (6.96)$$

$$e(k,i+1) = e(k,i) - v_i(k)e_b(k,i) \quad (6.68)$$

End

End

□

6.9 LATTICE RLS ALGORITHM BASED ON *A PRIORI* ERRORS

The lattice algorithms presented so far are based on *a posteriori* errors, however alternative algorithms based on *a priori* errors exist and one of them is derived in this section.

The time updating of the quantity $\delta(k,i)$ as a function of the *a priori* errors was previously derived, see equation (6.47), and is repeated here for convenience.

$$\delta(k,i) = \lambda\delta(k-1,i) + \gamma(k-1,i)e'_b(k-1,i)e'_f(k,i) \tag{6.97}$$

The time updating of the forward prediction *a priori* error can be obtained by using equation (6.32) as follows:

$$\begin{aligned} e'_f(k,i+1) &= \mathbf{x}^T(k,i+2)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,i+1) \end{bmatrix} \\ &= \mathbf{x}^T(k,i+2)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,i) \\ 0 \end{bmatrix} \\ &\quad + \frac{\delta(k-1,i)}{\xi^d_{b_{min}}(k-2,i)}\mathbf{x}^T(k,i+2)\begin{bmatrix} 0 \\ \mathbf{w}_b(k-2,i) \\ -1 \end{bmatrix} \\ &= e'_f(k,i) - \frac{\delta(k-1,i)}{\xi^d_{b_{min}}(k-2,i)}e'_b(k-1,i) \\ &= e'_f(k,i) - \kappa_f(k-1,i)e'_b(k-1,i) \end{aligned} \tag{6.98}$$

With equation (6.28), we can generate the time-updating equation of the backward prediction *a priori* error as follows:

$$\begin{aligned} e'_b(k,i+1) &= \mathbf{x}^T(k,i+2)\begin{bmatrix} 0 \\ -\mathbf{w}_b(k-2,i) \\ 1 \end{bmatrix} \\ &\quad - \frac{\delta(k-1,i)}{\xi^d_{f_{min}}(k-1,i)}\mathbf{x}^T(k,i+2)\begin{bmatrix} -1 \\ \mathbf{w}_f(k-1,i) \\ 0 \end{bmatrix} \\ &= e'_b(k-1,i) - \frac{\delta(k-1,i)}{\xi^d_{f_{min}}(k-1,i)}e'_f(k,i) \\ &= e'_b(k-1,i) - \kappa_b(k-1,i)e'_f(k,i) \end{aligned} \tag{6.99}$$

The order updating of $\gamma(k-1,i)$ can be easily derived by employing the relations of equations (6.50) and (6.60). The result is

$$\gamma(k-1,i+1) = \gamma(k-1,i) - \frac{\gamma^2(k-1,i)e_b'^2(k-1,i)}{\xi_{b_{min}}^d(k-1,i)} \quad (6.100)$$

The updating of the feedforward coefficients of the lattice realization based on *a priori* errors is performed by the equations below.

$$\delta_D(k,i) = \lambda\delta_D(k-1,i) + \gamma(k,i)e_b'(k,i)e'(k,i) \quad (6.101)$$

$$e'(k,i+1) = e'(k,i) - v_i(k-1)e_b'(k,i) \quad (6.102)$$

$$v_i(k-1) = \frac{\delta_D(k-1,i)}{\xi_{b_{min}}^d(k-1,i)} \quad (6.103)$$

The derivations are omitted since they follow the same steps of the predictor equations.

An LRLS algorithm based on *a priori* errors is described in Algorithm 6.4. The normalized and error-feedback versions of the LRLS algorithm based on *a priori* errors also exist and their derivations are left as problems.

6.10 QUANTIZATION EFFECTS

A major issue related to the implementation of adaptive filters is their behavior when implemented with finite-precision arithmetic. In particular, the roundoff errors arising from the quantization of the internal quantities of an algorithm propagate internally and can even cause instability. The numerical stability and accuracy are algorithm dependent. In this section, we summarize some of the results obtained in the literature related to the LRLS algorithms [3], [7]-[8].

One of the first attempts to study the numerical accuracy of the lattice algorithms was reported in [7]. Special attention was given to the normalized lattice RLS algorithm, since this algorithm is suitable for fixed-point arithmetic implementation, due to its internal normalization. In this study, it was shown that the bias error in the reflection coefficients was more significant than the variance of the estimate error. The bias in the estimated reflection coefficients is mainly caused by the quantization error associated with the calculation of the

Algorithm 6.4

LRLS Algorithm Based on *A Priori* Errors

Initialization

Do for $i = 0, 1 \ldots, N$

$\delta(-1,i) = \delta_D(-1,i) = 0$ (assuming $x(k) = 0$ for $k < 0$)

$\gamma(-1,i) = 1$

$\xi^d_{b_{min}}(-1,i) = \xi^d_{f_{min}}(-1,i) = \epsilon$ (a small positive constant)

$e'_b(-1,i) = 0$

End

Do for $k \geq 0$

$\gamma(k,0) = 1$

$e'_b(k,0) = e'_f(k,0) = x(k)$

$\xi^d_{f_{min}}(k,0) = \xi^d_{b_{min}}(k,0) = x^2(k) + \lambda \xi^d_{f_{min}}(k-1,0)$

$e'(k,0) = d(k)$

For each $k \geq 0$, do for $i = 0, 1 \ldots, N$

$\delta(k,i) = \lambda \delta(k-1,i) + \gamma(k-1,i) e'_b(k-1,i) e'_f(k,i)$ (6.47)

$\gamma(k,i+1) = \gamma(k,i) - \frac{\gamma^2(k,i) e'^2_b(k,i)}{\xi^d_{b_{min}}(k,i)}$ (6.100)

$\kappa_f(k-1,i) = \frac{\delta(k-1,i)}{\xi^d_{b_{min}}(k-2,i)}$

$\kappa_b(k-1,i) = \frac{\delta(k-1,i)}{\xi^d_{f_{min}}(k-1,i)}$

$e'_b(k,i+1) = e'_b(k-1,i) - \kappa_b(k-1,i) e'_f(k,i)$ (6.99)

$e'_f(k,i+1) = e'_f(k,i) - \kappa_f(k-1,i) e'_b(k-1,i)$ (6.98)

$\xi^d_{f_{min}}(k,i+1) = \xi^d_{f_{min}}(k,i) - \frac{\delta^2(k,i)}{\xi^d_{b_{min}}(k-1,i)}$ (6.31)

$\xi^d_{b_{min}}(k,i+1) = \xi^d_{b_{min}}(k-1,i) - \frac{\delta^2(k,i)}{\xi^d_{f_{min}}(k,i)}$ (6.27)

Feedforward Filtering

$\delta_D(k,i) = \lambda \delta_D(k-1,i) + \gamma(k,i) e'_b(k,i) e'(k,i)$ (6.101)

$v_i(k-1) = \frac{\delta_D(k-1,i)}{\xi^d_{b_{min}}(k-1,i)}$ (6.103)

$e'(k,i+1) = e'(k,i) - v_i(k-1) e'_b(k,i)$ (6.102)

End

End

□

square roots of $[1-\overline{e}_b^2(k-1,i)]$ and $[1-\overline{e}_f^2(k,i)]$ assuming they are calculated separately. An upper bound for this quantization error is given by

$$m_{sq} = 2^{-b} \tag{6.104}$$

assuming that b is the number of bits after the sign bit and that quantization is performed through rounding. In the analysis, the basic assumption that $1-\lambda \gg 2^{-b+1}$ was used. The upper bound of the bias error in the reflection coefficients is then given by [7]

$$\Delta\overline{\delta}(k,i) = \frac{2^{-b+1}\overline{\delta}(k,i)}{1-\lambda} \tag{6.105}$$

Obviously, the accuracy of this result depends on the validity of the assumptions used in the analysis [7], however it is a good indication of how the bias is generated in the reflection coefficients. It should also be noted that the result above is valid as long as the updating of the related reflection coefficient does not stop. An analysis for the case the updating stops is also included in [7].

The bias error of a given stage of the lattice realization propagates to the succeeding stages and its accumulation in the prediction errors can be expressed as

$$\Delta\overline{e}_b^2(k,i+1) = \Delta\overline{e}_f^2(k,i+1) \approx 2^{-b+2}\sum_{l=0}^{i}\frac{\overline{\delta}^2(k,l)}{1-\overline{\delta}^2(k,l)} \tag{6.106}$$

for $i = 0, 1, \ldots, N$. This equation indicates that whenever the value of the parameter $\overline{\delta}^2(k,l)$ is small, the corresponding term in the summation is also small. On the other hand, if the value of this parameter tends to one the corresponding term of the summation is large. Also note that the accumulated error tends to grow as the number of sections of the lattice is increased. In a finite-precision implementation, it is possible to determine the maximum order that the lattice can have such that the error signals at the end of the realization still represent actual signals and not only accumulated quantization noise.

The lattice algorithms remain stable even when using quite short wordlength in fixed- and floating-point implementations. In terms of accuracy the error-feedback algorithms are usually better than the conventional LRLS algorithms [3]. The reduction in the quantization effects of the error-feedback LRLS algorithms is verified in [3], where a number of examples show satisfactory performance for implementation with less than 10 bits in fixed-point arithmetic.

Another investigation examines the finite-wordlength implementation employing floating-point arithmetic of the unnormalized lattice with and without error feedback [8]. As expected, the variance of the accumulated error in the reflection coefficients of the error-feedback algorithms are smaller than that for the conventional LRLS algorithm. Another important issue relates to the so-called self-generated noise, that originates in the internal stages of the lattice realization when the order of adaptive filter is greater than necessary. In the cases where the signal-to-noise ratio is high in the desired signal, the internal signals of the last stages of the lattice realization can reach the quantization level and start self-generated noise, leading to an excess of mean-square error and possibly to instability. The stability problem can be avoided by turning off the stages after the one where the weighted forward and backward squared errors are smaller than a given threshold.

Example 6.1

The system identification problem described in Chapter 3 (subsection 3.6.2) is solved using the lattice algorithms presented in the present chapter. The main objective is to compare the performance of the algorithms when implemented in finite precision.

Solution:

We present here the results of using the unnormalized, the normalized and error-feedback *a posteriori* lattice RLS algorithms in the system identification example. All results presented here were obtained by running 200 independent experiments and calculating the average of the quantities of interest. We consider the case of eigenvalue spread 20, and $\lambda = 0.99$. Parameter ϵ was 0.1, 0.01, and 0.1 for the unnormalized, the normalized, and the error-feedback lattice filters, respectively. The measured misadjustments of the lattice algorithms are given in Table 6.1. As expected, the results are close to those obtained by the conventional RLS algorithm, where in the latter the misadjustment was 0.0421. Not included is the result for the normalized lattice because the *a posteriori* error is not available, in this case the measured normalized MSE was 0.00974.

Table 6.2 summarizes the results obtained by the implementation of the lattice algorithms with finite precision. Parameter ϵ in the finite-precision implementation was 0.1, 0.04 and 0.5 for the unnormalized, normalized and error-feedback lattices, respectively. These values of ϵ assured a good convergence behavior of the algorithms in this experiment. In short wordlength implementation of the

lattice algorithms, it is advisable to test if the denominator expressions of the algorithm steps involving division are not rounded to zero. In case of detection of a zero denominator, replace its value by the value of the least significant bit. Table 6.2 shows that for the unnormalized and error-feedback lattices, the mean-squared errors are comparable to the case of the conventional RLS previously shown in Table 5.2. The normalized lattice was found more sensible to quantization errors due to its higher computational complexity. The errors introduced by the calculations to obtain $\mathbf{w}(k)_Q$ from the lattice coefficients was the main reason for the increased values of $E[||\Delta\mathbf{w}(k)_Q||^2]$ shown in Table 6.2. Therefore, this result should not be considered as an indication of poor performance of the normalized lattice implemented in finite precision.

□

Table 6.1 Evaluation of the Lattice RLS Algorithms

Algorithm	Misadjustment
Unnorm.	0.0416
Error Feed.	0.0407

Table 6.2 Results of the Finite-Precision Implementation of the Lattice RLS Algorithms

	$\xi(k)_Q$			$E[\|\|\Delta\mathbf{W}(k)_Q\|\|^2]$		
No. of bits	Unnorm.	Norm.	Error Feed.	Unnorm.	Norm.	Error Feed.
16	$1.563\ 10^{-3}$	$8.081\ 10^{-3}$	$1.555\ 10^{-3}$	$9.236\ 10^{-4}$	$2.043\ 10^{-3}$	$9.539\ 10^{-4}$
12	$1.545\ 10^{-3}$	$8.096\ 10^{-3}$	$1.567\ 10^{-3}$	$9.317\ 10^{-4}$	$2.201\ 10^{-3}$	$9.271\ 10^{-4}$
10	$1.587\ 10^{-3}$	$10.095\ 10^{-3}$	$1.603\ 10^{-3}$	$9.347\ 10^{-4}$	$4.550\ 10^{-3}$	$9.872\ 10^{-4}$

Example 6.2

The channel equalization example first described in subsection 3.6.3 is used in simulations using the lattice RLS algorithm with error feedback. In the present example use a 25th-order equalizer.

Solution:

Applying the error-feedback lattice RLS algorithm, using $\lambda = 0.99$, with a 25th-order equalizer, we obtained after 100 iterations the equalizer whose impulse

response is shown in Fig. 6.4. The appropriate value of L for this case was 18. The algorithm was initialized with $\epsilon = 0.1$.

The convolution of this response with the channel impulse response is depicted in Fig. 6.5, which clearly approximates an impulse. In this case, the measured MSE was 0.3056, a value comparable with that obtained with the LMS algorithm in the example of subsection 3.6.3. Note that in the LMS case a 50th-order equalizer was used.

□

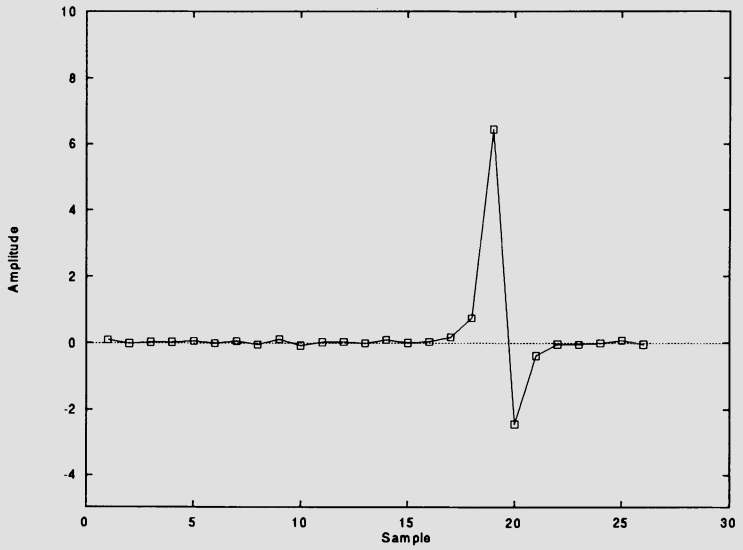


Figure 6.4 Equalizer impulse response, lattice RLS algorithm with error feedback.

6.11 CONCLUDING REMARKS

In this chapter, a number of alternative RLS algorithms based on the lattice realization were introduced. These algorithms consist of stages where growing-order forward and backward predictors of the input signal are built from stage to stage. This feature makes the lattice-based algorithms attractive in a number of applications where information about the statistics of the input signal, such as the order of the input-signal model, are useful. Another important feature

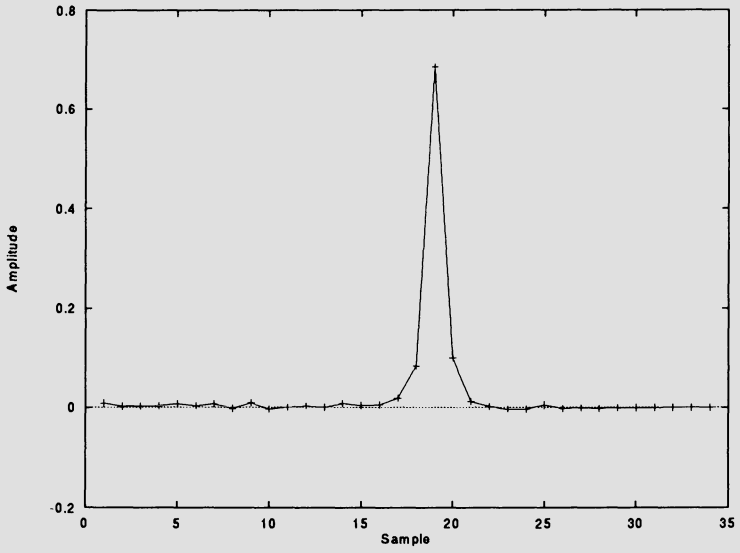


Figure 6.5 Convolution result, lattice RLS algorithm with error feedback.

of the lattice-based algorithms is their good performance when implemented in finite-precision arithmetic.

Also, their computational complexity of at least $16N$ multiplications per output sample is acceptable in a number of practical situations. However, by starting from the lattice formulation without making extensive use of order updating, it is possible to derive the fast transversal RLS algorithms, that can reduce the computational complexity to orders of $7N$ multiplications per output sample. The derivation of these algorithms is the subject of the following chapter.

Several interesting topics related to the lattice formulation of adaptive filters have been addressed in the open literature [9]-[13]. The geometric formulation of the least-squares estimation problem can be used to derive the lattice-based algorithms [9] in an elegant way. Also, an important situation that we usually find in practice is when the input data cannot be considered zero before the first iteration of the adaptive algorithm. The derivation of the lattice algorithms that account for nonzero initial condition for the input data is found in [10]. Another important problem is the characterization of the conditions under which the stability of the lattice algorithm is maintained when perturbations to the normal operation occur [11]. There is also a family of lattice-based algorithms that employ gradient type updating equations, these algorithms present reduced

computational complexity and good behavior when implemented with finite-precision arithmetic [12]-[13].

A number of simulation examples involving the lattice algorithms were presented in this chapter. These examples evaluated the performance of the lattice algorithm in different applications as well as in finite-precision implementations.

References

1. D. L. Lee, M. Morf, and B. Friedlander, "Recursive least squares ladder estimation algorithms," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-29, pp. 627-641, June 1981.

2. B. Friedlander, "Lattice filters for adaptive processing," *Proceedings of the IEEE*, vol. 70, pp. 829-867, Aug. 1982.

3. F. Ling, D. Manolakis, and J. G. Proakis, "Numerically robust least-squares lattice-ladder algorithms with direct updating of the reflection coefficients," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-34, pp. 837-845, Aug. 1986.

4. M. Bellanger, *Adaptive Digital Filters and Signal Processing*, Marcel Dekker Inc., New York, NY, 1987.

5. S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, NJ, 2nd edition 1991.

6. J. G. Proakis, C. M. Rader, F. Ling, and C. L. Nikias, *Advanced Digital Signal Processing*, MacMillan, New York, NY, 1992.

7. C. G. Samson, and V. U. Reddy, "Fixed point error analysis of the normalized ladder algorithm," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-31, pp. 1177-1191, Oct. 1983.

8. R. C. North, J. R. Zeidler, W. H. Ku, and T. R. Albert, "A floating-point arithmetic error analysis of direct and indirect coefficient updating techniques for adaptive lattice filters," *IEEE Trans. on Signal Processing*, vol. 41, pp. 1809-1823, May 1993.

9. H. Lev-Ari, T. Kailath, and J. Cioffi, "Least-squares adaptive lattice and transversal filters: A unified geometric theory," *IEEE Trans. on Information Theory*, vol. IT-30, pp. 222-236, March 1984.

10. J. Cioffi, "An unwindowed RLS adaptive lattice algorithm," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. 36, pp. 365-371, March 1988.

11. H. Lev-Ari, K.-F. Chiang, and T. Kailath, "Constrained-input/constrained-output stability for adaptive RLS lattice filters," *IEEE Trans. on Circuits and Systems*, vol. 38, pp. 1478-1483, Dec. 1991.

12. V. J. Mathews, and Z. Xie, "Fixed-point error analysis of stochastic gradient adaptive lattice filters," *IEEE Trans. on Signal Processing*, vol. 31, pp. 70-80, Jan. 1990.

13. M. Reed, and B. Liu, "Analysis of simplified gradient adaptive lattice algorithms using power-of-two quantization," *Proc. 1990 IEEE Intern. Symp. on Circuits and Systems*, New Orleans, LA, pp. 792-795, May 1990.

Problems

1. Deduce the time-updating formula for the backward predictor coefficients.

2. Given a square matrix $\mathbf{P}$ partitioned as follows:

$$\mathbf{P} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}$$

where $\mathbf{A}$ and $\mathbf{D}$ are also square matrices. The inverse of $\mathbf{P}$ can be expressed as follows:

$$\begin{aligned} \mathbf{P}^{-1} &= \begin{bmatrix} \mathbf{A}^{-1}[\mathbf{I}+\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}] & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \\ -(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & (\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \end{bmatrix} \\ &= \begin{bmatrix} (\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & -(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & \mathbf{D}^{-1}[\mathbf{I}+\mathbf{C}(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1}] \end{bmatrix} \end{aligned}$$

(a) Show the validity of this result.

(b) Use the appropriate partitioned forms of $\mathbf{R}_D(k-1,i)$ to derive the partitioned forms of $\mathbf{S}_D(k-1,i)$ of equations (6.56) and (6.59).

3. Derive the time-updating formula of $\delta_D(k,i)$.

4. Demonstrate that the backward *a posteriori* prediction errors $e_b(k,i)$ and $e_b(k,j)$ for $i \neq j$ are uncorrelated when the average is calculated over time.

5. Justify the initialization of $\xi^d_{b_{min}}(0)$ and $\xi^d_{f_{min}}(0)$ in the lattice RLS algorithm.

6. Derive the *a posteriori* lattice RLS algorithm for complex input signals.

7. Prove equation (6.71).

8. Derive the order-updating equation of the normalized forward and backward errors.

9. Prove the validity of the order-updating formula of the weighted least-squares error of the joint-process estimation described in equation (6.88).

10. Derive equation (6.89).

11. Derive the error-feedback LRLS algorithm based on *a priori* errors.

12. Derive the normalized LRLS algorithm based on *a priori* errors.

13. The lattice RLS algorithm based on *a posteriori* errors is used to predict the signal $x(k) = \sin \frac{\pi k}{4}$. Given $\lambda = 0.99$, calculate the error and the tap coefficients for the first 10 iterations.

14. The normalized lattice RLS algorithm based on *a posteriori* errors is used to predict the signal $x(k) = \sin \frac{\pi k}{4}$. Given $\lambda = 0.99$, calculate the error and the multiplier coefficients for the first 10 iterations.

15. The error-feedback LRLS algorithm was applied to identify a 7th-order time-varying unknown system whose coefficients are first-order Markov processes with $\lambda_{\mathbf{w}} = 0.999$ and $\sigma^2_{\mathbf{w}} = 0.033$. The initial time-varying system multiplier coefficients are

    $\mathbf{w}_o^T = [0.03490 - 0.01100 - 0.068640.223910.556860.35798 - 0.02390 - 0.07594]$

    The input signal is a Gaussian white noise with variance $\sigma_x^2 = 1$, and the measurement noise is also a Gaussian white noise independent of the input signal and of the elements of $\mathbf{n_w}(k)$, with variance $\sigma_n^2 = 0.01$.

    Simulate the experiment described above and measure the excess MSE, for $\lambda = 0.97$ and $\lambda = 0.99$.

16. Repeat the experiment described in problem 15, using the normalized lattice algorithm.

17. Suppose a 15th-order FIR digital filter with multiplier coefficients given below was identified through an adaptive FIR filter of the same order using the unnormalized LRLS algorithm. Considering that fixed-point arithmetic

was used, simulate the identification problem described using the following specifications.

Additional noise : white noise with variance	$\sigma_n^2 = 0.0015$
Coefficients wordlength:	$b_c = 16$ bits
Signal wordlength:	$b_d = 16$ bits
Input signal: Gaussian white noise with variance	$\sigma_x^2 = 0.7$
	$\lambda = 0.98$

$\mathbf{w}_o^T$ = [0.0219360 0.0015786 -0.0602449 -0.0118907 0.1375379 0.0574545 -0.3216703 -0.5287203 -0.2957797 0.0002043 0.290670 -0.0353349 -0.0068210 0.0026067 0.0010333 - 0.0143593]

Plot the learning curves for the finite- and infinite-precision implementations. Also plot $||\Delta\kappa_f(k,0)||^2$ and $||\Delta\kappa_b(k,0)||^2$ versus k, in both cases.

18. Repeat the problem above for the following cases

(a) $\sigma_n^2 = 0.01, b_c = 9$ bits, $b_d = 9$ bits, $\sigma_x^2 = 0.7, \lambda = 0.98$.

(b) $\sigma_n^2 = 0.1, b_c = 10$ bits, $b_d = 10$ bits, $\sigma_x^2 = 0.8, \lambda = 0.98$.

(c) $\sigma_n^2 = 0.05, b_c = 8$ bits, $b_d = 16$ bits, $\sigma_x^2 = 0.8, \lambda = 0.98$.

19. In the problem 17 above, rerun the simulations for $\lambda = 1, \lambda = 0.940$. Comment on the results.

20. Repeat the problem 18 above, using the normalized and error-feedback LRLS algorithms. Compare the results for the different algorithms.

21. Repeat problem 17 in the case the input signal is a first-order Markov process with $\lambda_{\mathbf{x}} = 0.98$.

22. Given a channel with impulse response given by

$$h(k) = 0.9^k + 0.4^k$$

for $k = 0, 1, 2, \ldots, 25$, design an adaptive equalizer. The input signal is white noise with unit variance and the adaptive filter input signal-to-noise ratio is of $-30dB$. Use the unnormalized lattice algorithm of order 35.

23. The unnormalized lattice algorithm is used to perform the forward prediction of a signal $x(k)$ generated by applying a white noise with unit variance to the input of a linear filter with transfer function given by

$$H(z) = \frac{0.5}{(1 - 1.512z^{-1} + 0.827z^{-2})(1 - 1.8z^{-1} + 0.87z^{-2})}$$

Calculate the zeros of the resulting predictor and compare with the poles of the linear filter.

24. Determine the computational complexity of the Algorithms 6.1, 6.2, 6.3 and 6.4.

7

FAST TRANSVERSAL RLS ALGORITHMS

7.1 INTRODUCTION

Among the large number of algorithms that solve the least-squares problem in a recursive form, the fast transversal recursive least-squares (FTRLS) algorithms are very attractive due to their reduced computational complexity [1]-[7].

The FTRLS algorithms can be derived by solving simultaneously the forward and backward linear prediction problems, along with two other transversal filters consisting of the joint-process estimator and an auxiliary filter whose desired signal vector has one as its first (i.e., $d(0)$) and unique nonzero element. Unlike the lattice-based algorithms, the FTRLS algorithms require only time-recursive equations. However, a number of relations required to derive some of the FTRLS algorithms can be taken from the previous chapter on LRLS algorithms. The FTRLS algorithm can also be considered as a fast version of the transversal adaptive filter for the solution of the RLS problem, since a fixed-order update for the transversal adaptive filter coefficient vector is computed in each iteration.

The relations derived for the backward and forward prediction in the lattice-based algorithms can be used to derive the FTRLS algorithms. The resulting algorithms have computational complexity of order N, making them especially attractive for practical implementation. As compared to the lattice-based algorithms, the computational complexity of the FTRLS algorithms is lower due to the absence of order-updating equations. Particularly, FTRLS algorithms typically require an order of $7N$ to $11N$ multiplications and divisions per output sample, as compared to the $14N$ to $29N$ for the LRLS algorithms. Therefore, FTRLS algorithms are considered as the fastest implementation solutions of the RLS problem [1]-[7].

P. S. R. Diniz, *Adaptive Filtering*

Several alternative FTRLS algorithms were proposed in the literature. The so-called fast Kalman algorithm [1], that is certainly one of the earlier fast transversal RLS algorithms, has computational complexity of $11N$ multiplications and divisions per output sample. In a later stage of research development in the area of fast transversal algorithms, the fast *a posteriori* error sequential technique (FAEST) [2], and the fast transversal filter (FTF) [3] algorithms were proposed, both requiring an order of $7N$ multiplications and divisions per output sample. The FAEST and FTF algorithms have the lowest complexity known for RLS algorithms, for problems where the input vector elements consist of delayed versions of a single input signal. Unfortunately, these algorithms are very sensitive to quantization effects and become unstable if certain actions are not taken [5]-[7], [9].

In this Chapter, a particular form of the FTRLS algorithm is presented, where most of the derivations are based on those presented for the lattice algorithms. It is well known that the quantization errors in the FTRLS algorithms present exponential divergence [1]-[7]. Since the unstable behavior of the FTRLS algorithms when implemented with finite-precision arithmetic is undesirable, we discuss the implementation of numerically stable FTRLS algorithms, providing the description of a particular algorithm [8]-[10].

7.2 RECURSIVE LEAST-SQUARES PREDICTION

All fast algorithms explore some structural property of the information data in order to achieve low computational complexity. In the particular case of the fast RLS algorithms discussed in this text, the reduction in the computational complexity is achieved for the cases where the input signal consists of consecutively delayed samples of the same signal. In this case, the pattern of the fast algorithms are similar in the sense that the forward and backward prediction filters are essential parts of these algorithms. The predictors perform the task of modeling the input signal, which as a result allows the replacement of matrix equations by vector and scalar relations.

In the derivation of the FTRLS algorithms, the solutions of the RLS forward and backward prediction problems are required in the time-update equations. In this section, these solutions are reviewed emphasizing the results that are relevant to the FTRLS algorithms. As previously mentioned, we will borrow a number of derivations from the previous chapter on lattice algorithms. It is

worth mentioning that the FTRLS could be introduced through an independent derivation, however the derivation based on the lattice is probably more insightful, and certainly more straightforward at this point.

7.2.1 Forward Prediction Relations

The instantaneous *a posteriori* forward prediction error for an Nth-order predictor is given by

$$\begin{aligned} e_f(k,N) &= x(k) - \mathbf{w}_f^T(k,N)\mathbf{x}(k-1,N) \\ &= \mathbf{x}^T(k,N+1)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k,N) \end{bmatrix} \end{aligned} \tag{7.1}$$

The relationship between *a posteriori* and *a priori* forward prediction error, first presented in equation (6.49) and repeated here for convenience, is given by

$$e'_f(k,N) = \frac{e_f(k,N)}{\gamma(k-1,N)} \tag{7.2}$$

A simple manipulation of equation (6.73), leads to the following relation for the time updating of the minimum weighted least-squares error, which will be used in the FTRLS algorithm

$$\xi^d_{f_{min}}(k,N) = \lambda\xi^d_{f_{min}}(k-1,N) + e'_f(k,N)e_f(k,N) \tag{7.3}$$

From the same equation (6.73), we can obtain the following equality that will also be required in the FTRLS algorithm

$$\gamma(k,N+1) = \frac{\lambda\xi^d_{f_{min}}(k-1,N)}{\xi^d_{f_{min}}(k,N)}\gamma(k-1,N) \tag{7.4}$$

The updating equation of the forward prediction tap-coefficient vector can be performed through equation (6.40) of the previous chapter, i.e.,

$$\mathbf{w}_f(k,N) = \mathbf{w}_f(k-1,N) + \boldsymbol{\phi}(k-1,N)e'_f(k,N) \tag{7.5}$$

where $\boldsymbol{\phi}(k-1,N) = \mathbf{S}_D(k-1,N)\mathbf{x}(k-1,N)$.

As will be seen, the updating of vector $\boldsymbol{\phi}(k-1,N)$ to $\boldsymbol{\phi}(k,N+1)$ is needed to update the backward predictor coefficient vector. Also, the last element of $\boldsymbol{\phi}(k,N+1)$ is used to update the backward prediction *a priori* error and to obtain $\gamma(k,N)$. Vector $\boldsymbol{\phi}(k,N+1)$ can be obtained by post-multiplying equation (6.56), at instant k and for order N, by $\mathbf{x}(k,N+1) = [x(k)\ \mathbf{x}^T(k-1,N)]^T$. The result can be expressed as

$$\boldsymbol{\phi}(k,N+1) = \begin{bmatrix} 0 \\ \boldsymbol{\phi}(k-1,N) \end{bmatrix} + \frac{1}{\xi^d_{f_{min}}(k,N)} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k,N) \end{bmatrix} e_f(k,N) \quad (7.6)$$

However, it is not convenient to use the equation above in the FTRLS algorithm because when deriving the backward prediction part, it would lead to extra computation. The solution is to use an alternative recursion involving $\hat{\boldsymbol{\phi}}(k,N+1) = \frac{\boldsymbol{\phi}(k,N+1)}{\gamma(k,N+1)}$ instead of $\boldsymbol{\phi}(k,N+1)$, see problem 7 for further details. The resulting recursion can be derived after some algebraic manipulations of the equation (7.6) above and equations (7.3) to (7.5).

$$\begin{aligned} \hat{\boldsymbol{\phi}}(k,N+1) &= \begin{bmatrix} 0 \\ \hat{\boldsymbol{\phi}}(k-1,N) \end{bmatrix} \\ &\quad + \frac{1}{\lambda\xi^d_{f_{min}}(k-1,N)} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,N) \end{bmatrix} e'_f(k,N) \end{aligned} \quad (7.7)$$

The forward prediction tap-coefficient vector should then be updated using $\hat{\boldsymbol{\phi}}(k-1,N)$ as follows:

$$\mathbf{w}_f(k,N) = \mathbf{w}_f(k-1,N) + \hat{\boldsymbol{\phi}}(k-1,N)e_f(k,N) \quad (7.8)$$

7.2.2 Backward Prediction Relations

In this subsection, the relations involving the backward prediction problem that are used in the FTRLS algorithm are derived.

The relationship between *a posteriori* and *a priori* backward prediction errors can be expressed as

$$e_b(k,N) = e'_b(k,N)\gamma(k,N) \quad (7.9)$$

It is also known, see equation (6.79) of the previous chapter, that the ratio of conversion factors for different orders is given by

$$\frac{\gamma(k,N+1)}{\gamma(k,N)} = \frac{\lambda\xi^d_{b_{min}}(k-1,N)}{\xi^d_{b_{min}}(k,N)} \tag{7.10}$$

We rewrite for convenience the last equality of equation (6.70)

$$\xi^d_{b_{min}}(k,N) = \lambda\xi^d_{b_{min}}(k-1,N) + \frac{e^2_b(k,N)}{\gamma(k,N)} \tag{7.11}$$

The equation above can be rewritten as follows:

$$1 + \frac{e^2_b(k,N)}{\lambda\gamma(k,N)\xi^d_{b_{min}}(k-1,N)} = \frac{\xi^d_{b_{min}}(k,N)}{\lambda\xi^d_{b_{min}}(k-1,N)} \tag{7.12}$$

Now we should recall that the time updating for the backward predictor filter is given by

$$\begin{aligned} \mathbf{w}_b(k,N) &= \mathbf{w}_b(k-1,N) + \boldsymbol{\phi}(k,N)e'_b(k,N) \\ &= \mathbf{w}_b(k-1,N) + \hat{\boldsymbol{\phi}}(k,N)e_b(k,N) \end{aligned} \tag{7.13}$$

Following a similar path used to derive equation (7.6), by first post-multiplying equation (6.59), at instant k and for order N, by $\mathbf{x}(k,N+1) = [\mathbf{x}^T(k,N)\, x(k-N)]^T$, and using relations (7.10), (7.11), and (7.13), we have

$$\begin{aligned} &\begin{bmatrix} \hat{\boldsymbol{\phi}}(k,N) \\ 0 \end{bmatrix} = \hat{\boldsymbol{\phi}}(k,N+1) \\ &-\frac{1}{\lambda\xi^d_{b_{min}}(k-1,N)} \begin{bmatrix} -\mathbf{w}_b(k-1,N) \\ 1 \end{bmatrix} e'_b(k,N) \end{aligned} \tag{7.14}$$

Note that in this equation the last element of $\hat{\boldsymbol{\phi}}(k,N+1)$ was already calculated in equation (7.6). In any case, it is worth mentioning that the last element of $\hat{\boldsymbol{\phi}}(k,N+1)$ can alternatively be expressed as

$$\hat{\phi}_{N+1}(k,N+1) = \frac{e'_b(k,N)}{\lambda\xi^d_{b_{min}}(k-1,N)} \tag{7.15}$$

By applying equations (7.9), (7.15), and (7.10) in equation (7.12), we can show that

$$1 + \hat{\phi}_{N+1}(k, N+1)e_b(k, N) = \frac{\gamma(k, N)}{\gamma(k, N+1)} \tag{7.16}$$

We are now in a position to derive an updating equation which is used in the FTRLS algorithm, by substituting equation (7.9) into the equation above. The resulting relation is

$$\gamma^{-1}(k, N) = \gamma^{-1}(k, N+1) - \hat{\phi}_{N+1}(k, N+1)e'_b(k, N) \tag{7.17}$$

The updating equations related to the forward and backward prediction problems and for the conversion factor $\gamma(k, N)$ are now available. We now proceed with the derivations to solve the more general problem of estimating a related process represented by the desired signal $d(k)$, known as joint-process estimation.

7.3 JOINT-PROCESS ESTIMATION

As for all previously presented adaptive filters algorithms, it is useful to derive a FTRLS algorithm that can match a desired signal $d(k)$ through the minimization of the weighted squared error. Starting with the *a priori* error

$$e'(k, N) = d(k) - \mathbf{w}^T(k-1, N)\mathbf{x}(k, N) \tag{7.18}$$

we can calculate the *a posteriori* error as follows:

$$e(k, N) = e'(k, N)\gamma(k, N) \tag{7.19}$$

Like in the conventional RLS algorithm, the time updating for the output tap coefficients of the joint-process estimator can be performed by

$$\begin{aligned} \mathbf{w}(k, N) &= \mathbf{w}(k-1, N) + \boldsymbol{\phi}(k, N)e'(k, N) \\ &= \mathbf{w}(k-1, N) + \hat{\boldsymbol{\phi}}(k, N)e(k, N) \end{aligned} \tag{7.20}$$

All the updating equations are now available to describe the fast transversal RLS algorithm. The FRLS algorithm consists of equations (7.1)-(7.3), (7.6)-(7.8),

and (7.4) related to the forward predictor; equations (7.15), (7.17), (7.9), (7.11), (7.14), and (7.13) related to the backward predictor and the conversion factor; and (7.18)-(7.20) related to the joint-process estimator. The FTRLS algorithm is described in Algorithm 7.1. The computational complexity of the FTRLS algorithm is $7(N) + 14$ multiplications per output sample. The key feature of the FTRLS algorithm is that it does not require matrix multiplications, that is why the implementation of the FTRLS algorithm has complexity of order N multiplications per output sample.

The initialization procedure consists of setting the tap coefficients of the backward prediction, forward prediction, and joint-process estimation filters to zero, namely

$$\mathbf{w}_f(-1, N) = \mathbf{w}_b(-1, N) = \mathbf{w}(-1, N) = \mathbf{0} \tag{7.21}$$

Vector $\hat{\boldsymbol{\phi}}(-1, N)$ is set to $\mathbf{0}$ assuming the input and desired signals are zero for $k < 0$, i.e., prewindowed data. The conversion factor should be initialized as follows:

$$\gamma(-1, N) = 1 \tag{7.22}$$

since no difference between *a priori* and *a posteriori* errors exists during the initialization period. The weighted least-square errors should be initialized with a positive constant ϵ

$$\xi^d_{f_{min}}(-1, N) = \xi^d_{b_{min}}(-1, N) = \epsilon \tag{7.23}$$

in order to avoid division by zero in the first iteration. The reason to introduce this initialization parameter suggests that it should be a small value. However for stability reasons the value of ϵ should not be small (see the examples at the end of this chapter).

It should be mentioned that there are exact initialization procedures for the fast transversal RLS filters, aiming the minimization of the objective function at all instants during the initialization period [3]. These procedures explore the fact that during the initialization period the number of data samples in both $d(k)$ and $x(k)$ is less than $N + 1$, therefore the objective function can be made zero since there are more parameters than needed. The exact initialization procedure of [3] replaces the computationally intensive backsubstitution algorithm and is rather simple when the adaptive filter coefficients are initialized with zero. The procedure can also be generalized to the cases where some nonzero initial values for the tap coefficients are available.

Algorithm 7.1

Fast Transversal RLS Algorithm

Initialization

$\mathbf{w}_f(-1,N) = \mathbf{w}_b(-1,N) = \mathbf{w}(-1,N) = \mathbf{0}$
$\hat{\boldsymbol{\phi}}(-1,N) = \mathbf{0},\ \gamma(-1,N) = 1$
$\xi^d_{b_{min}}(-1,N) = \xi^d_{f_{min}}(-1,N) = \epsilon$ (a small positive constant)

Prediction Part

Do for each $k \geq 0$,

$$e'_f(k,N) = \mathbf{x}^T(k,N+1)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,N) \end{bmatrix}$$
$$e_f(k,N) = e'_f(k,N)\gamma(k-1,N) \tag{7.2}$$
$$\xi^d_{f_{min}}(k,N) = \lambda\xi^d_{f_{min}}(k-1,N) + e'_f(k,N)e_f(k,N) \tag{7.3}$$
$$\mathbf{w}_f(k,N) = \mathbf{w}_f(k-1,N) + \hat{\boldsymbol{\phi}}(k-1,N)e_f(k,N) \tag{7.8}$$
$$\hat{\boldsymbol{\phi}}(k,N+1) = \begin{bmatrix} 0 \\ \hat{\boldsymbol{\phi}}(k-1,N) \end{bmatrix} + \frac{1}{\lambda\xi^d_{f_{min}}(k-1,N)}\begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,N) \end{bmatrix} e'_f(k,N) \tag{7.6}$$
$$\gamma(k,N+1) = \frac{\lambda\xi^d_{f_{min}}(k-1,N)}{\xi^d_{f_{min}}(k,N)}\gamma(k-1,N) \tag{7.4}$$
$$e'_b(k,N) = \lambda\xi^d_{b_{min}}(k-1,N)\hat{\boldsymbol{\phi}}_{N+1}(k,N+1) \tag{7.15}$$
$$\gamma^{-1}(k,N) = \gamma^{-1}(k,N+1) - \hat{\boldsymbol{\phi}}_{N+1}(k,N+1)e'_b(k,N) \tag{7.17}$$
$$e_b(k,N) = e'_b(k,N)\gamma(k,N) \tag{7.9}$$
$$\xi^d_{b_{min}}(k,N) = \lambda\xi^d_{b_{min}}(k-1,N) + e_b(k,N)e'_b(k,N) \tag{7.11}$$
$$\begin{bmatrix} \hat{\boldsymbol{\phi}}(k,N) \\ 0 \end{bmatrix} = \hat{\boldsymbol{\phi}}(k,N+1) - \hat{\boldsymbol{\phi}}_{N+1}(k,N+1)\begin{bmatrix} -\mathbf{w}_b(k-1,N) \\ 1 \end{bmatrix} \tag{7.14}$$
$$\mathbf{w}_b(k,N) = \mathbf{w}_b(k-1,N) + \hat{\boldsymbol{\phi}}(k,N)e_b(k,N) \tag{7.13}$$

Joint-Process Estimation

$$e'(k,N) = d(k) - \mathbf{w}^T(k-1,N)\mathbf{x}(k,N) \tag{7.18}$$
$$e(k,N) = e'(k,N)\gamma(k,N) \tag{7.19}$$
$$\mathbf{w}(k,N) = \mathbf{w}(k-1,N) + \hat{\boldsymbol{\phi}}(k,N)e(k,N) \tag{7.20}$$

End

□

As previously mentioned several fast RLS algorithms based on the transversal realization exist; the one presented here corresponds to the so-called FTF proposed in [3]. A number of alternative algorithms are introduced in the problems.

7.4 STABILIZED FAST TRANSVERSAL RLS ALGORITHM

Although the fast transversal algorithms proposed in the literature provide a nice solution to the computational complexity burden inherent to the conventional RLS algorithm, these algorithms are unstable when implemented with finite-precision arithmetic. Increasing the wordlength does not solve the instability problem. The only effect of employing a longer wordlength is that the algorithm will take longer to diverge. Earlier solutions to this problem consisted of restarting the algorithm when the accumulated errors in chosen variables reached prescribed thresholds [3]. Although the restart procedure would consider past information, the resulting performance is suboptimal due to the discontinuity of information in the corresponding deterministic correlation matrix.

The explanation for the unstable behavior of the fast transversal algorithms is the existence of an inherent positive feedback mechanism. This explanation led to the idea that if some specific measurements of the numerical errors were available, they could conveniently be fed back in order to make the negative feedback dominant in the error propagation dynamics. Fortunately, some measurements of the numerical errors can be obtained by introducing computational redundancy in the fast algorithm. The computational redundancy consists of calculating a given quantity using two different formulas. In finite-precision implementation, the resulting values for the quantity calculated by these formulas are not equal, and their difference is a good measurement of the accumulated errors in that quantity. This error can then be fed back in attempt to stabilize the algorithm. The key problem is to find out what are the quantities where the computational redundancy should be introduced such that the error propagation dynamics can be stabilized. In the early proposed solutions [6]-[7] only a single quantity was chosen to introduce the redundancy, later it was shown that at least two quantities are required in order to guarantee the stability of the FTRLS algorithm [9]. Another relevant question is to where should the error be fed back inside the algorithm. Note that any point could be chosen without affecting the behavior of the algorithm when implemented with infinite precision, since the feedback error is zero. A natural choice is to feed the error back

into the expressions of the quantities that are related to it. That means for each quantity that the redundancy is introduced, its final value is a combination of the two forms of computing it.

The FTRLS algorithm can be seen as a discrete-time nonlinear dynamic system [9], and when finite precision is used in the implementation, quantization errors will rise. In this case, the internal quantities will be perturbed when compared with the infinite-precision quantities. A nonlinear system modeling the error propagation can then be described, which if properly linearized allows the study of the error propagation mechanism. Using an averaging analysis which is meaningful for stationary input signals, it is possible to obtain a system characterized by its set of eigenvalues, whose dynamic behavior is similar to the error propagation behavior when $k \to \infty$ and $(1 - \lambda) \to 0$. Through these eigenvalues it is possible to determine the feedback parameters as well as the quantities to choose for the introduction of redundancy. The objective here is to modify the unstable modes through the error feedback in order to turn them stable [9]. Fortunately, it was found in [9] that the unstable modes can be modified and stabilized by the introduced error feedback. The unstable modes can be modified by introducing redundancy in $\gamma(k, N)$ and $e'_b(k, N)$. These quantities can be calculated using different relations, and to distinguish them an extra index is included in their description.

The *a priori* backward error can be described in a number of alternative forms such as:

$$e'_b(k, N, 1) = \lambda \xi^d_{b_{min}}(k-1, N)\hat{\phi}_{N+1}(k, N+1) \tag{7.24}$$

$$e'_b(k, N, 2) = \left[-\mathbf{w}_b^T(k-1, N)\ 1\right]\mathbf{x}(k, N+1) \tag{7.25}$$

and

$$\begin{aligned} e'_{b,i}(k, N, 3) &= e'_b(k, N, 2)\kappa_i + e'_b(k, N, 1)[1 - \kappa_i] \\ &= e'_b(k, N, 1) + \kappa_i[e'_b(k, N, 2) - e'_b(k, N, 1)] \end{aligned} \tag{7.26}$$

where the first form was employed in the FTRLS algorithm, and the second form corresponds to the inner product implementation of the *a priori* backward error. The third form corresponds to a linear combination of the first two forms, where the numerical difference between these forms is fed back to determine the final value of $e'_{b,i}(k, N, 3)$ which will be used in the stabilized algorithm at different places. For each κ_i, $i = 1, 2, 3$, we have a different value that is chosen in order to guarantee that the related eigenvalues are less than one.

The conversion factor $\gamma(k, N)$ is probably the first parameter to show signs that the algorithm is becoming unstable. This parameter can also be calculated

through different relations. These alternative relations are required to guarantee that all modes of the error propagation system become stable. The first equation is given by

$$\begin{aligned}
\gamma^{-1}(k, N+1, 1) &= \gamma^{-1}(k-1, N, 3)\frac{\xi^d_{f_{min}}(k, N)}{\lambda\xi^d_{f_{min}}(k-1, N)} \\
&= \gamma^{-1}(k-1, N, 3)\left[1 + \frac{e'_f(k, N)e_f(k, N)}{\lambda\xi^d_{f_{min}}(k-1, N)}\right] \\
&= \gamma^{-1}(k-1, N, 3) + \frac{e'^2_f(k, N)}{\lambda\xi^d_{f_{min}}(k-1, N)} \\
&= \gamma^{-1}(k-1, N, 3) + \hat{\phi}_0(k, N+1)e'_f(k, N) \qquad (7.27)
\end{aligned}$$

where $\hat{\phi}_0(k, N+1)$ is the first element of $\hat{\phi}(k, N+1)$. The equalities above are derived from equations (7.4), (7.3), (7.2) and (7.6), respectively. The second expression for the conversion factor is derived from equation (7.14), being given by

$$\gamma^{-1}(k, N, 2) = \gamma^{-1}(k, N+1, 1) - \hat{\phi}_{N+1}(k, N+1)e'_{b,3}(k, N, 3) \qquad (7.28)$$

The third expression is

$$\gamma^{-1}(k, N, 3) = 1 + \hat{\phi}^T(k, N)\mathbf{x}(k, N) \qquad (7.29)$$

In equation (7.27), the conversion factor was expressed in different ways, one of them first presented in the FTRLS algorithm of [9]. The second form already uses an *a priori* backward error with redundancy. The third form can be derived from equation (6.48) for the lattice RLS algorithms, see problem 10.

An alternative relation utilized in the stabilized fast transversal algorithm involves the minimum forward least-squares error. From equations (7.3) and (7.6), we can write

$$\begin{aligned}
[\xi^d_{f_{min}}(k, N)]^{-1} &= \lambda^{-1}[\xi^d_{f_{min}}(k-1, N)]^{-1} - \frac{e'_f(k, N)e_f(k, N)}{\lambda\xi^d_{f_{min}}(k-1, N)\xi^d_{f_{min}}(k, N)} \\
&= \lambda^{-1}[\xi^d_{f_{min}}(k-1, N)]^{-1} - \frac{\hat{\phi}_0(k, N)e_f(k, N)}{\xi^d_{f_{min}}(k, N)}
\end{aligned}$$

From (7.6), we can deduce that

$$\frac{e_f(k, N)}{\xi^d_{f_{min}}(k, N)} = \phi_0(k, N) = \hat{\phi}_0(k, N)\gamma(k, N+1, 1)$$

With this relation we can obtain the desired equation

$$[\xi^d_{f_{min}}(k,N)]^{-1} = \lambda^{-1}[\xi^d_{f_{min}}(k-1,N)]^{-1} - \gamma(k,N+1,1)\hat{\phi}_0^2(k,N+1) \tag{7.30}$$

where the choice of $\gamma(k,N+1,1)$ for the equation above results from the scheme to keep the error-system modes stable [9].

Using the equations for the conversion factor and for the *a priori* backward error with redundancy, we can obtain the stabilized fast transversal RLS algorithm (SFTRLS) whose equations are shown on Algorithm 7.2. The parameters κ_i for $i = 1,2,3$ were determined through computer simulation search [9], where the optimal values found were $\kappa_1 = 1.5$, $\kappa_2 = 2.5$, and $\kappa_3 = 1$. It was also found in [9], that the numerical behavior is quite insensitive to values of κ_i around the optimal, and that optimal values chosen for a given situation work well for a wide range of environments and algorithm setup situations (for example, for different choices of the forgetting factor).

Another issue related to the SFTRLS algorithm concerns the range of values for λ such that stability is guaranteed. Results of extensive simulation experiments [9] indicate that the range is

$$1 - \frac{1}{2(N+1)} \leq \lambda < 1 \tag{7.31}$$

where N is the order of the adaptive filter. It was also verified that the optimal numerical behavior is achieved when the value of λ is chosen as

$$\lambda = 1 - \frac{0.4}{N+1} \tag{7.32}$$

The range of values for λ as well as its optimal value can be very close to one for high-order filters. This can be a potential limitation for the use of the SFTRLS algorithm, especially in nonstationary environments where smaller values for λ are required.

The computational complexity of the SFTRLS algorithm is of order $9N$ multiplications per output sample. There is an alternative algorithm with computational complexity of order $8N$ (see problem 9).

Before leaving this section, it is worth mentioning a nice interpretation for the fast transversal RLS algorithm. The FTRLS algorithm can be viewed as four transversal filters working in parallel and exchanging quantities with each

Algorithm 7.2

Stabilized Fast Transversal RLS Algorithm

Initialization

$\mathbf{w}_f(-1,N) = \mathbf{w}_b(-1,N) = \mathbf{w}(-1,N) = \mathbf{0}$
$\hat{\boldsymbol{\phi}}(-1,N) = \mathbf{0},\ \gamma(-1,N,3) = 1$
$\xi^d_{b_{min}}(-1,N) = \xi^d_{f_{min}}(-1,N) = \epsilon$ (a small positive constant)
$\kappa_1 = 1.5, \kappa_2 = 2.5, \kappa_3 = 1$

Prediction Part

Do for each $k \geq 0$,

$$e'_f(k,N) = \mathbf{x}^T(k,N+1)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,N) \end{bmatrix}$$
$$e_f(k,N) = e'_f(k,N)\gamma(k-1,N,3) \tag{7.2}$$
$$\hat{\boldsymbol{\phi}}(k,N+1) = \begin{bmatrix} 0 \\ \hat{\boldsymbol{\phi}}(k-1,N) \end{bmatrix} + \frac{1}{\lambda\xi^d_{f_{min}}(k-1,N)}\begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,N) \end{bmatrix} e'_f(k,N) \tag{7.6}$$
$$\gamma^{-1}(k,N+1,1) = \gamma^{-1}(k-1,N,3) + \hat{\phi}_0(k,N+1)e'_f(k,N) \tag{7.27}$$
$$[\xi^d_{f_{min}}(k,N)]^{-1} = \lambda^{-1}[\xi^d_{f_{min}}(k-1,N)]^{-1} - \gamma(k,N+1,1)\hat{\phi}^2_0(k,N+1) \tag{7.30}$$
$$\mathbf{w}_f(k,N) = \mathbf{w}_f(k-1,N) + \hat{\boldsymbol{\phi}}(k-1,N)e_f(k,N) \tag{7.8}$$
$$e'_b(k,N,1) = \lambda\xi^d_{b_{min}}(k-1,N)\hat{\phi}_{N+1}(k,N+1) \tag{7.15}$$
$$e'_b(k,N,2) = \begin{bmatrix} -\mathbf{w}^T_b(k-1,N) & 1 \end{bmatrix}\mathbf{x}(k,N+1) \tag{7.25}$$
$$e'_{b,i}(k,N,3) = e'_b(k,N,2)\kappa_i + e'_b(k,N,1)[1-\kappa_i] \ \ for\ i = 1,2,3 \tag{7.25}$$
$$\gamma^{-1}(k,N,2) = \gamma^{-1}(k,N+1,1) - \hat{\phi}_{N+1}(k,N+1)e'_{b,3}(k,N,3) \tag{7.28}$$
$$e_{b,j}(k,N,3) = e'_{b,j}(k,N,3)\gamma(k,N,2)\ \ j = 1,2$$
$$\xi^d_{b_{min}}(k,N) = \lambda\xi^d_{b_{min}}(k-1,N) + e_{b,2}(k,N,3)e'_{b,2}(k,N,3) \tag{7.11}$$
$$\begin{bmatrix} \hat{\boldsymbol{\phi}}(k,N) \\ 0 \end{bmatrix} = \hat{\boldsymbol{\phi}}(k,N+1) - \hat{\boldsymbol{\phi}}_{N+1}(k,N+1)\begin{bmatrix} -\mathbf{w}_b(k-1,N) \\ 1 \end{bmatrix} \tag{7.14}$$
$$\mathbf{w}_b(k,N) = \mathbf{w}_b(k-1,N) + \hat{\boldsymbol{\phi}}(k,N)e_{b,1}(k,N,3) \tag{7.13}$$
$$\gamma^{-1}(k,N,3) = 1 + \hat{\boldsymbol{\phi}}^T(k,N)\mathbf{x}(k,N) \tag{7.29}$$

Joint-Process Estimation

$$e'(k,N) = d(k) - \mathbf{w}^T(k-1,N)\mathbf{x}(k,N) \tag{7.18}$$
$$e(k,N) = e'(k,N)\gamma(k,N,3) \tag{7.19}$$
$$\mathbf{w}(k,N) = \mathbf{w}(k-1,N) + \hat{\boldsymbol{\phi}}(k,N)e(k,N) \tag{7.20}$$

End

□

other, as depicted in Fig. 7.1. The first filter is the forward prediction filter that utilizes $\mathbf{x}(k-1,N)$ as input signal vector, with $\mathbf{w}_f(k,N)$ as the coefficient vector, and provides quantities $e_f(k,N), e'_f(k,N)$, and $\xi^d_{f_{min}}(k,N)$ as outputs. The second filter is the backward prediction filter that utilizes $\mathbf{x}(k,N)$ as input signal vector, with $\mathbf{w}_b(k,N)$ as the coefficient vector, and provides quantities $e_b(k,N), e'_b(k,N)$, and $\xi^d_{b_{min}}(k,N)$ as outputs. The third filter is an auxiliary filter whose coefficients are given by $-\hat{\boldsymbol{\phi}}(k,N)$, the input signal vector is $\mathbf{x}(k,N)$, and the output parameter is $\gamma^{-1}(k,N)$. For this filter the desired signal vector is constant and equal to $[1\ 0\ 0\ldots 0]^T$. The fourth and last filter is the joint-process estimator whose input signal vector is $\mathbf{x}(k,N)$, and the coefficient vector is $\mathbf{w}(k,N)$, providing the quantities $e(k,N)$, and $e'(k,N)$ as outputs.

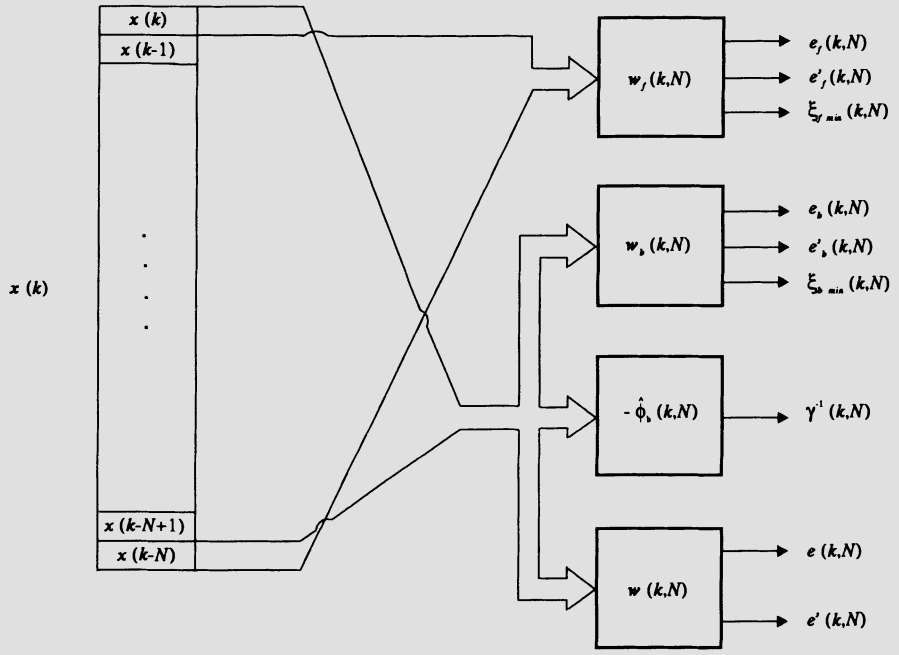


Figure 7.1 Fast transversal RLS algorithm: block diagram.

Example 7.1

The system identification problem described in subsection 3.6.2 is solved using the stabilized fast transversal algorithm presented in this chapter. The main

objective is to check the stability of the algorithm when implemented in finite precision.

Solution:

According to equation (7.31) the lower bound for λ in this case is 0.9375. A value $\lambda = 0.99$ was chosen. The stabilized fast transversal algorithm was applied to solve the identification problem and the measured MSE was 0.0432.

Using $\epsilon = 2$, we ran the algorithm with finite precision and the results are summarized in Table 7.1. No sign of instability was found for $\lambda = 0.99$. These are results generated by ensemble averaging 200 experiments. A comparison of the results of Table 7.1 with those of Tables 5.2 and 6.2 shows that the SFTRLS algorithm has similar performance compared to the conventional and lattice-based RLS algorithms, in terms of quantization error accumulation. The question is which algorithm remains stable in most situations. Regarding the SFTRLS, for large-order filters we are left with a limited range of values to choose λ. Also, it was found in our experiments that the choice of the initialization parameter ϵ plays an important role in the performance of this algorithm when implemented in finite precision. In some cases, even when the value of λ is within the recommended range the algorithm does not converge if ϵ is small. By increasing the value of ϵ we increase the usual convergence time while keeping the algorithm stable.

□

Table 7.1 Results of the Finite-Precision Implementation of the SFTRLS Algorithm

No of bits	$\xi(k)_Q$ Experiment	$E[\|\|\Delta\mathbf{w}(k)_Q\|\|^2]$ Experiment
16	$1.545\ 10^{-3}$	$6.089\ 10^{-5}$
12	$1.521\ 10^{-3}$	$3.163\ 10^{-5}$
10	$1.562\ 10^{-3}$	$6.582\ 10^{-5}$

Example 7.2

The channel equalization example described in subsection (3.6.3) was also used in simulations to test the SFTRLS algorithm. We used a 25th-order equalizer and a forgetting factor $\lambda = 0.99$.

Solution:

In order to solve the equalization problem the stabilized fast transversal RLS algorithm was initialized with $\epsilon = 0.5$. The results presented here were generated by ensemble averaging 200 experiments. The resulting learning curve of the MSE is shown in Fig. 7.2, and the measured MSE was 0.2973. The overall performance of the SFTRLS algorithm for this particular example is as good as any other RLS algorithm, such as lattice-based algorithms.

□

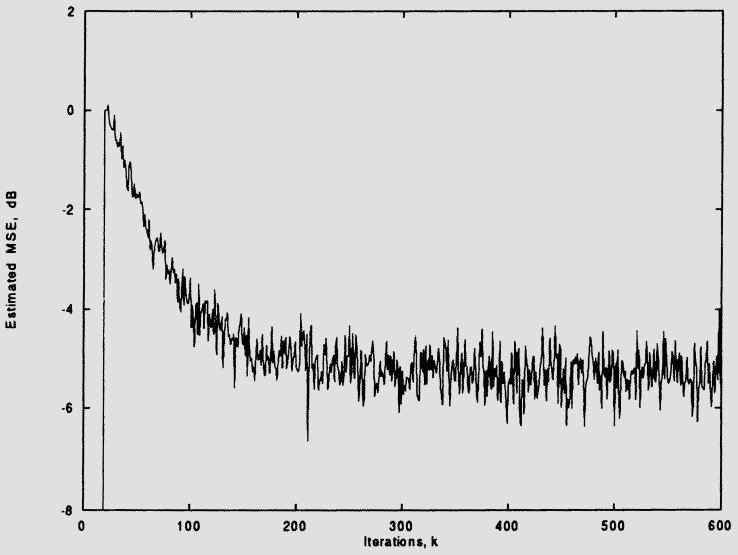


Figure 7.2 Learning curves for the stabilized fast transversal RLS algorithm.

7.5 CONCLUDING REMARKS

In this chapter we have presented some fast transversal RLS algorithms. This class of algorithms is computationally more efficient than conventional and lattice-based RLS algorithms. A number of alternative FTRLS algorithms as well as theoretical results can be found in [3]. The derivation of normalized versions of the FTRLS is also possible and was not addressed in the present chapter, for this result refer to [4]. The most computationally efficient FTRLS algorithms are known to be unstable. The error-feedback approach was briefly introduced that allows the stabilization of the FTRLS algorithm. The complete derivation and justification for the error-feedback approach is given in [9].

In nonstationary environments, it might be useful to employ a time-varying forgetting factor, therefore it is desirable to obtain FTRLS algorithms allowing the use of variable λ. This problem was first addressed in [11]. However a computationally more efficient solution was proposed in [8], where the concept of data weighting was introduced to replace the concept of error weighting.

The FTRLS algorithm has potential for a number of applications. In particular, the problem where the signals available from the environment are a noisy version of a transmitted signal and a noisy and filtered version of the same transmitted signal is an interesting application. In this problem, both the delay and unknown filter coefficients have to be estimated. The weighted squared errors have to be minimized considering both the delay and the unknown system parameters. This problem of joint estimation can be elegantly solved by employing the FTRLS algorithm [12].

Some simulation examples were included where the SFTRLS was employed. The finite-wordlength simulations are of special interest for the reader.

References

1. D. D. Falconer and L. Ljung, "Application of fast Kalman estimation to adaptive equalization," *IEEE Trans. on Communications*, vol. COM-26, pp. 1439-1446, Oct. 1978.

2. G. Carayannis, D. G. Manolakis, and N. Kalouptsidis, "A fast sequential algorithm for least-squares filtering and prediction," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-31, pp. 1394-1402, Dec. 1983.

3. J. M. Cioffi and T. Kailath, "Fast, recursive-least-squares transversal filters for adaptive filters," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-32, pp. 304-337, April 1984.

4. J. M. Cioffi and T. Kailath, "Windowed fast transversal filters adaptive algorithms with normalization," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-33, pp. 607-627, June 1985.

5. S. Ljung and L. Ljung, "Error propagation properties of recursive least-squares adaptation algorithms," *Automatica*, vol. 21, pp. 157-167, 1985.

6. J.-L. Botto and G. V. Moustakides, "Stabilizing the fast Kalman algorithms," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. 37, pp. 1342-1348, Sept. 1989.

7. M. Bellanger, "Engineering aspects of fast least squares algorithms in transversal adaptive filters," *Proc. IEEE Intern. Conf. on Acoust., Speech, Signal Processing*, pp. 49.14.1-49.14.4, 1987.

8. D. T. M. Slock and T. Kailath, "Fast transversal filters with data sequence weighting," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. 37, pp. 346-359, March 1989.

9. D. T. M. Slock and T. Kailath, "Numerically stable fast transversal filters for recursive least squares adaptive filtering," *IEEE Trans. on Signal Processing*, vol. 39, pp. 92-113, Jan. 1991.

10. J. G. Proakis, C. M. Rader, F. Ling, and C. L. Nikias, *Advanced Digital Signal Processing*, MacMillan, New York, NY, 1992.

11. B. Toplis and S. Pasupathy, "Tracking improvements in fast RLS algorithms using a variable forgetting factor," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. 36, pp. 206-227, Feb. 1988.

12. D. Boudreau and P. Kabal, "Joint-time delay estimation and adaptive recursive least squares filtering," *IEEE Trans. on Signal Processing*, vol. 41, pp. 592-601, Feb. 1993.

Problems

1. Show that

$$\boldsymbol{\phi}(k,N) \;=\; \mathbf{S}_D(k,N)\mathbf{x}(k,N)$$

$$= \frac{\mathbf{S}_D(k-1,N)\mathbf{x}(k,N)}{\lambda + \mathbf{x}^T(k,N)\mathbf{S}_D(k-1,N)\mathbf{x}(k,N)}$$

Hint: Use the matrix inversion lemma for $\mathbf{S}_D(k,N)$.

2. Show that

$$\phi_N(k-1,N) - \frac{\mathbf{w}_{f,N}(k)e_f(k,N)}{\xi^d_{f_{min}}(k,N)} = \frac{-e_b(k,N)}{\xi^d_{b_{min}}(k,N)} = \phi_{N+1}(k,N+1)$$

where $\mathbf{w}_{f,N}(k)$ represents the last element of $\mathbf{w}_f(k,N)$.

3. Using a proper mixture of relations of the lattice RLS algorithm based on *a posteriori* and the FTRLS algorithm, derive a fast exact initialization procedure for the transversal filter coefficients.

4. Show that the following relations are valid, assuming the input signals are prewindowed.

$$\frac{det[\mathbf{S}_D(k,N+1)]}{det[\mathbf{S}_D(k-1,N)]} = \frac{1}{\xi^d_{f_{min}}(k,N)}$$

$$\frac{det[\mathbf{S}_D(k,N+1)]}{det[\mathbf{S}_D(k,N)]} = \frac{1}{\xi^d_{b_{min}}(k,N)}$$

5. Show that

$$\gamma^{-1}(k,N) = \frac{det[\mathbf{R}_D(k,N)]}{\lambda^N det[\mathbf{R}_D(k-1,N)]}$$

Hint: $det[\mathbf{AB}] = det[\mathbf{BA}] = det[\mathbf{A}]det[\mathbf{B}]$.

6. Using the results of problems 4 and 5, prove that

$$\gamma^{-1}(k,N) = \frac{\xi^d_{f_{min}}(k,N)}{\lambda^N \xi^d_{b_{min}}(k,N)}$$

7. Derive equations (7.6) and (7.14). Also show that the use of $\phi(k,N)$ would increase the computational complexity of the FTRLS algorithm.

8. If one avoids the use of the conversion factor $\gamma(k,N)$, it is necessary to use inner products to derive the *a posteriori* errors in the fast algorithm. Derive a fast algorithm without conversion factor.

9. By replacing the relation for $\gamma(k, N, 3)$ in the SFTRLS algorithm by the relation

$$\gamma(k, N) = \frac{\lambda^N \xi^d_{b_{min}}(k, N)}{\xi^d_{f_{min}}(k, N)}$$

derived in problem 6, describe the resulting algorithm and show that it requires order $8N$ multiplications per output sample.

10. Derive the equation (7.29).

11. The FTRLS algorithm is applied to predict the signal $x(k) = \sin(\frac{\pi k}{4} + \frac{\pi}{3})$. Given $\lambda = 0.98$, calculate the error and the tap coefficients for the first 10 iterations.

12. The SFTRLS algorithm is applied to predict the signal $x(k) = \sin(\frac{\pi k}{4} + \frac{\pi}{3})$. Given $\lambda = 0.98$, calculate the error and the tap coefficients for the first 10 iterations.

13. The FTRLS algorithm was applied to identify a 7th-order unknown system whose coefficients are

$\mathbf{w}^T = [0.0272\ 0.0221\ -0.0621\ 0.1191\ 0.6116\ -0.3332\ -0.0190\ -0.0572]$

The input signal is a Gaussian white noise with variance $\sigma_x^2 = 1$, and the measurement noise is also a Gaussian white noise independent of the input signal, with variance $\sigma_n^2 = 0.01$.

Simulate the experiment described above and measure the excess MSE, for $\lambda = 0.97$ and $\lambda = 0.98$.

14. Repeat problem 13 in the case the input signal is a first-order Markov process with $\lambda_{\mathbf{x}} = 0.98$.

15. Run the problem 13 above using a fixed-point implementation with the FTRLS and SFTRLS algorithms. Use 12 bits in the fractional part of the signal and parameter representations.

16. Suppose a 15th-order FIR digital filter with multiplier coefficients given below was identified through an adaptive FIR filter of the same order using the FTRLS algorithm. Considering that fixed-point arithmetic was used, simulate the identification problem described using the following specifications.

Additional noise : white noise with variance	$\sigma_n^2 = 0.0015$
Coefficients wordlength:	$b_c = 16$ bits
Signal wordlength:	$b_d = 16$ bits
Input signal: Gaussian white noise with variance	$\sigma_x^2 = 0.7$
	$\lambda = 0.98$

$\mathbf{w}_o^T$ = [0.0219360 0.0015786 -0.0602449 -0.0118907 0.1375379 0.0574545 -0.3216703 -0.5287203 -0.2957797 0.0002043 0.290670 -0.0353349 -0.0068210 0.0026067 0.0010333 - 0.0143593]

Plot the learning curves for the finite- and infinite-precision implementations.

17. Repeat the problem above for the SFTRLS algorithm. Also reduce the wordlength used until a noticeable (10 percent increase) excess of MSE is noticed at output.

18. Repeat the problem 16 for the SFTRLS algorithm, using $\lambda = 0.999$ and $\lambda = 0.960$. Comment on the results.

19. The SFTRLS algorithm is used to perform the forward prediction of a signal $x(k)$ generated by applying a white noise with unit variance to the input of a linear filter with transfer function given by

$$H(z) = \frac{0.5}{(1 - 1.512z^{-1} + 0.827z^{-2})(1 - 1.8z^{-1} + 0.87z^{-2})}$$

Calculate the zeros of the resulting predictor and compare with the poles of the linear filter.

20. Perform the equalization of a channel with impulse response given by

$$h(k) = 0.96^k + (-0.9)^k$$

for $k = 0, 1, 2, \ldots, 15$. The transmitted signal is white noise with unit variance and the adaptive filter input signal-to-noise ratio is of $-30dB$. Use the SFTRLS algorithm of order 100.

8

QR-DECOMPOSITION-BASED RLS FILTERS

8.1 INTRODUCTION

The application of QR decomposition [1] to triangularize the input data matrix results in an alternative method for the implementation of the recursive least-squares (RLS) previously discussed. The main advantages brought about by the recursive least-squares algorithm based on QR decomposition are its possible implementation in systolic arrays [2]-[5] and its improved numerical behavior when quantization effects are taken into account [6].

The earlier proposed RLS algorithms based on the QR decomposition [2]-[4] aimed the triangularization of the information matrix in order to avoid the use of matrix inversion. However, their computational requirement was of $O[N^2]$ multiplications per output sample. Later, fast versions of the QR-RLS algorithms were proposed with a reduced computational complexity of $O[N]$ [5]-[8].

In this chapter, the QR-RLS algorithms based on Givens rotations are presented together with some stability considerations. Two fast algorithms are also discussed [5]-[8]. These fast algorithms are related to the tapped delay line FIR filter realization of the adaptive filter.

P. S. R. Diniz, *Adaptive Filtering*

8.2 QR-DECOMPOSITION-BASED RLS ALGORITHM

The RLS algorithm provides in a recursive way the coefficients of the adaptive filter which lead to the minimization of the following cost function

$$\begin{aligned}\xi^d(k) &= \sum_{i=0}^{k} \lambda^{k-i} e^2(i) \\ &= \sum_{i=0}^{k} \lambda^{k-i} [d(i) - \mathbf{x}^T(i)\mathbf{w}(k)]^2 \end{aligned} \tag{8.1}$$

where

$$\mathbf{x}(k) = [x(k)x(k-1)\ldots x(k-N)]^T$$

is the input signal vector,

$$\mathbf{w}(k) = [w_0(k)w_1(k)\ldots w_N(k)]^T$$

is the coefficient vector at instant k, $e(i)$ is the *a posteriori* error at instant i, and λ is the forgetting factor.

The same problem can be rewritten as a function of increasing dimension matrices and vectors which contain all the weighted signal information so far available to the adaptive filter. These matrices are redefined here for convenience

$$\begin{aligned}\underline{\mathbf{X}}^T(k) &= \mathbf{X}(k) \\ &= \begin{bmatrix} x(k) & \lambda^{1/2}x(k-1) & \cdots & \lambda^{(k-1)/2}x(1) & \lambda^{k/2}x(0) \\ x(k-1) & \lambda^{1/2}x(k-2) & \cdots & \lambda^{(k-1)/2}x(0) & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x(k-N) & \lambda^{1/2}x(k-N-1) & \cdots & 0 & 0 \end{bmatrix} \\ &= [\mathbf{x}(k)\lambda^{1/2}\mathbf{x}(k-1)\ldots\lambda^{k/2}\mathbf{x}(0)] \end{aligned} \tag{8.2}$$

$$\mathbf{y}(k) = \underline{\mathbf{X}}(k)\mathbf{w}(k) = \begin{bmatrix} y(k) \\ \lambda^{1/2}y(k-1) \\ \vdots \\ \lambda^{k/2}y(0) \end{bmatrix} \tag{8.3}$$

$$\mathbf{d}(k) = \begin{bmatrix} d(k) \\ \lambda^{1/2}d(k-1) \\ \vdots \\ \lambda^{k/2}d(0) \end{bmatrix} \tag{8.4}$$

$$\mathbf{e}(k) = \begin{bmatrix} e(k) \\ \lambda^{1/2}e(k-1) \\ \vdots \\ \lambda^{k/2}e(0) \end{bmatrix} = \mathbf{d}(k) - \mathbf{y}(k) \tag{8.5}$$

The objective function of equation (8.1) can now be rewritten as

$$\xi^d(k) = \mathbf{e}^T(k)\mathbf{e}(k) \tag{8.6}$$

As shown in chapter 5, equation (5.13), the optimal solution to the least-squares problem at a given instant of time k can be found by solving the following equation

$$\underline{\mathbf{X}}^T(k)\underline{\mathbf{X}}(k)\mathbf{w}(k) = \underline{\mathbf{X}}^T(k)\mathbf{d}(k) \tag{8.7}$$

However, solving this equation by using the conventional RLS algorithm can be a problem when the matrix $\mathbf{R}_D(k) = \underline{\mathbf{X}}^T(k)\underline{\mathbf{X}}(k)$ and its correspondent inverse estimate become ill-conditioned due to loss of persistency of excitation of the input signal or to quantization effects.

The QR decomposition approach avoids inaccurate solutions to the RLS problem and allows easy monitoring of the positive definiteness of a transformed information matrix in ill-conditioned situations.

8.2.1 Initialization Process

During the initialization period, i.e., from $k = 0$ to $k = N$, the solution of equation (8.7) can be found exactly without using any matrix inversion. From equation (8.7), it can easily be found that for $k = 0$ and $x(0) \neq 0$

$$w_0(0) = \frac{d(0)}{x(0)} \tag{8.8}$$

for $k = 1$

$$\begin{aligned} w_0(1) &= \frac{d(0)}{x(0)} \\ w_1(1) &= \frac{-x(1)w_0(1) + d(1)}{x(0)} \end{aligned} \tag{8.9}$$

for $k = 2$

$$\begin{aligned} w_0(2) &= \frac{d(0)}{x(0)} \\ w_1(2) &= \frac{-x(1)w_0(2) + d(1)}{x(0)} \\ w_2(2) &= \frac{-x(2)w_0(2) - x(1)w_1(2) + d(2)}{x(0)} \end{aligned} \tag{8.10}$$

at the instant k by induction we can show that

$$w_i(k) = \frac{-\sum_{j=1}^{i} x(j)w_{i-j}(k) + d(i)}{x(0)} \tag{8.11}$$

The equation above represents the so-called back-substitution algorithm.

8.2.2 Input data matrix triangularization

After the instant $k = N$, the equation (8.11) above is no longer valid and the inversion of $\mathbf{R}_D(k)$ or the calculation of $\mathbf{S}_D(k)$ is required to find the optimal solution for the coefficients $\mathbf{w}(k)$. This is exactly what makes the RLS more sensitive to quantization effects and input signal conditioning. The matrix $\underline{\mathbf{X}}(k)$ at instant $k = N + 1$ is given by

$$\begin{aligned} \underline{\mathbf{X}}(N+1) &= \begin{bmatrix} x(N+1) & x(N) & \cdots & x(1) \\ \lambda^{1/2}x(N) & \lambda^{1/2}x(N-1) & \cdots & \lambda^{1/2}x(0) \\ \lambda x(N-1) & \lambda x(N-2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda^{\frac{N+1}{2}}x(0) & 0 & \cdots & 0 \end{bmatrix} \\ &= \begin{bmatrix} x(N+1)x(N)\cdots x(1) \\ \lambda^{1/2}\underline{\mathbf{X}}(N) \end{bmatrix} = \begin{bmatrix} \mathbf{x}^T(N+1) \\ \lambda^{1/2}\underline{\mathbf{X}}(N) \end{bmatrix} \end{aligned} \tag{8.12}$$

as it is noted, the matrix $\underline{\mathbf{X}}(k)$ is no longer upper triangular, then the back-substitution algorithm cannot be employed to find the tap-weight coefficients.

The matrix $\underline{\mathbf{X}}(N + 1)$ can be triangularized through an orthogonal triangularization approach such as Givens rotations, Householder transformation, or Gram-Schmidt orthogonalization [1]. Since here the interest is to iteratively

apply the triangularization procedure to each new data vector added to $\underline{\mathbf{X}}(k)$, the Givens rotation seems to be the most appropriate approach.

In the Givens rotation approach, each element of the first line of equation (8.12) can be eliminated by premultiplying the matrix $\underline{\mathbf{X}}(N+1)$ by a series of Givens rotation matrices given by

$$\begin{aligned}
\tilde{\mathbf{Q}}(N+1) &= \mathbf{Q}'_N(N+1)\cdot\mathbf{Q}'_{N-1}(N+1)\cdots\mathbf{Q}'_0(N+1) \\
&= \begin{bmatrix} \cos\theta_N(N+1) & \cdots & 0 & \cdots & -\sin\theta_N(N+1) \\ \vdots & & & & \vdots \\ 0 & & \mathbf{I}_N & & 0 \\ \vdots & & & & \vdots \\ \sin\theta_N(N+1) & \cdots & 0 & \cdots & \cos\theta_N(N+1) \end{bmatrix} \\
&\cdot \begin{bmatrix} \cos\theta_{N-1}(N+1) & \cdots & 0 & \cdots & -\sin\theta_{N-1}(N+1) & 0 \\ \vdots & & & & \vdots & \vdots \\ 0 & & \mathbf{I}_{N-1} & & 0 & 0 \\ \vdots & & & & \vdots & \vdots \\ \sin\theta_{N-1}(N+1) & \cdots & 0 & \cdots & \cos\theta_{N-1}(N+1) & 0 \\ 0 & \cdots & 0 & \cdots & 0 & 1 \end{bmatrix} \\
&\cdots \begin{bmatrix} \cos\theta_0(N+1) & -\sin\theta_0(N+1) & \cdots & 0 & \cdots & 0 \\ \sin\theta_0(N+1) & \cos\theta_0(N+1) & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & & & & \\ 0 & 0 & & \mathbf{I}_N & & \\ \vdots & \vdots & & & & \\ 0 & 0 & & & & \end{bmatrix}
\end{aligned} \tag{8.13}$$

where $\mathbf{I}_i$ is an i by i identity matrix. The rotation angles θ_i are chosen such that each entry of the first row of the resulting matrix is zero. Consider first the matrix product $\mathbf{Q}'_0(N+1)\underline{\mathbf{X}}(N+1)$, if:

$$\cos\theta_0(N+1)x(1) - \sin\theta_0(N+1)\lambda^{1/2}x(0) = 0 \tag{8.14}$$

the element in the position $(1, N+1)$ of the resulting matrix product will be zero. If it is further considered that $\cos^2\theta_0(N+1) + \sin^2\theta_0(N+1) = 1$, it can be easily deduced that

$$\cos\theta_0(N+1) = \frac{\lambda^{1/2}x(0)}{\sqrt{\lambda x^2(0) + x^2(1)}} \tag{8.15}$$

$$\sin\theta_0(N+1) = \frac{x(1)}{\sqrt{\lambda x^2(0) + x^2(1)}} \tag{8.16}$$

Next, $\mathbf{Q}_1'(N+1)$ premultiplies $\mathbf{Q}_0'(N+1)\underline{\mathbf{X}}(N+1)$ with the objective of generating a zero element at the position $(1, N)$ in the matrix resulting from the product. Note that the present matrix product does not remove the zero of the element $(1, N+1)$. The required rotation angle can be calculated by first noting that the elements $(1, N)$ and $(3, N)$ of $\mathbf{Q}_0'(N+1)\underline{\mathbf{X}}(N+1)$ are respectively

$$a = \cos\theta_0(N+1)x(2) - \lambda^{1/2}x(1)\sin\theta_0(N+1) \tag{8.17}$$

$$b = \lambda x(0) \tag{8.18}$$

then it follows that

$$\cos\theta_1(N+1) = \frac{b}{\sqrt{a^2+b^2}} \tag{8.19}$$

$$\sin\theta_1(N+1) = \frac{a}{\sqrt{a^2+b^2}} \tag{8.20}$$

In this manner, after the last Givens rotation the input signal information matrix will be transformed in a matrix with null first row

$$\tilde{\mathbf{Q}}(N+1)\underline{\mathbf{X}}(N+1) = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ & & \mathbf{U}(N+1) & \end{bmatrix} \tag{8.21}$$

where $\mathbf{U}(N+1)$ is an upper triangular matrix.

In the next iteration, the input signal matrix $\underline{\mathbf{X}}(N+2)$ receives a new row that should be replaced by a zero vector through a QR decomposition. In this step, the matrices involved are the following

$$\underline{\mathbf{X}}(N+2) = \begin{bmatrix} x(N+2) & x(N+1) & \cdots & x(2) \\ & \lambda^{1/2}\underline{\mathbf{X}}(N+1) & & \end{bmatrix} \tag{8.22}$$

$$\begin{bmatrix} 1 & 0 & \cdots & \cdots \\ 0 & & & \\ \vdots & & \tilde{\mathbf{Q}}(N+1) & \\ \vdots & & & \end{bmatrix} \underline{\mathbf{X}}(N+2)$$

$$= \begin{bmatrix} x(N+2) & x(N+1) & \cdots & x(2) \\ 0 & 0 & \cdots & 0 \\ & \lambda^{1/2}\mathbf{U}(N+1) & & \end{bmatrix} \tag{8.23}$$

In order to eliminate the new input vector through rotations with the corresponding rows of the triangular matrix $\lambda^{1/2}\mathbf{U}(N+1)$, we apply the QR decomposition to equation (8.23) as follows:

$$\tilde{\mathbf{Q}}(N+2)\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Q}}(N+1) \end{bmatrix}\underline{\mathbf{X}}(N+2) = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ & & \mathbf{U}(N+2) & \end{bmatrix} \tag{8.24}$$

where again $\mathbf{U}(N+2)$ is an upper triangular matrix and $\tilde{\mathbf{Q}}(N+2)$ is given by

$$\begin{aligned}
\tilde{\mathbf{Q}}(N+2) &= \mathbf{Q}'_N(N+2)\mathbf{Q}'_{N-1}(N+2)\cdots\mathbf{Q}'_0(N+2) \\
&= \begin{bmatrix} \cos\theta_N(N+2) & \cdots & 0 & \cdots & -\sin\theta_N(N+2) \\ \vdots & & & & \vdots \\ 0 & & \mathbf{I}_{N+1} & & 0 \\ \vdots & & & & \vdots \\ \sin\theta_N(N+2) & \cdots & 0 & \cdots & \cos\theta_N(N+2) \end{bmatrix} \\
&\cdot \begin{bmatrix} \cos\theta_{N-1}(N+2) & \cdots & 0 & \cdots & -\sin\theta_{N-1}(N+2) & 0 \\ \vdots & & & & & \vdots \\ 0 & & & \mathbf{I}_N & & 0 \\ \vdots & & & & & \vdots \\ \sin\theta_{N-1}(N+2) & & & & \cos\theta_{N-1}(N+2) & 0 \\ 0 & \cdots & 0 & \cdots & 0 & 1 \end{bmatrix} \\
&\cdots \begin{bmatrix} \cos\theta_0(N+2) & 0 & -\sin\theta_0(N+2) & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \sin\theta_0(N+2) & 0 & \cos\theta_0(N+2) & \cdots & 0 \\ \vdots & \vdots & \vdots & & \\ \vdots & \vdots & \vdots & & \mathbf{I}_N \\ 0 & 0 & 0 & & \end{bmatrix}
\end{aligned} \tag{8.25}$$

The procedure above should be repeated for each new incoming input signal vector as follows:

$$\begin{aligned}
\mathbf{Q}(k)\underline{\mathbf{X}}(k) &= \tilde{\mathbf{Q}}(k)\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Q}}(k-1) \end{bmatrix}\begin{bmatrix} \mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Q}}(k-2) \end{bmatrix} \\
&\cdots \begin{bmatrix} \mathbf{I}_{k-N} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Q}}(k-N) \end{bmatrix}\underline{\mathbf{X}}(k) = \begin{bmatrix} \mathbf{0} \\ \\ \mathbf{U}(k) \\ \underbrace{\qquad}_{N+1} \end{bmatrix} \begin{matrix} \} k-N \\ \} N+1 \end{matrix}
\end{aligned} \tag{8.26}$$

where $\mathbf{Q}(k)$ is a $(k+1)$ by $(k+1)$ matrix which represents the overall triangularization matrix via elementary Givens rotations matrices $\mathbf{Q}'_i(m)$ for all $m \leq k$ and $0 \leq i \leq N$. Since each Givens rotation matrix is orthogonal then it can easily be proved that $\mathbf{Q}(k)$ is also orthogonal (actually orthonormal), i.e.,

$$\mathbf{Q}(k)\mathbf{Q}^T(k) = \mathbf{I}_{k+1} \tag{8.27}$$

Also, from equation (8.25), it is straightforward to note that

$$\mathbf{Q}(k) = \tilde{\mathbf{Q}}(k) \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}(k-1) \end{bmatrix} \tag{8.28}$$

where $\tilde{\mathbf{Q}}(k)$ is responsible for zeroing the latest input vector $\mathbf{x}^T(k)$ in the first row of $\underline{\mathbf{X}}(k)$. The matrix $\tilde{\mathbf{Q}}(k)$ is given by

$$\tilde{\mathbf{Q}}(k) = \begin{bmatrix} \cos\theta_N(k) & \cdots & 0 & \cdots & -\sin\theta_N(k) \\ \vdots & & & & \vdots \\ 0 & & \mathbf{I}_{k-1} & & 0 \\ \vdots & & & & \vdots \\ \sin\theta_N(k) & \cdots & 0 & \cdots & \cos\theta_N(k) \end{bmatrix}$$

$$\cdot \begin{bmatrix} \cos\theta_{N-1}(k) & \cdots & 0 & \cdots & -\sin\theta_{N-1}(k) & 0 \\ \vdots & & & & \vdots & \vdots \\ 0 & & \mathbf{I}_{k-2} & & 0 & 0 \\ \vdots & & & & \vdots & \vdots \\ \sin\theta_{N-1}(k) & \cdots & 0 & \cdots & \cos\theta_{N-1}(k) & 0 \\ 0 & \cdots & 0 & \cdots & 0 & 1 \end{bmatrix}$$

$$\cdots \begin{bmatrix} \cos\theta_0(k) & \cdots & 0 & \cdots & -\sin\theta_0(k) & 0 \\ \vdots & & & & \vdots & \vdots \\ 0 & & \mathbf{I}_{k-N-1} & & 0 & 0 \\ \vdots & & & & \vdots & \vdots \\ \sin\theta_0(k) & \cdots & 0 & \cdots & \cos\theta_0(k) & 0 \\ & & \mathbf{0} & & & \mathbf{I}_N \end{bmatrix}$$

$$
= \begin{bmatrix}
\prod_{i=0}^{N} \cos\theta_i(k) & \cdots & 0 & \cdots & -\prod_{i=1}^{N} \cos\theta_i(k)\sin\theta_0(k) \\
\vdots & & & & \vdots \\
0 & & \mathbf{I}_{k-N-1} & & 0 \\
\vdots & & & & \vdots \\
\sin\theta_0(k) & & & & \cos\theta_0(k) \\
\vdots & & \vdots & & \vdots \\
\prod_{i=0}^{j-1} \cos\theta_i(k)\sin\theta_j(k) & \cdots & 0 & \cdots & \vdots \\
\vdots & & \vdots & & -\sin\theta_N(k)\prod_{i=1}^{N-1} \cos\theta_i(k)\sin\theta_0(k)
\end{bmatrix}
$$

$$
\begin{bmatrix}
\cdots & -\prod_{i=j+1}^{N} \cos\theta_i(k)\sin\theta_j(k) & \cdots & -\sin\theta_N(k) \\
 & & 0 & \\
\ddots & & & \\
 & \ddots & & \mathbf{0} \\
 & & \cos\theta_{N-1}(k) & \\
 & & & \cos\theta_N(k)
\end{bmatrix} \tag{8.29}
$$

Note that the matrix $\tilde{\mathbf{Q}}(k)$ has the following general form

$$
\tilde{\mathbf{Q}}(k) = \begin{bmatrix}
* & 0 & \cdots & 0 & \cdots & 0 & * & \cdots & * \\
0 & & & & & & & & \\
\vdots & & & \mathbf{I}_{k-N-1} & & & \mathbf{0} & & \\
* & & & & & & * & & \\
\vdots & & & \mathbf{0} & & & & \ddots & \\
* & & & & & & * & & *
\end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \end{matrix} \overbrace{\qquad}^{N+1} \; \Big\} N+1 \tag{8.30}
$$

where $*$ is a nonzero element. This structure of $\tilde{\mathbf{Q}}(k)$ will be useful in developing the fast QR-RLS algorithms later on.

Returning to equation (8.25), we can conclude that

$$\mathbf{Q}(k)\underline{\mathbf{X}}(k) = \tilde{\mathbf{Q}}(k)\begin{bmatrix} x(k) & x(k-1) & \cdots & x(k-N) \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ & \lambda^{1/2}\mathbf{U}(k-1) & & \end{bmatrix} \tag{8.31}$$

The first Givens rotation angle required to replace $x(k-N)$ by a zero is $\theta_0(k)$ such that

$$\cos\theta_0(k)x(k-N) - \sin\theta_0(k)\lambda^{1/2}u_{1,N+1}(k-1) = 0 \tag{8.32}$$

where $u_{1,N+1}(k-1)$ is the element $(1, N+1)$ of $\mathbf{U}(k-1)$. Then, it follows that

$$\cos\theta_0(k) = \frac{\lambda^{1/2}u_{1,N+1}(k-1)}{u_{1,N+1}(k)} \tag{8.33}$$

$$\sin\theta_0(k) = \frac{x(k-N)}{u_{1,N+1}(k)} \tag{8.34}$$

where

$$u^2_{1,N+1}(k) = x^2(k-N) + \lambda u^2_{1,N+1}(k-1) \tag{8.35}$$

From equation (8.35), it is worth noting that the $(1, N+1)$ element of $\mathbf{U}(k)$ is the square root of the exponentially weighted input signal energy, i.e.,

$$u^2_{1,N+1}(k) = \sum_{i=0}^{k-N} \lambda^i x^2(k-N-i) \tag{8.36}$$

In the triangularization process, all the submatrices multiplying each column of $\underline{\mathbf{X}}(k)$ are orthogonal matrices and as a consequence the norm of each column in $\underline{\mathbf{X}}(k)$ and $\tilde{\mathbf{Q}}(k)\underline{\mathbf{X}}(k)$ should be the same. This confirms that equation (8.36) is valid, and also it can be easily shown that

$$\sum_{i=1}^{k+1} \underline{x}^2_{i,j}(k) = \sum_{i=1}^{N+2-j} u^2_{i,j}(k) = \sum_{i=1}^{k+1} \lambda^{i-1}x^2(k+2-i-j) \tag{8.37}$$

for $j = 1, 2, \ldots, N+1$.

Now consider that the intermediate calculations of equation (8.31) are performed as follows:

$$\tilde{\mathbf{Q}}(k)\begin{bmatrix}\mathbf{x}^T(k)\\ \mathbf{0}\\ \lambda^{1/2}\mathbf{U}(k-1)\end{bmatrix} = \mathbf{Q}'_N(k)\mathbf{Q}'_{N-1}(k)\cdots\mathbf{Q}'_i(k)\begin{bmatrix}\mathbf{x}'_i(k)\\ \mathbf{0}\\ \mathbf{U}'_i(k)\end{bmatrix} \tag{8.38}$$

where $\mathbf{x}'_i(k) = [x'_i(k)\ x'_i(k-1)\ldots x'_i(k-N-i)\ 0\cdots 0]$ and $\mathbf{U}'_i(k)$ is an intermediate upper triangular matrix. Note that $\mathbf{x}'_0(k) = \mathbf{x}^T(k)$, $\mathbf{U}'_0(k) = \lambda^{1/2}\mathbf{U}(k-1)$, and $\mathbf{U}'_{N+1}(k) = \mathbf{U}(k)$. In practice, the multiplication by the zero elements in equation (8.38) can be avoided. We start by removing the increasing $\mathbf{I}_{k-N-1}$ section of $\tilde{\mathbf{Q}}(k)$ (see equation (8.30)), generating a matrix with reduced dimension denoted by $\mathbf{Q}_\theta(k)$. The resulting equation is

$$\begin{aligned}\mathbf{Q}_\theta(k)\begin{bmatrix}\mathbf{x}^T(k)\\ \lambda^{1/2}\mathbf{U}(k-1)\end{bmatrix} &= \mathbf{Q}'_{\theta_N}(k)\mathbf{Q}'_{\theta_{N-1}}(k)\cdots\mathbf{Q}'_{\theta_i}(k)\begin{bmatrix}\mathbf{x}'_i(k)\\ \mathbf{U}'_i(k)\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{0}\\ \mathbf{U}(k)\end{bmatrix}\end{aligned} \tag{8.39}$$

where $\mathbf{Q}'_{\theta_i}(k)$ is derived from $\mathbf{Q}'_i(k)$ by removing the $\mathbf{I}_{k-N-1}$ section of $\mathbf{Q}'_i(k)$ along with the corresponding rows and columns, resulting in the following form

$$\mathbf{Q}'_{\theta_i}(k) = \begin{bmatrix}\cos\theta_i(k) & \cdots & 0 & \cdots & -\sin\theta_i(k) & \cdots & 0\\ \vdots & & & & \vdots & & \vdots\\ 0 & & \mathbf{I}_i & & 0 & \cdots & 0\\ \vdots & & & & \vdots & & \vdots\\ \sin\theta_i(k) & \cdots & 0 & \cdots & \cos\theta_i(k) & \cdots & 0\\ \vdots & & \vdots & & \vdots & \mathbf{I}_{N-i} & \\ 0 & \cdots & 0 & \cdots & 0 & & \end{bmatrix} \tag{8.40}$$

The Givens rotation elements are calculated by

$$\cos\theta_i(k) = \frac{[\mathbf{U}'_i(k)]_{i+1,N+1-i}}{c_i} \tag{8.41}$$

$$\sin\theta_i(k) = \frac{x'_i(k-N-i)}{c_i} \tag{8.42}$$

where $c_i = \sqrt{[\mathbf{U}'_i(k)]^2_{i+1,N+1-i} + x'^2_i(k-N-i)}$ and $[\cdot]_{i,j}$ is the (i,j) element of the matrix.

8.2.3 QR-Decomposition RLS Algorithm

The triangularization procedure discussed above can be applied to generate the QR-RLS algorithm that avoids the estimation of the $\mathbf{S}_D(k)$ matrix of the conventional RLS algorithm. The weighted *a posteriori* error vector can be written as a function of the input data matrix, that is

$$\mathbf{e}(k) = \begin{bmatrix} e(k) \\ \lambda^{1/2}e(k-1) \\ \vdots \\ \lambda^{k/2}e(0) \end{bmatrix} = \begin{bmatrix} d(k) \\ \lambda^{1/2}d(k-1) \\ \vdots \\ \lambda^{k/2}d(0) \end{bmatrix} - \underline{\mathbf{X}}(k)\mathbf{w}(k) \tag{8.43}$$

By premultiplying the above equation by $\mathbf{Q}(k)$, it follows that

$$\begin{aligned} \mathbf{e}_q(k) = \mathbf{Q}(k)\mathbf{e}(k) &= \mathbf{Q}(k)\mathbf{d}(k) - \mathbf{Q}(k)\underline{\mathbf{X}}(k)\mathbf{w}(k) \\ &= \mathbf{d}_q(k) - \begin{bmatrix} \mathbf{0} \\ \mathbf{U}(k) \end{bmatrix} \mathbf{w}(k) \end{aligned} \tag{8.44}$$

where

$$\mathbf{e}_q(k) = \begin{bmatrix} e_{q_1}(k) \\ e_{q_2}(k) \\ \vdots \\ e_{q_{k+1}}(k) \end{bmatrix}, \qquad \mathbf{d}_q(k) = \begin{bmatrix} d_{q_1}(k) \\ d_{q_2}(k) \\ \vdots \\ d_{q_{k+1}}(k) \end{bmatrix}$$

Since $\mathbf{Q}(k)$ is an orthogonal matrix, equation (8.6) is equivalent to

$$\xi^d(k) = \mathbf{e}_q^T(k)\mathbf{e}_q(k) \tag{8.45}$$

because

$$\mathbf{e}_q^T(k)\mathbf{e}_q(k) = \mathbf{e}^T(k)\mathbf{Q}^T(k)\mathbf{Q}(k)\mathbf{e}(k) = \mathbf{e}^T(k)\mathbf{e}(k)$$

The weighted-square error can be minimized in equation (8.45) by calculating $\mathbf{w}(k)$ such that $e_{q_{k-N+1}}(k)$ to $e_{q_{k+1}}(k)$ are made zero using a back-substitution algorithm such as

$$w_i(k) = \frac{-\sum_{j=1}^{i} u_{N+1-i,i-j+1}(k)w_{i-j}(k) + d_{q,k+1-i}(k)}{u_{N+1-i,i+1}(k)} \tag{8.46}$$

for $i = 0, 1, \ldots, N$, where $\sum_{j=i}^{i-1} \cdot = 0$. With this choice for $\mathbf{w}(k)$, the minimum weighted-square error at instant k is given by

$$\xi_{min}^{d}(k) = \sum_{i=1}^{k-N} e_{q_i}^2(k) \tag{8.47}$$

An important relation can be deduced by rewriting equation (8.44) as

$$\begin{aligned} \mathbf{d}_q(k) &= \begin{bmatrix} \mathbf{d}_{q_1}(k) \\ \hline \mathbf{d}_{q_2}(k) \end{bmatrix} = \begin{bmatrix} d_{q_1}(k) \\ \vdots \\ d_{q_{k-N}}(k) \\ \hline d_{q_{k-N+1}}(k) \\ \vdots \\ d_{q_{k+1}}(k) \end{bmatrix} \\ &= \begin{bmatrix} e_{q_1}(k) \\ \vdots \\ e_{q_{k-N}}(k) \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{U}(k) \end{bmatrix} \mathbf{w}(k) \end{aligned} \tag{8.48}$$

where $\mathbf{w}(k)$ is the optimum coefficient vector at instant k. By examining the equations (8.31) and (8.44), the rightmost side of equation (8.48) can then be expressed as

$$\begin{bmatrix} \mathbf{e}_{q_1}(k) \\ \mathbf{d}_{q_2}(k) \end{bmatrix} = \begin{bmatrix} e_{q_1}(k) \\ \vdots \\ e_{q_{k-N}}(k) \\ \mathbf{d}_{q_2}(k) \end{bmatrix} = \tilde{\mathbf{Q}}(k) \begin{bmatrix} d(k) \\ \lambda^{1/2} \begin{bmatrix} e_{q_1}(k-1) \\ \vdots \\ e_{q_{k-N-1}}(k-1) \\ \mathbf{d}_{q_2}(k-1) \end{bmatrix} \end{bmatrix} \tag{8.49}$$

Using similar arguments around equations (8.38) to (8.40), and staring from equation (8.49), the transformed weighted-error vector can be updated as de-

scribed below

$$\tilde{\mathbf{Q}}(k)\begin{bmatrix} d(k) \\ \lambda^{1/2}\begin{bmatrix} \mathbf{e}_{q_1}(k-1) \\ \mathbf{d}_{q_2}(k-1) \end{bmatrix} \end{bmatrix} = \mathbf{Q}'_N(k)\mathbf{Q}'_{N-1}(k)\cdots\mathbf{Q}'_i(k)\begin{bmatrix} d'_i(k) \\ \mathbf{e}'_{q_i}(k) \\ \mathbf{d}'_{q_{2i}}(k) \end{bmatrix} \tag{8.50}$$

where $d'_i(k)$, $\mathbf{e}'_{q_i}(k)$, and $\mathbf{d}'_{q_{2i}}(k)$ are intermediate quantities generated during the rotations. Note that $\mathbf{e}'_{q_{N+1}}(k) = [e_{q_2}(k)e_{q_3}(k)\ldots e_{q_{k-N}}(k)]^T$, $d'_{N+1}(k) = e_{q_1}(k)$, and $\mathbf{d}'_{q_{2N+1}} = \mathbf{d}_{q_2}(k)$.

If we delete all the columns and rows of $\tilde{\mathbf{Q}}(k)$ whose elements are zeros and ones, i.e., the $\mathbf{I}_{k-N-1}$ section of $\tilde{\mathbf{Q}}(k)$ with the respective bands of zeros below, above, and on each side of it in equation (8.30), one would obtain matrix $\mathbf{Q}_\theta(k)$. In this case, the resulting equation corresponding to (8.49) is given by

$$\underline{\mathbf{d}}(k) = \begin{bmatrix} e_{q_1}(k) \\ \mathbf{d}_{q_2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k)\begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q_2}(k-1) \end{bmatrix} \tag{8.51}$$

Therefore, we eliminate the vector $\mathbf{e}'_{q_{N+1}}(k)$ which is always increasing, such that in real-time implementation the updating is performed through

$$\begin{aligned} \underline{\mathbf{d}}(k) &= \mathbf{Q}_\theta(k)\begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q_2}(k-1) \end{bmatrix} \\ &= \mathbf{Q}'_{\theta_N}(k)\mathbf{Q}'_{\theta_{N-1}}(k)\cdots\mathbf{Q}'_{\theta_i}(k)\begin{bmatrix} d'_i(k) \\ \mathbf{d}'_{q_{2i}}(k) \end{bmatrix} \end{aligned} \tag{8.52}$$

Another important relation can be derived from equation (8.44) by premultiplying both sides by $\mathbf{Q}^T(k)$, transposing the result, and postmultiplying the result by the pinning vector

$$\mathbf{e}_q^T(k)\mathbf{Q}(k)\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \mathbf{e}^T(k)\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = e(k) \tag{8.53}$$

Then, from the definition of $\mathbf{Q}(k)$ in equations (8.28) and (8.29), the following relation is obtained

$$\begin{aligned} e(k) &= e_{q_1}(k)\prod_{i=0}^{N}\cos\theta_i(k) \\ &= e_{q_1}(k)\gamma(k) \end{aligned} \tag{8.54}$$

This relation shows that the *a posteriori* output error can be computed without the explicit calculation of $\mathbf{w}(k)$. The only information needed is the Givens rotation cosines. In applications where only the *a posteriori* output error is of interest, the computing intensive back-substitution algorithm of equation (8.46) to obtain $\mathbf{w}_i(k)$ can be avoided [3].

Now, all the mathematical background to develop the QR-RLS algorithm has been derived. After initialization, the Givens rotation elements are computed using equations (8.41) and (8.42). These rotations are then applied to the information matrix and the desired signal vector respectively as indicated in equations (8.39) and (8.52). The next step is to compute the error signal using equation (8.54). Finally, if the tap-weight coefficients are required we should calculate them using equation (8.46). Algorithm 8.1 summarizes the algorithm with all essential computations.

Example 8.1

In this example, we solve the system identification problem described in subsection 3.6.2 by using the QR-RLS algorithm described in this section.

Solution:

In the present example, we are mainly concerned in testing the algorithm implemented in finite precision, since the remaining characteristics (such as: misadjustment, convergence speed etc.) should follow the same pattern of the conventional RLS algorithm. We considered the case where eigenvalue spread of the input signal correlation matrix is 20, with $\lambda = 0.99$. The presented results were obtained by averaging the outcomes of 200 independent runs. Table 8.1 summarizes the results, where it can be noticed that the MSE is comparable to the case of the conventional RLS (consult Table 5.2). On the other hand, the quantization error introduced by the calculations to obtain $\mathbf{w}(k)_Q$ is considerable. After leaving the algorithm running for a large number of iterations, we found no sign of divergence.

In the infinite-precision implementation, the misadjustment measured was 0.0429. As expected, consult Table 5.1, this result is close to the misadjustment obtained by the conventional RLS algorithm.

□

Algorithm 8.1
QR-RLS Algorithm

$\mathbf{w}(-1)=[0\ 0\ \cdots\ 0]^T, \quad w_0(0)=\frac{d(0)}{x(0)}$

For $k=1$ to N (Initialization)

$w_0(k)=\frac{d(0)}{x(0)}$

Do for $i=1$ to k

$$w_i(k)=\frac{-\sum_{j=1}^{i} x(j)w_{i-j}(k)+d(i)}{x(0)} \tag{8.11}$$

End

End

$$\mathbf{U}'_0(N+1)=\lambda^{1/2}\underline{\mathbf{X}}(N) \tag{8.12}$$

$\mathbf{d}'_{q_2\,0}(N+1)=[\ \lambda^{1/2}d(N)\ \ \lambda d(N-1)\ \cdots\ \lambda^{(N+1)/2}d(0)]^T$

For $k\geq N+1$

Do for each k

$\gamma'_{-1}=1$

$d'_0(k)=d(k)$

$\mathbf{x}'_0(k)=\mathbf{x}^T(k)$

Do for $i=0$ to N

$c_i=\sqrt{[\mathbf{U}'_i(k)]^2_{i+1,N+1-i}+x'^2_i(k-N-i)}$

$$\cos\theta_i=\frac{[\mathbf{U}'_i(k)]_{i+1,N+1-i}}{c_i} \tag{8.41}$$

$$\sin\theta_i=\frac{x'_i(k-N-i)}{c_i} \tag{8.42}$$

$$\begin{bmatrix}\mathbf{x}'_{i+1}(k)\\ \mathbf{U}'_{i+1}(k)\end{bmatrix}=\mathbf{Q}'_{\theta_i}(k)\begin{bmatrix}\mathbf{x}'_i(k)\\ \mathbf{U}'_i(k)\end{bmatrix} \tag{8.39}$$

$$\gamma'_i=\gamma'_{i-1}\cos\theta_i \tag{8.54}$$

$$\begin{bmatrix}d'_{i+1}(k)\\ \mathbf{d}'_{q_{2i+1}}(k)\end{bmatrix}=\mathbf{Q}'_{\theta_i}(k)\begin{bmatrix}d'_i(k)\\ \mathbf{d}'_{q_{2i}}(k)\end{bmatrix} \tag{8.51}$$

End

$\mathbf{d}'_{q_{20}}(k+1)=\lambda^{1/2}\mathbf{d}'_{q_{2N+1}}(k)$

$\mathbf{U}'_0(k+1)=\lambda^{1/2}\mathbf{U}'_{N+1}(k)$

$\gamma(k)=\gamma'_N$

$$e(k)=d'_{N+1}(k)\gamma(k) \tag{8.51}$$

If required compute

$$\underline{\mathbf{d}}(k)=\begin{bmatrix}d'_{N+1}(k)\\ \mathbf{d}'_{q_{2N+1}}(k)\end{bmatrix} \tag{8.51}$$

$w_0(k)=\frac{\underline{d}_{N+1}(k)}{u_{N+1,1}(k)}$

Do for $i=1$ to N

$$w_i(k)=\frac{-\sum_{j=1}^{i} u_{N+1-i,j}(k)w_{i-j}(k)+\underline{d}_{N+1-i}(k)}{u_{N+1-i,i+1}(k)} \tag{8.46}$$

End

End

□

Table 8.1 Results of the Finite-Precision Implementation of the QR-RLS Algorithm

No. of bits	$\xi(k)_Q$ Experiment	$E[\|\|\Delta\mathbf{w}(k)_Q\|\|^2]$ Experiment
16	$1.544\ 10^{-3}$	0.03473
12	$1.563\ 10^{-3}$	0.03254
10	$1.568\ 10^{-3}$	0.03254

8.3 SYSTOLIC ARRAY IMPLEMENTATION

The systolic array implementation of a given algorithm consists of mapping the algorithm in a pipelined sequence of basic computation cells. These basic cells perform their task in parallel, such that in each clock period all the cells are activated. The complete treatment of systolic array implementation and parallelization of algorithms is beyond the scope of this text. Our objective in this section is to demonstrate in a summarized form that the QR-RLS algorithm can be mapped in a systolic array. Further details regarding this subject can be found in references [2]-[5], [10].

A Givens rotation requires two basic steps. The first step is the calculation of the sine and cosine which are the elements of the rotation matrix. The second step is the application of the rotation matrix to a given data. Therefore, the basic computational elements required to perform the systolic array implementation of the QR-RLS algorithm introduced in the last section are the angle and the rotation processors shown in Fig. 8.1. The angle processor computes the cosine and sine, transferring the results to outputs 1 and 2 respectively, whereas in output 3 the cell delivers a partial product of cosines meant to generate the error signal as in equation (8.54). The rotation processor performs the rotation between the data coming from input 1 with the internal element of the matrix $\mathbf{U}(l)$ and transfers the result to output 3. This processor also updates the elements of $\mathbf{U}(l)$ and transfers the cosine and sine values to the neighboring cell on the left.

Now, imagine that we have the upper triangular matrix $\mathbf{U}(k)$ arranged below the row consisting of the new information data vector as in equation (8.31). Following the same pattern, we can arrange the basic cells in order to compute the

rotations of the QR-RLS algorithm as shown in Fig. 8.2, with the input signal $x(k)$ entering the array serially. In this figure, do not consider for this moment the time indexes and the left hand side column. The input data weighting is performed by the processors of the systolic array.

Basically, the computations corresponding to the triangularization of equation (8.31) are performed through the systolic array shown in Fig. 8.2, where at each instant of time an element of the matrix $\mathbf{U}(k)$ is stored in the basic processor as shown inside the building blocks. Note that these stored elements are skewed in time, and are initialized with zero. The left-hand cells store the elements of the vector $\underline{\mathbf{d}}(k)$ defined in equation (8.51), which are also initialized with zero and updated in each clock cycle. The column on the left hand side of the array performs the rotation and stores the rotated values of the desired signal vector which are essential to compute the error signal.

In order to allow the pipelining, the outputs of each cell are computed at the present clock period and made available to the neighboring cells in the following clock period. Note that the neighboring cells on the left and below a given cell are performing computations related to a previous iteration, whereas the cells on the right and above are performing the computations of one iteration in advance. This is the pipelining scheme of Fig. 8.2.

Each row of cells in the array implements a basic Givens rotation between one row of $\lambda\mathbf{U}(k-1)$ and a vector related to the new incoming data $\mathbf{x}(k)$. The top row of the systolic array performs the zeroing of the last element of the most recently incoming $\mathbf{x}(k)$ vector. The result of the rotation is then passed to the second row of the array. This second row performs the zeroing of the second last element in the rotated input signal. The zeroing processing continues in the following rows by eliminating the remaining elements of the intermediate vectors $\mathbf{x}'_i(k)$, defined in equation (8.38), through Givens rotations. The angle processors compute the rotation angles that are passed to each row to perform the rotations.

More specifically, returning to equation (8.31), at the instant k, the element $x(k-N)$ of $\mathbf{x}(k)$ is eliminated by calculating the angle $\theta_0(k)$ in the upper angle processor. The same processor also performs the computation of $u_{1,N+1}(k)$ that will be stored and saved for later elimination of $x(k-N+1)$, which occurs during the triangularization of $\underline{\mathbf{X}}(k+1)$. In the same period of time, the neighboring rotation processor performs the computation of $u_{1,N}(k-1)$, using the angle $\theta_0(k-1)$ that was received from the angle processor in the beginning of the present clock period k. In the first row of the array are performed the modifications to the first row of the $\mathbf{U}(k)$ matrix and to the vector $\underline{\mathbf{d}}(k)$

related to the desired signal, due to the rotation responsible for the elimination of $x(k-N)$. Note that the effect of the angle $\theta_0(k)$ in the remaining elements of the first row of $\mathbf{U}(k)$ will be felt only in the following iterations, one element each time, starting from the right to the left hand side.

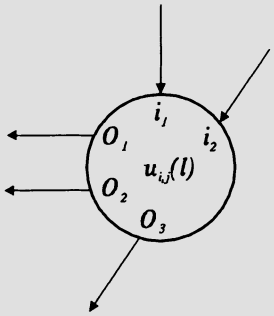
The second row of the systolic array is responsible for the rotation corresponding to $\theta_1(l)$ that eliminates the element $x_1'(l-N+1)$ of $\mathbf{x}_1'(l)$ defined in equation (8.38). The rotation $\theta_1(l)$ of course modifies the remaining nonzero elements of $\mathbf{x}_1'(l)$ generating $\mathbf{x}_2'(l)$, whose elements are calculated by the rotation processor and forwarded to the next row through output 3.

Likewise, the $(i+1)$th row performs the rotation $\theta_i(l)$ that eliminates $x_i'(l-N+i)$ and also the rotation in the vector $\underline{\mathbf{d}}(l)$.

In the bottom part of the systolic array, a product of $e_{q_1}(l)\gamma(l)$ is performed in each clock, in order to generate *a posteriori* output error given by $e(l)$. The output error obtained in a given sample period k corresponds to the error related to the input data vector of $2(N+1)$ clock periods before.

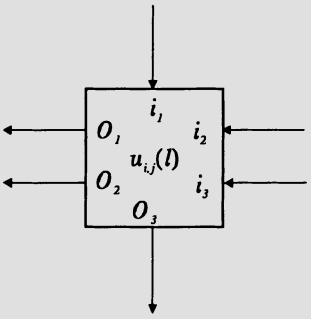
The systolic array of Fig. 8.2 exhibits several desirable features such as local interconnection, regularity, and simple control circuitry, that yields a simple implementation. A possible problem, as pointed out in [10], is the need to distribute a single clock throughout a large array, without incurring in clock skew.

The presented systolic array does not allow the computation of the tap-weight coefficients. A solution pointed out in [10] employs the array of Fig. 8.2, by freezing the array and applying an appropriate input signal sequence such that the tap-weight coefficients are made available at the array output $e(l)$. An alternative way is add a systolic array to solve the back-substitution problem [10]. The array is shown in Fig. 8.3 with the corresponding algorithm. The complete computation of the coefficient vector $\mathbf{w}(k)$ requires 2^{N+1} clock samples. In this array, the cells in the form of square produce the partial products involved in equation (8.11). The round cell performs the subtraction of the sum of product result with an element of the vector $\underline{\mathbf{d}}(k-8)$, namely $\underline{d}_{5-i}(k-8)$. This cell also performs the division of the subtraction result by the element $u_{N+1-i,i+1}(k-8)$ of the matrix $\mathbf{U}(k-8)$. Starting with $i=0$, the sum of products has no elements and as a consequence the round cell just performs the division $\frac{\underline{d}_{5-i}(k-8)}{u_{N+1-i,i+1}(k-8)}$. On the other hand, for $i=N$ all the square cells are actually taking part in the computation of the sum of products.



If i_1=0 then
$O_1 \leftarrow 1, O_2 \leftarrow 0, O_3 \leftarrow i_2$
$u_{i,j} = \lambda u_{i,j}$
Otherwise
$c \leftarrow \sqrt{\lambda^2 u_{i,j}^2(l) + i_1^2}$
$O_1 = \cos\theta_{i-1} \leftarrow \frac{\lambda u_{i,j}(l)}{c}$
$O_2 = \sin\theta_{i-1} \leftarrow \frac{i_1}{c}$
$O_3 \leftarrow i_2 O_1$
$u_{i,j}(l+1) \leftarrow c$
End

(a)



$O_1 \leftarrow i_2$
$O_2 \leftarrow i_3$
$O_3 \leftarrow i_1 i_2 - i_3 \lambda u_{i,j}(l)$
$u_{i,j}(l+1) \leftarrow i_1 i_3 + i_2 \lambda u_{i,j}(l)$

(b)

Figure 8.1 Basic cells: (a) Angle processor, (b) Rotation processor.

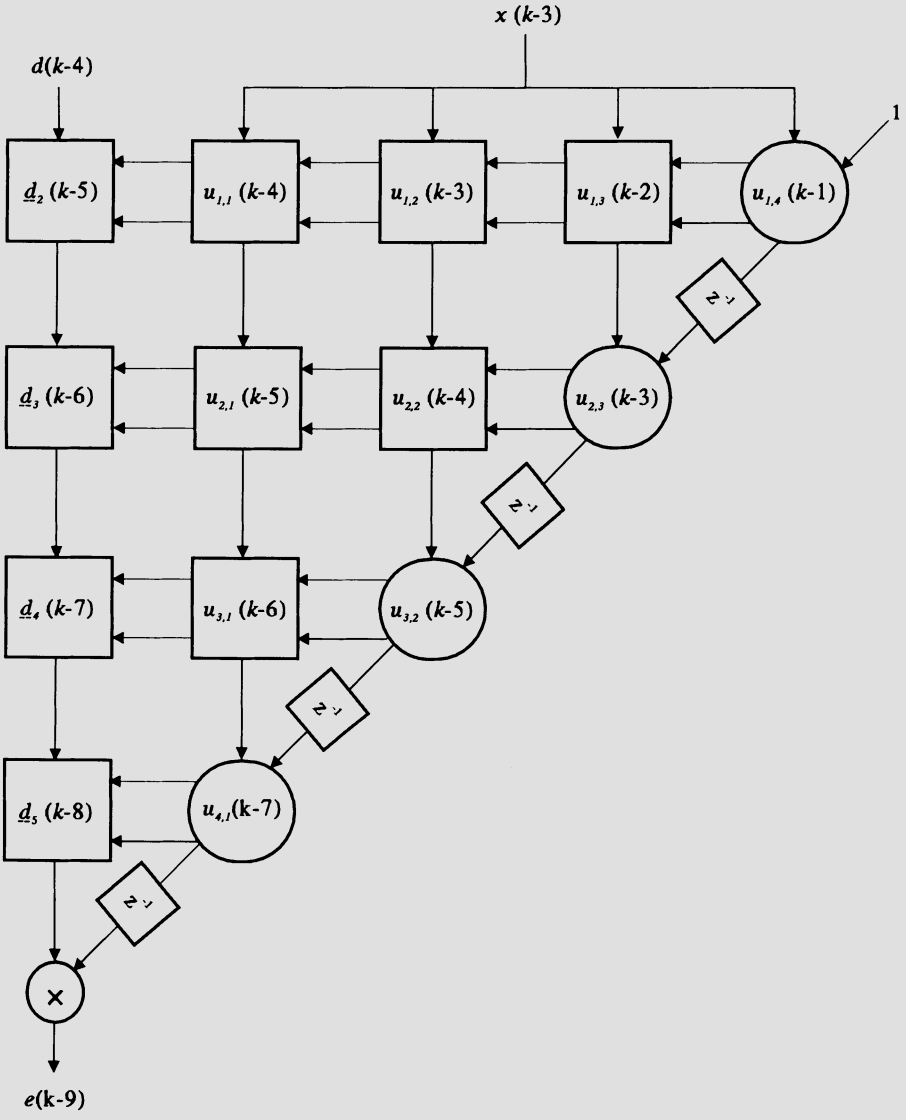
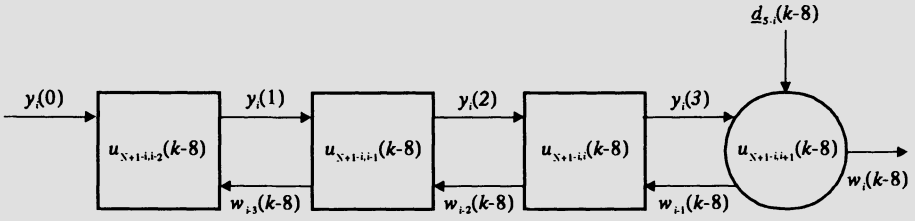


Figure 8.2 QR-Decomposition systolic array for N=3.

Note that in this case, in order to obtain $w_N(k-8)$, the results of all the cells starting from left to right must be ready, i.e., there is no pipelining involved.



$w_i = 0$ for $i < 0$
Do for $i = 0, 1, \ldots, N$
$y_i(N-i) = 0$
Do for l= $N - i + 1, \ldots, N$
$y_i(l) = y_i(l-1) + u_{N+1-i,i-N+l}(k-8)w_{i-N+l-1}(k-8)$
End

$$w_i(k-8) = \frac{\underline{d}_{5-i}(k-8) - y_i(3)}{u_{N+1-i,i+1}(k-8)}$$

End

Figure 8.3 Systolic array and algorithm for the computation of $\mathbf{w}(k)$.

Example 8.2

Let us choose a simple example, in order to illustrate how the systolic array implementation works, and compare the results with those belonging to the standard implementation of the QR-RLS algorithm. The chosen order is $N = 3$ and the forgetting factor is $\lambda = 0.99$.

Suppose that in an adaptive filtering environment, the input signal consists of

$$x(k) = \sin(\omega_0 k)$$

where $\omega_0 = \frac{\pi}{250}$.

The desired signal is generated by applying the same sinusoid to an FIR filter whose coefficients are given by

$$\mathbf{w}_o = [1.0\ 0.9\ 0.1\ 0.2]^T$$

Solution:

First consider the results obtained with the conventional QR-RLS algorithm. The contents of the vector $\underline{\mathbf{d}}(k)$ and of the matrix $\mathbf{U}(k)$ are given below for the first four iterations.

Iteration $k = 1$

$$\underline{\mathbf{d}}(k) = \begin{bmatrix} 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0126 \end{bmatrix} \quad \mathbf{U}(k) = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0126 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{8.55}$$

Iteration $k = 2$

$$\underline{\mathbf{d}}(k) = \begin{bmatrix} 0.0000 \\ 0.0000 \\ 0.0364 \\ 0.0125 \end{bmatrix} \quad \mathbf{U}(k) = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0251 & 0.0126 & 0.0000 & 0.0000 \\ 0.0125 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{8.56}$$

Iteration $k = 3$

$$\underline{\mathbf{d}}(k) = \begin{bmatrix} 0.0000 \\ 0.0616 \\ 0.0363 \\ 0.0124 \end{bmatrix} \quad \mathbf{U}(k) = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0377 & 0.0251 & 0.0126 & 0.0000 \\ 0.0250 & 0.0125 & 0.0000 & 0.0000 \\ 0.0124 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{8.57}$$

Iteration $k = 4$

$$\underline{\mathbf{d}}(k) = \begin{bmatrix} 0.0892 \\ 0.0613 \\ 0.0361 \\ 0.0124 \end{bmatrix} \quad \mathbf{U}(k) = \begin{bmatrix} 0.0502 & 0.0377 & 0.0251 & 0.0126 \\ 0.0375 & 0.0250 & 0.0125 & 0.0000 \\ 0.0249 & 0.0124 & 0.0000 & 0.0000 \\ 0.0124 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{8.58}$$

Iteration $k = 5$

$$\underline{\mathbf{d}}(k) = \begin{bmatrix} 0.1441 \\ 0.0668 \\ 0.0359 \\ 0.0123 \end{bmatrix} \quad \mathbf{U}(k) = \begin{bmatrix} 0.0785 & 0.0617 & 0.0449 & 0.0281 \\ 0.0409 & 0.0273 & 0.0136 & 0.0000 \\ 0.0248 & 0.0124 & 0.0000 & 0.0000 \\ 0.0123 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{8.59}$$

The data stored in the systolic array implementation represent the elements of the vector $\underline{\mathbf{d}}(k)$ and of the matrix $\mathbf{U}(k)$ skewed in time. These data are shown below starting from the the fourth iteration, since before that no data is available to the systolic array.

Observe when the elements of the $\mathbf{U}(k)$ appear stored at the systolic array. For example, the element $(4, 1)$ at instant $k = 4$ appears stored in the systolic array at instant $k = 10$, whereas the elements $(3, 1)$ and $(3, 2)$ at instant $k = 3$ appear stored in the systolic array at instants $k = 9$ and $k = 8$, respectively. Following the same line of thought, it is straightforward to understand how the remaining elements of the systolic array are calculated.

Iteration $k = 4$

$$\begin{bmatrix} 0. \\ 0. \\ 0. \\ 0. \end{bmatrix} \quad \begin{bmatrix} 0. & 0. & 0. & 0.0126 \\ 0. & 0. & 0. & \\ 0. & 0. & & \\ 0. & & & \end{bmatrix} \tag{8.60}$$

Iteration $k = 5$

$$\begin{bmatrix} 0. \\ 0. \\ 0. \\ 0. \end{bmatrix} \quad \begin{bmatrix} 0. & 0. & 0.0251 & 0.0281 \\ 0. & 0. & 0.0126 & \\ 0. & 0. & & \\ 0. & & & \end{bmatrix} \tag{8.61}$$

Iteration $k = 6$

$$\begin{bmatrix} 0. \\ 0. \\ 0. \\ 0. \end{bmatrix} \quad \begin{bmatrix} 0. & 0.0377 & 0.0449 & 0.0469 \\ 0. & 0.0251 & 0.0125 & \\ 0. & 0.0126 & & \\ 0. & & & \end{bmatrix} \tag{8.62}$$

Iteration $k = 7$

$$\begin{bmatrix} 0. \\ 0. \\ 0. \\ 0. \end{bmatrix} \quad \begin{bmatrix} 0.0502 & 0.0617 & 0.0670 & 0.0686 \\ 0.0377 & 0.0250 & 0.0136 & \\ 0.0251 & 0.0125 & & \\ 0.0126 & & & \end{bmatrix} \tag{8.63}$$

Iteration $k = 8$

$$\begin{bmatrix} 0.0892 \\ 0.0616 \\ 0.0364 \\ 0.0126 \end{bmatrix} \quad \begin{bmatrix} 0.0785 & 0.0870 & 0.0913 & 0.0927 \\ 0.0375 & 0.0273 & 0.0148 & \\ 0.0250 & 0.0124 & & \\ 0.0125 & & & \end{bmatrix} \tag{8.64}$$

Iteration $k = 9$

$$\begin{bmatrix} 0.1441 \\ 0.0613 \\ 0.0363 \\ 0.0125 \end{bmatrix} \quad \begin{bmatrix} 0.1070 & 0.1141 & 0.1179 & 0.1191 \\ 0.0409 & 0.0297 & 0.0160 & \\ 0.0249 & 0.0124 & & \\ 0.0124 & & & \end{bmatrix} \tag{8.65}$$

Iteration $k = 10$

$$\begin{bmatrix} 0.2014 \\ 0.0668 \\ 0.0361 \\ 0.0124 \end{bmatrix} \quad \begin{bmatrix} 0.1368 & 0.1430 & 0.1464 & 0.1475 \\ 0.0445 & 0.0319 & 0.0170 & \\ 0.0248 & 0.0123 & & \\ 0.0124 & & & \end{bmatrix} \tag{8.66}$$

Iteration $k = 11$

$$\begin{bmatrix} 0.2624 \\ 0.0726 \\ 0.0359 \\ 0.0124 \end{bmatrix} \quad \begin{bmatrix} 0.1681 & 0.1737 & 0.1768 & 0.1778 \\ 0.0479 & 0.0340 & 0.0180 & \\ 0.0246 & 0.0123 & & \\ 0.0123 & & & \end{bmatrix} \tag{8.67}$$

It is a good exercise for the reader to examine the elements of the vectors and matrices in equations (8.60)-(8.67) and detect when these elements appear in the corresponding vectors $\underline{\mathbf{d}}(k)$ and matrices $\mathbf{U}(k)$ of equations (8.55)-(8.59).

□

8.4 SOME IMPLEMENTATION ISSUES

Several articles related to implementation issues of the QR-RLS algorithm such as the elimination of square root computation [12], stability and quantization error analyses [13]-[16] are available in the open literature. In this section, some of these results are briefly reviewed.

The stability of the QR-RLS algorithm is the first issue to be concerned when considering a real implementation. Fortunately, the QR-RLS algorithm implemented in finite precision was proved stable in the bounded input/bounded output sense in [14]. The proof was based on the analysis of the bounds for the internal recursions of the algorithm [14]-[15]. From another study based on the quantization-error propagation in the finite-precision implementation of the QR-RLS algorithm, it was possible to derive the error recursions for the main quantities of the algorithm, leading to the stability conditions of the QR-RLS algorithm [16]. The convergence in average of the QR-RLS algorithm can be guaranteed if the following inequality is satisfied [16]

$$\lambda^{1/2} \parallel \tilde{\mathbf{Q}}_Q(k) \parallel_2 \leq 1 \tag{8.68}$$

where the two norm $\| \cdot \|_2$ of a matrix used here is the square root of the largest eigenvalue and the notation $\cdot_Q$ denotes finite-precision version of $\cdot$. Therefore,

$$\parallel \tilde{\mathbf{Q}}_Q(k) \parallel_2 = MAX_i \sqrt{\cos_Q^2 \theta_i(k) + \sin_Q^2 \theta_i(k)} \tag{8.69}$$

where $MAX_i[\cdot]$ is the maximum value of $[\cdot]$. The stability condition can be rewritten as follows:

$$\lambda \leq \frac{1}{MAX_i \left[\cos_Q^2 \theta_i(k) + \sin_Q^2 \theta_i(k)\right]} \tag{8.70}$$

It can then be concluded that keeping the product of the forgetting factor by the maximum eigenvalue of the Givens rotations smaller than unity is a sufficient condition to guarantee the stability.

For the implementation of any adaptive algorithm it is necessary to estimate quantitatively the dynamic range of all internal variables of the algorithm in order to determine the length of all the registers in the actual implementation. Although this issue should be considered in the implementation of any adaptive filtering algorithm, it is particularly relevant in the QR-RLS algorithms due to their large number of internal variables. The first attempt to address this problem was reported in [15], where expressions for the steady-state values of

the cosines and sines of the Givens rotations were determined, as well as the bounds for the dynamic range of the informations stored in the processing cells. The full quantitative analysis of the dynamic range of all internal quantities of the QR-RLS algorithm was presented in [16] for the conventional and systolic-array forms. For fixed-point implementation, it is important to determine the internal signal with the largest energy such that frequent overflow in the internal variables of the QR-RLS algorithm can be avoided. The first entry of the triangularized information matrix can be shown to have the largest energy [16] and its steady-state value is approximately

$$u_{0,0}(k) \approx \frac{\sigma_x}{\sqrt{1-\lambda}} \tag{8.71}$$

where σ_x^2 is the variance of the input signal.

The procedure to derive the results discussed above consists of first analyzing the QR-RLS algorithm for ideal infinite-precision implementation. The second step is modeling the quantization errors and deriving the recursive equations that include the overall error in each quantity of the QR-RLS algorithm [16]. Then conditions to guarantee the stability of the algorithm could be derived. A further step is to derive closed-form solutions to the mean-squared values of the deviations in the internal variables of the algorithm due to finite-precision operations. The main objective in this step is to obtain the excess of mean-square error and the variance of the deviation in the tap-weight coefficients. Analytical expressions for these quantities are not very simple unless a number of assumptions about the input and reference signals are assumed [16], however they are useful to the designer.

8.5 FAST QR-RLS ALGORITHM

For the derivation of the fast QR-RLS algorithms, it is first necessary to study the solutions of the forward and backward prediction problems. As seen in the last two chapters, the predictor solutions were also required in the derivation of the lattice-based and the fast transversal RLS algorithms. We start with the application of the conventional QR-RLS algorithm to the forward prediction problem.

8.5.1 Forward Prediction Problem

In the forward prediction problem, the following relations are valid

$$d_f(k) = x(k+1) \tag{8.72}$$

$$\mathbf{d}_f(k) = \begin{bmatrix} x(k+1) \\ \lambda^{1/2}x(k) \\ \vdots \\ \lambda^{\frac{k+1}{2}}x(0) \end{bmatrix} \tag{8.73}$$

$$\mathbf{e}_f(k) = \mathbf{d}_f(k) - \begin{bmatrix} \underline{\mathbf{X}}(k) \\ \mathbf{0} \end{bmatrix} \mathbf{w}_f(k) \tag{8.74}$$

where $d_f(k)$ is the desired signal, $\mathbf{d}_f(k)$ is the desired signal vector, and $\mathbf{e}_f(k)$ is the error signal vector, all for the forward prediction problem.

Now, we can consider applying the QR decomposition as previously done in equation (8.44) to the forward prediction error defined above. The reader should note that in the present case an extra row was added to the vectors $\mathbf{e}_f(k)$ and $\mathbf{d}_f(k)$, as can be verified in the following relations

$$\begin{bmatrix} \mathbf{Q}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{e}_f(k) = \begin{bmatrix} \mathbf{Q}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{d}_f(k) - \begin{bmatrix} \mathbf{0} \\ \mathbf{U}(k) \\ \mathbf{0} \end{bmatrix} \mathbf{w}_f(k) \tag{8.75}$$

$$\mathbf{e}_{fq}(k) = \mathbf{d}_{fq}(k) - \begin{bmatrix} \mathbf{0} \\ \mathbf{U}(k) \\ \mathbf{0} \end{bmatrix} \mathbf{w}_f(k) \tag{8.76}$$

where

$$\underbrace{\begin{bmatrix} \mathbf{0} \\ \mathbf{U}(k) \\ \mathbf{0} \end{bmatrix}}_{N+1} \begin{matrix} \}k-N \\ \}N+1 \\ \}1 \end{matrix}$$

and

$$\mathbf{e}_{fq}(k) = \begin{bmatrix} e_{fq_1}(k) \\ e_{fq_2}(k) \\ \vdots \\ e_{fq_{k-N}}(k) \\ e_{fq_{k-N+1}}(k) \\ \vdots \\ e_{fq_{k+1}}(k) \\ \lambda^{\frac{k+1}{2}} x(0) \end{bmatrix} \qquad \mathbf{d}_{fq}(k) = \begin{bmatrix} d_{fq_1}(k) \\ d_{fq_2}(k) \\ \vdots \\ d_{fq_{k-N}}(k) \\ d_{fq_{k-N+1}}(k) \\ \vdots \\ d_{fq_{k+1}}(k) \\ \lambda^{\frac{k+1}{2}} x(0) \end{bmatrix} \tag{8.77}$$

The $\mathbf{Q}(k)$ matrix here is the same used in the standard QR-decomposition algorithm, since its objective is to triangularize $\underline{\mathbf{X}}(k)$. With the elements of $\mathbf{w}_f(k)$ set to their optimum values at the instant k, the vector $\mathbf{e}_{fq}(k)$ above becomes

$$\mathbf{e}_{fq}(k) = \begin{bmatrix} e_{fq_1}(k) \\ \vdots \\ e_{fq_{k-N}}(k) \\ 0 \\ \vdots \\ 0 \\ \lambda^{\frac{k+1}{2}} x(0) \end{bmatrix} \tag{8.78}$$

As in equation (8.48), equation (8.74) can be rewritten as

$$\mathbf{e}_f(k) = \left[\begin{array}{c|c} \mathbf{d}_f(k) & \underline{\mathbf{X}}(k) \\ & \\ & \mathbf{0} \end{array}\right] \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix} \tag{8.79}$$

and

$$\begin{aligned} \mathbf{e}_{fq}(k) &= \begin{bmatrix} \mathbf{Q}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \left[\begin{array}{c|c} \mathbf{d}_f(k) & \underline{\mathbf{X}}(k) \\ & \\ & \mathbf{0} \end{array}\right] \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix} \\ &= \begin{bmatrix} e_{fq_1}(k) & \\ \vdots & \mathbf{0} \\ e_{fq_{k-N}}(k) & \\ \mathbf{x}_{q_2}(k) & \mathbf{U}(k) \\ \lambda^{\frac{k+1}{2}} x(0) & \mathbf{0} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix} \end{aligned} \tag{8.80}$$

where $\mathbf{x}_{q_2}(k)$ is the equivalent of $\mathbf{d}_{q_2}(k)$ in equation (8.49).

Note that:

$$\left[\begin{array}{c|c} \mathbf{d}_f(k) & \begin{array}{c}\underline{\mathbf{X}}(k)\\ \\ \mathbf{0}\end{array} \end{array}\right] = \underline{\mathbf{X}}^{(N+2)}(k+1) \tag{8.81}$$

which is an order extended version of $\underline{\mathbf{X}}(k+1)$ and has dimension $(k+2)$ by $(N+2)$.

In order to recursively solve equation (8.80) without dealing with ever increasing matrices, a set of Givens rotations are applied in order to eliminate $e_{fq_1}(k)$, $e_{fq_2}(k)$, $\ldots e_{fq_{k-N}}(k)$, such that the information matrix that premultiplies the vector $[1 \quad -\mathbf{w}_f^T(k)]^T$ is triangularized. The Givens rotations can recursively be obtained by

$$\begin{aligned} \mathbf{Q}_f(k) &= \tilde{\mathbf{Q}}_f(k)\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_f(k-1)\end{bmatrix} \\ &= \tilde{\mathbf{Q}}_f(k)\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Q}}_f(k-1)\end{bmatrix}\cdots\begin{bmatrix} \mathbf{I}_{k-N-1} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Q}}_f(N+1)\end{bmatrix} \end{aligned} \tag{8.82}$$

where $\tilde{\mathbf{Q}}_f(k)$ is defined below. The time indexes employed in $\mathbf{Q}_f(k)$ and $\tilde{\mathbf{Q}}_f(k)$ are clarified along the following derivations.

$$\tilde{\mathbf{Q}}_f(k) = \begin{bmatrix} \cos\theta_f(k) & \cdots & 0 & \cdots & -\sin\theta_f(k) \\ \vdots & & & & \vdots \\ 0 & & \mathbf{I}_k & & 0 \\ \vdots & & & & \vdots \\ \sin\theta_f(k) & \cdots & 0 & \cdots & \cos\theta_f(k) \end{bmatrix} \tag{8.83}$$

The matrix $\mathbf{Q}_f(k)$ has the following general form

$$\mathbf{Q}_f(k) = \begin{bmatrix} * & & & & & & * \\ & \ddots & & & \mathbf{0} & & \vdots \\ & & * & & & & * \\ & & & & & & 0 \\ & \mathbf{0} & & & \mathbf{I}_{N+1} & & \vdots \\ & & & & & & \vdots \\ * & \cdots & * & 0 & \cdots & 0 & \cdots & * \end{bmatrix} \tag{8.84}$$

$$\underbrace{\qquad\qquad}_{k-N}$$

where $*$ denotes nonzero elements.

If in each iteration the rotation above is applied to equation (8.80), we have

$$\begin{aligned}
\mathbf{e}'_{fq}(k) &= \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_f(k-1) \end{bmatrix} \begin{bmatrix} e_{fq_1}(k) & \\ \vdots & \mathbf{0} \\ e_{fq_{k-N}}(k) & \\ \mathbf{x}_{q_2}(k) & \mathbf{U}(k) \\ \lambda^{\frac{k+1}{2}} x(0) & \mathbf{0} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix} \\
&= \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} e_{fq_1}(k) & \\ 0 & \mathbf{0} \\ \vdots & \\ 0 & \\ \mathbf{x}_{q_2}(k) & \mathbf{U}(k) \\ \lambda^{1/2}||\mathbf{e}_f(k-1)|| & \mathbf{0} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix} \\
&= \begin{bmatrix} 0 & \\ \vdots & \mathbf{0} \\ 0 & \\ \mathbf{x}_{q_2}(k) & \mathbf{U}(k) \\ ||\mathbf{e}_f(k)|| & \mathbf{0} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix} \qquad (8.85)
\end{aligned}$$

where

$$\cos\theta_f(k) = \frac{\lambda^{1/2}||\mathbf{e}_f(k-1)||}{\sqrt{\lambda||\mathbf{e}_f(k-1)||^2 + e^2_{fq_1}(k)}} \qquad (8.86)$$

$$\sin\theta_f(k) = \frac{e_{fq_1}(k)}{\sqrt{\lambda||\mathbf{e}_f(k-1)||^2 + e^2_{fq_1}(k)}} \qquad (8.87)$$

and $||\mathbf{e}_f(k)||$ is the norm of the forward-prediction-error vector shown in equation (8.74). In order to show that the last element of $\mathbf{e}'_{fq}(k)$ is equal to $||\mathbf{e}_f(k)||$, first note that the $||\mathbf{e}'_{fq}(k)|| = ||\mathbf{e}_{fq}(k)|| = ||\mathbf{e}_f(k)||$, since these error vectors are related through unitary transformations. Also, by induction, it can easily be shown from equation (8.85) that:

For $k = 0, 1, \ldots N$

$$||\mathbf{e}_f(k)|| = \lambda^{\frac{k+1}{2}} x(0)$$

for $k = N+1$

$$||\mathbf{e}'_{fq}(k)|| = ||\mathbf{e}_f(k)|| = \sqrt{\lambda^{k+1} x^2(0) + e^2_{fq_1}(k)}$$

for $k = N + 2$

$$\begin{aligned} ||\mathbf{e}_f(k)|| &= \sqrt{\lambda^{k+1}x^2(0) + \lambda e_{fq_1}^2(k-1) + e_{fq_1}^2(k)} \\ &= \sqrt{\lambda||\mathbf{e}_f(k-1)||^2 + e_{fq_1}^2(k)} \end{aligned}$$

for $k > N + 2$

$$||\mathbf{e}_f(k)||^2 = \lambda||\mathbf{e}_f(k-1)||^2 + e_{fq_1}^2(k) \tag{8.88}$$

By combining the rotations aimed to triangularize $\underline{\mathbf{X}}(k)$ and to eliminate $e_{fq_1}(k)$, it follows that

$$\begin{aligned} &\mathbf{Q}_f(k) \begin{bmatrix} \mathbf{Q}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \\ &= \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_f(k-1) \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{Q}}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{Q}(k-1) & \mathbf{0} \\ 0 & \mathbf{0} & 1 \end{bmatrix} \end{aligned} \tag{8.89}$$

From the general forms of $\tilde{\mathbf{Q}}(k)$ and $\mathbf{Q}_f(k)$ shown in equations (8.30) and (8.84), respectively, one can conclude that the following matrix product can be commuted.

$$\begin{aligned} &\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_f(k-1) \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{Q}}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \tilde{\mathbf{Q}}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_f(k-1) \end{bmatrix} \end{aligned} \tag{8.90}$$

Because of that, it can be assumed that the overall triangularization can be performed at each iteration as follows:

$$\begin{bmatrix} 0 & \\ 0 & \mathbf{0} \\ \vdots & \\ 0 & \\ \mathbf{x}_{q_2}(k) & \mathbf{U}(k) \\ ||\mathbf{e}_f(k)|| & \mathbf{0} \end{bmatrix}$$

$$= \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} \tilde{\mathbf{Q}}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x(k+1) & \mathbf{x}^T(k) \\ \mathbf{0} & \mathbf{0} \\ \lambda^{1/2}\mathbf{x}_{q_2}(k-1) & \lambda^{1/2}\mathbf{U}(k-1) \\ \lambda^{1/2}\|\mathbf{e}_f(k-1)\| & \mathbf{0} \end{bmatrix} \tag{8.91}$$

Since the rotations at instant k are actually completing the triangularization of $\underline{\mathbf{X}}^{(N+2)}(k+1)$, we can conclude that

$$\tilde{\mathbf{Q}}^{(N+2)}(k+1) = \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} \tilde{\mathbf{Q}}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \tag{8.92}$$

Also, the complete rotation matrix can be expressed as

$$\begin{aligned} &\mathbf{Q}^{(N+2)}(k+1) = \\ &\tilde{\mathbf{Q}}_f(k) \begin{bmatrix} \tilde{\mathbf{Q}}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_f(k-1) \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{Q}(k-1) & \mathbf{0} \\ 0 & \mathbf{0} & 1 \end{bmatrix} \\ &= \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} \tilde{\mathbf{Q}}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}^{(N+2)}(k) \end{bmatrix} \end{aligned} \tag{8.93}$$

Where the superscript $(N+2)$ in some matrices above denote rotation matrices applied to data with $(N+2)$ elements. With the relations above, it is possible to generate $\tilde{\mathbf{Q}}^{(N+2)}(k+1)$ from $\tilde{\mathbf{Q}}(k)$ through a simple rotation matrix $\tilde{\mathbf{Q}}_f(k)$. Now, we will show that by developing a similar QR-decomposition algorithm for the backward prediction case it is possible to generate $\tilde{\mathbf{Q}}(k+1)$ from $\tilde{\mathbf{Q}}^{(N+2)}(k+1)$.

In the matrix description above, the dimensions of the matrices are growing with the iterations. In the actual implementation of the fast QR-RLS algorithm, the $(N+2)$ by $(N+2)$ reduced version of $\tilde{\mathbf{Q}}(k)$, denoted by $\mathbf{Q}_\theta(k)$, is employed. Obviously the matrix $\mathbf{Q}_\theta(k)$ is implemented as a product of Givens rotation matrices. Also, in the fast version of the QR-RLS algorithm a reduced version

of $\tilde{\mathbf{Q}}_f(k)$ is required that is given by

$$\mathbf{Q}_{\theta f}(k) = \begin{bmatrix} \cos\theta_f(k) & 0 & \cdots & 0 & \cdots & 0 & -\sin\theta_f(k) \\ 0 & & & & & & 0 \\ \vdots & & & & & & \vdots \\ 0 & & & \mathbf{I}_{N+1} & & & 0 \\ \vdots & & & & & & \vdots \\ 0 & & & & & & 0 \\ \sin\theta_f(k) & 0 & \cdots & 0 & \cdots & 0 & \cos\theta_f(k) \end{bmatrix} \tag{8.94}$$

The actual relation that will be used in the fast algorithm is given by

$$\mathbf{Q}_{\theta}^{(N+2)}(k+1) = \mathbf{Q}_{\theta f}(k) \begin{bmatrix} \mathbf{Q}_{\theta}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \tag{8.95}$$

In the following subsection, we study the application of the conventional QR-RLS algorithm to the backward prediction problem.

8.5.2 Backward Prediction Problem

In the backward prediction problem the desired signal and vector are respectively

$$d_b(k+1) = x(k-N) \tag{8.96}$$

$$\mathbf{d}_b(k+1) = \begin{bmatrix} x(k-N) \\ \lambda^{1/2}x(k-N-1) \\ \vdots \\ \lambda^{\frac{k-N}{2}}x(0) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{8.97}$$

The dimension of $\mathbf{d}_b(k+1)$ is $(k+2)$ by 1. The backward-prediction-error vector is given by

$$\begin{aligned} \mathbf{e}_b(k+1) &= \mathbf{d}_b(k+1) - \underline{\mathbf{X}}(k+1)\mathbf{w}_b(k+1) \\ &= [\underline{\mathbf{X}}(k+1)\ \mathbf{d}_b(k+1)] \begin{bmatrix} -\mathbf{w}_b(k+1) \\ 1 \end{bmatrix} \end{aligned} \tag{8.98}$$

The triangularization matrix $\mathbf{Q}(k+1)$ of the input data matrix can be applied to the backward prediction error above resulting in

$$\mathbf{Q}(k+1)\mathbf{e}_b(k+1) = \mathbf{Q}(k+1)\mathbf{d}_b(k+1) - \begin{bmatrix} \mathbf{0} \\ \mathbf{U}(k+1) \end{bmatrix} \mathbf{w}_b(k+1) \quad (8.99)$$

or equivalently

$$\mathbf{e}_{bq}(k+1) = \mathbf{d}_{bq}(k+1) - \begin{bmatrix} \mathbf{0} \\ \mathbf{U}(k+1) \end{bmatrix} \mathbf{w}_b(k+1) \quad (8.100)$$

where

$$\mathbf{e}_{bq}(k+1) = \begin{bmatrix} e_{bq_1}(k+1) \\ e_{bq_2}(k+1) \\ \vdots \\ \vdots \\ e_{bq_{k+2}}(k+1) \end{bmatrix}$$

$$\mathbf{d}_{bq}(k+1) = \begin{bmatrix} d_{bq_1}(k+1) \\ d_{bq_2}(k+1) \\ \vdots \\ d_{bq_{k-N+1}}(k+1) \\ \hline d_{bq_{k-N+2}}(k+1) \\ \vdots \\ d_{bq_{k+2}}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{q_1}(k+1) \\ \hline \mathbf{x}_{q_3}(k+1) \end{bmatrix} \quad (8.101)$$

With $\mathbf{w}_b(k+1)$ set to its optimum instantaneous value, the vector $\mathbf{e}_{bq}(k+1)$ has the following form

$$\mathbf{e}_{bq}(k+1) = \begin{bmatrix} e_{bq_1}(k+1) \\ e_{bq_2}(k+1) \\ \vdots \\ e_{bq_{k-N+1}}(k+1) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (8.102)$$

In this case, equation (8.100) can be rewritten as

$$\mathbf{e}_{bq}(k+1) = \mathbf{Q}(k+1)[\underline{\mathbf{X}}(k+1)\ \mathbf{d}_b(k+1)] \begin{bmatrix} -\mathbf{w}_b(k+1) \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} & e_{bq_1}(k+1) \\ \mathbf{0} & e_{bq_2}(k+1) \\ & \vdots \\ & e_{bq_{k-N+1}}(k+1) \\ \mathbf{U}(k+1) & \mathbf{x}_{q_3}(k+1) \end{bmatrix} \begin{bmatrix} -\mathbf{w}_b(k+1) \\ 1 \end{bmatrix} \tag{8.103}$$

where $\mathbf{x}_{q_3}(k+1)$ is the equivalent of $\mathbf{d}_{q_2}(k)$ in equation (8.49).

Also note that

$$[\underline{\mathbf{X}}(k+1)\ \mathbf{d}_b(k+1)] = \underline{\mathbf{X}}^{(N+2)}(k+1) \tag{8.104}$$

where $\underline{\mathbf{X}}^{(N+2)}(k+1)$ is an extended version of $\underline{\mathbf{X}}(k+1)$, with one input signal information vector added. In other words, $\underline{\mathbf{X}}^{(N+2)}(k+1)$ is the information matrix that would be obtained if one delay was added at the end of the delay line.

In order to avoid increasing vectors in the algorithm, the quantities $e_{bq_1}(k+1), e_{bq_2}(k+1), \ldots, e_{bq_{k-N}}(k+1)$, can be eliminated in equation (8.103) through Givens rotations, as follows:

$$\begin{aligned} & \mathbf{Q}_b(k+1)\mathbf{e}_{bq}(k+1) \\ &= \mathbf{Q}_b(k+1) \begin{bmatrix} & e_{bq_1}(k+1) \\ \mathbf{0} & e_{bq_2}(k+1) \\ & \vdots \\ & e_{bq_{k-N+1}}(k+1) \\ \mathbf{U}(k+1) & \mathbf{x}_{q_3}(k+1) \end{bmatrix} \begin{bmatrix} -\mathbf{w}_b(k+1) \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ & ||\mathbf{e}_b(k+1)|| \\ \mathbf{U}(k+1) & \mathbf{x}_{q_3}(k+1) \end{bmatrix} \begin{bmatrix} -\mathbf{w}_b(k+1) \\ 1 \end{bmatrix} \end{aligned} \tag{8.105}$$

The matrix $\mathbf{Q}_b(k+1)$ can be generated iteratively as follows:

$$\begin{aligned} \mathbf{Q}_b(k+1) &= \tilde{\mathbf{Q}}_b(k+1) \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_b(k) \end{bmatrix} \\ &= \tilde{\mathbf{Q}}_b(k+1) \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Q}}_b(k) \end{bmatrix} \cdots \begin{bmatrix} \mathbf{I}_{k-N-1} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Q}}_b(N+2) \end{bmatrix} \end{aligned} \tag{8.106}$$

where $\tilde{\mathbf{Q}}_b(k+1)$ is defined by

$$\tilde{\mathbf{Q}}_b(k+1)$$

$$= \begin{bmatrix} \cos\theta_b(k+1) & 0 & \cdots & 0 & \cdots & 0 & -\sin\theta_b(k+1) & 0 & \cdots \\ 0 & & & & & & & & \\ \vdots & & & \mathbf{I}_{k-N-1} & & & \mathbf{0} & & \\ \sin\theta_b(k+1) & 0 & \cdots & 0 & \cdots & 0 & \cos\theta_b(k+1) & 0 & \cdots \\ & & & \mathbf{0} & & & & & \mathbf{I}_{N+1} \end{bmatrix} \tag{8.107}$$

The matrix $\mathbf{Q}_b(k+1)$ as a consequence has the following general form

$$\mathbf{Q}_b(k+1) = \begin{bmatrix} * & * & \cdots & * & \mathbf{0} \\ 0 & * & \cdots & * & \mathbf{0} \\ \vdots & \mathbf{0} & \ddots & \vdots & \mathbf{0} \\ * & * & \cdots & * & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{N+1} \end{bmatrix} \tag{8.108}$$

$$\underbrace{\qquad\qquad\qquad\qquad}_{k-N+1}$$

where $*$ denotes possibly nonzero elements.

The Givens rotations included in $\mathbf{Q}_b(k+1)$ when applied to $\mathbf{e}_{bq}(k+1)$ result in

$$\begin{aligned} \mathbf{e}'_{bq}(k+1) &= \mathbf{Q}_b(k+1)\mathbf{e}_{bq}(k+1) \\ &= \begin{bmatrix} & & 0 \\ & \mathbf{0} & 0 \\ & & \vdots \\ & & \|\mathbf{e}_b(k+1)\| \\ \mathbf{U}(k+1) & & \mathbf{x}_{q_3}(k+1) \end{bmatrix} \begin{bmatrix} -\mathbf{w}_b(k+1) \\ 1 \end{bmatrix} \end{aligned} \tag{8.109}$$

Since $\mathbf{Q}(k+1)$ and $\mathbf{Q}_b(k+1)$ are orthogonal matrices, the first nonzero element in the last column of the matrix in equation (8.109) is the norm of the backward-estimation-error vector at the instant $(k+1)$.

The rotation angle for the backward prediction matrix $\tilde{\mathbf{Q}}_b(k+1)$ can be calculated recursively as follows:

$$\cos\theta_b(k+1) = \frac{\lambda^{1/2}\|\mathbf{e}_b(k)\|}{\sqrt{\lambda\|\mathbf{e}_b(k)\|^2 + e^2_{bq_1}(k+1)}} \tag{8.110}$$

$$\sin\theta_b(k+1) = \frac{e_{bq_1}(k+1)}{\sqrt{\lambda\|\mathbf{e}_b(k)\|^2 + e^2_{bq_1}(k+1)}} \tag{8.111}$$

The partial triangularization process involved in equation (8.99) followed by that of equation (8.109) can be performed step by step every time a new row is added to the information data matrix. The validation of the step by step updating, involving the matrices $\mathbf{Q}_b(k+1)$ and $\mathbf{Q}(k+1)$, and the corresponding angle pertaining to $\tilde{\mathbf{Q}}_b(k+1)$, can be shown by noting that the following matrix product commute:

$$\begin{aligned}\mathbf{Q}_b(k+1)\mathbf{Q}(k+1) &= \tilde{\mathbf{Q}}_b(k+1)\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_b(k) \end{bmatrix}\tilde{\mathbf{Q}}(k+1)\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}(k) \end{bmatrix} \\ &= \tilde{\mathbf{Q}}_b(k+1)\tilde{\mathbf{Q}}(k+1)\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_b(k) \end{bmatrix}\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}(k) \end{bmatrix}\end{aligned} \tag{8.112}$$

The matrix $\underline{\mathbf{X}}^{(N+2)}(k+1)$ that is an argument of $\mathbf{e}_b(k+1)$ in equation (8.98), can be fully triangularized if an additional orthogonal matrix $\mathbf{Q}'_b(k+1)$ is premultiplied to equation (8.109) in order eliminate the vector $\mathbf{x}_{q_3}(k+1)$ as follows:

$$\begin{aligned}\mathbf{e}''_{bq}(k+1) &= \mathbf{Q}'_b(k+1)\mathbf{Q}_b(k+1)\mathbf{Q}(k+1)\mathbf{e}_b(k+1) \\ &= \begin{bmatrix} \mathbf{0} \\ \mathbf{U}^{(N+2)}(k+1) \end{bmatrix}\begin{bmatrix} -\mathbf{w}_b(k+1) \\ 1 \end{bmatrix}\end{aligned} \tag{8.113}$$

where the lower part of the argument matrix is nothing but $\mathbf{U}^{(N+2)}(k+1)$, since it is the result of an orthogonal triangularization of $\underline{\mathbf{X}}^{(N+2)}(k+1)$.

The matrix $\mathbf{Q}'_b(k+1)$ consists of a series of rotations as follows:

$$\mathbf{Q}'_b(k+1) = \begin{bmatrix} \mathbf{I}_{k-N} & & \mathbf{0} & \mathbf{0} \\ & \cos\theta'_{b_1}(k+1) & \sin\theta'_{b_1}(k+1) & \\ & -\sin\theta'_{b_1}(k+1) & \cos\theta'_{b_1}(k+1) & \\ \mathbf{0} & & \mathbf{0} & \mathbf{I}_N \end{bmatrix}$$

$$\cdots \begin{bmatrix} \mathbf{I}_{k-N} & & & & \mathbf{0} & & & & 0 \\ & \cos\theta'_{b_N}(k+1) & 0 & \cdots & 0 & \cdots & 0 & \sin\theta'_{b_N}(k+1) & \vdots \\ & 0 & & & & & & 0 & \vdots \\ \mathbf{0} & \vdots & & & \mathbf{I}_{N-1} & & & \vdots & \vdots \\ & -\sin\theta'_{b_N}(k+1) & 0 & \cdots & 0 & \cdots & 0 & \cos\theta'_{b_N}(k+1) & \vdots \\ & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 1 \end{bmatrix}$$

$$\cdot \begin{bmatrix} \mathbf{I}_{k-N} & & & & \mathbf{0} & & & \\ & \cos\theta'_{b_{N+1}}(k+1) & 0 & \cdots & 0 & \cdots & 0 & \sin\theta'_{b_{N+1}}(k+1) \\ & 0 & & & & & & 0 \\ \mathbf{0} & \vdots & & & \mathbf{I}_N & & & \vdots \\ & \vdots & & & & & & \vdots \\ & -\sin\theta'_{b_{N+1}}(k+1) & 0 & \cdots & 0 & \cdots & 0 & \cos\theta'_{b_{N+1}}(k+1) \end{bmatrix} \tag{8.114}$$

where

$$\begin{aligned} \cos\theta'_{bi}(k+1) &= \frac{\mu_{i-1}}{\sqrt{\mu_{i-1}^2 + x_{q_3 i}^2(k+1)}} \\ \sin\theta'_{bi}(k+1) &= \frac{x_{q_3 i}(k+1)}{\sqrt{\mu_{i-1}^2 + x_{q_3 i}^2(k+1)}} \end{aligned} \tag{8.115}$$

$\mu_0 = ||\mathrm{e}_b(k+1)||$, and $\mu_{N+1} = [\mathbf{U}^{(N+2)}(k+1)]_{1,N+2}$. The quantities $x_{q_3 i}(k+1)$ denote the ith component of the vector $\mathbf{x}_{q_3}(k+1)$. Note that the required elements to compute these angles are not available, then an alternative strategy is required.

Since the matrix $\underline{\mathbf{X}}^{(N+2)}(k+1)$ is triangularized by $\mathbf{Q}^{(N+2)}(k+1)$, it can be concluded that

$$\begin{aligned} \mathbf{Q}^{(N+2)}(k+1) &= \mathbf{Q}'_b(k+1)\mathbf{Q}_b(k+1)\mathbf{Q}(k+1) \\ &= \mathbf{Q}'_b(k+1)\mathbf{Q}_b(k+1)\tilde{\mathbf{Q}}(k+1) \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}(k) \end{bmatrix} \end{aligned} \tag{8.116}$$

From the commutation properties of equation (8.112)

$$\mathbf{Q}^{(N+2)}(k+1) = \mathbf{Q}'_b(k+1)\tilde{\mathbf{Q}}_b(k+1)\tilde{\mathbf{Q}}(k+1) \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_b(k) \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}(k) \end{bmatrix} \tag{8.117}$$

In the instant k, the following relation is valid

$$\mathbf{Q}^{(N+2)}(k) = \mathbf{Q}'_b(k)\mathbf{Q}_b(k)\mathbf{Q}(k) \tag{8.118}$$

also, we know that

$$\mathbf{Q}^{(N+2)}(k+1) = \tilde{\mathbf{Q}}^{(N+2)}(k+1) \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}^{(N+2)}(k) \end{bmatrix} \tag{8.119}$$

From equations (8.116), (8.118), and (8.119), it can be concluded that

$$\tilde{\mathbf{Q}}^{(N+2)}(k+1)\begin{bmatrix} 1 & 0 \\ 0 & \mathbf{Q}_b'(k) \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & \mathbf{Q}_b(k) \end{bmatrix} = \mathbf{Q}_b'(k+1)\mathbf{Q}_b(k+1)\tilde{\mathbf{Q}}(k+1) \tag{8.120}$$

This equation is very important because it shows how $\tilde{\mathbf{Q}}(k+1)$ can be calculated from $\tilde{\mathbf{Q}}^{(N+2)}(k+1)$. Note that, from the forward prediction filter, $\mathbf{Q}^{(N+2)}(k+1)$ is available using rotation matrices of the instant k, i.e., $\mathbf{Q}^{(N+2)}(k)$ and $\tilde{\mathbf{Q}}(k)$, and the single rotation represented by $\tilde{\mathbf{Q}}_f(k)$, as can be verified in equation (8.93). In addition to $\tilde{\mathbf{Q}}_f(k)$, $\tilde{\mathbf{Q}}_b(k+1)$ and $\mathbf{Q}_b'(k+1)$ need to be calculated in order to generate $\tilde{\mathbf{Q}}(k+1)$. As can be seen in equation (8.106), $\tilde{\mathbf{Q}}_b(k+1)$ consists of a single rotation. On the other hand, $\mathbf{Q}_b'(k+1)$ consists of $(N+1)$ rotations that can be calculated in a more efficient way described below.

A way of computing $\theta_{bi}'(k+1)$ of equation (8.115) has to be found. If the equation (8.120) is examined closely, the unknown matrices are $\mathbf{Q}_b'(k+1)$ and $\tilde{\mathbf{Q}}(k+1)$. Now, if the equation (8.30) is recalled, we can skip $\tilde{\mathbf{Q}}(k+1)$ in equation (8.120) by posmultiplying each side by a vector with a single nonzero element as follows:

$$\begin{aligned} \tilde{\mathbf{Q}}^{(N+2)}(k+1)\begin{bmatrix} 1 & 0 \\ 0 & \mathbf{Q}_b'(k) \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & \mathbf{Q}_b(k) \end{bmatrix}\begin{bmatrix} \mathbf{0}_{k-N} \\ a \\ \mathbf{0}_{N+1} \end{bmatrix} \\ = \mathbf{Q}_b'(k+1)\mathbf{Q}_b(k+1)\begin{bmatrix} \mathbf{0}_{k-N} \\ a \\ \mathbf{0}_{N+1} \end{bmatrix} \end{aligned} \tag{8.121}$$

Since the desired angles of $\mathbf{Q}_b'(k+1)$ occupy only the $N+2$ bottom rows of the above equation, using equation (8.108), we can reduce it to

$$\begin{aligned} \mathbf{c}(k+1) &= \hat{\mathbf{Q}}_\theta^{(N+2)}(k+1)\mathbf{Q}_{\theta_b}'(k)\begin{bmatrix} a[\mathbf{Q}_b(k)]_{k-N,k-N} \\ \mathbf{0} \end{bmatrix} \\ &= \mathbf{Q}_{\theta b}'(k+1)\begin{bmatrix} a[\mathbf{Q}_b(k+1)]_{k-N+1,k-N+1} \\ \mathbf{0} \end{bmatrix} \end{aligned} \tag{8.122}$$

where $\hat{\mathbf{Q}}_\theta^{(N+2)}(k+1)$ is a matrix with diagonal elements given by $\cos^{(N+2)}\theta_0(k+1)$, $\cos^{(N+2)}\theta_1(k+1)$, ..., $\cos\theta_{N+1}^{(N+2)}(k+1)$, respectively. This matrix is obtained by deleting the first $k-N$ rows and columns of $\tilde{\mathbf{Q}}^{(N+2)}(k+1)$. This simplification is possible because, as can be seen from the general form of $\tilde{\mathbf{Q}}(k)$

in equation (8.30), the last $N + 2$ nonzero elements in the first column of $\tilde{\mathbf{Q}}^{(N+2)}(k+1)$ always multiply zero elements. The matrix $\mathbf{Q}'_{\theta b}(k)$ is also a submatrix of $\mathbf{Q}'_b(k)$ with the first $k - N$ rows and columns deleted.

The equation above can then be written as

$$\mathbf{c}(k+1) = \hat{\mathbf{Q}}_\theta^{(N+2)}(k+1)\mathbf{Q}'_{\theta b}(k)\begin{bmatrix} 1 \\ \mathbf{0}_{N+1} \end{bmatrix} = \mathbf{Q}'_{\theta b}(k+1)\begin{bmatrix} b \\ \mathbf{0} \end{bmatrix} \tag{8.123}$$

where a was chosen as

$$a = \frac{1}{[\mathbf{Q}_b(k+1)]_{k-N,k-N}}$$

and

$$b = a[\mathbf{Q}_b(k+1)]_{k-N-1,k-N-1}$$

The quantity b does not need to be explicitly calculated in order to obtain the angles θ'_b. Define α_i as an auxiliary variable representing the first element of the resulting vector of the righthand side after the ith rotation is applied. If the Euclidean norm is applied in both sides of equation (8.123) the following equality results

$$\alpha_0^2 = b^2 = ||\mathbf{c}(k+1)||^2$$

After the last rotation the relation below is valid.

$$\alpha_{N+1}^2 = c_1^2(k+1)$$

Also, since each element of $\mathbf{c}(k+1)$, except for the first element, is computed through a single rotation, it follows that

$$\alpha_{i-1}^2 = \alpha_i^2 + c_{N+3-i}^2(k+1)$$

With the relations above, the angles $\theta'_{bi}(k+1)$ can be calculated as

$$\alpha_{N+1} = c_1(k+1)$$

for $i = 1$ to $N + 1$ calculate

$$\alpha_{N+1-i}^2 = \alpha_{N+2-i}^2 + c_{i+1}^2(k+1) \tag{8.124}$$

$$\sin\theta'_{bi}(k+1) = -\frac{c_{i+1}(k+1)}{\alpha_{N+1-i}} \tag{8.125}$$

$$\cos\theta'_{bi}(k+1) = \frac{\alpha_{N+2-i}}{\alpha_{N+1-i}} \tag{8.126}$$

Finally, if the pinning vector $[1\ 0\ \ldots\ 0]^T$, is multiplied on both sides of equation (8.120), the following relation results

$$\begin{aligned}
\tilde{\mathbf{Q}}^{(N+2)}(k+1)\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix} &= \mathbf{Q}_b'(k+1)\mathbf{Q}_b(k+1)\tilde{\mathbf{Q}}(k+1)\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix} \\
&= \left.\begin{bmatrix}\gamma^{(N+2)}(k+1)\\0\\\vdots\\\mathbf{r}^{(N+2)}(k+1)\end{bmatrix}\right\} N+2 \\
&= \mathbf{Q}_b'(k+1)\mathbf{Q}_b(k+1)\begin{bmatrix}\gamma(k+1)\\0\\\vdots\\\mathbf{r}(k+1)\end{bmatrix} \\
&= \mathbf{Q}_b'(k+1)\tilde{\mathbf{Q}}_b(k+1)\left.\begin{bmatrix}\gamma(k+1)\\0\\\vdots\\\mathbf{r}(k+1)\end{bmatrix}\right\} N+1
\end{aligned} \tag{8.127}$$

where $\mathbf{r}^{(N+2)}(k)$ and $\mathbf{r}(k)$ are vectors representing the last nonzero elements in the first column of $\tilde{\mathbf{Q}}^{(N+2)}(k)$ and $\tilde{\mathbf{Q}}(k)$ respectively, as can be seen in equation (8.30). Also, in the last equation $\mathbf{Q}_b(k+1)$ was reduced to $\tilde{\mathbf{Q}}_b(k+1)$ since the remaining parts do not affect the product because it will multiply zero factors (see equations (8.106) and (8.106)). Actually only the first elements of $\tilde{\mathbf{Q}}_b(k+1)$ affect the product, that is,

$$\begin{aligned}
&\left.\begin{matrix}1\\k-N-1\\1\\N+1\end{matrix}\right\{\begin{bmatrix}\gamma(k+1)\cos\theta_b(k+1)\\0\\\vdots\\0\\\gamma(k+1)\sin\theta_b(k+1)\\\mathbf{r}(k+1)\end{bmatrix} \\
&= \mathbf{Q}_b'^T(k+1)\left.\begin{bmatrix}\gamma^{(N+2)}(k+1)\\0\\\vdots\\\mathbf{r}^{(N+2)}(k+1)\end{bmatrix}\right\}\begin{matrix}1\\k-N-1\\N+2\end{matrix}
\end{aligned} \tag{8.128}$$

Since the interest is calculating $\mathbf{r}(k+1)$, the equation above can be reduced to

$$\begin{bmatrix} 0 \\ \mathbf{r}(k+1) \end{bmatrix} = \mathbf{Q}'^{T}_{\theta b}(k+1)\mathbf{r}^{(N+2)}(k+1) \tag{8.129}$$

where $\mathbf{Q}'_{\theta b}(k+1)$ is the reduced $\mathbf{Q}'_{b}(k+1)$ where its unused $k-N$ rows and columns are deleted.

Now, since we have $\mathbf{r}(k+1)$ available as a function of known quantities. It is possible to calculate the angles of the reduced rotation matrix $\mathbf{Q}_{\theta}(k+1)$ (see equation (8.39)), as follows:

$$\begin{bmatrix} \gamma(k+1) \\ \mathbf{r}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta}(k+1) \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{8.130}$$

Although $\gamma(k+1)$ is not known, reporting back to equation (8.29) and considering that each angle θ_i is individually responsible for an element in the vector $\mathbf{r}(k+1)$, it is easy to show that this equation can be solved by

Initialize $\gamma'_0 = 1$

For $i = 1$ to $N+1$ calculate

$$\sin\theta_{i-1}(k+1) = \frac{r_i(k+1)}{\gamma'_{i-1}} \tag{8.131}$$

$$\begin{aligned} \gamma'^{2}_{i} &= \gamma'^{2}_{i-1}[1-\sin^2\theta_{i-1}(k+1)] \\ &= \gamma'^{2}_{i-1} - r_i^2(k+1) \end{aligned} \tag{8.132}$$

$$\cos\theta_{i-1}(k+1) = \frac{\gamma'_i}{\gamma'_{i-1}} \tag{8.133}$$

where $\gamma'_{N+1} = \gamma(k+1)$ and $r_i(k+1)$ represents the ith element of $\mathbf{r}(k+1)$.

In addition to the relations derived in the last two subsections, we need equations (8.51) and (8.54) to obtain the error of the joint-estimation problem.

The fast QR-RLS algorithm starts with the calculation of the rotated forward prediction error through equation (8.91), then the energy of the forward prediction error is calculated using equation (8.88), along with the cosine and sine of $\tilde{\mathbf{Q}}_f(k)$ given by (8.86) and (8.87), respectively. The elements of $\mathbf{c}(k+1)$ follow

from equation (8.123). These elements are used in equations (8.124), (8.125), and (8.126) in order to calculate the rotation matrices of $\mathbf{Q}_b'(k+1)$ (see equation (8.114)). The following step is to calculate the Givens rotation angles θ_i through equations (8.93) combined with (8.127), (8.130), (8.129), (8.131), (8.132), and (8.133). Finally, the joint-processor section and the *a posteriori* output error are calculated using equations (8.51), and (8.54), respectively. See Algorithm 8.2.

A major issue to consider when implementing the fast QR-RLS algorithm is the numerical behavior. The concern is if the fast QR-RLS algorithm faces the instability inherent to the fast transversal algorithms when implemented in finite precision. A formal proof of stability of the fast algorithm presented in this section is not known, however no instability was detected in our simulations. Fortunately, it has been shown that the alternative fast QR-RLS algorithm is numerically stable provided the finite-precision rotations corresponding to the angles θ_{bi}' remain passive [17]. This result assumes that the input signal is persistently exciting and the numerical operations are performed with sufficient resolution. Closed formulas are available for the dynamic range of the internal variables of the alternative fast QR-RLS algorithm [18]. These formulas are required to determine the wordlength of each register in order to guarantee accurate numerical operations.

Example 8.3

Solve the system identification problem of Example 8.1 using the fast QR-RLS algorithm. Also check the performance of the QR-based algorithms presented in this chapter in finite-precision implementations using fixed-point arithmetic.

Solution:

For the fast QR-RLS algorithm solving the same problem with infinite precision, the measured misadjustment was 0.05334 for $\lambda = 0.99$. Approximately the same value was obtained by the remaining RLS algorithms. Our main purpose in this example is to mention that the fast QR-RLS algorithm is not suitable for the cases where the coefficients of the unknown system are required, because there is no computationally efficient technique to calculate them.

In the finite-precision simulation, we ran the conventional QR-RLS algorithm, the fast QR-RLS algorithm and the alternative fast QR-RLS algorithm to be

Algorithm 8.2

Fast QR-RLS Algorithm

Initialization

$\|\mathbf{e}_f(-1)\|=\delta$ $\quad$ δ small

All cosines with 1 (use for $k \leq N+1$)

All other variables with zero.

Do for each $k \geq 0$

$$\begin{bmatrix} e_{fq_1}(k) \\ \mathbf{x}_{q_2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} x(k+1) \\ \lambda^{1/2}\mathbf{x}_{q_2}(k-1) \end{bmatrix} \tag{8.91}$$

$$\|\mathbf{e}_f(k)\|^2 = \lambda\|\mathbf{e}_f(k-1)\|^2 + e^2_{fq_1}(k) \tag{8.88}$$

$$\cos\theta_f(k) = \frac{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|}{\|\mathbf{e}_f(k)\|} \tag{8.86}$$

$$\sin\theta_f(k) = \frac{e_{fq_1}(k)}{\|\mathbf{e}_f(k)\|} \tag{8.87}$$

$\hat{\mathbf{Q}}_\theta^{(N+2)}(k+1)$= last $N+2$ by $N+2$ elements of $\mathbf{Q}_{\theta f}(k)\begin{bmatrix} \mathbf{Q}_\theta(k) & 0 \\ 0 & 1 \end{bmatrix}$

$$\mathbf{c}(k+1) = \hat{\mathbf{Q}}_\theta^{(N+2)}(k+1)\mathbf{Q}'_{\theta b}(k)\begin{bmatrix} 1 \\ \mathbf{0}_{N+1} \end{bmatrix} \tag{8.123}$$

$\alpha_{N+1} = c_1(k+1)$ $\quad$ $\gamma'_0 = 1$

Do for $i=1$ to $N+1$

$$\alpha^2_{N+1-i} = \alpha^2_{N+2-i} + c^2_{i+1}(k+1) \tag{8.124}$$

$$\sin\theta'_{bi}(k+1) = -\frac{c_{i+1}(k+1)}{\alpha_{N+1-i}} \tag{8.125}$$

$$\cos\theta'_{bi}(k+1) = \frac{\alpha_{N+2-i}}{\alpha_{N+1-i}} \tag{8.126}$$

End

$$\begin{bmatrix} \gamma^{(N+2)}(k+1) \\ \mathbf{r}^{(N+2)}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta f}(k)\begin{bmatrix} \mathbf{Q}_\theta(k) & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ \mathbf{0}_{N+2} \end{bmatrix} \tag{8.127}$$

$\mathbf{r}(k+1)$= last $N+1$ elements of $\mathbf{Q}'^T_{\theta b}(k+1)\mathbf{r}^{(N+2)}(k+1)$ (8.130)

Do for $i=1$ to $N+1$

$$\sin\theta_{i-1}(k+1) = \frac{r_i(k+1)}{\gamma'_{i-1}} \tag{8.131}$$

$$\gamma'^2_i = \gamma'^2_{i-1} - r^2_i(k+1) \tag{8.132}$$

$$\cos\theta_{i-1}(k+1) = \frac{\gamma'_i}{\gamma'_{i-1}} \tag{8.133}$$

End

Filter evolution

$$\begin{bmatrix} e_{q_1}(k+1) \\ \mathbf{d}_{q_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k+1)\begin{bmatrix} d(k+1) \\ \lambda^{1/2}\mathbf{d}_{q_2}(k) \end{bmatrix} \tag{8.51}$$

$$e(k+1) = e_{q_1}(k+1)\gamma'_{N+1} \tag{8.54}$$

End

□

presented in the section 8.6. We tested the algorithms reducing the wordlength up to eight bits, when the performance of the algorithms became distinctive. The obtained learning curves are depicted in Fig. 8.4, where each curve corresponds to the average of ten independent runs. In all cases, the rotations were kept passive. As can be noticed, the fast QR-RLS algorithm presented the poorest performance although no sign of instability was noticed.

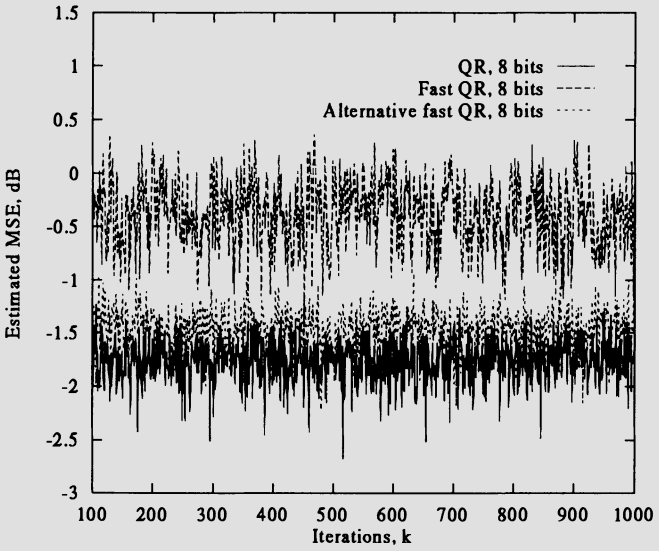


Figure 8.4 Learning curves after convergence

□

Example 8.4

In this example, the signal prediction problem described in subsection 4.6.2 is solved by using the fast QR-RLS algorithm described in this section.

Solution:

Using the fast QR-RLS algorithm, we solved the prediction problem. Choosing $\lambda = 1.0$, the resulting learning curve was as depicted in Fig. 8.5. As expected, for infinite-precision implementation the fast QR-RLS algorithm behaves as any other RLS algorithm. By varying the value of λ between 0.9 and 1 in this

example resulted in little change of the final result. Also, higher signal-to-noise ratio resulted in higher attenuation of the sinusoids at the output error.

□

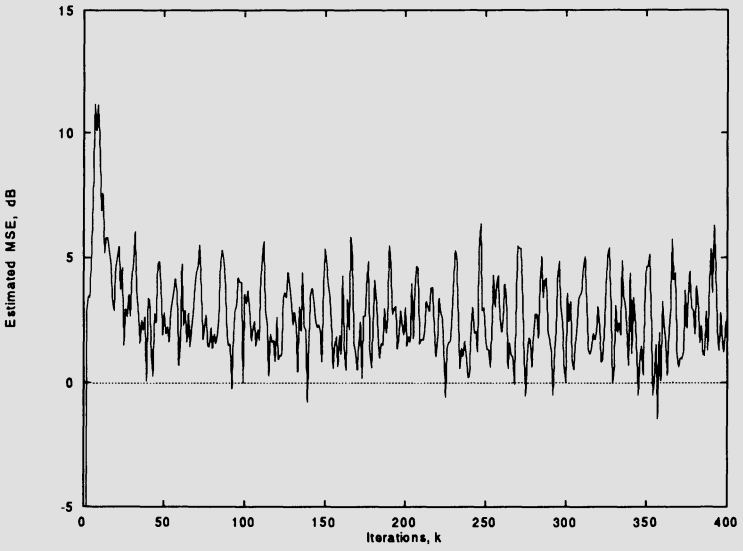


Figure 8.5 Learning curves for the fast QR-RLS algorithm.

8.6 ALTERNATIVE FAST QR-RLS ALGORITHM

An alternative and simpler fast QR-RLS algorithm can be derived by performing the triangularization of the information matrix in a different form, namely by generating a lower triangular matrix and by first applying the triangularization to the backward linear prediction problem. Originally, the algorithm to be presented here was proposed in [6] and later detailed in [8]. The derivations are quite similar to those presented for the standard and the fast QR-RLS algorithms, therefore we will use the previous results in order to avoid unnecessary repetition. In order to accomplish this objective while avoiding confusion, the following notations are respectively used for the triangularization matrix and the lower triangular matrices $\mathcal{Q}$ and $\mathcal{U}$. These matrices have the following

forms

$$\mathcal{U}(k) = \begin{bmatrix} 0 & 0 & \cdots & 0 & u_{1,N+1} \\ 0 & 0 & \cdots & u_{2,N} & u_{2,N+1} \\ \vdots & & & \vdots & \vdots \\ u_{N+1,1} & u_{N+1,2} & \cdots & u_{N+1,N} & u_{N+1,N+1} \end{bmatrix} \tag{8.134}$$

$$\begin{aligned} \tilde{\mathcal{Q}}(k) = & \begin{bmatrix} \cos\theta_N(k) & \cdots & 0 & \cdots & -\sin\theta_N(k) & \mathbf{0} \\ \vdots & & & & \vdots & \vdots \\ 0 & & \mathbf{I}_{k-N-1} & & 0 & \vdots \\ \vdots & & & & \vdots & \vdots \\ \sin\theta_N(k) & \cdots & 0 & \cdots & \cos\theta_N(k) & \mathbf{0} \\ & & \mathbf{0} & & & \mathbf{I}_N \end{bmatrix} \\ & \cdot \begin{bmatrix} \cos\theta_{N-1}(k) & \cdots & 0 & \cdots & -\sin\theta_{N-1}(k) & 0 \\ \vdots & & & & \vdots & \vdots \\ 0 & & \mathbf{I}_{k-N} & & 0 & \vdots \\ \vdots & & & & \vdots & \vdots \\ \sin\theta_{N-1}(k) & \cdots & 0 & \cdots & \cos\theta_{N-1}(k) & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \mathbf{I}_{N-1} \end{bmatrix} \\ & \cdots \cdot \begin{bmatrix} \cos\theta_0(k) & \cdots & 0 & \cdots & -\sin\theta_0(k) \\ \vdots & & & & \vdots \\ 0 & & \mathbf{I}_{k-1} & & 0 \\ \vdots & & & & \vdots \\ \sin\theta_0(k) & \cdots & 0 & \cdots & \cos\theta_0(k) \end{bmatrix} \end{aligned} \tag{8.135}$$

The triangularization procedure has the following general form

$$\begin{aligned} \mathcal{Q}(k)\underline{\mathbf{X}}(k) = & \tilde{\mathcal{Q}}(k) \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathcal{Q}}(k-1) \end{bmatrix} \begin{bmatrix} \mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathcal{Q}}(k-2) \end{bmatrix} \\ & \cdots \begin{bmatrix} \mathbf{I}_{k-N} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathcal{Q}}(k-N) \end{bmatrix} \underline{\mathbf{X}}(k) \end{aligned}$$

$$= \underbrace{\begin{bmatrix} \mathbf{0} \\ \mathcal{U}(k) \end{bmatrix}}_{N+1} \begin{matrix} \} k-N \\ \} N+1 \end{matrix} \tag{8.136}$$

where $\mathcal{Q}(k)$ is a $(k+1)$ by $(k+1)$ matrix which represents the overall triangularization matrix.

As usual the multiplication by zero elements can be avoided by replacing $\tilde{\mathcal{Q}}(k)$ by $\mathcal{Q}_\theta(k)$, where the increasing $\mathbf{I}_{k-N-1}$ section of $\tilde{\mathcal{Q}}(k)$ is removed. The resulting equation is

$$\mathcal{Q}_\theta(k) \begin{bmatrix} \mathbf{x}^T(k) \\ \lambda^{1/2}\mathcal{U}(k-1) \end{bmatrix} = \mathcal{Q}'_{\theta_N}(k)\mathcal{Q}'_{\theta_{N-1}}(k)\cdots\mathcal{Q}'_{\theta_i}(k) \begin{bmatrix} \mathbf{x}'_i(k) \\ \mathcal{U}'_i(k) \end{bmatrix} \tag{8.137}$$

where $\mathcal{Q}'_{\theta_i}(k)$ is derived from $\mathcal{Q}'_i(k)$ by removing the $\mathbf{I}_{k-N-1}$ section of $\mathcal{Q}'_i(k)$ along with the corresponding rows and columns, resulting in the following form

$$\mathcal{Q}'_{\theta_i}(k) = \begin{bmatrix} \cos\theta_i(k) & \cdots & 0 & \cdots & -\sin\theta_i(k) & \cdots & 0 \\ \vdots & & & & \vdots & & \vdots \\ 0 & & \mathbf{I}_{N-i} & & 0 & \cdots & 0 \\ \vdots & & & & \vdots & & \vdots \\ \sin\theta_i(k) & \cdots & 0 & \cdots & \cos\theta_i(k) & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \mathbf{I}_i \\ 0 & \cdots & 0 & \cdots & 0 & & \end{bmatrix} \tag{8.138}$$

The Givens rotation elements are calculated by

$$\cos\theta_i(k) = \frac{[\mathcal{U}'_i(k)]_{N+1-i,i+1}}{c_i} \tag{8.139}$$

$$\sin\theta_i(k) = \frac{x'_i(k-i)}{c_i} \tag{8.140}$$

where $c_i = \sqrt{[\mathcal{U}'_i(k)]^2_{i+1,N+1-i} + x'^2_i(k-N-i)}$, and $[\cdot]_{i,j}$ denotes the (i,j) element of the matrix.

The triangularization matrix $\mathcal{Q}(k+1)$ of the input data matrix can be applied to the backward prediction error as discussed in the previous section resulting in

$$\mathcal{Q}(k+1)\mathbf{e}_b(k+1) = \mathcal{Q}(k+1)\mathbf{d}_b(k+1) - \begin{bmatrix} \mathbf{0} \\ \mathcal{U}(k+1) \end{bmatrix} \mathbf{w}_b(k+1) \tag{8.141}$$

or equivalently

$$\mathbf{e}_{bq}(k+1) = \mathbf{d}_{bq}(k+1) - \begin{bmatrix} \mathbf{0} \\ \mathcal{U}(k+1) \end{bmatrix} \mathbf{w}_b(k+1) \tag{8.142}$$

From equation (8.142), it follows that

$$\begin{aligned} \mathbf{e}_{bq}(k+1) &= \mathcal{Q}(k+1)[\underline{\mathbf{X}}(k+1)\ \mathbf{d}_b(k+1)] \begin{bmatrix} -\mathbf{w}_b(k+1) \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} & e_{bq_1}(k+1) \\ \mathbf{0} & e_{bq_2}(k+1) \\ & \vdots \\ & e_{bq_{k-N+1}}(k+1) \\ \mathcal{U}(k+1) & \mathbf{x}_{q_3}(k+1) \end{bmatrix} \begin{bmatrix} -\mathbf{w}_b(k+1) \\ 1 \end{bmatrix} \end{aligned} \tag{8.143}$$

As before, in order to avoid increasing vectors in the algorithm, $e_{bq_1}(k+1)$,$e_{bq_2}(k+1)$,..., $e_{bq_{k-N}}(k+1)$ can be eliminated in equation (8.143) through Givens rotations, as follows:

$$\begin{aligned} & \mathbf{Q}_b(k+1)\mathbf{e}_{bq}(k+1) \\ &= \mathbf{Q}_b(k+1) \begin{bmatrix} & e_{bq_1}(k+1) \\ \mathbf{0} & e_{bq_2}(k+1) \\ & \vdots \\ & e_{bq_{k-N+1}}(k+1) \\ \mathcal{U}(k+1) & \mathbf{x}_{q_3}(k+1) \end{bmatrix} \begin{bmatrix} -\mathbf{w}_b(k+1) \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ & \|\mathbf{e}_b(k+1)\| \\ \mathcal{U}(k+1) & \mathbf{x}_{q_3}(k+1) \end{bmatrix} \begin{bmatrix} -\mathbf{w}_b(k+1) \\ 1 \end{bmatrix} \end{aligned} \tag{8.144}$$

Note that by induction $[\mathcal{U}]_{N+1-i,i+1}(k+1) = \|\mathbf{e}_{b,i}(k+1)\|$, where $\|\mathbf{e}_{b,i}(k+1)\|^2$ corresponds to the least-square backward prediction error of an ith-order predictor.

In the forward prediction case, we have the following relations

$$\mathbf{e}_f(k) = \left[\begin{array}{c|c} \mathbf{d}_f(k) & \underline{\mathbf{X}}(k) \\ & \mathbf{0} \end{array}\right] \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix} \tag{8.145}$$

and

$$\begin{aligned}\mathbf{e}_{fq}(k) &= \begin{bmatrix} \mathcal{Q}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \left[\begin{array}{c|c} \mathbf{d}_f(k) & \underline{\mathbf{X}}(k) \\ & \\ & \mathbf{0} \end{array}\right] \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix} \\ &= \begin{bmatrix} e_{fq_1}(k) & \\ \vdots & \mathbf{0} \\ e_{fq_{k-N}}(k) & \\ \mathbf{x}_{q_2}(k) & \mathcal{U}(k) \\ \lambda^{\frac{k+1}{2}} x(0) & \mathbf{0} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix} \end{aligned} \tag{8.146}$$

In order to recursively solve equation (8.146) without dealing with ever increasing matrices, a set of Givens rotations are applied in order to eliminate $e_{fq_1}(k)$, $e_{fq_2}(k)$, $\ldots$, $e_{fq_{k-N}}(k)$, such that the information matrix that premultiplies the vector $[1 \quad -\mathbf{w}_f(k)]^T$ is triangularized. The Givens rotations can recursively be obtained by

$$\begin{aligned}\mathbf{Q}_f(k) &= \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_f(k-1) \end{bmatrix} \\ &= \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Q}}_f(k-1) \end{bmatrix} \cdots \begin{bmatrix} \mathbf{I}_{k-N-1} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Q}}_f(N+1) \end{bmatrix} \end{aligned} \tag{8.147}$$

where $\tilde{\mathbf{Q}}_f(k)$ is defined as

$$\tilde{\mathbf{Q}}_f(k) = \begin{bmatrix} \cos\theta_f(k) & \cdots & 0 & \cdots & -\sin\theta_f(k) \\ \vdots & & & & \vdots \\ 0 & & \mathbf{I}_k & & 0 \\ \vdots & & & & \vdots \\ \sin\theta_f(k) & \cdots & 0 & \cdots & \cos\theta_f(k) \end{bmatrix} \tag{8.148}$$

If in each iteration, the rotation above is applied to equation (8.146), we have

$$\mathbf{e}'_{fq}(k) = \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_f(k-1) \end{bmatrix} \begin{bmatrix} e_{fq_1}(k) & \\ \vdots & \mathbf{0} \\ e_{fq_{k-N}}(k) & \\ \mathbf{x}_{q_2}(k) & \mathcal{U}(k) \\ \lambda^{\frac{k+1}{2}} x(0) & \mathbf{0} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix}$$

$$
\begin{aligned}
&= \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} e_{fq_1}(k) & \\ 0 & \mathbf{0} \\ \vdots & \\ 0 & \\ \mathbf{x}_{q_2}(k) & \mathcal{U}(k) \\ \lambda^{1/2}||\mathbf{e}_f(k-1)|| & \mathbf{0} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix} \\
&= \begin{bmatrix} 0 & \\ \vdots & \mathbf{0} \\ 0 & \\ \mathbf{x}_{q_2}(k) & \mathcal{U}(k) \\ ||\mathbf{e}_f(k)|| & \mathbf{0} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix} \qquad (8.149)
\end{aligned}
$$

where

$$\cos\theta_f(k) = \frac{\lambda^{1/2}||\mathbf{e}_f(k-1)||}{\sqrt{\lambda||\mathbf{e}_f(k-1)||^2 + e_{fq_1}^2(k)}} \qquad (8.150)$$

$$\sin\theta_f(k) = \frac{e_{fq_1}(k)}{\sqrt{\lambda||\mathbf{e}_f(k-1)||^2 + e_{fq_1}^2(k)}} \qquad (8.151)$$

and $||\mathbf{e}_f(k)||$ is the norm of the forward prediction error vector shown in equation (8.74). This result can be shown by once more evoking the fact that the last element of $\mathbf{e}'_{fq}(k)$ is equal to $||\mathbf{e}_f(k)||$, since $||\mathbf{e}'_{fq}(k)|| = ||\mathbf{e}_{fq}(k)|| = ||\mathbf{e}_f(k)||$ because these error vectors are related through unitary transformations. Also, it is worthwhile to recall that in equation (8.149) the relation $[\mathcal{U}]_{N+1-i,i+1}(k) = ||\mathbf{e}_{b,i}(k)||$ is still valid.

In the present case, it can be assumed that the partial triangularization can be performed at each iteration as follows:

$$\begin{bmatrix} 0 & \\ 0 & \mathbf{0} \\ \vdots & \\ 0 & \\ \mathbf{x}_{q_2}(k) & \mathcal{U}(k) \\ ||\mathbf{e}_f(k)|| & \mathbf{0} \end{bmatrix}$$

$$= \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} \tilde{\mathcal{Q}}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x(k+1) & \mathbf{x}^T(k) \\ \mathbf{0} & \mathbf{0} \\ \lambda^{1/2}\mathbf{x}_{q_2}(k-1) & \lambda^{1/2}\mathcal{U}(k-1) \\ \lambda^{1/2}\|\mathbf{e}_f(k-1)\| & \mathbf{0} \end{bmatrix} \tag{8.152}$$

Now we can eliminate $\mathbf{x}_{q_2}(k)$ through a set of rotations $\mathbf{Q'}_f(k+1)$ such that

$$\mathcal{U}^{(N+2)}(k+1) = \mathbf{Q'}_f(k+1) \begin{bmatrix} \mathbf{x}_{q_2}(k) & \mathcal{U}(k) \\ \|\mathbf{e}_f(k)\| & \mathbf{0} \end{bmatrix} \tag{8.153}$$

From the equation above, we can realize that $\mathbf{Q'}_f(k+1)$ consists of a series of rotations in the following order

$$\mathbf{Q}'_f(k+1) = \begin{bmatrix} \mathbf{I}_N & \mathbf{0} & \\ \mathbf{0} & \cos\theta'_{f_1}(k+1) & -\sin\theta'_{f_1}(k+1) \\ & \sin\theta'_{f_1}(k+1) & \cos\theta'_{f_1}(k+1) \end{bmatrix}$$

$$\cdots \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \cos\theta'_{f_N}(k+1) & 0 & \cdots & 0 & \cdots & 0 & -\sin\theta'_{f_N}(k+1) \\ \vdots & 0 & & & & & & 0 \\ \vdots & & & \mathbf{I}_{N-1} & & & \vdots & \vdots \\ 0 & \sin\theta'_{f_N}(k+1) & 0 & \cdots & 0 & \cdots & 0 & \cos\theta'_{f_N}(k+1) \end{bmatrix}$$

$$\cdot \begin{bmatrix} \cos\theta'_{f_{N+1}}(k+1) & 0 & \cdots & 0 & \cdots & 0 & -\sin\theta'_{f_{N+1}}(k+1) \\ 0 & & & & & & 0 \\ \vdots & & & \mathbf{I}_N & & & \vdots \\ \vdots & & & & & & \vdots \\ \sin\theta'_{f_{N+1}}(k+1) & 0 & \cdots & 0 & \cdots & 0 & \cos\theta'_{f_{N+1}}(k+1) \end{bmatrix} \tag{8.154}$$

where the rotation entries of $\mathbf{Q'}_f(k+1)$ are calculated as follows:

$$\begin{aligned} \mu_i &= \sqrt{\mu_{i-1}^2 + x_{q_2 i}^2(k)} \\ \cos\theta'_{f_{N+2-i}}(k+1) &= \frac{\mu_{i-1}}{\mu_i} \\ \sin\theta'_{f_{N+2-i}}(k+1) &= \frac{x_{q_2 i}(k)}{\mu_i} \end{aligned} \tag{8.155}$$

for $i = 1, \ldots, N+1$, where $\mu_0 = ||\mathbf{e}_f(k)||$. Note that μ_{N+1} is the norm of the weighted backward prediction error $||\mathbf{e}_{b,0}(k+1)||$, for a zero-order predictor (see equation (8.144)). The quantity $x_{q_2 i}(k)$ denotes the ith element of the vector $\mathbf{x}_{q_2}(k)$.

Since the rotations above, at instant k, are actually completing the triangularization of $\underline{\mathbf{X}}^{(N+2)}(k+1)$, it follows that

$$\tilde{Q}^{(N+2)}(k+1) = \begin{bmatrix} \mathbf{I}_{k-N} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q'}_f(k+1) \end{bmatrix} \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} \tilde{Q}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \tag{8.156}$$

If the pinning vector $[1 \ 0 \ \ldots \ 0]^T$, is posmultiplied on both sides of the above equation we obtain the following relation

$$\begin{aligned} \tilde{Q}^{(N+2)}(k+1) \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} &= \begin{bmatrix} \mathbf{I}_{k-N} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q'}_f(k+1) \end{bmatrix} \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} \tilde{Q}(k) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &= \left.\begin{bmatrix} \gamma^{(N+2)}(k+1) \\ 0 \\ \vdots \\ \mathbf{r}^{(N+2)}(k+1) \end{bmatrix}\right\} N+2 \\ &= \begin{bmatrix} \mathbf{I}_{k-N} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q'}_f(k+1) \end{bmatrix} \tilde{\mathbf{Q}}_f(k) \left.\begin{bmatrix} \gamma(k) \\ 0 \\ \vdots \\ \mathbf{r}(k) \\ 0 \end{bmatrix}\right\} N+1 \end{aligned} \tag{8.157}$$

where $\mathbf{r}^{(N+2)}(k)$ and $\mathbf{r}(k)$ are vectors representing the last nonzero elements in the first column of $\tilde{Q}^{(N+2)}(k)$ and $\tilde{Q}(k)$, respectively, as can be seen in equation (8.135). Now, we can proceed by performing the product involving the matrix $\tilde{\mathbf{Q}}_f(k)$ resulting in the following relation

$$\begin{matrix} 1 \\ k-N \\ N+1 \end{matrix} \left\{ \begin{bmatrix} \gamma(k)\cos\theta_f(k) \\ 0 \\ \vdots \\ \mathbf{r}(k) \\ \gamma(k)\sin\theta_f(k) \end{bmatrix} \right.$$

$$= \begin{bmatrix} \mathbf{I}_{k-N} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q'}_f^T(k+1) \end{bmatrix} \begin{bmatrix} \gamma^{(N+2)}(k+1) \\ 0 \\ \vdots \\ \mathbf{r}^{(N+2)}(k+1) \end{bmatrix} \begin{matrix} \} 1 \\ \\ \} k-N-1 \\ \} N+2 \end{matrix} \quad (8.158)$$

Since our interest is to calculate $\mathbf{r}(k+1)$, the equation above can be reduced to

$$\mathbf{Q'}_f(k+1) \begin{bmatrix} \mathbf{r}(k) \\ \gamma(k)\sin\theta_f(k) \end{bmatrix} = \mathbf{r}^{(N+2)}(k+1) \quad (8.159)$$

where the unused $k-N$ rows and columns were deleted and $\mathbf{r}(k+1)$ is the last $N+1$ rows of $\mathbf{r}^{(N+2)}(k+1)$. Now, since we have $\mathbf{r}(k+1)$ available as a function of known quantities, it is possible to calculate the angles of the reduced rotation matrix $\mathcal{Q}_\theta(k+1)$.

Like in the fast QR-RLS algorithm presented in the previous section, in the alternative fast QR-RLS algorithm we first calculate the rotated forward prediction error as in equation (8.91), followed by the calculation of the energy of the forward prediction error using equation (8.88) and the elements of $\tilde{\mathbf{Q}}_f(k)$ given in equations (8.86) and (8.87), respectively. The rotation entries of $\mathbf{Q'}_f(k+1)$ are calculated using the relations of (8.155), which in turn allow us to calculate $\mathbf{r}^{(N+2)}(k+1)$ through equation (8.159). Given $\mathbf{r}^{(N+2)}(k+1)$, the rotation angles θ_i can be easily calculated. The remaining equations of the algorithm are the joint-processor section and the computation of the forward prediction error given by equations (8.51) and (8.54), respectively.

The resulting Algorithm 8.3 is almost the same as the hybrid QR-lattice algorithm of [9]. The main difference is the order the of computation of the angles θ_i, that in [9] starts from θ_N by employing the relation

$$\gamma(k+1) = \sqrt{1 - ||\mathbf{r}(k+1)||^2} \quad (8.160)$$

This algorithm is closely related to the normalized lattice algorithm, see [9]. Some key results are needed to establish the relation between these algorithms, for example it can be shown that the parameter $\gamma(k, N+1)$ of the lattice algorithms corresponds to $\gamma^2(k)$ in the alternative fast QR algorithm.

Example 8.5

In this example, the system identification problem described in subsection 3.6.2 is solved using the alternative fast QR-RLS algorithm described in this section. We implemented the fast QR-RLS algorithm with finite precision.

Algorithm 8.3

Alternative Fast QR-RLS Algorithm

Initialization

$\|\mathbf{e}_f(-1)\|=\delta$ $\quad \delta$ small

All cosines with 1 (use for $k\leq N+1$)

All other variables with zero.

Do for each $k\geq 0$

$$\begin{bmatrix} e_{fq_1}(k) \\ \mathbf{x}_{q_2}(k) \end{bmatrix} = \mathcal{Q}_\theta(k) \begin{bmatrix} x(k+1) \\ \lambda^{1/2}\mathbf{x}_{q_2}(k-1) \end{bmatrix} \tag{8.91}$$

$$\|\mathbf{e}_f(k)\|^2=\lambda\|\mathbf{e}_f(k-1)\|^2+e^2_{fq_1}(k) \tag{8.88}$$

$$\sin\theta_f(k)=\frac{e_{fq_1}(k)}{\|\mathbf{e}_f(k)\|} \tag{8.86}$$

$\mu_0=\|\mathbf{e}_f(k)\|$

Do for $i=1$ to $N+1$

$$\mu_i=\sqrt{\mu^2_{i-1}+x^2_{q_2 i}(k)} \tag{8.155}$$

$$\cos\theta'_{f_{N+2-i}}(k+1)=\frac{\mu_{i-1}}{\mu_i} \tag{8.155}$$

$$\sin\theta'_{f_{N+2-i}}(k+1)=\frac{x_{q_2 i}(k)}{\mu_i} \tag{8.155}$$

End

$$\mathbf{r}^{(N+2)}(k+1)=\mathbf{Q}'_f(k+1)\begin{bmatrix} \mathbf{r}(k) \\ \gamma(k)\sin\theta_f(k) \end{bmatrix} \tag{8.159}$$

$\mathbf{r}(k+1)=$ last $N+1$ elements of $\mathbf{r}^{(N+2)}(k+1)$

$\gamma'_0=1$

Do for $i=1$ to $N+1$

$$\sin\theta_{i-1}(k+1)=\frac{r_{N+2-i}(k+1)}{\gamma'_0}$$

$$\gamma'^2_1=\gamma'^2_0-r^2_{N+2-i}(k+1)$$

$$\cos\theta_{i-1}(k+1)=\frac{\gamma'_1}{\gamma'_0}$$

$\gamma'_0=\gamma'_1$

End

$\gamma(k+1)=\gamma'_1$

Filter evolution

$$\begin{bmatrix} e_{q_1}(k+1) \\ \mathbf{d}_{q_2}(k+1) \end{bmatrix} = \mathcal{Q}_\theta(k+1)\begin{bmatrix} d(k+1) \\ \lambda^{1/2}\mathbf{d}_{q_2}(k) \end{bmatrix} \tag{8.51}$$

$$e(k+1)=e_{q_1}(k+1)\gamma(k+1) \tag{8.54}$$

End

□

Solution:

The main objective of this example is to test the stability of the alternative fast QR-RLS algorithm. For that we run the algorithm implemented with fixed-point arithmetic. The wordlengths used were 16, 12, and 10 bits respectively. We forced the rotations to be kept passive. In other words, for each rotation the sum of the squares of the quantized sine and cosine were kept less or equal than one. Also, we tested γ' to avoid that it becomes less than zero. With these measures we did not notice any sign of divergence in our experiments. Table 8.2 shows the measured MSE in the finite-precision implementation, where the expected MSE for the infinite-precision implementation was 0.0015. The analysis of these results shows that the alternative fast QR-RLS has low sensitivity to quantization effects being comparable to the other stable RLS algorithms presented in this text.

□

Table 8.2 Results of the Finite-Precision Implementation of the alternative fast QR-RLS Algorithm

No. of bits	$\xi(k)_Q$ Experiment
16	$1.7\ 10^{-3}$
12	$2.0\ 10^{-3}$
10	$2.1\ 10^{-3}$

8.7 CONCLUSIONS AND FURTHER READING

Motivated by the numerically well conditioned Givens rotations, two types of rotation-based algorithms were presented in this chapter. In both cases the QR decomposition implemented with orthogonal Givens rotations are employed. The first algorithm is computationally intensive (order N^2) and is mainly useful in applications where the input signal vector does not consist of time delayed elements. The advantages of this algorithm are its numerical stability and its systolic array implementation. The second class of algorithms explores the time-shift property of the input signal vector which is inherent to a number

of applications, yielding the fast QR-RLS algorithms with order N numerical operations per output sample.

It should be noticed that the subject of QR-decomposition-based algorithms is not fully covered here. A complete approach to generate fast QR-RLS algorithm using lattice formulation is known [19]-[22]. In [19], the author applied QR decomposition to avoid inversion of covariance matrices in the multichannel problem employing lattice RLS formulation. A full orthogonalization of the resulting algorithm was later proposed in [21]. By using different formulations, the works of [20], [21], and [22], propose virtually identical QR-decomposition-based lattice RLS algorithms. In terms of computational complexity, the fast QR-RLS algorithms presented in this chapter are more efficient.

Another family of algorithms employing QR decomposition are those that replace the Givens rotation by the Householder transformation [1]. The Householder transformation can be considered an efficient method to compute the QR decomposition and is known to yield more accurate results than the Givens rotations in finite-precision implementations. In [23], the fast Householder RLS adaptive filtering algorithm was proposed and shown to require order $7N$ of computational complexity. However, no stability proof for this algorithm exists so far. In another work, the Householder transformation is employed to derive a block-type RLS algorithm that can be mapped on a systolic-block Householder transformation [24].

A major drawback of the conventional QR-RLS algorithm is the backsubstitution algorithm which is required for computing the weight vector. In systolic array, it can be implemented as shown in this chapter, through bidirectional array that requires extra clock cycles. Alternatively a two-dimensional array can be employed despite of being more computationally expensive [10]. An approach called inverse QR method can be used to derive a QR-based RLS algorithm such that the weight vector can be calculated without backsubstitution [25]-[26].

The QR decomposition has also been shown to be useful for the implementation of numerically stable nonlinear adaptive filtering algorithms. In [27], a QR-based RLS algorithm for adaptive nonlinear filtering has been proposed.

Some performance evaluations of the QR-RLS and fast QR-RLS algorithms are found in this chapter, where these algorithms were employed in some simulation examples.

References

1. G. H. Golub and C. F. Van Loan, *Matrix Computations,* John Hopkins University Press, Baltimore, MD, 2nd edition, 1989.

2. W. H. Gentleman and H. T. Kung, "Matrix triangularization by systolic arrays," *Proc. of SPIE, Real Time Signal Processing* IV, vol. 298, pp. 19-26, 1981.

3. J. G. McWhirter, "Recursive least-squares minimization using a systolic array," *Proc. of SPIE, Real Time Signal Processing* VI, vol. 431, pp. 105-112, 1983.

4. S. Haykin, *Adaptive Filter Theory,* Prentice Hall, Englewood Cliffs, NY, 2nd edition, 1991.

5. J. M. Cioffi, "The fast adaptive ROTOR's RLS algorithm," *IEEE Trans. on Acoust., Speech, and Signal Processing,* vol. 38, pp. 631-653, April 1990.

6. I. K. Proudler, J. G. McWhirter, and Y. J. Shepherd, "Fast QRD-based algorithms for least squares linear prediction," *Proc. IMA Conference on Mathematics in Signal Processing,* Warwick, England, pp. 465-488, Dec. 1988.

7. M. G. Bellanger, "The FLS-QR algorithm for adaptive filtering," *Signal Processing,* vol. 17, pp. 291-304, Aug. 1984.

8. M. G. Bellanger and P. A. Regalia, "The FLS-QR algorithm for adaptive filtering: The case of multichannel signals," *Signal Processing,* vol. 22, pp. 115-126, March 1991.

9. P. A. Regalia, M. G. Bellanger, "On the duality between fast QR methods and lattice methods in least squares adaptive filtering," *IEEE Trans. on Signal Processing,* vol. 39, pp. 879-891, April 1991.

10. C. R. Ward, P. J. Hargrave, and J. G. McWhirter, "A novel algorithm and architecture for adaptive digital beamforming," *IEEE Trans. on Antennas and Propagation,* vol. 34, pp.338-346, March 1986.

11. C. A. Mead and L. A. Conway, *Introduction to VLSI Systems,* Addison-Wesley, Reading, MA, 1980.

12. W. H. Gentleman, "Least squares computations by Givens transformations without square roots," *Inst. Maths. Applics.,* vol. 12, pp. 329-336, 1973.

13. W. H. Gentleman, "Error analysis of QR decompositions by Givens transformations," *Linear Algebra and its Applications* , vol. 10, pp. 189-197, 1975.

14. H. Leung and S. Haykin, "Stability of recursive QRD-LS algorithms using finite-precision systolic array implementation," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. 37, pp. 760-763, May 1989.

15. K. J. R. Liu, S.-F. Hsieh, K. Yao, and C.-T. Chiu, "Dynamic range, stability, and fault-tolerant capability of finite-precision RLS systolic array based on Givens rotations," *IEEE Trans. on Circuits and Systems*, vol. 38, pp. 625-636, June 1991.

16. P. S. R. Diniz and M. G. Siqueira, "Fixed-point error analysis of the QR-recursive least squares algorithm," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, pp. 334-348, May 1995.

17. P. A. Regalia, "Numerical stability properties of a QR-based fast least squares algorithm," *IEEE Trans. on Signal Processing*, vol. 41, pp. 2096-2109, June 1993.

18. M. G. Siqueira, P. S. R. Diniz, and A. Alwan,"Infinite precision analysis of the fast QR-recursive least squares algorithm," *Proc. IEEE Intern. Symposium on Circuits and Systems*, London, England, pp. 2.293-2.296 , May 1994.

19. P. S. Lewis, "QR-based algorithms for multichannel adaptive least squares lattice filters," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. 38, pp. 421-432, May 1990.

20. I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, "Computationally efficient QR decomposition approach to least squares adaptive filtering," *IEE Proceedings-Part F*, vol. 148, pp. 341-353, Aug. 1991.

21. B. Yang, and J. F. Böhme, "Rotation-based RLS algorithms: Unified derivations, numerical properties, and parallel implementations," *IEEE Trans. on Signal Processing*, vol. 40, pp. 1151-1166, May 1992.

22. F. Ling, "Givens rotation based least squares lattice and related algorithms," *IEEE Trans. on Signal Processing*, vol. 39, pp. 1541-1551, July 1991.

23. J. M. Cioffi, "The fast Householder filters RLS adaptive filter," *Proc IEEE Intern. Conf. on Acoust., Speech, Signal Processing*, Albuquerque, NM, pp. 1619-1622, 1990.

24. K. J. R. Liu, S.-F. Hsieh, and K. Yao, "Systolic block Householder transformation for RLS algorithm with two-level pipelined implementation," *IEEE Trans. on Signal Processing*, vol. 40, pp. 946-957, April 1992.

25. A. Ghirnikar and S. T. Alexander,"Stable recursive least squares filtering using an inverse QR decomposition," *Proc IEEE Intern. Conf. on Acoust., Speech, Signal Processing*, Albuquerque, NM, pp. 1623-1626, 1990.

26. S. T. Alexander, and A. Ghirnikar, "A method for recursive least squares filtering based upon an inverse QR decomposition," *IEEE Trans. on Signal Processing*, vol. 41, pp. 20-30, Jan. 1993.

27. M. Syed and V. J. Mathews, "QR-Decomposition based algorithms for adaptive Volterra filtering," *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, pp. 372-382, June 1993.

Problems

1. If we consider each anti-diagonal element of $\mathbf{U}(k)$ as a scaling constant d_i, and we multiply the input signal vector initially by a constant δ, we can derive a QR-decomposition algorithm without square root as following described.

   The first two rows to be rotated are
   $\delta\tilde{x}(k) \quad \delta\tilde{x}(k-1) \cdots \quad \delta\tilde{x}(k-N)$
   $d_1\lambda^{1/2}\tilde{u}_{1,1}(k-1) \quad d_1\lambda^{1/2}\tilde{u}_{1,2}(k-1) \quad \cdots \quad d_1$
   where $d_1 = \lambda^{1/2}u_{1,N+1}(k-1)$. The parameter δ can be initialized with 1.
   Applying the Givens rotation to the rows above results in
   $\delta' x_1'(k) \quad \delta' x_1'(k-1) \cdots \quad \delta' x_1'(k-N+1) \quad 0$
   $d_1'\tilde{u}_{1,1}'(k) \quad d_1'\tilde{u}_{1,2}'(k) \cdots \quad d_1'\tilde{u}_{1,N}'(k) \quad d_1'$
   where
   $d_1'^2 = d_1^2 + \delta^2\tilde{x}^2(k-N)$
   $c = \frac{d_1^2}{d_1^2+\delta^2\tilde{x}^2(k-N)}$
   $\delta'^2 = \frac{d_1^2\delta^2}{d_1^2+\delta^2\tilde{x}^2(k-N)}$
   $s = \frac{\delta^2\tilde{x}(k-N)}{d_1^2+\delta^2\tilde{x}(k-N)}$
   $x_1'(k-N+i) = \tilde{x}(k-N+i) - \tilde{x}(k-N)\lambda^{1/2}\tilde{u}_{1,N-i+1}(k-1)$
   $\tilde{u}_{1,N-i+1}'(k) = c\lambda^{1/2}\tilde{u}_{1,N+1-i}(k-1) + s\tilde{x}(k-N+i)$
   The same procedure can be used to triangularize completely the input signal matrix.

(a) Using the procedure above derive a QR-RLS algorithm without square roots.

(b) Compare the computational complexity of the QR-RLS algorithms with and without square roots.

(c) Show that the triangularized matrix $\tilde{\mathbf{U}}(k)$ is related with $\mathbf{U}(k)$ through

$$\mathbf{U}(k) = \mathbf{D}^{'}\tilde{\mathbf{U}}(k)$$

where $\mathbf{D}^{'}$ is a diagonal matrix with the diagonal elements given by $d_i^{'}$ for $i = 1, 2, \ldots, N+1$.

2. Since $\mathbf{Q}^T(k)\mathbf{Q}(k) = \mathbf{I}_{k+1}$, the following identity is valid for any matrix $\mathbf{A}$ and $\mathbf{B}$:
$\mathbf{C}^T\mathbf{D} = \mathbf{A}^T\mathbf{B}$ for $\mathbf{Q}(k)\,[\mathbf{A} \mid \mathbf{B}] = [\mathbf{C} \mid \mathbf{D}]$
where $\mathbf{Q}(k), \mathbf{A}, \mathbf{B}, \mathbf{C}$,and $\mathbf{D}$ have the appropriate dimensions. By choosing $\mathbf{A}, \mathbf{B}, \mathbf{C}$, and $\mathbf{D}$ appropriately derive the following relations.
(a)$\mathbf{U}^T(k)\mathbf{U}(k) = \lambda\mathbf{U}^T(k-1)\mathbf{U}(k-1) + \mathbf{x}(k)\mathbf{x}^T(k)$
(b)$\mathbf{p}_D(k) = \lambda\mathbf{p}_D(k-1) + \mathbf{x}(k)d(k)$
where $\mathbf{p}_D(k) = \Sigma_{i=0}^{k}\mathbf{x}(i)d(i)$
(c)$\mathbf{U}(k)\mathbf{U}^{-T}(k)\mathbf{x}(k) = \mathbf{x}(k)$
where $\mathbf{U}^{-T}(k) = \left[\mathbf{U}^{-1}(k)\right]^T$
(d)$\mathbf{p}_D^T(k)\mathbf{U}^{-1}(k)\mathbf{U}^{-T}(k)\mathbf{x}(k) + e^{'}(k)\gamma(k) = d(k)$

3. Partitioning $\mathbf{Q}_\theta(k)$ as follows:

$$\mathbf{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & \mathbf{q}_\theta^T(k) \\ \mathbf{q}_\theta'(k) & \mathbf{Q}_{\theta r}(k) \end{bmatrix}$$

show from equation(8.51) and (8.39) that
$\mathbf{q}_\theta^T(k)\lambda^{1/2}\mathbf{U}(k-1) + \gamma(k)\mathbf{x}^T(k) = \mathbf{0}^T$

$\mathbf{q}_\theta^T(k)\lambda^{1/2}\mathbf{d}_{q2}(k-1) + \gamma(k)d(k) = e_{q1}(k)$

4. Using the relations of the previous two problems and the fact that $\mathbf{U}(k)\mathbf{w}(k) = \mathbf{d}_{q2}(k)$, show that
(a)$e^{'}(k) = \frac{e_{q1}(k)}{\gamma(k)}$
(b)$e(k) = e^{'}(k)\gamma^2(k)$
(c)$e_{q1}(k) = \sqrt{e(k)e^{'}(k)}$

5. Show that $\mathbf{U}^T(k)\mathbf{d}_{q2}(k) = \mathbf{p}_D(k)$.

6. Using some of the formulas of the conventional RLS algorithm show that $\gamma^2(k) = 1 - \mathbf{x}^T(k)\mathbf{R}_D^{-1}(k)\mathbf{x}(k)$.

7. The QR-RLS algorithm is used to predict the signal $x(k) = \cos(\pi k/3)$ using a second-order FIR filter with the first tap fixed at 1. Note that we are interested in minimizing the MSE of the FIR output error. Given $\lambda = 0.985$, calculate $y^2(k)$ and the filter coefficients for the first 10 iterations.

8. Use the QR-RLS algorithm to identify a system with the transfer function given below. The input signal is a uniformly distributed white noise with variance $\sigma_x^2 = 1$, and the measurement noise is a Gaussian white noise uncorrelated with the input with variance $\sigma_n^2 = 10^{-3}$. The adaptive filter has 12 coefficients.

$$H(z) = \frac{1 - z^{-12}}{1 - z^{-1}}$$

(a) Run the algorithm for $\lambda = 1$, $\lambda = 0.99$, and $\lambda = 0.97$. Comment on the convergence behavior in each case.
(b) Plot the obtained FIR filter frequency response in any iteration after convergence is achieved and compare with the unknown system.

9. Perform the equalization of a channel with the following impulse response

$$h(k) = \sum_{l=k}^{10}(l - 10)[u(k) - u(k - 10)]$$

where $u(k)$ is a step sequence.

Using a known training signal that consists of a binary (-1,1) random signal. An additional Gaussian white noise with variance 10^{-2} is present at the channel output.
(a) Apply the QR-RLS with an appropriate λ and find the impulse response of an equalizer with 50 coefficients.
(b) Convolve one of the equalizers impulse response after convergence, with the channels impulse response and comment on the result.

10. In a system identification problem the input signal is generated by an autoregressive process given by

$$x(k) = -1.2x(k - 1) - 0.81x(k - 2) + n_x(k)$$

where $n_x(k)$ is a zero-mean Gaussian white noise with variance such that $\sigma_x^2 = 1$. The unknown system is described by

$$H(z) = 1 + 0.9z^{-1} + 0.1z^{-2} + 0.2z^{-3}$$

The adaptive filter is also a third-order FIR filter. Using the QR-RLS algorithm:
Choose an appropriate λ, run an ensemble of 20 experiments, and plot the average learning curve.

11. The QR-RLS algorithm was applied to identify a 7th-order time-varying unknown system whose coefficients are first-order Markov processes with $\lambda_{\mathbf{W}} = 0.999$ and $\sigma^2_{\mathbf{W}} = 0.001$. The initial time-varying system multiplier coefficients are

    $\mathbf{w}_o^T = [0.03490 - 0.01100 - 0.068640.223910.556860.35798 - 0.02390 - 0.07594]$

    The input signal is a Gaussian white noise with variance $\sigma_x^2 = 0.7$, and the measurement noise is also a Gaussian white noise independent of the input signal and of the elements of $\mathbf{n}_{\mathbf{W}}(k)$, with variance $\sigma_n^2 = 0.01$.

    (a) For $\lambda = 0.97$ measure the excess MSE.

    (b) Repeat (a) for $\lambda = \lambda_{opt}$

12. Suppose a 15th-order FIR digital filter with multipliers coefficients given below was identified through an adaptive FIR filter of the same order using the QR-RLS algorithm. Considering that fixed-point arithmetic was used and for 10 independent runs, calculate an estimate of the expected value of $||\Delta\mathbf{w}(k)_Q||$ and $\xi(k)_Q$ for the following case.

Additional noise : white noise with variance	$\sigma_n^2 = 0.0015$
Coefficients wordlength:	$b_c = 16$ bits
Signal wordlength:	$b_d = 16$ bits
Input signal: Gaussian white noise with variance	$\sigma_x^2 = 0.7$
	$\lambda = 0.99$

    $\mathbf{w}_o^T = [\ 0.0219360\ 0.0015786\ -0.0602449\ -0.0118907\ 0.1375379\ 0.0574545\ -0.3216703\ -0.5287203\ -0.2957797\ 0.0002043\ 0.290670\ -0.0353349\ -0.0068210\ 0.0026067\ 0.0010333\ -\ 0.0143593]$

    Plot the learning curves for the finite- and infinite-precision implementations.

13. Repeat the problem above for the following cases

    (a) $\sigma_n^2 = 0.01, b_c = 9$ bits, $b_d = 9$ bits, $\sigma_x^2 = 0.7, \lambda = 0.98$.

    (b) $\sigma_n^2 = 0.1, b_c = 10$ bits, $b_d = 10$ bits, $\sigma_x^2 = 0.8, \lambda = 0.98$.

    (c) $\sigma_n^2 = 0.05, b_c = 8$ bits, $b_d = 16$ bits, $\sigma_x^2 = 0.8, \lambda = 0.98$.

14. Repeat problem 12 in the case the input correlation matrix $\mathbf{R}$ is a first-order Markov process with $\lambda_{\mathbf{x}} = 0.95$

15. Repeat problem 9 using the fast QR-RLS algorithm.

16. From equation (8.136) it is straightforward to show that

$$\begin{aligned} \underline{\mathbf{X}}(k) &= \mathcal{Q}^T(k) \begin{bmatrix} \mathbf{0} \\ \mathcal{U}(k) \end{bmatrix} \\ &= [\mathcal{Q}_u(k) \;\; \mathcal{Q}_d(k)] \begin{bmatrix} \mathbf{0} \\ \mathcal{U}(k) \end{bmatrix} \end{aligned}$$

where $\mathcal{Q}(k) = [\mathcal{Q}_u(k)\mathcal{Q}_d(k)]^T$.

(a) Using the relation above show that the elements of $\mathbf{x}_{q_2}(k)$ in (8.149) are given by

$$x_{q_2 i}(k) = [\mathbf{q}_{di}^T(k) \;\; 0]\mathbf{d}_f(k)$$

where $\mathbf{q}_{di}(k)$ is the ith column of $\mathcal{Q}_d(k)$.

(b) Show that the *a posteriori* error vector for a Nth-order forward predictor can be given by

$$\mathbf{e}_f(k, N+1) = \mathbf{d}_f(k) - \sum_{i=1}^{N+1} x_{q_2 i}(k) \begin{bmatrix} \mathbf{q}_{di}(k) \\ 0 \end{bmatrix}$$

(c) Can the expression above be generalized to represent the *a posteriori* error vector for a $(N-j)$th-order forward predictor? See the expression below

$$\mathbf{e}_f(k, N+1-j) = \mathbf{d}_f(k) - \sum_{i=j}^{N+1} x_{q_2 i}(k) \begin{bmatrix} \mathbf{q}_{di}(k) \\ 0 \end{bmatrix}$$

17. For the alternative fast QR-RLS algorithm, show that the elements of $\mathbf{r}(k+1)$ correspond to a normalized backward prediction *a posteriori* error

defined as

$$r_{N+1-i}(k) = \overline{e}_b(k, i+1) = \frac{e_b(k, i+1)}{||\mathbf{e}_{b,i}(k)||} = \frac{e_{bq_i}(k, i+1)}{||\mathbf{e}_{b,i}(k)||} \prod_{j=0}^{i-1} \cos\theta_j(k)$$

where $\prod_{j=0}^{-1} = 1$, and $e_b(k, i+1)$ is the *a posteriori* backward prediction error for a predictor of order i, with $i = 0, 1, \ldots$. Note that $||\mathbf{e}_{b,i}(k)||^2$ corresponds to $\xi^d_{b_{min}}(k, i+1)$ used in the lattice derivations.

18. Repeat problem 9 using the alternative fast QR-RLS algorithm.

9

ADAPTIVE IIR FILTERS

9.1 INTRODUCTION

Adaptive infinite impulse response (IIR) filters are those in which the zeros and poles of the filter can be adapted. For that benefit the adaptive IIR filters usually have adaptive coefficients on the transfer function numerator and denominator. Adaptive IIR filters present several advantages as compared with the adaptive FIR filters, including reduced computational complexity. If both have the same number of coefficients the frequency response of the IIR filter can approximate much better a desired characteristic. Therefore, an IIR filter in most cases requires fewer coefficients, mainly when the desired model has poles and zeros, or sharp resonances [1]. There are applications requiring hundreds and sometimes thousands of taps in an FIR filter where the use of an adaptive IIR filter is highly desirable. Among these applications are satellite-channel and mobile-radio equalizers, acoustic echo cancellation, etc.

The advantages of the adaptive IIR filters come with a number of difficulties, some of them not encountered in the adaptive FIR counterparts. The main drawbacks are: possible instability of the adaptive filter, slow convergence, and error surface with local minima or biased global minimum depending on the objective function [2].

In this chapter, several strategies to implement adaptive IIR filters will be discussed. First, adaptive IIR filters having as objective function the minimization of the mean-square output error is discussed. Several alternative structures are presented and some properties of the error surface are addressed. In addition, some algorithms based on the minimization of alternative objective functions

P. S. R. Diniz, *Adaptive Filtering*

are discussed. The algorithms are devised to avoid the multimodality inherent to the methods based on output error.

9.2 OUTPUT-ERROR IIR FILTERS

In the present section we examine strategies to reduce a function of the output error given by

$$\xi(k) = F(e(k)) \tag{9.1}$$

using an adaptive filter with IIR structure. The output error is defined by

$$e(k) = d(k) - y(k) \tag{9.2}$$

as illustrated in Fig. 9.1.a. As usual, an adaptation algorithm determines how the coefficients of the adaptive IIR filter should change in order to get the objective function reduced.

Let us consider that the adaptive IIR filter is realized using the direct form structure of Fig. 9.1.b. The signal information vector in this case is defined by

$$\phi(k) = [y(k-1)\ y(k-2)\ \ldots\ y(k-N)\ x(k)\ x(k-1)\ \ldots\ x(k-M)]^T \tag{9.3}$$

where N and M are the adaptive filter denominator and numerator orders, respectively.

The direct form adaptive filter can be characterized in time domain by the following difference equation

$$y(k) = \sum_{j=0}^{M} b_j(k)x(k-j) - \sum_{j=1}^{N} a_j(k)y(k-j) \tag{9.4}$$

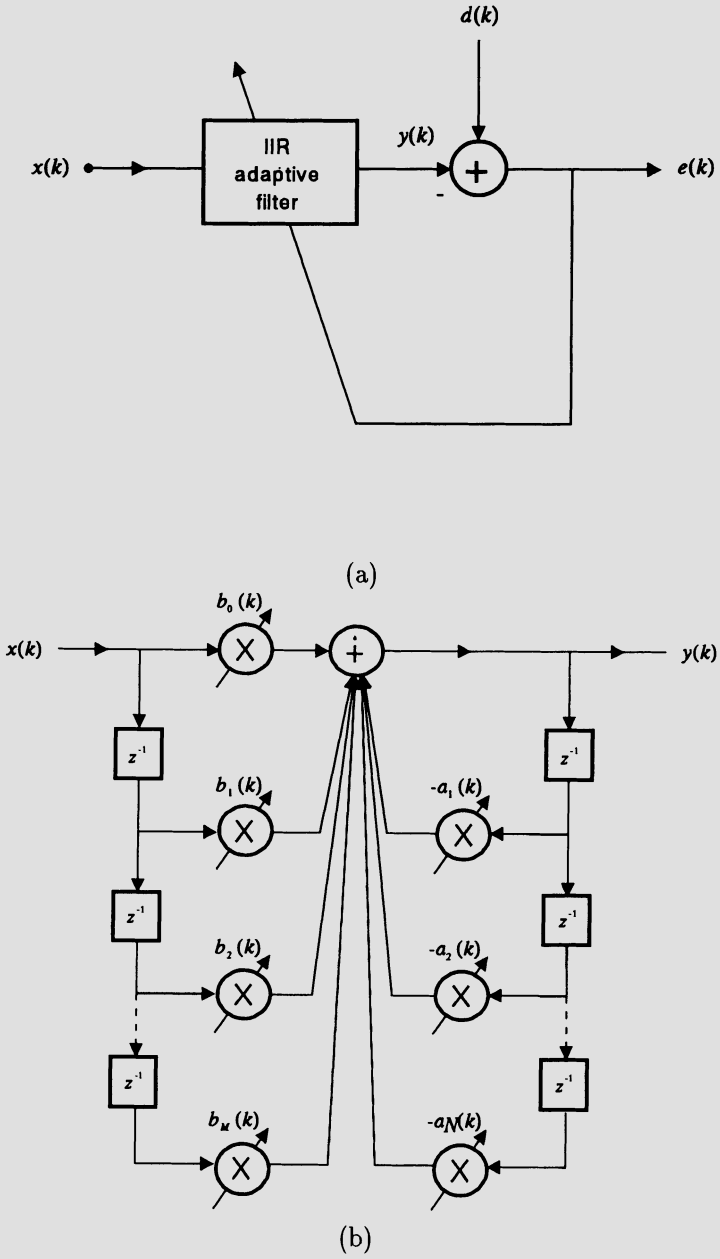


Figure 9.1 Adaptive IIR Filtering: (a) General configuration, (b) Adaptive IIR direct form realization.

In the system identification field [7], the above difference equation is in general described through polynomial operator as follows:

$$y(k) = \frac{B(k,q^{-1})}{A(k,q^{-1})}x(k) \tag{9.5}$$

where

$$\begin{aligned} B(k,q^{-1}) &= b_0(k) + b_1(k)q^{-1} + \cdots + b_M(k)q^{-M} \\ A(k,q^{-1}) &= 1 + a_1(k)q^{-1} + \cdots + a_N(k)q^{-N} \end{aligned}$$

and q^{-j} denotes a delay operation in a time domain signal of j samples, i.e., $q^{-j}x(k) = x(k-j)$. The difference equation (9.4) can also be rewritten in a vector form, which is more convenient for the algorithm description and implementation, as described below

$$y(k) = \boldsymbol{\theta}^T(k)\boldsymbol{\phi}(k) \tag{9.6}$$

where $\boldsymbol{\theta}(k)$ is the adaptive filter coefficient vector given by

$$\boldsymbol{\theta}(k) = [-a_1(k) \; -a_2(k) \; \ldots \; -a_N(k) \; b_0(k) \; b_1(k) \; \ldots \; b_M(k)]^T \tag{9.7}$$

In a given iteration k the adaptive filter transfer function can be expressed as follows:

$$\begin{aligned} H_k(z) &= z^{N-M}\frac{b_0(k)z^M + b_1(k)z^{M-1} + \cdots + b_{M-1}(k)z + b_M(k)}{z^N + a_1(k)z^{N-1} + \cdots + a_{N-1}(k)z + a_N(k)} \\ &= z^{N-M}\frac{N_k(z)}{D_k(z)} \end{aligned} \tag{9.8}$$

Given the objective function $F(e(k))$, the gradient vector required to be employed in the adaptive algorithm is given by

$$\boldsymbol{g}(k) = \frac{\partial F(e(k))}{\partial e(k)}\frac{\partial e(k)}{\partial \boldsymbol{\theta}(k)} \tag{9.9}$$

where $e(k)$ is the output error.

The first derivative in the above gradient equation is a scalar dependent on the objective function, while the second derivative is a vector whose elements are obtained by

$$\frac{\partial e(k)}{\partial a_i(k)} = \frac{\partial(d(k) - y(k))}{\partial a_i(k)} = -\frac{\partial y(k)}{\partial a_i(k)}$$

for $i = 1, 2, \ldots, N$, and

$$\frac{\partial e(k)}{\partial b_j(k)} = \frac{\partial(d(k) - y(k))}{\partial b_j(k)} = -\frac{\partial y(k)}{\partial b_j(k)} \tag{9.10}$$

for $j = 1, 2, \ldots, M$, where it was used the fact that the desired signal $d(k)$ is not dependent of the adaptive filter coefficients.

The error derivatives with respect to the filter coefficients can be calculated from the difference equation (9.4) as follows:

$$\frac{\partial y(k)}{\partial a_i(k)} = -y(k-i) - \sum_{j=1}^{N} a_j(k) \frac{\partial y(k-j)}{\partial a_i(k)}$$

for $i = 1, 2, \ldots, N$, and

$$\frac{\partial y(k)}{\partial b_j(k)} = x(k-j) - \sum_{i=1}^{N} a_i(k) \frac{\partial y(k-i)}{\partial b_j(k)} \tag{9.11}$$

for $j = 0, 1, \ldots, M$. The partial derivatives of $y(k-i)$ with respect to the coefficients, for $i = 1, 2, \ldots, N$, are different from zero because the adaptive filter is recursive. As a result, the present coefficients $a_i(k)$ and $b_j(k)$ are dependent on the past output samples $y(k-i)$. The precise evaluation of these partial derivatives is a very difficult task, and does not have a simple implementation. However, as first pointed out in [4] and [5], if small step sizes are used in the coefficient updating, the following approximations are valid

$$a_i(k) \approx a_i(k-j) \quad \text{for } i, j = 1, 2, \ldots, N$$

and

$$b_j(k) \approx b_j(k-i) \quad \text{for } j = 0, 1, \ldots, M \text{ and } i = 1, 2, \ldots, N \tag{9.12}$$

As a consequence, equations (9.11) can be rewritten as

$$\frac{\partial y(k)}{\partial a_i(k)} \approx -y(k-i) - \sum_{j=1}^{N} a_j(k) \frac{\partial y(k-j)}{\partial a_i(k-j)}$$

for $i = 1, 2, \ldots, N$, and

$$\frac{\partial y(k)}{\partial b_j(k)} \approx x(k-j) - \sum_{i=1}^{N} a_i(k) \frac{\partial y(k-i)}{\partial b_j(k-i)} \tag{9.13}$$

for $j = 0, 1, \ldots, M$. Note that these equations are standard difference equations.

The equations above can be implemented by all-pole filters having as input signals $-y(k-i)$ and $x(k-j)$ for the first and second set of equations, respectively. The implementation of the derivative signals of equations (9.13) is depicted in Fig. 9.2. The all-pole sections realization can be performed through IIR direct form structure, with transfer function given by

$$S^{a_i}(z) = \mathcal{Z}\left[\frac{\partial y(k)}{\partial a_i(k)}\right] = \frac{-z^{N-i}}{D_k(z)} Y(z)$$

for $i = 1, 2, \ldots, N$, and

$$S^{b_j}(z) = \mathcal{Z}\left[\frac{\partial y(k)}{\partial b_i(k)}\right] = \frac{z^{N-j}}{D_k(z)} X(z) \tag{9.14}$$

for $j = 0, 1, \ldots, M$ respectively, where $\mathcal{Z}[\cdot]$ denotes the $\mathcal{Z}$-transform of $[\cdot]$.

The amount of computation spent to obtain the derivatives is relatively high, as compared with the adaptive filter computation itself. A considerable reduction in the amount of computation can be achieved, if it is considered that the coefficients of the adaptive filter denominator polynomial are slowly varying, such that

$$D_k(z) \approx D_{k-i}(z) \qquad \text{for } i = 1, 2, \ldots, \max(N, M) \tag{9.15}$$

where $\max(a, b)$ denotes maximum between a and b. The interpretation is that the denominator polynomial is kept almost constant for a number of iterations. With this approximation, it is possible to eliminate the duplicating all-pole filters of Fig. 9.2, and replace them by a single all-pole in front of the two sets of delays as depicted in Fig. 9.3.a. In addition, if the recursive part of the adaptive filter is implemented before the numerator part, one more all-pole section can be saved as illustrated in Fig. 9.3.b [6].

Note that in the time domain the approximations of equation (9.15) imply the following relations

$$\frac{\partial y(k)}{\partial a_i(k)} \approx q^{-i+1} \frac{\partial y(k)}{\partial a_1(k)}$$

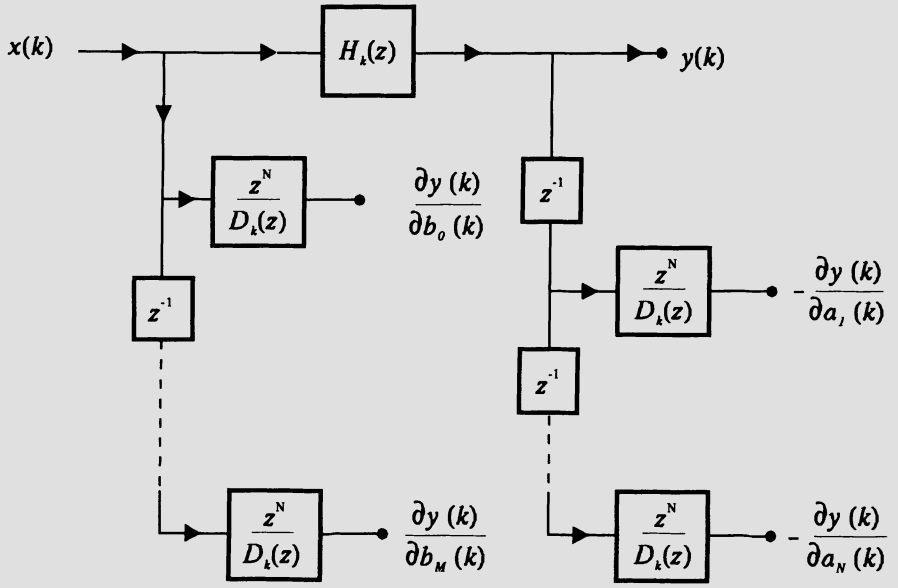


Figure 9.2 Derivative implementation.

for $i = 1, 2, \ldots, N$, and

$$\frac{\partial y(k)}{\partial b_j(k)} \approx q^{-j} \frac{\partial y(k)}{\partial b_0(k)} \tag{9.16}$$

for $j = 0, 1, \ldots, M$.

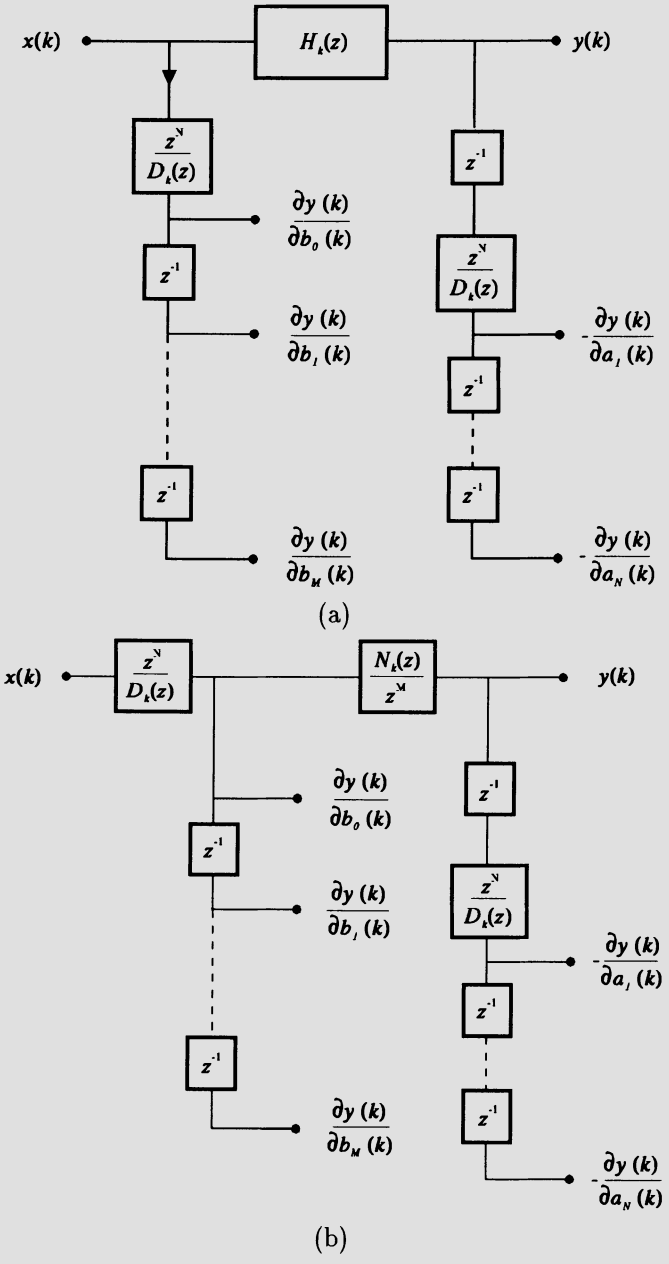


Figure 9.3 Simplified derivative implementation: (a) Simplification I, (b) Simplification II.

9.3 GENERAL DERIVATIVE IMPLEMENTATION

The derivatives of the output signal as related to the adaptive filter coefficients are always required to generate the gradient vector that is used in most adaptive algorithms. These derivatives can be obtained in a systematic form by employing a sensitivity property of digital filters with fixed coefficients [1], if the adaptive filter coefficients are slowly varying as assumed in equation (9.12).

Refer to Fig. 9.4.a, where the multiplier with coefficient c is an internal multiplier of a digital filter with fixed coefficients. A good measure of how the digital filter characteristics change when the value of c changes is the sensitivity function defined as the partial derivative of the digital filter transfer function $H(z)$ as related to the coefficient c. It is well known from classical digital filtering theory [1] that the partial derivative of the digital filter transfer function, with respect to a given multiplier coefficient c, is given by the product of the transfer function from the filter input to the multiplier input and the transfer function from the multiplier output to the filter output, that is

$$S^c(z) = H_{13}(z) \cdot H_{42}(z) \tag{9.17}$$

Fig. 9.4.b illustrates the derivative implementation. It can be noted that implementation of the derivatives for the direct form structure shown in Fig. 9.2 can be obtained by employing equation (9.17). In the time domain, the filtering operation performed in the implementation of Fig. 9.4.b is given by

$$\frac{\partial y(k)}{\partial c} = h_{13}(k) * h_{42}(k) * x(k) \tag{9.18}$$

where $*$ denotes convolution and $h_{ij}(k)$ is the impulse response related to $H_{ij}(z)$. When the digital filter coefficients are slowly varying, the desired derivatives can be derived as in Fig. 9.4 for each adaptive coefficient. In this case, only an approximated derivative is obtained

$$\frac{\partial y(k)}{\partial c(k)} \approx h_{13k}(k) * h_{42k}(k) * x(k) \tag{9.19}$$

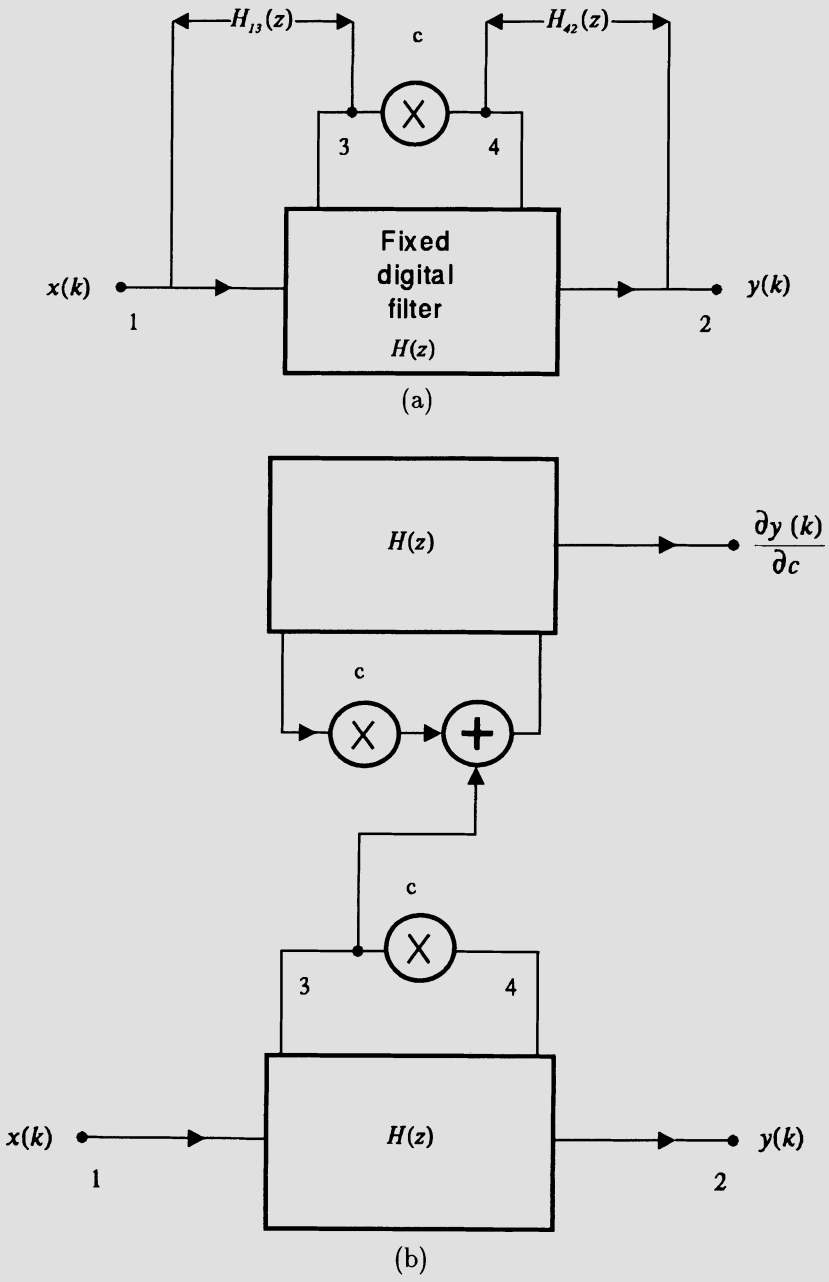


Figure 9.4 General derivative implementation: (a) General structure, (b) Derivative implementation.

9.4 ADAPTIVE ALGORITHMS

In this section, the adaptation algorithms used in IIR adaptive filtering are described. In particular, we present the RLS, the Gauss-Newton, and the gradient-based algorithms.

9.4.1 Recursive least-squares algorithm

A possible objective function for adaptive IIR filtering based on output error is the least-squares function

$$\xi^d(k) = \sum_{i=0}^{k} \lambda^{k-i} e^2(i) = \sum_{i=0}^{k} \lambda^{k-i} [d(i) - \boldsymbol{\theta}^T(k)\boldsymbol{\phi}(i)]^2 \tag{9.20}$$

The forgetting factor λ is usually chosen in the range $0 << \lambda < 1$, with the objective of turning the distant past information increasingly negligible. By differentiating $\xi^d(k)$ with respect to $\boldsymbol{\theta}(k)$, it follows that

$$\begin{aligned} 2\mathbf{p}_D(k) &= \frac{\partial \xi^d(k)}{\partial \boldsymbol{\theta}(k)} = -2\sum_{i=0}^{k} \lambda^{k-i} \boldsymbol{\psi}(i)[d(i) - \boldsymbol{\theta}^T(k)\boldsymbol{\phi}(i)] \\ &= -2\boldsymbol{\psi}(k)e(k) + 2\lambda \frac{\partial \xi^d(k-1)}{\partial \boldsymbol{\theta}(k)} \end{aligned} \tag{9.21}$$

where the vector $\boldsymbol{\psi}(k)$ is the derivative of $e(k)$ with respect to $\boldsymbol{\theta}(k)$, i.e.,

$$\boldsymbol{\psi}(k) = \frac{\partial e(k)}{\partial \boldsymbol{\theta}(k)} = -\frac{\partial y(k)}{\partial \boldsymbol{\theta}(k)} \tag{9.22}$$

The second-derivative matrix of $\xi^d(k)$ with respect to $\boldsymbol{\theta}(k)$ is then given by

$$2\mathbf{R}_D(k) = -2\lambda \mathbf{R}_D(k-1) + 2\boldsymbol{\psi}(k)\boldsymbol{\psi}^T(k) - \frac{\partial^2 y(k)}{\partial \boldsymbol{\theta}^2(k)} e(k) \tag{9.23}$$

Now, several assumptions are made to generate a recursive algorithm. First, the adaptive filter parameters are considered to be updated by

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) - \mathbf{R}_D^{-1}(k)\mathbf{p}_D(k) \tag{9.24}$$

As can be noted from equations (9.21) and (9.22), the calculations of the last terms in both $\mathbf{R}_D(k)$ and $\mathbf{p}_D(k)$ require a knowledge of the signal information

vector since the beginning of the algorithm operation, namely $\psi(k)$ for $i < k$. However, if the algorithm step sizes, i.e., the elements of $|\boldsymbol{\theta}(k+1) - \boldsymbol{\theta}(k)|$, are considered small, then

$$\frac{\partial \xi^d(k-1)}{\partial \boldsymbol{\theta}(k)} \approx 0 \tag{9.25}$$

assuming that the vector $\boldsymbol{\theta}(k)$ is the optimal estimate for the parameters at the instant $k-1$. This conclusion can be drawn by approximating $\xi^d(k-1)$ by a Taylor series around $\boldsymbol{\theta}(k-1)$ and considering only the first-order term [7]. Also, close to the minimum solution, the output error $e(k)$ can be considered approximately a white noise, if the measurement noise is also a white noise and independent of $\frac{\partial^2 y(k)}{\partial \boldsymbol{\theta}^2(k)}$. This assumption allows us to consider the expected value of the last term in equation (9.23) negligible as compared to the remaining terms.

Applying the approximations above, an RLS algorithm for adaptive IIR filtering is derived in which the basic steps are:

$$e(k) = d(k) - \boldsymbol{\theta}^T(k)\boldsymbol{\phi}(k) \tag{9.26}$$

$$\boldsymbol{\psi}(k) = \frac{\partial y(k)}{\partial \boldsymbol{\theta}(k)} \tag{9.27}$$

$$\mathbf{S}_D(k+1) = \frac{1}{\lambda}\left[\mathbf{S}_D(k) - \frac{\mathbf{S}_D(k)\boldsymbol{\psi}(k)\boldsymbol{\psi}^T(k)\mathbf{S}_D(k)}{\lambda + \boldsymbol{\psi}^T(k)\mathbf{S}_D(k)\boldsymbol{\psi}(k)}\right] \tag{9.28}$$

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + e(k)\mathbf{S}_D(k+1)\boldsymbol{\psi}(k) \tag{9.29}$$

The description of the RLS adaptive IIR filter is given in Algorithm 9.1.

Note that the primary difference between the RLS algorithm for FIR and IIR adaptive filtering relies on the signal information vector, $\boldsymbol{\psi}(k)$, that in the IIR case is obtained through a filtering operation while in the FIR case it corresponds to the input signal vector $\mathbf{x}(k)$.

9.4.2 The Gauss-Newton algorithm

Consider as objective function the mean-square error (MSE) defined as

$$\xi = E[e^2(k)] \tag{9.30}$$

In the Gauss-Newton algorithm, the minimization of the objective function is obtained by performing searches in the Newton direction, using estimates of the inverse Hessian matrix and the gradient vector.

Algorithm 9.1

OE Algorithm, RLS Version

Initialization

$a_i(k)=b_i(k)=e(k)=0$

$y(k)=x(k)=0\,,\ k<0$

$\mathbf{S}_D(0)=\delta^{-1}\mathbf{I}$

Definition

$\boldsymbol{\psi}^T(k)=[-y'(k-1)\ldots-y'(k-N)\ -x'(k)\ -x'(k-1)\ldots-x'(k-M)]$

For each $x(k)$, $d(k)$, $k \geq 0$, do

$y(k)=\boldsymbol{\phi}^T(k)\boldsymbol{\theta}(k)$

$y'(k)=-y(k)-\sum_{i=1}^{N} a_i(k)y'(k-i)$

$x'(k)=x(k)-\sum_{i=1}^{M} b_i(k)x'(k-i)$

$e(k)=d(k)-y(k)$

$$\mathbf{S}_D(k+1)=\frac{1}{\lambda}\left[\mathbf{S}_D(k)-\frac{\mathbf{S}_D(k)\boldsymbol{\psi}(k)\boldsymbol{\psi}^T(k)\mathbf{S}_D(k)}{\lambda+\boldsymbol{\psi}^T(k)\mathbf{S}_D(k)\boldsymbol{\psi}(k)}\right]$$

$\boldsymbol{\theta}(k+1)=\boldsymbol{\theta}(k)+\mathbf{S}_D(k+1)\boldsymbol{\psi}(k)e(k)$

Stability test

□

The gradient vector is calculated as follows:

$$\frac{\partial \xi}{\partial \boldsymbol{\theta}(k)} = E[2e(k)\boldsymbol{\psi}(k)] \tag{9.31}$$

where

$$\boldsymbol{\psi}(k) = \frac{\partial e(k)}{\partial \boldsymbol{\theta}(k)}$$

The Hessian matrix is then given by

$$\frac{\partial^2 \xi}{\partial \boldsymbol{\theta}^2(k)} = 2E\left[\boldsymbol{\psi}(k)\boldsymbol{\psi}^T(k) + \frac{\partial^2 e(k)}{\partial \boldsymbol{\theta}^2(k)}e(k)\right] \tag{9.32}$$

where the expected value of the second term in the above equation is approximately zero, since close to a solution the output error $e(k)$ is "almost" a white

noise independent of the following term

$$\frac{\partial^2 e(k)}{\partial \boldsymbol{\theta}^2(k)} = -\frac{\partial^2 y(k)}{\partial \boldsymbol{\theta}^2(k)}$$

The determination of the gradient vector and the Hessian matrix requires statistical expectation calculations. In order to derive a recursive algorithm, estimates of the gradient vector and Hessian matrix have to be used. For the gradient vector, the most commonly used estimation is the stochastic gradient given by

$$\frac{\partial \hat{\xi}}{\partial \boldsymbol{\theta}(k)} = 2e(k)\boldsymbol{\psi}(k) \tag{9.33}$$

where $\hat{\xi}$ is an estimate of ξ. Such approximation was also used in the derivation of the LMS algorithm. The name stochastic gradient originates from the fact that the estimates point to random directions around the true gradient direction.

The Hessian estimate can be generated by employing a weighted summation as follows:

$$\begin{aligned} \hat{\mathbf{R}}(k+1) &= \alpha\boldsymbol{\psi}(k)\boldsymbol{\psi}^T(k) + \alpha\sum_{i=0}^{k-1}(1-\alpha)^{k-i}\boldsymbol{\psi}(i)\boldsymbol{\psi}^T(i) \\ &= \alpha\boldsymbol{\psi}(k)\boldsymbol{\psi}^T(k) + (1-\alpha)\hat{\mathbf{R}}(k) \end{aligned} \tag{9.34}$$

where α is a small factor chosen in the range $0 < \alpha < 0.1$. By taking the expected value on both sides of the equation above and assuming that $k \to \infty$, it follows that

$$\begin{aligned} E[\hat{\mathbf{R}}(k+1)] &= \alpha\sum_{i=0}^{k}(1-\alpha)^{k-i}E[\boldsymbol{\psi}(i)\boldsymbol{\psi}^T(i)] \\ &\approx E[\boldsymbol{\psi}(k)\boldsymbol{\psi}^T(k)] \end{aligned} \tag{9.35}$$

Applying the approximation discussed and the matrix inversion lemma to calculate the inverse of $\hat{\mathbf{R}}(k+1)$, i.e., $\hat{\mathbf{S}}(k+1)$, the Gauss-Newton algorithm for IIR adaptive filtering is derived, consisting of the following basic steps

$$e(k) = d(k) - \boldsymbol{\theta}^T(k)\boldsymbol{\phi}(k) \tag{9.36}$$

$$\boldsymbol{\psi}(k) = \frac{\partial e(k)}{\partial \boldsymbol{\theta}(k)} \tag{9.37}$$

$$\hat{\mathbf{S}}(k+1) = \frac{1}{1-\alpha}\left[\hat{\mathbf{S}}(k) - \frac{\hat{\mathbf{S}}(k)\boldsymbol{\psi}(k)\boldsymbol{\psi}^T(k)\hat{\mathbf{S}}(k)}{\frac{1-\alpha}{\alpha} + \boldsymbol{\psi}^T(k)\hat{\mathbf{S}}(k)\boldsymbol{\psi}(k)}\right] \tag{9.38}$$

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + \mu\hat{\mathbf{S}}(k+1)\boldsymbol{\psi}(k)e(k) \tag{9.39}$$

where μ is the convergence factor. In most cases, μ is chosen approximately equal to α.

In the updating of the $\hat{\mathbf{R}}(k)$ matrix, the factor $(1-\alpha)$ plays the role of a forgetting factor that determines the effective memory of the algorithm when computing the present estimate. The closer α is to zero the more important is the past information, in other words, the longer is the memory of the algorithm.

9.4.3 Gradient-based algorithm

If in the Gauss-Newton algorithm, the estimate of the Hessian matrix is replaced by the identity matrix, the resulting basic algorithm is given by

$$e(k) = d(k) - \boldsymbol{\theta}^T(k)\boldsymbol{\phi}(k) \tag{9.40}$$

$$\boldsymbol{\psi}(k) = \frac{\partial e(k)}{\partial \boldsymbol{\theta}(k)} \tag{9.41}$$

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + \mu\boldsymbol{\psi}(k)e(k) \tag{9.42}$$

These are the steps of a gradient-based algorithm for IIR filtering. The computational complexity is much lower in gradient-based algorithm than in the Gauss-Newton algorithm. With the latter, however, faster convergence is in general achieved.

9.5 ALTERNATIVE ADAPTIVE FILTER STRUCTURES

The direct form structure is historically the most widely used realization for the IIR adaptive filter. The main advantages of the direct form are the minimum number of multiplier coefficients required to realize a desired transfer function and the computationally efficient implementation for the gradient which is possible under the assumption that the denominator coefficients are slowly varying, as illustrated in Fig. 9.3. On the other hand, the stability monitoring of the direct form is difficult because it requires either the factorization of a high-order denominator polynomial in each algorithm step or the use of a sophisticated stability test. In addition, the coefficient sensitivities and output quantization noise are known to be high in the direct form [1].

Alternate solutions are the cascade and parallel realizations using first- or second-order sections as building blocks [8]-[9]. Also, the lattice structures are popular in the implementation of adaptive filters [11]-[17]. All these structures allow easy stability monitoring while the parallel form appears to be most efficient in the gradient computation. The standard parallel, however, may converge slowly if two poles approach each other, as will be discussed later and, when a Newton-based algorithm is employed, the estimated Hessian matrix becomes ill-conditioned bringing convergence problems. This problem can be alleviated by applying a preprocessing to the input signal [9]-[10].

9.5.1 Cascade Form

Any Nth-order transfer function can be realized by connecting several first- or second-order sections in series, generating the so-called cascade form. Here we consider that all subfilters are second-order sections without loss of generality, and if an odd-order adaptive filter is required we add a single first-order section. Also, only filters with real multiplier coefficients are discussed. The cascade realization transfer function is given by

$$H_k(z) = z^{N-M} \prod_{i=1}^{m} \frac{b_{0i} z^2 + b_{1i}(k) z + b_{2i}(k)}{z^2 + a_{1i}(k) z + a_{2i}(k)} = \prod_{i=1}^{m} H_{ki}(z) \tag{9.43}$$

where m denotes the number of sections.

The parameter vector in the cascade form is

$$\begin{aligned} \boldsymbol{\theta}(k) \;=\; & [-a_{11}(k) \; -a_{21}(k) \; b_{01}(k) \; b_{11}(k) \; b_{21}(k) \\ & \ldots - a_{1m}(k) \; -a_{2m}(k) \; b_{0m}(k) \; b_{1m}(k) \; b_{2m}(k)]^T \end{aligned} \tag{9.44}$$

The transfer function derivatives as related to the multiplier coefficients can be generated by employing the general result of Fig. 9.4. Fig. 9.5 depicts the cascade realization along with the generation of the derivative signals of interest, where the sections were realized through the direct form of Fig. 9.1.

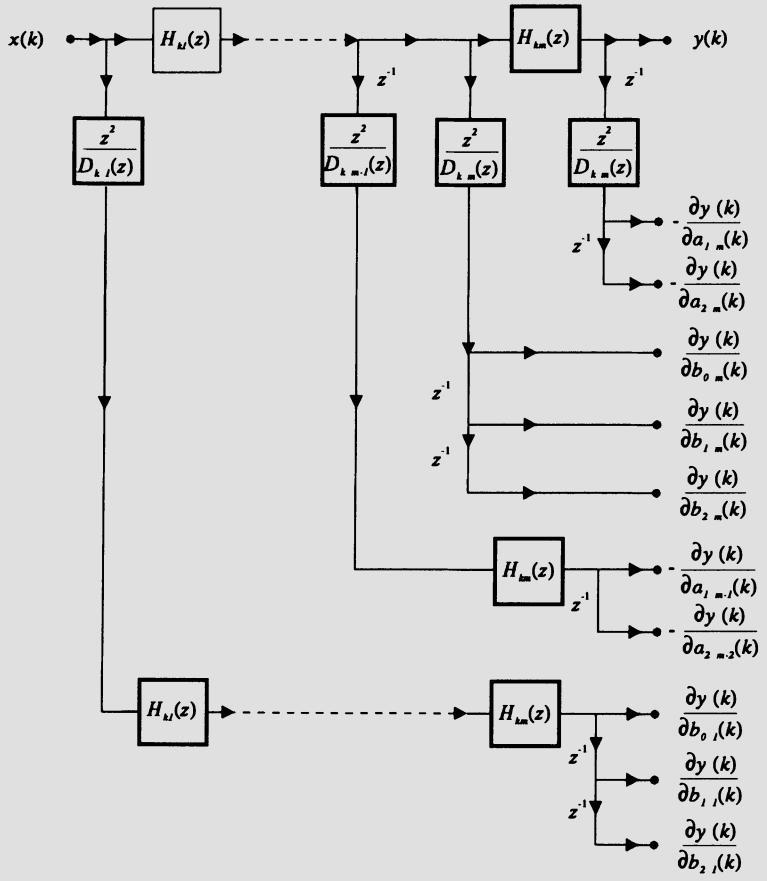


Figure 9.5 Cascade form.

Any alternative second-order section can be used in the cascade form and the appropriate choice depends on a tradeoff between quantization effects, hardware resources, computation time, and other factors. The main drawbacks of the cascade form are the amount of extra computations required to generate the gradients, and the manifolds (see sections 9.6 and 9.7) generated on the error surface which may result in slow convergence of the gradient-based algorithms.

9.5.2 Lattice Structure

In this subsection we discuss the lattice algorithm starting from its realization. Although this might appear to be a recipe approach, the development presented here allows us to access the nice properties of the lattice realization. The book by Regalia [53] provides a detailed presentation of the various forms of lattice realization.

The two-multiplier lattice structure [11]-[16] for IIR filters is depicted in Fig. 9.6 with a sample of gradient computation. The coefficients $\kappa_i(k)$ in the recursive part of the structure are called reflection coefficients. The internal signals $\hat{f}_i(k)$ and $\hat{b}_i(k)$ are the forward and backward residuals, respectively. These internal signals are calculated as follows:

$$\begin{aligned} \hat{f}_{N+1}(k) &= x(k) \\ \hat{f}_{N-i}(k) &= \hat{f}_{N-i+1}(k) - \kappa_{N-i}(k)\hat{b}_{N-i}(k) \\ \hat{b}_{N-i+1}(k+1) &= \kappa_{N-i}(k)\hat{f}_{N-i}(k) + \hat{b}_{N-i}(k) \end{aligned}$$

for $i = 0, 1, \ldots, N$, and

$$\hat{b}_0(k+1) = \hat{f}_0(k) \tag{9.45}$$

The zero placement is implemented by a weighted sum of the backward residuals $\hat{b}_i(k)$, generating the filter output according to

$$y(k) = \sum_{i=0}^{N+1} \hat{b}_i(k+1)v_i(k) \tag{9.46}$$

where $v_i(k)$, for $i = 0, 1, \ldots, N+1$, are the output coefficients.

The derivatives of the filter output $y(k)$ with respect to the output tap coefficients $v_i(k)$ are given by the backward residuals $\hat{b}_i(k+1)$. On the other hand,

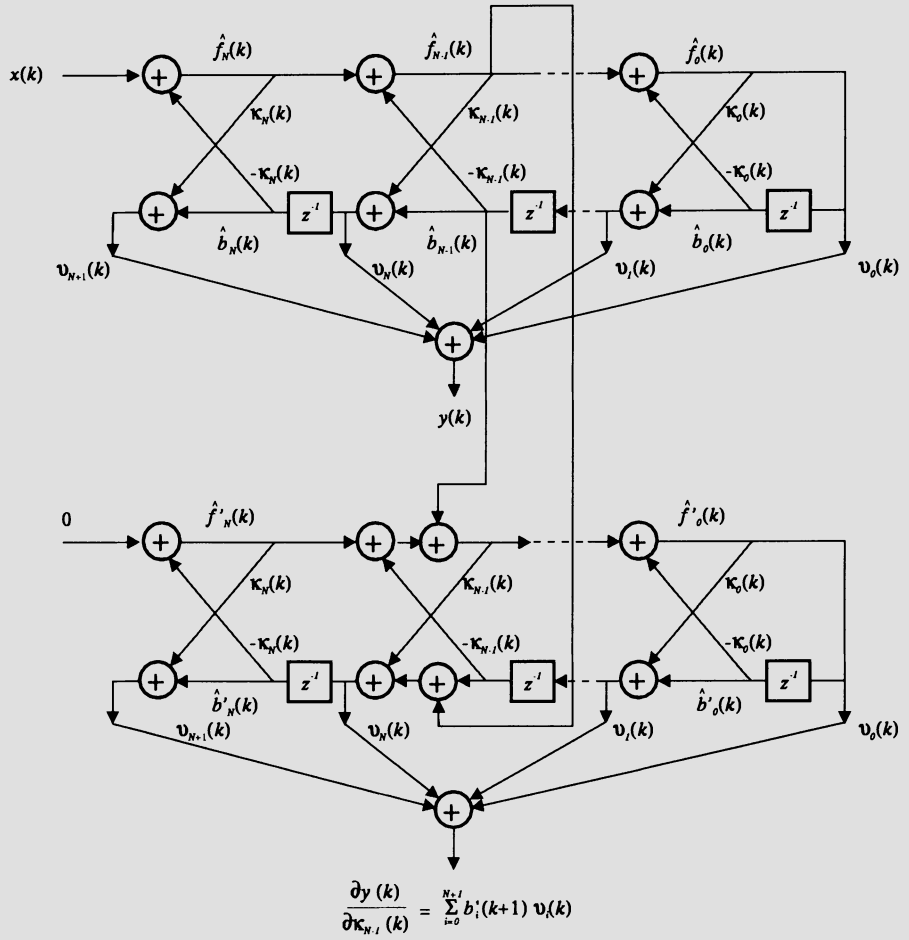


Figure 9.6 Lattice structure including a sample of gradient computation.

the derivatives of $y(k)$ as related to the reflection multiplier coefficients $\kappa_i(k)$ require one additional lattice structure for each $\kappa_i(k)$. In Fig. 9.6, the extra lattice required to calculate $\frac{\partial y(k)}{\partial \kappa_{N-1}(k)}$ is shown for illustration. For each derivative $\frac{\partial y(k)}{\partial \kappa_j(k)}$ the following algorithm must be used

$$\begin{aligned} \hat{f}'_{N+1}(k) &= 0 \\ \text{If } i &\neq N-j \\ \hat{f}'_{N-i}(k) &= \hat{f}'_{N-i+1}(k) - \kappa_{N-i}(k)\hat{b}'_{N-i}(k) \end{aligned}$$

$$
\begin{aligned}
\hat{b}'_{N-i+1}(k+1) &= \kappa_{N-i}(k)\hat{f}'_{N-i}(k) + \hat{b}'_{N-i}(k) \\
\text{for } i &= 0, 1, \ldots, N-j-1, N-j+1, \ldots, N \\
\text{If } i &= N-j \\
\hat{f}'_j(k) &= \hat{f}'_{j+1}(k) - \kappa_j(k)\hat{b}'_j(k) - \hat{b}_j(k) \\
\hat{b}'_{j+1}(k+1) &= \kappa_j(k)\hat{f}'_j(k) + \hat{b}'_j(k) + \hat{f}_j(k) \\
\hat{b}'_o(k+1) &= \hat{f}_o(k) \\
&\text{Then} \\
\frac{\partial y(k)}{\partial \kappa_j(k)} &= \sum_{i=0}^{N+1} \hat{b}'_i(k+1)v_i(k)
\end{aligned}
\tag{9.47}
$$

The main desirable feature brought about by the lattice IIR realization is the simple stability test. The stability requires only that reflection coefficients $\kappa_i(k)$ be maintained with modulus less than one [15]. However, the gradient computations are extremely complex, and of order N^2 in terms of multiplication count. Recently, an approach for the gradient computations with order N multiplications and divisions was proposed [14], which is still more complex than for the direct form realization. It should be noticed that in the direct form, all the signals at the multipliers input are delayed versions of each other, and the transfer function from the multipliers output to the filter output are the same. These facts make the gradient computational complexity in the direct form low. The lattice IIR realization does not have these features.

When the two-multiplier lattice structure is realizing a transfer function with poles close to the unit circle, the internal signals may present a large dynamic range, resulting in poor performance due to quantization effects. In this case, the normalized lattice [17] is a better choice despite its higher computational complexity. There are alternative lattice structures, such as the two-multiplier with distinct reflection coefficients and the one-multiplier structures [13], that can also be employed in adaptive filtering. For all these options the stability test is trivial, retaining the main feature of the two-multiplier lattice structure.

9.5.3 Parallel Form

In the parallel realization, the transfer function is realized by a parallel connection of sections as shown in Fig. 9.7. The sections are in most of the cases of first- or second-order, turning the stability test trivial. The transfer function when second-order sections are employed is given by

$$H_k(z) = \sum_{i=0}^{m-1} \frac{b_{0i}(k)z^2 + b_{1i}(k)z + b_{2i}(k)}{z^2 + a_{1i}(k)z + a_{2i}(k)} \tag{9.48}$$

The parameter vector for the parallel form is

$$\begin{aligned}\boldsymbol{\theta}(k) = [&-a_{10}(k) \; -a_{20}(k) \; b_{00}(k) \; b_{10}(k) \; b_{20}(k) \\ &\ldots - a_{1\,m-1}(k) \; -a_{2\,m-1}(k) \; b_{0\,m-1}(k) \; b_{1\,m-1}(k) \; b_{2\,m-1}(k)]^T\end{aligned} \tag{9.49}$$

The transfer function derivatives as related to the multiplier coefficients in the parallel form are simple to calculate because they depend on the derivative of the individual section transfer function with respect to the multiplier coefficients belonging to that section. Basically, the technique of Fig. 9.4 can be applied to each section individually.

Because the interchange of sections in the parallel form does not alter the transfer function, there are $m!$ global minimum points each located in separate subregions of the MSE surface. These subregions are separated by boundaries that are reduced-order manifolds as will be discussed in section 9.7. These boundaries contain saddle points and if the filter parameters are initialized on a boundary, the convergence rate is most probably slow. Consider that the internal signals cross-correlation matrix is approximately estimated

$$\hat{\mathbf{R}}(k+1) = \alpha \sum_{i=0}^{k} (1-\alpha)^{k-i} \boldsymbol{\psi}(i)\boldsymbol{\psi}^T(i) \tag{9.50}$$

when k is large. In this case, if the sections coefficients are identical the information vector consists of a set of identical subvectors $\boldsymbol{\psi}(i)$, which in turn makes $\hat{\mathbf{R}}(k+1)$ ill-conditioned. The above discussion suggests that the sections in the parallel realization should be initialized differently, although there is no guarantee that this will avoid the ill-conditioning problems.

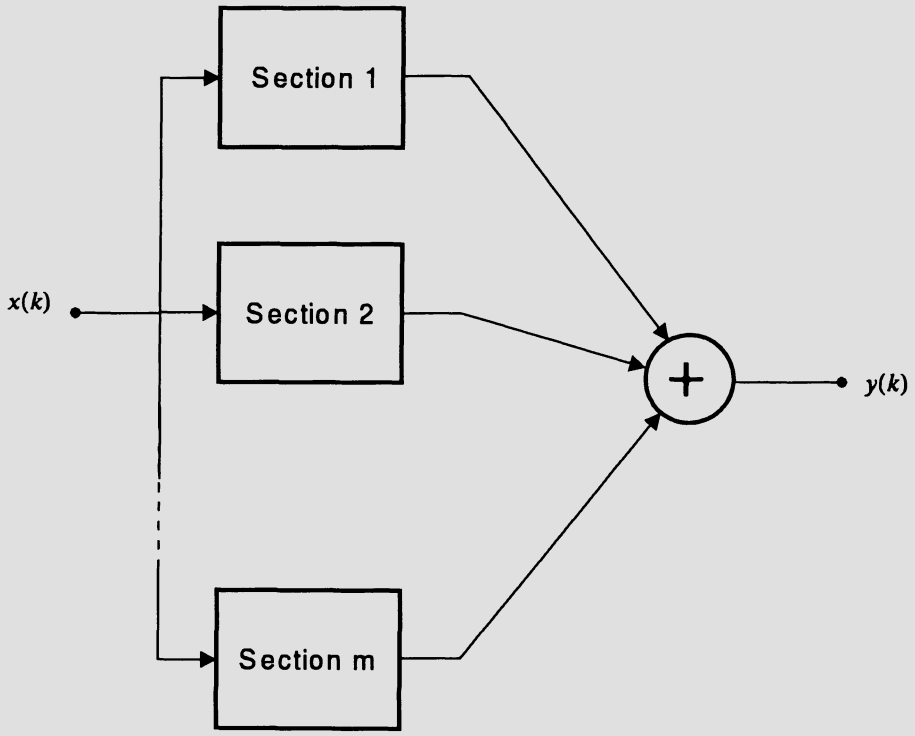


Figure 9.7 Parallel form.

9.5.4 Frequency-Domain Parallel Structure

A possible alternative parallel realization first proposed in [9] incorporates a preprocessing to the input signal using a discrete-time Fourier transform, generating m signals that are individually applied as input to first-order complex-coefficients sections. With this strategy, the matrix $\hat{\mathbf{R}}(k)$ is more unlikely to become ill-conditioned. Also, it is more difficult for a gradient-based algorithm to get stuck on a reduced-order manifold, resulting in faster convergence. The parallel realization can also be implemented using a real-coefficient transform for the preprocessing, and second-order sections.

The frequency-domain parallel structure is illustrated in Fig. 9.8, where $d(k)$ is the reference signal, $x(k)$ is the input signal, $n(k)$ is an additive noise source, and $y(k)$ is the output. The ith parallel section is represented by the transfer

function

$$H_i(z) = \frac{b_{0i}(k)z^2 + b_{1i}(k)z + b_{2i}(k)}{z^2 - a_{1i}(k)z - a_{2i}(k)} \qquad k = 0, 1, \ldots, m-1 \qquad (9.51)$$

where $a_{1i}(k)$, $a_{2i}(k)$, $b_{0i}(k)$, $b_{1i}(k)$, and $b_{2i}(k)$ are adjustable real coefficients. The inputs of the filter sections are preprocessed as shown in Fig. 9.8.

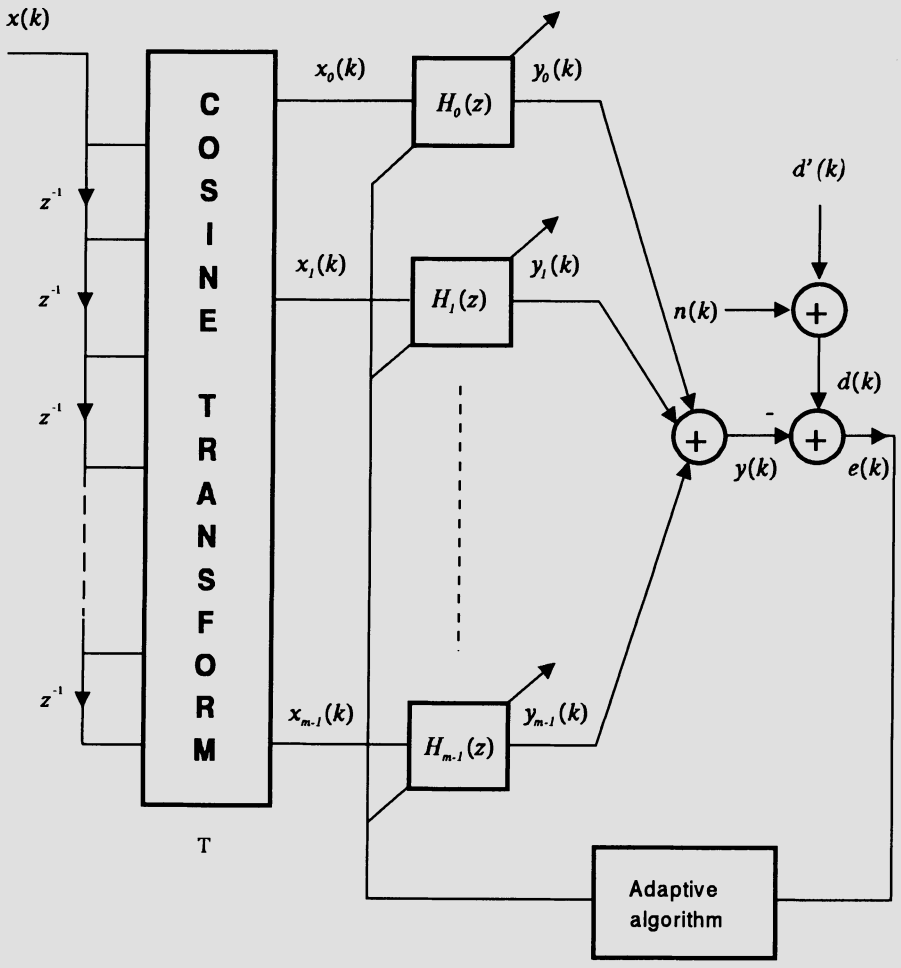


Figure 9.8 Real coefficients frequency-domain adaptive filter.

The purpose of preprocessing in Fig. 9.8 is to generate a set of uncorrelated signals $x_1(k), x_2(k), \ldots, x_m(k)$ in order to reduce the probability that two or more

sections converge to the same solution, to simplify the adaptation algorithm, and to improve the rate of convergence.

On employing the discrete-time cosine transform (DCT), the input signals to the subfilters in Fig. 9.8 are given by

$$x_0(k) = \frac{\sqrt{2}}{m} \sum_{l=0}^{m-1} x(k-l)$$

and

$$x_i(k) = \sqrt{\frac{2}{m}} \sum_{l=0}^{m-1} x(k-l) \cos[\pi i(2l+1)/(2m)] \tag{9.52}$$

The transfer function from the input to the outputs of the DCT preprocessing filter (or prefilter) can be described through the recursive frequency-domain description given by

$$T_i(z) = \frac{k_0}{m} \cos\tau \frac{[z^m - (-1)^i](z-1)}{z^{m-1}[z^2 - (2\cos 2\tau)z + 1]} \tag{9.53}$$

where

$$k_0 = \begin{cases} \sqrt{2} & \text{if } i = 0 \\ \sqrt{2m} & \text{if } i = 1, 2, \ldots, m-1 \end{cases}$$

and $\tau = \pi i/(2m)$. The DCT can be efficiently implemented through some fast algorithms, or by employing equation (9.53). In the latter case, special consideration must be given to the poles on the unit circle.

Alternatively, the transfer functions of the prefilter can be expressed as

$$T_i(z) = \frac{1}{m} \sum_{j=0}^{m-1} t_{ij} z^{-j} = \frac{1}{m} \prod_{r=0}^{m-2} \frac{(z - \tau_{ir})}{z} = \frac{1}{z^{m-1}} \frac{(z-1)[z^m - (-1)^i]}{[z^2 - (2\cos\frac{\pi i}{m})z + 1]} \tag{9.54}$$

where the t_{ij} are the coefficients of the transform matrix $\mathbf{T}$, and the τ_{ir} are the zeros of $T_i(z)$. The gain constants k_0 and $\cos\tau$ were dropped in equation (9.54) and will not be considered from now on, since they can be absorbed by the numerator coefficients $b_{0i}(k)$, $b_{1i}(k)$, and $b_{2i}(k)$ of $H_i(z)$.

The overall transfer function of the frequency-domain adaptive filter of Fig. 9.8 is given by

$$\begin{aligned} H(z) &= \sum_{i=0}^{m-1} T_i(z) H_i(z) \\ &= \frac{1}{m}\left(\frac{1}{z^{m-1}}\right)\left[\sum_{k=0}^{m-1}\left(\frac{b_{0i}z^2 + b_{1i}z + b_{2i}}{z^2 - a_{1i}z - a_{2i}}\right)\prod_{r=0}^{m-2}(z-\tau_{ir})\right] \\ &= \frac{1}{m}\frac{1}{z^{3m+1}}\left[\frac{\sum_{k=0}^{m-1}(b_{0i}z^2+b_{1i}z+b_{2i})\prod_{j=0,\neq i}^{m-1}(z^2-a_{1j}z-a_{2j})\prod_{r=0}^{m-2}(z-\tau_{ir})}{\prod_{l=0}^{m-1}(z^2-a_{1l}z-a_{2l})}\right] \end{aligned} \tag{9.55}$$

Now assume that the realization discussed is used to identify a system of order $2N_p$ described by

$$H_D(z) = Kz^{2N_p-P}\frac{\prod_{r=0}^{P-1}(z-\gamma_r)}{\prod_{i=0}^{N_p-1}(z^2-\alpha_{1i}z-\alpha_{2i})} \tag{9.56}$$

where K is a gain constant, p_{0i} and p_{1i} are the poles of section i, and γ_r are the zeros of $H_D(z)$ such that

$$\gamma_r \neq p_{0i}, p_{1i} \quad \text{for} \quad r = 0, \ldots, P-1 \quad \text{andfor} \quad i = 0, \ldots, N_p - 1$$

It can be shown that if the conditions outlined below are satisfied, the filter of Fig. 9.8 can identify exactly systems with $N_p \leq m$ and $P \leq 3m + 1$. The sufficient conditions are

i) The transformation matrix $\mathbf{T}$ of the prefilter is square and has linearly independent rows.

ii) $a_{1i} \neq a_{1j}$, and $a_{2i} \neq a_{2j}$ for $i \neq j$; a_{1i} and a_{2i} are not simultaneously zero for all i.

iii) The zeros of the prefilter do not coincide with the system's poles, i.e., $\tau_{ij} \neq p_{0l}$, $\tau_{ij} \neq p_{1l}$, for all i, j, and l.

Adaptation Algorithm

The adaptation algorithm entails the manipulation of a number of vectors, namely, the coefficient vector

$$\boldsymbol{\theta}(k) = \left[\boldsymbol{\theta}_0^T(k) \ldots \boldsymbol{\theta}_{m-1}^T(k)\right]^T$$

where

$$\boldsymbol{\theta}_i(k) = [-a_{1i}(k) \ -a_{2i}(k) \ b_{0i}(k) \ b_{1i}(k) \ b_{2i}(k)]^T$$

the internal data vector

$$\boldsymbol{\phi}(k) = \left[\boldsymbol{\phi}_0^T(k) \ldots \boldsymbol{\phi}_{m-1}^T(k)\right]^T$$

where

$$\boldsymbol{\phi}_i(k) = [y_i(k-1) \ y_i(k-2) \ x_i(k) \ x_i(k-1) \ x_i(k-2)]^T$$

the gradient vector

$$\boldsymbol{\Psi}(k) = [\boldsymbol{\psi}_0^T(k) \ldots \boldsymbol{\psi}_{m-1}^T(k)]^T$$

where

$$\boldsymbol{\psi}_i(k) = [y_i'(k-1) \ y_i'(k-2) \ x_i'(k) \ x_i'(k-1) \ x_i'(k-2)]^T$$

and the matrix $\hat{\mathbf{S}}(k)$ which is in estimate of the inverse Hessian $\hat{\mathbf{R}}^{-1}(k)$.

The elements of the gradient vector can be calculated by using the relations

$$x_i'(k) = x_i(k) - a_{1i}(k)x_i'(k-1) - a_{2i}(k)x_i'(k-2)$$

and

$$y_i'(k) = y_i(k) - a_{1i}(k)y_i'(k-1) - a_{2i}(k)y_i'(k-2)$$

An adaptation algorithm for updating the filter coefficients based on the Gauss-Newton algorithm is summarized Algorithm 9.2. The algorithm includes the updating of matrix $\hat{\mathbf{S}}(k)$, which is obtained through the matrix inversion lemma

The stability monitoring consists of verifying whether each set of coefficients $a_{1i}(k)$ and $a_{2i}(k)$ defines a point outside the stability triangle [1], i.e., by testing whether

$$1 - a_{1i}(k) + a_{2i}(k) < 0 \quad \text{or} \quad 1 + a_{1i}(k) + a_{2i}(k) < 0 \quad \text{or} \quad |a_{2i}(k)| \geq 1 \quad (9.57)$$

If instability is detected in a particular section, the poles must be projected back inside the unit circle. A possible strategy is to project each pole by keeping its

angle and inverting its modulus. In this case, a_{2i} and a_{1i} should be replaced by $1/a_{2i}(k)$ and $-a_{1i}(k)/a_{2i}(k)$, respectively.

If the outputs of the DCT prefilter $x_i(k)$ are sufficiently uncorrelated, the Hessian matrix is approximately block-diagonal consisting of 5×5 submatrices $\hat{\mathbf{R}}_i(k)$. In this case, instead of computing a $5m \times 5m$ inverse Hessian estimate $\hat{\mathbf{S}}(k)$, several 5×5 submatrices are computed and applied in the above algorithm as follows:

For $i = 0, 1, \ldots, m-1$

$$\boldsymbol{h}_i(k) = \hat{\mathbf{S}}_i(k)\boldsymbol{\psi}_i(k)$$

$$\hat{\mathbf{S}}_i(k+1) = \left[\hat{\mathbf{S}}_i(k) - \frac{\boldsymbol{h}_i(k)\boldsymbol{h}_i^T(k)}{(\frac{1}{\alpha} - 1) + \boldsymbol{h}_i^T(k)\boldsymbol{\psi}_i(k)}\right](\frac{1}{1-\alpha})$$

$$\boldsymbol{\theta}_i(k+1) = \boldsymbol{\theta}_i(k) + \mu\hat{\mathbf{S}}_i(k+1)\boldsymbol{\psi}_i(k)e(k)$$

□

The choice of the adaptive filter realization has implications on the computational complexity as well as on the convergence speed. Some studies exploring this aspect related to the frequency-domain realization can be found in [18]. The exploration of realization related properties of the IIR adaptive MSE surface led to a fast parallel realization where no transform preprocessing is required [19]. In this approach, the reduced-order manifolds are avoided by properly configuring the parallel sections which are implemented with general purpose second-order sections [20]. An analysis of the asymptotic convergence speed of some adaptive IIR filtering algorithms from the realization point of view can be found in [21].

Algorithm 9.2

Frequency-domain parallel algorithm, RLS Version

Initialization

$\hat{\mathbf{S}}(0) = \delta I(\delta > 0)$

$\boldsymbol{\theta}_i(k)$, $0 \leq i \leq m-1$

For each $x(k)$ and $d(k)$ given for $k \geq 0$, compute:

$X_{\mathrm{DCT}}(k) = \mathrm{DCT}[x(k) \ldots x(k-m+1)]$

Do for $i = 0, 1, \ldots, m-1$:

$x_i'(k) = x_i(k) - a_{1i}(k)x_i'(k-1) - a_{2i}(k)x_i'(k-2)$

$y_i(k) = \boldsymbol{\theta}_i^T(k)\boldsymbol{\phi}_i(k)$

$y_i'(k) = y_i(k) - a_{1i}(k)y_i'(k-1) - a_{2i}(k)y_i'(k-2)$

End

$e(k) = d(k) - \sum_{i=0}^{m-1} y_i(k)$

$\boldsymbol{h}(k) = \hat{\mathbf{S}}(k)\boldsymbol{\Psi}(k)$

$$\hat{\mathbf{S}}(k+1) = \left[\hat{\mathbf{S}}(k) - \frac{\boldsymbol{h}(k)\boldsymbol{h}^T(k)}{(\frac{1}{\alpha}-1)+\boldsymbol{h}^T(k)\boldsymbol{\Psi}(k)}\right](\tfrac{1}{1-\alpha})$$

$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + \mu\hat{\mathbf{S}}(k+1)\boldsymbol{\Psi}(k)e(k)$

Carry out stability test.
End

□

Example 9.1

An IIR adaptive filter of sufficient order is used to identify a system with the transfer function given below.

$$H(z) = \frac{0.8(z^2 - 1.804z + 1)^2}{(z^2 - 1.512z + 0.827)(z^2 - 1.567z + 0.736)}$$

The input signal is a uniformly distributed white noise with variance $\sigma_x^2 = 1$, and the measurement noise is a Gaussian white noise uncorrelated with the input with variance $\sigma_n^2 = 10^{-1.5}$. Use a gradient-based algorithm.
(a) Choose the appropriate values of μ.
(b) Run the algorithm using the direct form structure, the lattice structure, the parallel realization with preprocessing, and the cascade realization with direct form sections. Compare their convergence speed.
(c) Measure the MSE.
(d) Plot the obtained IIR filter frequency response in any iteration after convergence is achieved and compare with the unknown system. Consider for this item only the direct form realization.

Solution:

A convergence factor $\mu = 0.002$ was used in all examples, except for the lattice realization where for the updating of the feedforward coefficients a larger $\mu = 10$ was employed, otherwise the convergence would be too slow. Although the chosen value of μ is not optimal value in any sense, it led to the convergence of all algorithms. Fig. 9.9 depicts the magnitude response of the adaptive filter at a given iteration after convergence. For comparison the magnitude response of the unknown system is also plotted. As can be seen, the responses are close. Fig. 9.10 shows the learning curves of the algorithms obtained by averaging the results of 200 independent runs. As can be seen the faster algorithms led to higher MSE. The cascade realization presented faster convergence, followed by the parallel and lattice realizations. The measured MSE are given in Table 9.1.

There are very few results published in the literature addressing the finite-precision implementation of IIR adaptive filters. For this particular example, all algorithms were also implemented with fixed point arithmetic, with 12 and 16 bits. No sign of divergence was detected during the early 2000 iterations. However, the reader should not take this result as conclusive.

□

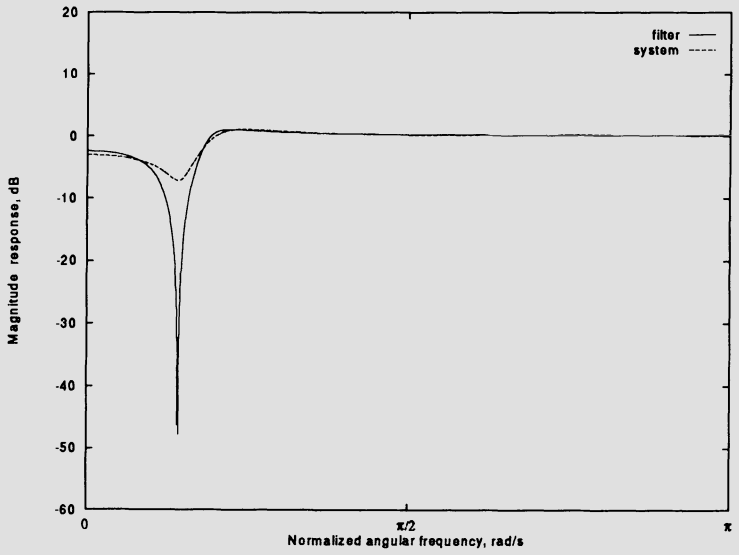
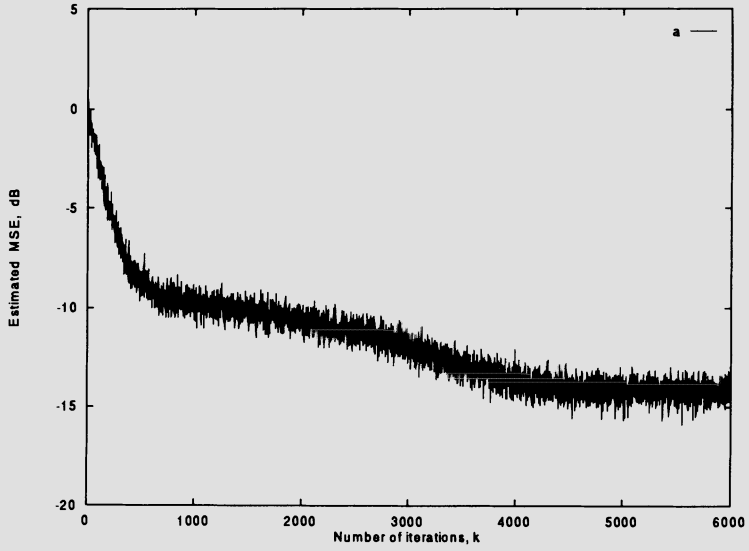


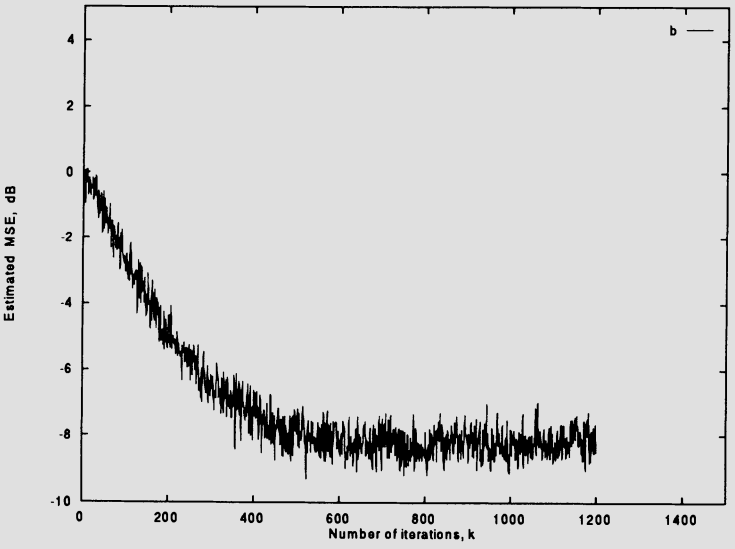
Figure 9.9 Magnitude response of the IIR adaptive filter with direct form at a given iteration after convergence.

Table 9.1 Evaluation of the IIR Algorithms

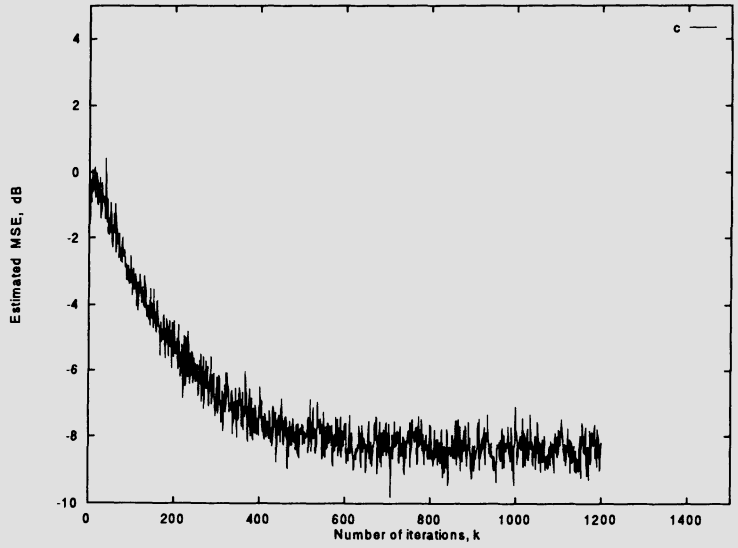
Realization	MSE
Direct Form	0.0391
Lattice	0.1514
Transf. Dom. Parallel	0.1478
Cascade	0.1592


a
Estimated MSE, dB
Number of iterations, k
5
0
-5
-10
-15
-20
0
1000
2000
3000
4000
5000
6000

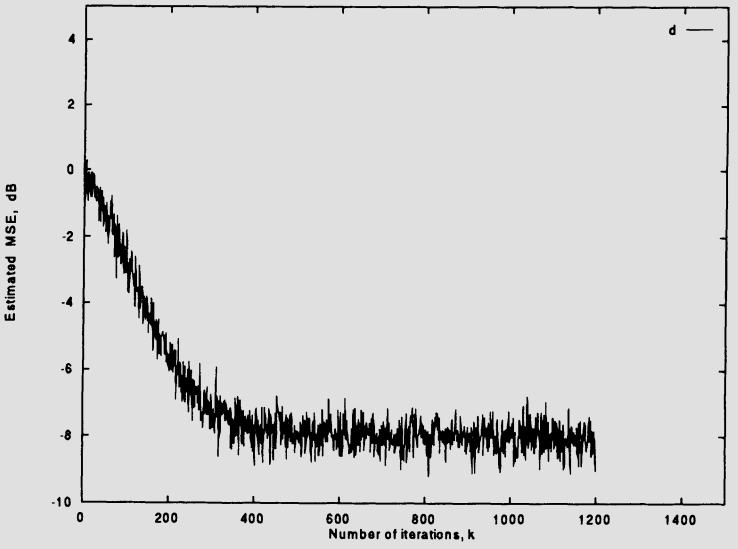
(a)


b
Estimated MSE, dB
Number of iterations, k
4
2
0
-2
-4
-6
-8
-10
0
200
400
600
800
1000
1200
1400

(b)



(c)



(d)

Figure 9.10 Learning curves for IIR adaptive filters with (a) Direct form, (b) Parallel form with preprocessing, (c) Lattice, and (d) Cascade realizations.

9.6 MEAN-SQUARE ERROR SURFACE

The error surface properties in the case of adaptive IIR filtering are key in understanding the difficulties in applying gradient-based algorithms to search for the optimal filter coefficient vector. In this section, the main emphasis is given to the system identification application where the unknown system is modeled by

$$d(k) = \frac{G(q^{-1})}{C(q^{-1})}x(k) + n(k) \tag{9.58}$$

where

$$\begin{aligned} G(q^{-1}) &= g_0 + g_1 q^{-1} + \cdots + g_{M_d} q^{-M_d}, \\ C(q^{-1}) &= 1 + c_1 q^{-1} + \cdots + c_{N_d} q^{-N_d}, \end{aligned}$$

and $n(k)$ is the measurement noise that is considered uncorrelated with the input signal $x(k)$.

The unknown transfer function is

$$\begin{aligned} H_o(z) &= z^{N_d - M_d} \frac{g_0 z^{M_d} + g_1 z^{M_d - 1} + \cdots + g_{M_d - 1} z + g_{M_d}}{z^{N_d} + c_1 z^{N_d - 1} + \cdots + c_{N_d - 1} z + c_{N_d}} \\ &= z^{N_d - M_d} \frac{N_o(z)}{D_o(z)} \end{aligned} \tag{9.59}$$

The desired feature of the identification problem is that the adaptive filter transfer function $H_k(z)$ approximates $H_o(z)$ as much as possible in each iteration. If the performance criterion is the mean-square error (MSE), the objective function is expressed in terms of the input signal and the desired signals as follows:

$$\begin{aligned} \xi &= E[e^2(k)] = E[(d(k) - y(k))^2] \\ &= E[d^2(k) - 2d(k)y(k) + y^2(k)] \\ &= E\left\{ \left[\left(\frac{G(q^{-1})}{C(q^{-1})} x(k) + n(k) \right) - \frac{B(k, q^{-1})}{A(k, q^{-1})} x(k) \right]^2 \right\} \end{aligned} \tag{9.60}$$

Since $n(k)$ is not correlated to $x(k)$ and $E[n(k)] = 0$, equation (9.60) can be rewritten as

$$\xi = E\left\{ \left[\left(\frac{G(q^{-1})}{C(q^{-1})} - \frac{B(k, q^{-1})}{A(k, q^{-1})} \right) x(k) \right]^2 \right\} + E[n^2(k)] \tag{9.61}$$

The interest here is to study the relation between the objective function ξ and the model filter coefficients, independently if these coefficients are adaptive or not. The polynomials operators $B(k, q^{-1})$ and $A(k, q^{-1})$ will be considered fixed, denoted respectively by $B(q^{-1})$ and $A(q^{-1})$.

The power spectra of the signals involved in the identification process are given by

$$\begin{aligned} R_{xx}(z) &= \mathcal{Z}[r_{xx}(l)] \\ R_{nn}(z) &= \mathcal{Z}[r_{nn}(l)] \\ R_{dd}(z) &= H_o(z)\ H_o(z^{-1})\ R_{xx}(z) + R_{nn}(z) \\ R_{yy}(z) &= H_k(z)\ H_k(z^{-1})\ R_{xx}(z) \\ R_{dy}(z) &= H_o(z)\ H_k(z^{-1})\ R_{xx}(z) \end{aligned} \tag{9.62}$$

By noting that for any processes $x_1(k)$ and $x_2(k)$

$$E[x_1(k)x_2(k)] = \frac{1}{2\pi j}\oint R_{x_1x_2}(z)\frac{dz}{z} \tag{9.63}$$

where the integration path is the counterclockwise unit circle. The objective function, as in equation (9.60), can be rewritten as

$$\begin{aligned} \xi &= \frac{1}{2\pi j}\oint \left[|H_o(z) - H_k(z)|^2 R_{xx}(z) + R_{nn}(z)\right]\frac{dz}{z} \\ &= \frac{1}{2\pi j}\left[\oint H_o(z)H_o(z^{-1})R_{xx}(z)\frac{dz}{z} - 2\oint H_o(z)H_k(z^{-1})R_{xx}(z)\frac{dz}{z}\right. \\ &\quad \left. + \oint H_k(z)H_k(z^{-1})R_{xx}(z)\frac{dz}{z} + \oint R_{nn}(z)\frac{dz}{z}\right] \end{aligned} \tag{9.64}$$

For the case the input and additional noise signals are white with variances respectively given by σ_x^2 and σ_n^2, the equation (9.64) can be simplified to

$$\xi = \frac{\sigma_x^2}{2\pi j}\oint \left[H_o(z)H_o(z^{-1}) - 2H_o(z)H_k(z^{-1}) + H_k(z)H_k(z^{-1})\right]\frac{dz}{z} + \sigma_n^2 \tag{9.65}$$

This expression provides the relation between the MSE surface represented by ξ and the coefficients of the adaptive filter. The following example illustrates the use of the equation above.

Example 9.2

An all-pole adaptive filter of second-order is used to identify a system with

transfer function

$$H_o(z) = \frac{1}{z^2 + 0.9z + 0.81}$$

The input signal and the measurement (additional) noise are white with $\sigma_x^2 = 1$ and $\sigma_n^2 = 0.1$, respectively. Compute the MSE as function of the adaptive filter multiplier coefficients.

Solution
The adaptive filter transfer function is given by

$$H_k(z) = \frac{b_2}{z^2 + a_1 z + a_2}$$

Equation (9.65) can be solved by employing the residue theorem [1] which results in

$$\begin{aligned}\xi = & \frac{b_2^2(1+a_2)}{(1-a_2)(1+a_2-a_1)(1+a_2+a_1)} \\ & - \frac{2b_2(1-0.81a_2)}{1-0.9a_1-0.81a_2-0.729a_1a_2+0.81a_1^2+0.6561a_2^2} \\ & + 3.86907339 + 0.1 \end{aligned} \tag{9.66}$$

If the adaptive filter coefficients are set to their optimal values, i.e., $b_2 = 1$, $a_1 = 0.9$ and $a_2 = 0.81$, indicating a perfect identification of the unknown system, the resulting MSE is

$$\begin{aligned}\xi &= 3.86907339 - 7.73814678 + 3.86907339 + 0.1 \\ &= 0.1\end{aligned}$$

Note that the minimum MSE is equal to the measurement noise variance.

□

Equations (9.64) and (9.65), and more specifically (9.66), indicate clearly that the MSE surface is a nonquadratic function of the multiplier coefficients of the adaptive filter. This is particularly true for the multiplier coefficients pertaining to the denominator of the adaptive filter. As a consequence, the MSE surface may have several local minima, some of those corresponding to the desired global minimum. The multiplicity of minimum points depends upon the order of the adaptive IIR filter as compared to the unknown system that shapes the desired signal and also upon the input signal properties when it is a colored noise.

Note that when the adaptive filter is FIR there is only a minimum point because the MSE surface is quadratic, independently of the unknown system and input signal characteristics. If the input or the desired signal are not stationary, the minimum point of the MSE surface moves in time but it is still unique.

The main problem brought about by the multimodality of the MSE surface is that gradient and Newton direction search algorithms will converge to a local minimum. Therefore, the adaptive filter may converge to a very bad point where the MSE assumes a large and unacceptable value. For example, in the system identification application, the generated transfer function may differ significantly from the unknown system transfer function.

Example 9.3
An unknown system with transfer function

$$H_o(z) = \frac{z - 0.85}{z + 0.99}$$

is supposed to be identified by a first-order adaptive filter described by

$$H_k(z) = \frac{bz}{z - a}$$

Plot the error surface.

Solution
The expression for the MSE is given by

$$\xi = 171.13064 - \frac{(2 - 1.7a)b}{1 + 0.99a} + \frac{b^2}{1 - a^2}$$

The MSE surface is depicted in Fig. 9.11, where the MSE is clipped at 1 for a better view.

□

Several results regarding the uniqueness of the minimum point in the MSE surface are available in the literature [22]-[27]. Here, some of these results are summarized without proof, in order to give the designer some tools to support the appropriate choice of the adaptive IIR filter order.

First consider the case of inverse filtering or equalization, where the adaptive filter is placed in cascade with an unknown system and the desired signal is a delayed version of the overall cascade input signal. This case had been originally

explored by Åström and Söderström [22], and they proved that if the adaptive filter is of sufficient order to find the inverse filter of the unknown system all the local minima will correspond to global minima if the input signal is a white noise. The sufficient order means that

$$N \geq M_d$$

and

$$M \geq N_d \tag{9.67}$$

where N and M are the numerator and denominator orders of adaptive filter as indicated in equation (9.5), N_d and M_d are the corresponding orders for the unknown system as indicated in (9.59).

When $N > M_d$ and $M > N_d$, there are infinitely many solutions given by

$$N(z) = L(z)D_o(z)$$

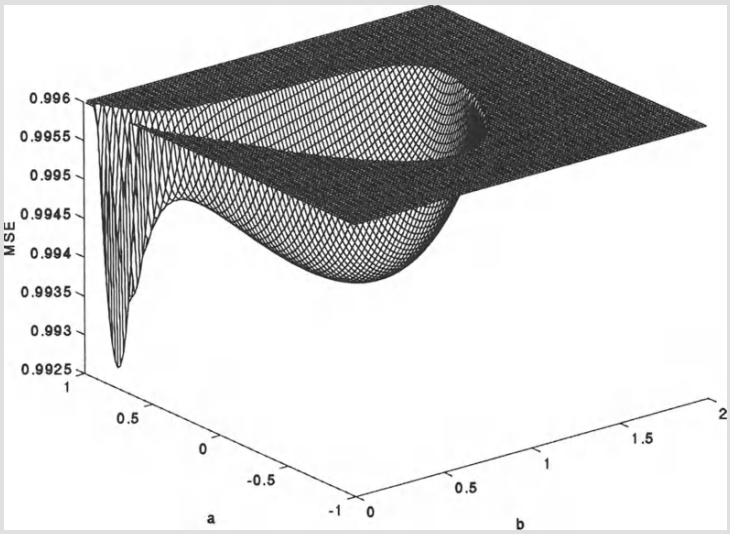
and

$$D(z) = L(z)N_o(z) \tag{9.68}$$

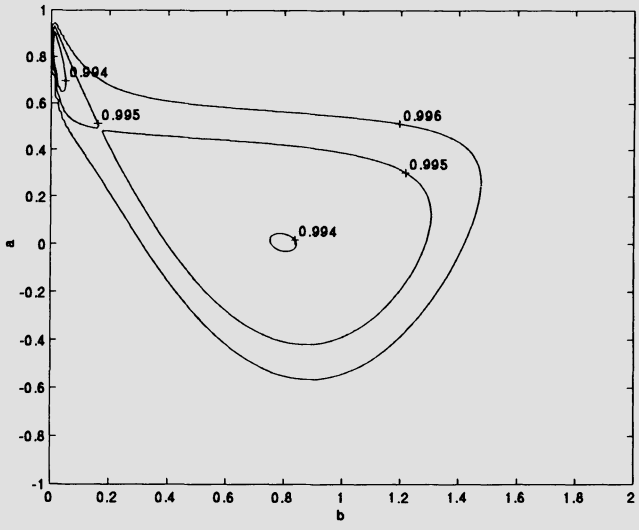
where $L(z) = z^{-N_l}(z^{N_l} + l_1 z^{N_l-1} + \cdots + l_{N_l})$, $N_l = \min(N - M_d, M - N_d)$, and l_i for $i = 1, 2, \ldots, N_l$, are arbitrary.

The input signal can be colored noise generated for example by applying an IIR filter to a white noise. In this case, the adaptive filter must have order sufficient to generate the inverse of the unknown system and the input signal must be persistently exciting of order $\max(N + M_d, M + N_d)$, see for example [22]-[23], in order to guarantee that all local minima correspond to global minima.

For insufficient-order equalization, several local minima that do not correspond to a global minimum may occur. In this case, the MSE may not attain its minimum value after the algorithm convergence.



(a)



(b)

Figure 9.11 (a) MSE error surface, (b) MSE contours.

The situation is not the same in system identification application, as thought in the early investigations [24]. For this application, the sufficient order means

$$N \geq N_d$$

and

$$M \geq M_d \tag{9.69}$$

since the desired feature is to reproduce the unknown system frequency response, and not its inverse as in the equalization case. For $N > N_d$ and $M > M_d$, the local minima corresponding to global minima must satisfy the following conditions

$$N(z) = L(z)N_o(z)$$

and

$$D(z) = L(z)D_o(z) \tag{9.70}$$

where $L(z) = z^{-N_l}(z^{N_l} + l_i z^{N_l - 1} + \cdots + l_{N_l})$, $N_l = \min(N - M_d, M - N_d)$, and l_i for $i = 1, 2, \ldots, N_l$ are arbitrary.

The strongest result derived so far regarding the error surface property in system identification was derived by Söderström and Stoica [25]. The result states: For white noise input, all the stationary points correspond to global minima if

$$M \geq N_d - 1$$

and

$$\min(N - N_d, M - M_d) \geq 0 \tag{9.71}$$

Suppose that the input signal is an ARMA process generated by filtering a white noise with an IIR filter of orders M_n by N_n, and that there are no common zeros between the unknown system denominator and the input coloring IIR filter. In this case, all stationary points correspond to global minima if

$$M - N_d + 1 \geq N_n$$

and

$$\min(N - N_d, M - M_d) \geq M_n \tag{9.72}$$

The conditions resumed by equations (9.71) and (9.72) are sufficient but not necessary to guarantee that all stationary solutions correspond to the minimum MSE.

For $N = N_d = 1$, $M \geq M_d \geq 0$ and the input signal persistently exciting of order M_d there is a unique solution given by [25]

$$D(z) = D_o(z)$$

and

$$N(z) = N_o(z) \tag{9.73}$$

Also, when the adaptive filter and unknown system are all-pole second-order sections the unique solution is given by equation (9.73) [26].

Another particular result of some interest presented in [27], states that if

$$N - N_d = M - M_d = 0$$

and

$$M \geq N_d - 2 \tag{9.74}$$

the MSE surface has a unique stationary point corresponding to a global minimum.

For the case of insufficient-order identification [28], i.e., $\min(N - N_d, M - M_d) < 0$, or of sufficient order not satisfying the condition related to equations (9.72)-(9.74), the MSE surface may have local minima not attaining the minimum MSE, i.e., that are not global minima.

To satisfy any of the conditions (9.72)-(9.74) a knowledge of the unknown system numerator and denominator orders is required. This information is not in general available or easy to obtain. This is one of the reasons adaptive IIR filters are not as popular as their FIR counterparts. However, there are situations where a local minima is acceptable or some information about the unknown system is available.

It should be noted that a vast literature is available for system identification [7],[29]-[30]. Here, the concentration was to resume some properties of the MSE surface, when the unknown system is modeled as an IIR filter with additive, white, and uncorrelated measurement noise. The assumptions regarding the measurement noise are quite reasonable for most applications of adaptive filtering.

9.7 INFLUENCE OF THE FILTER STRUCTURE ON MSE SURFACE

Some characteristics of the MSE surface differ when alternative structures are used in the realization of the adaptive filter. In each realization there is a different relation between the filter transfer function and the multiplier coefficients, originating the modification in the MSE surface [31].

The MSE surfaces related to two alternative realizations for the adaptive filter can be described as functions of the filter multiplier coefficients by $F_1(\boldsymbol{\theta}_1)$ and $F_2(\boldsymbol{\theta}_2)$, respectively. Note that no index was used to indicate the varying characteristics of the adaptive filter parameters, since this simplifies the notation while keeping the relevant MSE surface properties. It is assumed that the desired signal and the input signal are the same in the alternative experiments. Also, it is considered that for any set of parameters $\boldsymbol{\theta}_1$ leading to a stable filter, there is a continuous mapping given by $\boldsymbol{f}_3(\boldsymbol{\theta}_1) = \boldsymbol{\theta}_2$, where $\boldsymbol{\theta}_2$ also leads to a stable filter. Both $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are N' by 1 vectors.

The two alternative structures are equivalent if the objective functions are equal, i.e.,

$$F_1(\boldsymbol{\theta}_1) = F_2(\boldsymbol{\theta}_2) = F_2(\boldsymbol{f}_3(\boldsymbol{\theta}_1)) \tag{9.75}$$

First consider the case where $\boldsymbol{f}_3$ is differentiable, and then from the equation above it follows that

$$\frac{\partial F_1(\boldsymbol{\theta}_1)}{\partial \boldsymbol{\theta}_1} = \frac{\partial F_2(\boldsymbol{f}_3(\boldsymbol{\theta}_1))}{\partial \boldsymbol{\theta}_1} = \frac{\partial F_2^T(\boldsymbol{f}_3(\boldsymbol{\theta}_1))}{\partial \boldsymbol{f}_3(\boldsymbol{\theta}_1)} \frac{\partial \boldsymbol{f}_3(\boldsymbol{\theta}_1)}{\partial \boldsymbol{\theta}_1} \tag{9.76}$$

where the first partial derivative on the rightmost side of the above equation is a 1 by N' vector while the second partial derivative is a matrix with dimensions N' by N', where N' is the number of parameters in $\boldsymbol{\theta}_1$. Suppose that $\boldsymbol{\theta}_2'$ is a

stationary point of $F_2(\boldsymbol{\theta}_2)$, it then follows that

$$\frac{\partial F_2(\boldsymbol{\theta}_2)}{\partial \boldsymbol{\theta}_2}|_{\boldsymbol{\theta}_2=\boldsymbol{\theta}_2'} = \mathbf{0} = \frac{\partial F_1(\boldsymbol{\theta}_1)}{\partial \boldsymbol{\theta}_1}|_{\boldsymbol{\theta}_1=\boldsymbol{\theta}_1'} \tag{9.77}$$

where $\boldsymbol{\theta}_2' = \boldsymbol{f}_3(\boldsymbol{\theta}_1')$. Note that the type of the stationary points of $F_1(\boldsymbol{\theta}_1)$ and $F_2(\boldsymbol{\theta}_2)$ are the same, since their second derivatives have the same properties at these stationary points (see problem 1).

Now consider the case where

$$\frac{\partial F_2^T(\boldsymbol{f}_3(\boldsymbol{\theta}_1))}{\partial \boldsymbol{f}_3(\boldsymbol{\theta}_1)}|_{\boldsymbol{\theta}_1=\boldsymbol{\theta}_1''} = \mathbf{0} \tag{9.78}$$

but

$$\frac{\partial F_1(\boldsymbol{\theta}_1)}{\partial \boldsymbol{\theta}_1}|_{\boldsymbol{\theta}_1=\boldsymbol{\theta}_1''} \neq \mathbf{0} \tag{9.79}$$

that can happen only when $\boldsymbol{f}_3(\boldsymbol{\theta}_1)$ has a discontinuity at $\boldsymbol{\theta}_1 = \boldsymbol{\theta}_1''$. In this case, the chain rule of equation (9.76) does not apply. The new generated stationary points in $F_2^T(\boldsymbol{\theta}_2)$ can be shown to be saddle points (see problem 2).

Example 9.4
An unknown second-order system described by

$$H_o(z) = \frac{2z + c_1}{z^2 + c_1 z + c_2}$$

is to be identified by using two different structures for the adaptive filter, namely the direct form and the parallel form described respectively by

$$H_d(z) = \frac{2z + a_1}{z^2 + a_1 z + a_2}$$

and

$$H_p(z) = \frac{1}{z + p_1} + \frac{1}{z + p_2} = \frac{2z + p_1 + p_2}{z^2 + (p_1 + p_2)z + p_1 p_2}$$

verify the existence of new saddle points in the parallel realization.

Solution
The function relating the parameters of the two realizations can be given by

$$\boldsymbol{\theta}_2 = \begin{bmatrix} \frac{a_1 + \sqrt{a_1^2 - 4a_2}}{2} \\ \frac{a_1 - \sqrt{a_1^2 - 4a_2}}{2} \end{bmatrix} = \boldsymbol{f}_3(\boldsymbol{\theta}_1)$$

where function $\boldsymbol{f}_3(\boldsymbol{\theta}_1)$ is not differentiable when $a_2 = \frac{a_1^2}{4}$.

The inverse of the matrix $\frac{\partial \boldsymbol{f}_3(\boldsymbol{\theta}_1)}{\partial \boldsymbol{\theta}_1}$ is given by

$$\left[\frac{\partial \boldsymbol{f}_3(\boldsymbol{\theta}_1)}{\partial \boldsymbol{\theta}_1}\right]^{-1} = \begin{bmatrix} 1 & 1 \\ p_2 & p_1 \end{bmatrix}$$

and, if $p_1 = p_2$, the above matrix is singular, which makes possible that $\frac{\partial F_1(\boldsymbol{\theta}_1)}{\partial \boldsymbol{\theta}_1} \neq 0$ when $\frac{\partial F_2^T(\boldsymbol{\theta}_2)}{\partial \boldsymbol{\theta}_2} = 0$, as previously mentioned in equations (9.76) and (9.77).

Note that, as expected, $p_1 = p_2$ only when $a_2 = \frac{a_1^2}{4}$. On this parabola, the objective function $F_1(\boldsymbol{\theta}_1)$ has a minimum that corresponds to a saddle point of the function $F_2(\boldsymbol{\theta}_2)$. Also, this is the situation where the parallel realization is of reduced order, i.e., first order.

□

Basically, the manifold generated by the parallel realization is due to the fact that a given section can identify any pole of the unknown system, leaving the other poles to the remaining sections in parallel. This means that in a sufficient-order identification problem, if for the direct form realization there is a unique global minimum point, in the case of parallel realization with first-order sections there will be $N!$ global minima, where N is the number of poles in the unknown system. When using a parallel realization it is assumed that no multiple poles exist in the unknown system.

In the initialization of the algorithm, the adaptive filter parameters should not be in a reduced-order manifold, because by employing a gradient-based algorithm the parameters may be kept in the manifold and eventually reach a saddle point. The measurement noise, that is in general present in the adaptive filtering process, will help the parameters to skip the manifolds, but despite that the convergence will be slowed. A similar phenomenon occurs with the cascade realization of the adaptive filter.

9.8 ALTERNATIVE ERROR FORMULATIONS

The error signal (in some cases the regressor) can be chosen in alternative ways in order to avoid some of the drawbacks related to the output error formula-

tion, as for example the multiple local minima. Several formulations have been investigated in the literature [32]-[33], [35], [36]-[38], [41]-[42], [47]-[48], where each of them has its own advantages and disadvantages. The choice of the best error formulation depends on the application and on the information available about the adaptive filtering environment. In this section, we present two alternative error formulations, namely the equation error and Steiglitz-McBride methods, and discuss some of their known properties. Throughout the section other error formulations are briefly mentioned.

9.8.1 Equation Error Formulation

In the equation error (EE) formulation, the information vector instead of having past samples of the adaptive filter output, uses delayed samples of the desired signal as follows:

$$\boldsymbol{\phi}_e(k) = [-d(k-1) \; -d(k-2) \ldots -d(k-N) \; x(k) \; x(k-1) \ldots x(k-M)]^T \tag{9.80}$$

The equation error is defined by

$$e_e(k) = d(k) - \boldsymbol{\theta}^T(k)\boldsymbol{\phi}_e(k) \tag{9.81}$$

as illustrated in Fig. 9.12. The parameter vector $\boldsymbol{\theta}(k)$ is given by

$$\boldsymbol{\theta}(k) = [a_1(k) \; a_2(k) \; \ldots \; a_N(k) \; b_0(k) \; \ldots \; b_M(k)]^T \tag{9.82}$$

The equation error can be described in a polynomial form as follows:

$$e_e(k) = A(k, q^{-1})d(k) - B(k, q^{-1})x(k) \tag{9.83}$$

where, once again

$$\begin{aligned} B(k, q^{-1}) &= b_0(k) + b_1(k)q^{-1} + \cdots + b_M(k)q^{-M} \\ A(k, q^{-1}) &= 1 + a_1(k)q^{-1} + \cdots + a_N(k)q^{-N} \end{aligned}$$

The output signal related to the EE formulation is obtained through the following linear difference equation

$$\begin{aligned} y_e(k) &= -\sum_{j=0}^{M} b_j(k)x(k-j) + \sum_{j=1}^{N} a_j(k)d(k-j) \\ &= \boldsymbol{\theta}^T(k)\boldsymbol{\phi}_e(k) \end{aligned} \tag{9.84}$$

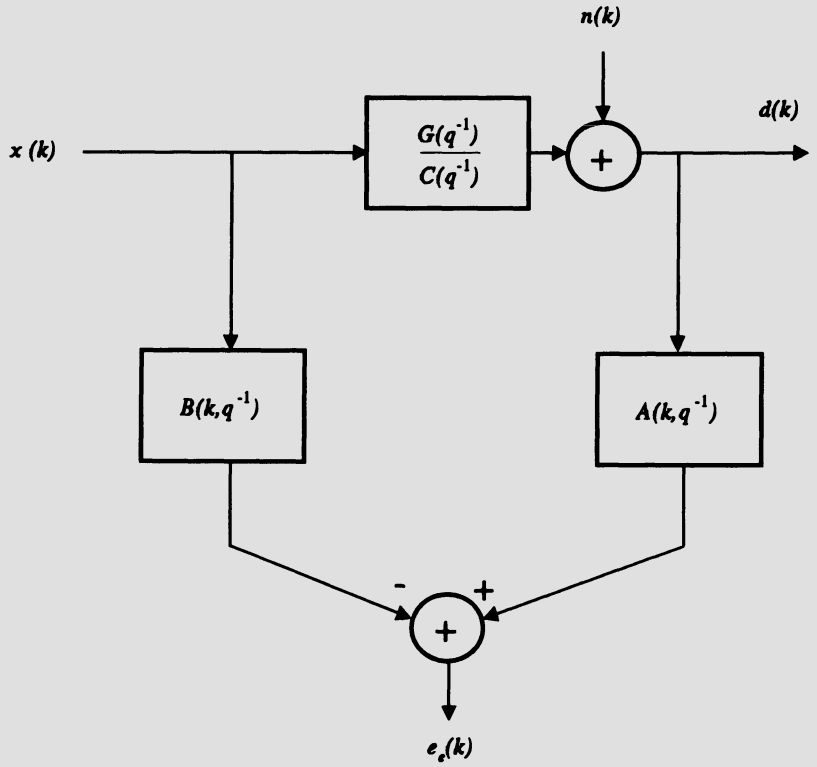


Figure 9.12 Equation error configuration.

As can be noted, the adaptive filter does not have feedback and $y_e(k)$ is a linear function of the parameters.

In the EE formulation, the adaptation algorithm determines how the coefficients of the adaptive IIR filter should change in order to minimize an objective function which involves $e_e(k)$ defined as

$$\xi_e = F(e_e(k)) \tag{9.85}$$

Usually, the objective function to be minimized is the mean-squared value of the EE (MSEE), i.e.,

$$\xi_e(k) = E[e_e^2(k)] \tag{9.86}$$

Since the input and desired signals are not functions of the adaptive filter parameters, it can be expected that the only approximation in the gradient computa-

tion is due to the estimate of the expected value required in practical implementations. The key point is to note that since the MSEE is a quadratic function of the parameters, only a global minimum exists provided the signals involved are persistently exciting. When the estimate of the MSEE is the instantaneous squared equation error, the gradient vector is given by minus the information vector. In this case, the resulting algorithm is called LMSEE algorithm whose coefficient updating equation is given by

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + 2\mu\boldsymbol{\phi}_e(k)e_e(k) \tag{9.87}$$

A number of approaches with different points of view are available to analyze the convergence properties of this method. A particularly interesting result is that if the convergence factor is chosen in the range

$$0 < \mu < \frac{1}{\lambda_{max}} \tag{9.88}$$

the convergence in the mean of the LMSEE algorithm can be guaranteed [33], where λ_{max} is the maximum eigenvalue of $E[\boldsymbol{\phi}_e(k)\boldsymbol{\phi}_e^T(k)]$. This result can be easily proved by exploring the similarity between the LMSEE algorithm and the standard FIR LMS algorithm. Some stability results of the LMSEE algorithm can be found in [34].

An alternative objective function for adaptive IIR filtering based on equation error is the least-squares function

$$\xi_e(k) = \sum_{i=0}^{k} \lambda^{k-i} e_e^2(i) = \sum_{i=0}^{k} \lambda^{k-i}[d(i) - \boldsymbol{\theta}^T(k)\boldsymbol{\phi}_e(i)]^2 \tag{9.89}$$

The forgetting factor λ, as usual is chosen in the range $0 << \lambda < 1$, allowing the distant past information to be increasingly negligible. In this case, the corresponding RLS algorithm consists of the following basic steps

$$e(k) = d(k) - \boldsymbol{\theta}^T(k)\boldsymbol{\phi}_e(k) \tag{9.90}$$

$$\mathbf{S}_{De}(k+1) = \frac{1}{\lambda}\left[\mathbf{S}_{De}(k) - \frac{\mathbf{S}_{De}(k)\boldsymbol{\phi}_e(k)\boldsymbol{\phi}_e^T(k)\mathbf{S}_{De}(k)}{\lambda + \boldsymbol{\phi}_e^T(k)\mathbf{S}_{De}(k)\boldsymbol{\phi}_e(k)}\right] \tag{9.91}$$

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + e_e(k)\mathbf{S}_{De}(k+1)\boldsymbol{\phi}_e(k) \tag{9.92}$$

In a given iteration k, the adaptive IIR filter transfer function related to the EE formulation can be expressed as follows:

$$H_k(z) = z^{N-M}\frac{b_0(k)z^M + b_1(k)z^{M-1} + \cdots + b_{M-1}(k)z + b_M(k)}{z^N + a_1(k)z^{N-1} + \cdots + a_{N-1}(k)z + a_N(k)} \tag{9.93}$$

In Fig. 9.13 an alternative structure for the EE approach where the IIR adaptive filter appears explicitly is depicted. Note that the structure shows clearly that the polynomial $A(k, q^{-1})$ is meant to model the denominator polynomial of the unknown system, in system identification applications. During the adaptation process, it is necessary to monitor the stability of the poles, as described for the OE method. The full description of the RLS equation error algorithm is given in Algorithm 9.3.

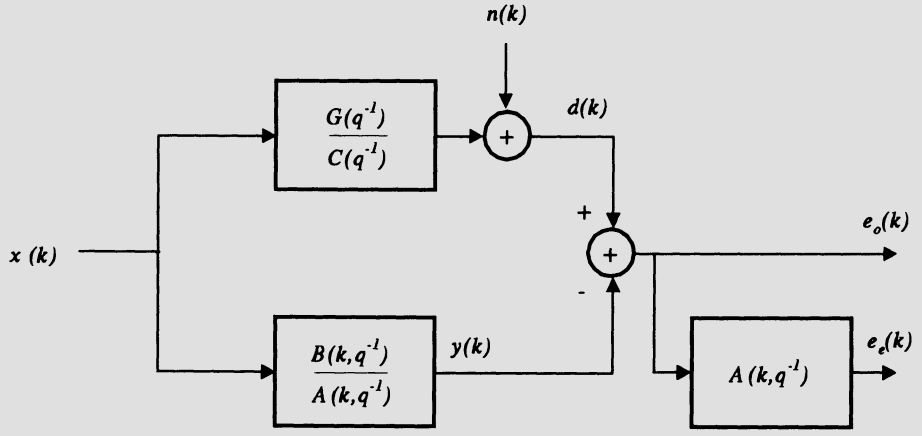


Figure 9.13 Basic configuration for system identification using equation error.

The basic problem related to this method is the parameter bias induced by the measurement noise [33]-[34], even for sufficient-order case. The bias is caused by the fact that the additional noise $n(k)$ is filtered by the FIR filter represented by the polynomial $A(k, q^{-1})$. Since the coefficients of this polynomial are updated with the objective of minimizing the EE signal, they also attempt to minimize the contribution of $n(k)$ to the EE power. The bias is induced by the fact that the additional noise does not belong to the unknown system model. An increase in the power of $n(k)$ leads to higher bias in the parameter estimate.

The Instrumental Variable methods [35] were proposed to solve the bias problem. In these methods the stability cannot be guaranteed under the same general conditions as for the LMSEE method.

Another approach was proposed in [36], and extended in [37] and [38], where a family of asymptotically stable algorithms was introduced. The resulting algorithms are based on a modification of the basic LMSEE updating equations, that within sufficiently general conditions lead to consistent parameter

Algorithm 9.3

EE Algorithm, RLS Version

Initialization

$a_i(k)=b_i(k)=e(k)=0$

$y(k)=x(k)=0\,,\ k<0$

$\mathbf{S}_{De}(0)=\delta^{-1}\mathbf{I}$

For each $x(k)$, $d(k)$, $k \geq 0$, do

$e_e(k)=d(k)-\boldsymbol{\phi}_e^T(k)\boldsymbol{\theta}(k)$

$$\mathbf{S}_{De}(k+1)=\frac{1}{\lambda}\left[\mathbf{S}_{De}(k)-\frac{\mathbf{S}_{De}(k)\boldsymbol{\phi}_e(k)\boldsymbol{\phi}_e^T(k)\mathbf{S}_{De}(k)}{\lambda+\boldsymbol{\phi}_e^T(k)\mathbf{S}_{De}(k)\boldsymbol{\phi}_e(k)}\right]$$

$\boldsymbol{\theta}(k+1)=\boldsymbol{\theta}(k)+\mathbf{S}_{De}(k+1)\boldsymbol{\phi}_e(k)e_e(k)$

Stability test

□

estimates. These algorithms employ a type of output error feedback to the information vector. There are also algorithms that combine different algorithms to define the objective function [39]-[40].

9.8.2 The Steiglitz-McBride Method

The Steiglitz-McBride (SM) error formulation [41], by employing some extra all-pole filtering, leads to algorithms whose behavior resembles the EE approach in the initial iterations and the OE approach after convergence. The main motivation of the SM method is the global convergence behavior for some cases of insufficient-order system identification. Such interest sparkled investigations which resulted in a number of on-line algorithms based on the SM method that are suitable for adaptive IIR filtering [42]. The main problem associated with the SM method is the inconsistent behavior when the measurement noise is colored [43]. Since the on-line method converges asymptotically to the off-line solution, the bias error also affects the on-line algorithms proposed in [42].

In order to introduce the SM method, consider the identification of a system whose model is described by

$$d(k) = \frac{G(q^{-1})}{C(q^{-1})}x(k) + n(k) = y_d(k) + n(k) \tag{9.94}$$

where $d(k)$ is the reference signal, $x(k)$ is the input signal, $n(k)$ is the measurement noise, and $y_d(k)$ is the output signal of the plant, with $C(q^{-1}) = 1 - \sum_{i=1}^{N_d} c_i q^{-i}$ and $G(q^{-1}) = \sum_{i=0}^{M_d} g_i q^{-i}$ coprime. The polynomial $C(q^{-1})$ has zeros inside the unit circle, and the input signal $x(k)$ and the measurement noise $n(k)$ are assumed independent. The estimation of the parameters associated with the polynomials $C(q^{-1})$ and $G(q^{-1})$ through the SM method is based on the minimization of the following criterion [41]

$$\xi_s(\boldsymbol{\theta}(k+1)) = E\left\{\left[A(k+1,q^{-1})\frac{d(k)}{A(k,q^{-1})} - B(k+1,q^{-1})\frac{x(k)}{A(k,q^{-1})}\right]^2\right\} \tag{9.95}$$

where $A(k,q^{-1}) = 1 + \sum_{i=1}^{N} a_i(k)q^{-i}$ and $B(k,q^{-1}) = \sum_{i=0}^{M} b_i(k)q^{-i}$ are the denominator and numerator estimator polynomials, respectively, and

$$\boldsymbol{\theta}(k) = [a_1(k)\ a_2(k)\ ...a_N(k)\ b_0(k)\ ...\ b_M(k)]^T \tag{9.96}$$

is the adaptive filter parameter vector.

The estimate $\boldsymbol{\theta}(k+1)$ is obtained by minimizing (9.95) assuming $\boldsymbol{\theta}(k)$ known. The solution of this MSE minimization problem at iteration $(k+1)$ is

$$\boldsymbol{\theta}(k+1) = \left[E\left\{\boldsymbol{\phi}_s(k)\boldsymbol{\phi}_s^T(k)\right\}\right]^{-1} E\left[\boldsymbol{\phi}_s(k)\frac{d(k)}{A(k,q^{-1})}\right] \tag{9.97}$$

where

$$\boldsymbol{\phi}_s(k) = \left[\frac{-d(k-1)}{A(k,q^{-1})} \cdots \frac{-d(k-N)}{A(k,q^{-1})}\ \frac{x(k)}{A(k,q^{-1})} \cdots \frac{x(k-M)}{A(k,q^{-1})}\right]^T \tag{9.98}$$

is the regressor related to the SM method.

If the input signal is persistently exciting of sufficient order and the adaptive filter has strictly sufficient order, some properties of the estimate resulting of (9.97) are known [43]: a) The estimate that minimizes (9.95) is unique; b) If the measurement noise is not white, the estimate resulting of (9.97) is biased.

In real-time signal processing applications, it is important to consider an online version of the SM method. In this case, some approximations are necessary.

First note that the error criterion whose variance is to be minimized in equation (9.97) is

$$e_s(k) = \frac{d(k)}{A(k, q^{-1})} - \boldsymbol{\theta}^T(k+1)\boldsymbol{\phi}_s(k) \tag{9.99}$$

The SM error is computed as illustrated in Fig. 9.14. Assuming a sufficiently slow parameter variation, we can consider that $\boldsymbol{\theta}(k+1) \approx \boldsymbol{\theta}(k)$. Therefore, equation (9.99) can be rewritten as follows:

$$e_s(k) \approx \frac{d(k)}{A(k, q^{-1})} - \boldsymbol{\theta}^T(k)\boldsymbol{\phi}_s(k) \tag{9.100}$$

The exact implementation of the regressor $\boldsymbol{\phi}_s(k)$ requires an independent filtering of each component by an all-pole filter with denominator polynomial $A(k, q^{-1})$. A useful approximation that reduces considerably the computational complexity is possible by assuming slow parameter variation [42] in such way that

$$\boldsymbol{\theta}(k-1) \approx \boldsymbol{\theta}(k-2) \ldots \approx \boldsymbol{\theta}(k-N) \tag{9.101}$$

With these simplifications only one all-pole filtering is required. Note that hypothesis similar to (9.101) was utilized in the output error method in order to simplify the implementation. However, in the case of the OE method, the measurement noise does not affect the regressor, since the regressor vector is composed of delayed samples of the adaptive filter output. For the SM method, except for white measurement noise, the simplification in (9.101) is not easily justified.

The updating equation of the on-line SM algorithm for system identification employing a stochastic gradient search is given by

$$\begin{aligned} \boldsymbol{\theta}(k+1) &= \boldsymbol{\theta}(k) + 2\mu\boldsymbol{\phi}_s(k)\left(\frac{d(k)}{A(k, q^{-1})} - \boldsymbol{\phi}_s^T(k)\boldsymbol{\theta}(k)\right) \\ &= \boldsymbol{\theta}(k) + 2\mu\boldsymbol{\phi}_s(k)e_s(k) \end{aligned} \tag{9.102}$$

The description of a gradient SM algorithm is given in the Algorithm 9.4.

The SM method can be implemented using different realizations such as cascade [44], lattice [45], and the series-parallel realization [46]. These realizations allow easy stability monitoring, and their choice affects the convergence speed [46].

It should be mentioned that a family of algorithms based on the SM method that solves the problem of inconsistency of the parameter estimates was proposed in [47]-[48]. These algorithms are very attractive for adaptive IIR filtering due

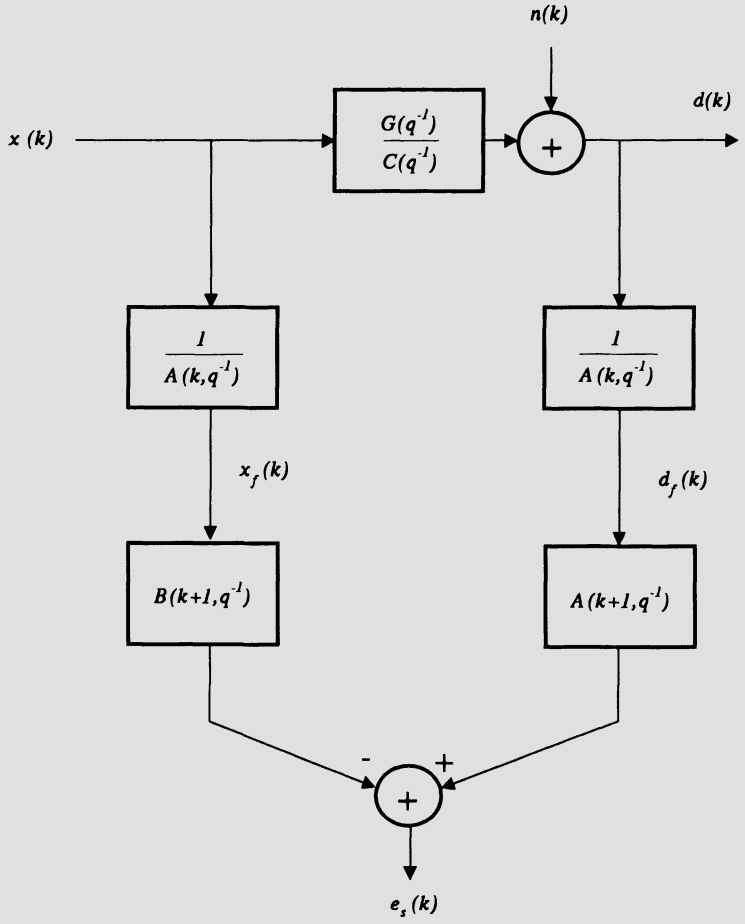


Figure 9.14 Steiglitz-McBride configuration.

to their behavior in terms of consistency (i.e., definition of stationary points) and convergence properties. In [51], simulation results as well as an interesting implementation for the consistent SM method was presented.

The interested reader can also find some interesting results about the convergence behavior of the SM-based algorithms in [49]-[50] and in the references therein. Also, applications of the SM algorithm to equalization can be found in [52].

Algorithm 9.4

SM Based Algorithm, Gradient Version

Initialization

$a_i(k)=b_i(k)=0$

$d_f(k)=x_f(k)=0\,,\ k<0$

For each $x(k)$, $d(k)$, $k \geq 0$ do

$x_f(k)=x(k)-\sum_{i=1}^{N} a_i(k)\, x_f(k-i)$

$d_f(k)=d(k)-\sum_{i=1}^{N} a_i(k)\, d_f(k-i)$

$e_s(k)=d_f(k)-\boldsymbol{\phi}_s^T(k)\boldsymbol{\theta}(k)$

$\boldsymbol{\theta}(k+1)=\boldsymbol{\theta}(k)+2\mu\boldsymbol{\phi}_s(k)e_s(k)$

Stability test

□

9.9 CONCLUSION

It is recognized that the adaptive IIR filter can be potentially used in a number of applications due to their superior system modeling owing to poles. These advantages come with the drawbacks such as possible local minima in the performance surface and the possible instability during the adaptation process. Also, the nonlinear relation between the adaptive filter parameters and the internal signals in some formulations turns the gradient computation and convergence analysis much more complicated as compared to the FIR case. In this chapter, the theory of adaptive IIR filters was presented exposing several solutions to the above mentioned drawbacks, such that the designer can decide what is the best configuration for a given application.

In this chapter, an example of application of adaptive IIR filters in system identification was presented. In this example, some of the realizations presented here were tested and compared.

References

1. A. Antoniou, *Digital Filters: Analysis, Design, and Applications*, McGraw-Hill, New York, NY, 2nd edition, 1992.

2. C. R. Johnson, Jr., "Adaptive IIR filtering: Current results and open issues," *IEEE Trans. on Information Theory*, vol. IT-30, pp. 237-250, Mar. 1984.

3. J. J. Shynk, "Adaptive IIR filtering," *IEEE ASSP Magazine*, vol. 6, pp. 4-21, April 1989.

4. S. A. White, "An adaptive recursive digital filter," *Proc. 9th Asilomar Conf. on Circuits, Systems, and Computers*, Pacific Grove, CA, pp. 21-25, Nov. 1975.

5. S. Hovarth, Jr., "A new adaptive recursive LMS filter," *Proc. Digital Signal Processing Conference*, Florence, Italy, pp. 21-26, 1980.

6. T. C. Hsia, "A simplified adaptive recursive filter design," *Proceedings of the IEEE*, vol. 69, pp. 1153-1155, Sept. 1981.

7. L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, MA, 1983.

8. R. A. David, "A modified cascade structure for IIR adaptive algorithms," *Proc. 15th Asilomar Conf. on Circuits, Systems, and Computers*, Pacific Grove, CA, pp. 175-179, Nov. 1981.

9. J. J. Shynk, "Adaptive IIR filtering using parallel-form realization," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. 37, pp. 519-533, April 1989.

10. P. S. R. Diniz, J. E. Cousseau, and A. Antoniou, "Improved parallel realization of IIR adaptive filters," *Proceedings of the IEE Part G: Circuits, Devices, and Systems*, vol. 140, pp. 322-328, Oct. 1993.

11. D. Parikh, N. Ahmed, and S. D. Stearns, "An adaptive lattice algorithm for recursive filters," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-28, pp. 110-112, Feb. 1980.

12. I. L. Ayala, "On a new adaptive lattice algorithm for recursive filters," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-30, pp. 316-319, April 1982.

13. J. J. Shynk, "On lattice-form algorithms for adaptive IIR filtering," *Proc. Int. Conf. on Acoust., Speech, Signal Processing*, New York, NY, pp. 1554-1557, April 1988.

14. J. A. Rodríguez-Fonollosa and E. Masgrau, "Simplified gradient calculation in adaptive IIR lattice filters," *IEEE Trans. on Signal Processing*, vol. 39, pp. 1702-1705, July 1991.

15. A. H. Gray, Jr., and J. D. Markel, "Digital lattice and ladder filter synthesis," *IEEE Trans. on Audio Electroacoust.*, vol. AU-21, pp. 492-500, Dec. 1973.

16. F. Itakura and S. Saito, "Digital filtering techniques for speech analysis and synthesis," *Proc. 7th Int. Congr. Acoustics*, Paper 25C-1, Budapest, Hungary, pp. 261-264, 1971.

17. M. Tummala, "New adaptive normalised lattice algorithm for recursive filters," *Electronics Letters*, vol. 24, pp. 659-661, May 1988.

18. H. Fan and Y. Yang, "Analysis of a frequency-domain adaptive IIR filter," *IEEE Trans. on Acoust. Speech, and Signal Processing*, vol. 38, pp. 864-870, May 1990.

19. P. S. R. Diniz, J. E. Cousseau, and A. Antoniou, "Fast parallel realization for IIR adaptive filters," *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 41, pp. 561-567, Aug. 1994.

20. P. S. R. Diniz and A. Antoniou, "Digital-filter structures based on the concept of the voltage-conversion generalized immittance converter," *Canadian J. of Electrical and Computer Engineering*, vol. 13, pp. 90-98, April 1988.

21. H. Fan, "A structural view of asymptotic convergence speed of adaptive IIR filtering algorithms: Part I-Infinite precision implementation," *IEEE Trans. on Signal Processing*, vol. 41, pp. 1493-1517, April 1993.

22. K. J. Åström and T. Söderström, "Uniqueness of the maximum likelihood estimates of the parameters of an ARMA model," *IEEE Trans. on Automatic Control*, vol. AC-19, pp. 769-773, Dec. 1974.

23. T. Söderström, "On the uniqueness of maximum likelihood identification," *Automatica*, vol. 11, pp. 193-197, Mar. 1975.

24. S. D. Stearns, "Error surfaces of recursive adaptive filters," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-29, pp. 763-766, June 1981.

25. T. Söderström and P. Stoica, "Some properties of the output error model," *Automatica*, vol. 18, pp. 93-99, Jan. 1982.

26. H. Fan and M. Nayeri, "On error surfaces of sufficient order adaptive IIR filters: Proofs and counterexamples to a unimodality conjecture," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. 37, pp. 1436-1442, Sept. 1989.

27. M. Nayeri, "Uniqueness of MSOE estimates in IIR adaptive filtering; a search for necessary conditions," *Proc. Int. Conf. on Acoust., Speech, Signal Processing*, Glasgow, Scotland, pp. 1047-1050, May 1989.

28. M. Nayeri, H. Fan, and W. K. Jenkins, "Some characteristics of error surfaces for insufficient order adaptive IIR filters," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. 38, pp. 1222-1227, July 1990.

29. L. Ljung, *System Identification: Theory for the User*, Prentice Hall Inc., Englewood Cliffs, NJ, 1987.

30. T. Söderström and P. Stoica, *System Identification*, Prentice Hall International, Hempstead, Hertfordshire, 1989.

31. M. Nayeri and W. K. Jenkins, "Alternate realizations to adaptive IIR filters and properties of their performance surfaces," *IEEE Trans. on Circuits and Systems*, vol. 36, pp. 485-496, April 1989.

32. S. L. Netto, P. S. R. Diniz, and P. Agathoklis, "Adaptive IIR filtering algorithms for system identification: A general framework," *IEEE Trans. on Education*, vol. 26, pp. 54-66, Feb. 1995.

33. J. M. Mendel, *Discrete Techniques of Parameter Estimation: The Equation Error Formulation*, Marcel Dekker, New York, NY, 1973.

34. T. Söderström and P. Stoica, "On the stability of dynamic models obtained by least squares identification," *IEEE Trans. on Automatic Control*, vol. AC-26, pp. 575-577, April 1981.

35. T. Söderström and P. Stoica, *Instrumental Variable Methods for System Identification*, Springer Verlag, New York, NY, 1983.

36. J. -N. Lin and R. Unbehauen, "Bias-remedy least mean square equation error algorithm for IIR parameter recursive estimation," *IEEE Trans. on Signal Processing*, vol. 40, pp. 62-69, Jan. 1992.

37. P. S. R. Diniz and J. E. Cousseau, "A family of equation-error based IIR adaptive algorithms," *IEEE Proc. Midwest Symposium of Circuits and Systems*, Lafayette, LA, pp. 1083-1086, 1994.

38. J. E. Cousseau and P. S. R. Diniz, "A general consistent equation-error algorithm for adaptive IIR filtering," *Signal Processing*, vol. 56, 1997.

39. J. B. Kenney and C. E. Rohrs, "The composite regressor algorithm for IIR adaptive systems," *IEEE Trans. on Signal Processing*, vol. 41, pp. 617-628, Feb. 1993.

40. S. L. Netto and P. S. R. Diniz, "Composite algorithms for adaptive IIR filtering," *IEE Electronics Letters*, vol. 28, pp. 886-888, April 1992.

41. K. Steiglitz and L. E. McBride, "A technique for the identification of linear systems," *IEEE Trans. on Automatic Control*, vol. AC-10, pp. 461-464, Oct. 1965.

42. H. Fan and W. K. Jenkins, "A new adaptive IIR filter," *IEEE Trans. on Circuits and Systems*, vol. CAS-33, pp. 939-947, Oct. 1986.

43. P. Stoica and T. Söderström, "The Steiglitz-McBride identification algorithm revisited–convergence analysis and accuracy aspects," *IEEE Trans. on Automatic Control*, vol. AC-26, pp. 712-717, June 1981.

44. B. E. Usevitch and W. K. Jenkins, "A cascade implementation of a new IIR adaptive digital filter with global convergence and improved convergence rates," *IEEE Proc. International Symposium of Circuits and Systems*, Portland, OR, pp. 2140-2142, 1989.

45. P. A. Regalia, "Stable and efficient lattice algorithms for adaptive IIR filtering," *IEEE Trans. on Signal Processing*, vol. 40, pp. 375-388, Feb. 1992.

46. J. E. Cousseau and P. S. R. Diniz, "A new realization of IIR echo cancellers using the Steiglitz-McBride method," *Proc. IEEE International Telecommunication Symposium*, Rio de Janeiro, Brazil, pp. 11-14, 1994.

47. J. E. Cousseau and P. S. R. Diniz, "A consistent Steiglitz-McBride algorithm," *Proc. IEEE International Symposium of Circuits and Systems*, Chicago, IL, pp. 52-55, 1993.

48. J. E. Cousseau and P. S. R. Diniz, "New adaptive IIR filtering algorithms based on Steiglitz-McBride method," *IEEE Trans. on Signal Processing*, accepted for publication 1997.

49. H. Fan and M. Doroslovački, "On 'global convergence' of Steiglitz-McBride adaptive algorithm," *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 40, pp. 73-87, Feb. 1993.

50. M. H. Cheng and V. L. Stonick, "Convergence, convergence point and convergence rate for Steiglitz-McBride method: A unified approach," *Proc. IEEE Intern. Conf. on Acoustics Speech and Signal Processing*, Adelaide, Australia, 1994.

51. V. L. Stonick and M. H. Cheng, "Adaptive IIR filtering: composite regressor method," *Proc. IEEE Intern. Conf. on Acoustics Speech and Signal Processing*, Adelaide, Australia, 1994.

52. P. M. Crespo and M. L. Honig, "Pole-zero decision feedback equalization with a rapidly converging adaptive IIR algorithm," *IEEE J. on Selected Areas in Communications*, vol. 9, pp. 817-828, Aug. 1991.

53. P. A. Regalia, *Adaptive IIR Filtering for Signal Processing Control*, Marcel Dekker, New York, NY, 1995.

Problems

1. Show that the stationary points related to two equivalent adaptive realizations of the type in equation (9.77) have the same nature, i.e., are minimum, maximum or saddle point.

2. Show that the new stationary points generated by the discontinuity in $\boldsymbol{f}_3(\boldsymbol{\theta}_1)$ as discussed after equation (9.79) are saddle points.

3. Describe how the manifolds are formed in the MSE surface when a cascade realization is used for the adaptive filter implementation. Give a generic example.

4. Derive a general expression for the transfer function of the two-multiplier lattice structure.

5. Derive an adaptive filtering algorithm which employs the canonic direct form structure shown in Fig. 9.15. Consider that the adaptive filter parameters are slowly varying in order to derive an efficient implementation for the gradient vector.

6. A second-order all-pole adaptive filter is used to find the inverse model of the signal $x(k) = 1.7n(k-1) + 0.81n(k-2) + n(k)$, where $n(k)$ is a white noise with variance 0.1. Using the gradient algorithm, calculate the error and the filter coefficients for the first 10 iterations. Start with $a_1(0) = 0, a_2(0) = 0$.

7. Repeat the problem 6 using the Gauss-Newton algorithm.

8. Use an IIR adaptive filter of sufficient order to identify a system with the transfer function given below. The input signal is a uniformly distributed white noise with variance $\sigma_x^2 = 1$, and the measurement noise is a Gaussian white noise uncorrelated with the input with variance $\sigma_n^2 = 10^{-2}$. Use a Gauss-Newton based algorithm and the direct form structure.

$$H(z) = \frac{0.000058(z^2 - 2z + 1)^3}{(z^2 + 1.645z + 0.701)(z^2 + 1.575z + 0.781)(z^2 + 1.547z + 0.917)}$$

(a) Run the algorithm for three values of μ. Comment on the convergence behavior in each case.
(b) Measure the MSE in each example.
(c) Plot the obtained IIR filter frequency response in any iteration after convergence is achieved and compare with the unknown system.

9. Repeat the previous problem using a second-order adaptive filter and interpret the results.

10. Replace the direct form structure in problem 8 by the lattice structure.

11. Replace the direct form structure in problem 8 by the parallel realization with preprocessing.

12. Replace the direct form structure in problem 8 by the cascade realization.

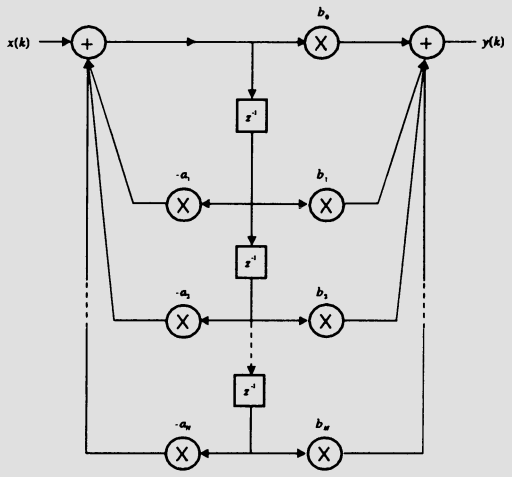


Figure 9.15 Direct form of Problem 5.

13. Repeat problem 8 in case the input signal is a uniformly distributed white noise with variance $\sigma_x^2 = 0.1$, filtered by all-pole filter given by

$$H(z) = \frac{z}{z - 0.95}$$

14. In problem 8 consider that the additional noise has the following variances (a) $\sigma_n^2 = 0$, (b) $\sigma_n^2 = 1$. Comment on the results obtained in each case.

15. Perform the equalization of a channel with the following transfer function

$$H(z) = \frac{z^2 - 1.359z + 0.81}{z^2 - 1.919z + 0.923}$$

using a known training signal that consists of a binary (-1,1) random signal. An additional Gaussian white noise with variance 10^{-2} is present at the channel output.
(a) Apply a Newton-based algorithm with direct form structure.
(b) Plot the magnitude response of the cascade of the channel and the adaptive filter transfer functions. Comment on the result.

16. In a system identification problem the input signal is generated by an autoregressive process given by

$$x(k) = -1.2x(k-1) - 0.81x(k-2) + n_x(k)$$

where $n_x(k)$ is a zero-mean Gaussian white noise with variance such that $\sigma_x^2 = 1$. The unknown system is described by

$$H(z) = \frac{80z^3(z^2 + 0.81)(z - 0.9)}{(z^2 - 0.71z + 0.25)(z^2 + 0.75z + 0.56)(z^2 - 0.2z + 0.81)}$$

The adaptive filter is also a sixth-order IIR filter.
Choose an appropriate μ, run an ensemble of 20 experiments, and plot the average learning curve. Use the RLS algorithm for IIR filters.

17. A second-order IIR adaptive filtering algorithm was applied to identify a 3rd-order time-varying unknown system whose coefficients are first-order Markov processes with $\lambda_{\mathbf{W}} = 0.999$ and $\sigma_{\mathbf{W}}^2 = 0.001$. The initial time-varying system multiplier coefficients are

$$\mathbf{w}_o^T = [0.03490 \ -0.011 \ -0.06864 \ 0.22391]$$

The input signal is a Gaussian white noise with variance $\sigma_x^2 = 0.7$, and the measurement noise is also a Gaussian noise independent of the input signal and of the elements of $\mathbf{n}_{\mathbf{W}}(k)$, with variance $\sigma_n^2 = 0.01$.
Simulate the experiment described and plot the learning curve.

18. Suppose a second-order IIR digital filter, with multiplier coefficients given below, was identified by an adaptive IIR filter of the same order using the gradient algorithm. Considering that fixed-point arithmetic was used, measure the values of $||\Delta\boldsymbol{\theta}(k)_Q||$ and $\xi(k)_Q$ for the following case.

Additional noise: white noise with variance	$\sigma_n^2 = 0.0015$
Coefficient wordlength:	$b_c = 16$ bits
Signal wordlength:	$b_d = 16$ bits
Input signal: Gaussian white noise with variance	$\sigma_x^2 = 0.7$

$$H(z) = \frac{z^2 - 1.804z + 1}{z^2 - 1.793z + 0.896}$$

19. Repeat the problem above for the following cases
(a) $\sigma_n^2 = 0.01$, $b_c = 9$ bits, $b_d = 9$ bits, $\sigma_x^2 = 0.7$.
(b) $\sigma_n^2 = 0.1$, $b_c = 10$ bits, $b_d = 10$ bits, $\sigma_x^2 = 0.8$.
(c) $\sigma_n^2 = 0.05$, $b_c = 8$ bits, $b_d = 16$ bits, $\sigma_x^2 = 0.8$.

20. Replace the direct form structure in problem 18 by the lattice structure, and comment on the results.

21. For the examples in problem 19 plot the learning curves for the finite- and infinite-precision implementations. Also plot $||\Delta\boldsymbol{\theta}(k)||^2$ versus k in both cases.

22. Repeat problem 8 using the LMSEE algorithm.

23. Show the inequality in (9.88).

24. Repeat problem 15 using the LMSEE algorithm.

25. Repeat problem 8 using a gradient-type algorithm based on the SM method.

26. Repeat problem 15 using a gradient-type algorithm based on the SM method.

27. Derive the RLS-type algorithm based on the SM method.

INDEX