Adaptive Digital Filters

Second Edition, Revised and Expanded

Maurice G. Bellanger
Conservatoire National des Arts et Métiers (CNAM)
Paris, France

MARCEL DEKKER, INC. NEW YORK • BASEL

The first edition was published as *Adaptive Digital Filters and Signal Analysis*, Maurice G. Bellanger (Marcel Dekker, Inc., 1987).

ISBN: 0-8247-0563-7

This book is printed on acid-free paper.

Headquarters
Marcel Dekker, Inc.
270 Madison Avenue, New York, NY 10016
tel: 212-696-9000; fax: 212-685-4540

Eastern Hemisphere Distribution
Marcel Dekker AG
Hutgasse 4, Postfach 812, CH-4001 Basel, Switzerland
tel: 41-61-261-8482; fax: 41-61-261-8896

World Wide Web
http://www.dekker.com

Current printing (last digit):
10 9 8 7 6 5 4 3 2 1

PRINTED IN THE UNITED STATES OF AMERICA

Signal Processing and Communications

Series Introduction

Over the past 50 years, digital signal processing has evolved as a major engineering discipline. The fields of signal processing have grown from the origin of fast Fourier transform and digital filter design to statistical spectral analysis and array processing, and image, audio, and multimedia processing, and shaped developments in high-performance VLSI signal processor design. Indeed, there are few fields that enjoy so many applications—signal processing is everywhere in our lives.

When one uses a cellular phone, the voice is compressed, coded, and modulated using signal processing techniques. As a cruise missile winds along hillsides searching for the target, the signal processor is busy processing the images taken along the way. When we are watching a movie in HDTV, millions of audio and video data are being sent to our homes and received with unbelievable fidelity. When scientists compare DNA samples, fast pattern recognition techniques are being used. On and on, one can see the impact of signal processing in almost every engineering and scientific discipline.

Because of the immense importance of signal processing and the fast-growing demands of business and industry, this series on signal processing serves to report up-to-date developments and advances in the field. The topics of interest include but are not limited to the following:

- Signal theory and analysis
- Statistical signal processing
- Speech and audio processing
- Image and video processing
- Multimedia signal processing and technology
- Signal processing for communications
- Signal processing architectures and VLSI design

I hope this series will provide the interested audience with high-quality, state-of-the-art signal processing literature through research monographs, edited books, and rigorously written textbooks by experts in their fields.

K. J. Ray Liu

Preface

The main idea behind this book, and the incentive for writing it, is that strong connections exist between adaptive filtering and signal analysis, to the extent that it is not realistic—at least from an engineering point of view—to separate them. In order to understand adaptive filters well enough to design them properly and apply them successfully, a certain amount of knowledge of the analysis of the signals involved is indispensable. Conversely, several major analysis techniques become really efficient and useful in products only when they are designed and implemented in an adaptive fashion. This book is dedicated to the intricate relationships between these two areas. Moreover, this approach can lead to new ideas and new techniques in either field.

The areas of adaptive filters and signal analysis use concepts from several different theories, among which are estimation, information, and circuit theories, in connection with sophisticated mathematical tools. As a consequence, they present a problem to the application-oriented reader. However, if these concepts and tools are introduced with adequate justification and illustration, and if their physical and practical meaning is emphasized, they become easier to understand, retain, and exploit. The work has therefore been made as complete and self-contained as possible, presuming a background in discrete time signal processing and stochastic processes.

The book is organized to provide a smooth evolution from a basic knowledge of signal representations and properties to simple gradient algorithms, to more elaborate adaptive techniques, to spectral analysis methods, and finally to implementation aspects and applications. The characteristics of determinist, random, and natural signals are given in Chapter 2, and fundamental results for analysis are derived. Chapter 3 concentrates on the correlation matrix and spectrum and their relationships; it is intended to familiarize the reader with concepts and properties that have to be fully understood for an in-depth knowledge of necessary adaptive techniques in engineering. The gradient or least mean squares (LMS) adaptive filters are treated in Chapter 4. The theoretical aspects, engineering design options, finite word-length effects, and implementation structures are covered in turn. Chapter 5 is entirely devoted to linear prediction theory and techniques, which are crucial in deriving and understanding fast algorithms operations. Fast least squares (FLS) algorithms of the transversal type are derived and studied in Chapter 6, with emphasis on design aspects and performance. Several complementary algorithms of the same family are presented in Chapter 7 to cope with various practical situations and signal types.

Time and order recursions that lead to FLS lattice algorithms are presented in Chapter 8, which ends with an introduction to the unified geometric approach for deriving all sorts of FLS algorithms. In other areas of signal processing, such as multirate filtering, it is known that rotations provide efficiency and robustness. The same applies to adaptive filtering, and rotation based algorithms are presented in Chapter 9. The relationships with the normalized lattice algorithms are pointed out. The major spectral analysis and estimation techniques are described in Chapter 10, and the connections with adaptive methods are emphasized. Chapter 11 discusses circuits and architecture issues, and some illustrative applications, taken from different technical fields, are briefly presented, to show the significance and versatility of adaptive techniques. Finally, Chapter 12 is devoted to the field of communications, which is a major application area.

At the end of several chapters, FORTRAN listings of computer subroutines are given to help the reader start practicing and evaluating the major techniques.

The book has been written with engineering in mind, so it should be most useful to practicing engineers and professional readers. However, it can also be used as a textbook and is suitable for use in a graduate course. It is worth pointing out that researchers should also be interested, as a number of new results and ideas have been included that may deserve further work.

I am indebted to many friends and colleagues from industry and research for contributions in various forms and I wish to thank them all for their help. For his direct contributions, special thanks are due to J. M. T. Romano, Professor at the University of Campinas in Brazil.

Maurice G. Bellanger

Contents

1
Adaptive Filtering and Signal Analysis

Digital techniques are characterized by flexibility and accuracy, two properties which are best exploited in the rapidly growing technical field of adaptive signal processing.

Among the processing operations, linear filtering is probably the most common and important. It is made adaptive if its parameters, the coefficients, are varied according to a specified criterion as new information becomes available. That updating has to follow the evolution of the system environment as fast and accurately as possible, and, in general, it is associated with real-time operation. Applications can be found in any technical field as soon as data series and particularly time series are available; they are remarkably well developed in communications and control.

Adaptive filtering techniques have been successfully used for many years. As users gain more experience from applications and as signal processing theory matures, these techniques become more and more refined and sophisticated. But to make the best use of the improved potential of these techniques, users must reach an in-depth understanding of how they really work, rather than simply applying algorithms. Moreover, the number of algorithms suitable for adaptive filtering has grown enormously. It is not unusual to find more than a dozen algorithms to complete a given task. Finding the best algorithm is a crucial engineering problem. The key to properly using adaptive techniques is an intimate knowledge of signal makeup. That is why signal analysis is so tightly connected to adaptive processing. In reality, the class of the most performant algorithms rests on a real-time analysis of the signals to be processed.

Conversely, adaptive techniques can be efficient instruments for performing signal analysis. For example, an adaptive filter can be designed as an intelligent spectrum analyzer.

So, for all these reasons, it appears that learning adaptive filtering goes with learning signal analysis, and both topics are jointly treated in this book.

First, the signal analysis problem is stated in very general terms.

1.1. SIGNAL ANALYSIS

By definition a signal carries information from a source to a receiver. In the real world, several signals, wanted or not, are transmitted and processed together, and the signal analysis problem may be stated as follows.

Let us consider a set of N sources which produce N variables $x_0, x_1, \dots, x_{N-1}$ and a set of N corresponding receivers which give N variables $y_0, y_1, \dots, y_{N-1}$, as shown in Figure 1.1. The transmission medium is assumed to be linear, and every receiver variable is a linear combination of the source variables:

$$y_i = \sum_{j=0}^{N-1} m_{ij} x_j, \qquad 0 \leqslant i \leqslant N-1 \tag{1.1}$$

The parameters m_{ij} are the transmission coefficients of the medium.

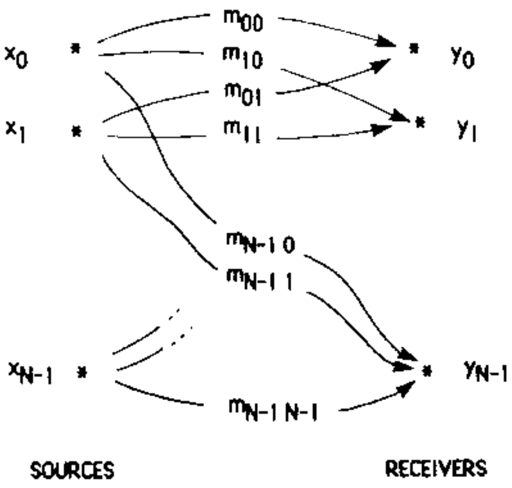


FIG. 1.1 A transmission system of order N.

Now the problem is how to retrieve the source variables, assumed to carry the useful information looked for, from the receiver variables. It might also be necessary to find the transmission coefficients. Stated as such, the problem might look overly ambitious. It can be solved, at least in part, with some additional assumptions.

For clarity, conciseness, and thus simplicity, let us write equation (1.1) in matrix form:

$$Y = MX \tag{1.2}$$

with

$$X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}, \quad Y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix}$$

$$M = \begin{bmatrix} m_{00} & m_{01} & \cdots & m_{0\,N-1} \\ m_{10} & m_{11} & \cdots & m_{1\,N-1} \\ \vdots & & & \vdots \\ m_{N-10} & \cdots & & m_{N-1\,N-1} \end{bmatrix}$$

Now assume that the x_i are random centered uncorrelated variables and consider the $N \times N$ matrix

$$YY^t = MXX^tM^t \tag{1.3}$$

where M^t denotes the transpose of the matrix M. Taking its mathematical expectation and noting that the transmission coefficients are deterministic variables, we get

$$E[YY^t] = ME[XX^t]M^t \tag{1.4}$$

Since the variables $x_i (0 \leqslant i \leqslant N-1)$ are assumed to be uncorrelated, the $N \times N$ source matrix is diagonal:

$$E[XX^t] = \begin{bmatrix} P_{x_0} & 0 & \cdots & 0 \\ 0 & P_{x_1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & P_{x_{N-1}} \end{bmatrix} = \mathrm{diag}[P_{x_0}, P_{x_1}, \ldots, P_{x_{N-1}}]$$

where

$$P_{x_i} = E[x_i^2]$$

is the power of the source with index i. Thus, a decomposition of the receiver covariance matrix has been achieved:

$$E[YY^t] = M\,\mathrm{diag}[P_{x_0}, P_{x_1}, \ldots, P_{x_{N-1}}]M^t \tag{1.5}$$

Finally, it appears possible to get the source powers and the transmission matrix from the diagonalization of the covariance matrix of the receiver variables. In practice, the mathematical expectation can be reached, under suitable assumptions, by repeated measurements, for example. It is worth noticing that if the transmission medium has no losses, the power of the sources is transferred to the receiver variables in totality, which corresponds to the relation $MM^t = I_N$; the transmission matrix is unitary in that case.

In practice, useful signals are always corrupted by unwanted externally generated signals, which are classified as noise. So, besides useful signal sources, noise sources have to be included in any real transmission system. Consequently, the number of sources can always be adjusted to equal the number of receivers. Indeed, for the analysis to be meaningful, the number of receivers must exceed the number of useful sources.

The technique presented above is used in various fields for source detection and location (for example, radio communications or acoustics); the set of receivers is an array of antennas. However, the same approach can be applied as well to analyze a signal sequence when the data $y(n)$ are linear combinations of a set of basic components. The problem is then to retrieve these components. It is particularly simple when $y(n)$ is periodic with period N, because then the signal is just a sum of sinusoids with frequencies that are multiples of $1/N$, and the matrix M in decomposition (1.5) is the discrete Fourier transform (DFT) matrix, the diagonal terms being the power spectrum. For an arbitrary set of data, the decomposition corresponds to the representation of the signal as sinusoids with arbitrary frequencies in noise; it is a harmonic retrieval operation or a principal component analysis procedure.

Rather than directly searching for the principal components of a signal to analyze it, extract its information, condense it, or clear it from spurious noise, we can approximate it by the output of a model, which is made as simple as possible and whose parameters are attributed to the signal. But to apply that approach, we need some characterization of the signal.

1.2. CHARACTERIZATION AND MODELING

A straightforward way to characterize a signal is by waveform parameters. A concise representation is obtained when the data are simple functions of the index n. For example, a sinusoid is expressed by

$$x(n) = S\sin(n\omega + \varphi) \tag{1.6}$$

where S is the sinusoid amplitude, ω is the angular frequency, and φ is the phase. The same signal can also be represented and generated by the recurrence relation

$$x(n) = (2\cos\omega)x(n-1) - x(n-2) \tag{1.7}$$

for $n \geqslant 0$, and the initial conditions

$$\begin{aligned} x(-1) &= S\sin(-\omega + \varphi) \\ x(-2) &= S\sin(-2\omega + \varphi) \\ x(n) &= 0 \quad \text{for } n < -2 \end{aligned}$$

Recurrence relations play a key role in signal modeling as well as in adaptive filtering. The correspondence between time domain sequences and recurrence relations is established by the z-transform, defined by

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \tag{1.8}$$

Waveform parameters are appropriate for synthetic signals, but for practical signal analysis the correlation function $r(p)$, in general, contains the relevant characteristics, as pointed out in the previous section:

$$r(p) = E[x(n)x(n-p)] \tag{1.9}$$

In the analysis process, the correlation function is first estimated and then used to derive the signal parameters of interest, the spectrum, or the recurrence coefficients.

The recurrence relation is a convenient representation or modeling of a wide class of signals, which are those obtained through linear digital filtering of a random sequence. For example, the expression

$$x(n) = e(n) - \sum_{i=1}^{N} a_i x(n-i) \tag{1.10}$$

where $e(n)$ is a random sequence or noise input, defines a model called autoregressive (AR). The corresponding filter is of the infinite impulse response (IIR) type. If the filter is of the finite impulse response (FIR) type, the model is called moving average (MA), and a general filter FIR/IIR is associated to an ARMA model.

The coefficients a_i in (1.10) are the FIR, or transversal, linear prediction coefficients of the signal $x(n)$; they are actually the coefficients of the inverse FIR filter defined by

$$e(n) = \sum_{i=0}^{N} a_i x(n-i), \qquad a_0 = 1 \tag{1.11}$$

The sequence $e(n)$ is called the prediction error signal. The coefficients are designed to minimize the prediction error power, which, expressed as a matrix form equation is

$$E[e^2(n)] = A^t E[XX^t]A \tag{1.12}$$

So, for a given signal whose correlation function is known or can be estimated, the linear prediction (or AR modeling) problem can be stated as follows: find the coefficient vector A which minimizes the quantity $A^t E[XX^t]A$ subject to the constraint $a_0 = 1$. In that process, the power of a white noise added to the useful input signal is magnified by the factor $A^t A$.

To provide a link between the direct analysis of the previous section and AR modeling, and to point out their major differences and similarities, we note that the harmonic retrieval, or principal component analysis, corresponds to the following problem: find the vector A which minimizes the value $A^t E[XX^t]A$ subject to the constraint $A^t A = 1$. The frequencies of the sinusoids in the signal are then derived from the zeros of the filter with coefficient vector A. For deterministic signals without noise, direct analysis and AR modeling lead to the same solution; they stay close to each other for high signal-to-noise ratios.

The linear prediction filter plays a key role in adaptive filtering because it is directly involved in the derivation and implementation of least squares (LS) algorithms, which in fact are based on real-time signal analysis by AR modeling.

1.3. ADAPTIVE FILTERING

The principle of an adaptive filter is shown in Figure 1.2. The output of a programmable, variable-coefficient digital filter is subtracted from a reference signal $y(n)$ to produce an error sequence $e(n)$, which is used in combination with elements of the input sequence $x(n)$, to update the filter coefficients, following a criterion which is to be minimized. The adaptive filters can be classified according to the options taken in the following areas:

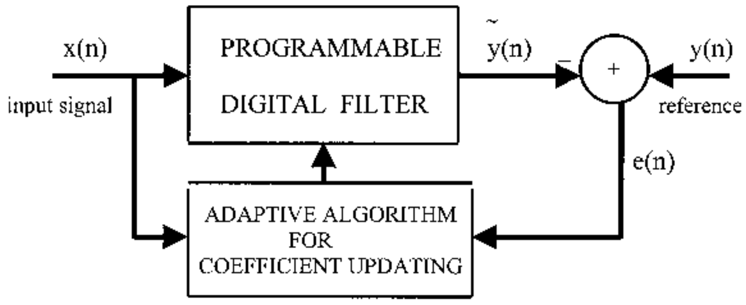

x(n)
input signal
PROGRAMMABLE DIGITAL FILTER
$\tilde{y}(n)$
−
+
y(n)
reference
e(n)
ADAPTIVE ALGORITHM FOR COEFFICIENT UPDATING

FIG. 1.2 Principle of an adaptive filter.

The optimization criterion
The algorithm for coefficient updating
The programmable filter structure
The type of signals processed—mono- or multidimensional.

The optimization criterion is in general taken in the LS family in order to work with linear operations. However, in some cases, where simplicity of implementation and robustness are of major concern, the least absolute value (LAV) criterion can also be attractive; moreover, it is not restricted to minimum phase optimization.

The algorithms are highly dependent on the optimization criterion, and it is often the algorithm that governs the choice of the optimization criterion, rather than the other way round. In broad terms, the least mean squares (LMS) criterion is associated with the gradient algorithm, the LAV criterion corresponds to a sign algorithm, and the exact LS criterion is associated with a family of recursive algorithms, the most efficient of which are the fast least squares (FLS) algorithms.

The programmable filter can be a FIR or IIR type, and, in principle, it can have any structure: direct form, cascade form, lattice, ladder, or wave filter. Finite word-length effects and computational complexity vary with the structure, as with fixed coefficient filters. But the peculiar point with adaptive filters is that the structure reacts on the algorithm complexity. It turns out that the direct-form FIR, or transversal, structure is the simplest to study and implement, and therefore it is the most popular.

Multidimensional signals can use the same algorithms and structures as their monodimensional counterparts. However, computational complexity constraints and hardware limitations generally reduce the options to the simplest approaches.

The study of adaptive filtering begins with the derivation of the normal equations, which correspond to the LS criterion combined with the FIR direct form for the programmable filter.

1.4. NORMAL EQUATIONS

In the following, we assume that real-time series, resulting, for example, from the sampling with period $T = 1$ of a continuous-time real signal, are processed.

Let $H(n)$ be the vector of the N coefficients $h_i(n)$ of the programmable filter at time n, and let $X(n)$ be the vector of the N most recent input signal samples:

$$H(n)\begin{bmatrix} h_0(n) \\ h)_1(n) \\ \vdots \\ h_{N-1}(n) \end{bmatrix}, \quad X(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n+1-N) \end{bmatrix} \tag{1.13}$$

The error signal $\varepsilon(n)$ is

$$\varepsilon(n) = y(n) - H^t(n)X(n) \tag{1.14}$$

The optimization procedure consists of minimizing, at each time index, a cost function $J(n)$, which, for the sake of generality, is taken as a weighted sum of squared error signal values, beginning after time zero:

$$J(n) = \sum_{p=1}^{n} W^{n-p}[y(p) - H^t(n)X(p)]^2 \tag{1.15}$$

The weighting factor, W, is generally taken close to $1 (0 \ll W \leqslant 1)$.

Now, the problem is to find the coefficient vector $H(n)$ which minimizes $J(n)$. The solution is obtained by setting to zero the derivatives of $J(n)$ with respect to the entries $h_i(n)$ of the coefficient vector $H(n)$, which leads to

$$\sum_{p=1}^{n} W^{n-p}[y(p) - H^t(n)X(p)]X(p) = 0 \tag{1.16}$$

In concise form, (1.16) is

$$H(n) = R_N^{-1}(n)r_{yx}(n) \tag{1.17}$$

with

$$R_N(n) = \sum_{p=1}^{n} W^{n-p}X(p)X^t(p) \tag{1.18}$$

$$r_{yx}(n) = \sum_{p=1}^{n} W^{n-p} X(p) y(p) \tag{1.19}$$

If the signals are stationary, let R_{xx} be the $N \times N$ input signal autocorrelation matrix and let r_{yx} be the vector of cross-correlations between input and reference signals:

$$R_{xx} = E[X(p)X^t(p)], \quad r_{yx} = E[X(p)y(p)] \tag{1.20}$$

Now

$$E[R_N(n)] = \frac{1 - W^n}{1 - W} R_{xx}, \quad E[r_{yx}(n)] = \frac{1 - W^n}{1 - W} r_{yx} \tag{1.21}$$

So $R_N(n)$ is an estimate of the input signal autocorrelation matrix, and $r_{yx}(n)$ is an estimate of the cross-correlation between input and reference signals.

The optimal coefficient vector H_{opt} is reached when n goes to infinity:

$$H_{\text{opt}} = R_{xx}^{-1} r_{yx} \tag{1.22}$$

Equations (1.22) and (1.17) are the normal (or Yule–Walker) equations for stationary and evolutive signals, respectively. In adaptive filters, they can be implemented recursively.

1.5. RECURSIVE ALGORITHMS

The basic goal of recursive algorithms is to derive the coefficient vector $H(n+1)$ from $H(n)$. Both coefficient vectors satisfy (1.17). In these equations, autocorrelation matrices and cross-correlation vectors satisfy the recursive relations

$$R_N(n+1) = W R_N(n) + X(n+1)X^t(n+1) \tag{1.23}$$

$$r_{yx}(n+1) = W r_{yx}(n) + X(n+1)y(n+1) \tag{1.24}$$

Now,

$$H(n+1) = R_N^{-1}(n+1)[W r_{yx}(n) + X(n+1)y(n+1)]$$

But

$$W r_{yx}(n) = [R_N(n+1) - X(n+1)X^t(n+1)]H(n)$$

and

$$H(n+1) = H(n) + R_N^{-1}(n+1)X(n+1)[y(n+1) - X^t(n+1)H(n)] \tag{1.25}$$

which is the recursive relation for the coefficient updating. In that expression, the sequence

$$e(n+1) = y(n+1) - X^t(n+1)H(n) \tag{1.26}$$

is called the a priori error signal because it is computed by using the coefficient vector of the previous time index. In contrast, (1.14) defines the a posteriori error signal $\varepsilon(n)$, which leads to an alternative type of recurrence equation

$$H(n+1) = H(n) + W^{-1}R_N^{-1}(n)X(n+1)e(n+1) \tag{1.27}$$

For large values of the filter order N, the matrix manipulations in (1.25) or (1.27) lead to an often unacceptable hardware complexity. We obtain a drastic simplification by setting

$$R_N^{-1}(n+1) \approx \delta I_N$$

where I_N is the $(N \times N)$ unity matrix and δ is a positive constant called the adaptation step size. The coefficients are then updated by

$$H(n+1) = H(n) + \delta X(n+1)e(n+1) \tag{1.28}$$

which leads to just doubling the computations with respect to the fixed-coefficient filter. The optimization process no longer follows the exact LS criterion, but LMS criterion. The product $X(n+1)e(n+1)$ is proportional to the gradient of the square of the error signal with opposite sign, because differentiating equation (1.26) leads to

$$-\frac{\partial e^2(n+1)}{\partial h_i(n)} = 2x(n+1-i)e(n+1), \quad 0 \leqslant i \leqslant N-1 \tag{1.29}$$

hence the name gradient algorithm.

The value of the step size δ has to be chosen small enough to ensure convergence; it controls the algorithm speed of adaptation and the residual error power after convergence. It is a trade-off based on the system engineering specifications.

The gradient algorithm is useful and efficient in many applications; it is flexible, can be adjusted to all filter structures, and is robust against implementation imperfections. However, it has some limitations in performance and weaknesses which might not be tolerated in various applications. For example, its initial convergence is slow, its performance depends on the input signal statistics, and its residual error power may be large. If one is prepared to accept an increase in computational complexity by a factor usually smaller than an order of magnitude (typically 4 or 5), then the exact recursive LS algorithm can be implemented. The matrix manipulations can be avoided in the coefficient updating recursion by introducing the vector

$$G(n) = R_N^{-1}(n)X(n) \tag{1.30}$$

called the adaptation gain, which can be updated with the help of linear prediction filters. The corresponding algorithms are called FLS algorithms.

Up to now, time recursions have been considered, based on the cost function $J(n)$ defined by equation (1.15) for a set of N coefficients. It is also possible to work out order recursions which lead to the derivation of the coefficients of a filter of order $N+1$ from the set of coefficients of a filter of order N. These order recursions rely on the introduction of a different set of filter parameters, called the partial correlation (PARCOR) coefficients, which correspond to the lattice structure for the programmable filter. Now, time and order recursions can be combined in various ways to produce a family of LS lattice adaptive filters. That approach has attractive advantages from the theoretical point of view—for example, signal orthogonalization, spectral whitening, and easy control of the minimum phase property—and also from the implementation point of view, because it is robust to word-length limitations and leads to flexible and modular realizations.

The recursive techniques can easily be extended to complex and multidimensional signals. Overall, the adaptive filtering techniques provide a wide range of means for fast and accurate processing and analysis of signals.

1.6. IMPLEMENTATION AND APPLICATIONS

The circuitry designed for general digital signal processing can also be used for adaptive filtering and signal analysis implementation. However, a few specificities are worth point out. First, several arithmetic operations, such as divisions and square roots, become more frequent. Second, the processing speed, expressed in millions of instructions per second (MIPS) or in millions of arithmetic operations per second (MOPS), depending on whether the emphasis is on programming or number crunching, is often higher than average in the field of signal processing. Therefore specific efficient architectures for real-time operation can be worth developing. They can be special multibus arrangements to facilitate pipelining in an integrated processor or powerful, modular, locally interconnected systolic arrays.

Most applications of adaptive techniques fall into one of two broad classes: system identification and system correction.

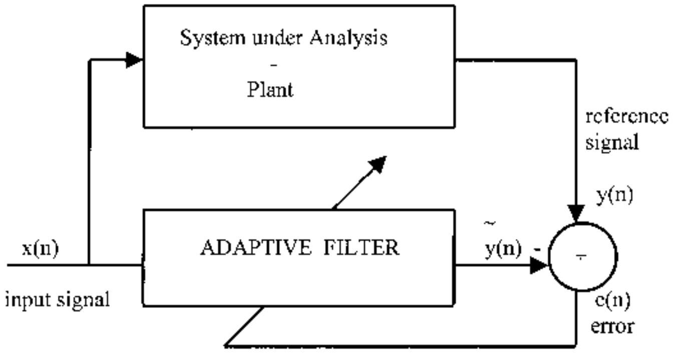


FIG. 1.3 Adaptive filter for system identification.

The block diagram of the configuration for system identification is shown in Figure 1.3. The input signal $x(n)$ is fed to the system under analysis, which produces the reference signal $y(n)$. The adaptive filter parameters and specifications have to be chosen to lead to a sufficiently good model for the system under analysis. That kind of application occurs frequently in automatic control.

System correction is shown in Figure 1.4. The system output is the adaptive filter input. An external reference signal is needed. If the reference signal $y(n)$ is also the system input signal $u(n)$, then the adaptive filter is an inverse filter; a typical example of such a situation can be found in communications, with channel equalization for data transmission. In both application classes, the signals involved can be real or complex valued, mono- or multidimensional. Although the important case of linear prediction for signal analysis can fit into either of the aforementioned categories, it is often considered as an inverse filtering problem, with the following choice of signals: $y(n) = 0$, $u(n) = e(n)$.

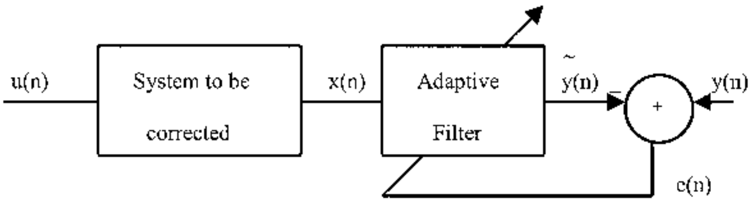


FIG. 1.4 Adaptive filter for system correction.

Another field of applications corresponds to the restoration of signals which have been degraded by addition of noise and convolution by a known or estimated filter. Adaptive procedures can achieve restoration by deconvolution.

The processing parameters vary with the class of application as well as with the technical fields. The computational complexity and the cost efficiency often have a major impact on final decisions, and they can lead to different options in control, communications, radar, underwater acoustics, biomedical systems, broadcasting, or the different areas of applied physics.

1.7. FURTHER READING

The basic results, which are most necessary to read this book, in signal processing, mathematics, and statistics are recalled in the text as close as possible to the place where they are used for the first time, so the book is, to a large extent, self-sufficient. However, the background assumed is a working knowledge of discrete-time signals and systems and, more specifically, random processes, discrete Fourier transform (DFT), and digital filter principles and structures. Some of these topics are treated in [1]. Textbooks which provide thorough treatment of the above-mentioned topics are [2–4]. A theoretical veiw of signal analysis is given in [5], and spectral estimation techniques are described in [6]. Books on adaptive algorithms include [7–9]. Various applications of adaptive digital filters in the field of communications are presented in [10–11].

REFERENCES

1. M. Bellanger, *Digital Processing of Signals — Theory and Practice* (3rd edn), John Wiley, Chichester, 1999.
2. A. V. Oppenheim, S. A. Willsky, and I. T. Young, *Signals and Systems*, Prentice-Hall, Englewood Cliffs, N.J., 1983.
3. S. K. Mitra and J. F. Kaiser, *Handbook for Digital Signal Processing*, John Wiley, New York, 1993.
4. G. Zeilniker and F. J. Taylor, *Advanced Digital Signal Processing*, Marcel Dekker, New York, 1994.
5. A. Papoulis, *Signal Analysis*, McGraw-Hill, New York, 1977.
6. L. Marple, *Digital Spectrum Analysis with Applications*, Prentice-Hall, Englewood Cliffs, N.J., 1987.
7. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, N.J., 1985.
8. S. Haykin, *Adaptive Filter Theory* (3rd edn), Prentice-Hall, Englewood Cliffs, N.J., 1996.

9. P. A. Regalia, *Adaptive IIR Filtering in Signal Processing and Control*, Marcel Dekker, New York, 1995.
10. C. F. N. Cowan and P. M. Grant, *Adaptive Filters*, Prentice-Hall, Englewood Cliffs, N.J., 1985.
11. O. Macchi, *Adaptive Processing: the LMS Approach with Applications in Transmission*, John Wiley, Chichester, 1995.

2
Signals and Noise

Signals carry information from sources to receivers, and they take many different forms. In this chapter a classification is presented for the signals most commonly used in many technical fields.

A first distinction is between useful, or wanted, signals and spurious, or unwanted, signals, which are often called noise. In practice, noise sources are always present, so any actual signal contains noise, and a significant part of the processing operations is intended to remove it. However, useful signals and noise have many features in common and can, to some extent, follow the same classification.

Only data sequences or time series are considered here, and the leading thread for the classification proposed is the set of recurrence relations, which can be established between consecutive data and which are the basis of several major analysis methods [1–3]. In the various categories, signals can be characterized by waveform functions, autocorrelation, and spectrum.

An elementary, but fundamental, signal is introduced first—the damped sinusoid.

2.1. THE DAMPED SINUSOID

Let us consider the following complex sequence, which is called the damped complex sinusoid, or damped cisoid:

$$y(n) = \begin{cases} e^{(\alpha + j\omega_0)n}, & n \geqslant 0 \\ 0, & n < 0 \end{cases} \tag{2.1}$$

where α and ω_0 are real scalars.

The z-transform of that sequence is, by definition

$$Y(z) = \sum_{n=0}^{\infty} y(n) z^{-n} \tag{2.2}$$

Hence

$$Y(z) = \frac{1}{1 - e^{(\alpha + j\omega_0)} z^{-1}} \tag{2.3}$$

The two real corresponding sequences are shown in Figure 2.1(a). They are

$$y(n) = y_R(n) + j y_I(n) \tag{2.4}$$

with

$$y_R(n) = e^{\alpha n} \cos n\omega_0, \quad y_I(n) = e^{\alpha n} \sin n\omega_0, \quad n \geqslant 0 \tag{2.5}$$

The z-transforms are

$$Y_R(z) = \frac{1 - (e^{\alpha} \cos \omega_0) z^{-1}}{1 - (2e^{\alpha} \cos \omega_0) z^{-1} + e^{2\alpha} z^{-2}} \tag{2.6}$$

$$Y_I(z) = \frac{1 - (e^{\alpha} \sin \omega_0) z^{-1}}{1 - (2e^{\alpha} \cos \omega_0) z^{-1} + e^{2\alpha} z^{-2}} \tag{2.7}$$

In the complex plane, these functions have a pair of conjugate poles, which are shown in Figure 2.1(b) for $\alpha < 0$ and $|\alpha|$ small. From (2.6) and (2.7) and also by direct inspection, it appears that the corresponding signals satisfy the recursion

$$y_R(n) - 2e^{\alpha} \cos \omega_0 y_R(n-1) + 3^{2\alpha} y_R(n-2) = 0 \tag{2.8}$$

with initial values

$$y_R(-1) = e^{-\alpha} \cos(-\omega_0), \quad y_R(-2) = e^{-2\alpha} \cos(-2\omega_0) \tag{2.9}$$

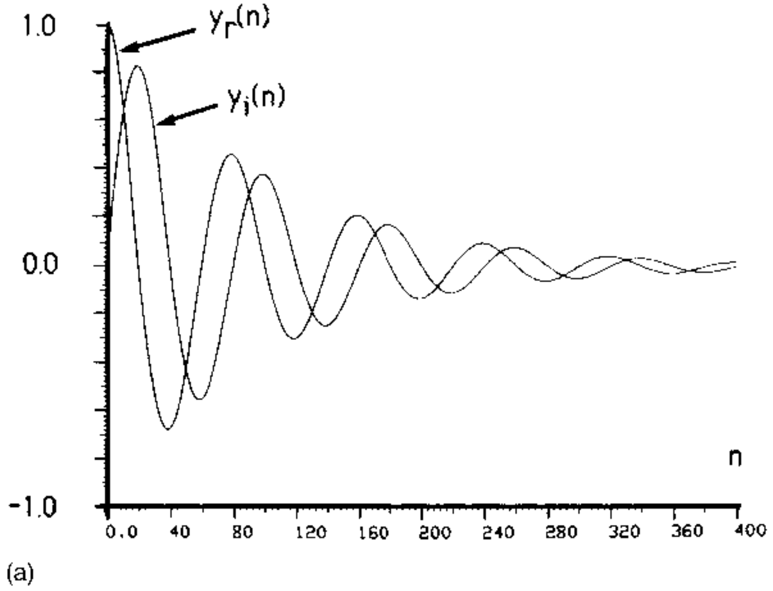
and

$$y_I(-1) = e^{-\alpha} \sin(-\omega_0), \quad y_I(-2) = e^{2\alpha} \sin(-2\omega_0) \tag{2.10}$$

More generally, the one-sided z-transform, as defined by (2.2), of equation (2.8) is

$$Y_R(z) = -\frac{b_1 y_R(-1) + b_2[y_R(-2) + y_R(-1) z^{-1}]}{1 + b_1 z^{-1} + b_2 z^{-2}} \tag{2.11}$$

with $b_1 = -2e^{\alpha} \cos \omega$ and $b_2 = e^{2\alpha}$.



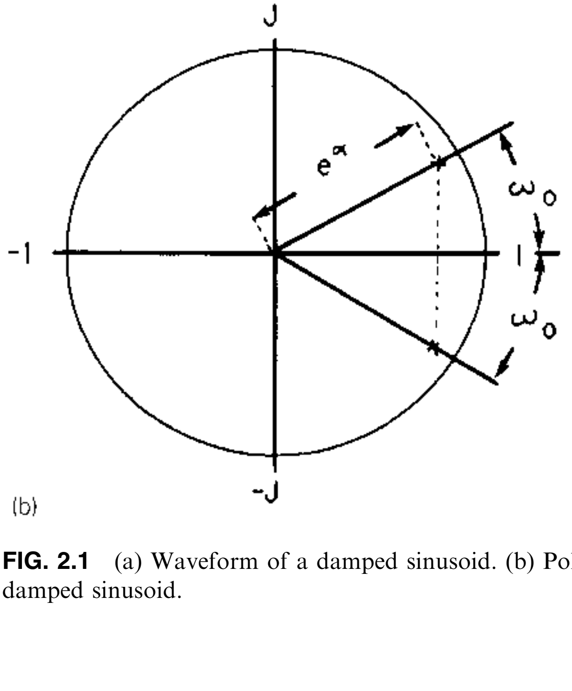


FIG. 2.1 (a) Waveform of a damped sinusoid. (b) Poles of the *z*-transform of the damped sinusoid.

The above-mentioned initial values are then obtained by identifying (2.11) and (2.6), and (2.11) and (2.7), respectively.

The energy spectra of the sequences $y_R(n)$ and $y)_I(n)$ are obtained from the z-transforms by replacing z by $e^{j\omega}$ [4]. For example, the function $|Y_I(\omega)|$ is shown in Figure 2.2; it is the frequency response of a purely recursive second-order filter section.

As n grows to infinity the signal $y(n)$ vanishes; it is nonstationary. Damped sinusoids can be used in signal analysis to approximate the spectrum of a finite data sequence.

2.2. PERIODIC SIGNALS

Periodic signals form an important category, and the simplest of them is the single sinusoid, defined by

$$x(n) = S\sin(n\omega_0 + \varphi) \tag{2.12}$$

where S is the amplitude, ω_0 is the radial frequency, and φ is the phase.

For $n \geqslant 0$, the results of the previous section can be applied with $\alpha = 0$. So the recursion

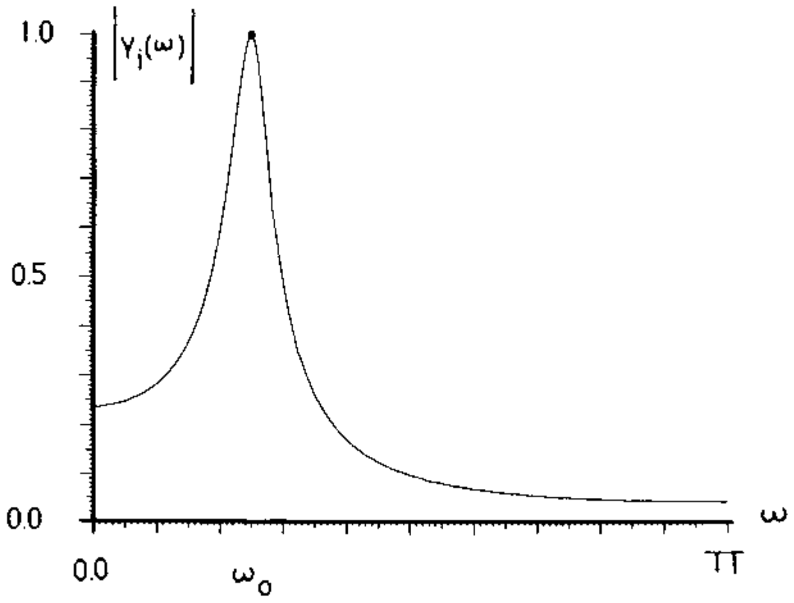


FIG. 2.2 Spectrum of the damped sinusoid.

$$x(n) - 2\cos\omega_0 x(n-1) + x(n-2) = 0 \tag{2.13}$$

with initial conditions

$$x(-1) = S\sin(-\omega_0 + \varphi), \quad x(-2) = S\sin(-2\omega_0 + \varphi) \tag{2.14}$$

is satisfied. The z-transform is

$$X(z) = S\frac{\sin\varphi - \sin(-\omega_0 + \varphi)z^{-1}}{1 - (2\cos\omega_0)z^{-1} + z^{-2}} \tag{2.15}$$

Now the poles are exactly on the unit circle, and we must consider the power spectrum. It cannot be directly derived from the z-transform. The sinusoid is generated for $n > 0$ by the purely recursive second-order filter section in Figure 2.3 with the above-mentioned initial conditions, the circuit input being zero. For a filter to cancel a sinusoid, it is necessary and sufficient to implement the inverse filter—that is, a filter which has a pair of zeros on the unit circle at the frequency of the sinusoid; such filters appear in linear prediction.

The autocorrelation function (ACF) of the sinusoid, which is a real signal, is defined by

$$r(p) = \lim_{N\to\infty} \frac{1}{N}\sum_{n=0}^{N-1} x(n)x(n-p) \tag{2.16}$$

Hence,

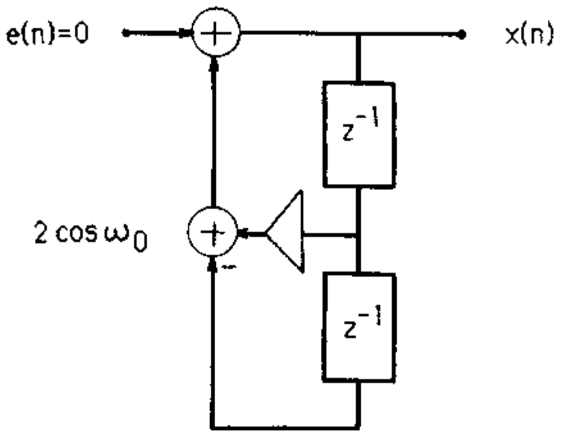


FIG. 2.3 Second-order filter section to generate a sinusoid.

$$r(p) = \frac{S^2}{2} \cos p\omega_0 - \lim_{N \to \infty} \frac{1}{N} \frac{S^2}{2} \sum_{n=0}^{N-1} \cos 2\left(\frac{2n-p}{2}\omega_0 + \varphi\right) \tag{2.17}$$

and for any ω_0,

$$r(p) = \frac{S^2}{2} \cos p\omega_0 \tag{2.18}$$

The power spectrum of the signal is the Fourier transform of the ACF; for the sinusoid it is a line with magnitude $S^2/2$ at frequency ω_0.

Now, let us proceed to periodic signals. A periodic signal with period N consists of a sum of complex sinusoids, or cisoids, whose frequencies are integer multiples of $1/N$ and whose complex amplitudes S_k are given by the discrete Fourier transform (DFT) of the signal data:

$$\begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_{N-1} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W & \cdots & W^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W^{N-1} & \cdots & W^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix} \tag{2.19}$$

with $W = e^{-j(2\pi/N)}$.

Following equation (2.3), with $\alpha = 0$, we express the z-transform of the periodic signal by

$$X(z) = \sum_{k=0}^{N-1} \frac{S_k}{1 - e^{j(2\pi/N)k} z^{-1}} \tag{2.20}$$

and its poles are uniformly distributed on the unit circle as shown in Figure 2.4 for N even. Therefore, the signal $x(n)$ satisfies the recursion

$$\sum_{i=0}^{N} a_i x(n-i) = 0 \tag{2.21}$$

where the a_i are the coefficients of the polynomial $P(z)$:

$$P(z) = \sum_{i=0}^{N} a_i z^{-1} = \prod_{k=1}^{N} (1 - e^{j(2\pi/N)k} z^{-1}) \tag{2.22}$$

So $a_0 = 1$, and if all the cisoids are present in the periodic signal, then $a_N = 1$ and $a_i = 0$ for $1 \leqslant i \leqslant N-1$. The N complex amplitudes, or the real amplitudes and phases, are defined by the N initial conditions. If some of the N possible cisoids are missing, then the coefficients take on values according to the factors in the product (2.22).

The ACF of the periodic signal $x(n)$ is calculated from the following expression, valid for complex data:

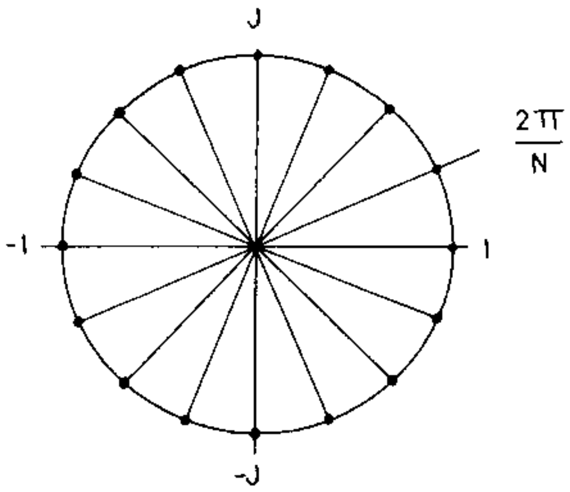


FIG. 2.4 Poles of a signal with period N.

$$r(p) = \frac{1}{N} \sum_{n=0}^{N-1} x(n)\bar{x}(n-p) \tag{2.23}$$

where $\bar{x}(n)$ is the complex conjugate of $x(n)$. According to the inverse DFT, $x(n)$ can be expressed from its frequency components by

$$x(n) = \sum_{k=0}^{N-1} S_k e^{j(2\pi/N)kn} \tag{2.24}$$

Now, combining (2.24) and (2.23) gives

$$r(p) = \sum_{k=0}^{N-1} |S_k|^2 e^{j(2\pi/N)kp} \tag{2.25}$$

and, for $x(n)$ a real signal and for the configuration of poles shown in Figure 2.4 with N even,

$$r(p) = S_0^2 + S_{N/2}^2 + 2 \sum_{k=1}^{N/2-1} |S_k|^2 \cos\left(\frac{2\pi}{N} kp\right) \tag{2.26}$$

The corresponding spectrum is made of lines at frequencies which are integer multiples of $1/N$.

The same analysis as above can be carried out for a signal composed of a sum of sinusoids with arbitrary frequencies, which just implies that the period N may grow to infinity. In that case, the roots of the polynomial $P(z)$ take on arbitrary positions on the unit circle. Such a signal is said to be deterministic because it is completely determined by the recurrence relationship (2.21) and the set of initial conditions; in other words, a signal value at time n can be exactly calculated from the N preceding values; there is no innovation in the process; hence, it is also said to be predictable.

The importance of $P(z)$ is worth emphasizing, because it directly determines the signal recurrence relation. Several methods of analysis primarily aim at finding out that polynomial for a start.

The above deterministic or predictable signals have discrete power spectra. To obtain continuous spectra, one must introduce random signals. They bring innovation in the processes.

2.3. RANDOM SIGNALS

A random real signal $x(n)$ is defined by a probability law for its amplitude at each time n. The law can be expressed as a probability density $p(x, n)$ defined by

$$p(x,n) = \lim_{\Delta x \to 0} \frac{\text{Prob}[x \leqslant x(n) \leqslant x + \Delta x]}{\Delta x} \tag{2.27}$$

It is used to calculate, by ensemble averages, the statistics of the signal or process [5].

The signal is second order if it possesses a first-order moment $m_1(n)$ called the mean value or expectation of $x(n)$, denoted $E[x(n)]$ and defined by

$$m_1(n) = E[x(n)] = \int_{-\infty}^{\infty} xp(x,n)\,dx \tag{2.28}$$

and a second-order moment, called the covariance:

$$E[x(n_1)x(n_2)] = m_2(n_1, n_2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_1 x_2 p(x_1, x_2; n_1, n_2)\,dx_1 dx_2 \tag{2.29}$$

where $p(x_1, x_2; , n_1, n_2)$ is the joint probability density of the pair of random variables $[x(n_1), x(n_2)]$.

The signal is *stationary* if its statistical properties are independent of the time index n—that is, if the probability density is independent of time n:

$$p(x,n) = p(x) \tag{2.30}$$

The stationarity can be limited to the moments of first and second order. Then the signal is *wide-sense stationary*, and it is characterized by the following equations:

$$E[x(n)] = \int_{-\infty}^{\infty} xp(x)\,dx = m_1 \tag{2.31}$$

$$E[x(n)x(n-p)] = r(p) \tag{2.32}$$

The function $r(p)$ is the (ACF) of the signal.

The statistical parameters are, in general, difficult to estimate or measure directly, because of the ensemble averages involved. A reasonably accurate measurement of an ensemble average requires that many process realizations be available or that the experiment be repeated many times, which is often impractical. On the contrary, time averages are much easier to come by, for time series. Therefore the ergodicity property is of great practical importance; it states that, for a stationary signal, ensemble and time averages are equivalent:

$$m_1 = E[x(n)] = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} x(n) \tag{2.33}$$

$$r(p) = E[x(n)x(n-p)] = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} x(n)x(n-p) \tag{2.34a}$$

For complex signals, the ACF is

$$r(p) = E[x(n)\bar{x}(n-p)] = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{-N}^{N} x(n)\bar{x}(n-p) \tag{2.34b}$$

The factor $x(n-p)$ is replaced by its complex conjugate $\bar{x}(n-p)$; note that $r(0)$ is the signal power and is always a real number.

In the literature, the factor $x(n+p)$ is generally taken to define $r(p)$; however, we use $x(n-p)$ throughout this book because it comes naturally in adaptive filtering.

In some circumstances, moments of order $k > 2$ might be needed. They are defined by

$$m_k = \int_{-\infty}^{\infty} x^k p(x)\,dx \tag{2.35}$$

and they can be calculated efficiently through the introduction of a function $F(u)$, called the characteristic function of the random variable x and defined by

$$F(u) = \int_{-\infty}^{\infty} e^{jux} p(x)\,dx \tag{2.36}$$

Using definition (2.35), we obtain the series expansion

$$F(u) = \sum_{k=0}^{\infty} \frac{(ju)^k}{k!} m_k \tag{2.37}$$

Since $F(u)$ is the inverse Fourier transform of the probability density $p(x)$, it can be easy to calculate and can provide the high-order moments of the signal.

The moment of order 4 is used in the definition of the kurtosis K_x, or coefficient of flatness of a probability distribution

$$K_x = \frac{E[x^4(n)]}{E^2[x^2(n)]} \tag{2.38}$$

For example, a binary symmetric distribution (± 1 with equal probability) leads to $K_x = 1$. For the Gaussian distribution of the next section, $K_x = 3$, and for the exponential distribution

$$p(x) = \frac{1}{\sigma\sqrt{2}} e^{-\sqrt{2}|x|/\sigma} \tag{2.39}$$

$K_x = 9$.

An important concept is that of statistical independence of random variables. Two random variables, x_1 and x_2, are independent if and only if their joint density $p(x_1, x_2)$ is the product of the individual probability densities:

$$p(x_1, x_2) = p(x_1)p(x_2) \tag{2.40}$$

which implies the same relationship for the characteristic functions:

$$F(u_1, u_2) = \iint_{-\infty}^{\infty} e^{j(u_1x_1 + u_2x_2)} p(x_1, x_2)\, dx_1 dx_2 \tag{2.41}$$

and

$$F(u_1, u_2) = F(u_1)F(u_2) \tag{2.42}$$

The correlation concept is related to linear dependency. Two noncorrelated variables, such that $E[x_1x_2] = 0$, have no linear dependency. But, in general, that does not mean statistical independency, since higher-order dependency can exist.

Among the probability laws, the Gaussian law has special importance in signal processing.

2.4. GAUSSIAN SIGNALS

A random variable x is said to be normally distributed or Gaussian if its probability law has a density $p(x)$ which follows the normal or Gaussian law:

$$p(x) = \frac{1}{\sigma_x\sqrt{2\pi}} e^{-(x-m)^2/2\sigma_x^2} \tag{2.43}$$

The parameter m is the mean of the variable x; the variance σ_x^2 is the second-order moment of the centered random variable $(x-m)$; σ_x is also called the standard deviation.

The characteristic function of the centered Gaussian variable is

$$F(u) = e^{-\sigma_x^2 u^2/2} \tag{2.44}$$

Now, using the series expansion (2.37), the moments are

$$m_{2k+1} = 0$$

$$m_2 = \sigma_x^2, \quad m_4 = 3\sigma_x^4, \quad m_{2k} = \frac{2k!}{2^k k!}\sigma_x^{2k} \tag{2.45}$$

The normal law can be generalized to multidimensional random variables. The characteristic function of a k-dimensional Gaussian variable $x(x_1, x_2, \dots, x_k)$ is

$$F(u_1, u_2, \dots, u_k) = \exp\left(-\frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k} r_{ij}u_i u_j\right) \tag{2.46}$$

with $r_{ij} = E[x_i x_j]$.

If the variables are not correlated, then they are independent, because $r_{ij} = 0$ for $i \neq j$ and $F(u_1, u_2, \dots, u_k)$ is the product of the characteristic functions. So noncorrelation means independence for Gaussian variables.

A random signal $x(n)$ is said to be Gaussian if, for any set of k time values $n_i (1 \leqslant i \leqslant k)$, the k-dimensional random variable $x = [x(n_1), x(n_2), \dots, x(n_k)]$ is Gaussian. According to (2.46), the probability law of that variable is completely defined by the ACF $r(p)$ of $x(n)$. The power spectral density $S(f)$ is obtained as the Fourier transform of the ACF:

$$S(f) = \sum_{p=-\infty}^{\infty} r(p)e^{-j2\pi pf} \tag{2.47}$$

or, since $r(p)$ is an even function,

$$S(f) = r(0) + 2\sum_{p=1}^{\infty} r(p)\cos(2\pi pf) \tag{2.48}$$

If the data in the sequence $x(n)$ are independent, then $r(p)$ reduces to $r(0)$ and the spectrum $S(f)$ is flat; the signal is then said to be white.

An important aspect of the Gaussian probability laws is that they preserve their character under any linear operation, such as convolution, filtering, differentiation, or integration.

Therefore, if a Gaussian signal is fed to a linear system, the output is also Gaussian. Moreover, there is a natural trend toward Gaussian probability densities, because of the so-called central limit theorem, which states that the random variable

$$x = \frac{1}{\sqrt{N}}\sum_{i=1}^{N} x_i \tag{2.49}$$

where the x_i are N independent identically distributed (i.i.d.) second-order random variables, becomes Gaussian when N grows to infinity.

The Gaussian approximation can reasonably be made as soon as N exceeds a few units, and the importance of Gaussian densities becomes apparent because in nature many signal sources and, particularly, noise sources at the micro- or macroscopic levels add up to make the sequence to be processed. So Gaussian noise is present in virtually every signal processing application.

2.5. SYNTHETIC, MOVING AVERAGE, AND AUTOREGRESSIVE SIGNALS

In simulation, evaluation, transmission, test, and measurement, the data sequences used are often not natural but synthetic signals. They appear also in some analysis techniques, namely analysis by synthesis techniques.

Deterministic signals can be generated in a straightforward manner as isolated or recurring pulses or as sums of sinusoids. A diagram to produce a single sinusoid is shown in Figure 2.3. Note that the sinusoids in a sum must have different phases; otherwise an impulse shape waveform is obtained.

Flat spectrum signals are characterized by the fact that their energy is uniformly distributed over the entire frequency band. Therefore an approach to produce a deterministic white-noise-like waveform is to generate a set of sinusoids uniformly distributed in frequency with the same amplitude but different phases.

Random signals can be obtained from sequences of statistically independent real numbers generated by standard computer subroutines through a rounding process. The magnitudes of these numbers are uniformly distributed in the interval (0, 1), and the sequences obtained have a flat spectrum.

Several probability densities can be derived from the uniform distribution. Let the Gaussian, Rayleigh, and uniform densities be $p(x)$, $p(y)$, and $p(z)$, respectively. The Rayleigh density is

$$p(y) = \frac{y}{\sigma^2} \exp\left[-\frac{y^2}{2\sigma^2}\right] \tag{2.50}$$

and the second-order moment of the corresponding random variable is $2\sigma^2$, the mean is $\sigma\sqrt{\pi/2}$, and the variance is $(2 - \pi/2)\sigma^2$. It is a density associated with the peak values of a narrowband Gaussian signal. The changes of variables

$$p(z)\,dz = dz = p(y)\,dy$$

leads to

$$\frac{dz}{dy} = \frac{y}{\sigma^2} \exp\left[-\frac{y^2}{2\sigma^2}\right]$$

Hence,

$$z = \exp\left[-\frac{y^2}{2\sigma^2}\right]$$

and a Rayleigh sequence $y(n)$ is obtained from a uniform sequence $z(n)$ in the magnitude interval (0, 1) by the following operation:

$$y(n) = \sigma\sqrt{2\ln[1/z(n)]} \tag{2.51}$$

Now, independent Rayleigh and uniform sequences can be used to derive a Gaussian sequence $x(n)$:

$$x(n) = y(n)\cos[2\pi z(n)] \tag{2.52}$$

In the derivation, a companion variable is introduced:

$$x'(n) = y(n)\sin 2\pi z(n) \tag{2.53}$$

Now, let us consider the joint probability $p(x, x')$ and apply the relation between rectangular and polar coordinates:

$$p(x, x')\,dx\,dx' = p(x, x')y\,dy\,dz = p(y)p(z)\,dy\,dz \tag{2.54}$$

Then

$$p(x, x') = \frac{1}{2\pi y} p(y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+x'^2)/2\sigma^2} = p(x)p(x') \tag{2.55}$$

and finally

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/2\sigma^2} \tag{2.56}$$

The two variables $x(n)$ and $x'(n)$ have the same distribution and, considered jointly, they make a complex Gaussian noise of power $2\sigma^2$. The above derivation shows that this complex noise can be represented in terms of its modulus, which has a Rayleigh distribution, and its phase, which has a uniform distribution.

Correlated random signals can be obtained by filtering a white sequence with either uniform or Gaussian amplitude probability density, as shown in Figure 2.5. The filter $H(z)$ can take on different structures, corresponding to different models for the output signal [6].

The simplest type is the finite impulse response (FIR) filter, corresponding to the so-called moving average (MA) model and defined by

$$H(z) = \sum_{i=0}^{N} h_i z^{-i} \tag{2.57}$$

and, in the time domain,

$$x(n) = \sum_{i=0}^{N} h_i e(n-i) \tag{2.58}$$

where the h_i are the filter impulse response.

The output signal ACF is obtained by direct application of definition (2.34), considering that

$$E[e^2(n)] = \sigma_e^2, \quad E[e(n)e(n-i)] = 0 \quad \text{for } i \neq 0$$

The result is

$$r(p) = \begin{cases} \sigma_e^2 \sum\limits_{i=0}^{N-p} h_i h_{i+p}, & |p| \leqslant N \\ 0, & |p| > N \end{cases} \tag{2.59}$$

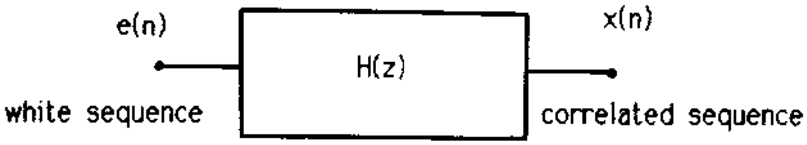


FIG. 2.5 Generation of a correlated random signal.

Several remarks are necessary. First, the ACF has a finite length in accordance with the filter impulse response. Second, the output signal power σ_x^2 is related to the input signal power by

$$\sigma_x^2 = r(0) = \sigma_e^2 \sum_{i=0}^{N} h_i^2 \tag{2.60}$$

Equation (2.60) is frequently used in subsequent sections. The power spectrum can be computed from the ACF $r(p)$ by using equation (2.48), but another approach is to use $H(z)$, since it is available, via the equation

$$S(f) = \sigma_e^2 \left| \sum_{i=0}^{N} h_i e^{j2\pi if} \right|^2 \tag{2.61}$$

An infinite impulse response (IIR) filter corresponds to an autoregressive (AR) model. The equations are

$$H(z) = \frac{1}{1 - \sum_{i=1}^{N} a_i z^{-i}} \tag{2.62}$$

and, in the time domain,

$$x(n) = e(n) + \sum_{i=1}^{N} a_i x(n-i) \tag{2.63}$$

The ACF can be derived from the corresponding filter impulse response coefficients h_i:

$$H(z) = \sum_{i=0}^{\infty} h_i z^{-i} \tag{2.64}$$

and, accordingly, it is an infinite sequence:

$$r(p) = \sigma_e^2 \sum_{i=0}^{\infty} h_i h_{i+p} \tag{2.65}$$

The power spectrum is

$$S(f) = \frac{\sigma_e^2}{\left| 1 - \sum_{i=1}^{N} a_i e^{-j2\pi if} \right|^2} \tag{2.66}$$

An example is shown in Figure 2.6 for the filter transfer function:

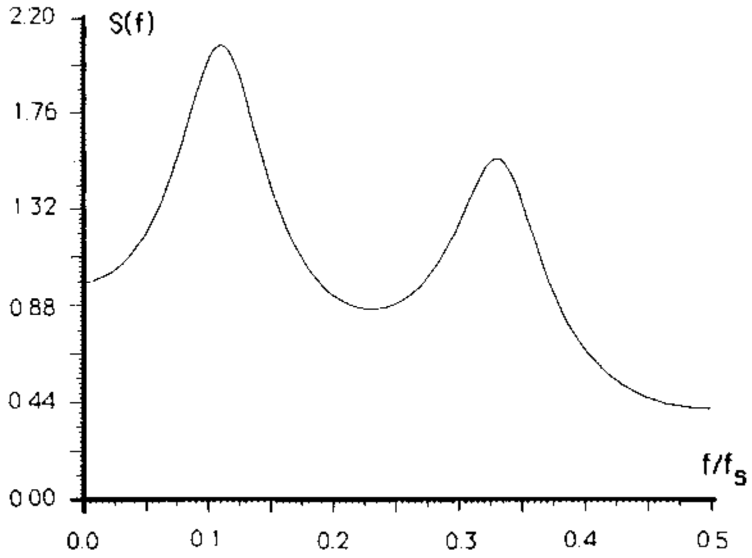


FIG. 2.6 Spectrum of an AR signal.

$$H(z) = \frac{1}{(1 + 0.80z^{-1} + 0.64z^{-2})(1 - 1.23z^{-1} + 0.64z^{-2})}$$

Since the spectrum of a real signal is symmetric about the zero frequency, only the band $[0, f_{s/2}]$, where f_s is the sampling frequency, is represented.

For MA signals, the direct relation (2.59) has been derived between the ACF and filter coefficients. A direct relation can also be obtained here by multiplying both sides of the recursion definition (2.63) by $x(n-p)$ and taking the expectation, which leads to

$$r(0) = \sigma_e^2 + \sum_{i=1}^{N} a_i r(i) \tag{2.67}$$

$$r(p) = \sum_{i=1}^{N} a_i r(p-i), \quad p \geqslant 1 \tag{2.68}$$

For $p \geqslant N$, the sequence $r(p)$ is generated recursively from the N preceding terms. For $0 \leqslant p \leqslant N-1$, the above equations establish a linear dependence between the two sets of filter coefficients and the first ACF values.

They can be expressed in matrix form to derive the coefficients from the ACF terms:

$$\begin{bmatrix} r(0) & r(1) & \cdots & r(N) \\ r(1) & r(1) & \cdots & r(N-1) \\ \vdots & \vdots & \ddots & \vdots \\ r(N) & r(N-1) & \cdots & r(0) \end{bmatrix} \begin{bmatrix} 1 \\ -a_1 \\ \vdots \\ -a_N \end{bmatrix} = \begin{bmatrix} \sigma_e^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.69}$$

Equation (2.69) is a normal equation, called the order N forward linear prediction equation, studied in a later chapter.

To complete the AR signal analysis, note that the generating filter impulse response is

$$h_p = r(p) - \sum_{i=1}^{N} a_i r(p+i) \tag{2.70}$$

This equation is a direct consequence of definition relations (2.63) and (2.64), if we notice that

$$h_p = E[x(n)e(n-p)] \tag{2.71}$$

Since $r(p) = r(-p)$, equation (2.68) shows that the impulse response h_p is zero for negative p, which reflects the filter causality.

It is also possible to relate the AC function of an AR signal to the poles of the generating filter.

For complex poles, the filter z-transfer function can be expressed in factorized form:

$$H(z) = \frac{1}{\prod\limits_{i=1}^{N/2} (1 - P_i z^{-1})(1 - \bar{P}_i z^{-1})} \tag{2.72}$$

Using the equality

$$S(f) = \sigma_e^2 |H(z)H(z^{-1})|_{|z|=1} = \left| \sum_{p=-\infty}^{\infty} r(p) z^{-p} \right|_{|z|=1} \tag{2.73}$$

the series development of the product $H(z)H(z^{-1})$ leads to the AC function of the AR signal. The rational function decomposition of $H(z)H(z^{-1})$ yields, after simplification,

$$r(p) = \sum_{i=1}^{N/q} \alpha_i |P_i|^n \cos[n \operatorname{Arg}(p_i) + \beta_i] \tag{2.74}$$

where the real parameters α_i and β_i are the parameters of the decomposition and hence are related to the poles P_i.

It is worth pointing out that the same expression is obtained for the generating filter of the type FIR/IIR, but then the parameters α_i and β_i are no longer related to the poles: they are independent.

A limitation of AR spectra is that they do not take on zero values, whereas MA spectra do. So it may be useful to combine both [7].

2.6. ARMA SIGNALS

An ARMA signal is obtained through a filter with a rational z-transfer function:

$$H(z) = \frac{\sum_{i=0}^{N} b_i z^{-1}}{1 - \sum_{i=1}^{N} a_i z^{-1}} \tag{2.75}$$

In the time domain,

$$x(n) = \sum_{i=0}^{N} b_i e(n-i) + \sum_{i=1}^{N} a_i x(n-i) \tag{2.76}$$

The denominator and numerator polynomials of $H(z)$ can always be assumed to have the same order; if necessary, zero coefficients can be added.

The power spectral density is

$$S(f) = \sigma_e^2 \frac{\left| \sum_{i=0}^{N} b_i e^{-j2\pi if} \right|^2}{\left| 1 - \sum_{i=1}^{N} a_i e^{-j2\pi if} \right|^2} \tag{2.77}$$

A direct relation between the ACF and the coefficients is obtained by multiplying both sides of the time recursion (2.76) by $x(n-p)$ and taking the expectation:

$$r(p) = \sum_{i=1}^{N} a_i r(p-i) + \sum_{i=0}^{N} b_i E(e(n-i)x(n-p)] \tag{2.78}$$

Now the relationships between ACF and filter coefficients become nonlinear, due to the second term in (2.78). However, that nonlinear term vanishes for $p > N$ because $x(n-p)$ is related to the input signal value with the same index and the preceding values only, not future ones. Hence, a matrix equation can again be derived involving the AR coefficients of the ARMA signal:

$$\begin{bmatrix} r(N) & r(N-1) & \cdots & r(0) \\ r(N+1) & r(N) & \cdots & r(1) \\ \vdots & \vdots & \ddots & \vdots \\ r(2N) & r(2N-1) & \cdots & r(N) \end{bmatrix} \begin{bmatrix} 1 \\ -a_1 \\ \vdots \\ -a_N \end{bmatrix} = b_0 b_N \begin{bmatrix} \sigma_e^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.79}$$

For $p > N$, the sequence $r(p)$ is again generated recursively from the N preceding terms.

The relationship between the first $(N+1)$ ACF terms and the filter coefficients can be established through the filter impulse response, whose coefficients h_i satisfy, by definition,

$$x(n) = \sum_{i=0}^{\infty} h_i e(n-i) \tag{2.80}$$

Now replacing $x(n-i)$ in (2.76) gives

$$x(n) = \sum_{i=0}^{N} b_i e(n-i) + \sum_{i=1}^{N} a_i \sum_{j=0}^{\infty} h_j e(n-i-j)$$

and

$$x(n) = \sum_{i=0}^{N} b_i e(n-i) + \sum_{k=1}^{\infty} e(n-k) \sum_{i=1}^{N} a_i h_{k-i} \tag{2.81}$$

Clearly, the impulse response coefficients can be computed recursively:

$$\begin{aligned} &h_0 = b_0, \quad h_k = 0 \quad \text{for } k < 0 \\ &h_k = b_k + \sum_{i=1}^{N} a_i h_{k-i}, \quad k \geqslant 1 \end{aligned} \tag{2.82}$$

In matrix form, for the $N+1$ first terms we have

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -a_1 & 1 & 0 & \cdots & 0 \\ -a_2 & -a_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_N & -a_{N-1} & -a_{N-2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} h_0 & 0 & 0 & \cdots & 0 \\ h_1 & h_0 & 0 & \cdots & 0 \\ h_2 & h_1 & h_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_N & h_{N-1} & h_{N-2} & \cdots & h_0 \end{bmatrix}$$

$$= \begin{bmatrix} b_0 & 0 & 0 & \cdots & 0 \\ b_1 & b_0 & 0 & \cdots & 0 \\ b_2 & b_1 & b_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_N & b_{N-1} & b_{N-2} & \cdots & b_0 \end{bmatrix} \tag{2.83}$$

Coming back to the ACF and (2.78), we have

$$\sum_{i=0}^{N} b_i E[e(n-i)x(n-p)] = \sigma_e^2 \sum_{i=0}^{N} b_i h_{i-p}$$

and, after simple manipulations,

$$r(p) = \sum_{i=1}^{N} a_i r(p-i) + \sigma_e^2 \sum_{j=0}^{N-p} b_{j+p} h_j \tag{2.84}$$

Now, introducing the variable

$$d(p) = \sum_{j=0}^{N-p} b_{j+p} h_j \tag{2.85}$$

we obtain the matrix equation

$$\mathscr{A} \begin{bmatrix} r(0) \\ r(1) \\ \vdots \\ r(N) \end{bmatrix} + \mathscr{A}' \begin{bmatrix} r(0) \\ r(-1) \\ \vdots \\ r(-N) \end{bmatrix} = \sigma_e^2 \begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(N) \end{bmatrix} \tag{2.86}$$

where

$$\mathscr{A} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -a_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -a_N & -a_{N-1} & \cdots & 1 \end{bmatrix}$$

$$\mathscr{A}' = \begin{bmatrix} 0 & -a_1 & \cdots & -a_N \\ 0 & -a_2 & \cdots & 0 \\ \vdots & \vdots & ⋰ & \vdots \\ 0 & -a_N & & \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

For real signals, the first $(N+1)$ ACF terms are obtained from the equation

$$\begin{bmatrix} r(0) \\ r(1) \\ \vdots \\ r(N) \end{bmatrix} = \sigma_e^2 [\mathscr{A} + \mathscr{A}']^{-1} \begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(N) \end{bmatrix} \tag{2.87}$$

In summary, the procedure to calculate the ACF of an ARMA signal from the generating filter coefficients is as follows:

1. Compute the first $(N+1)$ terms of the filter impulse response through recursion (2.82).
2. Compute the auxiliary variables $d(p)$ for $0 \leqslant p \leqslant N$.
3. Compute the first $(N+1)$ ACF terms from matrix equation (2.87).
4. Use recursion (2.68) to derive $r(p)$ when $p \geqslant N+1$.

Obviously, finding the ACF is not a simple task, particularly for large filter orders N. Conversely, the filter coefficients and input noise power can be retrieved from the ACF. First the AR coefficients a_i and the scalar $b_0 b_N \sigma_e^2$ can be obtained from matrix equation (2.79). Next, from the time domain definition (2.76), the following auxiliary signal can be introduced:

$$u(n) = x(n) - \sum_{i=1}^{N} a_i x(n-i) = e(n) + \sum_{i=1}^{N} b_i e(n-i) \tag{2.88}$$

where $b_0 = 1$ is assumed.

The ACF $r_u(p)$ of the auxiliary signal $u(n)$ is derived from the ACF of $x(n)$ by the equation

$$\begin{aligned} r_u(p) &= E[u(n)u(n-p)] \\ &= r(p) - \sum_{i=1}^{N} a_i r(p+i) - \sum_{i=1}^{N} a_i r(p-i) + \sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j r(p+j-i) \end{aligned}$$

or, more concisely by

$$r_u(p) = \sum_{i=-N}^{N} c_i r(p-i) \tag{2.89}$$

where

$$\begin{aligned} c_i &= c_{-i}, \quad c_0 = 1 + \sum_{j=1}^{N} a_j^2 \\ c_i &= -a_i + \sum_{j=i+1}^{N} a_j a_{j-i} \end{aligned} \tag{2.90}$$

But $r_u(p)$ can also be expressed in terms of MA coefficients, because of the second equation in (2.88). The corresponding expressions, already given in the previous section, are

$$r_u(p) = \begin{cases} \sigma_e^2 \sum\limits_{i=0}^{N-p} b_i b_{i+p}, & |p| \leqslant N \\ 0, & |p| > N \end{cases}$$

From these $N+1$ equations, the input noise power σ_e^2 and the MA coefficients $b_i(1 \leqslant i \leqslant N; b_0 = 1)$ can be derived from iterative Newton–Raphson algorithms. It can be verified that $b_0 b_N \sigma_e^2$ equals the value we previously found when solving matrix equation (2.79) for AR coefficients.

The spectral density $S(f)$ can be computed with the help of the auxiliary signal $u(n)$ by considering the filtering operation

$$x(n) = u(n) + \sum_{i=1}^{N} a_i x(n-i) \tag{2.91}$$

which, in the spectral domain, corresponds to

$$S(f) = \frac{r_u(0) + 2\sum\limits_{p=1}^{N} r_u(p)\cos(2\pi pf)}{\left|1 - \sum\limits_{i=1}^{N} a_i e^{-j2\pi if}\right|^2} \tag{2.92}$$

This expression is useful in spectral analysis.

Until now, only real signals have been considered in this section. Similar results can be obtained with complex signals by making appropriate complex conjugations in equations. An important difference is that the ACF is no longer symmetrical, which can complicate some procedures. For example, the matrix equation (2.86) to obtain the first $(N+1)$ ACF terms becomes

$$\mathscr{A} r + \mathscr{A}' \bar{r} = \sigma_e^2 d \tag{2.93}$$

where r is the correlation vector, $\bar{r}$ the vector with complex conjugate entries, and d the auxiliary variable vector. The conjugate expression of (2.86) is

$$\bar{\mathscr{A}} \bar{r} + \bar{\mathscr{A}}' r = \sigma_e^2 \bar{d} \tag{2.94}$$

The above equations, after some algebraic manipulations, lead to

$$[\mathscr{A} - \mathscr{A}'(\bar{\mathscr{A}})^{-1} \bar{\mathscr{A}}'] r = \sigma_e^2 [d - \mathscr{A}'(\bar{\mathscr{A}})^{-1} \bar{d}] \tag{2.95}$$

Now two matrix inversions are needed to get the correlation vector. Note that $\mathscr{A}^{-1}$ is readily obtained from (2.83) by calculating the first $N+1$ values of the impulse response of the AR filter through the recursion (2.82).

Next, more general signals of the types often encountered in control systems are introduced.

2.7. MARKOV SIGNALS

Markov signals are produced by state variable systems whose evolution from time n to time $n+1$ is governed by a constant transition matrix [8].

The state of a system of order N at time n is defined by a set of N internal variables represented by a vector $X(n)$ called the state vector. The block diagram of a typical system is shown in Figure 2.7, and the equations are

$$\begin{aligned} X(n+1) &= AX(n) + Bw(n) \\ y(n) &= C^t X(n) + v(n) \end{aligned} \tag{2.96}$$

The matrix A is the $N \times N$ transition matrix, B is the control vector, and C is the observation vector [9]. The input sequence is $w(n)$; $v(n)$ can be a measurement noise contaminating the output $y(n)$.

The state of the system at time n is obtained from the initial state at time zero by the equation

$$X(n) = A^n X(0) + \sum_{i=1}^{n} A^{n-i} Bw(i-1) \tag{2.97}$$

Consequently, the behavior of such a system depends on successive powers of the transition matrix A.

The z-transfer function of the system $H(z)$, obtained by taking the z-transform of the state equations, is

$$H(z) = C^t (ZI_N - A)^{-1} B \tag{2.98}$$

with I_N the $N \times N$ unity matrix.

The poles of the transfer function are the values of z for which the determinant of the matrix $(ZI_N - A)$ is zero. That is also the definition of the eigenvalues of A.

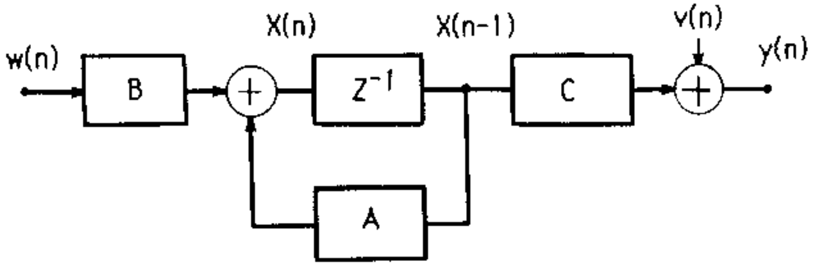


FIG. 2.7 State variable system.

The system is stable if and only if the poles are inside the unit circle in the complex plane or, equivalently, if and only if the absolute values of the eigenvalues are less than unity, which can be seen directly from equation (2.97).

Let us assume that $w(n)$ is centered white noise with power σ_w^2. The state variables are also centered, and their covariance matrix can be calculated. Multiplying state equation (2.96) on the right by its transpose yields

$$\begin{aligned} X(n+1)X^t(n+1) = {} & AX(n)X^t(n)A + Bw^2(n)B^t \\ & + AX(n)w(n)B^t + Bw(n)X^t(n)A^t \end{aligned}$$

The expected values of the last two terms of this expression are zero, because $x(n)$ depends only on the past input values. Hence, the covariance matrix $R_{xx}(n+1)$ is

$$R_{xx}(n+1) = E[X(n+1)X^t(n+1)] = AR_{xx}(n)A^t + \sigma_w^2 BB^t \tag{2.99}$$

It can be computed recursively once the covariance of the initial conditions $R_{xx}(0)$ is known. If the elements of the $w(n)$ sequence are Gaussian random variables, the state variables themselves are Gaussian, since they are linear combinations of past input values.

The Markovian representation applies to ARMA signals. Several sets of state variables can be envisaged. For example, in linear prediction, a representation corresponding to the following state equations is used:

$$\begin{aligned} x(n) &= C^t\hat{X}(n) + e(n) \\ \hat{X}(n) &= A\hat{X}(n-1) + Be(n-1) \end{aligned} \tag{2.100}$$

with

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & 1 \\ a_N & a_{N-1} & a_{N-2} & \cdots & a_1 \end{bmatrix}, \quad B = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_N \end{bmatrix}$$

$$C = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \hat{X}(n) = \begin{bmatrix} \hat{x}_0(n) \\ \hat{x}_1(n) \\ \vdots \\ \hat{x}_{N-1}(n) \end{bmatrix}$$

The elements of vector B are the filter impulse response coefficients of equation (2.80), and those of the state vector, $\hat{x}_i(n)$ are the i-step linear predictions of $x(n)$, defined, for the ARMA signal and as shown later, by

$$\hat{x}_i(n) = \sum_{k=1}^{i} a_k \hat{x}(n-k) + \sum_{j=1}^{N-i} a_{i+j} x(n-i-j) + \sum_{j=1}^{N} b_{i+j} e(n-i-j) \tag{2.101}$$

It can be verified that the characteristic polynomial of the matrix A, whose roots are the eigenvalues, is the denominator of the filter transfer function $H(z)$ in (2.75).

Having presented methods for generating signals, we now turn to analysis techniques. First we introduce some important definitions and concepts [10].

2.8. LINEAR PREDICTION AND INTERPOLATION

The operation which produces a sequence $e(n)$ from a data sequence $x(n)$, assumed centered and wide-sense stationary, by the convolution

$$e(n) = x(n) - \sum_{i=1}^{\infty} a_i x(n-i) \tag{2.102}$$

is called one-step linear prediction error filtering, if the coefficients are calculated to minimize the variance of the output $e(n)$. The minimization is equivalent, through derivation, to making $e(n)$ orthogonal to all previous data, because it leads to:

$$E[e(n)x(n-i)] = 0, \quad i \geqslant 1 \tag{2.103}$$

Since $e(n)$ is a linear combination of past data, the following equations are also valid:

$$E[e(n)e(n-i)] = 0, \quad i \geqslant 1 \tag{2.104}$$

and the sequence $e(n)$, called the prediction error or the innovation, is a white noise. Therefore the one-step prediction error filter is also called the whitening filter. The data $x(n)$ can be obtained from the innovations by the inverse filter, assumed realizable, which is called the model or innovation filter. The operations are shown in Figure 2.8.

The prediction error variance $E_a = E[e^2(n)]$ can be calculated from the data power spectrum density $S(e^{j\omega})$ by the conventional expressions for digital filtering:

$$E_a = \frac{1}{2\pi} \int_{-\pi}^{\pi} |A(e^{j\omega})|^2 S(e^{j\omega})\, d\omega \tag{2.105}$$

or, in terms of z-transforms,

$$E_a = \frac{1}{j2\pi} \int_{|z|=1} A(z)A(z^{-1})S(z)\frac{dz}{z} \tag{2.106}$$

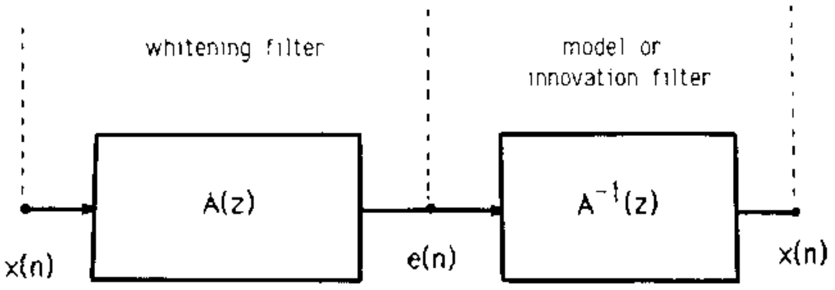


FIG. 2.8 Linear prediction filter and inverse filter.

where $A(z)$ is the transfer function of the prediction error filter. The prediction filter coefficients depend only on the input signal, and the error power can be expressed as a function of $S(e^{j\omega})$ only. To derive that expression, we must first show that the prediction error filter is minimum phase; in other words, all its zeros are inside or on the unit circle in the complex z-plane.

Let us assume that a zero of $A(z)$, say z_0, is outside the unit circle, which means $|z_0| > 1$, and consider the filter $A'(z)$ given by

$$A'(z) = A(z)\frac{z - \bar{z}_0^{-1}}{z - z_0}\frac{z - z_0^{-1}}{z - \bar{z}_0} \tag{2.107}$$

As Figure 2.9 shows,

$$\left|\frac{z - \bar{z}_0^{-1}}{z - z_0}\right|_{z=e^{j\omega}} \left|\frac{z - z_0^{-1}}{z - \bar{z}_0}\right|_{z-e^{j\omega}} = \frac{1}{|z_0|^2} \tag{2.108}$$

and the corresponding error variance is

$$E'_a = \frac{1}{|z_0|^2} E_a < E_a \tag{2.109}$$

which contradicts the definition of the prediction filter. Consequently, the prediction filter $A(z)$ is minimum phase.

In (2.106) for E_a, we can remove the filter transfer function with the help of logarithms, taking into account that the innnovation sequence has a constant power spectrum density; thus,

$$2\pi j \ln E_a = \int_{|z|=1} \ln A(z)\frac{dz}{z} + \int_{|z|=1} \ln A(z^{-1})\frac{dz}{z} + \int_{|z|=1} \ln S(z)\frac{dz}{z} \tag{2.110}$$

Now, since $A(z)$ is minimum phase, $\ln A(z)$ is analytic for $|z| \geqslant 1$ and the unit circle can be replaced in the above integral with a circle whose radius is arbitrarily large, and since

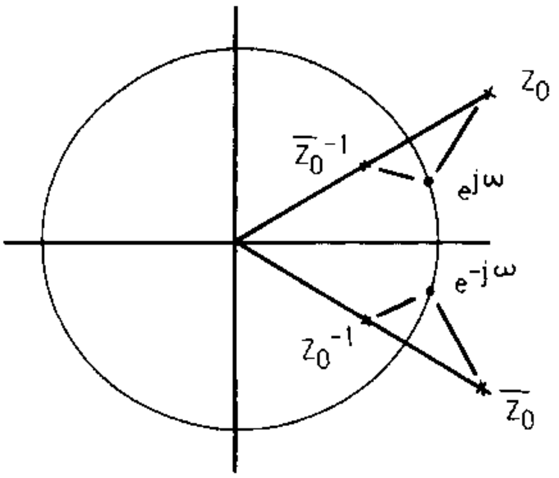


FIG. 2.9 Reflection of external zero in the unit circle.

$$\lim_{z\to\infty} A(z) = a_0 = 1$$

the first integral vanishes on the right side of (2.110). The second integral also vanishes because it can be shown, by a change of variables from z^{-1} to z that it is equal to the first one.

Finally, the prediction error power is expressed in terms of the signal power spectrum density by

$$E_a = \exp\left\{\frac{1}{2\pi}\int_{-\pi}^{\pi} \ln S(e^{j\omega})\, d\omega\right\} \tag{2.111}$$

This very important result is known as the Kolmogoroff–Szegö formula.

A useful signal parameter is the prediction gain G, defined as the signal-to-prediction-error ratio:

$$G = \frac{1}{2\pi}\int_{-\pi}^{\pi} S(e^{j\omega})\, d\omega \Big/ \exp\left\{\frac{1}{2\pi}\int_{-\pi}^{\pi} \ln S(e^{j\omega})\, d\omega\right\} \tag{2.112}$$

Clearly, for a white noise $G = 1$.

At this stage, it is interesting to compare linear prediction and interpolation. Interpolation is the filtering operation which produces from the data $x(n)$ the sequence

$$e_i(n) = \sum_{j=-\infty}^{\infty} h_j x(n-j), \quad h_0 = 1 \tag{2.113}$$

with coefficients calculated to minimize the output power. Hence, $e_i(n)$ is orthogonal to past and future data:

$$E[e_i(n)x(n-k)] = E_i\delta(k) \tag{2.114}$$

where $\delta(k)$ is the Dirac distribution and

$$E_i = E[e_i^2(n)] \tag{2.115}$$

Clearly, the interpolation error $e_i(n)$ is not necessarily a white noise. Taking the z-transform of both sides of the orthogonal relationship (2.114) leads to

$$H(z)S(z) = E_i \tag{2.116}$$

Also

$$E_i = \frac{1}{j2\pi}\int_{|z|=1} H(z)H(z^{-1})S(z)\frac{dz}{z} \tag{2.117}$$

Combining equations (2.116) and (2.117) gives

$$E_i = 1\Big/\frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{d\omega}{S(e^{j\omega})} \tag{2.118}$$

Now, it is known from linear prediction that

$$S(e^{j\omega}) = \frac{E_a}{|A(e^{j\omega})|^2} \tag{2.119}$$

and

$$E_i = E_a\Big/\frac{1}{2\pi}\int_{-\pi}^{\pi}|A(e^{j\omega})|^2 d\omega = E_a\Big/\sum_{i=0}^{\infty}a_i^2 \tag{2.120}$$

Since $a_0 = 1$, we can conclude that $E_i \leqslant E_a$; the interpolation error power is less than or equal to the prediction error power, which is a not unexpected result.

Linear prediction is useful for classifying signals and, particular, distinguishing between deterministic and random processes.

2.9. PREDICTABLE SIGNALS

A signal $x(n)$ is predictable if and only if its prediction error power is null:

$$E_a = \frac{1}{2\pi}\int_{-\pi}^{\pi}|A(e^{j\omega})|^2 S(e^{j\omega})\,d\omega = 0 \tag{2.121}$$

or, in the time domain,

$$x(n) = \sum_{i=1}^{\infty} a_i x(n-i) \tag{2.122}$$

which means that the present value $x(n)$ of the signal can be expressed in terms of its past values. The only signals which satisfy the above equations are those whose spectrum consists of lines:

$$S(e^{j\omega}) = \sum_{i=1}^{N} |S_i|^2 \delta(\omega - \omega_i) \tag{2.123}$$

The scalars $|S_i|^2$ are the powers of individual lines. The integer N can be arbitrarily large. The minimum degree prediction filter is

$$A_m(z) = \prod_{i=1}^{N} (1 - e^{j\omega_i} z^{-1}) \tag{2.124}$$

However all the filters $A(z)$ with

$$A(z) = 1 - \sum_{i=1}^{\infty} a_i z^{-1} \tag{2.125}$$

and such that $A(e^{j\omega_i}) = 0$ for $1 \leqslant i \leqslant N$ satisfy the definition and are prediction filters.

Conversely, since $A(z)$ is a power series, $A(e^{j\omega})$ cannot equal zero for every ω in an interval, and equations (2.121) and (2.122) can hold only if $S(e^{j\omega}) = 0$ everywhere except at a countable set of points. It follows that $S(e^{j\omega})$ must be a sum of impulses as in (2.123), and $A(z)$ has corresponding zeros on the unit circle.

Finally, a signal $x(n)$ is predictable if and only if its spectrum consists of lines.

The line spectrum signals are an extreme case of the more general class of bandlimited signals. A signal $x(n)$ is said to be bandlimited if $S(e^{j\omega}) = 0$ in one or more frequency intervals. Then a filter $H(\omega)$ exists such that

$$H(\omega) S(e^{j\omega}) \equiv 0 \tag{2.126}$$

and, in the time domain,

$$\sum_{i=-\infty}^{\infty} h_i x(n-i) = 0$$

With proper scaling, we have

$$x(n) = -\sum_{i=1}^{\infty} h_i x(n-i) - \sum_{i=1}^{\infty} h_{-i} x(n+i) \tag{2.127}$$

Thus the present value can be expressed in terms of past and future values. Again the representation is not unique, because the function $H(\omega)$ is arbitrary, subject only to condition (2.126). It can be shown that a band-limited signal can be approximated arbitrarily closely by a sum involving only its past values. Equality is obtained if $S(e^{j\omega})$ consists of lines only.

The above sections are mainly intended to serve as a gradual preparation for the introduction of one of the most important results in signal analysis, the fundamental decomposition.

2.10. THE FUNDAMENTAL (WOLD) DECOMPOSITION

Any signal is the sum of two orthogonal components, an AR signal and a predictable signal. More specifically:

Decomposition Theorem

An arbitrary unpredictable signal $x(n)$ can be written as a sum of two orthogonal signals:

$$x(n) = x_p(n) + x_r(n) \tag{2.128}$$

where $x_p(n)$ is predictable and $x_r(n)$ is such that its spectrum $S_r(E^{j\omega})$ can be factored as

$$S_r(e^{j\omega}) = |H(e^{j\omega})|^2, \quad H(z) = \sum_{i=0}^{\infty} h_i z^{-i} \tag{2.129}$$

and $H(z)$ is a function analytic for $|z| > 1$.

The component $x_r(n)$ is sometimes said to be regular. Following the development in [10], the proof of the theorem begins with the computation of the prediction error sequence

$$e(n) = x(n) - \sum_{i=1}^{\infty} a_i x(n-i) \tag{2.130}$$

As previously mentioned, the prediction coefficients are computed so as to make $e(n)$ orthogonal to all past data values, and the error sequence is a white noise with variance E_a.

Conversely, the least squares estimate of $x(n)$ in terms of the sequence e (n) and its past is the sum

$$x_r(n) = \sum_{i=0}^{\infty} h_i e(n-i) \tag{2.131}$$

and the corresponding error signal

$$x_p(n) = x(n) - x_r(n)$$

is orthogonal to $e(n-i)$ for $i \geqslant 0$. In other words, $e(n)$ is orthogonal to $x_p(n+k)$ for $k \geqslant 0$.

Now, $e(n)$ is also orthogonal to $x_r(n-k)$ for $k \geqslant 1$, because $x_r(n-k)$ depends linearly on $e(n-k)$ and its past and $e(n)$ is white noise. Hence,

$$E[e(n)[x(n-k) - x_r(n-k)]] = 0 = E[e(n)x_p(n-k)], \quad k \geqslant 1$$

and

$$E[e(n)x_p(n-k)] = 0, \quad \text{all } k \tag{2.132}$$

Expression (2.131) yields

$$E[x_r(n)x_p(n-k) = 0, \quad \text{all } k \tag{2.133}$$

The signals $x_r(n)$ and $x_p(n)$ are orthogonal, and their powers add up to give the input signal power:

$$E[x^2(n)] = E[x_p^2(n)] + E[x_r^2(n)] \tag{2.134}$$

Now (2.131) also yields

$$E[x_r^2(n)] = E_a \sum_{i=0}^{\infty} h_i^2 \leqslant E[x^2(n)] \tag{2.135}$$

Therefore,

$$H(z) = \sum_{i=0}^{\infty} h_i z^{-i}$$

converges for $|z| > 1$ and defines a linear causal system which produces $x_r(n)$ when fed with $e(n)$.

In these conditions, the power spectrum of $x_r(n)$ is

$$S_r(e^{j\omega}) = E_a |H(e^{j\omega})|^2 \tag{2.136}$$

The filtering operations which have produced $x_r(n)$ from $x(n)$ are shown in Figure 2.10. If instead of $x(n)$ the component in a signal sequence $x(n) - x_r(n) = x_p(n)$ is fed to the system, the error $e_p(n)$, instead of $e(n)$, is obtained. The sequence

$$e_p(n) = e(n) - \left[x_r(n) - \sum_{i=1}^{\infty} a_i x_r(n-i) \right] \tag{2.137}$$

is a linear combination of $e(n)$ and its past, via equation (2.131). But, by definition,

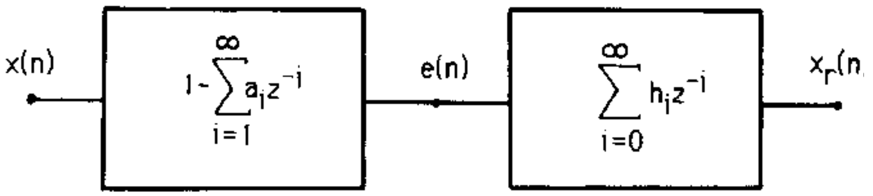

x(n)
$1-\sum_{i=1}^{\infty} a_i z^{-i}$
e(n)
$\sum_{i=0}^{\infty} h_i z^{-i}$
$x_r(n$

FIG. 2.10 Extraction of the regular component in a signal.

$$e_p(n) = x_p(n) - \sum_{i=1}^{\infty} a_i x_p(n-i) \tag{2.138}$$

which, using equations (2.132) and (2.133), yields

$$\begin{aligned}E[e_p^2(n)] &= E\left\{\left[e(n) - \left(x_r(n) - \sum_{i=1}^{\infty} a_i x_r(n-i)\right)\right]\right.\\ &\qquad \left.\times \left[x_p(n) - \sum_{i=1}^{\infty} a_i x_p(n-i)\right]\right\}\\ &= 0\end{aligned}$$

Therefore $x_p(n)$ is a predictable signal and the whitening filter $A(z)$ is a prediction error filter, although not necessarily the minimum degree filter, which is given by (2.124). On the contrary, $A(z)$ is the unique prediction error filter of $x(n)$.

Finally, the spectrum $S(e^{j\omega})$ of the unpredictable signal $x(n)$ is a sum

$$S(e^{j\omega}) = S_r(e^{j\omega}) + S_p(e^{j\omega}) \tag{2.139}$$

where $S_r(e^{j\omega})$ is the continuous spectrum of the regular signal $x_r(n)$, and $S_p(e^{j\omega})$ is the line spectrum of the deterministic component, the two components being uncorrelated.

2.11. HARMONIC DECOMPOSITION

The fundamental decomposition is used in signal analysis as a reference for selecting a strategy [11]. As an illustration let us consider the case, frequently occurring in practice, where the signal to be analyzed is given as a set of 2 $N+1$ autocorrelation coefficients $r(p)$ with $-N \leqslant p \leqslant N$, available from a measuring procedure. To perform the analysis, we have two extreme hypotheses. The first one consists of assuming that the signal has no deterministic component; then a set of N prediction coefficients can be calculated as indicated in the section dealing with AR signals by (2.69), and the power spectrum is obtained from (2.66).

But another hypothesis is that the signal is essentially deterministic and consists of N sinusoids in noise. The associated ACF for real data is

$$r(p) = 2\sum_{k=1}^{N} |S_k|^2 \cos(p\omega_k) + \sigma_e^2 \delta(p) \tag{2.140}$$

where ω_k are the radial frequencies of the sinusoids and S_k are the amplitudes. In matrix form,

$$\begin{bmatrix} r(0) - \sigma_e^2 \\ r(1) \\ r(2) \\ \vdots \\ r(N) \end{bmatrix} = 2 \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \cos\omega_1 & \cos\omega_2 & \cdots & \cos\omega_N \\ \cos 2\omega_1 & \cos 2\omega_2 & \cdots & \cos 2\omega_N \\ \vdots & \vdots & & \vdots \\ \cos N\omega_1 & \cos N\omega_2 & \cdots & \cos N\omega_N \end{bmatrix}$$

$$\times \begin{bmatrix} |S_1|^2 \\ |S_2|^2 \\ \vdots \\ |S_N|^2 \end{bmatrix} \tag{2.141}$$

The analysis of the signal consists of finding out the sinusoid frequencies and amplitudes and the noise power σ_e^2. To perform that task, we use the signal sequence $x(n)$. According to the above hypothesis, it can be expressed by

$$x(n) = x_p(n) + e(n) \tag{2.142}$$

with

$$x_p(n) = \sum_{i=1}^{N} a_i x_p(n-i)$$

Now, the data signal satisfies the recursion

$$x(n) = \sum_{i=1}^{N} a_i x(n-i) + e(n) - \sum_{i=1}^{N} a_i e(n-i) \tag{2.143}$$

which is just a special kind of ARMA signal, with $b_0 = 1$ and $b_i = -a_i$ in time domain relation (2.76). Therefore results derived in Section 2.6 can be applied.

The impulse response can be computed recursively, and relations (2.82) yield $h_k = \delta(k)$. The auxiliary variable in (2.85) is $d(p) = -a_p (1 \leqslant p \leqslant N)$. Rewriting the equations giving the autocorrelation values (2.84) leads to

$$r(p) = \sum_{i=1}^{N} a_i r(p-i) + \sigma_e^2(-a_p), \quad 1 \leqslant p \leqslant N \tag{2.144}$$

or, in matrix form for real data,

$$\begin{bmatrix} r(0) & r(1) & \cdots & r(N) \\ r(1) & r(0) & \cdots & r(N-1) \\ \vdots & \vdots & \ddots & \vdots \\ r(N) & r(N-1) & \cdots & r(0) \end{bmatrix} \begin{bmatrix} 1 \\ -a_1 \\ \vdots \\ -a_N \end{bmatrix} = \sigma_e^2 \begin{bmatrix} 1 \\ -a_1 \\ \vdots \\ -a_N \end{bmatrix} \tag{2.145}$$

This is an eigenvalue equation. The signal autocorrelation matrix is symmetric, and therefore all eigenvalues are greater than or equal to zero. For N sinusoids without noise, the $(N+1) \times (N+1)$ autocorrelation matrix has one eigenvalue equal to zero; adding to the signal a white noise component of power σ_e^2 results in adding σ_e^2 to all eigenvalues of the autocorrelation matrix. Thus, the noise power σ_e^2 is the smallest eigenvalue of the signal, and the recursion coefficients are the entries of the associated eigenvector. As shown in the next chapter, the roots of the filter

$$A(z) = 1 - \sum_{i=1}^{N} a_i z^{-1} \tag{2.146}$$

called the minimum eigenvalue filter, are located on the unit circle in the complex plane and give the frequencies of the sinusoids. The analysis is then completed by solving the linear system (2.141) for the individual sinusoid powers. The complete procedure, called the Pisarenko method, is presented in more detail in a subsequent chapter [12].

So, it is very important to notice that a signal given by a limited set of correlation coefficients can always be viewed as a set of sinusoids in noise. That explains why the study of sinusoids in noise is so important for signal analysis and, more generally, for processing.

In practice, the selection of an analysis strategy is guided by a priori information on the signal and its generation process.

2.12. MULTIDIMENSIONAL SIGNALS

Most of the algorithms and analysis techniques presented in this book are for monodimensional real or complex sequences, which make up the bulk of the applications. However, the extension to multidimensional signals can be quite straightforward and useful in some important cases—for example, those involving multiple sources and receivers, as in geophysics, underwater acoustics, and multiple-antenna transmission systems [13].

A multidimensional signal is defined as a vector of N sequences

$$X(n) = \begin{bmatrix} x_1(n) \\ x_2(n) \\ \vdots \\ x_N(n) \end{bmatrix}$$

For example, the source and receiver vectors in Figure 1.1 are multidimensional signals. The N sequences are assumed to be dependent; otherwise they could be treated as N different scalar signals. They are characterized by the joint density function between them.

A second-order stationary multidimensional random signal is characterized by a mean vector M_x and a covariance matrix R_{xx}:

$$M_x = \begin{bmatrix} E[x_1(n)] \\ E[x_2(n)] \\ \vdots \\ E[x_N(n)] \end{bmatrix}, \quad R_{xx} = E[(X(n) - M_x)(X(n) - M_x)^t] \tag{2.147}$$

The diagonal terms of R_{xx} are the variances of the signal elements. If the elements in the vector are each Gaussian, then they are jointly Gaussian and have a joint density:

$$p(X) = \frac{1}{(2\pi)^{N/2}[\det R_{xx}]^{1/2}} \exp[-\tfrac{1}{2}(X - M_x)^t R_{xx}^{-1}(X - M_x)] \tag{2.148}$$

For the special case $N = 2$,

$$R_{xx} = \begin{bmatrix} \sigma_{x_1}^2 & \rho\sigma_{x_1}\sigma_{x_2} \\ \rho\sigma_{x_1}\sigma_{x_1} & \sigma_{x_2}^2 \end{bmatrix} \tag{2.149}$$

with ρ the correlation coefficient defined by

$$\rho = \frac{1}{\sigma_{x_1}\sigma_{x_2}} E[(x_1 - m_1)(x_2 - m_2)] \tag{2.150}$$

If the signal elements are independent, R_{xx} is a diagonal matrix and

$$p(X) = \prod_{i=1}^{N} \frac{1}{\sigma_i^2\sqrt{2\pi}} \exp\left[-\frac{(x_i - m_i)^2}{2\sigma_i^2}\right] \tag{2.151}$$

Furthermore, if all the variances are equal, then

$$R_{xx} = \sigma^2 I_N \tag{2.152}$$

This situation is frequently encountered in roundoff noise analysis in implementations.

For complex data, the Gaussian joint density (2.148) takes a slightly different form:

$$p(X) = \frac{1}{\pi^N} \frac{1}{\det R_{xx}} \exp[-(X - M_x)^{*t} R_{xx}^{-1}(X - M_x)] \tag{2.153}$$

Multidimensional signals appear naturally in state variable systems, as shown in Section 2.7.

2.13. NONSTATIONARY SIGNALS

A signal is nonstationary if its statistical character changes with time. The fundamental decomposition can be extended to such a signal, and the regular component is

$$x_r(n) = \sum_{i=0}^{\infty} h_i(n) e(n-i) \tag{2.154}$$

where $e(n)$ is a stationary white noise. The generating filter impulse response coefficients are time dependent. An instantaneous spectrum can be defined as

$$S(f, n) = \sigma_e^2 \left| \sum_{i=0}^{\infty} h_i(n) e^{-j2\pi fi} \right|^2 \tag{2.155}$$

So, nonstationary signals can be generated or modeled by the techniques developed for stationary signals, but with additional means to make the system coefficients time varying [14]. For example, the ARMA signal is

$$x(n) = \sum_{i=0}^{N} b_i(n) e(n-i) + \sum_{i=1}^{N} a_i(n) x(n-i) \tag{2.156}$$

The coefficients can be generated in various ways. For example, they can be produced as weighted sums of K given time functions $f_k(n)$:

$$a_i(n) = \sum_{k=1}^{K} a_{ik} f_k(n) \tag{2.157}$$

These time functions may be periodic functions or polynomials; a simple case is the one-degree polynomial, which corresponds to a drift of the coefficients. The signal depends on $(2N+1)K$ time-independent parameters.

The set of coefficients can also be a multidimensional signal. A realistic example in that class is shown in Figure 2.11. The N time-varying filter coefficients $a_i(n)$ are obtained as the outputs of N fixed-coefficient filters fed by independent white noises with same variances. A typical choice for the coefficient filter transfer function is the first-order low-pass function

$$H_i(z) = \frac{1}{1 - \gamma z^{-1}}, \quad 0 \ll \gamma < 1 \tag{2.158}$$

whose time constant is

$$\tau = \frac{1}{1 - \gamma} \tag{2.159}$$

For γ close to unity, the time constant is large and the filter coefficients are subject to slow variations.

The analysis of nonstationary signals is complicated because the ergodicity assumption can no longer be used and statistical parameters cannot be computed through time averages. Natural signals are nonstationary. However, they are often slowly time varying and can then be assumed stationary for short periods of time.

2.14. NATURAL SIGNALS

To illustrate the preceding developments, we give several signals from different application fields in this section.

Speech is probably the most commonly processed natural signal through digital communication networks. The waveform for the word "FATHER" is shown in Figure 2.12. The sampling rate is 8 kHz, and the duration is

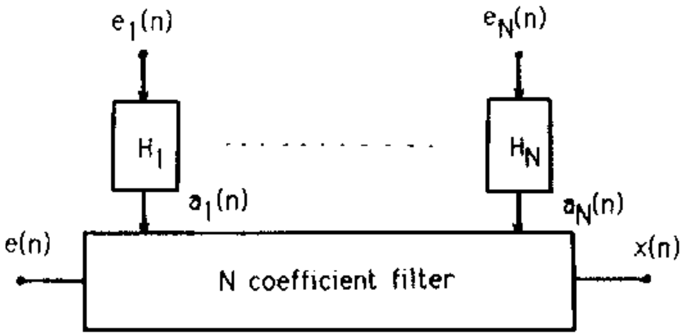


FIG. 2.11 Generation of a nonstationary signal.

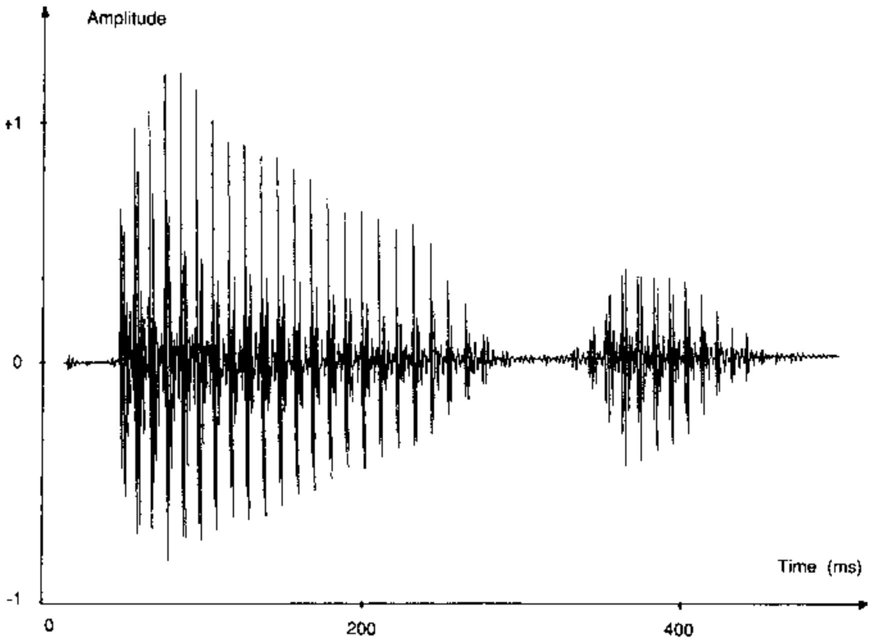


FIG. 2.12 Speech waveform for the word "father."

about 0.5 s. Clearly, it is nonstationary. Speech consists of phonemes and can be considered as stationary on durations ranging from 10 to 25 ms.

It can be modeled as the output of a time-varying purely recursive filter (AR model) fed by either a string of periodic pulses for voiced sections or a string of random pulses for unvoiced sections [15].

The output of the demodulator of a frequency-modulated continuous wave (FMCW) radar is shown in Figure 2.13. It is basically a distorted sinusoid corrupted by noise and echoes. The main component frequency is representative of the distance to be measured.

An image can be represented as a one-dimensional signal through scanning. In Figure 2.14, three lines of a black-and-white contrasted picture are shown; a line has 256 samples. The similarities between consecutive lines can be observed, and the amplitude varies quickly within every line. The picture represents a house.

2.15. SUMMARY

Any stationary signal can be decomposed into periodic and random components. The characteristics of both classes can be studied by considering

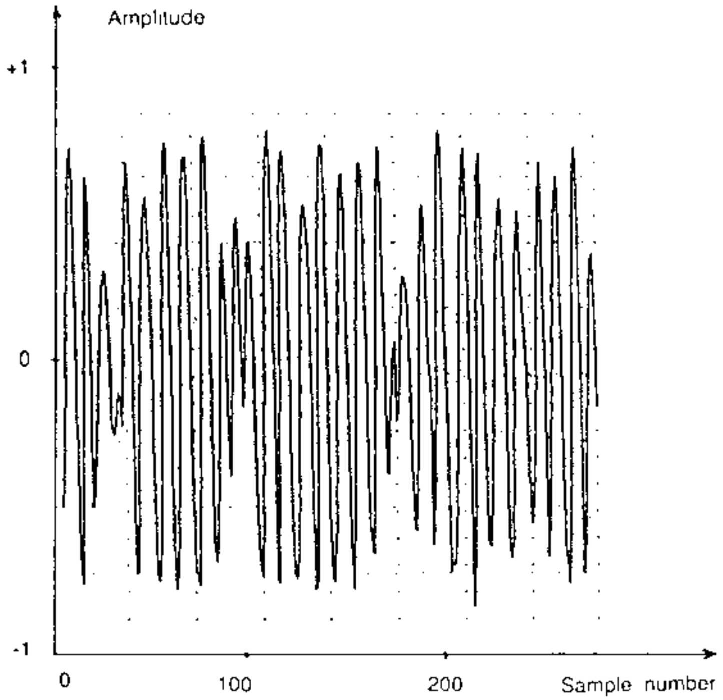


FIG. 2.13 FMCW radar signal.

as main parameters, the ACF, the spectrum, and the generating model. Periodic signals have been analyzed first. Then random signals have been defined, with attention being focused on wide-sense stationary signals; they have second-order statistics which are independent of time. Synthetic random signals can be generated by a filter fed with white noise. The Gaussian amplitude distribution is especially important because of its nice statistical properties, but also because it is a model adequate for many real situations. The generating filter structures correspond to various output signal classes: MA, AR, and ARMA. The concept of linear prediction is related to a generating filter model, and the class of predictable signals has been defined. A proof of the fundamental Wold decomposition has been presented, and, as an application, it has been shown that a signal specified by a limited set of correlation coefficients can be viewed as a set of sinusoids in noise. That is the harmonic decomposition.

In practice, signals are nonstationary, and, in general, short-term stationarity or slow variations have to be assumed. Several natural signal exam-

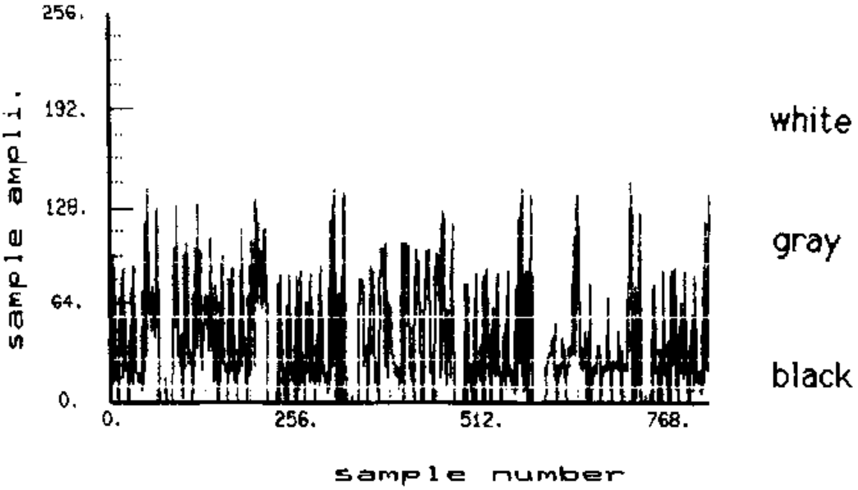


FIG. 2.14 Image signal: three lines of a black-and-white picture.

ples, namely speech, radar, and image samples, have been selected to illustrate the theory.

EXERCISES

1. Calculate the z-transform $Y_R(z)$ of the damped cosinusoid

$$y_R(n) = \begin{cases} 0, & n < 0 \\ e^{-0.1n} \cos \dfrac{n\pi}{2}, & n \geqslant 0 \end{cases}$$

and show the poles in the complex plane.

Give the signal energy spectrum and verify the energy relationship

$$E_y = \sum_{n=0}^{\infty} y_R^2(n) = \frac{1}{2\pi j} \int_{|z|=1} Y_R(z) Y_R(z^{-1}) z^{-1} \, dz$$

Give the coefficients, initial conditions, and diagram of the second-order section which generates $y_R(n)$.

2. Find the ACF of the signal

$$x(n) = \cos n\frac{\pi}{3} + \frac{1}{2} \sin n\frac{\pi}{4}$$

Determine the recurrence equation satisfied by $x(n)$ and give the initial conditions.

3. Evaluate the mean and variance associated with the uniform probability density function on the interval $[x_1, x_2]$. Comment on the results.
4. Consider the signal

$$x(n) = \begin{cases} 0 & n < 0 \\ 0.8x(n-1) + e(n), & n \geqslant 1 \end{cases}$$

assuming $e(n)$ is a stationary zero mean random sequence with power $\sigma_e^2 = 0.5$. The initial condition is deterministic with value $x(0) = 1$.

Calculate the mean sequence $m_n = E[x(n)]$. Give the recursion, for the variance sequence. What is the stationary solution. Calculate the ACF of the stationary signal.
5. Find the first three terms of the ACF of the AR signal.

$$x(n) = 1.27x(n-1) - 0.81x(n-2) + e(n)$$

where $e(n)$ is a unit power centered white noise.
6. An ARMA signal is defined by the recursion

$$x(n) = e(n) + 0.5e(n-1) + 0.9e(n-2) + x(n-1) - 0.5x(n-2)$$

where $e(n)$ is a unit variance centered white noise. Calculate the generating filter z-transfer function and its impulse response. Derive the signal ACF.
7. A two-dimensional signal is defined by

$$X(n) = \begin{cases} \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, & n \leqslant 0 \\ \begin{bmatrix} 0.63 & 0.36 \\ 0.09 & 0.86 \end{bmatrix} X(n-1) + \begin{bmatrix} 0.01 \\ 0.06 \end{bmatrix} e(n), & n \geqslant 1 \end{cases}$$

where $e(n)$ is a unit power centered white noise. Find the covariance propagation equation and calculate the stationary solution.
8. A measurement has supplied the signal autocorrelation values $r(0) = 5.75$, $r(1) = 4.03$, $r(2) = 0.46$. Calculate the two coefficients of the second-order linear predictor and the prediction error power. Give the corresponding signal power spectrum.
9. Find the eigenvalues of the matrix

$$R_3 = \begin{bmatrix} 1.00 & 0.70 & 0.08 \\ 0.70 & 1.00 & 0.70 \\ 0.08 & 0.70 & 1.00 \end{bmatrix}$$

and the coefficients of the minimum eigenvalue filter. Locate the zeros of that filter and give the harmonic spectrum. Compare with the prediction spectrum obtained in the previous exercise.

REFERENCES

1. T. W. Anderson, *The Statistical Analysis of Time Series*, Wiley, New York, 1971.
2. G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, Holden-Day, San Francisco, 1976.
3. J. E. Cadzow and H. Van Landingham, *Signals, Systems and Transforms*, Prentice-Hall, Englewood Cliffs, N.J., 1985.
4. A. V. Oppenheim, A. S. Willsky, and I. T. Young, *Signals and Systems*, Prentice-Hall, Englewood Cliffs, N.J., 1983.
5. W. B. Davenport, *Probability and Random Processes*, McGraw-Hill, New York, 1970.
6. T. J. Terrel, *Introduction to Digital Filters*, Wiley, New York, 1980.
7. D. Graupe, D. J. Krause, and J. B. Moore, ''Identification of ARMA Parameters of Time Series,'' *IEEE Transactions* **AC-20**, 104–107 (February 1975).
8. J. Lamperti, *Stochstic Processes*, Springer, New York, 1977.
9. R. G. Jacquot, *Modern Digital Control Systems*, Marcel Dekker, New York, 1981.
10. A. Papoulis, ''Predictable Processes and Wold's Decomposition: A Review,'' *IEEE Transactions* **ASSP-33**, 933–938 (August 1985).
11. S. M. Kay and S. L. Marple, ''Spectrum Analysis: A Modern Perspective,'' *Proc. IEEE* **69**, 1380–1419 (November 1981).
12. V. F. Pisarenko, ''The Retrieval of Harmonics from a Covariance Function,'' *Geophysical J. Royal Astronomical Soc.* **33**, 347–366 (1973).
13. D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing*, Prentice-Hall, Englewood-Cliffs, N.J., 1984.
14. Y. Grenier, ''Time Dependent ARMA Modeling of Non Stationary Signals,'' *IEEE Transactions* **ASSP-31**, 899–911 (August 1983).
15. L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, N.J., 1978.

3
Correlation Function and Matrix

The operation and performance of adaptive filters are tightly related to the statistical parameters of the signals involved. Among these parameters, the correlation functions take a significant place. In fact, they are crucial because of their own value for signal analysis but also because their terms are used to form correlation matrices. These matrices are exploited directly in some analysis techniques. However, in the efficient algorithms for adaptive filtering considered here, they do not, in general, really show up, but they are implied and actually govern the efficiency of the processing. Therefore an in-depth knowledge of their properties is necessary. Unfortunately it is not easy to figure out their characteristics and establish relations with more accessible and familiar signal features, such as the spectrum.

This chapter presents correlation functions and matrices, discusses their most useful properties, and, through examples and applications, makes the reader accustomed to them and ready to exploit them. To begin with, the correlation functions, which have already been introduced, are presented in more detail.

3.1. CROSS-CORRELATION AND AUTOCORRELATION

Assume that two sets of N real data, $x(n)$ and $y(n)$, have to be compared, and consider the scalar a which minimizes the cost function

$$J(N) = \sum_{n=1}^{N} [y(n) - ax(n)]^2 \tag{3.1}$$

Setting to zero the derivative of $J(N)$ with respect to a yields

$$a = \frac{\sum_{n=1}^{N} x(n)y(n)}{\sum_{n=1}^{N} x^2(n)} \tag{3.2}$$

The minimum of the cost function is

$$J_{\min}(N) = [1 - k^2(N)] \sum_{n=1}^{N} y^2(n) \tag{3.3}$$

with

$$k(N) = \frac{\sum_{n=1}^{N} x(n)y(n)}{\sqrt{\sum_{n=1}^{N} x^2(n)}\sqrt{\sum_{n=1}^{N} y^2(n)}} \tag{3.4}$$

The quantity $k(N)$, cross-correlation coefficient, is a measure of the degree of similarity between the two sets of N data. To point out the practical significance of that coefficient, we mention that it is the basic parameter of an important class of prediction filters and adaptive systems—the least squares (LS) lattice structures in which it is computed in real time recursively.

From equations (3.2) and (3.4), the correlation coefficient $k(N)$ is bounded by

$$|k(N)| \leqslant 1 \tag{3.5}$$

and it is independent of the signal energies; it is said to be normalized.

If instead of $x(n)$ we consider a delayed version of the signal in the above derivation, a cross-correlation function can be obtained. The general, unnormalized form of the cross-correlation function between two real sequences $x(n)$ and $y(n)$ is defined by

$$r_{yx}(p) = E[y(n)x(n-p)] \tag{3.6}$$

For stationary and ergodic signals we have

$$r_{yx}(p) = \lim_{N \to \infty} \frac{1}{2N+1} \sum_{n=-N}^{N} y(n)x(n-p) \tag{3.7}$$

Several properties result from the above definitions. For example:

$$r_{yx}(-p) = E\{x(n+p)y[(n+p)-p]\} = r_{xy}(p) \tag{3.8}$$

If two random zero mean signals are independent, their cross-correlation functions are zero. In any case, when p approaches infinity the cross-correlation approaches zero. The magnitudes of $r_{yx}(p)$ are not, in general, maximum at the origin, but they are bounded. The inequality

$$[y(n) - x(n-p)]^2 \geqslant 0 \tag{3.9}$$

yields the bound

$$|r_{yx}(p)| \leqslant \tfrac{1}{2}[r_{xx}(0) + r_{yy}(0)] \tag{3.10}$$

If the signals involved are the input and output of a filter

$$y(n) = \sum_{i=0}^{\infty} h_i x(n-i) \tag{3.11}$$

and

$$r_{yx}(p) = E[y(n)x(n-p)] = \sum_{i=0}^{\infty} h_i r_{xx}(p-i) \tag{3.12}$$

the following relationships, in which the convolution operator is denoted $*$, can be derived:

$$\begin{aligned} r_{yx}(p) &= r_{xx}(p) * h(p) \\ r_{xy}(p) &= r_{xx}(p) * h(-p) \\ r_{yy}(p) &= r_{xx}(p) * h(p) * h(-p) \end{aligned} \tag{3.13}$$

When $y(n) = x(n)$, the autocorrelation function (ACF) is obtained; it is denoted $r_{xx}(p)$ or, more simply, $r(p)$, if there is no ambiguity. The following properties hold:

$$r(p) = r(-p), \quad |r(p)| \leqslant r(0) \tag{3.14}$$

For $x(n)$ a zero mean white noise with power σ_x^2,

$$r(p) = \sigma_x^2 \delta(p) \tag{3.15}$$

and for a sine wave with amplitude S and radial frequency ω_0,

$$r(p) = \frac{S^2}{2} \cos p\omega_0 \tag{3.16}$$

The ACF is periodic with the same period. Note that from (3.15) and (3.16) a simple and efficient noise-elimination technique can be worked out to retrieve periodic components, by just dropping the terms $r(p)$ for small p in the noisy signal ACF.

The Fourier transform of the ACF is the signal spectrum. For the cross-correlation $r_{yx}(p)$ it is the cross spectrum $S_{yx}(f)$.

Considering the Fourier transform $X(f)$ and $Y(f)$ of the sequences $x(n)$ and $y(n)$, equation (3.7) yields

$$S_{yx}(f) = Y(f)\bar{X}(f) \tag{3.17}$$

where $\bar{X}(f)$ is the complex conjugate of $X(f)$.

The frequency domain correspondence for the set of relationships (3.13) is found by introduction of the filter transfer function:

$$H(f) = \frac{Y(f)}{X(f)} = \frac{Y(f)\bar{X}(f)}{|X(f)|^2} \tag{3.18}$$

Now

$$\begin{aligned} S_{yx}(f) &= S_{xx}(f)H(f) \\ S_{xy}(f) &= S_{xx}(f)\bar{H}(f) \\ S_{yy}(f) &= S_{xx}(f)|H(f)|^2 \end{aligned} \tag{3.19}$$

The spectra and cross spectra can be used to compute ACF and cross-correlation function, through Fourier series development, although it is often the other way round in practice.

Most of the above definitions and properties can be extended to complex signals. In that case the cross-correlation function (3.6) becomes

$$r_{yx}(p) = E[\,y(n)\bar{x}(n-p)] \tag{3.20}$$

In the preceding chapter the relations between correlation functions and model coefficients have been established for MA, AR, and ARMA stationary signals. In practice, the correlation coefficients must be estimated from available data.

3.2. ESTIMATION OF CORRELATION FUNCTIONS

The signal data may be available as a finite-length sequence or as an infinite sequence, as for stationary signals. In any case, due to the limitations in processing means, the estimations have to be restricted to a finite time window. Therefore a finite set of N_0 data is assumed to be used in estimations.

A first method to estimate the ACF $r(p)$ is to calculate $r_1(p)$ by

$$r_1(p) = \frac{1}{N_0} \sum_{n=p+1}^{N_0} x(n)x(n-p) \tag{3.21}$$

The estimator is biased because

$$E[r_1(p)] = \frac{N_0 - p}{N_0} r(p) \tag{3.22}$$

However, the bias approaches zero as N_0 approaches infinity, and $r_1(p)$ is asymptotically unbiased.

An unbiased estimator is

$$r_2(p) = \frac{1}{N_0 - p} \sum_{n=p+1}^{N_0} x(n)x(n-p) \tag{3.23}$$

In order to limit the range of the estimations, which are exploited subsequently, we introduce a normalized form, given for the unbiased estimator by

$$r_{n2}(p) = \frac{\sum_{n=p+1}^{N_0} x(n)x(n-p)}{\left[\sum_{n=p+1}^{N_0} x^2(n) \sum_{n=p+1}^{N_0} x^2(n-p) \right]^{1/2}} \tag{3.24}$$

The variance is

$$\text{var}\{r_{n2}(p)\} = E[r_{n2}^2(p)] - E^2[r_{n2}(p)] \tag{3.25}$$

and it is not easily evaluated in the general case because of the nonlinear functions involved. However, a linearization method, based on the first derivatives of Taylor expansions, can be applied [1]. For uncorrelated pairs in equation (3.24), we obtain

$$\text{var}\{r_{n2}(p)\} \approx \frac{[1 - r_n^2(p)]^2}{N_0 - p} \tag{3.26}$$

$$r_n(p) = \frac{E[x(n)x(n-p)]}{[E[x^2(n)]E[x^2(n-p)]]^{1/2}} \tag{3.27}$$

is the theoretical normalized ACF.

Thus, the variance also approaches zero as the number of samples approaches infinity, and $r_{n2}(p)$ is a consistent estimate.

The calculation of the estimator according to (3.24) is a demanding operation for large N_0. In a number of applications, like radiocommunications, the correlation calculation may be the first processing operation, and it has to be carried out on high-speed data. Therefore it is useful to have less costly methods available. Such methods exist for Gaussian random signals, and they can be applied as well to many other signals.

The following property is valid for a zero mean Gaussian signal $x(n)$:

$$r(p) = \frac{\pi}{2} r_{yx}(p) r_{yx}(0) \tag{3.28}$$

where

$$y(n) = \text{sign}\{x(n)\}, \quad y(n) = \pm 1$$

Hence the ACF estimate is

$$r_3(p) = c \frac{1}{N_0 - p} \sum_{n=p+1}^{N_0} x(n-p)\,\text{sign}\{x(n)\} \tag{3.29}$$

where

$$c = \frac{\pi}{2} r_{yx}(0) = \frac{\pi}{2N_0} \sum_{n=1}^{N_0} |x(n)|$$

In normalized form, we have

$$r_{n3}(p) = \frac{N_0}{N_0 - p} \frac{\sum_{n=p+1}^{N_0} x(n-p)\,\text{sign}\{x(n)\}}{\sum_{n=1}^{N_0} |x(n)|} \tag{3.29a}$$

A multiplication-free estimate is obtained [2], which is sometimes called the hybrid sign correlation or relay correlation. For uncorrelated pairs and p small with respect to N_0, the variance is approximately [3]

$$\text{var}\,\{r_{n3}(p)\} \approx \frac{1}{N_0}\left[\frac{\pi}{2} - 2r_n(p)\text{Arcsin}[r_n(p)] + \frac{\pi}{2} r_n^2(p) - 2r_n^2(p)\sqrt{1 - r_n^2(p)}\right] \tag{3.30}$$

This estimator is also consistent.

The simplification process can be carried one step further, through the polarity coincidence technique, which relies on the following property of zero mean Gaussian signals:

$$r(p) = r(0) \sin\left[\frac{\pi}{2} E[\text{sign}\{x(n)x(n-1)\}]\right] \tag{3.31}$$

The property reflects the fact that a Gaussian function is determined by its zero crossings, except for a constant factor. Hence we have the simple estimate

$$r_{n4}(p) = \sin\left(\frac{\pi}{2}\frac{1}{N_0 - p}\sum_{n=p+1}^{N_0} \mathrm{sign}\{x(n)x(n-p)\}\right) \tag{3.32}$$

which is called the sign or polarity coincidence correlator. Its variance can be approximated for N_0 large by [4]

$$\mathrm{var}\{r_{n4}(p)\} \approx \frac{1}{N_0}\frac{\pi^2}{4}[1 - r_n^2(p)]\left[1 - \left(\frac{2}{\pi}\mathrm{Arcsin}\, r(p)\right)^2\right] \tag{3.33}$$

In a Gaussian context, a more precise estimator is based on the mean of the absolute differences. Consider the sequence

$$z_p(n) = x(n) - x(n-p) \tag{3.34}$$

Its variance is

$$E[z_p^2(n)] = 2[r(0) - r(p)] = 2r(0)\left[1 - \frac{r(p)}{r(0)}\right] \tag{3.35}$$

and,

$$\frac{r(p)}{r(0)} = 1 - \frac{1}{2}\frac{E[z_p^2(n)}{r_0} \tag{3.36}$$

Using the Gaussian assumption and equation (3.28), an estimator is obtained as

$$r_{n5}(p) = 1 - \frac{1}{2}\left|\frac{\sum_{n=p}^{N_0} |x(n) - x(n-p)|}{\sum_{n=p}^{N_0} (|x(n)| + |x(n-p)|)}\right|^2 \tag{3.37}$$

The variances of the three normalized estimators r_{n2}, r_{n3}, and r_{n4} are shown in Figure 3.1 versus the theoretical autocorrelation (AC) $r(p)$. Clearly the lower computational cost of the hybrid sign and polarity coincidence correlators is paid for by a lower accuracy. As concerns the estimator r_{n5}, it has the smallest variance and is closer to the theory [6].

The performance evaluation of the estimators has been carried out under the assumption of uncorrelated sample pairs, which is no longer valid when the estimate is extracted on the basis of a single realization of a correlated process, i.e., a single data record. The evaluation can be carried out by considering the correlation between pairs of samples; it shows a degradation in performance [5].

For example, if the sequence $x(n)$ is a bandlimited noise with bandwidth B, the following bound can be derived for a large number of data N_0 [7]:

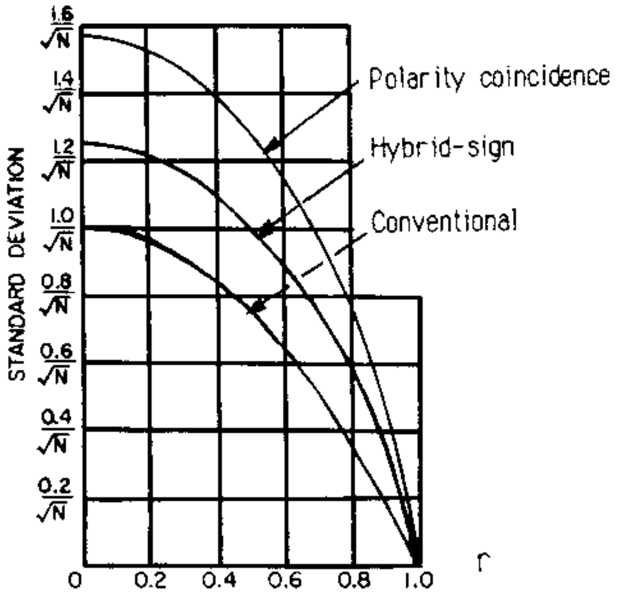


FIG. 3.1 Standard deviation of estimators versus theoretical autocorrelation for large number of data N.

$$\operatorname{var}\{r_2(p)\} \leqslant \frac{r^2(0)}{B(N_0 - p)} \tag{3.38}$$

The worst case occurs when the bandwidth B is half the sampling frequency; then $x(n)$ is a white noise, and the data are independent, which leads to

$$\operatorname{var}\{r_2(p)\} \leqslant \frac{2r^2(0)}{N_0 - p} \tag{3.39}$$

This bound is compatible with estimation (3.26). Anyway the estimator for correlated data is still consistent for fixed p.

Furthermore, the Gaussian hypothesis is also needed for the hybrid sign and polarity coincidence estimators. So, these estimators have to be used with care in practice. An example of performance comparison is presented in Figure 3.2 for a speech sentence of 1.25 s corresponding to $N_0 = 10{,}000$ samples.

In spite of noticeable differences between conventional and polarity coincidence estimators for small AC values, the general shape of the function is the same for both.

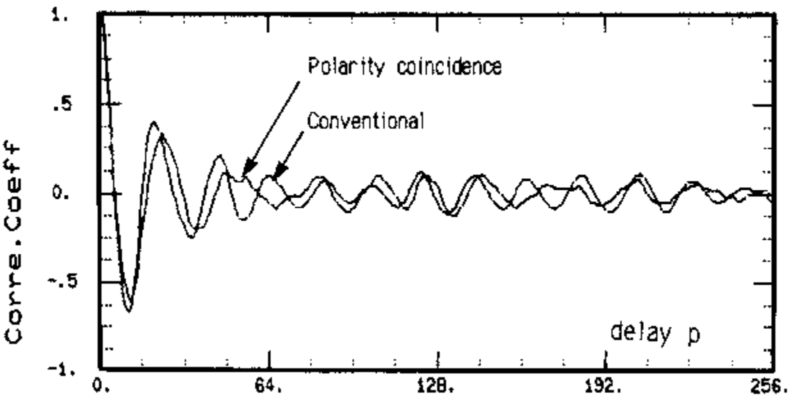


FIG. 3.2 Correlation function estimation for a speech sentence.

Concerning correlated data, an important aspect of simplified correlators applied to real-life data is that they may attenuate or even cancel small useful components. Therefore, if small critical components in the signal have to be kept, the correlation operation accuracy in equipment must be determined to ensure that they are kept. Otherwise, reduced word lengths, such as 8 bits or 4 bits or even less, can be employed.

The first estimator introduced, $r_1(p)$, is just a weighted version of $r_2(p)$; hence its variance is

$$\text{var}\{r_1(p)\} = \text{var}\left\{\frac{N_0 - p}{N_0} r_2(p)\right\} = \left(\frac{N_0 - p}{N_0}\right)^2 \text{var}\{r_2(p)\} \tag{3.40}$$

The estimator $r_1(p)$ is biased, but it has a smaller variance than $r_2(p)$. It is widely used in practice.

The above estimation techniques can be expanded to complex signals, using definition (3.20). For example, the hybrid complex estimator, the counterpart of $r_3(p)$ in (3.29), is defined by

$$r_{3c}(p) = \frac{\pi}{2}\bar{r}_{yxc}(0) r_{yxc}(p) \tag{3.41}$$

with

$$r_{yxc}(p) = \frac{1}{N}\sum_{m=1}^{4} e^{-j(m-1)\pi/2} \sum_{I_m} x(n)$$

where the summation domain itself is defined by

$$I_m = \begin{cases} 1 \leqslant n \leqslant N_0 - p \\ (m-1)\pi/2 \leqslant \text{Arg}[x(n-p)] \leqslant m\dfrac{\pi}{2} \end{cases}$$

The sign function has been replaced by a phase discretization operator that uses the signs of the real components. This computationally efficient estimator is accurate for the complex Gaussian stationary processes [8].

So far, stationarity has been assumed. However, when the signal is just short-term stationary, the estimation has to be carried out on a compatible short-time window. An updated estimation is obtained every time if the window slides on the time axis; it is a sliding window technique, in which the oldest datum is discarded as a new datum enters the summation.

An alternative, more convenient, and widely used approach is recursive estimation.

3.3. RECURSIVE ESTIMATION

The time window estimation, according to (3.21) or (3.23), is a finite impulse response (FIR) filtering, which can be approximated by an infinite impulse response (IIR) filtering method. The simplest IIR filter is the first-order low-pass section, defined by

$$y(n) = x(n) + by(n-1), \quad 0 < b < 1 \tag{3.42}$$

Before investigating the properties of the recursive estimator, let us consider the simple case where the input sequence $x(n)$ is the sum of a constant m and a zeor mean white noise $e(n)$ with powerσ_e^2. Furthermore, if $y(n) = 0$ for $n < 0$, then

$$y(n) = m\frac{1 - b^{n+1}}{1 - b} + \sum_{i=0}^{n} b^i e(n-i) \tag{3.43}$$

Taking the expectation gives

$$E[y(n)] = m\frac{1 - b^{n+1}}{1 - b} \tag{3.44}$$

Therefore, an estimation of the input mean m is provided by the product $(1 - b)y(n)$, that is by the first-order section with z-transfer function:

$$H(z) = \frac{1 - b}{1 - bz^{-1}} \tag{3.45}$$

The noise power σ_0^2 at the output of such a filter is

$$\sigma_0^2 = \sigma_e^2 \frac{1 - b}{1 + b} \tag{3.46}$$

Consequently, the input noise is all the more attenuated than b is close to unity. Taking $b = 1 - \delta$, $0 < \delta \ll 1$ yields

$$\sigma_0^2 \approx \sigma_e^2 \frac{\delta}{2} \tag{3.47}$$

The diagram of the recursive estimator is shown in Figure 3.3. The corresponding recursive equation is

$$M(n) = (1 - \delta)M(n - 1) + \delta x(n) \tag{3.48}$$

According to equation (3.44) the estimation is biased and the duration needed to reach a good estimation is inversely proportional to δ. In digital filter theory, a time constant τ can be defined by

$$e^{-1/\tau} = b \tag{3.49}$$

which for b close to 1, leads to

$$\tau \approx \frac{1}{1 - b} = \frac{1}{\delta} \tag{3.50}$$

In order to relate recursive and window estimations, we define an equivalence. The FIR estimator

$$y(n) = \frac{1}{N_0} \sum_{i=0}^{N_0 - 1} x(n - i) \tag{3.51}$$

which is unbiased, yields the output noise power

$$(\sigma_0')^2 = \frac{\sigma_e^2}{N_0} \tag{3.52}$$

Comparing with (3.47), we get

$$2\tau \approx N_0 \tag{3.53}$$

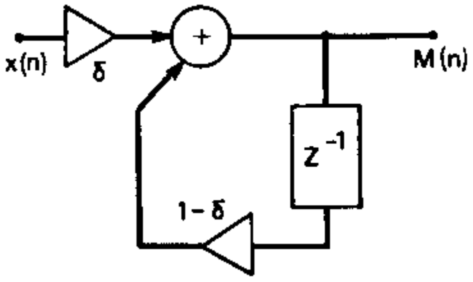


FIG. 3.3 Recursive estimator.

The recursive estimator can be considered equivalent to a window estimator whose width is twice the time constant.

For example, consider the recursive estimation of the power of a white Gaussian signal $x(n)$, the true value being σ_x^2. The input to the recursive estimator, $x^2(n)$, can be viewed as the sum of the constant $m = \sigma_x^2$ and a zero mean white noise, with variance

$$\sigma_e^2 = e[x^4(n)] - \sigma_x^4 = 2\sigma_x^4 \tag{3.54}$$

The standard deviation of the output, ΔP, is

$$\Delta P = \sigma_x^2 \sqrt{\delta} \tag{3.55}$$

and the relative error on the estimated power is $\sqrt{\delta}$.

Recursive estimation techniques can be applied to the ACF and to cross-correlation coefficients; a typical example is the lattice adaptive filter.

Once the ACF has been estimated, it can be used for analysis or any further processing.

3.4. THE AUTOCORRELATION MATRIX

Often in signal analysis or adaptive filtering, the ACF appears under the form of a square matrix, called the autocorrelation matrix.

The $N \times N$ AC matrix R_{xx} of the real sequence $x(n)$ is defined by

$$R_{xx} = \begin{bmatrix} r(0) & r(1) & \cdots & r(N-1) \\ r(1) & r(0) & \cdots & r(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(N-1) & r(N-2) & \cdots & r(0) \end{bmatrix} \tag{3.56}$$

It is a symmetric matrix and $R_{xx}^t = R_{xx}$. For complex data the definition is slightly different:

$$R_{xx} \begin{bmatrix} r(0) & r(1) & \cdots & r(N-1) \\ r(-1) & r(0) & \cdots & r(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ r[-(N-1)] & r[-(N-2)] & \cdots & r(0) \end{bmatrix} \tag{3.57}$$

Since $r(-p)$ is the complex conjugate of $r(p)$, the matrix is Hermitian; that is,

$$R_{xx}^* = R_{xx} \tag{3.58}$$

where "*" denotes transposition and complex conjugation.

To illustrate how naturally the AC matrix appears, let us consider an FIR filtering operating with N coefficients:

$$y(n) = \sum_{i=0}^{N-1} h_i x(n-i) \tag{3.59}$$

In vector notation (3.58) is

$$y(n) = H^t X(n) = X^t(n) H$$

The output power is

$$E[y^2(n)] = E[H^t X(n) X^t(n) H] = H^t R_{xx} H \tag{3.60}$$

The inequality

$$H^t R_{xx} H \geqslant 0 \tag{3.61}$$

is valid for any coefficient vector H and characterizes positive semidefinite or nonnegative definite matrices [9]. A matrix is positive definite if

$$H^t R_{xx} H > 0 \tag{3.62}$$

The matrix R_{xx} is also symmetrical about the secondary diagonal; hence it is said to be doubly symmetric or persymmetric. Define by J_N the $N \times N$ co-identity matrix, which acts as a reversing operator on vectors and shares a number of properties with the identity matrix I_N:

$$I_N = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}, \quad J_N = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{3.63}$$

The double symmetry property is expressed by

$$R_{xx} J_N = J_N R_{xx} \tag{3.64}$$

Autocorrelation matrices have an additional property with respect to doubly symmetric matrices, namely their diagonal entries are identical; they are said to have a Toeplitz form or, in short, to be Toeplitz. This property is crucial and leads to drastic simplifications in some operations and particularly the inverse calculation, needed in the normal equations introduced in Section 1.4, for example. Examples of AC matrices can be given for MA and AR signals. If $x(n)$ is an MA signal, generated by filtering a white noise with power σ_e^2 by an FIR filter having $P < N/2$ coefficients, then R_{xx} is a band matrix. For $P = 2$,

$$x(n) = h_0 e(n) + h_1 e(n-1) \tag{3.65}$$

Using the results of Section 2.5 yields

$$R_{\mathrm{MA1}} = \sigma_e^2 \begin{bmatrix} h_0^2+h_1^2 & h_0h_1 & 0 & \cdots & 0 & 0 \\ h_0h_1 & h_0^2+h_1^2 & h_0h_1 & \cdots & 0 & 0 \\ 0 & h_0h_1 & h_0^2+h_1^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_0^2+h_1^2 & h_0h_1 \\ 0 & 0 & 0 & \cdots & h_0h_1 & h_0^2+h_1^2 \end{bmatrix} \tag{3.66}$$

Similarly, for a first-order AR process, we have

$$x(n) = ax(n-1) + e(n)$$

The matrix takes the form

$$R_{\mathrm{AR1}} = \frac{\sigma_e^2}{1-a^2} \begin{bmatrix} 1 & a & a^2 & \cdots & a^{N-1} \\ a & 1 & a & \cdots & a^{N-2} \\ a^2 & a & 1 & \cdots & a^{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a^{N-1} & a^{N-2} & a^{N-3} & \cdots & 1 \end{bmatrix} \tag{3.67}$$

The inverse of the AR signal AC matrix is a band matrix because the inverse of the filter used to generate the AR sequence is an FIR filter. In fact, except for edge effects, it is an MA matrix.

Adjusting the first entry gives for the first-order case

$$R_{\mathrm{AR1}}^{-1} = \frac{1}{\sigma_e^2} \begin{bmatrix} 1 & -a & 0 & \cdots & 0 \\ -a & 1+a^2 & -a & \cdots & 0 \\ 0 & -a & 1+a^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & 1+a^2 & -a \\ 0 & 0 & 0 & -a & 1 \end{bmatrix} \tag{3.68}$$

This is an important result, which is extended and exploited in subsequent sections.

Since AC matrices often appear in linear systems, it is useful, before further exploring their properties, to briefly review linear systems.

3.5. SOLVING LINEAR EQUATION SYSTEMS

Let us consider a set of N_0 linear equations represented by the matrix equation

$$MH = Y \tag{3.69}$$

The column vector Y has N_0 elements. The unknown column vector H has N elements, and the matrix M has N_0 rows and N columns. Depending on the respective values of N_0 and N, three cases can be distinguished. First, when $N_0 = N$, the system is exactly determined and the solution is

$$H = M^{-1}Y \tag{3.70}$$

Second, when $N_0 > N$, the system is overdetermined because there are more equations than unknowns. A typical example is the filtering of a set of N_0 data $x(n)$ by an FIR filter whose N coefficients must be calculated so as to make the output set equal to the given vector Y:

$$\begin{bmatrix} x(0) & 0 & \cdots & 0 \\ x(1) & x(0) & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ x(N-1) & x(N-2) & \cdots & x(0) \\ \vdots & \vdots & & \vdots \\ x(N_0-1) & x(N_0-2) & \cdots & x(N_0-N) \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-1} \end{bmatrix} = \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N_0-1) \end{bmatrix} \tag{3.71}$$

A solution in the LS sense is found by minimizing the scalar J:

$$J = (Y - MH)^t(Y - MH)$$

Through derivation with respect to the entries of the vector H, the solution is found to be

$$H = (M^tM)^{-1}M^tY \tag{3.72}$$

Third, when $N_0 < N$, the system is underdetermined and there are more unknowns than equations. The solution is then

$$H = M^t(MM^t)^{-1}Y \tag{3.73}$$

The solution of an exactly determined system must be found in all cases. The matrix (M^tM) is symmetrical, and standard algorithms exist to solve equation systems based on such matrices, which are assumed positive definite. The Cholesky method uses a triangular factorization of the matrix and needs about $N^3/3$ multiplications; the subroutine is given in Annex 3.1.

Iterative techniques can also be used to solve equation (3.69). The matrix M can be decomposed as

$$M = D + E$$

where D is a diagonal matrix and E is a matrix with zeros on the main diagonal. Now

$$H = D^{-1}Y - D^{-1}EH$$

and an iterative procedure is as follows:

$$\begin{aligned} H_0 &= D^{-1}Y \\ H_1 &= D^{-1}Y - D^{-1}EH_0 \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ H_{n+1} &= D^{-1}Y - D^{-1}EH_n \end{aligned} \tag{3.74}$$

The decrement after n iterations is

$$H_{n+1} - H_n = -(D^{-1}E)^{n+1}D^{-1}Y$$

The procedure may be stopped when the norm of the vector $H_{n+1} - H_n$ falls below a specified value.

3.6. EIGENVALUE DECOMPOSITION

The eigenvalue decomposition of an AC matrix leads to the extraction of the basic components of the corresponding signal [10–13]—hence its significance.

The eigenvalues λ_i and eigenvectors V_i of the $N \times N$ matrix R are defined by

$$RV_i = \lambda_i V_i, \quad 0 \leqslant i \leqslant N-1 \tag{3.75}$$

If the matrix R now denotes the AC matrix R_{xx}, it is symmetric for real signals and Hermitian for complex signals because

$$\bar{\lambda} V_i^* V_i = (V_i^* R V_i)^* = \lambda V_i^* V_i \tag{3.76}$$

The eigenvalues are the real solutions of the characteristic equation

$$\det(R - \lambda I_N) = 0 \tag{3.77}$$

The identity matrix I_N has $+1$ as single eigenvalue with multiplicity N, and the co-identity matrix J_N has ± 1.

The relations between the zeros and coefficients of polynomials yield the following important results:

$$\det R = \prod_{i=0}^{N-1} \lambda_i \tag{3.78}$$

$$Nr(0) = N\sigma_x^2 = \sum_{i=0}^{N-1} \lambda_i \tag{3.79}$$

That is, if the determinant of the matrix is nonzero, each eigenvalue is nonzero and the sum of the eigenvalues is equal to N times the signal power. Furthermore, since the AC matrix is nonnegative definite, all the eigenvalues are nonnegative:

$$\lambda_i \geqslant 0, \quad 0 \leqslant i \leqslant N-1 \tag{3.80}$$

Once the eigenvalues have been found, the eigenvectors are obtained by solving equations (3.68). The eigenvectors associated with different eigenvalues of a symmetric matrix are orthogonal because of the equality

$$V_i^t V_j = \frac{1}{\lambda_i} V_i^t R V_j = \frac{\lambda_j}{\lambda_i} V_i^t V_j \tag{3.81}$$

When all the eigenvalues are distinct, the eigenvectors make an orthonormal base and the matrix can be diagonalized as

$$R = M^t \Lambda M \tag{3.82}$$

with M the $N \times N$ orthonormal modal matrix made of the N eigenvectors, and Λ the diagonal matrix of the eigenvalues; when they have a unit norm, the eigenvectors are denoted by U_i and:

$$\begin{aligned} M^t &= [U_0, U_1, \dots U_{N-1}]; \quad M^t = M^{-1} \\ \Lambda &= \operatorname{diag}(\lambda_0, \lambda_1, \dots, \lambda_{N-1}) \end{aligned} \tag{3.83}$$

For example, take a periodic signal $x(n)$ with period N. The AC function is also periodic with the same period and is symmetrical. The AC matrix is a circulant matrix, in which each row is derived from the preceding one by shifting. Now, if $|S(k)|^2$ denotes the signal power spectrum and T_N the discrete Fourier transform (DFT) matrix of order N:

$$T_N = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & w & \cdots & w^{N-1} \\ \vdots & \vdots & & \vdots \\ 1 & w^{N-1} & \cdots & w^{(N-1)(N^{-1})} \end{bmatrix}, \quad w = e^{-j2\pi/N} \tag{3.84}$$

it can be directly verified that

$$RT_N = T_N \operatorname{diag}(|S(k)|^2) \tag{3.85}$$

Due to the periodicity assumed for the AC function, the same is also true for the discrete cosine Fourier transform matrix, which is real and defined by

$$T_{cN} = \tfrac{1}{2}[T_N + T_N^*] \tag{3.86}$$

Thus

$$RT_{cN} = T_{cN} \operatorname{diag}(|S(k)|^2) \tag{3.87}$$

and the N column vectors of T_{cN} are the N orthogonal eigenvectors of the matrix R. Then

$$R = \frac{1}{N} T_{cN} \mathrm{diag}(|S(k)|^2) T_{cN} \tag{3.88}$$

So, it appears that the eigenvalues of the AC matrix of a periodic signal are the power spectrum; and the eigenvector matrix is the discrete cosine Fourier transform matrix.

However, the diagonalization of an AC matrix is not always unique. Let us assume that the N cisoids in the signal $x(n)$ have frequencies ω_i which are no longer multiples of $2\pi/N$:

$$x(n) = \sum_{i=1}^{N} S_i e^{jn\omega_i} \tag{3.89}$$

The ACF is

$$r(p) = \sum_{i=1}^{N} |S_i|^2 e^{jp\omega_i} \tag{3.90}$$

and the AC matrix can be expressed as

$$R = M^* \mathrm{diag}(|S_i|^2) M \tag{3.91}$$

with

$$M = \begin{bmatrix} 1 & e^{j\omega_1} & \cdots & e^{j(N-1)\omega_1} \\ 1 & e^{j\omega_2} & \cdots & e^{j(N-1)\omega_2} \\ \vdots & \vdots & & \vdots \\ 1 & e^{j\omega_N} & \cdots & e^{j(N-1)\omega_N} \end{bmatrix}$$

But the column vectors in M^* are neither orthogonal nor eigenvectors of R, as can be verified. If there are K cisoids with $K < N$, M becomes a $K \times N$ rectangular matrix and factorization (3.91) is still valid. But then the signal space dimension is restricted to the number of cisoids K, and $N - K$ eigenvalues are zero.

The white noise is a particularly simple case because $R = \sigma_e^2 I_N$ and all the eigenvalues are equal. If that noise is added to the useful signal, the matrix $\sigma_e^2 I_N$ is added to the AC matrix and all the eigenvalues are increased by σ_e^2.

Example

Consider the sinusoid in white noise

$$x(n) = \sqrt{2}\sin(n\omega) + e(n) \tag{3.92}$$

The AC function is

$$r(p) = \cos(p\omega) + \sigma_e^2 \delta(p) \tag{3.93}$$

The eigenvalues of the 3×3 AC matrix are

$$R = \begin{bmatrix} r(0) & r(1) & r(2) \\ r(1) & r(0) & r(1) \\ r(2) & r(1) & r(0) \end{bmatrix}, \qquad \begin{array}{l} \lambda_1 = \sigma_e^2 + 1 - \cos 2\omega \\ \lambda_2 = \sigma_e^2 + 2 + \cos 2\omega \\ \lambda_3 = \sigma_e^2 \end{array}$$

and the unit norm eigenvectors are

$$U_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, \quad U_2 = \frac{1}{(1 + 2\cos^2\omega)^{1/2}} \begin{bmatrix} \cos\omega \\ 1 \\ \cos\omega \end{bmatrix}, \tag{3.94}$$

$$U_3 = \frac{1}{(2 + 4\cos^2\omega)^{1/2}} \begin{bmatrix} 1 \\ -2\cos\omega \\ 1 \end{bmatrix}$$

The variations of the eigenvalues with frequency are shown in Figure 3.4.

Once a set of N orthogonal eigenvectors has been obtained, any signal vector $X(n)$ can be expressed as a linear combination of these vectors, which, when scaled to have a unit norm, are denoted by U_i:

$$X(n) = \sum_{i=0}^{N-1} \alpha_i(n) U_i \tag{3.95}$$

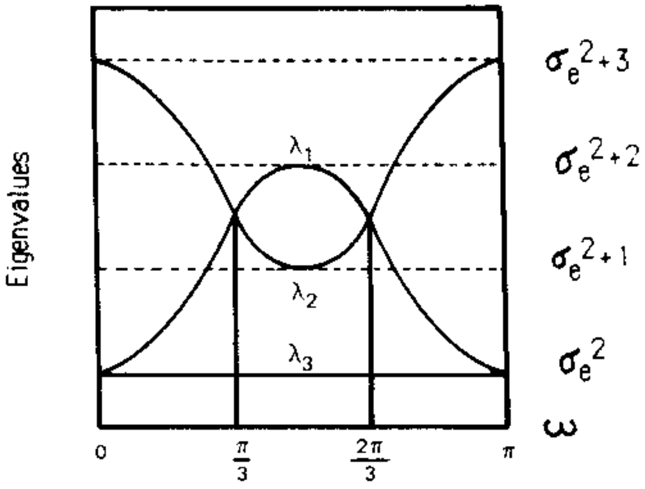


FIG. 3.4 Variation of eigenvalues with frequency.

The coefficients $a_i(n)$ are the projection of $X(n)$ on the vectors U_i. Another expression of the AC matrix can then be obtained, assuming real signals:

$$R = E[X(n)X^t(n)] = \sum_{i=0}^{N-1} E[\alpha_i^2(n)]U_iU_i^t \tag{3.96}$$

The definition of the eigenvalues yields

$$E[\alpha_i^2(n)] = \lambda_i \tag{3.97}$$

Equation (3.97) provides an important interpretation of the eigenvalues: they can be considered as the powers of the projections of the signal vectors on the eigenvectors. The subspace spanned by the eigenvectors corresponding to nonzero eigenvalues is called the signal subspace.

The eigenvalue or spectral decomposition is derived from (3.96):

$$R = \sum_{i=0}^{N-1} \lambda_i U_i U_i^t \tag{3.98}$$

which is just a more explicit form of diagonalization (3.82). It is a fundamental result which shows the actual constitution of the signal and is exploited in subsequent sections. For signals in noise, expression (3.98) can serve to separate signal subspace and noise subspace.

Among the eigenparameters the minimum and maximum eigenvalues have special properties.

3.7. EIGENFILTERS

The maximization of the signal-to-noise ratio (SNR) through FIR filtering leads to an eigenvalue problem [14].

The output power of an FIR filter is given in terms of the input AC matrix and filter coefficients by equation (3.60):

$$E[y^2(n)] = H^tRH$$

If a white noise with power σ_e^2 is added to the input signal, the output SNR is

$$SNR = \frac{H^tRH}{H^tH\sigma_e^2} \tag{3.99}$$

It is maximized by the coefficient vector H, which maximizes H^tRH, subject to the constraint $H^tH = 1$. Using a Lagrange multiplier, one has to maximize $H^tRH + \lambda(1 - H^tH)$ with respect to H, and the solution is $RH = \lambda H$. Therefore the optimum filter is the signal AC matrix eigenvector associated with the largest eigenvalue, and is called the maximum eigenfilter. Similarly, the minimum eigenfilter gives the smallest output signal power. These filters are characterized by their zeros in the complex plane.

The investigation of the eigenfilter properties begins with the case of distinct maximum or minimum eigenvalues; then it will be shown that the filter zeros are on the unit circle.

Let us assume that the smallest eigenvalue $\lambda_{\min}$ is zero. The corresponding eigenvector $U_{\min}$ is orthogonal to the other eigenvectors, which span the signal space. According to the harmonic decomposition of Section 3.11, the matrix R is the AC matrix of a set of $N-1$ cisoids, and the signal space is also spanned by $N-1$ vectors V_i:

$$V_i = \begin{bmatrix} 1 \\ e^{j\omega_i} \\ \vdots \\ e^{j(N-1)\omega_i} \end{bmatrix}, \quad 1 \leqslant i \leqslant N-1$$

Therefore $U_{\min}$ is orthogonal to all the vectors V_i, and the $N-1$ zeros of the corresponding filter are $e^{j\omega_i}(1 \leqslant i \leqslant N-1)$, and they are on the unit circle in the complex plane.

Now, if $\lambda_{\min}$ is not zero, the above development applies to the matrix $(R - \lambda_{\min} I_N)$, which has the same eigenvectors as R, as can be readily verified.

For the maximum eigenvector $U_{\max}$ corresponding to $\lambda_{\max}$, it is sufficient to consider the matrix $(\lambda_{\max} I_N - R)$, which has all the characteristics of an AC matrix. Thus the maximum eigenfilter also has its zeros on the unit circle in the z-plane as soon as $\lambda_{\max}$ is distinct.

The above properties can be checked for the example in the preceding section, which shows, in particular, that the zeros for $U_{\min}$ are $e^{\pm j\omega}$.

Next, if the minimum (or maximum) eigenvalue is multiple, for example $N-K$, it means that the dimension of the signal space is K and that of the noise space is $N-K$. The minimum eigenfilters, which are orthogonal to the signal space, have K zeros on the unit circle, but the remaining $N-1-K$ zeros may or may not be on the unit circle.

We give an example for two simple cases of sinusoidal signals in noise.

The AC matrix of a single cisoid, with power S^2, in noise is

$$R = \begin{bmatrix} S^2 + \sigma_e^2 & S^2 e^{j\omega} & \cdots & S^2 e^{j(N-1)\omega} \\ S^2 e^{-j\omega} & S^2 + \sigma_e^2 & \cdots & S^2 e^{j(N-2)\omega} \\ \vdots & \vdots & \ddots & \vdots \\ S^2 e^{-j(N-1)\omega} & S^2 e^{-j(N-2)\omega} & \cdots & S^2 + \sigma_e^2 \end{bmatrix} \tag{3.100}$$

The eigenvalues are

$$\lambda_1 = NS^2 + \sigma_e^2, \quad \lambda_i = \sigma_e^2, \quad 2 \leqslant i \leqslant N$$

and the maximum eigenfilter is

$$U_{\max} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 \\ e^{-j\omega} \\ \vdots \\ e^{-j(N-1)\omega} \end{bmatrix} \tag{3.101}$$

The corresponding filter z-transfer function is

$$H_M(z) = \frac{1}{\sqrt{N}} \frac{z^N - e^{jN\omega}}{z - e^{j\omega}} e^{-j(N-1)\omega} \tag{3.102}$$

and the $N-1$ roots

$$z_i = e^{j(\omega + 2\pi i/N)}, \quad 1 \leqslant i \leqslant N-1$$

are spread on the unit circle, except at the frequency ω. $H_M(z)$ is the conventional matched filter for a sine wave in noise.

Because the minimum eigenvalue is multiple, the unnormalized eigenvector $V_{\min}$ is

$$V_{\min} = \begin{bmatrix} -\sum_{i=2}^{N} v_i e^{j(i-1)\omega} \\ v_2 \\ \vdots \\ v_N \end{bmatrix} \tag{3.103}$$

where $N-1$ arbitrary scalars v_i are introduced.

Obviously there are $N-1$ linearly independent minimum eigenvectors which span the noise subspace. The associated filter z-transfer function is

$$H_m(z) = (z - e^{j\omega}) \sum_{i=2}^{N} v_i [z^{i-2} + z^{i-3} e^{j\omega} + \cdots + e^{j(i-2)\omega}] \tag{3.104}$$

One zero is at the cisoid frequency on the unit circle; the others may or may not be on that circle.

The case of two cisoids, with powers S_1^2 and S_2^2 in noise leads to more complicated calculations. The correlation matrix

$$R = \begin{bmatrix} S_1^2 + S_2^2 + \sigma_e^2 & S_1^2 e^{j\omega_1} + S_2^2 e^{j\omega_2} & \vdots & S_1^2 e^{j(N-1)\omega_1} + S_2^2 e^{j(N-1)\omega_2} \\ S_1^2 e^{-j\omega_1} + S_2^2 e^{-j\omega_2} & S_1^2 + S_2^2 + \sigma_e^2 & \vdots & S_1^2 e^{j(N-2)\omega_1} + S_2^2 e^{j(N-2)\omega_2} \\ \vdots & \vdots & \ddots & \vdots \\ S_1^2 e^{-j(N-1)\omega_1} + S_2^2 e^{-j(N-1)\omega_2} & S_1^2 e^{-j(N-2)\omega_1} + S_2^2 e^{-j(N-2)\omega_2} & \cdots & S_1^2 + S_2^2 + \sigma_e^2 \end{bmatrix}$$

has eigenvalues [15]

$$\lambda_1 = \sigma_e^2 + \frac{N}{2}[S_1^2 + S_2^2] + \sqrt{\frac{N^2}{4}(S_1^2 - S_2^2)^2 + N^2 S_1^2 S_2^2 F^2(\omega_1 - \omega_2)}$$

$$\lambda_2 = \sigma_e^2 + \frac{N}{2}[S_1^2 + S_1^2] + \sqrt{\frac{N^2}{4}(S_1^2 - S_2^2)^2 + N^2 S_1^2 S_2^2 F^2(\omega_1 - \omega_2)}$$

$$\lambda_i = \sigma_e^2, \quad 3 \leqslant i \leqslant N \tag{3.105}$$

$F(\omega)$ is the familiar function

$$F(\omega) = \frac{\sin(N\omega/2)}{N\sin(\omega/2)} \tag{3.106}$$

These results, when applied to a sinusoid amplitude A, $x(n) = A\sin(n\omega)$, yield

$$\lambda_{1,2} = \sigma_e^2 + \frac{A^2}{4}\left(N \pm \frac{\sin(N\omega)}{\sin\omega}\right) \tag{3.107}$$

The extent to which λ_1 and λ_2 reflect the powers of the two cisoids depends on their respective frequencies, through the function $F(\omega)$, which corresponds to a length-N rectangular time wnidow. For N large and frequencies far apart enough,

$$F(\omega_1 - \omega_2) \approx 0, \quad \lambda_1 = NS_1^2 + \sigma_e^2, \quad \lambda_2 = NS_2^2 + \sigma_e^2 \tag{3.108}$$

and the largest eigenvalues represent the cisoid powers.

The z-transfer function of the minimum eigenfilters is

$$H_m(z) = (z - e^{j\omega_1})(z - e^{j\omega_2})P(z) \tag{3.109}$$

with $P(z)$ a polynomial of degree less than $N - 2$. Two zeros are on the unit circle at the cisoid frequencies; the other zeros may or may not be on that circle.

To conclude: for a given signal the maximum eigenfilter indicates where the power is in the frequency domain, and the zeros of the minimum eigenvalue filter give the exact frequencies associated with the harmonic decomposition of that signal.

Together, the maximum and minimum eigenfilters constitute a powerful tool for signal analysis. However, in practice, the appeal of that technique is somewhat moderated by the computation load needed to extract the eigenparameters, which becomes enormous for large matrix dimensions. Savings can be obtained by careful exploitation of the properties of AC matrices [16]. For example, the persymmetry relation (3.64) yields, for any eigenvector V_i,

$$J_N R V_i = \lambda_i J_N V_i = R J_N V_i$$

Now, if λ_i is a distinct eigenvalue, the vectors V_i and $J_N V_i$ are colinear, which means that V_i is also an eigenvector of the co-identity matrix J_N, whose eigenvalues are ± 1. Hence the relation

$$J_N V_i = \pm V_i \tag{3.110}$$

holds.

The corresponding property of the AC matrix can be stated as follows: the eigenvectors associated with distinct eigenvalues are either symmetric or skew symmetric; that is, they verify (3.110).

Iterative techniques help manage the computation load. Before presenting such techniques, we give additional properties of extremal eigenvalues.

3.8. PROPERTIES OF EXTREMAL EIGENVALUES

In the design process of an adaptive filter it is sometimes enough to have simple evaluations of the extremal eigenvalues $\lambda_{\max}$ and $\lambda_{\min}$. A loose bound for the maximum eigenvalue of an AC matrix, derived from (3.79), is

$$\lambda_{\max} \leqslant N\sigma_x^2 \tag{3.111}$$

with σ_x^2 the signal power and $N \times N$ the matrix dimension. A tighter bound, valid for any square matrix R with entries r_{ij}, is known from matrix theory to be

$$\lambda_{\max} \leqslant \max_j \sum_{i=0}^{N-1} |r_{ij}| \tag{3.112}$$

or

$$\lambda_{\max} \leqslant \max_i \sum_{j=0}^{N-1} |r_{ij}|$$

To prove the inequality, single out the entry with largest magnitude in the eigenvector $V_{\max}$ and bound the elements of the vector $RV_{\max}$.

In matrix theory, $\lambda_{\max}$ is called the spectral radius. It serves as a matrix norm as well as the right side of (3.112).

The Rayleigh quotient of R is defined by

$$R_a(V) = \frac{V^t R V}{V^t V}, \quad V \neq 0 \tag{3.113}$$

As shown in the preceding section,

$$\lambda_{\max} = \max_V R_a(V) \tag{3.114}$$

The diagonalization of R yields

$$R = M^{-1}\operatorname{diag}(\lambda_i)M \tag{3.115}$$

It is readily verified that

$$R^{-1} = M^{-1}\operatorname{diag}\left(\frac{1}{\lambda_i}\right)M \tag{3.116}$$

Therefore $\lambda_{\min}^{-1}$ is the maximum eigenvalue of R^{-1}. The condition number of R is defined by

$$\operatorname{cond}(R) = \|R\| \;\; \|R^{-1}\| \tag{3.117}$$

If the matrix norm $\|R\|$ is $\lambda_{\max}$, then

$$\operatorname{cond}(R) = \frac{\lambda_{\max}}{\lambda_{\min}} \tag{3.118}$$

The condition number is a matrix parameter which impacts the accuracy of the operations, particularly inversion [9]. It is crucial in solving linear systems, and it is directly related to some stability conditions in LS adaptive filters.

In adaptive filters, sequences of AC matrices with increasing dimensions are sometimes encountered, and it is useful to know how the extremal eigenvalues vary with matrix dimensions for a given signal. Let us denote by $U_{\max,N}$ the maximum unit-norm eigenvector of the $N \times N$ AC matrix R_N. The maximum eigenvalue is

$$\lambda_{\max,N} = U_{\max,N}^t R_N U_{\max,N} \tag{3.119}$$

Now, because of the structure of the $(N+1) \times (N+1)$ AC matrix, the following equation is valid:

$$\lambda_{\max,N} = [U_{\max,N}^t, 0]$$
$$\times \left[\begin{array}{cccc:c} r(0) & r(1) & \cdots & r(N-1) & r(N) \\ r(1) & r(0) & \cdots & r(N-2) & r(N-1) \\ \vdots & \vdots & R_N & \vdots & \vdots \\ r(N-1) & r(N-2) & \cdots & r(0) & r(1) \\ \hdashline r(N) & r(N-1) & \cdots & r(1) & r(0) \end{array}\right] \left[\begin{array}{c} U_{\max,N} \\ \hdashline 0 \end{array}\right] \tag{3.120}$$

At the dimension $N+1$, $\lambda_{\max,N+1}$ is defined as the maximum of the product $U_{N+1}^t R_{N+1} U_{N+1}$ for any unit-norm vector U_{N+1}. The vector obtained by appending a zero to $U_{\max,N}$ is such a vector, and the following inequality is proven:

$$\lambda_{\max,N} \leqslant \lambda_{\max,N+1} \tag{3.121}$$

Also, considering the minimization procedure, we have

$$\lambda_{\min,N} \geqslant \lambda_{\min,N+1} \tag{3.122}$$

When N approaches infinity, $\lambda_{\max}$ and $\lambda_{\min}$ approach the maximum and the minimum, respectively, of the signal power spectrum, as shown in the next section.

3.9. SIGNAL SPECTRUM AND EIGENVALUES

According to relation (3.79), the eigenvalue extraction can be viewed as an energy decomposition of the signal. In order to make comparisons with the spectrum, we choose the following definition for the Fourier transform $Y(f)$ of the signal $x(n)$:

$$Y(f) = \lim_{n\to\infty} \frac{1}{\sqrt{2N+1}} \sum_{-N}^{N} x(n)e^{-j2\pi fn} \tag{3.123}$$

The spectrum is the square of the modulus of $Y(f)$:

$$S(f) = Y(f)\bar{Y}(f) = |Y(f)|^2 \tag{3.124}$$

When the summations in the above definition of $S(f)$ are rearranged, the correlation function $r(p)$ shows up, and the following expression is obtained:

$$S(f) = \sum_{p=-\infty}^{\infty} r(p)e^{-j2\pi fp} \tag{3.125}$$

Equation (3.125) is appropriate for random signals with statistics that are known or that can be measured or estimated.

Conversely, the spectrum $S(f)$ is a periodic function whose period is the reciprocal of the sampling frequency, and the correlation coefficients are the coefficients of the Fourier series expansion of $S(f)$:

$$r(p) = \int_{-1/2}^{1/2} S(f)e^{j2\pi pf}\,df \tag{3.126}$$

In practice, signals are time limited, and often a finite-duration record of N_0 data representing a single realization of the process is available. Then it is sufficient to compute the spectrum at frequencies which are integer multiples of $1/N_0$, since intermediate values can be interpolated, and the DFT with appropriate scaling factor

$$Y(k) = \frac{1}{\sqrt{N_0}} \sum_{n=0}^{N_0-1} x(n) e^{-j(2\pi/N)nk} \tag{3.127}$$

is employed to complete that task. The operation is equivalent to making the signal periodic with period N_0; the corresponding AC function is also periodic, with the same period, and the eigenvalues of the AC matrix are $|Y(k)|^2, 0 \leqslant k \leqslant N_0 - 1$.

Now, the N eigenvalues λ_i of the $N \times N$ AC matrix R_N and their associated eigenvectors V_i are related by

$$\lambda_i V_i^* V_i = V_i^* R_N V_i \tag{3.128}$$

The right side is the power of the output of the eigenfilter; it can be expressed in terms of the frequency response by

$$V_i^* R_N V_i = \int_{-1/2}^{1/2} |H_i(f)|^2 S(f) df \tag{3.129}$$

The left side of (3.115) can be treated similarly, which leads to

$$\min_{-1/2 \leqslant f \leqslant 1/2} S(f) \leqslant \lambda_i \leqslant \max_{-1/2 \leqslant f \leqslant 1/2} S(f) \tag{3.130}$$

It is also interesting to relate the eigenvalues of the order N AC matrix to the DFT of a set of N data, which is easily obtained and familiar to practitioners. If we denote the set of N data by the vector X_N, the DFT, expressed by the matrix T_N (3.84), yields the vector Y_N:

$$Y_N = \frac{1}{\sqrt{N}} T_N X_N$$

The energy conservation relation is verified by taking the Euclidean norm of the complex vector Y_N:

$$\| Y_N \|^2 = Y_N^* Y_N = X_N^* X_N$$

Or, explicitly, we can write

$$\sum_{k=0}^{N-1} |Y(k)|^2 = \sum_{n=0}^{N-1} |x(n)|^2$$

The covariance matrix of the DFT output is

$$E[Y_N Y_N^*] = \frac{1}{N} T_N R T_N \tag{3.131}$$

The entries of the main diagonal are

$$E[|Y(k)|^2] = \frac{1}{N} V_k^* R_N V_k \tag{3.132}$$

with

$$V_k^* = [1, e^{j2\pi/N}, \ldots, e^{j(2\pi/N)(N-1)}]$$

From the properties of the eigenvalues, the following inequalities are derived:

$$\lambda_{\max} \geqslant \max_{0 \leqslant f \leqslant N-1} E[|Y(k)|^2]$$

$$\lambda_{\min} \geqslant \max_{0 \leqslant k \leqslant N-1} E[|Y(k)|^2] \tag{3.133}$$

These relations state that the DFT is a filtering operation and the output signal power is bounded by the extreme eigenvalues.

When the data vector length N approaches infinity, the DFT provides the exact spectrum, and, due to relations (3.130) and (3.133), the extreme eigenvalues $\lambda_{\min}$ and $\lambda_{\max}$ approach the extreme values of the signal spectrum [17].

3.10. ITERATIVE DETERMINATION OF EXTREMAL EIGENPARAMETERS

The eigenvalues and eigenvectors of an AC matrix can be computed by classical algebraic methods [9]. However, the computation load can be enormous, and it is useful to have simple and efficient methods to derive the extremal eigenparameters, particularly if real-time operation is envisaged.

A first, gradient-type approach is the unit-norm constrained algorithm [18]. It is based on minimization or maximization of the output power of a filter with coefficient vector $H(n)$, as shown in Figure 3.5, using the eigenfilter properties presented in Section 3.7. The output of the unit-norm filter is

$$e(n) = \frac{H^t(n)X(n)}{[H^t(n)H(n)]^{1/2}} \tag{3.134}$$

The gradient of $e(n)$ with respect to $H(n)$ is the vector

$$\nabla e(n) = \frac{1}{[H^t(n)H(n)]^{1/2}} \left[X(n) - e(n) \frac{H(n)}{[H^t(n)H(n)]^{1/2}} \right] \tag{3.135}$$

Now, the power of the sequence $e(k)$ is minimized if the coefficient vector at time $n+1$ is taken as

$$H(n+1) = H(n) - \delta e(n) \nabla e(n) \tag{3.136}$$

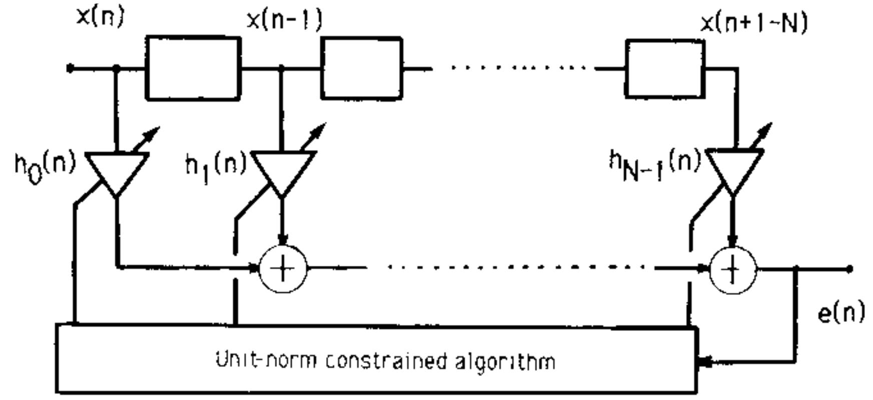


FIG. 3.5 Unit-norm constrained adaptive filter.

where δ, the adaptation step size, is a positive constant. After normalization, the unit-norm filter coefficient vector is

$$\frac{H(n+1)}{\|H(n+1)\|} = \frac{1}{\|H(n+1)\|}\left[H(n) - \frac{\delta e(n)}{\|H(n)\|}\left(X(n) - e(n)\frac{H(n)}{\|H(n)\|}\right)\right] \tag{3.137}$$

with

$$\|H(n)\| = [H^t(n)H(n)]^{1/2}$$

In the implementation, the expression contained in the brackets is computed first and the resulting coefficient vector is then normalized to unit norm. In that way there is no roundoff error propagation. The gradient-type approach leads to the eigenequation, as can be verified by rewriting equation (3.136):

$$H(n+1) = H(n) - \frac{\delta}{\|H(n)\|}\left[X(n)X^t(n)\frac{H(n)}{\|H(n)\|} - e^2(n)\frac{H(n)}{\|H(n)\|}\right] \tag{3.138}$$

Taking the expectation of both sides, after convergence, yields

$$R\frac{H(\infty)}{\|H(\infty)\|} = E[e^2(n)]\frac{H(\infty)}{\|H(\infty)\|} \tag{3.139}$$

The output signal power is the minimum eigenvalue, and $H(\infty)$ is the corresponding eigenvector. Changing the sign in equation (3.136) leads to the maximum eigenvalue instead.

The step size δ controls the adaptation process. Its impact is analyzed indepth in the next chapter.

Faster convergence can be obtained by minimizing the conventional cost function

$$J(n) = \sum_{p=1}^{n} W^{n-p} e^2(p), \quad 0 \ll W \leqslant 1 \tag{3.140}$$

using a recursive LS algorithm [19]. The improvement in speed and accuracy is paid for by a significant increase in computation load. Furthermore, because of approximations made in the derivation, an initial guess for the coefficient vector sufficiently close to the exact solution is needed to achieve convergence. In contrast, a method based on the conjugate gradient technique converges for any initial guess in approximately M steps, where M is the number of independent eigenvalues of the AC matrix [20].

The method assumes that the AC matrix R is known, and it begins with an initial guess of the minimum eigenvector $H_{\min}(0)$ and with an initial direction vector. The minimum eigenvalue is computed as $U^t_{\min}(0)RU_{\min}(0)$, and then successive approximations $U_{\min}(k)$ are developed to minimize the cost function U^tRU in successive directions, which are R-conjugates, until the desired minimum eigenvalue is found.

The FORTRAN subroutine is given in Annex 3.2.

3.11. ESTIMATION OF THE AC MATRIX

The AC matrix can be formed with the estimated values of the AC function. The bias and variance of the estimators impact the eigenparameters. The bias can be viewed as a modification of the signal. For example, windowing effects, as in (3.21), smear the signal spectrum and increase the dimension of the signal subspace, giving rise to spurious eigenvalues [21]. The effects of the estimator variance can be investigated by considering small random perturbations on the elements of the AC matrix. In adaptive filters using the AC matrix, explicitly or implicitly as in fast least squares (FLS) algorithms, random perturbations come from roundoff errors and can affect, more or less independently, all the matrix entries.

Let us assume that the matrix R has all its eigenvalues distinct and is affected by a small perturbation matrix ΔR. The eigenvalues and vectors are explicit functions of the matrix elements, and their alteration can be developed in series; considering only the first term in the series, the eigenvalue equation with unit-norm vectors is

$$(R + \Delta R)(U_i + \Delta U_i) = (\lambda_i + \Delta\lambda_i)(U_i + \Delta U_i), \quad 0 \leqslant i \leqslant N-1 \tag{3.141}$$

Neglecting the second-order terms and premultiplying by U_i^t yields

$$\Delta\lambda_i = U_i^r \Delta R U_i \tag{3.142}$$

Due to the summing operation in the right side, the perturbation of the eigenvalue is very small, if the error matrix elements are i.i.d. random variables.

In order to investigate the eigenvector deviation, we introduce the normalized error matrix ΔE, associated with the diagonalization (3.82) of the matrix R:

$$\Delta E = \Lambda^{1/2} M \Delta R M^t \Lambda^{-1/2} \tag{3.143}$$

We can write (3.141), without the second-order terms and taking (3.142) into account,

$$(R - \lambda_i I_N)\Delta U_i = (U_i U_i^t - I_N)\Delta R U_i \tag{3.144}$$

After some algebraic manipulations, we get

$$\Delta U_i = \sum_{\substack{k=0 \\ k\neq 1}}^{N-1} \frac{\sqrt{\lambda_i \lambda_k}}{\lambda_i - \lambda_k} \Delta E(k,i) U_k \tag{3.145}$$

where the $\Delta E(k,i)$ are the elements of the normalized error matrix.

Clearly, the deviation of the unit-norm eigenvectors U_i depends on the spread of the eigenvalues, and large deviations can be expected to affect eigenvectors corresponding to close eigenvalues [22].

Overall, the bias of the AC function estimator affects the AC matrix eigenvalues, and the variance of errors on the AC matrix elements affects the eigenvector directions.

In recursive algorithms, the following estimation appears:

$$R_N(n) = \sum_{p=1}^{n} W^{n-p} X(n) X^t(n) \tag{3.146}$$

where W is a weighting factor ($0 \ll W \leqslant 1$) and $X(n)$ is the vector of the N most recent data. In explicit form, assuming $X(0) = 0$, we can write

$$R_N(n) = \begin{bmatrix} \sum_{i=1}^{n} W^{n-i} x^2(i) & \sum_{i=2}^{n} W^{n-i} x(i)x(i-1) & \cdots & \sum_{i=N}^{n} W^{n-i} x(i)x(i-N+1) \\ \sum_{i=2}^{n} W^{n-i} x(i-1)x(i) & \sum_{i=2}^{n} W^{n-i} x^2(i-1) & \cdots & \vdots \\ \vdots & \vdots & \ddots & \\ \sum_{i=N}^{n} W^{n-i} x(i)x(i-N+1) & \cdots & \cdots & \sum_{i=N}^{n} W^{n-i} x^2(i-N+1) \end{bmatrix} \tag{3.147}$$

The matrix is symmetric. For large n it is almost doubly symmetric. Its expectation is

$$E(R_N(n)] = \frac{1}{1-W}$$

$$\times \begin{bmatrix} (1-W^n)r(0) & (1-W^{n-1})r(1) & \cdots & (1-W)^{n-N-1}r(N-1) \\ (1-W^{n-1}r(1) & (1-W^{n-1}r(0) & \cdots & \vdots \\ \vdots & \vdots & \ddots & \\ (1-W^{n-N+1})r(N-1) & \cdots & \cdots & (1-W^{n-N+1})r(0) \end{bmatrix} \tag{3.148}$$

For large n

$$E[R_N(n)] \approx \frac{1}{1-W} R \tag{3.149}$$

In these conditions, the eigenvectors of $R_N(n)$ are those of R, and the eigenvalues are multiplied by $(1-W)^{-1}$.

Example

$$x(n) = \sin\left(n\frac{\pi}{4}\right), \qquad n > 0$$
$$x(n) = 0, \qquad n \leqslant 0$$

The eigenvalues of the 8×8 AC matrix can be found from (3.105) in which

$$S_1 = S_2 = \frac{1}{2}, \quad \omega_1 - \omega_2 = \frac{\pi}{2}$$

so that the term in the square root vanishes. Expression (3.107) can be used as well, with $A = 1$:

$$\lambda_1 = \lambda_2 = 2, \quad \lambda_3 = \cdots = \lambda_8 = 0$$

The eigenvalues of the matrix $R'(n)$

$$R'(n) = \frac{R_N(n)}{2\sum_{i=1}^{n} W^{n-i}x^2(i)}, \qquad W = 0.95$$

are shown in Figure 3.6 for the first values of n. They approach the theoretical values as n increases.

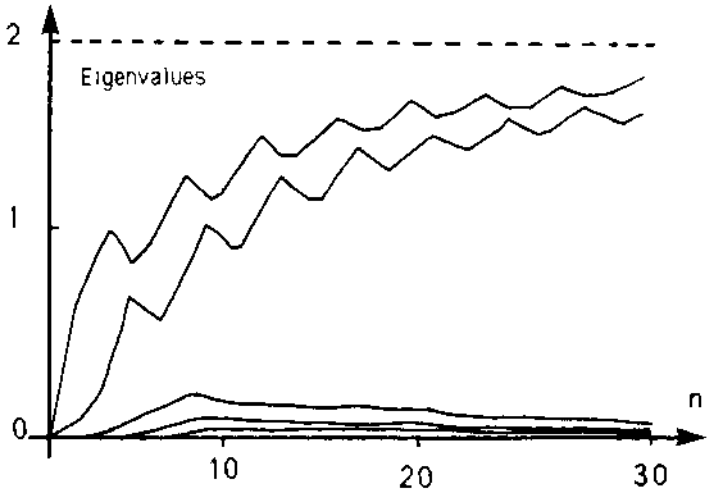


FIG. 3.6 Eigenvalues of the matrix $R'(n)$.

3.12. EIGEN (KL) TRANSFORM AND APPROXIMATIONS

The projections of a signal vector X on the eigenvectors of the AC matrix form a vector

$$[\alpha] = M^t X \tag{3.150}$$

where M is the $N \times N$ orthonormal modal matrix defined in Section 3.6. The transform is unitary ($M^t M = I_N$) and called the Karhunen-Loève (KL) transform. It is optimal for the class of all signals having the same second-order statistics [23]. Optimality means the efficiency of a transform in achieving data compression: the KL transform provides the optimum sets of data to represent signal vectors within a specified mean square error. For example, if M out of the N eigenvalues are zero or negligible, the N element data vectors can be represented by $N - M$ numbers only.

To prove that property we assume that the elements of the vector X are N centered random variables and look for the unitary transform I which best compresses the N elements of X into $M(M < N)$ elements out of the N elements y_i of the vector Y given by

$$Y = TX$$

The mean square error is

$$\text{MSE} = \sum_{i=M+1}^{N} E(y_i^2)$$

If the new vectors of T are designated by V_{Ti}^t then

$$\text{MSE} = \sum_{i=M+1}^{N} V_{Ti}^t E[XX^t] V_{Ti}$$

The minimization of the above expression under the contraint of unit norm vectors, using Lagrange multipliers, leads to:

$$E[XX^t]V_{Ti} = \lambda_i V_{Ti}, \quad M+1 \leqslant i \leqslant N$$

The minimum is obtained if the scalars λ_i are the $N-M$ smallest eigenvalues of the matrix $E[XX^t]$ and V_{Ti} the corresponding unit norm eigenvectors. The minimum mean square error is

$$(\text{MSE})_{\min} = \sum_{i=M+1}^{N} \lambda_i$$

and, in fact, referring to Section 3.6, it is the amount of signal energy which is lost in the compression process.

However, compared with other unitary transforms like the DFT, the KL transform suffers from several drawbacks in practice. First, it has to be adjusted when the signal second-order statistics change. Second, as seen in the preceding sections, it requires a computation load proportional to N^2. Therefore it is helpful to find approximations which are sufficiently close for some signal classes and amenable to easy calculation through fast algorithms. Such approximations can be found for the first-ordr AR signal.

Because of the dual diagonalization relation

$$R^{-1} = M^t \Lambda^{-1} M \tag{3.151}$$

the KL transform coefficients can be found from the inverse AC matrix as well. For the first-order unity-variance AR signal, the AC matrix is given by (3.67). The inverse (3.68) is a tridiagonal matrix, and the elements of the KL transform for N even are [24]

$$m_{kn} = c_n \sin\left[\omega_n \left(k - \frac{N+1}{2}\right) + n\frac{\pi}{2}\right] \tag{3.152}$$

where c_n are normalization constants and ω_n are the positive roots of

$$\tan(N\omega) = -\frac{(1-a^2)\sin\omega}{\cos\omega - 2a + a^2\cos\omega} \tag{3.153}$$

The eigenvalues of R are

$$\lambda_i = \frac{1 - a^2}{(1 - 2a\cos\omega_i + a^2)^{1/2}}, \quad 1 \leqslant i \leqslant N \tag{3.154}$$

Now, the elements of the KL transform of a data vector are

$$\alpha_k = \sum_{n=1}^{N} c_n x(n) \sin\left[\omega_n\left(k - \frac{N+1}{2}\right) + n\frac{\pi}{2}\right] \tag{3.155}$$

Due to the nonharmonicity of sine terms, a fast algorithm is unavailable in calculating the above expressions, and N^2 computations are required. However, if R^{-1} is replaced by

$$R' = \frac{1}{1 - a^2}\begin{bmatrix} 1+a^2 & -a & 0 & \cdots & 0 \\ -a & 1+a^2 & -a & \cdots & 0 \\ 0 & -a & 1+a^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & -a \\ 0 & 0 & 0 & -a & 1+a^2 \end{bmatrix} \tag{3.156}$$

where R' differs by just the first and last entries in the main diagonal, the elements of the modal matrix become

$$m'_{kn} = \sqrt{\frac{2}{N+1}}\sin\left(\frac{kn\pi}{N+1}\right) \tag{3.157}$$

and the eigenvalues are

$$\lambda'_i = 1 - 2\frac{a}{1+a^2}\cos\left(\frac{i\pi}{N+1}\right), \quad i = 1, \ldots, N \tag{3.158}$$

The elements of the corresponding transform of a data vector are

$$\alpha'_k = \sqrt{\frac{2}{N+1}}\sum_{n=1}^{N} x(n)\sin\left(\frac{nk\pi}{N+1}\right) \tag{3.159}$$

This defines the discrete sine transform (DST), which can be implemented via a fast Fourier transform (FFT) algorithm.

Finally, for an order 1 AR signal, the DST is an efficient approximation of the KL tranform.

Another approximation is the discrete cosine transform (DCT), defined as

$$\alpha''_0 = \frac{\sqrt{2}}{N}\sum_{n=1}^{N} x(n)$$

$$\alpha_k'' = \frac{2}{N}\sum_{n=1}^{N} x(n)\cos\frac{(2n-1)k\pi}{2N}, \quad 1 \leqslant k \leqslant N-1 \tag{3.160}$$

It can be extended to two dimensions and is widely used in image processing [25].

3.13. SUMMARY

Estimating the ACF is often a preliminary step in signal analysis. After definition and basic properties have been introduced, efficient estimation techniques have been compared.

The AC matrix is behind adaptive filtering operations, and it is essential to be familiar with its major characteristics, which have been presented and illustrated by several simple examples. The eigenvalue decomposition has a profound meaning, because it leads to distinguishing between the signal or source space and the noise space, and to extracting the basic components. The filtering aspects help to understand and assess the main properties of eigenvalues and vectors. The extremal eigenparameters are especially crucial not only for the theory but also because they control adaptive filter performance and because they can provide superresolution analysis techniques.

Perturbations of the matrix elements, caused by bias and variance in the estimation process, affect the processing performance and particularly the operation of FLS algorithms. It has been shown that the bias can affect the eigenvalues and the variance causes deviations of eigenvectors. The KL transform is an illustrative application of the theoretical results.

EXERCISES

1. Use the estimators $r_1(p)$ and $r_2(p)$ to calculate the ACF of the sequence

$$x(n) = \sin\left(n\frac{\pi}{5}\right), \quad 0 \leqslant n \leqslant 15$$

   How are the deviations from theoretical values affected by the signal frequency?
2. For the symmetric matrix

$$R = \begin{bmatrix} 1.1 & -0.6 & 0.2 \\ -0.6 & 1.0 & -0.4 \\ 0.2 & -0.4 & 0.6 \end{bmatrix}$$

   calculate R^2 and R^3 and the first element $r_{00}^{(4)}$ of the main diagonal of R^4. Compare the ratio $r_{00}^{(4)}/r_{00}^{(3)}$ with the largest eigenvalue $\lambda_{\max}$.

Show that the following approximation is valid for a symmetric matrix R and N sufficiently large:

$$\left(\frac{R}{\lambda_{\max}}\right)^{N+1} \approx \left(\frac{R}{\lambda_{\max}}\right)^{N}$$

This expression can be used for the numerical calculation of the extremal eigenvalues.

3. For the AC matrix

$$R = \begin{bmatrix} 1.0 & 0.7 & 0.0 \\ 0.7 & 1.0 & 0.7 \\ 0.0 & 0.7 & 1.0 \end{bmatrix}$$

calculate its eigenvalues and eigenvectors and check the properties given in Section 3.6. Verify the spectral decomposition (3.98).

4. Find the frequency and amplitude of the sinusoid contained in the signal with AC matrix

$$R = \begin{bmatrix} 1.00 & 0.65 & 0.10 \\ 0.65 & 1.00 & 0.65 \\ 0.10 & 0.65 & 1.00 \end{bmatrix}$$

What is the noise power? Check the results with the curves in Figure 3.4.

5. Find the spectral decomposition of the matrix

$$R = \begin{bmatrix} 1.0 & 0.7 & 0.0 & -0.7 \\ 0.7 & 1.0 & 0.7 & 0.0 \\ 0.0 & 0.7 & 1.0 & 0.7 \\ -0.7 & 0.0 & 0.7 & 1.0 \end{bmatrix}$$

What is the dimension of the signal space? Calculate the projections of the vectors

$$X^t(n) = \left[\cos\left(n\frac{\pi}{4}\right), \cos\left[(n-1)\frac{\pi}{4}\right], \cos\left[(n-2)\frac{\pi}{4}\right], \right.$$
$$\left. \times \cos\left[(n-3)\frac{\pi}{4}\right]\right], \qquad n = 0, 1, 2, 3$$

on the eigenvectors.

6. Consider the order 2 AR signal

$$x(n) = 0.9x(n-1) - 0.5x(n-2) + e(n)$$

with $E[e^2(n)] = \sigma_e^2 = 1$. Calculate its ACF and give its 3×3 AC matrix R_3. Find the minimum eigenvalue and eigenvector. Give the corresponding harmonic decomposition of the signal and compare with the spectrum.

Calculate the 4×4 matrix R_4 and its inverse R_4^{-1}. Comment on the results.

7. Give expressions to calculate the DST (3.159) and the DCT by a standard DFT. Estimate the computational complexity for $N = 2^p$.

ANNEX 3.1 FORTRAN SUBROUTINE TO SOLVE A LINEAR SYSTEM WITH SYMMETRICAL MATRIX

```
      SUBROUTINE CHOL(N,A,X,B)
C
C     SOLVES THE SYSTEM [A]X=B
C     A : SYMMETRIC COVARIANCE MATRIX (N*N)
C     N : SYSTEM ORDER (N > 2)
C     X : SOLUTION VECTOR
C     B : RIGHT SIDE VECTOR

      DIMENSION A(20,20),X(1),B(1)
      A(2,1)=A(2,1)/A(1,1)
      A(2,2)=A(2,2)-A(2,1)*A(1,1)*A(2,1)
      D040I=3,N
      A(I,1)=A(I,1)/A(1,1)
      D020J=2,I-1
      S=A(I,J)
      D010K=1,J-1
   10 S=S-A(I,K)*A(K,K)*A(J,K)
   20 A(I,J)=S/A(J,J)
      S=A(I,I)
      D030K=1,I-1
   30 S=S-A(I,K)*A(K,K)*A(I,K)
   40 A(I,I)=S
      X(1)=B(1)
      D060I=2,N
      S=B(I)
      D050J=1,I-1
   50 S=S-A(I,J)*X(J)
   60 X(I)=S
      X(N)=X(N)/A(N,N)
      D080K=1,N-1
      I=N-K
      S=X(I)/A(I,I)
      D070J=I+1,N
   70 S=S-A(J,I)*X(J)
   80   X(I)=S
        RETURN
        END
C
```

ANNEX 3.2 FORTRAN SUBROUTINE TO COMPUTE THE EIGENVECTOR CORRESPONDING TO THE MINIMUM EIGENVALUE BY THE CONJUGATE GRADIENT METHOD [20]

(*Courtesy of Tapan K. Sarkar, Department of Electrical Engineering, Syracuse University, Syracuse, N.Y. 13244-1240*)

```
        SUBROUTINE GMEVCG(N, X, A, B, U, SML, W, M)
C
C       THIS SUBROUTINE IS USED FOR ITERATIVELY FINDING THE
C       EIGENVECTOR CORRESPONDING TO THE MINIMUM EIGENVALUE
C       OF A GENERALIZED EIGENSYSTEM AX = UBX.
C
C  A      - INPUT REAL SYMMETRIC MATRIX OF ORDER N, WHOSE
C           MINIMUM EIGENVALUE AND THE CORRESPONDING
C           EIGENVECTOR ARE TO BE COMPUTED.
C  B      - INPUT REAL POSITIVE DEFINITE MATRIX OF ORDER N.
C  N      - INPUT ORDER OF THE MATRIX A.
C  X      - OUTPUT EIGENVECTOR OF LENGTH  N CORRESPONDING TO
C           THE MINIMUM EIGENVALUE AND ALSO PUT INPUT
C           INITIAL GUESS IN IT.
C  U      - OUTPUT MINIMUM EIGENVALUE.
C  SML    - INPUT UPPER BOUND OF THE MINIMUM EIGENVALUE.
C  W      - INPUT ARBITRARY VECTOR OF LENGTH N.
C  M      - OUTPUT NUMBER OF ITERATIONS.
C
        LOGICAL AAEZ, BBEZ
        REAL A(N,N), B(N,N), X(N), P(5), R(5), W(N), AP(5),
   *    BP(5), AX(5), BX(5)
        NU = 0
        M = 0
        U1 = 0.0
   1    DO 20 I=1,N
          BX(I) = 0.0
          DO 10 J=1,N
            BX(I) = BX(I) + B(I,J)*X(J)
```

```
10      CONTINUE
20    CONTINUE
      XBX = 0.0
      DO 30 I=1,N
        XBX = XBX + BX(I)*X(I)
30    CONTINUE
      XBX = SQRT(XBX)
      DO 40 I=1,N
        X(I) = X(I)/XBX
40    CONTINUE
      DO 60 I=1,N
        AX(I) = 0.0
        DO 50 J=1,N
          AX(I) = AX(I) + A(I,J)*X(J)
50      CONTINUE
60    CONTINUE
      U = 0.0
      DO 70 I=1,N
        U = U + AX(I)*X(I)
70    CONTINUE
      DO 80 I=1,N
        R(I) = U*BX(I) - AX(I)
        P(I) = R(I)
80    CONTINUE
2     DO 100 I=1,N
        AP(I) = 0.0
        DO 90 J=1,N
          AP(I) = AP(I) + A(I,J)*P(J)
90      CONTINUE
100   CONTINUE
      DO 120 I=1,N
        BP(I) = 0.0
        DO 110 J=1,N
          BP(I) = BP(I) + B(I,J)*P(J)
110     CONTINUE
120   CONTINUE
      PA = 0.0
      PB = 0.0
      PC = 0.0
      PD = 0.0
      DO 130 I=1,N
        PA = PA + AP(I)*X(I)
        PB = PB + AP(I)*P(I)
        PC = PC + BP(I)*X(I)
        PD = PD + BP(I)*P(I)
```

```
130 CONTINUE
    AA = PB*PC - PA*PD
    BB = PB - U*PD
    CC = PA - U*PC
    AAEZ = ABS(AA) .LE. 1.OE-75
    BBEZ = ABS(BB) .LE. 1.OE-75
    IF(AAEZ .AND. BBEZ) GO TO 12
    IF(AAEZ) GO TO 11
    DD = -BB + SQRT(BB*BB-4.O*AA*CC)
    T = DD/(2.O*AA)
    GO TO 15
11  T = -CC/BB
    GO TO 15
12  T = 0.0
15  DO 140 I=1,N
      X(I) = X(I) + T*P(I)
140 CONTINUE
    DO 160 I=1,N
      BX(I) = 0.0
      DO 150 J=1,N
        BX(I) = BX(I) + B(I,J)*X(J)
150   CONTINUE
160 CONTINUE
    XBX = 0.0
    DO 170 I=1,N
      XBX = XBX + BX(I)*X(I)
170 CONTINUE
    XBX = SQRT(XBX)
    DO 180 I=1,N
      X(I) = X(I)/XBX
180 CONTINUE
    DO 200 I=1,N
      AX(I) = 0.0
      DO 190 J=1,N
        AX(I) = AX(I) + A(I,J)*X(J)
190   CONTINUE
200 CONTINUE
    U = 0.0
    DO 210 I=1,N
      U = U + AX(I)*X(I)
210 CONTINUE
    AI = ABS(U1 - U)
    AJ = ABS(U)*1.OE-03
    AK = AI - AJ
    IF(AK .LT. 0.0) GO TO 3
```

```
      DO 220 I=1,N
        R(I) = U*BX(I) - AX(I)
  220 CONTINUE
      QN = 0.0
      DO 230 I=1,N
        QN = QN + R(I)*AP(I)
  230 CONTINUE
      Q = -QN/PB
      DO 240 I=1,N
        P(I) = R(I) + Q*P(I)
  240 CONTINUE
      M = M + 1
      U1 = U
C     WRITE (3, 9998) M
9998  FORMAT (/1X, 3HM =, I3)
C     WRITE (3,9997)
9997  FORMAT (/2H, U/)
C     WRITE (3, 9996) U
9996  FORMAT (1X, E14.6)
C     WRITE (3, 9995)
9995  FORMAT (/5H X(I)/)
C     WRITE (3, 9994) X
9994  FORMAT (1X, F11.6)
      GO TO 2
  3   CONTINUE
      IF (U .LT. SML) RETURN
      NU = NU + 1
      CX = 0.0
      DO 250 I=1,N
        CX = CX + W(I)*BX(I)
  250 CONTINUE
      CX = CX/XBX
      DO 260 I=1,N
        W(I) = W(I) - CX*X(I)
        X(I) = W(I)
  260 CONTINUE
      IF(NU .GT. N) GO TO 4
      GO TO 1
  4   WRITE (3, 9999)
9999  FORMAT (28H NO EIGENVALUE LESS THAN SML)
      STOP
      END
```

REFERENCES

1. H. C. Cramer, *Mathematical Methods of Statistics*, Princeton University Press, Princeton, N.J., 1974, pp. 341–359.
2. D. Hertz, “A Fast Digital Method of Estimating the Autocorrelation of a Gaussian Stationary Process,” *IEEE Trans*. **ASSP-30**, 329 (April 1982).
3. S. Cacopardi, “Applicability of the Relay Correlator to Radar Signal Processing,” *Electronics Lett*. **19**, 722–723 (September 1983).
4. K. J. Gabriel, “Comparison of 3 Correlation Coefficient Estimators for Gaussian Stationary Processes,” *IEEE Trans*. **ASSP-31**, 1023–1025 (August 1983).
5. G. Jacovitti and R. Cusani, “Performances of the Hybrid-Sign Correlation Coefficients Estimator for Gaussian Stationary Processes,” *IEEE Trans*. **ASSP-33**, 731–733 (June 1985).
6. G. Jacovitti and R. Cusani, “An Efficient Technique for High Correlation Estimation,” *IEEE Trans*. **ASSP-35**, 654–660 (May 1987).
7. J. Bendat and A. Piersol, *Measurement and Analysis of Random Data*, Wiley, New York, 1966.
8. G. Jacovitti, A. Neri, and R. Cusani, “Methods for Estimating the AC Function of Complex Stationary Gaussian Processes,” *IEEE Trans*. **ASSP-35**, 1126–1138 (1987).
9. G. H. Golub and C. F. Van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, 1983.
10. A. R. Gourlay and G. A. Watson, *Computational Methods for Matrix Eigenproblems*, Wiley, New York, 1973.
11. V. Clema and A. Laub, “The Singular Value Decomposition: Its Computation and Some Applictions,” *IEEE Trans*. **AC-25**, 164–176 (April 1980).
12. S. S. Reddi, “Eigenvector Properties of Toeplitz Matrices and Their Applications to Spectral Analysis of Time Series,” in *Signal Processing*, vol. 7, North-Holland, 1984, pp. 46–56.
13. J. Makhoul, “On the Eigenvectors of Symmetric Toeplitz Matrices,” *IEEE Trans*. **ASSP-29**, 868–872 (August 1981).
14. J. D. Mathews, J. K. Breakall, and G. K. Karawas, “The Discrete Prolate Spheroidal Filter as a Digital Signal Processing Tool,” *IEEE Trans*. **ASSP-33**, 1471–1478 (December 1985).
15. L. Genyuan, X. Xinsheng, and Q. Xiaoyu, “Eigenvalues and Eigenvectors of One or Two Sinusoidal Signals in White Noise,” *Proce. IEEE-ASSP Workshop*, Academia Sinica, Beijing. 1986, pp. 310–313.
16. A. Cantoni and P. Butler, “Properties of the Eigenvectors of Persymmetric Matrices with Applications to Communication Theory,” *IEEE Trans*. **COM-24**, 804–809 (August 1976).
17. R. M. Gray, “On the Asymptotic Eigenvalue Distribution of Toeplitz Matrices,” *IEEE Trans*. **IT-16**, 725–730 (1972).
18. O. L. Frost, “An Algorithm for Linearly Constrained Adaptive Array Processing,” *Proc. IEEE* **60**, 926–935 (August 1972).

19. V. U. Reddy, B. Egardt, and T. Kailath, "Least Squares Type Algorithm for Adaptive Implementation of Pisarenko's Harmonic Retrieval Method," *IEEE Trans.* **ASSP-30**, 399–405 (June 1982).
20. H. Chen, T. K. Sarkar, S. A. Dianat, and J. D. Brule, "Adaptive Spectral Estimation by the Conjugate Gradient Method," *IEEE Trans.* **ASSP-34**, 272–284 (April 1986).
21. B. Lumeau and H. Clergeot, "Spatial Localization—Spectral Matrix Bias and Variance—Effects on the Source Subspace," in *Signal Processing*, no. 4, North-Holland, 1982, pp. 103–123.
22. P. Nicolas and G. Vezzosi, "Location of Sources with an Antenna of Unknown Geometry," *Proc. GRETSI-85*, Nice, France, 1985, pp. 331–337.
23. V. R. Algazi and D. J. Sakrison, "On the Optimality of the Karhunen-Loève Expansion," *IEEE Trans.* **IT-15**, 319–321 (March 1969).
24. A. K. Jain, "A fast Karhunen-Loève Transform for a Class of Random Processes," *IEEE Trans.* **COM-24**, 1023–1029 (1976).
25. N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform," *IEEE Trans.* **C-23**, 90–93 (1974).

4 Gradient Adaptive Filters

The adaptive filters based on gradient techniques make a class which is highly appreciated in engineering for its simplicity, flexibility, and robustness. Moreover, they are easy to design, and their performance is well characterized. By far, it is the most widely used class in all technical fields, particularly in communications and control [1, 2].

Gradient techniques can be applied to any structure and provide simple equations. However, because of the looped structure, the exact analysis of the filters obtained may be extremely difficult, and it is generally carried out under restrictive hypotheses not verified in practice [3, 4]. However, simplified approximate investigations provide sufficient results in the vast majority of applications.

The emphasis is on engineering aspects in this chapter. Our purpose is to present the results and information necessary to design an adaptive filter and build it successfully, taking into account the variety of options which make the approach flexible.

4.1. THE GRADIENT—LMS ALGORITHM

The diagram of the gradient adaptive filter is shown in Figure 4.1. The error sequence $e(n)$ is obtained by subtracting from the reference signal $y(n)$ the filtered sequence $\tilde{y}(n)$. The coefficients $C_i(n)$, $0 \leqslant i \leqslant N-1$, are updated by the equation

$$c_i(n+1) = c_i(n) - \delta \frac{\partial e(n+1)}{\partial c_i(n)} e(n+1) \tag{4.1}$$

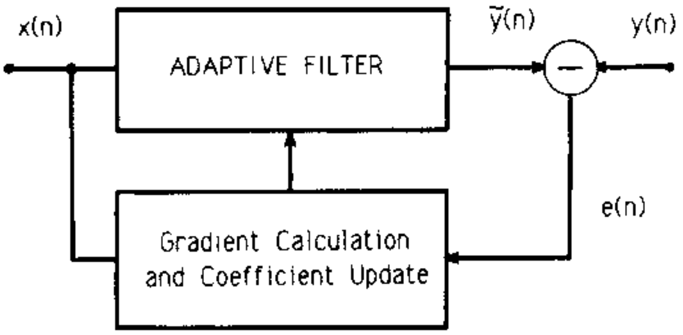


FIG. 4.1 Principle of a gradient adaptive filter.

The products $[\partial e(n+1)/\partial c_i(n)]e(n+1)$ are the elements of the vector V_G, which is the gradient of the function $\frac{1}{2}e^2(n+1)$. The scalar δ is the adaptation step. In the mean, the operation corresponds to minimizing the error power, hence the denomination least means squares (LMS) for the algorithm.

The adaptive filter can have any structure. However, the most straightforward and most widely used is the transversal or FIR structure, for which the error gradient is just the input data vector.

The equations of the gradient adaptive transversal filter are

$$e(n+1) = y(n+1) - H^t(n)X(n+1) \tag{4.2}$$

and

$$H(n+1) = H(n) + \delta X(n+1)e(n+1) \tag{4.3}$$

where $H^t(n)$ is the transpose of the coefficient vector and $X(n+1)$ is the vector of the N most recent input data.

The implementation is shown in Figure 4.2. It closely follows the implementation of the fixed FIR filter, a multiplier accumulator circuit being added to produce the time-varying coefficients. Clearly, $2N+1$ multiplications are needed, as well as $2N$ additions and $2N$ active memories.

Once the number of coefficients N has been chosen, the only filter parameter to be adjusted is the adaptation step δ.

In view of the looped configuration, our first consideration is stability.

4.2. STABILITY CONDITION AND SPECIFICATIONS

The error sequence calculated by equation (4.2) is called "a priori," because it employs the coefficients before updating. The "a posteriori" error is defined as

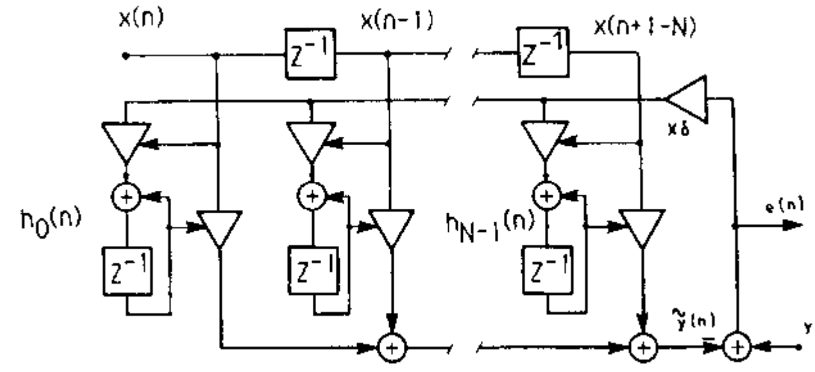


FIG. 4.2 Gradient adaptive transversal filter.

$$\varepsilon(n+1) = y(n+1) - H^t(n+1)X(n+1) \tag{4.4}$$

and it can be computed after (4.2) and (4.3) have been completed. Now, from (4.2) and (4.3), (4.4) can be written as

$$\varepsilon(n+1) = e(n+1)[1 - \delta X^t(n+1)X(n+1)] \tag{4.5}$$

The system can be considered stable if the expectation of the a posteriori error magnitude is smaller than that of the a priori error, which is logical since more information is incorporated in $\varepsilon(n+1)$. If the error $e(n+1)$ is assumed to be independent of the N most recent input data, which is approximately true after convergence, the stability condition is

$$|1 - \delta E[X^t(n+1)X(n+1)]| < 1 \tag{4.6}$$

which yields

$$0 < \delta < \frac{2}{N\sigma_x^2} \tag{4.7}$$

where the input signal power σ_x^2 is generally known or easy to estimate.

The stability condition (4.7) is simple and easy to use. However, in practice, to account for the hypotheses made in the derivation, it is wise to take some margin. For example, a detailed analysis for Gaussian signals shows that stability is guaranteed if [5, 6]

$$0 < \delta < \frac{1}{3}\frac{2}{N\sigma_x^2} \tag{4.8}$$

So, a margin factor of a few units is recommended when using condition (4.7). Once the stability is achieved, the final determination of the step δ in the allowed range is based on performance, compared to specifications.

The two main specifications for gradient adaptive filtering are the system gain and the time constant. The system gain G_S^2 can be defined as the reference to error signal power ratio:

$$G_S^2 = \frac{E[y^2(n)]}{E[e^2(n)]} \tag{4.9}$$

For example, in adaptive prediction, G_S is the prediction gain. The specification is given as a lower bound for the gain, and the adaptation step and the computation accuracy must be chosen accordingly.

The speed of adaptation is controlled by a time constant specification τ_e, generally imposed on the error sequence. The filter time constant τ can be taken as an effective initial time constant obtained by fitting the sequence $E[e^2(n)]$ to an exponential for $n = 0$ and $n = 1$, which yields

$$(E[e^2(0)] - E[e^2(\infty)])e^{-2/\tau} = E[e^2(1)] - E[e^2(\infty)] \tag{4.10}$$

Since τ is related to the adaptation step δ, as shown in the following sections, imposing an upper limit τ_e puts a constraint on δ. Indeed the gain and speed specifications must be compatible and lead to a nonempty range of values for δ; otherwise another type of algorithm, like least squares, must be relied upon.

First, the relation between adaptation step and residual error is investigated.

4.3. RESIDUAL ERROR

The gradient adaptive filter equations (4.2) and (4.3) yield

$$H(n+1) = [I_N - \delta X(n+1)X^t(n+1)]H(n) + \delta X(n+1)y(n+1) \tag{4.11}$$

When the time index n approaches infinity, the coefficients reach their steady-state values and the average of $H(n+1)$ becomes equal to the average of $H(n)$. Hence, assuming independence between coefficient variations and input data vectors, we get

$$E[H(\infty)] = R^{-1}r_{yx} = H_{\text{opt}} \tag{4.12}$$

Using the notation of Section 1.4, we write

$$R = E[X(n)X^t(n)], \quad r_{yx} = E[X(n+1)y(n+1)] \tag{4.13}$$

Therefore the gradient algorithm provides the optimal coefficient set H_{opt} after convergence and in the mean. The vector r_{yx} is the cross-correlation between the reference and input signals.

The minimum output error power $E_{\min}$ can also be expressed as a function of the signals and their cross-correlation.

For the set of coefficients $H(n)$, the mean square output error $E(n)$ is

$$E(n) = E[(y(n) - H^t(n)X(n))^2] \tag{4.14}$$

Now, setting the coefficients to their optimal values gives

$$E_{\min} = E[y^2(n)] - H^t_{\text{opt}} R H_{\text{opt}} \tag{4.15}$$

or

$$E_{\min} = E[y^2(n)] - H^t_{\text{opt}} r_{yx} \tag{4.16}$$

or

$$E_{\min} = E[y^2(n)] - r^t_{yx} R^{-1} r_{yx} \tag{4.17}$$

In these equations the filter order N appears as the dimension of the AC matrix R and of the cross-correlation vector r_{yx}.

For fixed coefficients $H(n)$ the mean square error (MSE) $E(n)$ can be rewritten as a deviation from the minimum:

$$E(n) = E_{\min} + [H_{\text{opt}} - H(n)]^t R [H_{\text{opt}} - H(n)] \tag{4.18}$$

The input data AC matrix R can be diagonalized as

$$R = M^t \operatorname{diag}(\lambda_i) M, \quad M^t M = I_N \tag{4.19}$$

where, as shown in the preceding chapter, $\lambda_i (0 \leqslant i \leqslant N-1)$ are the eigenvalues and M the modal unitary matrix.

Letting

$$[\alpha(n)] = M[H_{\text{opt}} - H(n)] \tag{4.20}$$

be the coefficient difference vector in the transformed space, we obtain the concise form of (4.18)

$$E(n) = E_{\min} + [\alpha(n)]^t \operatorname{diag}(\lambda_i)[\alpha(n)] \tag{4.21}$$

Completing the products, we have

$$E(n) = E_{\min} + \sum_{i=0}^{N-1} \lambda_i \alpha_i^2(n) \tag{4.22}$$

If Λ denotes the column vector of the eigenvalues λ_i, and $[\alpha^2(n)]$ denotes the column vector with elements $\alpha_i^2(n)$, then

$$E(n) = E_{\min} + \Lambda^t[\alpha^2(n)] \tag{4.23}$$

The analysis of the gradient algorithm is carried out by following the evolution of the vector $[\alpha(n)]$ according to the recursion

$$[\alpha(n+1)] = [\alpha(n)] - \delta M X(n+1) e(n+1) \tag{4.24}$$

The corresponding covariance matrix is

$$\begin{aligned}[\alpha(n+1)][\alpha(n+1)]^t = {} & [\alpha(n)][\alpha(n)]^t - 2\delta M X(n+1) e(n+1)[\alpha(n)]^t \\ & + \delta^2 e^2(n+1) M X(n+1) X^t(n+1) M^t\end{aligned} \tag{4.25}$$

The definition of $e(n+1)$ yields

$$e(n+1) = y(n+1) - H^r_{\text{opt}} X(n+1) + X^t(n+1) M^t [\alpha(n)] \tag{4.26}$$

Equations (4.25) and (4.26) determine the evolution of the system. In order to get useful results, we make simplifying hypotheses, particularly about $e^2(n)$ [7].

It is assumed that the following variables are independent:

The error sequence when the filter coefficients are optimal
The data vector $X(n+1)$
The coefficient deviations $H(n) - H_{\text{opt}}$

Thus

$$E\{[y(n+1) - H^t_{\text{opt}} X(n+1)] X^t(n+1) M^t [\alpha(n)]\} = 0 \tag{4.27}$$

Although not rigorously verified, the above assumptions are reasonable approximations, because the coefficient deviations and optimum output error are noiselike sequences and the objective of the filter is to make them uncorrelated with the N most recent input data. Anyway, the most convincing argument in favor is that the results derived are in good agreement with experiments.

Now, taking the expectation of both sides of (4.25), yields

$$\begin{aligned}E\{[\alpha(n+1)][\alpha(n+1)]^t\} = {} & [I_N - 2\delta \operatorname{diag}(\lambda_i)] E\{[\alpha(n)][\alpha(n)]^t\} \\ & + \delta^2 E[e^2(n+1)] \operatorname{diag}(\lambda_i)\end{aligned} \tag{4.28}$$

For varying coefficients, under the above independence hypotheses, expression (4.23) becomes

$$E[e^2(n+1)] = E_{\min} + \Lambda^t E[\alpha^2(n)] \tag{4.29}$$

Considering the main diagonals of the matrices, and using vector notation and expression (4.29) for the error power, we derive the equation

$$E[\alpha^2(n+1)] = [I_N - 2 \operatorname{diag}(\lambda_i) + \delta^2 \Lambda \Lambda^t] E[\alpha^2(n)] + \delta^2 E_{\min} \Lambda \tag{4.30}$$

A sufficient condition for convergence is that the sum of the absolute values of the elements of any row in the matrix multiplying the vector $E[\alpha^2(n)]$ be less than unity:

$$0 < 1 - 2\delta\lambda_i + \delta^2\lambda_i\left(\sum_{j=0}^{n-1}\lambda_j\right) < 1, \quad 0 \leqslant 1i \leqslant N-1 \tag{4.31}$$

from which we obtain the stability condition

$$0 < \delta < \frac{2}{\sum\limits_{j=0}^{N-1}\lambda_j} = \frac{2}{N\sigma_x^2}$$

which is the condition already found in Section 4.2, through a different approach.

Once the stability conditions are fulfilled, recursion (4.28) yields, as $n \to \infty$,

$$E\{[\alpha(\infty)][\alpha(\infty)]^t\} = \frac{\delta}{2}E(\infty)I_N \tag{4.32}$$

Due to the definition of the vector $[\alpha(n)]$, equation (4.32) also applies to the coefficient deviations themselves. Thus the coefficient deviations, after convergence, are statistically independent and have the same power.

Now, combining (4.32) and (4.29) yields the residual error E_R:

$$E(\infty) = E_R = \frac{E_{\min}}{1-(\delta/2)N\sigma_x^2} \tag{4.33}$$

Finally, the gradient algorithm produces an excess output MSE related to the adaptation step. Indeed, when δ approaches the stability limit, the output error power approaches infinity. The ratio of the steady-state MSE to the minimum attainable MSE is called the final misadjustment M_{adj}:

$$M_{\text{adj}} = \frac{E_R}{E_{\min}} = \frac{1}{1-(\delta/2)N\sigma_x^2} \tag{4.34}$$

In practical realizations, due to the margin generally taken for the adaptation step size, the approximation

$$E_R \approx E_{\min}\left(1 + \frac{\delta}{2}N\sigma_x^2\right) \tag{4.35}$$

is often valid, and the excess output MSE is approximately proportional to the step size. In fact, it can be viewed as a gradient noise, due to the approximation of the true cost function gradient by an instantaneous value.

4.4. LEARNING CURVE AND TIME CONSTANT

The adaptive filter starts from an initial state, which often corresponds to zero coefficients. From there, its evolution is controlled by the input and reference signals, and it is possible to define learning curves by parameter averaging.

The evolution of the coefficient difference vector in the transformed space is given by quation (4.24). Substituting equation (4.26) into this equation and taking the expectation yields, under the hypotheses of Section 4.3,

$$E[\alpha(n+1)] = [I_N - \delta\ \mathrm{diag}(\lambda_i)]E[\alpha(n)] \tag{4.36}$$

Substituting into equation (4.29) and iterating from the time origin leads to

$$E(n) - E_{\min} = \Lambda^t\ \mathrm{diag}(1-\delta\lambda_i)^{2n} E[\alpha^2(0)] \tag{4.37}$$

The same results can also be derived from equation (4.30) after some simplification, assuming the step size δ is small.

Clearly, the evolution of the coefficients and the output MSE depends on the input signal matrix eigenvalues, which provide as many different modes. In the long run, it is the smallest eigenvalue which controls the convergence.

The filter time constant τ_e obtained from an exponential fitting to the output rms error is obtained by applying definition (4.10) and neglecting the residual error:

$$E(0)e^{-2/\tau_e} = \Lambda^t\ \mathrm{diag}(1-\delta\lambda_i)E[\alpha^2(0)] \tag{4.38}$$

We can also obtain it approximately by applying (4.29) at the time origin:

$$\Lambda^t E[\alpha^2(0)]\left[1-\frac{2}{\tau_e}\right] = \Lambda^t\ \mathrm{diag}(1-2\delta\lambda_i)E[\alpha^2(0)] \tag{4.39}$$

Hence

$$\tau_e = \frac{1}{\delta}\frac{\sum_{i=0}^{N-1}\lambda_i E\{\alpha_i^2(0)\}}{\sum_{i=0}^{N-1}\lambda_i E\{\alpha_i^2(0)\}\lambda_i} \tag{4.40}$$

If the eigenvalues are not too dispersed, we have

$$\tau_e \approx \frac{N}{\delta\sum_{i=0}^{N-1}\lambda_i} = \frac{1}{\delta\sigma_x^2} \tag{4.41}$$

The filter time constant is proportional to the inverse of the adaptation step size and of the input signal power. Therefore, an estimation of the signal power is needed to adjust the adaptation speed. Moreover, if the signal is nonstationary, the power estimation must be carried out in real time to reach a high level of performance.

A limit on the adaptation speed is imposed by the stability condition (4.7).

From equation (4.30), it appears that the rows of the square matrix are quadratic functions of the adaptation step and all take their minimum norm for

$$\delta_m = \frac{1}{\sum_{i=0}^{N-1} \lambda_i} = \frac{1}{N\sigma_x^2} \tag{4.42}$$

which corresponds to the fastest convergence. Therefore the smallest time constant is

$$\tau_{e,\min} = N \tag{4.43}$$

In these conditions, if the eigenvalues are approximately equal to the signal power, which occurs for noiselike signals in certain modeling applications, the learning curve, taken as the output MSE function, is obtained from (4.36) by

$$E(n) - E_R = (E(0) - E_R)\left(1 - \frac{1}{N}\right)^{2n} \tag{4.44}$$

For zero initial values of the coefficients, $E(0)$ is just the reference signal power.

Overall, the three expressions (4.7), (4.33), and (4.41) give the basic information to choose the adaptation step δ and evaluate a transversal gradient adaptive filter. They are sufficient in many practical cases.

Example

Consider the second-order adaptive FIR prediction filter in Figure 4.3, with equations

$$e(n+1) = x(n+1) - a_1(n)x(n) - a_2(n)x(n-1)$$

$$\begin{bmatrix} a_1(n+1) \\ a_2(n+1) \end{bmatrix} = \begin{bmatrix} a_1(n) \\ a_2(n) \end{bmatrix} + \delta \begin{bmatrix} x(n) \\ (n-1) \end{bmatrix} e(n+1) \tag{4.45}$$

The input signal is a sinusoid in noise:

$$x(n) = \sin\left(\frac{n\pi}{4}\right) + b(n) \tag{4.46}$$

The noise $b(n)$ has power $\sigma_b^2 = 5 \times 10^{-5}$. The input signal power is $\sigma_x^2 = 0.5$. The step size δ is 0.05. Starting from zero-valued coefficients, the evolution

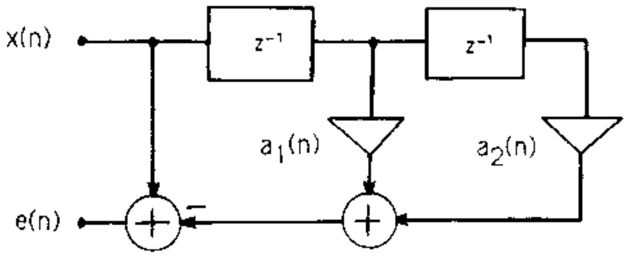


FIG. 4.3 Second-order prediction filter.

of the output error, the two coefficients, and the corresponding zeros in the complex plane are shown in Figure 4.4. Clearly the output error time constant is in reasonably good agreement with estimation (4.41).

In the filter design process, the next step is the estimation of the coefficient and internal data word lengths needed to meet the adaptive filter specifications.

4.5. WORD-LENGTH LIMITATIONS

Word-length limitations introduce roundoff error sources, which degrade the filter performance. The roundoff process generally takes place at the output of the multipliers, as represented by the quantizers Q in Figure 4.5.

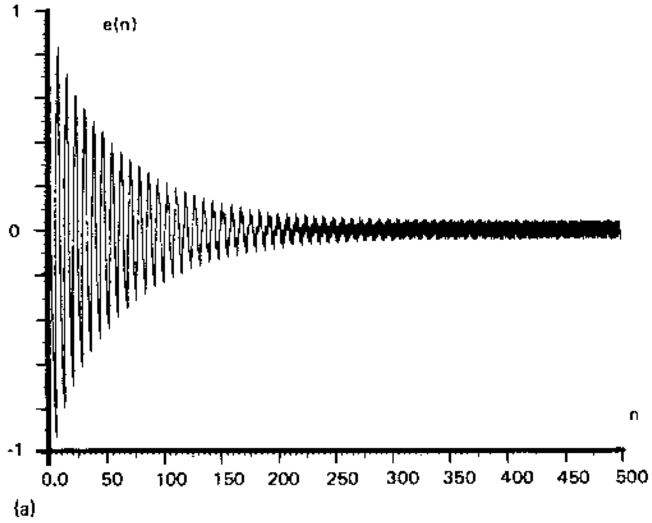
In roundoff noise analysis a number of simplifying hypotheses are generally made concerning the source statistics. The errors are identically distributed and independent; with rounding, the distribution law is uniform in the interval $[-q/2, q/2]$, where q is the quantization step size, the power is $q^2/12$, and the spectrum is flat.

Concerning the adaptive transversal filter, there are two different categories of roundoff errors, corresponding to internal data and coefficients [8].

The quantization processes at each of the N filter multiplication outputs amount to adding N noise sources at the filter output. Therefore, the output MSE is augmented by $Nq_2^2/12$, assuming q_2 is the quantization step.

The quantization with step q_1 of the multiplication result in the coefficient updating section is not so easily analyzed. Recursion (4.28) is modified as follows, taking into account the hypotheses on the roundoff noise sources and their independence of the other variables:

$$E\{[\alpha(n+1)][\alpha(n+1)]^t\} = [I_N - 2\delta\,\mathrm{diag}(\lambda_i)]E\{[\alpha(n)][\alpha(n)]^t\} + \delta^2 E[e^2(n+1)]\mathrm{diag}(\lambda_i) + \frac{q_1^2}{12} I_N \tag{4.47}$$



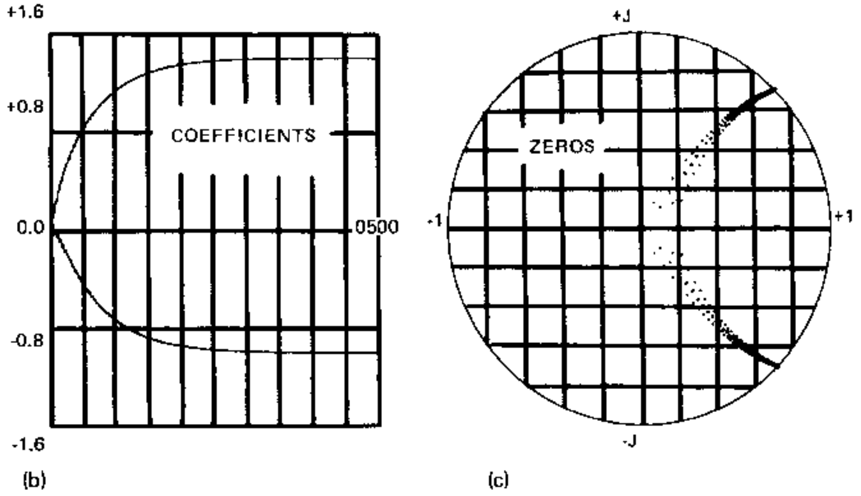


FIG. 4.4 The second-order adaptive FIR prediction filter: (a) output error sequence; (b) coefficient versus time; (c) zeros in the complex plane.

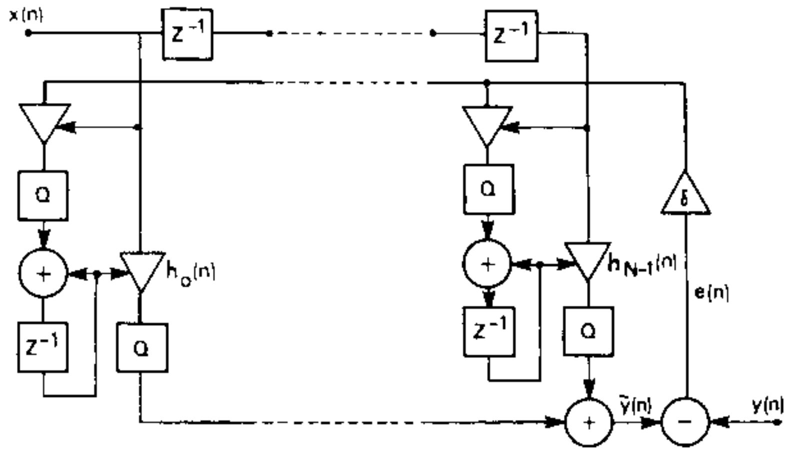


FIG. 4.5 Adaptive FIR filter with word-length limitations.

An additional gradient noise is introduced.

When $n \to \infty$, equation (4.29) yields, as before,

$$E_{RT}\left(1 - \frac{\delta}{2} N\sigma_x^2\right) = E_{\min} + \frac{q_1^2}{12}\frac{N}{2\delta} \tag{4.48}$$

Hence, the total residual error, taking into account the quantization of the filter coefficients with step q_1 and the quantization of internal data with step q_2, as shown in Figure 4.5, is

$$E_{RT} = \frac{1}{1 - (\delta/2)N\sigma_x^2}\left[E_{\min} + \frac{N}{2\delta}\frac{q_1^2}{12} + N\frac{q_2^2}{12}\right] \tag{4.49}$$

or, assuming a small excess output MSE,

$$E_{RT} \approx E_{\min}\left(1 + \frac{\delta}{2} N\sigma_x^2\right) + \frac{N}{2\delta}\frac{q_1^2}{12} + N\frac{q_2^2}{12} \tag{4.50}$$

This expression shows that the effects of the two kinds of quantizations are different. Because of the factor $\frac{1}{\delta}$, the coefficient quantization and the corresponding word length can be very sensitive. In fact, there is an optimum δ_{opt} for the adaptation step size which minimizes the total residual error; according to (4.50) it is obtained through derivation as

$$\tfrac{1}{2} E_{\min} N\sigma_x^2 - \frac{N}{2}\frac{q_1^2}{12}\frac{1}{\delta_{\text{opt}}^2} = 0 \tag{4.51}$$

and

$$\delta_{\text{opt}} = \frac{1}{\sqrt{E_{\min}}\sigma_x} \frac{1}{\sqrt{3}} \frac{q_1}{2} \tag{4.52}$$

The curve of the residual error versus the adaptation step size is shown in Figure 4.6. For δ decreasing from the stability limit, the minimum is reached for δ_{opt}; if δ is decreased further, the curve indicates that the total error should grow, which indeed has no physical meaning. The hypotheses which led to (4.50) are no longer valid, and a different phenomenon occurs, namely blocking.

According to the coefficient evolution equation (4.3), the coefficient $h_i(n)$ is frozen if

$$|\delta x(n-i)e(n)| < \frac{q_1}{2} \tag{4.53}$$

Let us assume that the elements of the vector $\delta X(n)e(n)$ are uncorrelated with each other and distribute uniformly in the interval $[q_1/2, q_1/2]$. Then

$$\delta^2 E\{e^2(n)X(n)X^t(n)\} = \frac{q_1^2}{12} I_N \tag{4.54}$$

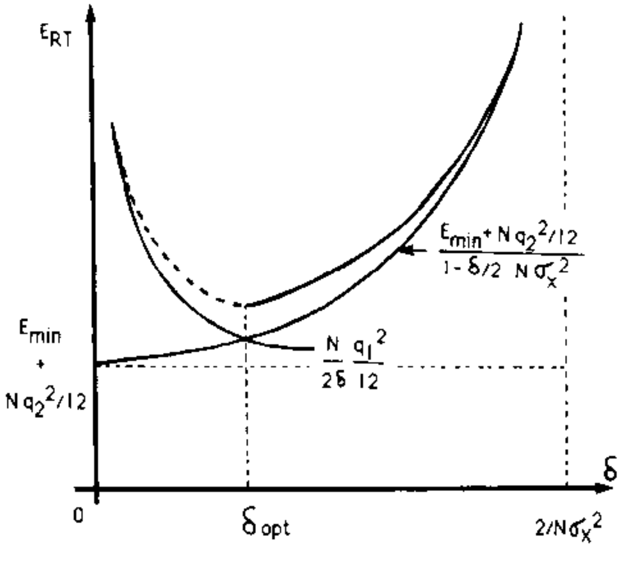


FIG. 4.6 Residual error against adaptation step size.

If the coefficients are close to their optimal values and if the input signal can be approximated by a white noise, then equations (4.54) and (4.51) are equivalent. A blocking radius ρ can then be defined for the coefficients by

$$\rho^2 = E\{[H(n) - H_{\text{opt}}]^t (H(n) - H)_{\text{opt}}]\} \tag{4.55}$$

Now, considering that

$$H(n) - H_{\text{opt}} = R^{-1} E[e(n)X(n)] \tag{4.56}$$

we have, from (4.54) and the identity $X^t X = \text{trace}(XX^t)$,

$$\rho^2 = \frac{1}{12}\left(\frac{q_1}{\delta}\right)^2 \sum_{i-0}^{N-1} \lambda_i^{-2} \tag{4.57}$$

The blocking radius is a function of the spread of the input AC matrix eigenvalues. Blocking can occur for adaptation step sizes well over δ_{opt}, given by (4.52), if there are small eigenvalues.

In adaptive filter implementations, the adaptation step size is often imposed by system specifications (e.g., the time constant), and the coefficient quantization step size q_1 is chosen small enough to avoid the blocking zone with some margin.

Quantization steps q_1 and q_2 are generally derived from expression (4.50). Considering the crucial advantage of digital processing, which is that operations can be carried out with arbitrary accuracy, the major contribution in the total residual error should be the theoretical minimal error $E_{\min}$. In a balanced realization, the degradations from different origins should be similar. Hence, a reasonable design choice is

$$\frac{1}{2}\left[E_{\min} \frac{N\delta\sigma_x^2}{2}\right] = \frac{N}{2\delta}\frac{q_1^2}{12} = N\frac{q_2^2}{12} \tag{4.58}$$

If b_c is the number of bits of the coefficients and $h_{\max}$ is the largest coefficient magnitude, then, assuming fixed-point binary representation, we have

$$q_1 = h_{\max} 2^{1-b_c} \tag{4.59}$$

Under these conditions

$$2^{2b_c} = \frac{2}{3}\frac{h_{\max}^2}{\delta^2 E_{\min}\sigma_x^2} \tag{4.60}$$

with the assumption that $E_{\min}$ is the dominant term in (4.50), that is,

$$G_S^2 E_{\min} \approx \sigma_y^2$$

By introducing the time constant specification τ_e, one has approximately

$$b_c \approx \log_2(\tau_e) + \log_2(G_S) + \log_2\left(h_{\max}\frac{\sigma_x}{\sigma_y}\right) \tag{4.61}$$

This expression gives an estimation of the coefficient word length necessary to meet the specifications of a gradient adaptive filter. However there is one variable which is not readily available, $h_{\max}$; a simple bound can be derived, if we assume a large system gain and refer to the eigenfilters of Section 3.7:

$$\sigma_y^2 = E[y^2(n)] \approx H^t(n)RH(n) \geqslant \lambda_{\min}H^t(n)H(n) \tag{4.62}$$

Now

$$\sigma_y^2 \geqslant \lambda_{\min}h_{\max}^2$$

and

$$\left(h_{\max}\frac{\sigma_x}{\sigma_y}\right)^2 \leqslant \frac{\sigma_x^2}{\lambda_{\min}} \tag{4.63}$$

Therefore, the last term on the right side of (4.61) is bounded by zero for input signals whose spectrum is approximately flat, but it can take positive values for narrowband signals.

Estimate (4.61) can produce large values for b_c; that word length is necessary in the coefficient updating accumulator but not in the filter multiplications.

In practice, additional quantizers can be introduced just before the multiplications by $h_i(n)$ in Figure 4.5 in order to avoid multiplications with high precision factors. The effects of the additional roundoff noise sources introduced that way can be investigated as above.

Often, nonstationary signals are handled, and estimate (4.61) is for stationary signals. In this case, a first approach is to incorporate the signal dynamic range in the last term of (4.61).

To complete the filter design, the number of bits b_i of the internal data can be determined by setting

$$q_2 = \max\{|x(n)|, |y(n)|\}2^{1-b_i} \tag{4.64}$$

with the assumption that $\sigma_x^2 \geqslant \sigma_y^2$, which is true in linear prediction and often valid in system modeling, and taking the value 4 as the peak factor of the signal $x(n)$ as in the Gaussian case. Thus

$$q_2 = 4\sigma_x 2^{1-b_i}$$

Now, (4.58) yields

$$2^{2b_i} = 2^4\frac{4}{3}\frac{1}{E_{\min}\delta}$$

By introducing the specifications we obtain

$$b_i \approx 2 + \log_2\left(\frac{\sigma_x}{\sigma_y}\right) + \log_2(G_S) + \tfrac{1}{2}\log_2(\tau_e) \tag{4.65}$$

This completes the implementation parameter estimation for the standard gradient algorithm. However, some modifications can be made to this algorithm, which are either useful or even mandatory.

4.6. LEAKAGE FACTOR

When the input signal vanishes, the driving term in recursion (4.3) becomes zero and the coefficients are locked up. In such conditions, it might be preferable to have them return to zero. This is achieved by the introduction of a leakage factor γ in the updating equation:

$$H(n+1) = (1-\gamma)H(n) + \delta X(n+1)e(n+1) \tag{4.66}$$

The coefficient recursion is

$$H(n+1) = [(1-\gamma)I_N - \delta X(n+1)X^t(n+1)]H(n) + \delta y(n+1)X(n+1) \tag{4.67}$$

After convergence,

$$H_\infty = E[H(\infty)] = \left[R + \frac{\gamma}{\delta} I_N\right]^{-1} r_{yx} \tag{4.68}$$

The leakage factor γ introduces a bias on the filter coefficients, which can be expressed in terms of the optimal values as

$$H_\infty = \left[R + \frac{\gamma}{\delta} I_N\right]^{-1} R H_{\text{opt}} \tag{4.69}$$

The same effect is obtained when a white noise is added to the input signal $x(n)$; a constant equal to the noise power is added to the elements of the main diagonal of the input AC matrix.

To evaluate the impact of the leakage factor, we rewrite the coefficient vector H_∞ as

$$H_\infty = M^t \text{diag}\left(\frac{\lambda_i}{\lambda_i + \gamma/\delta}\right) M H_{\text{opt}} \tag{4.70}$$

The significance of the bias depends on the relative values of $\lambda_{\min}$ and $\frac{\gamma}{\delta}$.

Another aspect is that the cost function actually minimized in the whole process is

$$J_\gamma(n) = E\left\{[y(n) - X^t(n)H(n-1)]^2 + \frac{\gamma}{\delta} H^t(n-1)H(n-1)\right\} \tag{4.71}$$

The last term represents a constraint which is imposed on the coefficient magnitudes [9].

The LS solution is given by (4.68), and the coefficient bias is

$$H - H_{\text{opt}} = \left[\left(R + \frac{\gamma}{\delta} \mathrm{I}_N \right)^{-1} R - I_N \right] H_{\text{opt}} \tag{4.72}$$

Hence the filter output MSE becomes

$$E_R = E_{\min} + [H - H_{\text{opt}}]^t R [H - H_{\text{opt}}] \tag{4.73}$$

The leakage factor is particularly useful for handling nonstationary signals. With such signals, the leakage value can be chosen to reduce the output error power.

If the coefficients are computed by minimizing the above cost function taken on a limited set of data, the coefficient variance can be estimated by

$$E\{[H - H_0][H - H_0]^t\} = E_R \left[R + \frac{\gamma}{\delta} I_N \right]^{-1} R \left[R + \frac{\gamma}{\delta} \right]^{-1} \tag{4.74}$$

and the coefficient MSE H_{MSE} is

$$H_{\text{MSE}} = [H - H_{\text{opt}}]^t [H - H_{\text{opt}}] + \operatorname{trace}(E\{[H - H_0][H - H_0]^t\}) \tag{4.75}$$

When γ increases from zero, H_{MSE} decreases from $E_R \operatorname{trace}(R^{-1})$, then reaches a minimum and increases, because in (4.75) the variance decreases faster than the bias increases at the beginning, as can be seen directly for dimension $N = 1$ [9]. A minimal output MSE corresponds to the minimum of H_{MSE}.

A similar behavior can be observed when the gradient algorithm is applied to nonstationary signals. An illustration is provided by applying a speech signal to an order 8 linear predictor. The prediction gain measured is shown in Figure 4.7 versus the leakage factor for several adaptation step sizes δ. The maximum of the prediction gain is clearly visible. It is also a justification for the values sometimes retained for speech prediction, which are $\delta = 2^{-6}$ and $\gamma = 2^{-8}$.

The leakage factor, which can nicely complement the conventional gradient algorithm, is recommended for the sign algorithm because it bounds the coefficients and thus prevents divergence.

4.7. THE LMAV AND SIGN ALGORITHMS

Instead of the LS, the least absolute value (LAV) criterion can be used to compare variables, vectors, or functions. It has two specific advantages: it does not necessarily lead to minimum phase solutions; it is robust to outliers

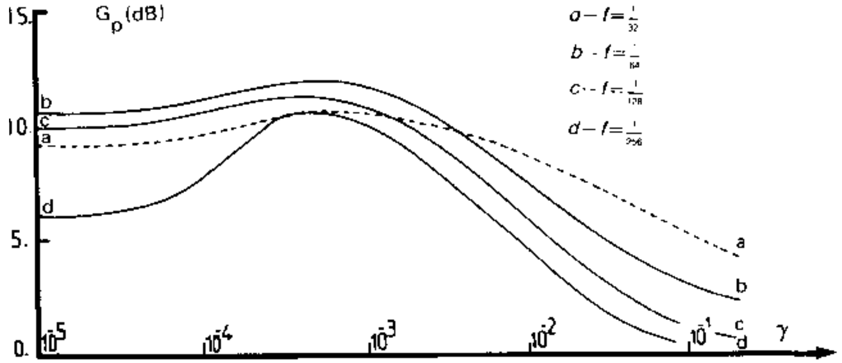


FIG. 4.7 Prediction gain vs. leakage factor for a speech sentence.

in a data set. Similarly, the least mean absolute value (LMAV) can replace the LMS in adaptive filters [10].

The gradient of the function $|e(n+1)|$ is the vector whose elements are

$$\begin{aligned}\frac{\partial|e(n+1)|}{\partial h_i} &= \frac{\partial}{\partial h_i}|\,y(n+1) - X^t(n+1)H(n)| \\ &= -x(n+1-i)\text{sign}\ e(n+1)\end{aligned} \tag{4.76}$$

where sign e is $+1$ if e is positive and -1 otherwise. The LMAV algorithm for the transversal adaptive filter is

$$H(n+1) = H(n) + \delta X(n+1)\text{sign}\ e(n+1) \tag{4.77}$$

where Δ, a positive constant, is the adaptation step.

The convergence can be studied by considering the evolution of the coefficient vector toward the optimum H_{opt}. Equation (4.77) can be rewritten as

$$H(n+1) - H_{\text{opt}} = H(n) - H_{\text{opt}} + \Delta X(n+1)\text{sign}\ e(n+1)$$

Taking the norm squared of both sides yields

$$\begin{aligned}[H(n+1) - H_{\text{opt}}]^t[H(n+1) - H_{\text{opt}}] &= [H(n) - H_{\text{opt}}]^t[H(n) - H_{\text{opt}}] \\ &+ 2\Delta\ \text{sign}\ e(n+1)X^t(n+1)[H(n) - H_{\text{opt}}] + \Delta^2 X^t(n+1)X(n+1)\end{aligned} \tag{4.78}$$

or, with further decomposition,

$$\begin{aligned}\|H(n+1) - H_{\text{opt}}\|^2 =& \|H(n) - H_{\text{opt}}\|^2 + \Delta^2\|X(n+1)\|^2 - 2\Delta|e(n+1)| \\ &+ 2\Delta\ \text{sign}\ e(n+1)[\,y(n+1) - X^t(n+1)H_{\text{opt}}]\end{aligned}$$

Hence we have the inequality

$$\|H(n+1) - H_{\text{opt}}\|^2 \leqslant \|H(n) - H_{\text{opt}}\|^2 + \Delta^2\|X(n+1)\|^2 - 2\Delta|e(n+1)| \\ + 2\Delta|y(n+1) - X^t(n+1)H_{\text{opt}}|$$

Taking the expectation of both sides gives

$$E\{\|H(n+1) - H_{\text{opt}}\|^2\} \leqslant \|H(n) - H_{\text{opt}}\|^2 \\ + \Delta^2 N\sigma_x^2 - 2\Delta E\{|e(n+1)|\} + 2\Delta E_{\min} \tag{4.79}$$

where the minimal error $E_{\min}$ is

$$E_{\min} = E[|y(n+1) - X^t(n+1)H_{\text{opt}}|] \tag{4.80}$$

If the system starts with zero coefficients, then

$$E\{\|H(n+1) - H_{\text{opt}}\|^2\} \leqslant \|H_{\text{opt}}\|^2 \\ + (n+1)(\Delta^2 N\sigma_x^2 + 2\Delta E_{\min}) - 2\Delta \sum_{p=1}^{n+1} E\{|e(p)|\}$$

Since the left side is nonnegative, the accumulated error is bounded by

$$\frac{1}{n+1} E\left\{\sum_{p=1}^{n+1} |e(p)|\right\} \leqslant \frac{\Delta}{2} N\sigma_x^2 + E_{\min} + \frac{\|H_{\text{opt}}\|^2}{2\Delta(n+1)} \tag{4.81}$$

This is the basic equation of LMAV adaptive filters. It has the following implications:

Convergence is obtained for any positive step size Δ.
After convergence the residual error E_R is bounded by

$$E_R \leqslant E_{\min} + \frac{\Delta}{2} N\sigma_x^2 \tag{4.82}$$

It is difficult to define a time constant as in Section 4.1. However, an adaptation time τ_A can be defined as the number of iterations needed for the last term in (4.81) to become smaller than $E_{\min}$. Then we have

$$\tau_A = \frac{1}{\Delta} \frac{\|H_{\text{opt}}\|^2}{2E_{\min}} \tag{4.83}$$

The performance of the LMAV adaptive filters can be assessed from the above expressions. A comparison with the results given in Sections 4.3 and 4.4 for the standard LMS algorithm clearly shows the price paid for the simplification in the coefficient updating circuitry. The main observation is that, if a small excess output MSE is required, the adaptation time can become very large.

Another way of simplifying gradient adaptive filters is to use the following coefficient updating technique:

$$H(n+1) = H(n) + \Delta e(n+1) \mathrm{sign}\, X(n+1) \tag{4.84}$$

This algorithm can be viewed as belonging to the LMS family, but with a normalized step size. Since

$$\mathrm{sign}\, x = \frac{x}{|x|} \tag{4.85}$$

and $|x|$ can be coarsely approximated by the efficient value σ_x, equation (4.84) corresponds to a gradient filter with adaptation step size

$$\delta = \frac{\Delta}{\sigma_x} \tag{4.86}$$

The performance can be assessed by replacing δ in the relevant equations. Pursuing further in that direction, we obtain the sign algorithm

$$H(n+1) = H(n) + \Delta \ \mathrm{sign}\, e(n+1) \mathrm{sign}\, X(n+1) \tag{4.87}$$

The detailed analysis is rather complicated. However, a coarse but generally sufficient approach consists of assuming a standard gradient algorithm with step size

$$\delta = \frac{\Delta}{\sigma_x \sigma_e} \tag{4.88}$$

where σ_x and σ_e are the efficient values of the input signal and output error, respectively.

In the learning phase, starting with zero-valued coefficients, it can be assumed that $\sigma_e \approx \sigma_y$ and the initial time constant τ_S of the sign algorithm can be roughly estimated by

$$\tau_S \approx \frac{1}{\Delta} \frac{\sigma_y}{\sigma_x} \tag{4.89}$$

After convergence it is reasonable to assume $\sigma_e^2 = E_{\min}$. If the adaptation step is small, the residual error E_{RS} in the sign algorithm can be estimated by

$$E_{RS} \approx E_{\min} \left(1 + \frac{N\Delta}{2} \frac{\sigma_x}{\sqrt{E_{\min}}} \right) \tag{4.90}$$

A condition for the above estimation to be valid is obtained by combining (4.7) and (4.88), which yields

$$\Delta \ll \frac{2}{N}$$

If the step size is not small enough, the convergence will stop when the error becomes so small that the stability limit is reached, approximately

$$\Delta \approx \frac{2}{N}\frac{\sigma_e}{\sigma_x}$$

In that situation, the residual error can be estimated by

$$\sqrt{E_{RS}} \approx \frac{\Delta}{2} N\sigma_x \tag{4.91}$$

which can be compared with (4.82) when $E_{\min}$ is neglected.

It is worth pointing out that, for stability reasons, a leakage term is generally introduced in the sign algorithm coefficient, giving

$$H(n+1) = (1-\gamma)H(n) + \Delta \text{ sign } e(n+1)\text{sign } X(n+1) \tag{4.92}$$

Under these conditions, the coefficients are bounded by

$$|h_i(n)| \leqslant \frac{\Delta}{\gamma}, \quad 0 \leqslant i \leqslant N-1 \tag{4.93}$$

Overall, it can be stated that the sign algorithm is slower than the standard gradient algorithm and leads to larger excess output MSE [11–12]. However, it is very simple; moreover it is robust because of the built-in normalization of its adaptation step, and it can handle nonstationary signals. It is one of the most widely used adaptive filter algorithms.

4.8. NORMALIZED ALGORITHMS FOR NONSTATIONARY SIGNALS

When handling nonstationary signals, adaptive filters are expected to trace as closely as possible the evolution of the signal parameters. However, due to the time constant there is a delay which leads to a tracking error. Therefore the excess output MSE has two components: the gradient misadjustment error, and the tracking error.

The efficiency of adaptive filters depends on the signal characteristics. Clearly, the most favorable situation is that of slow variations, as mentioned in Section 2.13. The detailed analysis of adaptive filter performance is based on nonstationary signal modeling techniques. Nonstationarity can affect the reference signal as well as the filter input signal. In this section a highly simplified example is considered to illustrate the filter behavior.

When only the reference signal is assumed to be nonstationary, the developments of the previous sections can, with adequate modifications, be kept. The nonstationarity of the reference is reflected in the coefficient updating equation (4.3) by the fact that the optimal vector is time dependent:

$$H(n+1) - H_{\text{opt}}(n+1) = H(n) - H_{\text{opt}}(n) + \delta e(n+1)X(n+1) \tag{4.94}$$

If it can be assumed that the optimal coefficients are generated by a first-order model whose inputs are zero mean i.i.d. random variables $e_{nS,i}(n)$, with variance σ_{nS}^2, as in Section 2.13, then

$$H_{\text{opt}}(n+1) = (1-\gamma|H_{\text{opt}}(n) + [e_{nS,0}(n+1), \dots, e_{nS,(N-1)}(n+1)]^t \tag{4.95}$$

Furthermore, if the variations are slow, which implies $\gamma \approx 1$, the net effect of the nonstationarity is the introduction of the extra term $\sigma_{nS} I_N$ in recursion (4.28). As already seen for the coefficient roundoff, the residual error E_{RTnS} is

$$E_{RTnS}\left(1 - \frac{\delta}{2} N\sigma_x^2\right) = E_{\min} + \frac{N}{2\delta}\sigma_{nS}^2 \tag{4.96}$$

or, for small adaptation step size,

$$E_{RTnS} \approx E_{\min}\left(1 + \frac{\delta}{2} N\sigma_x^2\right) + \frac{N}{2\delta}\sigma_{nS}^2 \tag{4.97}$$

In this simplified expression for the residual output error power with a nonstationary reference signal, the contributions of the gradient misadjustment and the tracking error are well characterized. Clearly, there is an optimum for the adaptation step size, δ_{opt}, which is

$$\delta_{\text{opt}} = \frac{\sigma_{nS}}{\sigma_x \sqrt{E_{\min}}} \tag{4.98}$$

which corresponds to balanced contributions.

The above model is indeed sketchy, but it provides hints for the filter behavior in more complicated circumstances [13]. For example, an order 12 FIR adaptive predictor is applied to three different speech signals: (a) a male voice, (b) a female voice, and (c) unconnected words. The prediction gain is shown in Figure 4.8(a) for various adaptation step sizes. The existence of an optimal step size is clearly visible in each case.

The performance of adaptive filters can be significantly improved if the most crucial signal parameters can be estimated in real time. For the gradient algorithms the most important parameter is the input signal power, which determines the step size. If the signal power can be estimated, then the normalized LMS algorithm

$$H(n+1) = H(n) + \frac{\delta}{\sigma_x^2} X(n+1)e(n+1) \tag{4.99}$$

can be implemented. The most straightforward estimation σ_x^2 is $P_{x1}(n)$ given by

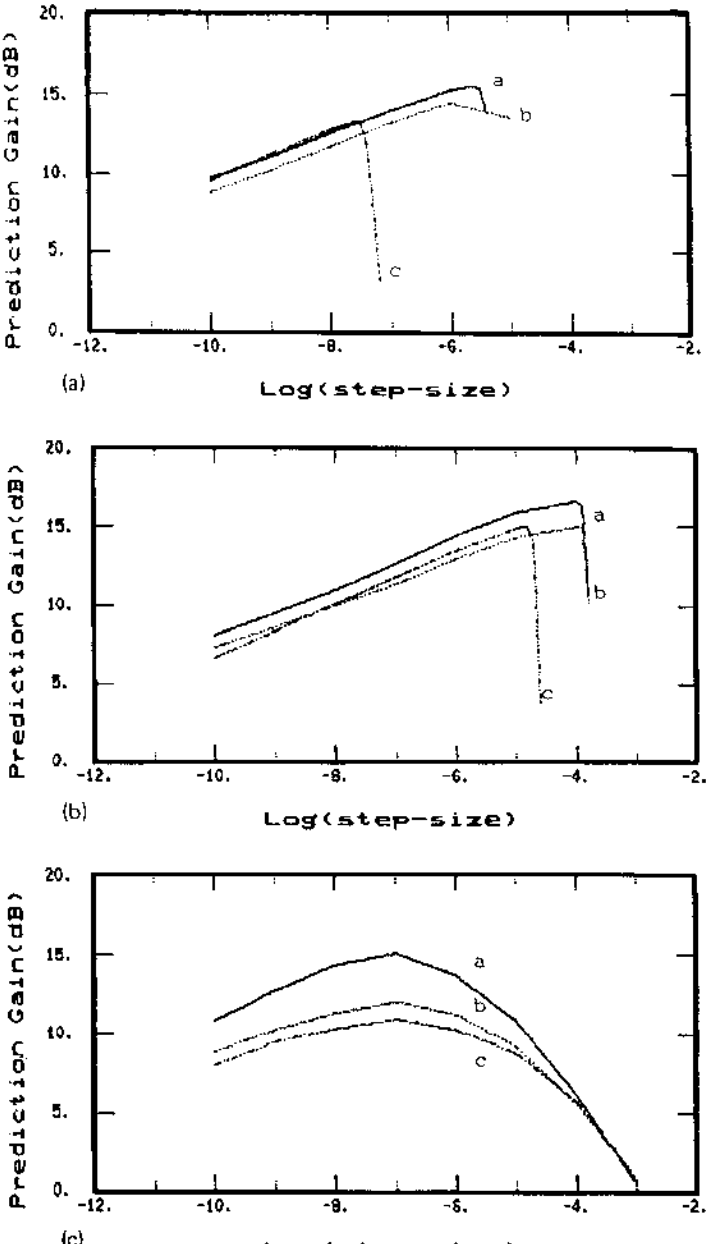


FIG. 4.8 Prediction gain vs. adaptation step size for three speech signals: (a) LMS with fixed step; (b) normalized LMS; (c) sign algorithm.

$$P_{x1}(n) = P_0 + \frac{1}{N_0} \sum_{i=0}^{N_0-1} x^2(n-i) \tag{4.100}$$

where P_0 is a positive constant which prevents division by zero. The parameter N_0, the observation time window, is the duration over which the signal can be assumed to be stationary.

For the prediction filter example mentioned above, the results corresponding to $P_0 = 0.5$ and $N_0 = 100$ (the long-term speech power is unity) are given in Figure 4.8(b). The improvements brought by normalization are clearly visible for all three sentences. The results obtained with the sign algorithm (4.87) are shown in Figure 4.8(c) for comparison purposes. The prediction gain is reduced, particularly for sentences b and c, but the robustness is worth pointing out: there is no steep divergence for too large δ, but a gradual performance degradation instead.

In practice, equation (4.100) is costly to implement, and the recursive estimate of Section 3.3 is preferred:

$$P_{x2}(n+1) = (1-\gamma)P_{x2}(n) + \gamma x^2(n+1) \tag{4.101}$$

Estimates (4.100) and (4.101) are additive. For faster reaction to rapid changes, exponential estimations can be worked out. An efficient and simple method to implement corresponds to a variable adaptation step size $\Delta(n)$ given by

$$\Delta(n) = \frac{\delta}{P_x(n)} = 2^{-I(n)} \tag{4.102}$$

where $I(n)$ is an integer variable, itself updated through an additive process (e.g., a sign algorithm [14]).

The step responses of $P_{x1}(n)$, $P_{x2}(n)$ and the exponential estimate are sketched in Figure 4.9. Better performance can be expected with the exponential technique for rapidly changing signals.

Adaptation step size normalization can also be achieved indirectly by reusing the data at each iteration.

The a posteriori error $\varepsilon(n+1)$ in equation (4.4) is calculated with the updated coefficients. It can itself be used to update the coefficients a second time, leading to a new error $\varepsilon_1(n+1)$. After K such iterations, the a posteriori error $\varepsilon_K(n+1)$ is

$$\varepsilon_K(n+1) = [1 - \delta X^t(n+1)X(n+1)]^{K+1} e(n+1) \tag{4.103}$$

For δ sufficiently small and K large, $\varepsilon_K(n+1) \approx 0$, which would have been obtained with a step size Δ satisfying

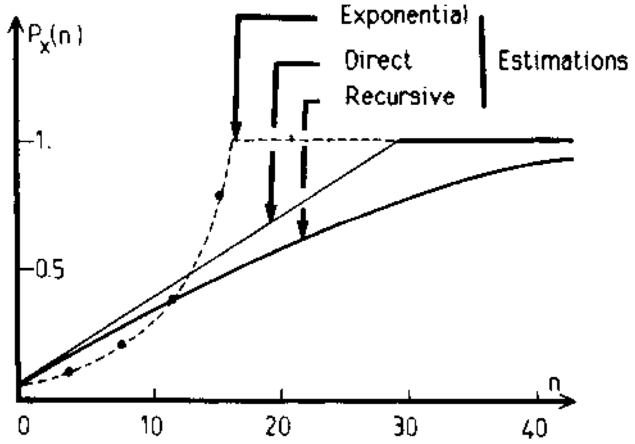


FIG. 4.9 Step responses of signal power estimations.

$$1 - \Delta X^t(n+1)X(n+1) = 0$$

that is

$$\Delta = \frac{1}{X^t(n+1)X(n+1)} \tag{4.104}$$

The equivalent step size corresponds to the fastest convergence defined in Section 4.4 by equation (4.42). So, the data reusing method can lead to fast convergence, while preserving the stability, in the presence of nonstationary signals.

The performance of normalized LMS algorithms can be studied as in the above sections, with the additional complication brought by the variable step size. For example, considering the so-called projection LMS algorithm

$$H(n+1) = H(n) + \frac{\delta}{X^t(n+1)X(n+1)} X(n+1)e(n+1) \tag{4.105}$$

one can show that a bias is introduced on the coefficients, which becomes independent of the step size for small values, while the variance remains proportional to δ [15].

A coarse approach to performance evaluation consists of keeping the results obtained for fixed step algorithms and considering the extreme parameter values.

4.9. DELAYED LMS ALGORITHMS

In the implementation, it can be advantageous to update the coefficients with some delay, say d sampling periods. For example, with integrated signal processors a delay $d = 1$ can ease programming. In these conditions it is interesting to investigate the effects of the updating delay on the adaptive filter performance [16].

The delayed LMS algorithm corresponds to the equation

$$H(n+1) = H(n) + \delta X(n+1-d)e(n+1-d) \tag{4.106}$$

The developments of Section 4.3 can be carried out again based on the above equation. For the sake of brevity and conciseness, a simplified analysis is performed here, starting from equation (4.24), rewritten as

$$[\alpha(n+1)] = [\alpha(n)] - \delta M X(n+1-d)e(n+1-d) \tag{4.107}$$

Substituting (4.26) in this equation and taking the expectation yields, under the hypotheses of Section 4.3,

$$E\{[\alpha(n+1)]\} = E\{[\alpha(n)]\} - \delta \operatorname{diag}(\lambda_i)E\{[\alpha(n-d)]\} \tag{4.108}$$

The system is stable if the roots of the characteristic equation

$$r^{d+1} - r^d + \delta\lambda_i = 0 \tag{4.109}$$

are inside the unit circle in the complex plane. Clearly, for $d = 0$, the condition is

$$0 < \delta < \frac{2}{\lambda_{\max}} \tag{4.110}$$

which is a stability condition sometimes used for the conventional LMS algorithms, less stringent than (4.7).

When $d = 1$, the stability condition is

$$0 < \delta < \frac{1}{\lambda_{\max}} \tag{4.111}$$

which implies that delay makes the stability condition more stringent. If δ is small enough ($\delta < \frac{1}{4}\lambda_{\max}$), the roots of the second-order characteristic equation are real:

$$r_1 \approx 1 - \delta\lambda_i(1+\delta\lambda_i), \quad r_2 \approx \delta\lambda_i(1+\delta\lambda_i) \tag{4.112}$$

The corresponding digital filter can be viewed as a cascade of two first-order sections, whose time constants can be calculated; its step response is approximately proportional to $1 - (1+\delta\lambda_i)r_1^n$, where the factor $1 + \delta\lambda_i$ reflects the effect of the root r_2. However, neglecting the root r_2, we can state that, for small adaptation step sizes, the adaptation speed of the delayed algorithm is similar to that of the conventional gradient algorithm. In the context of this simplified analysis, the time constant τ_i for each mode is roughly

$$\tau_i \approx \frac{1}{\delta\lambda_i} \tag{4.113}$$

Now, for $d \geqslant 2$, the characteristic equation (4.109) has a root on the unit circle if

$$e^{j(d+1)\omega} - e^{jd\omega} + \delta\lambda_i = 0 \tag{4.114}$$

The imaginary part of the equation is

$$\sin(d+1)\omega - \sin d\omega = 0 \tag{4.115}$$

whose solutions are

$$\omega = 0; \qquad (2d+1)\omega = (2k+1)\pi \quad (-d \leqslant k \leqslant d)$$

As concerns the real part, it provides the equality

$$\delta\lambda_i = 2(-1)^k \sin\frac{2k+1}{2(2d+1)}\pi \tag{4.116}$$

At this stage, the root locus technique can be employed. If $\delta\lambda_i$ is increased from zero, the first value which corresponds to a root of the equation is obtained for $k = 0$ and $k = -1$, and

$$\omega = \pi/2(2d+1)$$

The stability is guaranteed if $\delta\lambda_i$ remains smaller than the limit above. Hence the stability condition

$$0 < \delta < \frac{2}{\lambda_{\max}} \sin\frac{\pi}{2(2d+1)} \tag{4.117}$$

For large d, the condition simplifies to

$$0 < \delta < \frac{1}{\lambda_{\max}} \frac{\pi}{2d+1} \tag{4.118}$$

Turning to the excess output MSE, a first estimation can be obtained by considering only the largest root of the characteristic equation and assuming that the delayed LMS is equivalent to the conventional LMS with a slightly larger adaptation step. For $d = 1$, referring to equation (4.112), we can take the multiplying factor to be $1 + \delta\lambda_{\max}$. The most adverse situation for delayed LMS algorithms is the presence of nonstationary signals, because the tracking error can grow substantially.

4.10. THE MOMENTUM ALGORITHM

The momentum algorithm is an alternative approach to improve on the performance of the gradient algorithm, while sacrificing little in computational complexity.

The starting point is the recursive equation for the output error energy in the least squares approach. In Chapter 6, it will be shown that the following equation holds:

$$E(n+1) = WE(n) + e(n+1)\varepsilon(n+1) \tag{4.119}$$

where W is the weighting factor ($0 < W < 1$). Assuming that the coefficient vector is updated proportionally to the gradient of the error energy $E(n+1)$ and approximating the "a posteriori" error $\varepsilon(n+1)$ by the "a priori" error $e(n+1)$, the momentum algorithm is obtained:

$$\begin{aligned} e(n+1) &= y(n+1) + H^t(n)X(n+1) \\ H(n+1) &= H(n) + \alpha[H(n) - H(n-1)] + \delta e(n+1)X(n+1) \end{aligned} \tag{4.120}$$

The scalar α is called the momentum factor, by analogy with the use of the term in mechanics. An obvious condition for stability is $|\alpha| < 1$. In fact, the stability of the momentum algorithm can be investigated in a way similar to that of the gradient algorithm. The evolution of the coefficients is governed by the equation

$$\begin{aligned} H(n+1) = {} & [I_N(1+\alpha) - \delta X(n+1)X^t(n+1)]H(n) \\ & + \delta y(n+1)X(n+1) - \alpha H(n-1) \end{aligned} \tag{4.121}$$

Replacing $X(n+1)X^t(n+1)$ by $I_N N\sigma_x^2$, to take a conservative approach, the second-order characteristic equation of the system has its roots inside the unit circle if

$$|1 + \alpha - \delta N\sigma_x^2| < 1 + \alpha, \qquad \alpha < 1 \tag{4.122}$$

which leads to the stability conditions

$$0 < \delta < \frac{2(1+\alpha)}{N\sigma_x^2}, \alpha < 1 \tag{4.123}$$

The performance of the algorithm can be evaluated by following a procedure similar to that of the standard gradient algorithm, but with increased complexity. However, considering that the momentum term introduces a first-order difference equation with factor α, a coarse assessment of the algorithm's behavior is obtained by replacing δ by $\delta/(1-\alpha)$ in the expressions obtained for the gradient algorithm. For example, this accounts for the gain in convergence time observed in simulations [17].

4.11. VARIABLE STEP SIZE ADAPTIVE FILTERING

The performance of gradient adaptive filters is a compromise between speed of convergence and accuracy. A large step size makes the adaptation fast, while a small value can make the residual error close to the minimum. Therefore, a variable step size can offer a potential for improvement, and a possible approach is to apply the gradient algorithm to the step size itself [18].

Assuming a time-varying step size, the filter output error can be expressed by

$$e(n+1) = y(n+1) - [H(n-1) + \delta(n)e(n)X(n)]^t X(n+1) \tag{4.124}$$

The step size $\delta(n)$ can be updated with the help of the derivative of $e^2(n+1)$ with respect to δ. At time $(n+1)$, the following operations have to be carried out:

$$\begin{aligned} e(n+1) &= y(n+1) - H^t(n)X(n+1) \\ \delta(n+1) &= \delta(n) + \rho e(n+1)e(n)X^t(n)X(n+1) \\ H(n+1) &= H(n) + \delta(n+1)e(n+1)X(n+1) \end{aligned} \tag{4.125}$$

The above equations define a variable-step-size gradient algorithm, and the parameter ρ is a real positive scalar that controls the step size variations. To figure out the evolution of the step size, its updating equation can be rewritten as

$$\begin{aligned} \delta(n+1) = {} & [1 - \rho e^2(n)[X^t(n)X(n+1)]^2]\delta(n) \\ & + \rho[y(n+1) - H^t(n-1)X(n+1)]e(n)X^t(n)X(n+1) \end{aligned} \tag{4.126}$$

Clearly, the step size $\delta(n)$ decreases as the filter converges, and its mean value stabilizes at a limit which is determined by the correlation of the input signal and the correlation of the residual error.

4.12. CONSTRAINED LMS ALGORITHMS

The adaptive filters considered so far use a reference signal to compute the output error, which serves to update the coefficients. It might happen that this reference signal is zero, as in linear prediction. In such a situation, at least one constraint must be imposed on the coefficients, to prevent the trivial solution of all the coefficients being null. In linear prediction, it is the first coefficient which is a one. Another example has been given in Section 3.10 with the iterative calculation of the coefficients of an eigenfilter.

The case of a set of K independent linear constraints can be dealt with by forming a reference signal from the input signal and the constraints, as shown in Figure 4.10. The system is defined by the equations

$$\begin{aligned} e(n+1) &= H^t(n)X(n+1) \\ C^t H(n) &= F \end{aligned} \tag{4.127}$$

The matrix C is formed by the K constraint vectors, F being a K-element vector which is part of the constraint system. Now, a reference signal $y_q(n)$ can be formed from the input signal with the help of the coefficient vector WQ defined by

$$WQ = C[C^t C]^{-1} F \tag{4.128}$$

The matrix WS is orthogonal to the constraint vector and it has the rank $N - K$. The adaptive filter $Ha(z)$ has $N - K$ coefficients, which are updated according to the LMS algorithm [19].

The constraints may also come as an addition to an adaptive filter with a reference signal. Then the coefficients must be updated in a space which is orthogonal to the constraint space. The algorithm is as follows

$$\begin{aligned} e(n+1) &= y(n+1) - H^t(n)X(n+1) \\ H(n+1) &= P[H(n) + \delta e(n+1)X(n+1)] + m \end{aligned} \tag{4.129}$$

with

$$P = I_N - C[C^t C]^{-1} C^t \qquad m = C[C^t C]^{-1} F$$

The derivation of the equations (4.128) and (4.129) is obtained through the Lagrange multiplier technique, which is detailed in Chapter 7, in the context of least squares adaptive filtering.

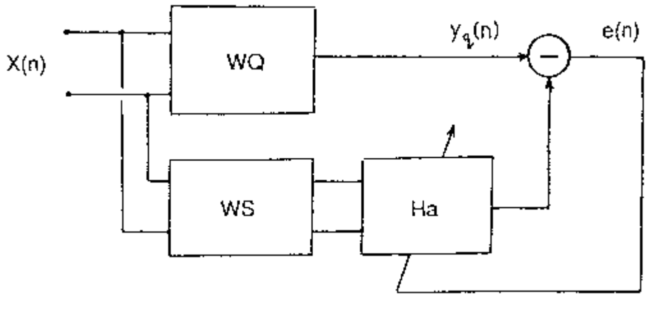


FIG. 4.10 Constrained adaptive filter.

4.13. THE BLOCK LMS ALGORITHM

In some applications, it can be convenient to perform the coefficient adaptation less often than each sampling period. In block adaptive filtering, the data sequences are arranged into blocks of length L and adaptation is carried out only once per block.

Let $X_{NL}(m)$ denote the $N \times L$-element input signal matrix associated with block m and $[y(m)]$ and $[e(m)]$ represent the L-element vectors of reference signal and output error respectively. Then, the block LMS algorithm is defined by the set of equations

$$\begin{aligned} e(m+1)] &= [y(m+1)] - X_{NL}^t(m+1)H(m) \\ H(m+1) &= H(m) + \delta \frac{1}{L} X_{NL}(m+1)[e(m+1)] \end{aligned} \tag{4.130}$$

The evolution of the N-element coefficient vector $H(m)$ is determined by substituting the error equation into the updating equation, to yield

$$\begin{aligned} H(n+1) = {} & [I_N - \delta \frac{1}{L} X_{NL}(m+1) X_{NL}^t(m+1)]H(m) \\ & + \frac{\delta}{L} X_{NL}(m+1)[y(m+1)] \end{aligned} \tag{4.131}$$

The important point here is that the data are averaged. For L sufficiently large, the following approximation is valid:

$$X_{NL}(m+1)X_{NL}^t(m+1) \approx LR_{xx} \tag{4.132}$$

Thus the stability condition for the step size δ is

$$0 < \delta < \frac{2}{\lambda_{\max}} \tag{4.133}$$

If the input signal is close to a white noise, the adaptation time constant, expressed in terms of the data period, is

$$\tau = L \frac{1}{\delta \sigma_x^2} \tag{4.134}$$

where σ_x^2 is the input signal power, as usual. As concerns the residual error power, it is not necessary to go through all the equations to assess the impact of the block processing. The averaging operation carried out on the driving term in the equation which gives the evolution of the coefficients (4.131) produces a reduction of the error variance by the averaging factor L. Thus, the residual error power can be expressed by

$$E_R = E_{\min} \frac{1}{1 - \frac{\delta}{2}\frac{1}{L} N \sigma_x^2} \tag{4.135}$$

Compared to the standard LMS algorithm, it appears that the block algorithm is slower but has a smoother operation. Also, it cannot track changes in the data sequence which are limited to a single block.

It must be pointed out that some advantages in implementation can be gained from the block processing of the data.

4.14. FIR FILTERS IN CASCADE FORM

In certain applications it is important to track the roots of the adaptive filter z-transfer function—for instance, for stability control if the inverse system is to be realized. It is then convenient to design the filter as a cascade of L second-order sections $H_l(z)$, $1 \leqslant l \leqslant L$, such that

$$H_l(z) = 1 + h_{1l} z^{-1} + h_{2l} z^{-2}$$

For real coefficients, if the roots z_l are complex, then

$$h_{1l} = 2\mathrm{Re}(z_l), \quad h_{2l} = |z_l|^2 \tag{4.136}$$

The roots are inside the unit circle if

$$|h_{2l}| < 1, \quad |h_{1l}| < 1 + h_{2l}, \quad 1 \leqslant l \leqslant L \tag{4.137}$$

The filter transfer function is

$$H(z) = \prod_{l=1}^{L} (1 + h_{1l} z^{-1} + h_{2l} z^{-2})$$

The error gradient vector is no longer the input data vector, and it must be calculated.

The filter output sequence can be obtained from the inverse z-transform

$$\tilde{y}(n) = \frac{1}{2\pi j} \int_\Gamma z^{n-1} \prod_{l=1}^{L} (1 + h_{1l} z^{-1} + h_{2l} z^{-2}) X(z)\, dz \tag{4.138}$$

where Γ is a suitable integration contour. Hence

$$\begin{aligned}\frac{\partial e(n+1)}{\partial h_{ki}} &= -\frac{\partial \tilde{y}(n+1)}{\partial h_{ki}} \\ &= -\frac{1}{2\pi j} \int_\Gamma z^n z^{-k} \prod_{\substack{l=1 \\ l \neq i}}^{L} (1 + h_{1l} z^{-1} + h_{2l} z^{-2}) X(z)\, dz\end{aligned}$$

or, more concisely,

$$\frac{\partial e(n+1)}{\partial h_{ki}} = -\frac{1}{2\pi j}\int_{\Gamma} z^n z^{-k}\frac{H(z)}{1+h_{1i}z^{-1}+h_{2i}z^{-2}}X(z)\,dz \tag{4.139}$$

Therefore, to form the gradient term $g_{ki}(n) = \partial e(n)/\partial h_{ki}$, it is sufficient to apply the filter output $\tilde{y}(n)$ to a purely recursive second-order section, whose transfer function is just the reciprocal of the section with index i. The recursive section has the same coefficients, but with the opposite sign. The corresponding diagram is given in Figure 4.11.

The coefficients are updated as follows:

$$h_{ki}(n+1) = h_{ki}(n) + \delta e(n+1)g_{ki}(n+1), \quad k = 1,2,\ 1 \leqslant i \leqslant L \tag{4.140}$$

The filter obtained in this way is more complicated than the transversal FIR filter, but it offers a simple method of finding and tracking the roots, which, due to the presence of the recursive part, should be inside the unit circle in the z-plane to ensure stability [20].

However, there are some implementation problems, because the individual sections have to be characterized for the filter to work properly. That can be achieved by imposing different initial conditions or by separating the zero trajectories in the z-plane.

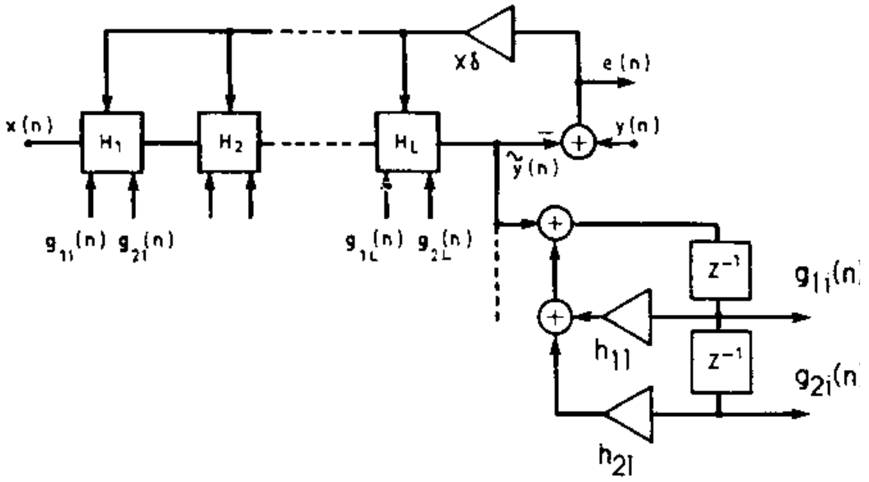


FIG. 4.11 Adaptive FIR filter in cascade form.

4.15. IIR GRADIENT ADAPTIVE FILTERS

In general, IIR filters achieve given minimum phase functions with fewer coefficients than their FIR counterparts. Moreover, in some applications, it is precisely an IIR function that is looked for. Therefore, IIR adaptive filters are an important class, particularly useful in modeling or identifying systems [21].

The output of an IIR filter is

$$\tilde{y}(n) = \sum_{l=0}^{L} a_l x(n-l) + \sum_{k=1}^{K} b_k \tilde{y}(n-k) \tag{4.141}$$

The elements of the error gradient vector are calculated from the derivatives of the filter output:

$$\frac{\partial \tilde{y}(n)}{\partial a_l} = x(n-l) + \sum_{k=1}^{K} b_k \frac{\partial \tilde{y}(n-k)}{\partial a_l}, \quad 0 \leqslant l \leqslant L \tag{4.142}$$

and

$$\frac{\partial \tilde{y}(n)}{\partial b_k} = \tilde{y}(n-k) + \sum_{i=1}^{K} b_i \frac{\partial \tilde{y}(n-i)}{\partial b_k}, \quad 1 \leqslant k \leqslant K \tag{4.143}$$

To show the method of realization, let us consider the z-transfer function

$$H(z) = \frac{\sum\limits_{l=0}^{L} a_l z^{-1}}{1 - \sum\limits_{k=1}^{K} b_k z^{-k}} = \frac{N(z)}{D(z)} \tag{4.144}$$

The filter output can be written

$$\tilde{y}(n) = \frac{1}{2\pi j} \int_{\Gamma} z^{n-1} H(z) X(z)\, dz$$

Consequently

$$\frac{\partial \tilde{y}(n)}{\partial a_l} = \frac{1}{2\pi j} \int_{\Gamma} z^{n-1} z^{-1} \frac{X(z)}{D(z)}\, dz \tag{4.145}$$

$$\frac{\partial \tilde{y}(n)}{\partial b_k} = \frac{1}{2\pi j} \int_{\Gamma} z^{n-1} z^{-k} \frac{1}{D(z)} H(z) X(z)\, dz \tag{4.146}$$

The gradient is thus calculated by applying $x(n)$ and $\tilde{y}(n)$ to the circuits corresponding to the transfer function $\frac{1}{D(z)}$.

To simplify the implementation, the second terms in (4.142) and (4.143) can be dropped, which leads to the following set of equations for the adaptive filter (in vector notation):

$$e(n+1) = y(n+1) - [A^t(n), B^t(n)]\begin{bmatrix} X(n+1) \\ \tilde{Y}(n) \end{bmatrix} \tag{4.147}$$

$$\begin{bmatrix} A(n+1) \\ B(n+1) \end{bmatrix} = \begin{bmatrix} A(n) \\ B(n) \end{bmatrix} + \delta \begin{bmatrix} X(n+1) \\ \tilde{Y}(n) \end{bmatrix} e(n+1) \tag{4.148}$$

The approach is called the output error technique. The block diagram is shown in Figure 4.12(a). The filter is called a parallel IIR gradient adaptive filter.

The analysis of the performance of such a filter is not simple, because of the vector $\tilde{Y}(n)$ of the most recent filter output data in the system equations. To begin with, the stability can only be ensured if the error sequence $e(n)$ is

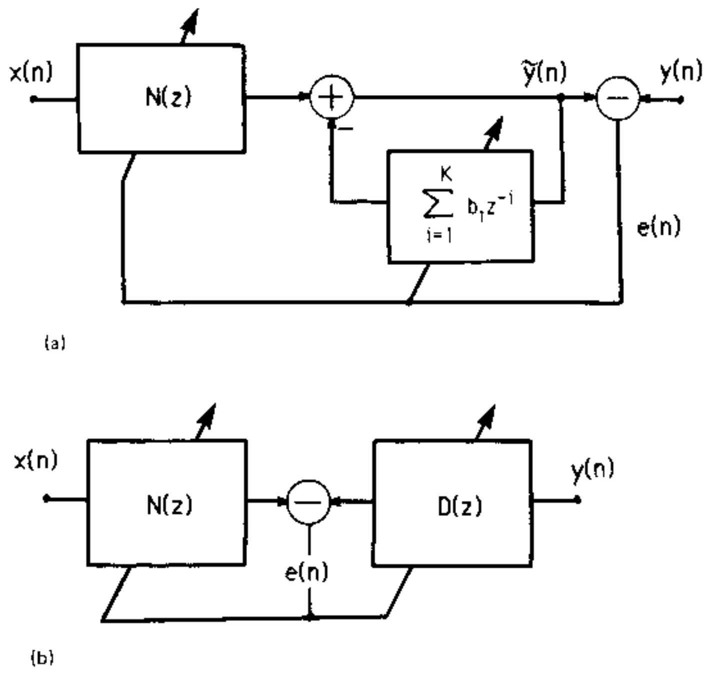


FIG. 4.12 Simplified gradient IIR adaptive filters: (a) Parallel type (output error); (b) series-parallel type (equation error).

filtered by a z-transfer function $C(z)$, such that the function $C(z)/D(z)$ be strictly positive real, which means

$$\mathrm{Re}\left[\frac{C(z)}{D(z)}\right] > 0, \quad |z| = 1 \tag{4.149}$$

An obvious choice is $C(z) = D(z)$.

An alternative approach to get realizable IIR filters is based on the observation that, after convergence, the error signal is generally small and the filter output $\tilde{y}(n)$ is close to the reference $y(n)$. Thus, in the system equations, the filter output vector can be replaced by the reference vector:

$$e(n+1) = y(n+1) - [A^t(n), B^t(n)]\begin{bmatrix} X(n+1) \\ \tilde{Y}(n) \end{bmatrix} \tag{4.150}$$

$$\begin{bmatrix} A(n+1) \\ B(n+1) \end{bmatrix} = \begin{bmatrix} A(n) \\ B(n) \end{bmatrix} + \delta \begin{bmatrix} X(n+1) \\ Y(n) \end{bmatrix} e(n+1) \tag{4.151}$$

This is the equation error technique. The filter is said to be of the series-parallel type; its diagram is shown in Figure 4.12(b). Now, only FIR filter sections are used, and there is no fundamental stability problem anymore. The performance analysis can be carried out as in the above sections. The stability bound for the adaptation step size is

$$0 < \delta < \frac{2}{L\sigma_x^2 + K\sigma_y^2} \tag{4.152}$$

Overall the performance of the series-parallel IIR gradient adaptive filter can be derived from that of the FIR filter by changing $N\sigma_x^2$ into $L\sigma_x^2 + K\sigma_y^2$.

In order to compare the performance of the parallel type and series-parallel approaches, let us consider the expectation of the recursive coefficient vector after convergence, B_∞, for the parallel case. Equations (4.147) and (4.148) yield

$$B_\infty = E[\tilde{Y}(n)\tilde{Y}^t(n)]^{-1} E\{\tilde{Y}(n)[y(n+1) - A^t(n)X(n+1)]\} \tag{4.153}$$

The parallel-series type yields a similar equation, but with $E[Y(n)Y^t(n)]^{-1}$; if the output error is approximated by a white noise with power σ_e^2, then

$$E[Y(n)Y^t(n)] = \sigma_e^2 I_N + E[\tilde{Y}(n)\tilde{Y}^t(n)] \tag{4.154}$$

and a bias is introduced on the recursive coefficients. The above equation clearly illustrates the stability hazards associated with using $\tilde{Y}(n)$, because the matrix can become singular. Therefore, the residual error is larger with the parallel-series approach, while the adaptation speed is not significantly modified, particularly for small step sizes, because the initial error sequences are about the same for both types.

Finally, several structures are available, and IIR gradient adaptive filters can be an attractive alternative to FIR filters in relevant applications.

4.16. NONLINEAR FILTERING

The digital filters considered up to now have been linear filters, which means that the output is a linear function of the input data. We can have a nonlinear scalar function of the input data vector:

$$\tilde{y}(n) = f[X(n)] \tag{4.155}$$

The Taylor series expansion of the function $f(X)$ about the vector zero is

$$f(X) = \sum_{k=0}^{\infty} \frac{1}{k!} \left[\sum_{i=1}^{N} x_i \frac{\partial}{\partial x_i} \right]^k f(X) \tag{4.156}$$

with differential operator notation. When limited to second order, the expansion is

$$\tilde{y}(n) = y_0 + H^t X(n) + \text{trace}(M X(n) X^t(n)) \tag{4.157}$$

where y_0 is a constant, H is the vector of the linear coefficients, and M is the square matrix of the quadratic coefficients, the filter length N being the number of elements of the data vector $X(n)$. This nonlinear filter is called the second-order Volterra filter (SVF) [22].

The quadratic coefficient matrix M is symmetric because the data matrix $X(n)X^t(n)$ is symmetric. Also, if the input and reference signals are assumed to have zero mean, $\tilde{y}(n)$ must also have zero mean, which implies

$$E[\tilde{y}(n)] = y_0 + \text{trace}(MR) \tag{4.158}$$

Therefore (4.157) can be rewritten as

$$\tilde{y}(n) = H^t X(n) + \text{trace}(M[X(n)X^t(n) - R]) \tag{4.159}$$

When this structure is used in an adaptive filter configuration, the coefficients must be calculated to minimize the output MSE, $E\{(y(n) - \tilde{y}(n))^2\}$.

For Gaussian signals, the optimum coefficients are

$$\begin{aligned} H_{\text{opt}} &= R^{-1} E[y(n)X(n)] \\ M_{\text{opt}} &= \tfrac{1}{2} R^{-1} E[y(n)X(n)X^t(n)] R^{-1} \end{aligned} \tag{4.160}$$

It is worth pointing out that the linear operator of the optimum SVF, in these conditions, is exactly the optimum linear filter. Thus, the nonlinear filter can be constructed by adding a quadratic section in parallel to the linear filter, as shown in Figure 4.13.

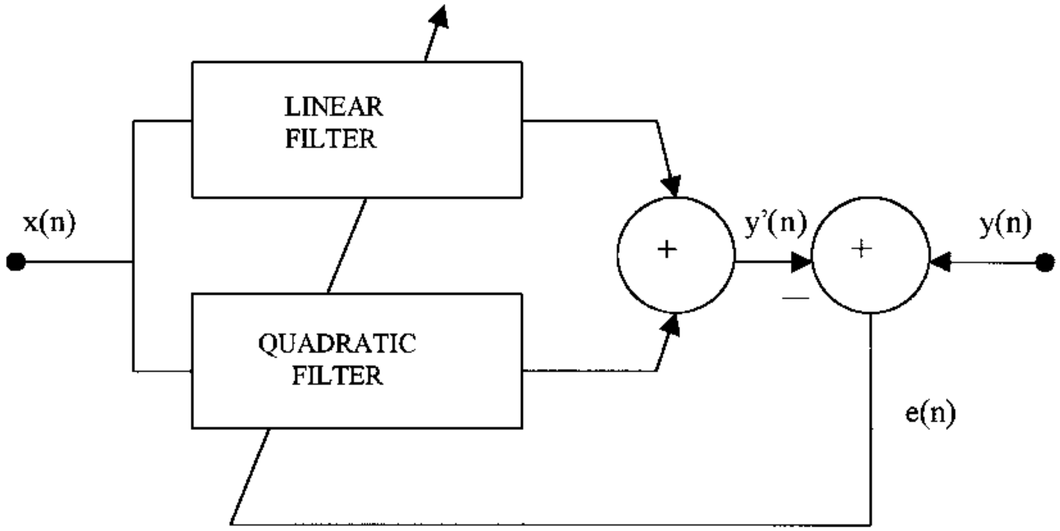


FIG. 4.13 Second-order nonlinear filter for Gaussian signals.

The minimum output MSE is

$$E_{\min} = E[y^2(n)] - E[y(n)X(n)]^t R^{-1} E[y(n)X(n)] \\ - \tfrac{1}{2}\operatorname{trace}(R^{-1}E[y(n)X(n)X^t(n)]R^{-1}E[y(n)X(n)X^t(n)]) \tag{4.161}$$

The gradient techniques can be implemented by calculating the derivatives of the output error with respect to the coefficients. The gradient adaptive SVF equations are

$$\begin{aligned} e(n+1) &= y(n+1) - H^t(n)X(n+1) \\ &\qquad - \operatorname{trace}(M(n)[X(n+1)X^t(n+1) - R]) \\ H(n+1) &= H(n) + \delta_h X(n+1)e(n+1) \\ M(n+1) &= M(n) + \delta_m X(n+1)X^t(n+1)e(n+1) \end{aligned} \tag{4.162}$$

where δ_h and δ_m are the adaptation steps.

The zeroth-order term trace$(M(n)R)$ is not constant in the adaptive implementation. It can be replaced by an estimate of the mean value of the quadratic section output, for example, using the recursive estimator of Section 3.3.

The stability bounds for the adaptation steps can be obtained as in Section 4.2 by considering the a posteriori error $\varepsilon(n+1)$:

$$\varepsilon(n+1) = e(n+1)[1 - \delta_h X^t(n+1)X(n+1) \\ - \delta_m \operatorname{trace}(X(n+1)X^t(n+1)[X(n+1)X^t(n+1) - R])] \tag{4.163}$$

Assuming that the linear operator acts independently, we adopt condition (4.7) for δ_h. Now, the stability condition for δ_m is

$$|1 - \delta_m(\operatorname{trace} E[X(n+1)X^t(n+1)X(n+1)X^t(n+1)] - \operatorname{trace} R^2)| < 1$$

The following approximation can be made:

$$\operatorname{trace} E[X(n+1)X^t(n+1)X(n+1)X^t(n+1)] \approx (N\sigma_x^2)^2 > \operatorname{trace} R^2 \tag{4.164}$$

Hence, we have the stability condition

$$0 < \delta_m < \frac{2}{(N\sigma_x^2)^2} \tag{4.165}$$

The total output error is the sum of the minimum error $E_{\min}$ given by (4.140) and the excess MSEs of the linear and quadratic sections. Using developments as in Section 4.3, one can show the excess MSE of the quadratic section E_M can be approximated by

$$E_M \approx \frac{\delta_m}{8} E_{\min}[(N\sigma_x^2)^2 + 2 \text{ trace } R^2] \tag{4.166}$$

In practice, the quadratic section in general serves as a complement to the linear section. Indeed the improvement must be worth the price paid in additional computational complexity [23].

4.17. STRENGTHS AND WEAKNESSES OF GRADIENT FILTERS

The strong points of the gradient adaptive filters, illustrated throughout this chapter, are their ease of design, their simplicity of realization, their flexibility, and their robustness against signal characteristic evolution and computation errors.

The stability conditions have been derived, the residual error has been estimated, and the learning curves have been studied. Simple expressions have been given for the stability bound, the residual error, and the time constant in terms of the adaptation step size. Word-length limitation effects have been investigated, and estimates have been derived for the coeffficient and internal data word lengths as a function of the specifications. Useful variations from the classical LMS algorithm have been discussed. In short, all the knowledge necessary for a smart and successful engineering application has been provided.

Although gradient adaptive filters are attractive, their performance is severely limited in some applications. Their main weakness is their dependence on signal statistics, which can lead to low speed or large residual errors. They give their best results with flat spectrum signals, but if the signals have a fine structure they can be inefficient and unable, for example, to perform simple analysis tasks. For these cases LS adaptive filters offer an attractive solution.

EXERCISES

1. A sinusoidal signal $x(n) = \sin(n\pi/2)$ is applied to a second-order linear predictor as in Figure 4.3. Calculate the theoretical ACF of the signal and the prediction coefficients. Verify that the zeros of the FIR prediction filter are on the unit circle at the right frequency.

   Using the LMS algorithm (4.3) with $\delta = 0.1$, show the evolution of the coefficients from time $n = 0$ to $n = 10$. How is that evolution modified if the MLAV algorithm (4.77) and the sign algorithm (4.87) are used instead.
2. A second-order adaptive FIR filter has the above $x(n)$ as input and

$$y(n) = x(n) + x(n-1) + 0.5x(n-2)$$

as reference signal. Calculate the coefficients, starting from zero initial values, from time $n = 0$ to $n = 10$. Calculate the theoretical residual error and the time constant and compare with the experimental results.

3. *Adaptive line enhancer*. Consider an adaptive third-order FIR predictor. The input signal is

$$x(n) = \sin(n\omega_0) + b(n)$$

where $b(n)$ is a white noise with power σ_b^2. Calculate the optimal coefficients $a_{i,\text{opt}}$, $1 \leqslant i \leqslant 3$. Give the noise power in the sequence

$$s(n) = \sum_{i=1}^{3} a_{i,\text{opt}} x(n-i)$$

as well as the signal power. Calculate the SNR enhancement.

The predictor is now assumed to be adaptive with step $\delta = 0.1$. Give the SNR enhancement.

4. In a transmission system, an echo path is modeled as an FIR filter, and an adaptive echo canceler with 500 coefficients is used to remove the echo. At unity input signal power, the theoretical system gain, the echo attenuation, is 53 dB, and the time constant specification is 800 sampling periods. Calculate the range of the adaptation step size δ if the actual system gain specification is 50 dB.

Assuming the echo path to be passive, estimate the coefficient and internal data word lengths, considering that the power of the signals can vary in a 40-dB range.

5. An adaptive notch filter is used to remove a sinusoid from an input signal. The filter transfer function is

$$H(z) = \frac{1 + az^{-1} + z^{-2}}{1 + 0.9az^{-1} + 0.81z^{-2}}$$

Give the block diagram of the adaptive filter. Calculate the error gradient. Simplify the error gradient and give the coefficient updating equation. The signal $x(n) = \sin(n\pi/4)$ is fed to the filter from time zero on. For an initial coefficient value of zero what are the trajectories, in the z-plane, of the zeros and poles of the notch filter. Verify experimentally with $\delta = 0.1$.

6. An order 4 FIR predictor is realized as a cascade of two second-order sections. Show that only one section is needed to compute the error gradient and give the block diagram. What happens for any input signal if the filter is made adaptive and the initial coefficient values are zero. Now the predictor transfer function is

$$H(z) = (1 - az^{-1} + az^{-2})(1 + bz^{-1} + bz^{-2})$$

and the coefficients a and b are updated. Give the trajectories, in the z-plane, of the predictor zeros.

Calculate the maximum predicting gain for the signal $x(2p+1) = 1$, $x(2p) = 0$.

7. Give the block diagram of the gradient second-order Volterra adaptive filter according to equations (4.162). Evaluate the computational complexity in terms of numer of multiplications and additions per sampling period and point out the cost of the quadratic section.

REFERENCES

1. R. W. Lucky, ‘‘Techniques for Adaptive Equalization of Digital Communication Systems,’’ *Bell System Tech. J.* **45**, 255–286 (1966).
2. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, N.J., 1985.
3. W. A. Gardner, ‘‘Learning Characteristics of Stochastic Gradient Descent Algorithms: A General Study, Analysis and Critique,’’ in *Signal Processing*, No. 6, North-Holland, 1984, pp. 113–133.
4. O. Macchi, *Adaptive Processing: the LMS Approach with Applications in Transmission*, John Wiley and Sons, Chichester, UK, 1995.
5. L. L. Horowitz and K. D. Senne, ‘‘Performance Advantage of Complex LMS for Controlling Narrow-Band and Adaptive Arrays,’’ *IEEE Trans.* **CAS-28**, 562–576 (June 1981).
6. C. N. Tate and C. C. Goodyear, ‘‘Note on the Convergence of Linear Predictive Filters, Adapted Using the LMS Algorithm,’’ *IEE Proc.* **130**, 61–64 (April 1983).
7. M. S. Mueller and J. J. Werner, Adaptive Echo Cancellation with Dispersion and Delay in the Adjustment Loop,’’ *IEEE Trans.* **ASSP-33**, 520–526 (June 1985).
8. C. Caraiscos and B. Liu, ‘‘A Round-Off Error Analysis of the LMS Adaptive Algorithm,’’ *IEEE Trans.* **ASSP-32**, 34–41 (February 1984).
9. A. Segalen and G. Demoment, ‘‘Constrained LMS Adaptive Algorithm,’’ *Electronics Lett.* **18**, 226–227 (March 1982).
10. A. Gersho, ‘‘Adaptive Filtering with Binary Reinforcement,’’ *IEEE Trans.* **IT-30**, 191–199 (March 1984).
11. T. Claasen and W. Mecklenbrauker, ‘‘Comparison of the Convergence of Two Algorithms for Adaptive FIR Digital Filters,’’ *IEEE Trans.* **ASSP-29**, 670–678 (June 1981).
12. N. J. Bershad, ‘‘On the Optimum Data Non-Linearity in LMS Adaptation,’’ *IEEE Trans.* **ASSP-34**, 69–76 (February 1986).
13. B. Widrow, J. McCool, M. Larimore, and R. Johnson, ‘‘Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter,’’ *Proc. IEEE* **64**, 151–1162 (August 1976).

14. D. Mitra and B. Gotz, "An Adaptive PCM System Designed for Noisy Channels and Digital Implementation," *Bell System Tech. J.* **57**, 2727–2763 (September 1978).
15. S. Abu E. Ata, "Asymptotic Behavior of an Adaptive Estimation Algorithm with Application to *M*-Dependent Data," *IEEE Trans.* **AC-27**, 1225–1257 (December 1981).
16. J. W. M. Bergmans, "Effect of Loop Delay on the Stability of Discrete Time PLL," *IEEE Trans.* **CAS-42**, 229–231 (April 1995).
17. S. Roy and J. J. Shynk, "Analysis of the Momentum LMS Algorithm," *IEEE Trans.* **ASSP-38**, 2088–2098 (December 1990).
18. J. B. Evans, P. Xue, and B. Liu, "Analysis and Implementation of Variable Step Size Adaptive Algorithms," *IEEE Trans.* **ASSP-41**, 2517–2535 (August 1993).
19. L. J. Griffiths and K. M. Buckley, "Quiescent Pattern Control in Linearly Constrained Adaptive Arrays," *IEEE Trans.* **ASSP-35**, 917–926 (July 1987).
20. L. B. Jackson and S. L. Wood, "Linear Prediction in Cascade form," *IEEE Trans. ASSP-26*, 578–528 (December 1978).
21. P. A. Regalia, *Adaptive IIR Filtering in Signal Processing and Control*, Marcel Dekker, Inc., New York, 1995.
22. T. Koh and E. Powers, "Second Order Volterra Filtering and Its Application to Non Linear System Identification," *IEEE Trans.* **ASSP-33**, 1445–1455 (December 1985).
23. V. J. Mathews and G. L. Sicuranza, "Polynomial Signal Processing," Wiley Interscience, 2000.

5 Linear Prediction Error Filters

Linear prediction error filters are included in adaptive filters based on FLS algorithms, and they represent a significant part of the processing. They crucially influence the operation and performance of the complete system. Therefore it is important to have a good knowledge of the theory behind these filters, of the relations between their coefficients and the signal parameters, and of their implementation structures. Moreover, they are needed as such in some application areas like signal compression or analysis [1].

5.1. DEFINITION AND PROPERTIES

Linear prediction error filters form a class of digital filters characterized by constraints on the coefficients, specific design methods, and some particular implementation structures.

In general terms, a linear prediction error filter is defined by its transfer function $H(z)$, such that

$$H(z) = 1 - \sum_{i=1}^{N} a_i z^{-1} \tag{5.1}$$

where the coefficients are computed so as to minimize a function of the output $e(n)$ according to a given criterion. If the output power is minimized, then the definition agrees wth that given in Section 2.8 for linear prediction.

When the number of coefficients N is a finite integer, the filter is a FIR type. Otherwise the filter is IIR type, and its transfer function often takes the form of a rational fraction:

$$H(z) = \frac{1 - \sum_{i=1}^{L} a_i z^{-i}}{1 - \sum_{i=1}^{M} b_i z^{-i}} \tag{5.2}$$

For simplicity, the same number of coefficients $N = L = M$ is often assumed in the numerator and denominator of $H(z)$, implying that some may take on zero values.

The block diagram of the filter associated with equation (5.2) is shown in Figure 5.1, where the recursive and the nonrecursive sections are represented.

As seen in Section 2.5, linear prediction corresponds to the modeling of the signal as the output of a generating filter fed by a white noise, and the linear prediction error filter transfer function in the inverse of the generating filter transfer function. Therefore, the linear prediction error filter associated with $H(z)$ in (5.2) is sometimes designated by extension as an ARMA (L, M) predictor, which means that the AR section of the signal model has L coefficients and the MA section has M coefficients.

For a stationary signal, the linear prediction coefficients can be calculated by LS techniques. A direct application of the general method presented in Section 1.4 yields the set of N equations:

$$\frac{\partial}{\partial a_j} E[e^2(n)] = r(j) - \sum_{i=1}^{N} a_i r(j-i) = 0, \quad 1 \leqslant j \leqslant N$$

which can be completed by the power relation (4.16)

$$E_{aN} = E[e^2(n)] = r(0) - \sum_{i=1}^{N} a_i r(i)$$

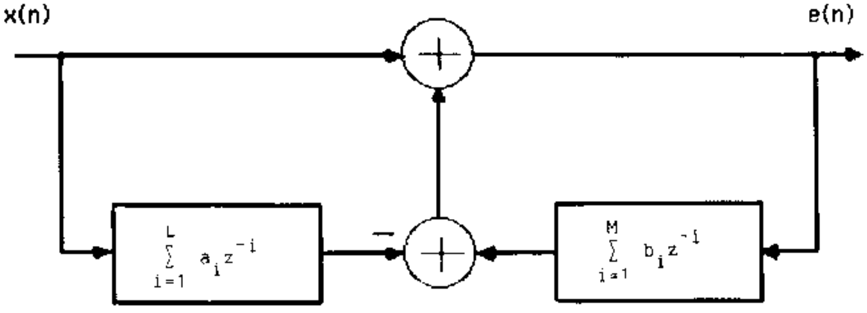


FIG. 5.1 IIR linear prediction error filter.

In concise form, the linear prediction matrix equation is

$$R_{N+1}\begin{bmatrix} 1 \\ -A_N \end{bmatrix} = \begin{bmatrix} E_{aN} \\ 0 \end{bmatrix} \tag{5.3}$$

where A_N is the N-element prediction coefficient vector and E_{aN} is the prediction error power. The $(N+1) \times (N+1)$ signal AC matrix, R_{N+1}, is related to R_N by

$$R_{N+1} = \left[\begin{array}{c|ccc} r(0) & r(1) & \cdots & r(N) \\ \hline r(1) & & & \\ \vdots & & & \\ r(N) & & R_N & \end{array}\right], \quad R_N = E[X(n)X^t(n)] \tag{5.4}$$

The linear prediction equation is also the AR modeling equation (2.63) given in Section 2.5.

The above coefficient design method is valid for any stationary signal. An alternative and illustrative approach can be derived, which is useful when the signal is made of determinist, or predictable, components in noise.

Let us assume that the input signal is

$$x(n) = s(n) + b(n) \tag{5.5}$$

where $s(n)$ is a useful signal with power spectral density $S(\omega)$ and $b(n)$ a zero mean white noise with power σ_b^2. The independence relation between the sequences $s(n)$ and $b(n)$ leads to

$$E_{aN} = E[e^2(n)] = \frac{1}{2\pi}\int_{-\pi}^{\pi} |H(\omega)|^2 S(\omega)\, d\omega + \sigma_b^2(1 + A_N^t A_N) \tag{5.6}$$

The factor $|H(\omega)|^2$ is a function of the prediction coefficients which can be calculated to minimize E_{aN} by setting to zero the derivatives of (5.6) with respect to the coefficients. The two terms on the right side of (5.6) can be characterized as the residual prediction error and the amplified noise, respectively. Indeed their relative values reflect the predictor performance and the degradation caused by the noise added to the useful signal.

If $E_{aN} = 0$, then there is no noise, $\sigma_b^2 = 0$, and the useful signal is predictable; in other words, it is the sum of at most N cisoids. In that case, the zeros of the prediction error filter are on the unit circle, at the signal frequencies, like those of the minimal eigenvalue filter. These filters are identidal, up to a constant factor, because the prediction equation

$$R_{N+1}\begin{bmatrix} 1 \\ -A_N \end{bmatrix} = 0 \tag{5.7}$$

is also an eigenvalue equation, corresponding to $\lambda_{\min} = 0$.

A characteristic property of linear prediction error filters is that they are minimum phase, as shown in Section 2.8; all of their zeros are within or on the unit circle in the complex plane.

As an illustration, first- and second-ordr FIR predictors are studied next.

5.2. FIRST- AND SECOND-ORDER FIR PREDICTORS

The transfer function of the first-order FIR predictor is

$$H(z) = 1 - az^{-1} \tag{5.8}$$

Indeed its potential is very limited. It can be applied to a constant signal in white noise with power σ_b^2:

$$x(n) = 1 + b(n)$$

The prediction error power is

$$E[e^2(n)] = |H(1)|^2 + \sigma_b^2(1 + a^2) = (1 - a)^2 + \sigma_b^2(1 + a^2) \tag{5.9}$$

Setting to zero the derivative of $E[e^2(n)]$ with respect to the coefficient a yields

$$a = \frac{1}{1 + \sigma_b^2} \tag{5.10}$$

The zero of the filter is on the real axis in the z-plane when $\sigma_b^2 = 0$ and moves away from the unit circle toward the origin when the noise power is increased.

The ratio of residual prediction error to amplified noise power is maximal for $\sigma_b^2 = \sqrt{2}$, which corresponds to a SNR ratio of -1.5 dB. Its maximum value is about 0.2, which means that the residual prediction error power is much smaller than the amplified noise power.

The transfer function of the second-order FIR predictor is

$$H(z) = 1 - a_1 z^{-1} - a_2 z^{-2} \tag{5.11}$$

It can be applied to a sinusoid in noise:

$$x(n) = \sqrt{2}\sin(n\omega_0) + b(n)$$

The prediction error power is

$$E[e^2(n)] = |H(\omega_0)|^2 + \sigma_b^2(1 + a_1^2 + a_2^2)$$

Hence,

$$a_1 = 2\cos\omega_0 \frac{\sin^2\omega_0 + \dfrac{\sigma_b^2}{2}}{\sin^2\omega_0 + \sigma_b^2(2+\sigma_b^2)} \tag{5.12}$$

and

$$a_2 = -1\left[1 - \sigma_b^2 \frac{1+\sigma_b^2 + 2\cos^2\omega_0}{(1+\sigma_b^2)^2 - \cos^2\omega_0}\right] \tag{5.13}$$

When the noise power vanishes, the filter zeros reach the unit circle in the complex plane and take on the values $e^{\pm j\omega_0}$. They are complex if $a_1^2 + 4a_2 < 0$, which is always verified as soon as $|\cos\omega_0| < \frac{\sqrt{2}}{2}$; that is, $\frac{\pi}{4} \leqslant \omega_0 \leqslant \frac{3\pi}{4}$. Otherwise the zeros are complex when the noise power is small enough. The noise power limit σ_{bL}^2 is the solution of the following third-degree equation in the variable $x = 1 + \sigma_b^2$:

$$x^3 + x^2 \frac{3\cos^2\omega_0}{8\cos^2\omega_0 - 4} - x\frac{3}{2}\cos^2\omega_0 + \frac{4\cos^6\omega_0 + \cos^2\omega_0}{8\cos^2\omega_0 - 4} = 0 \tag{5.14}$$

This equation has only one positive and real solution for the relevant values of the frequency ω_0. So, σ_{bL}^2 can be calculated; a simple approximation is [2]

$$\sigma_{bL}^2 \approx 1.33\omega_0^3 \quad (\omega_0 \text{ in radians}) \tag{5.15}$$

The trajectory of the zeros in the complex plane when the additive noise power varies is shown in Figure 5.2. When the noise power increases from zero, the filter zeros move from the unit circle on a circle centered at $+1$ and with radius approximately ω_0; beyond σ_{bL}^2 they move on the real axis toward the origin.

The above results are useful for the detection of sinusoids in noise.

5.3. FORWARD AND BACKWARD PREDICTION EQUATIONS

The linear prediction error is also called the process innovation to illustrate the fact that new information has become available. However, when a limited fixed number of data is handled, as in FIR or transversal filters, the oldest data sample is discarded every time a new sample is acquired. Therefore, to fully analyze the system evolution, one must characterize the loss of the oldest data sample, which is achieved by backward linear prediction.

The forward linear prediction error $e_a(n)$ is

$$e_a(n) = x(n) - \sum_{i=1}^{N} a_i x(n-i)$$

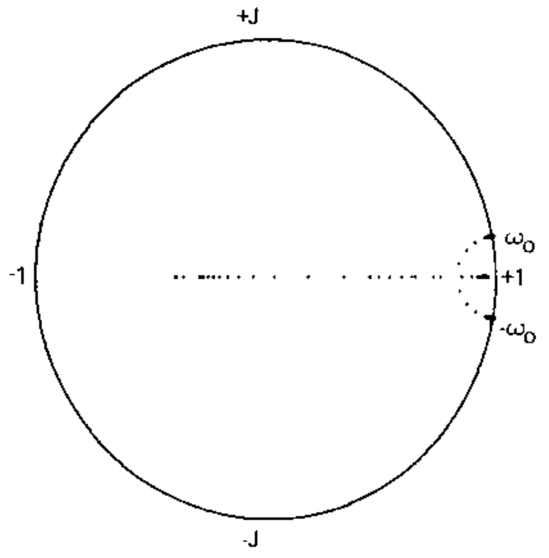


FIG. 5.2 Location of the zeros of a second-order FIR predictor applied to a sinusoid in noise with varying power.

or, in vector notation,

$$e_a(n) = x(n) - A_N^t X(n-1) \tag{5.16}$$

The backward linear prediction error $e_b(n)$ is defined by

$$e_b(n) = x(n-N) - B_N^t X(n) \tag{5.17}$$

where B_N is the vector of the backward coefficients. The two filters are shown in Figure 5.3.

The minimization of $E[e_b^2(n)]$ with respect to the coefficients yields the backward linear prediction matrix equation

$$R_{N+1} \begin{bmatrix} -B_N \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ E_{bN} \end{bmatrix} \tag{5.18}$$

Premultiplication by the co-identity matrix J_{N+1} gives

$$J_{N+1} R_{N+1} \begin{bmatrix} -B_N \\ 1 \end{bmatrix} = \begin{bmatrix} E_{bN} \\ 0 \end{bmatrix}$$

which, considering relation (3.57) in Chapter 3, yields

$$R_{N+1} \begin{bmatrix} 1 \\ -J_N B_N \end{bmatrix} = \begin{bmatrix} E_{bN} \\ 0 \end{bmatrix} \tag{5.19}$$

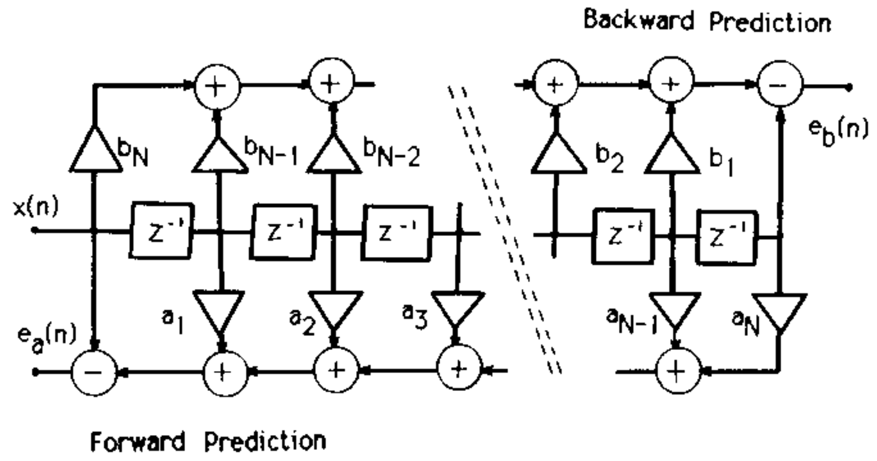


FIG. 5.3 Forward and backward linear prediction error filters.

Hence

$$A_N = J_N B_N, \quad E_{aN} = E_{bN} = E_N \tag{5.20}$$

For a stationary input signal, forward and backward prediction error powers are equal and the coefficients are the same, but in reverse order. Therefore, in theory, linear prediction analysis can be performed by the forward and backward approaches. However, it is in the transition phases that a difference appears, as seen in the next chapter. When the AC matrix is estimated, the best performance is achieved by combining both approaches, which gives the forward-backward linear prediction (FBLP) technique presented in Section 9.6.

Since the forward linear prediction error filter is minimum phase, the backward filter is maximum phase, due to (5.20).

An important property of backward linear prediction is that it provides a set of uncorrelated signals. The errors $e_{bi}(n)$ for successive orders $0 \leqslant i \leqslant N$ are not correlated. To show this useful result, let us express the vector of backward prediction errors in terms of the corresponding coefficients by repeatedly applying equation (5.17):

$$\begin{bmatrix} e_{b0}(n) \\ e_{b1}(n) \\ e_{b2}(n) \\ \vdots \\ e_{b(N-1)}(n) \end{bmatrix}^t = X^t(n) \begin{bmatrix} 1 & -B_1 & & & \\ 0 & 1 & -B_2 & & \\ 0 & 0 & 1 & & -B_{N-1} \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{5.21}$$

In more concise form, (5.21) is

$$[e_b(n)]^t = X^t(n)M_B$$

To check for the correlation, let us compute the backward error covariance matrix:

$$E\{[e_b(n)][e_b(n)]^t\} = M_B^t R_N M_B \tag{5.22}$$

By definition it is a symmetrical matrix. The product $R_N M_B$ is a lower triangular matrix, because of equation (5.18). The main diagonal consists of the successive prediction error powers $E_i (0 \leqslant i \leqslant N-1)$. But M_B^t is also a lower triangular matrix. Therefore, the product must have the same structure; since it must be symmetrical, it can only be a diagonal matrix. Hence

$$E\{[e_b(n)][e_b(n)]^t\} = \text{diag}(E_i) \tag{5.23}$$

and the backward prediction error sequences are uncorrelated. It can be verified that the same reasoning cannot be applied to forward errors.

The AC matrix R_{N+1} used in the above prediction equations contains R_N, as shown in decomposition (5.4), and order iterative relations can be derived for linear prediction coefficients.

5.4. ORDER ITERATIVE RELATIONS

To simplify the equations, let

$$r_N^a = \begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(N) \end{bmatrix}, \quad r_N^b = J_N r_N^a \tag{5.24}$$

Now, the following equation is considered, in view of deriving relations between order N and order $N-1$ linear prediction equations:

$$\left[\begin{array}{c:c} R_N & r_N^b \\ \hdashline (r_N^b)^t & r(0) \end{array}\right] \left[\begin{array}{c} 1 \\ -A_{N-1} \\ \hdashline 0 \end{array}\right] = \left[\begin{array}{c} E_{N-1} \\ 0 \\ \hdashline K_N \end{array}\right] \tag{5.25}$$

where

$$K_N = r(N) - \sum_{i=1}^{N-1} a_{i,N-1} r(N-i) \tag{5.26}$$

For backward linear prediction, using (5.20), we have

$$\left[\begin{array}{c:c} r(0) & (r_N^a)^t \\ \hdashline r_N^a & R_N \end{array}\right]\left[\begin{array}{c} 0 \\ -B_{N-1} \\ \hdashline 1 \end{array}\right]=\left[\begin{array}{c} K_N \\ 0 \\ \hdashline E_{N-1} \end{array}\right] \tag{5.27}$$

Multiplying both sides by the factor $k_N = K_N/E_{N-1}$ yields

$$R_{N+1}\left[\begin{bmatrix} 0 \\ -B_{N-1} \\ 1 \end{bmatrix} k_N\right] = \begin{bmatrix} k_N^2 E_{N-1} \\ 0 \\ K_N \end{bmatrix} \tag{5.28}$$

Subtracting (5.28) from (5.25) leads to the order N linear prediction equation, which for the coefficients implies the recursion

$$A_N = \begin{bmatrix} A_{N-1} \\ 0 \end{bmatrix} - k_N \begin{bmatrix} B_{N-1} \\ -1 \end{bmatrix} \tag{5.29}$$

and

$$E_N = E_{N-1}(1-k_N^2) \tag{5.30}$$

for the prediction error power. The last row of recursion (5.29) gives the important relation

$$a_{NN} = k_N \tag{5.31}$$

Finally the order N linear prediction matrix equation (5.3) can be solved recursively by the procedure consisting of equations (5.28), (5.31), (5.29), and (5.30) and called the Levinson–Durbin algorithm. It is given in Figure 5.4, and the corresponding FORTRAN subroutine to solve a linear system is given in Annex 5.1. Solving a system of N linear equations when the matrix to be inverted is Toeplitz requires N divisions and $N(N+1)$ multiplications, instead of the $\frac{N^3}{3}$ multiplications mentioned in Section 3.4 for the triangular factorization.

An alternative approach to compute the scalars k_i is to use the cross-correlation variables h_{jN} defined by

$$h_{jN} = E[x(n)e_{aN}(n-j)] \tag{5.32}$$

where $e_{aN}(n)$ is the output of the forward prediction error filter having N coefficients [3]. As mentioned in Section 2.5, the sequence h_{jN} is the impulse response of the generating filter when $x(n)$ is an order N AR signal. From the definition (5.16) for $e_{aN}(n)$, the variables h_{jN} are expressed by

$$h_{jN} = r(j) - \sum_{i=1}^{N} a_{iN} r(i+j)$$

```
Initialization
    $E_0 = r(0)$
For $1 \le j \le N$:
    $k_j = \frac{1}{E_{j-1}} \left[ r(j) - \sum_{i=1}^{j-1} a_{i\,j-1}\, r(j-i) \right]$
    $a_{jj} = k_j$
    $a_{ij} = a_{i\,j-1} - k_j a_{j-i,j-1} \quad ; \quad 1 \le i \le j-1$
    $E_j = E_{j-1}(1 - k_j^2)$
```

FIG. 5.4 The Levinson–Durbin algorithm for solving the linear prediction equation.

or, in vector notation,

$$h_{jN} = r(j) - (r_{jN})^t A_N \tag{5.33}$$

where

$$(r_{jN})^t = [r(j+1), r(j+2), \ldots, r(j+N)]$$

Clearly, the above definition leads to

$$h_{0N} = E_N \tag{5.34}$$

and

$$k_N = \frac{h_{(-N)(N-1)}}{E_{N-1}} = \frac{h_{(-N)(N-1)}}{h_{0(N-1)}} \tag{5.35}$$

A recursion can be derived from the prediction coefficient recursion (5.29) as follows:

$$h_{jN} = h_{j(N-1)} + k_N (r_{jN})^t \begin{bmatrix} B_{N-1} \\ -1 \end{bmatrix} \tag{5.36}$$

Developing the second term on the right gives

$$(r_{jN})^t \begin{bmatrix} B_{N-1} \\ -1 \end{bmatrix} = -h_{(-j-N)(N-1)} \tag{5.37}$$

Thus

$$h_{jN} = h_{j(N-1)} - k_N h_{(-j-N)(N-1)} \tag{5.38}$$

which yields, as a particular case if we take relation (5.35) into account,

$$h_{0N} = h_{0(N-1)}(1 - k_N^2) = E_N \tag{5.39}$$

Now a complete algorithm is available to compute the coefficients k_i. It is based entirely on the variables h_{ji} and consists of equations (5.35) and (5.38). The FORTRAN subroutine is given in Annex 5.2. The initial conditions are given by definition (5.33):

$$h_{j0} = r(j) \tag{5.40}$$

According to the h_{jN} definition (5.32) and the basic decorrelation property of linear prediction, the following equations hold:

$$h_{ji} = 0, \quad -i \leqslant j \leqslant -1 \tag{5.41}$$

If N coefficients k_i have to be computed, the indexes of the variables h_{ji} involved are in the range $(-N, N-1)$, as can be seen from equations (5.35) and (5.38). The multiplication count is about $N(N-1)$.

An additional property of the above algorithm is that the variables h_{ij} are bounded, which is useful for fixed-point implementation. Considering the definition (5.32), the cross-correlation inequality (3.10) of Chapter 3 yields

$$|h_{jN}| = |E[x(n)e(n-j)]| \leqslant \tfrac{1}{2}(r(0) + E_N)$$

Since $E_N \leqslant r(0)$ for all N,

$$|h_{jN}| \leqslant r(0) \tag{5.42}$$

The variables h_{jN} are bounded in magnitude by the signal power.

The number of operations needed in the two methods presented above to compute the k_i coefficients is close to N^2. However, it is possible to improve that count by a factor 2, using second-order recursions.

5.5. THE SPLIT LEVINSON ALGORITHM

The minimization of the quantity

$$E\{[x(n) - P_N^t X(n-1)]^2 + [x(n-1-N) - P_N^t X(n-1)]^2\}$$

with respect to the elements of the vector P_N yields

$$2R_N P_N = r_N^a + r_N^b$$

or

$$P_N = \tfrac{1}{2}(A_N + B_N) \tag{5.43}$$

which reflects the fact that the coefficients P_N are the symmetrical part of the prediction coefficients.

The associated matrix equation is

$$R_{N+2}\begin{bmatrix} 1 \\ -2P_N \\ 1 \end{bmatrix} = \begin{bmatrix} r(0) & (r_N^a)^t & r(N+1) \\ r_N^a & R_N & r_N^b \\ r(N+1) & (r_N^b)^t & r(0) \end{bmatrix}\begin{bmatrix} 1 \\ -2P_N \\ 1 \end{bmatrix} = \begin{bmatrix} E_{pN} \\ 0 \\ E_{pN} \end{bmatrix} \tag{5.44}$$

with

$$E_{pN} = E_N + K_{N+1} = E_N\left(1 + \frac{K_{N+1}}{E_N}\right) = E_N(1 + k_{N+1}) \tag{5.45}$$

This equation can be exploited to compute the reflection coefficients recursively, with the help of the matrix equations

$$R_{N+2}\begin{bmatrix} 1 \\ -2P_{N-1} \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} E_{p(N-1)} \\ 0 \\ E_{p(N-1)} \\ K' \end{bmatrix} \tag{5.46}$$

and

$$R_{N+2}\begin{bmatrix} 0 \\ 1 \\ -2P_{N-1} \\ 1 \end{bmatrix} = \begin{bmatrix} K' \\ E_{p(N-1)} \\ 0 \\ E_{p(N-1)} \end{bmatrix} \tag{5.47}$$

with

$$K' = r(N+1) + r(1) - 2[r(N), \ldots, r(2)]P_{N-1}$$

and finally

$$R_{N+2}\begin{bmatrix} 0 \\ 1 \\ -2P_{N-2} \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} K'' \\ E_{p(N-2)} \\ 0 \\ E_{p(N-2)} \\ K'' \end{bmatrix} \tag{5.48}$$

with

$$K'' = r(1) + r(N) - 2[r(2), \ldots, r(N-1)]P_{N-2}$$

By recursion, the order two recursion is obtained as

$$\begin{bmatrix} 1 \\ -2P_N \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -2P_{N-1} \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ -2P_{N-1} \\ 1 \end{bmatrix} - \frac{E_{p(N-1)}}{E_{p(N-2)}} \begin{bmatrix} 0 \\ 1 \\ -2P_{N-2} \\ 1 \\ 0 \end{bmatrix} \tag{5.49}$$

Thus, the coefficients P_N can be computed from P_{N-1} and P_{N-2}, with the help of the error power variables $E_{p(N-1)}$ and $E_{p(N-2)}$. The reflection coefficient k_N itself can also be computed recursively, combining recursion (5.30) for prediction error powers with equation (5.45), which leads to

$$\frac{E_{p(N-1)}}{E_{p(N-2)}} = (1 + k_N)(1 - k_{N-1}) \tag{5.50}$$

The initialization is

$$p_{11} = \frac{r(1)}{r(0)} = k_1, \qquad 2p_{12} = 2p_{22} = \frac{r(1) + r(2)}{r(0) + r(1)} \tag{5.51}$$

The error power is computed directly, according to its definition

$$E_{pN} = r(0) - 2 \sum_{i=1}^{N} r(i) p_{iN} + r(N+1) \tag{5.52}$$

The main advantage of this method is the gain in operation count by a factor close to two, with respect to the classical Levinson algorithm, because of the symmetry of the coefficients ($p_{iN} = p_{(N+1-i)N}$). The resulting algorithm consists of equations (5.49), (5.50), and (5.52) and it is called the split Levinson algorithm.

It is worth pointing out that the antisymmetric part of the prediction coefficients can be processed in a similar manner.

The order recursions can be associated with a particular structure, the lattice prediction filter.

5.6. THE LATTICE LINEAR PREDICTION FILTER

The coefficients k_i establish direct relations between forward and backward prediction errors for consecutive orders. From the definition of the order N forward prediction error $e_{aN}(n)$, we have

$$e_{aN}(n) = x(n) - A_N^t X(n-1) \tag{5.53}$$

and the coefficient recursion (5.29), we derive

$$e_{aN}(n) = e_{a(N-1)}(n) - k_N [-B_{N-1}^t, 1] X(n-1) \tag{5.54}$$

The order N backward prediction error $e_{bN}(n)$ is

$$e_{bN}(n) = x(n-N) - B_N^t X(n) \tag{5.55}$$

For order $N-1$,

$$e_{b(N-1)}(n) = x(n+1-N) - \sum_{i=1}^{N-1} b_{i(N-1)} x(n+1-i) = [-B_{N-1}^t, 1]X(n) \tag{5.56}$$

Therefore, the prediction errors can be rewritten as

$$e_{aN}(n) = e_{a(N-1)}(n) - k_N e_{b(N-1)}(n-1) \tag{5.57}$$

and

$$e_{bN}(n) = e_{b(N-1)}(n-1) - k_N e_{a(N-1)}(n) \tag{5.58}$$

The corresponding structure is shown in Figure 5.5; it is called a lattice filter section, and a complete FIR filter of order N is realized by cascading N such sections. Indeed, to start, $e_{b0}(n) = x(n)$. Now the lattice coefficients k_i can be further characterized. Consider the cross-correlation

$$E[e_{aN}(n)e_{bN}(n-1)] = r(N+1) - B_N^t r_N^a - A_N^t J_N r_N^a + A_N^t R_N B_N \tag{5.59}$$

Because of the backward prediction equation

$$R_N B_N = r_N^b = J_N r_N^a \tag{5.60}$$

the sum of the last two terms in the above cross-correlation is zero. Hence

$$E[e_{aN}(n)e_{bN}(n-1)] = r(N+1) - B_N^t r_N^a = K_{N+1}$$

and

$$k_N = \frac{E[e_{a(N-1)}(n)e_{b(N-1)}(n-1)]}{E_{N-1}} \tag{5.61}$$

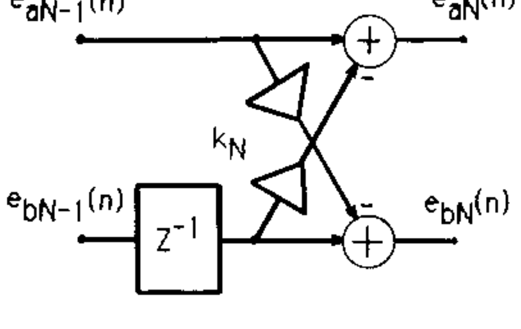


FIG. 5.5 Lattice linear prediction filter section.

The lattice coefficients represent a normalized cross-correlation of forward and backward prediction errors. They are often called the PARCOR coefficients [4]. Due to wave propagation analogy, they are also called the reflection coefficients.

The lattice coefficient k_N is related to the N zeros z_i of the order N FIR prediction error filter, whose transfer function is

$$H_N(z) = 1 - \sum_{i=1}^{N} a_{iN} z^{-1} = \prod_{i=1}^{N} (1 - z_i z^{-1}) \tag{5.62}$$

Since $k_N = a_{NN}$, we have

$$k_N = (-1)^{N+1} \prod_{i=1}^{N} z_i \tag{5.63}$$

From the filter linear phase property, we know that $|z_i| \leqslant 1$, which yields

$$|k_N| \leqslant 1 \tag{5.64}$$

Conversely, using (5.29), it can be shown iteratively that, if the lattice coefficient absolute values are bounded by unity, then the prediction error filter has all its roots inside the unit circle and, thus, it is minimum phase. Therefore, it is very easy to check for the minimum phase property of a lattice FIR filter. Just check that the magnitude of the lattice coefficients does not exceed unity.

The correspondence between PARCOR and the transversal filter coefficients is provided by recursion (5.29). In order to get the set of $a_{iN} (1 \leqslant i \leqslant N)$ from the set of $k_i (1 \leqslant i \leqslant N)$, we need to iterate the recursion N times with increasing indexes. To get the k_i from the a_{iN}, we must proceed in the reverse order and calculate the intermediate coefficients $a_{ji} (N-1 \geqslant i \geqslant 1; j \leqslant i)$ by the following expression:

$$a_{j(i-1)} = \frac{1}{1 - k_i^2} [a_{ji} + k_i a_{(i-j)i}], \quad k_i = a_{ii} \tag{5.65}$$

The procedure is stopped if $k_i = 1$, which means that the signal consists of i sinusoids without additive noise.

Two additional relations are worth pointing out:

$$1 - \sum_{i=1}^{N} a_i = \prod_{i=1}^{N} (1 - k_i) \tag{5.66}$$

$$r(0) = \sigma_x^2 = \sum_{i=1}^{N} k_i E_i \tag{5.67}$$

A set of interesting properties of the transversal filter coefficients can be deduced from the magnitude limitation of the PARCOR coefficients [5]. For example,

$$|a_{iN}| \leqslant \frac{N!}{(N-i)!i!} \leqslant 2^{N-1} \tag{5.68}$$

which can be useful for coefficient scaling in fixed-point implementation and leads to

$$\sum_{i=1}^{N} |a_{iN}| \leqslant 2^N - 1 \tag{5.69}$$

and

$$\|A_N\|^2 = A_N^t A_N \leqslant \frac{(2n)!}{(n!)^2} - 1 \tag{5.70}$$

This bound is reached for the two theoretical extreme cases where $k_i = -1$ and $k_i = (-1)^{i-1} (1 \leqslant i \leqslant N)$.

The results we have obtained in linear prediction now allow us to complete our discussion on AC matrices and, particularly, their inverses.

5.7. THE INVERSE AC MATRIX

When computing the inverse of a matrix, first compute the determinant. The linear prediction matrix equation is

$$\begin{bmatrix} 1 \\ -A_N \end{bmatrix} = \begin{bmatrix} r(0) & (r_N^a)^t \\ r_N^a & R_N \end{bmatrix}^{-1} \begin{bmatrix} E_N \\ 0 \end{bmatrix} \tag{5.71}$$

The first row yields

$$1 = \frac{\det R_N}{\det R_{N+1}} E_N \tag{5.72}$$

which, using the Levinson recursions, leads to

$$\det R_N = [r(0)]^N \prod_{i=1}^{N-1} (1 - k_i^2)^{N-i} \tag{5.73}$$

To exploit further equation (5.59), let us denote by V_i the column vectors of the inverse matrix R_{N+1}^{-1}.

Considering the forward and backward linear prediction equations, we can write the vectors V_1 and V_{N+1} as

$$V_1 = \frac{1}{E_N}\begin{bmatrix} 1 \\ -A_N \end{bmatrix}, \quad V_{N+1} = \frac{1}{E_N}\begin{bmatrix} -B_N \\ 1 \end{bmatrix} \tag{5.74}$$

Thus, the prediction coefficients show up directly in the inverse AC matrix, which can be completely expressed in terms of these coefficients.

Let us consider the $2(N+1) \times (N+1)$ rectangular matrix M_A defined by

$$M_A^t = \underbrace{\begin{bmatrix} 1 & -a_{1N} & -a_{2N} & \cdots & -a_{NN} \\ 0 & 1 & -a_{1N} & \cdots & -a_{(N-1)N} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}}_{N+1}\underbrace{\begin{matrix} 0 & \cdots & 0 & 0 \\ -a_{NN} & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots \\ -a_{1N} & \cdots & -a_{NN} & 0 \end{matrix}}_{N+1} \tag{5.75}$$

The prediction equation (5.3) and relations (2.64) and (2.72) for AR signals yield the equality

$$M_A^t R_{2(N+1)} M_A = E_N I_{N+1} \tag{5.76}$$

where $R_{2(N+1)}$ is the AC matrix of the order N AR signal. Pre- and post-multiplying by M_A and M_A^t; respectively, gives

$$(M_A M_A^t) R_{2(N+1)} (M_A M_A^t) = (M_A M_A^t) E_N \tag{5.77}$$

The expression of the matrix R_{N+1}^{-1} is obtained by partitioning M_A into two square $(N+1) \times (N+1)$ matrices M_{A1} and M_{A2},

$$M_A^t = [M_{A1}^t, M_{A2}^t] \tag{5.78}$$

and taking into account the special properties of the triangular matrices involved

$$R_{N+1}^{-1} = \frac{1}{E_N}(M_{A1}^t M_{A1} - M_{A2} M_{A2}^t) \tag{5.79}$$

This expression shows that the inverse AC matrix is doubly symmetric. If the signal is AR with order less than N, then R_{N+1}^{-1} is Toeplitz in the center, but edge effects appear in the upper left and lower right corners. A simple example is given in Section 3.4.

Decomposition (5.67) can be extended to matrices which are not doubly symmetric. In that case, the matrices M_{B1} and M_{B2} of the backward prediction coefficients are involved, and the equation becomes

$$R_{N+1}^{-1} = \frac{1}{E_{aN}}(M_{A1}^t M_{B1} - M_{B2} M_{A2}^t) \tag{5.79a}$$

An alternative decomposition of R_{N+1}^{-1} can be derived from the cross-correlation properties of data and error sequences.

Since the error signal $e_N(n)$ is not correlated with the input data $x(n-1), \dots, x(n-N)$, the sequences $e_{N-i}(n-i)$, $0 \leqslant i \leqslant N$, are not correlated. In vector form they are written

$$\begin{bmatrix} e_N(n) \\ e_{N-1}(n-1) \\ \vdots \\ e_0(n-N) \end{bmatrix} = \begin{bmatrix} 1 & \cdots & -A_N & \cdots & \cdots \\ 0 & 1 & \cdots & -A_{N-1} & \cdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & 1 \end{bmatrix} \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-N) \end{bmatrix} \tag{5.80}$$

The covariance matrix is the diagonal matrix of the prediction errors. After algebraic manipulations we have

$$R_{N+1}^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -a_{1N} & 1 & \cdots & 0 \\ -a_{2N} & -a_{2(N-1)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -a_{NN} & -a_{(N-1)(N-1)} & \cdots & 1 \end{bmatrix} \begin{bmatrix} E_N^{-1} & 0 & \cdots & 0 \\ 0 & E_{N-1}^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & E_0^{-1} \end{bmatrix}$$

$$\times \begin{bmatrix} 1 & -a_{1N} & -a_{2N} & \cdots & -a_{NN} \\ 0 & 1 & -a_{1(N-1)} & \cdots & -a_{1(N-1)(N-1)} \\ \vdots & \vdots & \ddots & \cdots & \vdots \\ 0 & 0 & \cdots & \cdots & 1 \end{bmatrix} \tag{5.81}$$

This is the triangular Cholesky decomposition of the inverse AC matrix. It can also be obtained by considering the backward prediction errors, which are also uncorrelated, as shown in Section 5.3.

The important point in this section is that the inverse AC matrix is completely represented by the forward prediction error power and the prediction coefficients. Therefore, LS algorithms which implement R_N^{-1} need not manipulate that matrix, but need only calculate the forward prediction error power and the forward and backward prediction coefficients. This is the essence of FLS algorithms.

5.8. THE NOTCH FILTER AND ITS APPROXIMATION

The ideal predictor is the filter which cancels the predictable components in the signal without amplifying the unpredictable ones. That favorable situation occurs with sinusoids in white noise, and the ideal filter is the notch filter with frequency response

$$H_{NI}(\omega) = 1 - \sum_{i=1}^{M} \delta(\omega - \omega_i) \tag{5.82}$$

where $\delta(x)$ is the Dirac distribution and the ω_i, $1 \leqslant i \leqslant M$, are the frequencies of the sinusoids. Clearly, such a filter completely cancels the sinusoids and does not amplify the input white noise.

An arbitrarily close realization $H_N(\omega)$ of the ideal filter is achieved by

$$H_N(z) = \frac{\prod_{i=1}^{M}(1 - e^{j\omega_i} z^{-1})}{\prod_{i=1}^{M}(1 - (1-\varepsilon)e^{j\omega_i} z^{-1})} \tag{5.83}$$

where the positive scalar ε is made arbitrarily small [6]. The frequency response of a second-order notch filter is shown in Figure 5.6, with the location of poles and zeros in the z-plane.

The notch filter cannot be realized by an FIR predictor. However, it can be approximated by developing in series the factors in the denominator of $H_N(z)$, which yields

$$\frac{1}{1 - P_i z^{-1}} = 1 + \sum_{n=1}^{\infty} (P_i z^{-1})^n \tag{5.84}$$

This approach is used to figure out the location in the z-plane of the zeros and poles of linear prediction filters.

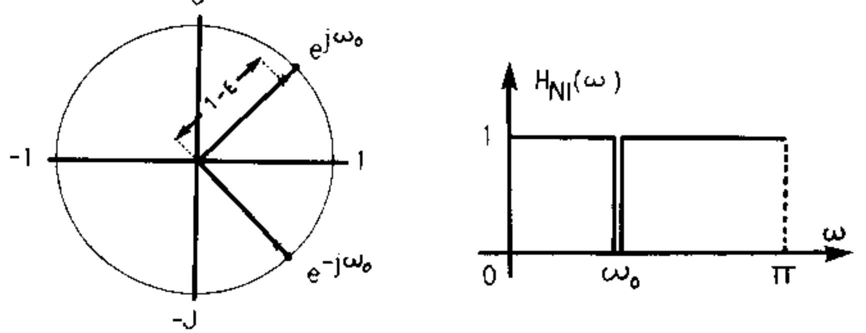


FIG. 5.6 The notch filter response, zeros and poles.

5.9. ZEROS OF FIR PREDICTION ERROR FILTERS

The first-order notch filter $H_{N1}(z)$ is adequate to handle zero frequency signals:

$$H_{N1}(z) = \frac{1 - z^{-1}}{1 - (1 - \varepsilon)z^{-1}} \tag{5.85}$$

A simple Tchebycheff FIR approximation is

$$H(z) = \frac{1 - z^{-1}}{1 - (1 - \varepsilon)z^{-1}} [1 - (bz^{-1})^N]$$

where b is a positive real constant. Now, a realizable filter is obtained for $b = 1 - \varepsilon$, because

$$H(z) = (1 - z^{-1})[1 + bz^{-1} + \cdots + b^{N-1}z^{-(N-1)}] \tag{5.86}$$

Now constant b can be calculated to minimize the prediction error power. For a zero frequency signal $s(n)$ of unit power, a white input noise with power σ_b^s, the output power of the filter with transfer function $H(z)$ given by (5.74) is

$$E[e^2(n)] = 2\sigma_b^2 \frac{1 + b^{2N-1}}{1 + b} \tag{5.87}$$

The minimum is reached by setting to zero the derivative with respect to b; thus

$$b = \left[\frac{1}{2N - 1 + (2N - 2)b}\right]^{1/2(N-1)} \tag{5.88}$$

For b reasonably close to unity the following approximation is valid:

$$b \approx \left[\frac{1}{4N - 3}\right]^{1/2(N-1)} \tag{5.89}$$

According to (5.86) the zeros of the filter which approximates the prediction error filter are located at $+1$ and $be^{j2\pi i/N}$, $1 \leqslant i \leqslant N - 1$, in the complex plane. And the circle radius b does not depend on the noise power. For large N, b comes close to unity, and estimate (5.89) is all the better. Figure 5.7(a) shows true and estimated zeros for a 12-order prediction error filter.

A refinement in the above procedure is to replace $1 - z^{-1}$ by $1 - az^{-1}$ in $H(z)$ and optimize the scalar a because, in the prediction of noisy signals, the filter zeros are close to but not on the unit circle, as pointed out earlier, particularly in Section 5.2.

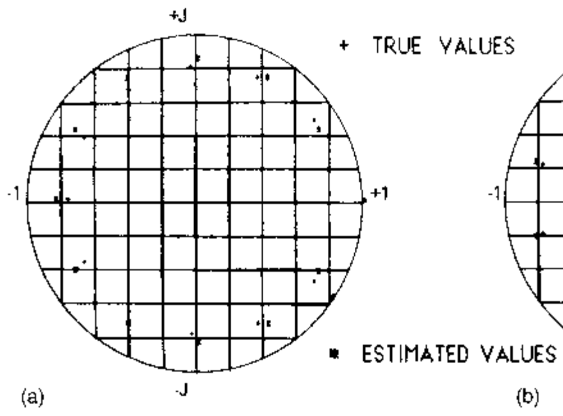


FIG. 5.7 Zeros of a 12-order predictor applied to (a) a zero frequency signal and (b) a $\frac{\pi}{12}$ frequency sinusoid.

The above approach can be extended to estimate the prediction error filter zeros when the input signal consists of M real sinusoids of equal amplitude and uniformly distributed on the frequency axis. The approximating transfer function is

$$H(z) = \frac{1 - z^{-2M}}{1 - b^{2M} z^{-2M}} (1 - b^N z^{-N}) \tag{5.90}$$

If $N = k\,2M$, for integer k, the output error power is

$$E[e^2(n)] = 2\sigma_b^2 \frac{1 + b^{2N-2M}}{1 + b^{2M}} \tag{5.91}$$

the minimization procedure leads to

$$b \approx \left[\frac{M}{2N - 3M}\right]^{1/2(N-2M)} \tag{5.92}$$

Equation (5.89) corresponds to the above expression when $M = \frac{1}{2}$. Note that the zero circle radius b depends on the number $N - 2M$, which can be viewed as the number of free or uncommitted zeros in the filter; the mission of these zeros is to bring down the amplification of the input noise power. If the noise is not flat, they are no longer on a circle within the unit circle.

The validity of the above derivation might look rather restricted, since the sinusoidal frequencies have to be uniformly distributed and the filter order N must be a multiple of the number of sinusoids M. Expression (5.92) remains a reasonably good approximation of the zero modulus as soon as

$N > 2M$. For example, the true and estimated zeros of an order 12 linear prediction error filter, applied to the sinusoid with frequency $\frac{\pi}{12}$, are shown in Figure 5.7(b).

When the sinusoidal frequencies are arbitrarily distributed on the frequency, the output noise power is increased with respect to the uniform case and the zeros in excess of $2M$ come closer to the unit circle center. Therefore expression (5.92) may be regarded as an estimation of the upper bound of the distance of the zeros in excess of $2M$ to the center of the unit circle. That result is useful for the retrieval of sinusoids in noise [7].

The foregoing results provide useful additional information about the magnitude of the PARCOR coefficients.

When the PARCOR coefficients k_i are calculated iteratively, their magnitudes grow, monotonically or not, up to a maximum value which, because of equation (5.53), corresponds to the prediction filter order best fitted to the signal model. Beyond, the k_i decrease in magnitude, due to the presence of the zeros in excess.

If the signal consists of M real sinusoids, then

$$|k_N| \approx b^{N-2M}, \quad N \geqslant 2M \tag{5.93}$$

Substituting (5.80) into (5.81) gives

$$k_n \approx \left(\frac{M}{2N-3M}\right)^{1/2} \quad N \geqslant 2M \tag{5.94}$$

Equation (5.94) is a decreasing law which can be extended to any signal and considered as an upper bound estimate for the lattice coefficient magnitudes for predictor orders exceeding the signal model order. In Figure 5.8 true lattice coefficients are compared with estimates for sinusoids at freqeuncies $\frac{\pi}{2}$ and $\frac{\pi}{12}$.

The magnitude of the maximum PARCOR coefficient is related to the input SNR. The relation is simple for M sinusoids uniformly distributed on the frequency axis, because the order $2M$ prediction error filter is

$$H(z) = 1 - b^{2M} z^{-2M} \tag{5.95}$$

The optimum value of b is derived from the prediction error power as before, so

$$b^{2M} = |k_{2M}| = \frac{\mathrm{SNR}}{1+\mathrm{SNR}} \tag{5.96}$$

The approach taken to locate the predictor zeros can also be applied to the poles of an IIR filter.

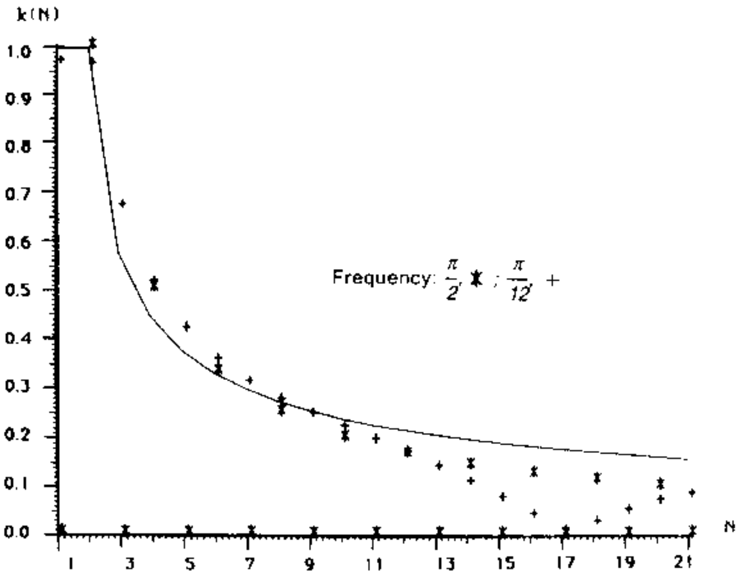


FIG. 5.8 Lattice coefficients vs. predictor order for sinusoids.

5.10. POLES OF IIR PREDICTION ERROR FILTERS

The transfer function of a purely recursive IIR filter of order N is

$$H(z) = \frac{1}{1 - \sum_{i=1}^{N} b_i z^{-i}} \tag{5.97}$$

Considering a zero frequency signal in noise, to begin with, we can obtain a Tchebycheff approximation of the prediction error filter $1 - az^{-1}$ by the expression

$$H(z) = \frac{1 - az^{-1}}{1 - a^{N+1} z^{-(N+1)}} = \frac{1}{1 + az^{-1} + \cdots + a^N z^{-N}} \tag{5.98}$$

where $0 \ll a < 1$. Now the prediction error power is

$$E[e^2(n)] = |H(1)|^2 + \sigma_b^2 \left(\sum_{i=0}^{\infty} h_i^2 \right)$$

where the h_i is the filter impulse response. A simple approximation is

$$E[e^2(n)] \approx |H(1)|^2 + \sigma_b^2 \frac{1+a^2}{1-a^{2(N+1)}} \tag{5.99}$$

The parameter a is obtained by setting to zero the derivative of the prediction error power. However, a simple expression is not easily obtained.

Two different situations must be considered separately, according to the noise power σ_b^2. For small noise power

$$\frac{\partial}{\partial a}\left(\frac{1-a}{1-a^{N+1}}\right)^2 \approx -\frac{1}{N+1}, \frac{\partial}{\partial a}\left[\frac{1+a^2}{1-a^{2(N+1)}}\right] \approx \frac{1}{N+1}\frac{1}{(1-a)^2}$$

and

$$a \approx 1 - \sigma_b \tag{5.100}$$

On the other hand, for large noise power, simple approximations are

$$\frac{\partial}{\partial a}\left(\frac{1-a}{1-a^{N+1}}\right)^2 \approx -2(1-a), \quad \frac{\partial}{\partial a}\left[\frac{1+a^2}{1-a^{2(N+1)}}\right] \approx 2a$$

which yield

$$a \approx \frac{1}{1+\sigma_b^2} \tag{5.101}$$

In any case, for a zero frequency signal the poles of the IIR filter are uniformly distributed in the complex plane on a circle whose radius depends on the SNR. We can rewrite $H(z)$ as

$$H(z) = \frac{1}{\prod\limits_{n=1}^{N}(1 - ae^{jn\omega_0}z^{-1})}, \quad \omega_0 = \frac{2\pi}{N+1} \tag{5.102}$$

There is no pole at the signal frequency and, in some sense, the IIR predictor operates by default.

The prediction gain is limited. Since $|a| < 1$ for stability reasons, we derive a simple bound $E_{\min}$ for the prediction power from (5.99) and (5.98), neglecting the input noise:

$$E_{\min} = \frac{1}{(N+1)^2} \tag{5.103}$$

The above derivations can be extended to signals made of sinusoids in noise. The results show, as above, that the purely recursive IIR predictors are not as efficient as their FIR counterparts.

5.11. GRADIENT ADAPTIVE PREDICTORS

The gradient techniques described in the previous chapter can be applied to prediction filters. A second-order FIR filter is taken as an example in Section 4.4. The reference signal is the input signal itself, which simplifies some expressions, such as coefficient and internal data word-length estimations (4.61) and (4.65) in Chapter 4, which in linear prediction become

$$b_c \approx \log_2(\tau_e) + \log_2(G_p) + \log_2(a_{\max}) \tag{5.104}$$

and

$$b_i \approx 2 + \tfrac{1}{2}\log_2(\tau_e) + \log_2(G_p) \tag{5.105}$$

where G_p^2 is the prediction gain, defined, according to equation (4.9) in Chapter 4, as the input signal-to-prediction-error power ratio. The maximum magnitude of the coefficients, $a_{\max}$, is bounded by 2^{N-1} according to inequality (5.68).

The purely recursive IIR prediction error filter in Figure 5.9 is a good illustration of adaptive IIR filters. Its equations are

$$\begin{aligned} e(n+1) &= x(n+1) - B^t(n)E(n) \\ B(n+1) &= B(n) + \delta e(n+1)E(n) \end{aligned} \tag{5.106}$$

with

$$B^t(n) = [b_1(n), \ldots, b_N(n)], \quad E^t(n) = [e(n), \ldots, e(n+1-N)]$$

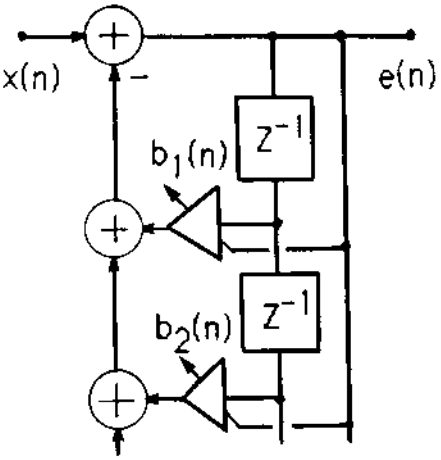


FIG. 5.9 Purely recursive IIR prediction filter.

The coefficient updating equation can be rewritten as

$$B(n+1) = [I_N - \delta E(n)E^t(n)]B(n) + \delta x(n+1)E(n) \tag{5.107}$$

The steady-state position is reached when the error $e(n+1)$ is no longer correlated with the elements of the error vctor; the filter tends to decorrelate the error sequence. The steady-state coefficient vector B_∞ is

$$B_\infty = (E[E(n)E^t(n)])^{-1}E[x(n+1)E(n)] \tag{5.108}$$

and the error covariance matrix should be close to a diagonal matrix:

$$E[E(n)E^t(n)] \approx \sigma_e^2 I_N \tag{5.109}$$

The output power is

$$E[e^2(n+1)] = E[x^2(n+1)] - B_\infty^t E[E(n)E^t(n)]B_\infty \tag{5.110}$$

which yields the prediction gain

$$G_p^2 = \frac{\sigma_x^2}{\sigma_e^2} \approx 1 + B_\infty^t B_\infty \tag{5.111}$$

Therefore the coefficients should take as large values as possible.

Note that, in practice, a local instability phenomenon can occur with recursive gradient predictors [8]. As indicated in the previous section, the additive input noise keeps the poles inside the unit circle. If that noise is small enough, in a gradient scheme with given step δ, the poles jump over the unit circle. The filter becomes unstable, which can be interpreted as the addition to the filter input of a spurious sinusoidal component, exponentially growing in magnitude and at the frequency of the pole. The adaptation process takes that component into account, reacts exponentially as well, and the pole is pushed back in the unit circle, which eliminates the above spurious component. Hence the local instability, which can be prevented by the introduction of a leakage factor as in Section 4.6, which yields the coefficient updating equation

$$B(n+1) = (1-\gamma)B(n) + \delta e(n+1)E(n) \tag{5.112}$$

The bound on the adaptation step size δ can be determined, as in Section 4.2, by considering the a posteriori error

$$\varepsilon(n+1) = e(n+1)[1 - \delta E^t(n)E(n)] \tag{5.113}$$

which leads to the bound

$$0 < \delta < \frac{2}{N\sigma_e^2} \tag{5.114}$$

Since the output error power is at most equal to the input signal power, the bound is the same as for the FIR structure. The initial time constant is also about the same, if the step size is small enough, due to the following approximation, which is valid for small coefficient magnitudes:

$$\frac{1}{1+\sum_{i=1}^{N} b_i z^{-1}} \approx 1 - \sum_{i=1}^{N} b_i z^{-1} \tag{5.115}$$

As an illustration, the trajectories of the six poles of a purely recursive IIR prediction error filter applied to a sinusoid with frequency $\frac{2\pi}{3}$ are shown in Figure 5.10. After the initial phase, there are no poles at frequencies $\pm\frac{2\pi}{3}$.

The lattice structure presented in Section 5.6 can also be implemented in a gradient adaptive prediction error filter, as shown in Figure 5.11 for the FIR case. Several criteria can be used to update the coefficient k_i. A simple one is the minimization of the sum of forward and backward prediction error powers at each stage. The derivation of equations (5.57) and (5.58) with respect to the coefficients leads to the updating relations ($1 \leqslant i \leqslant N$)

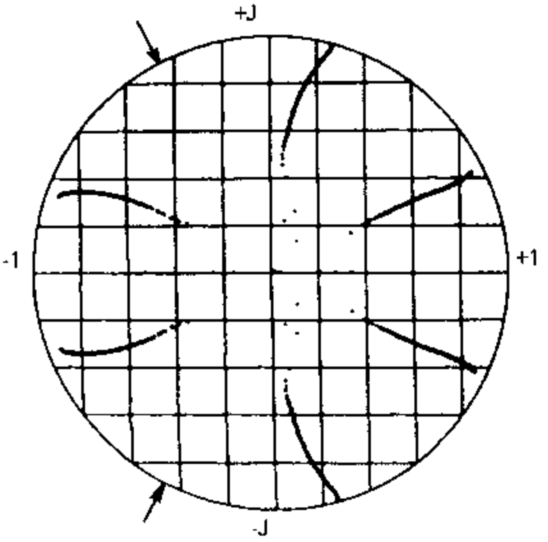


FIG. 5.10 Pole trajectories of a gradient adaptive IIR predictor applied to a sinusoid at frequency $2\frac{\pi}{3}$.

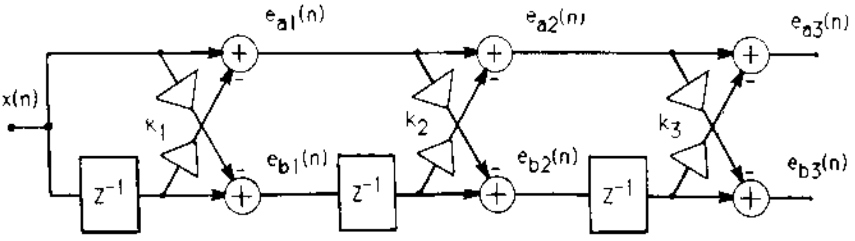


FIG. 5.11 FIR lattice prediction error filter.

$$k_i(n+1) = k_i(n) + \frac{\delta}{2}[e_{ai}(n+1)e_{b(i-1)}(n) + e_{bi}(n+1)e_{a(i-1)}(n+1)] \tag{5.116}$$

which, from (5.57) and (5.58), can be rewritten as

$$k_i(n+1) = k_i(n) + \delta\left[e_{a(i-1)}(n+1)e_{b(i-1)}(n) - k_i(n)\frac{e^2_{b(i-1)}(n) + e^2_{a(i-1)}(n+1)}{2}\right] \tag{5.117}$$

Clearly, the steady-state solution $k_{i\infty}$ agrees with the PARCOR coefficient definition (5.61).

The performance of the lattice gradient algorithm can be assessed through the methods developed in Chapter 4, and comparisons can be made with the transversal FIR structure, including computation accuracies [9, 10]. However, the lattice filter is made of sections which have to be analyzed in turn.

The coefficient updating for the first lattice section, according to Figure 5.11, is

$$k_1(n+1) = k_1(n) + \delta\left[x(n+1)x(n) - k_1(n)\frac{x^2(n+1) + x^2(n)}{2}\right] \tag{5.118}$$

For comparison, the updating equation of the coefficient of the first-order FIR filter can be written as

$$a(n+1) = a(n) + \delta[x(n+1)x(n) - a(n)x^2(n)] \tag{5.119}$$

The only difference resides in the better power estimation performed by the last term on the right side of (5.118), and it can be assumed that the first lattice section performs like a first-order FIR prediction error filter, which leads to the residual error

$$E_{1R} = (1 - k_1^2)\sigma_x^2 \left(1 + \frac{\delta}{2}\sigma_x^2\right) \tag{5.120}$$

To assess the complete lattice prediction error filter, we now consider the subsequent sections. However, the adaptation step sizes are adjusted in these sections to reflect the decrease in signal powers. To make the time constant homogeneous, the adaptation step sizes in different sections are made inversely proportional to the input signal powers.

In such conditions, the first section is crucial for global performance and accuracy requirements. For example, the first section is the major contributor to the filter excess output noise power, and E_{1R} can be taken as the total lattice filter residual error.

Thus, transversal and lattice filters have the same excess output noise power if the following equality holds:

$$\sigma_x^2 \prod_{i=1}^{N} (1 - k_i^2) \frac{\delta}{2} N \sigma_x^2 = (1 - k_1^2)\sigma_x^2 \frac{\delta}{2} \sigma_x^2$$

Therefore, the lattice gradient filter is attractive, under the above hypotheses, if

$$\prod_{i=2}^{N} \frac{1}{1 - k_i^2} < N \tag{5.121}$$

that is, when the system gain is small and when the first section is very efficient, which can be true in linear prediction of speech, for example. Combinations of lattice and transversal adaptive filters can be envisaged, and the above results suggest cascading a lattice section and a transversal filter [11].

As for computational accuracy, the coefficient magnitudes of lattice filters are bounded by unity. Therefore, the coefficient word length for the lattice prediction error filter can be estimated by

$$b_{cl} \approx \log_2(\tau_e) + \log_2(G_p) \tag{5.122}$$

which can be appreciably smaller than estimate (5.104) for the transversal counterpart.

Naturally, simplified adaptive approaches, like LAV and sign algorithms, can also be used in linear prediction with any structure.

5.12. ADAPTIVE LINEAR PREDICTION OF SINUSOIDS

The AC matrix of order N of a real sinusoid with unit power is given by the following expression, as mentioned earlier, for example in Section 3.7.

$$R_N = \begin{bmatrix} 1 & \cos\omega & \dots & \cos(N-1)\omega \\ \cos\omega & 1 & \dots & \cos(N-2)\omega \\ \vdots & \vdots & \ddots & \vdots \\ \cos(N-1)\omega & \cos(N-2)\omega & \dots & 1 \end{bmatrix} \tag{5.123}$$

For $\omega = \pi k/N$ (h integer), the vector

$$U(\omega) = \frac{1}{\sqrt{N}}[1, e^{-j\omega}, \dots, e^{-j(N-1)\omega}]^t$$

is a unitary eigenvector, as is $U(-\omega)$, and the corresponding eigenvalues are

$$\lambda_i = \lambda_2 = \frac{N}{2} \tag{5.124}$$

If a white noise with power σ_b^2 is added, the eigenvalues become

$$\lambda_i = \lambda_2 = \frac{N}{2} + \sigma_b^2; \qquad \lambda_i = \sigma_b^2 \quad (3 \leqslant i \leqslant N) \tag{5.125}$$

and the eigenvectors remain unchanged.

As shown in Section 3.7, the matrix R_N is diagonalized as

$$R_N = M^{-t}\Lambda M \tag{5.126}$$

where the columns of M^{-1} are the two eigenvectors $U(\omega)$ and $U(-\omega)$, completed by a set of orthogonal eigenvectors

Now, according to the linear prediction matrix equation (5.3), the vector of the transversal prediction coefficients of order N is

$$A_N = R_N^{-1}[\cos\omega, \cos 2\omega, \dots, \cos N\omega]^t \tag{5.127}$$

As shown in Section 3.6, the correlation vector can be expressed in terms of the eigenvectors

$$\begin{bmatrix} \cos\omega \\ \cos 2\omega \\ \vdots \\ \cos N\omega \end{bmatrix} = \frac{\sqrt{N}}{2}[e^{-j\omega}U(\omega) + e^{j\omega}U(-\omega)] \tag{5.128}$$

Substituting (5.128) into (5.127) and using (5.126) with the orthogonality property

$$A_N = \frac{1}{N/2+\sigma_b^2}[\cos\omega, \cos 2\omega, \dots, \cos N\omega] \tag{5.129}$$

If ω is not an integer multiple of π/N, the above results are not strictly applicable. However, for $\pi/N < \omega < \pi - \pi/N$, the eigenvalues remain close to each other as indicated by equation (3.105), and the above expression can be retained as a reasonably close approximation of the prediction coefficients. In fact, the results given in this section are an alternative and a complement to those of Section 5.9.

If, instead of a single sinusoid, a set of M sinusoids is considered, and if they all have unit power and are separated in frequency by more than π/N, then the eigenvalues are approximately given by:

$$\begin{aligned} \lambda_i &\approx N/2+\sigma_b^2, \quad 1 \leqslant i \leqslant 2M \\ \lambda_i &= \sigma_b^2, \quad 2M+1 \leqslant i \leqslant N \end{aligned} \tag{5.130}$$

and the linear prediction coefficient vector can be approximated by

$$A_N \approx \frac{1}{N/2+\sigma_b^2}\sum_{i=1}^{M}\begin{bmatrix} \cos\omega_i \\ \cos 2\omega_i \\ \vdots \\ \cos N\omega_i \end{bmatrix} \tag{5.131}$$

An adaptive FIR predictor provides this vector, on average and in its steady state. As concerns the learning curve, as indicated in Section (4.4), the time constant associated with the eigenvalue λ_i is

$$\tau_i = 1/\delta\lambda_i \tag{5.132}$$

For a single sinusoid in white noise, the two modes which form the coefficients have the same time constant:

$$\tau_1 = \tau_2 = \frac{1}{\delta(N/2+\sigma_b^2)} \tag{5.133}$$

which is also the time constant of the coefficients themselves and, hence, of the prediction error.

It is worth pointing out that, according to the above results, the time constant for a sinusoid without noise is $N/2$ times smaller than that of a white noise with the same power. However, when the frequency of the sinusoid approaches the limits of the frequency domain, i.e., 0 or π, one of the two eigenvalues approaches zero and the corresponding time constant grows to infinity. The same applies to the case of a signal consisting of M sinusoids. More generally, the above properties stem from the fact that the coefficients of the adaptive filter move in the signal subspace, as is clearly shown by updating equation (4.3) for the gradient algorithm.

For the sake of completeness, similar results will now be derived for complex sinusoids, for which a different approach will be used.

Let us consider the case of a single cisoid in noise:

$$x(n) = e^{jn\omega} + b(n) \tag{5.134}$$

with $b(n)$ a white noise with power σ_b^2. The AC matrix is given by

$$R_N = \sigma_b^2 I_N + \bar{V}_1 V_1^t \tag{5.135}$$

where

$$V_1^t = [1, e^{j\omega}, e^{j2\omega}, \ldots, e^{j(N-1)\omega}]$$

The inverse matrix can be calculated with the help of the matrix inversion lemma, presented in detail in Section 6.2 below,

$$R_N^{-1} = \frac{I_N}{\sigma_b^2} - \frac{I_N}{\sigma_b^2}\bar{V}_1\left[\frac{1}{\sigma_b^2}V_1^t\bar{V}_1 + 1\right]^{-1} V_1^t \frac{1}{\sigma_b^2}$$

and, in concise form

$$R_N^{-1} = \frac{1}{\sigma_b^2}\left[I_N - \frac{\bar{V}_1 V_1^t}{\sigma_b^2 + N}\right] \tag{5.136}$$

The linear prediction coefficients are obtained through the minimization of the cost function

$$J = E[|x(n+1) - (\bar{A})^t X(n)|^2] \tag{5.137}$$

which, as shown in Section 1.4, yields

$$A = R_N^{-1} E[\bar{x}(n+1)X(n)] \tag{5.138}$$

Since it is readily verified that

$$E[\bar{x}(n+1)X(n)] = \bar{V}_1 e^{-j\omega} \tag{5.139}$$

the final expression is

$$A = \frac{1}{N + \sigma_b^2}[e^{-j\omega}, e^{-j2\omega}, \ldots, e^{-jN\omega}]^t \tag{5.140}$$

The same procedure can be applied to a signal made of two sinusoids in noise:

$$x_1(n) = e^{jn\omega_1} + e^{jn\omega_2} + b(n) \tag{5.141}$$

with the AC matrix

$$R_1 = \sigma_b^2 I_N + \bar{V}_1 V_1^t + \bar{V}_2 V_2^t \tag{5.142}$$

The matrix inversion lemma can be invoked again to obtain

$$R_1^{-1} = R_N^{-1} - R_N^{-1} \bar{V}_2 [V_2^t R_N^{-1} \bar{V}_2 + 1]^{-1} V_2^t R_N^{-1} \tag{5.143}$$

and, since

$$E[\bar{x}_1(n+1) X_1(n)] = \bar{V}_1 e^{-j\omega_1} + \bar{V}_2 e^{-j\omega_2} \tag{5.144}$$

the prediction coefficient vector is

$$A_1 = R_1^{-1} [\bar{V}_1 e^{-j\omega_1} + \bar{V}_2 e^{-j\omega_2}] \tag{5.145}$$

This is a complicated expression. In the special case when $V_2 = \bar{V}_1$, i.e., when $\omega_2 = -\omega_1 = \omega$, and ω is a multiple of π/N, it is readily verified that expression (5.129) is obtained.

The approach can be extended to signals made of M sinusoids in noise, to yield an exact solution for the prediction coefficient vector.

5.13. LINEAR PREDICTION AND HARMONIC DECOMPOSITION

Two different representations of a signal given by the first $N+1$ terms $[r(0), r(1), \ldots, r(N)]$ of its ACF have been obtained. The harmonic decomposition presented in Section 2.11 corresponds to the modeling by a set of sinusoids and is also called composite sinusoidal modeling (CSM); it yields the following expression for the signal spectrum $S(\omega)$ according to relation (2.127) of Chapter 2:

$$S(\omega) = \sum_{k=1}^{N/2} |S_k|^2 [\delta(\omega - \omega_k) + \delta(\omega + \omega_k)] \tag{5.146}$$

Linear prediction provides a representation of the signal spectrum by

$$S(\omega) = \frac{\sigma_e^2}{\left| 1 - \sum_{i=1}^{N} a_i e^{-ji\omega} \right|^2} \tag{5.147}$$

Relations between these two approaches can be established by considering the decomposition of the z-transfer function of the prediction error filter into two parts with symmetric and antisymmetric coefficients, which is the line spectrum pair (LSP) representation [12].

The order recursion (5.29) is expressed in terms of z-polynomials by

$$1 - A_N(z) = 1 - A_{N-1}(z) - k_N z^{-N} [1 - A_{N-1}(z^{-1})] \tag{5.148}$$

where

$$1 - A_N(z) = 1 - \sum_{i=1}^{N} a_{iN} z^{-i} \tag{5.149}$$

Let us consider now the order $N+1$ and denote by $P_N(z)$ the polynomial obtained when $k_{N+1} = 1$:

$$P_N(z) = 1 - A_N(z) - z^{-(N+1)}[1 - A_N(z^{-1})] \tag{5.150}$$

Let $Q_N(z)$ be the polynomial obtained when $k_{N+1} = -1$:

$$Q_N(z) = 1 - A_N(z) + z^{-(N+1)}[1 - A_N(z^{-1})] \tag{5.151}$$

Clearly, this is a decomposition of the polynomial (5.114):

$$1 - A_N(z) = \tfrac{1}{2}[P_N(z) + Q_N(z)] \tag{5.152}$$

and $\frac{1}{2}P_N(z)$ and $\frac{1}{2}Q_N(z)$ are polynomials with antisymmetric and symmetric coefficients, respectively.

Since $k_{N+1} = \pm 1$, due to the results in Section 5.6 and equation (5.63), $P_N(z)$ and $Q_N(z)$ have all their zeros on the unit circle. Furthermore, if N is even, it is readily verified that $P_N(1) = 0 = Q_N(-1)$. Therefore, the following factorization is obtained:

$$\begin{aligned} P_N(z) &= (1 - z^{-1}) \prod_{i=1}^{N/2} (1 - 2\cos(\theta_i) z^{-1} + z^{-2}) \\ Q_N(z) &= (1 + z^{-1}) \prod_{i=1}^{N/2} (1 - 2\cos(\omega_i) z^{-1} + z^{-2}) \end{aligned} \tag{5.153}$$

The two sets of parameters θ_i and $\omega_i (1 \leqslant i \leqslant N)$ are called the LSP parameters.

If $z_0 = e^{j\omega_0}$ is a zero of the polynomial $1 - A(z)$ on the unit circle, it is also a zero of $P_N(z)$ and $Q_N(z)$. Now if this zero moves inside the unit circle, the corresponding zeros of $P_N(z)$ and $Q_N(z)$ move on the unit circle in opposite directions from ω_0. A necessary and sufficient condition for the polynomial $1 - A(z)$ to be minimum phase is that the zeros of $P_N(z)$ and $Q_N(z)$ be simple and alternate on the unit circle [13].

The above approach provides a realization structure for the prediction error filter in Figure 5.12. The z-transfer functions $F(z)$ and $G(z)$ are the linear phase factors in (5.153). This structure is amenable to implementation as a cascade of second-order sections, and the overall minimum phase property is checked by observing the alternation of the z^{-1} coefficients. It can be used for predictors with poles and zeros [14].

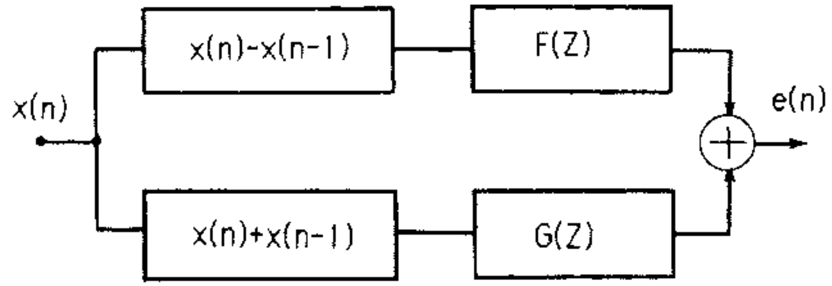


FIG. 5.12 Line pair spectrum predictor.

Equations (5.153) show that the LSP parameters θ_i and ω_i are obtained by harmonic decomposition of the sequences $x(n) - x(n-1)$ and $x(n) + x(n-1)$. This is an interesting link beween harmonic decomposition, or CSM, and linear prediction.

So far, the linear prediction problem has been solved using the ACF function of the signal. However, it is also possible, and in some situations necessary, to find the prediction coefficients directly from the signal samples.

5.14. ITERATIVE DETERMINATION OF THE RECURRENCE COEFFICIENTS OF A PREDICTABLE SIGNAL

A predictable signal of order p, by definition satisfies the recurrence relation

$$x(n) = \sum_{i=1}^{p} a_i x(n-i) \tag{5.154}$$

Considering this equation for p different values of the index n leads to a system of p equations and p unknowns, which can be solved for the p prediction coefficients. In matrix form,

$$\begin{bmatrix} x(p) & x(p-1) & \cdots & x(1) \\ x(p+1) & x(p) & \cdots & x(2) \\ \vdots & \vdots & & \vdots \\ x(2p-1) & x(2p-2) & \cdots & x(p) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} x(p+1) \\ x(p+2) \\ \vdots \\ x(2p) \end{bmatrix} \tag{5.155}$$

An efficient solution is provided by an iterative technique consisting of pth-order recursions. The approach is as follows. Assume that the system has been solved at order $N < p$. A set of N prediction coefficients has been found satisfying

$$\begin{bmatrix} x(p) & x(p-1) & \cdots & x(p+1-N) \\ x(p+1) & x(p) & \cdots & x(p+2-N) \\ \vdots & \vdots & & \vdots \\ x(p+N-1) & x(p+N-2) & \cdots & x(p) \end{bmatrix} \begin{bmatrix} a_{1N} \\ a_{2N} \\ \vdots \\ a_{NN} \end{bmatrix} = \begin{bmatrix} x(p+1) \\ x(p+2) \\ \vdots \\ x(p+N) \end{bmatrix} \tag{5.156}$$

In a more concise form,

$$R_N A_N = J X_N(p+N) \tag{5.157}$$

where J is the coidentity matrix

$$J = \begin{bmatrix} 0 & \dots & 1 \\ \vdots & \cdot^{\cdot^{\cdot}} & \vdots \\ 1 & \dots & 0 \end{bmatrix}; \; X_N(p+N) = \begin{bmatrix} x(p+N) \\ x(p+N-1) \\ \vdots \\ x(p+1) \end{bmatrix}$$

and R_N designates the $N \times N$ matrix of the input data involved in the system of equations (5.156).

Referring to the forward linear prediction matrix equation, one can write

$$R_{N+1} \begin{bmatrix} 1 \\ -A_N \end{bmatrix} = \begin{bmatrix} e_N \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.158}$$

where

$$e_N = x(p) - \sum_{i=1}^{N} a_{iN} x(p-i) \tag{5.159}$$

and, in concise form,

$$e_N = x(p) - A_N^t X_N(p-1) = x(p) - X_N^t(p+N) J (R_N^{-1})^t X_N(p-1)$$

The same procedure can be applied to the backward linear prediction, and a coefficient vector B_N can be computed by

$$R_N \begin{bmatrix} b_{NN} \\ b_{N-1\,N} \\ \vdots \\ b_{1N} \end{bmatrix} = \begin{bmatrix} x(p-N) \\ x(p+1-N) \\ \vdots \\ x(p-1) \end{bmatrix} = J X_N(p-1) \tag{5.160}$$

From the definition of R_{N+1}, the following equation is obtained:

$$R_{N+1}\begin{bmatrix} -B_N \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ e_N \end{bmatrix} \tag{5.161}$$

The presence of e_N in the right-hand side comes from the equation

$$x(p) - X_N^t(p+N)B_N = x(p) - X_N^t(p+N)R_N^{-1}JX_N(p-1)$$

Now, since

$$(R_N^{-1})^t = JR^{-1}J, \qquad JJ = I_N \tag{5.162}$$

it is clear that

$$e_N = x(p) - X_N^t(p+N)B_N = x(p) - X_N^t(p-1)A_N$$

At this stage, the prediction coefficient vectors A_{N+1} and B_{N+1} can be readily obtained, starting from the equation

$$R_{N+2}\begin{bmatrix} 1 \\ -A_N \\ 0 \end{bmatrix} = \begin{bmatrix} e_N \\ 0 \\ \vdots \\ 0 \\ e_{aN} \end{bmatrix} \tag{5.163}$$

where

$$e_{aN} = x(p+N+1) - X_N^t(p+N)A_N \tag{5.164}$$

As concerns backward prediction, the equation is

$$R_{N+2}\begin{bmatrix} 0 \\ -B_N \\ 1 \end{bmatrix} = \begin{bmatrix} e_{bN} \\ 0 \\ \vdots \\ 0 \\ e_N \end{bmatrix} \tag{5.164a}$$

where

$$e_{bN} = x(p-N-1) - X_N^t(p-1)B_N \tag{5.165}$$

In fact, two different decompositions of R_{N+2} are exploited, namely

$$\begin{aligned} R_{N+2} &= \begin{bmatrix} R_{N+1} & JX_{N+1}(p-1) \\ X_{N+1}^t(p+N+1) & x(p) \end{bmatrix} \\ &= \begin{bmatrix} x(p) & X_{N+1}^t(p-1) \\ JX_{N+1}(p+N+1) & R_{N+1} \end{bmatrix} \end{aligned}$$

In order to get the equations for linear prediction at order $N+1$, it is necessary to get rid of the last element in the right-hand side of equation (5.163) and the first element in the right-hand side of equation (5.164). This can be accomplished, assuming $e_N \neq 0$, by the substitution leading to the matrix equation

$$R_{N+2}\left[\begin{bmatrix} 1 \\ -A_N \\ 0 \end{bmatrix} - \frac{e_{aN}}{e_N}\begin{bmatrix} 0 \\ -B_N \\ 1 \end{bmatrix}\right] = \begin{bmatrix} e_N - \dfrac{e_{aN}e_{bN}}{e_N} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.166}$$

and, for backward prediction

$$R_{N+2}\left[\begin{bmatrix} 0 \\ -B_N \\ 1 \end{bmatrix} - \frac{e_{bN}}{e_N}\begin{bmatrix} 1 \\ -A_N \\ 1 \end{bmatrix}\right] = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ e_N - \dfrac{e_{aN}e_{bN}}{e_N} \end{bmatrix} \tag{5.167}$$

Through direct identification of the factors in the equations for forward and backward linear prediction at order $N+1$, the recurrence relations for the coefficient vectors are obtained. For forward linear prediction, one gets

$$A_{N+1} = \begin{bmatrix} A_N \\ 0 \end{bmatrix} + \frac{e_{aN}}{e_N}\begin{bmatrix} -B_N \\ 1 \end{bmatrix} \tag{5.168}$$

and, for backward linear prediction,

$$B_{N+1} = \begin{bmatrix} 0 \\ B_N \end{bmatrix} + \frac{e_{bN}}{e_N}\begin{bmatrix} 1 \\ -A_N \end{bmatrix} \tag{5.169}$$

The variable e_N can itself be computed recursively by

$$e_{N+1} = e_N - \frac{e_{aN}e_{bN}}{e_N} = e_N\left(1 - \frac{e_{aN}e_{bN}}{e_N^2}\right) \tag{5.170}$$

Finally, the algorithm is given in Figure 5.13. The computational complexity, at order N is $4(N+1)$ multiplications and one division. The total operation count for order p is $2(p+1)(p+2)$ multiplications and p divisions.

Available at order N : A_N, B_N, e_N
New data $x(p+N+1)$, $x(p-N-1)$

$$e_{aN} = x(p+N+1), \ -X_N^t(p+N)A_N$$

$$e_{bN} = x(p-N-1) - X_N^t(p-1)B_N$$

$$e_{N+1} = e_N\left(1 - \frac{e_{aN}}{e_N}\frac{e_{bN}}{e_N}\right)$$

$$A_{N+1} = \begin{bmatrix} A_N \\ 0 \end{bmatrix} + \frac{e_{aN}}{e_N}\begin{bmatrix} -B_N \\ 1 \end{bmatrix}$$

$$B_{N+1} = \begin{bmatrix} 0 \\ B_N \end{bmatrix} + \frac{e_{bN}}{e_N}\begin{bmatrix} 1 \\ -A_N \end{bmatrix}$$

FIG. 5.13 Algorithm for the computation of the linear prediction coefficients.

The algorithm obtained is useful in some spectral analysis techniques. Its counterpart in finite fields is used in error correction, for example, for the decoding of Reed–Solomon codes.

5.15. CONCLUSION

Linear prediction error filters have been studied. Properties and coefficient design techniques have been presented. The analysis of first- and second-order filters yields simple results which are useful in signal analysis, particularly for the detection of sinusoidal components in a spectrum. Backward linear prediction provides a set of uncorrelated sequences. Combined with forward prediction, it leads to order iterative relations which correspond to a particular structure, the lattice filter. The lattice or PARCOR coefficients enjoy a number of interesting properties, and they can be calculated from the signal ACF by efficient algorithms.

The inverse AC matrix, which is involved in LS algorithms, can be expressed in terms of forward and backward prediction coefficients and prediction error power. To manipulate prediction filters and fast algorithms, it is important that we be able to locate the zeros in the unit circle; the analysis based on the notch filter and carried out for sinusoids in noise provides an insight useful for more general signals.

The gradient adaptive techniques apply to linear prediction filters with a number of simplifications, and the lattice structure is an appealing alternative to the transversal structure. An additional realization option is offered by the LSP approach, which provides an interesting link between linear prediction and harmonic decomposition.

EXERCISES

1. Calculate the impulse responses $h_{ji}(1 \leqslant j \leqslant 3; 0 \leqslant i \leqslant 6)$ corresponding to the following z-transfer functions:

$$H_1(z) = (1 + z^{-1} + 0.5z^{-2})^2$$
$$H_2(z) = \tfrac{1}{2}(1 + z^{-1} + 0.5z^{-2})(1 + 2z^{-1} + 2z^{-2})$$
$$H_3(z) = \tfrac{1}{4}(1 + 2z^{-1} + 2z^{-2})^2$$

   Calculate the functions

$$E_j(n) = \sum_{i=0}^{n} h_{ji}^2, \quad 0 \leqslant n \leqslant 6, 1 \leqslant j \leqslant 3$$

   and draw the curves $E_j(n)$ versus n.

   Explain the differences between minimum phase, linear phase, and maximum phase.

2. Calculate the first four terms of the ACF of the signal

$$x(n) = \sqrt{2}\sin\left(n\frac{\pi}{4}\right)$$

   Using the normal equations, calculate the coefficients of the predictor of order $N = 3$. Locate the zeros of the prediction error filter in the complex z-plane. Perform the same calculations when a white noise with power $a_b^2 = 0.1$ is added to the signal and compare with the above results.

3. Consider the signal

$$x(n) = \sin(n\omega_1) + \sin(n\omega_2)$$

   Differentiating (5.6) with respect to the coefficients and setting these derivatives to zero, calculate the coefficients of the predictor of order $N = 2$. Show the equivalence with solving linear prediction equations. Locate the zeros of the prediction error filter in the complex z-plane and comment on the results.

4. Calculate the coefficients a_1 and a_2 of the notch filter with transfer function

$$H(z) = \frac{1 + a_1 z^{-1} + a_2 z^{-2}}{1 + (1-\varepsilon)a_1 z^{-1} + (1-\varepsilon)^2 a_2 z^{-2}}, \quad \varepsilon = 0.1$$

which cancels the signal $x(n) = \sin(0.7n)$.

Locate the poles and zeros in the complex plane. Give the frequencies which satisfy $|H(e^{j\omega})| = 1$ and calculate $H(1)$ and $H(-1)$. Draw the function $|H(\omega)|$.

Express the white noise amplification factor of the filter as a function of the parameter ε.

5. Use the Levinson–Durbin algorithm to compute the PARCOR coefficients associated with the correlation sequence

$$r(0) = 1, \quad r(n) = \alpha 0.9^n \quad 0 < \alpha \leqslant 1$$

Give the diagram of the lattice filter with three sections. Comment on the case $\alpha = 1$.

6. Calculate the inverse of the 3×3 AC matrix R_3. Express the prediction coefficients a_1 and a_2 and the prediction error E_2. Compute R_3^{-1} using relation (5.67) and compare with the direct calculation result.

7. Consider the ARMA signal

$$x(n) = e(n) - 0.5e(n-1) - 0.9x(n-1)$$

where $e(n)$ is a unit power white noise. Express the coefficients of the FIR predictor of infinite order.

Using the results of Section 2.6 on ARMA signals, calculate the AC function $r(n)$ for $0 \leqslant n \leqslant 3$. Give the coefficients of the prediction filters of orders 1, 2, and 3 and compare with the first coefficients of the infinite predictor. Locate the zeros in the complex plane.

8. The continuous signal $x(n) = 1$ is applied from time zero on to the adaptive IIR prediction error filter, whose equations are

$$e(n+1) = x(n+1) - b(n)e(n)$$
$$b(n+1) = b(n) + \delta e(n+1)e(n)$$

For $\delta = 0.2$ and zero initial conditions, calculate the coefficient sequence $b(n)$, $1 \leqslant n \leqslant 20$. How does the corresponding pole move in the complex z-plane?

A noise with power σ_b^2 is added to the input signal. Calculate the optimum value of the first-order IIR predictor. Give a lower bound for σ_b^2 which prevents the pole from crossing the unit circle. When there is no noise, what value of the leakage factor has the same effect.

9. Give the LSP decomposition of the prediction filter

$$1 - A_N(z) = (1 - 1.6z^{-1} + 0.9z^{-2})(1 - z^{-1} + z^{-2})$$

Locate the zeros of the polynomials obtained. Give the diagram of the adaptive realization, implemented as a cascade of second-order filter sections.

10. Use the algorithm of Figure 5.13 to show that the linear prediction coefficients of the length $2p = 12$ sequence

$$1.707; 1; -0.293; 0; 0.293; -1; -1.707; 0; 1.707; 1; -0.293; 0$$

are given by

$$1 - A_N(z) = (1 + z^{-2})(1 - 1.414z^{-1} + z^{-2})$$

Give the general expression of the input sequence $x(n)$.

ANNEX 5.1 LEVINSON ALGORITHM

```
      SUBROUTINE LEV(N,Q,X,B)
C
C     SOLVES THE SYSTEM : [R]X=B WITH [R] TOEPLITZ MATRIX
C     N = SYSTEM ORDER ( 2 < N < 17 )
C     Q = N+1 ELEMENT AUTOCORRELATION VECTOR :
      r(0, ......,N)
C     X = SOLUTION VECTOR
C     B = RIGHT SIDE VECTOR

      DIMENSION Q(1),X(1),B(1),A(16),Y(16)
      A(1)=-Q(2)/Q(1)
      X(1)=B(1)/Q(1)
      RE=Q(1)+A(1)*Q(2)
      DO60I=2,N
      T=Q(I+1)
      DO10J=1,I-1
   10 T=T+Q(I-J+1)*A(J)
      A(I)=-T/RE
      DO20J=1,I-1
   20 Y(J)=A(J)
      DO30J=1,I-1
   30 A(J)=Y(J)+A(I)*Y(I-J)
      S=B(I)
      DO40J=1,I-1
   40 S=S-Q(I-J+1)*X(J)
      X(I)=S/RE
      DO50J=1,I-1
   50 X(J)=X(J)+X(I)*Y(I-J)
      RE=RE+A(I)*T
   60 CONTINUE
      RETURN
      END
```

ANNEX 5.2 LEROUX-GUEGUEN ALGORITHM

```
      SUBROUTINE LGPC(N,R,RK)
C
C     LEROUX-GUEGUEN  Algorithm for computing the PARCOR
C     coeff. from AC-function.
C     N =Number of coefficients
C     R =Correlation coefficients (INPUT)
C     RK=Reflexion coefficients (OUTPUT)
C
      DIMENSION R(20),RK(20),RE(20),RH(20)
      RK(1)=R(2)/R(1)
      RE(1)=R(2)
      RE(2)=R(1)-RK(1)*R(2)
      DO10I=2,N
      X=R(I+1)
      RH(1)=X
      I1=I-1
      DO20J=1,I1
      RH(J+1)=RE(J)-RK(J)*X
      X=X-RK(J)*RE(J)
   20 RE(J)=RH(J)
      RK(I)=X/RE(I)
      RE(I+1)=RE(I)-RK(I)*X
      RE(I)=RH(I)
   10 CONTINUE
      RETURN
      END
```

REFERENCES

1. J. Makhoul, “Linear Prediction: A Tutorial Review,” *Proc. IEEE* **63**, 561–580 (April 1975).
2. J. L. Lacoume, M. Gharbi, C. Latombe, and J. L. Nicolas, “Close Frequency Resolution by Maximal Entropy Spectral Estimators,” *IEEE Trans.* **ASSP-32**, 977–983 (October 1984).
3. J. Leroux and C. Gueguen, “A Fixed Point Computation of Partial Correlation Coefficients,” *IEEE Trans.* **ASSP-25**, 257–259 (June 1977).
4. J. D. Markel and A. H. Gray, *Linear Prediction of Speech*, Springer-Verlag, New York, 1976.
5. B. Picinbono and M. Benidir, “Some Properties of Lattice Autoregressive Filters,” *IEEE Trans.* **ASSP-34**, 342–349 (April 1986).
6. D. V. B. Rao and S. Y. Kung, “Adaptive Notch Filtering for the Retrieval of Sinusoids in Noise,” *IEEE Trans.* **ASSP-32**, 791–802 (August 1984).

7. J. M. Travassos-Romano and M. Bellanger, ‘‘Zeros and Poles of Linear Prediction Digital Filters,’’ *Proc. EUSIPCO-86*, North-Holland, The Hague, 1986, pp. 123–126.
8. M. Jaidane-Saidane and O. Macchi, ‘‘Self Stabilization of IIR Adaptive Predictors, with Application to Speech Coding,’’ *Proc. EUSIPCO-86*, North-Holland, The Hague, 1986, pp. 427–430.
9. M. Honig and D. Messerschmitt, *Adaptive Filters, Structures, Algorithms and Applications*, Kluwer Academic, Boston, 1985, Chaps. 5–7.
10. G. Sohie and L. Sibul, ‘‘Stochastic Convergence Properties of the Adaptive Gradient Lattice,’’ *IEEE Trans.* **ASSP-32**, 102–107 (February 1984).
11. P. M. Grant and M. J. Rutter, ‘‘Application of Gradient Adaptive Lattice Filters to Channel Equalization,’’ *Proc. IEEE* **131F**, 473–479 (August 1984).
12. S. Sagayama and F. Itakura, ‘‘Duality Theory of Composite Sinusoidal Modeling and Linear Prediction,’’ *Proc. of ICASSP-86*, Tokyo, 1986, pp. 1261–1265.
13. H. W. Schussler, ‘‘A Stability Theorem for Discrete Systems,’’ *IEEE Trans.* **ASSP-24**, 87–89 (February 1976).
14. K. Hosoda and A. Fukasawa, ‘‘ADPCM Codec Composed by the Prediction Filter Including Poles and Zeros,’’ *Proc. EUSIPCO-83*, Elsevier, 1983, pp. 391–394.
15. N. Kalouptsidis and S. Theodoridis, *Adaptive System Identification and Signal Processing Algorithms*, Prentice-Hall, Englewood Cliffs, N.J., 1993.

6
Fast Least Squares Transversal Adaptive Filters

Least squares techniques require the inversion of the input signal AC matrix. In adaptive filtering, which implies real-time operations, recursive methods provide means to update the inverse AC matrix whenever new information becomes available. However, the inverse AC matrix is completely determined by the prediction coefficients and error power. The same applies to the real-time estimation of the inverse AC matrix, which is determined by FBLP coefficients and prediction error power estimations. In these conditions, all the information necessary for recursive LS techniques is contained in these parameters, which can be calculated and updated. Fast transversal algorithms perform that function efficiently for FIR filters in direct form.

The first-order LS adaptive filter is an interesting case, not only because it provides a gradual introduction to the recursive mechanisms, the initial conditions, and the algorithm performance, but also because it is implemented in several approaches and applications.

6.1. THE FIRST-ORDER LS ADAPTIVE FILTER

The first-order filter, whose diagram is shown in Figure 6.1, has a single coefficient $h_0(n)$ which is computed to minimize at time n a cost function, which is the error energy

$$E_1(n) = \sum_{p=1}^{n} [y(p) - h_0(n)x(p)]^2 \tag{6.1}$$

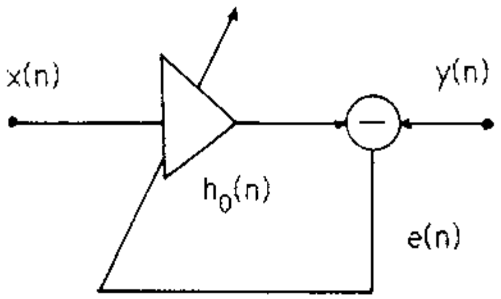


FIG. 6.1 Adaptive filter with a single coefficient.

The solution, obtained by setting to zero the derivative of $E_1(n)$ with respect to $h_0(n)$ is

$$h_0(n) = \frac{\sum_{p=1}^{n} y(p)x(p)}{\sum_{p=1}^{n} x^2(p)} = \frac{r_{yx}(n)}{r_{xx}(n)} \tag{6.2}$$

In order to derive a recursive procedure, let us consider

$$h_0(n+1) = r_{xx}^{-1}(n+1)[r_{yx}(n) + y(n+1)x(n+1)] \tag{6.3}$$

From expression (6.2), we have

$$[r_{xx}(n+1) - x^2(n+1)]h_0(n) = r_{yx}(n) \tag{6.4}$$

Hence

$$h_0(n+1) = h_0(n) + r_{xx}^{-1}(n+1)x(n+1)[y(n+1) - h_0(n)x(n+1)] \tag{6.5}$$

The filter coefficient is updated using the new data and the a priori error, defined previously by

$$e(n+1) = y(n+1) - h_0(n)x(n+1) \tag{6.6}$$

Recall that this error is named "a priori" because it uses the preceding coefficient value.

The scalar $r_{xx}(n+1)$ is the input signal energy estimate; it is updated by

$$r_{xx}(n+1) = r_{xx}(n) + x^2(n+1) \tag{6.7}$$

Together expressions (6.5) and (6.7) make a recursive procedure for the first-order LS adaptive filter. However, in practice, the recursive approach cannot be exactly equivalent to the theoretical LS algorithm, because of the initial conditions.

At time $n = 1$, a coefficient initial value $h_0(0)$ is needed by equation (6.5). If it is taken as zero, relation (6.5) yields

$$h_0(1) = \frac{y(1)}{x(1)} \tag{6.8}$$

which is the solution. However, in the second equation (6.7) it is not possible to take $r_{xx}(0) = 0$ because there is a division in (6.5) and $r_{xx}(1)$ has to be greater than zero. Thus, the algorithm is started with a positive value, $r_{xx}(0) = r_0$, and the actual coefficient updating equation is

$$h_0(n+1) = h(n) + \frac{x(n+1)}{r_0 + \sum_{p=1}^{n+1} x^2(p)} [y(n+1) - h_0(n)x(n+1)], \quad n \geqslant 0 \tag{6.9}$$

This equation still is a LS equation, but the criterion is different from (6.1). Instead, it can be verified that it is

$$E_1'(n) = \sum_{p=1}^{n} [y(p) - h_0(n)x(p)]^2 + r_0 h_0^2(n) \tag{6.10}$$

The consequence is the introduction of a time constant, which can be evaluated by considering the simplified case $y(n) = x(n) = 1$. With these signals, the coefficient evolution equation is

$$h_0(n+1) = h_0(n) + \frac{1}{r_0 + (n+1)} [1 - h_0(n)], \quad n \geqslant 0$$

or

$$h_0(n+1) = \left(1 - \frac{1}{r_0 + n + 1}\right) h_0(n) + \frac{1}{r_0 + n + 1}, \quad n \geqslant 0 \tag{6.11}$$

which, assuming $h_0(0) = 0$, leads to

$$h_0(n) = \frac{n}{r_0 + n} = 1 - \frac{1}{1 + n/r_0} \tag{6.12}$$

The evolution of the coefficient is shown in Figure 6.2 for different values of the initial constant r_0. Note that negative values can also be taken for r_0.

Definition (4.10) in Chapter 4 yields the coefficient time constant $\tau_c \approx r_0$. Clearly, the initial constant r_0 should be kept as small as possible; the lower limit is determined by the computational accuracy in the realization.

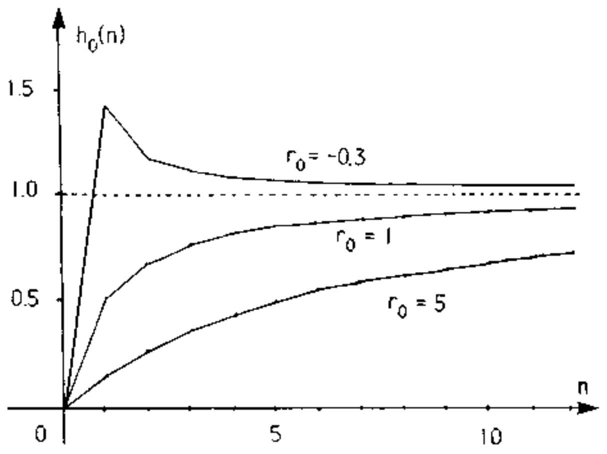


FIG. 6.2 Evolution of the coefficient of a first-order LS adaptive filter.

Adaptive filters, in general, are designed with the capability of handling nonstationary signals, which is achieved through the introduction of a limited memory. An efficient approach consists of introducing a memory-limiting or -forgetting factor $W (0 \ll W < 1)$, which corresponds to an exponential weighting operation in the cost function:

$$E_{W1}(n) = \sum_{p=1}^{n} W^{n-p} [y(p) - h_0(n)x(p)]^2 \tag{6.13}$$

Taking into account the initial constant r_0, we obtain the actual cost function

$$E'_{W1}(n) = \sum_{p=1}^{n} W^{n-p} [y(p) - h_0(n)x(p)]^2 + W^n r_0 h_0^2(n) \tag{6.14}$$

The updating equation for the coefficient becomes

$$h_0(n+1) = h_0(n) + \frac{x(n+1)}{r_0 W^{n+1} + \sum_{p=1}^{n+1} W^{n+1-p} x^2(p)} \times [y(n+1) - h_0(n)x(n+1)], \quad n \geqslant 0 \tag{6.15}$$

In the simplified case $x(n) = y(n) = 1$, if we assume $h_0(0) = 0$, we get

$$h_0(n+1) - h_0(n) + \frac{1}{r_0 W^{n+1} + (1 - W^{n+1})/(1-W)}[1 - h_0(n)], \quad n \geqslant 0 \tag{6.16}$$

Now the coefficient time constant is $\tau_{cW} \approx Wr_0$. But for n sufficiently large, the updating equation approaches

$$h_0(n+1) = Wh_0(n) + 1 - W \tag{6.17}$$

which corresponds to the long-term time constant

$$\tau \approx \frac{1}{1-W}$$

The curves $1 - h_0(n)$ versus time are shown in Figure 6.3 for $r_0 = 1$ and $W = 0.95$ and $W = 1$. Clearly, the weighting factor W can accelerate the convergence of $h_0(n)$ toward its limit.

For the LMS algorithm with step size δ under the same conditions, one gets

$$h_0(n) = 1 - (1-\delta)^n \tag{6.18}$$

The corresponding curve in Figure 6.3 illustrates the advantage of LS techniques in the initial phase.

In the recursive procedure, only the input signal power estimate is affected by the weighting operation, and equation (6.7) becomes

$$r_{xx}(n+1) = Wr_{xx}(n) + x^2(n+1)$$

In transversal filters with several coefficients, the above scalar operations become matrix operations and a recursive procedure can be worked out to avoid matrix inversion.

6.2. RECURSIVE EQUATIONS FOR THE ORDER *N* FILTER

The adaptive filter of order N is defined in matrix equations by

$$e(n+1) = y(n+1) - H^t(n)X(n+1) \tag{6.19}$$

where the vectors $H(n)$ and $X(n)$ have N elements. The cost function, which is the error energy

$$E_N(n) = \sum_{p=1}^{n} W^{n-p}[y(p) - H^t(n)X(p)]^2 \tag{6.20}$$

leads, as shown in Section 1.4, to the least squares solution

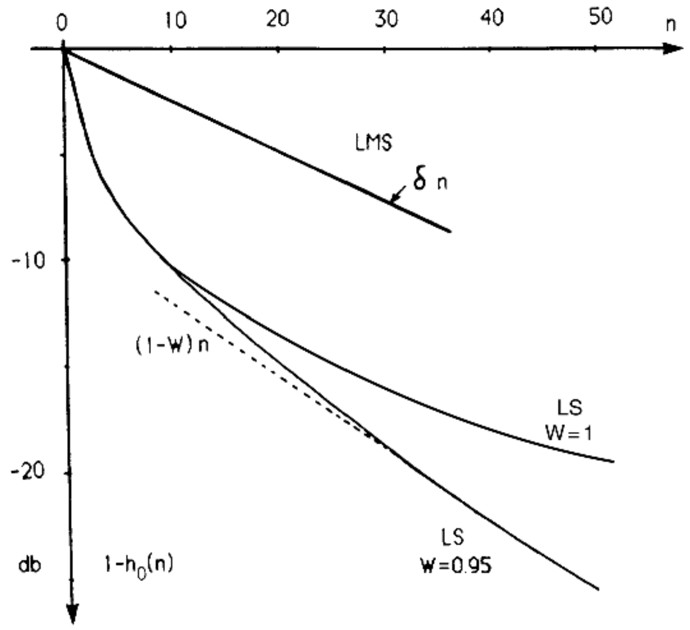


FIG. 6.3 Evolution of the coefficient error for two weighting factor values.

$$H(n) = R_N^{-1}(n) r_{yx}(n) \tag{6.21}$$

with

$$R_N(n) = \sum_{p=1}^{n} W^{n-p} X(p) X^t(p), \quad r_{yx}(n) = \sum_{p=1}^{n} W^{n-p} y(p) X(p) \tag{6.22}$$

As shown in Section 1.5, two recurrence relations can be derived from (6.21) and (6.22). Equation (1.25) is repeated here for convenience

$$H(n+1) = H(n) + R_N^{-1}(n+1) X(n+1)[y(n+1) - X^t(n+1)H(n)] \tag{6.23}$$

The matrix $R_N^{-1}(n+1)$ in that expression can be updated recursively with the help of a matrix identity called the matrix inversion lemma [1]. Given matrices A, B, C, and D satisfying the equation

$$A = B + CDC^t$$

the inverse of matrix A is

$$A^{-1} = B^{-1} - B^{-1}C[C^tB^{-1}C + D^{-1}]^{-1}C^tB^{-1} \tag{6.24}$$

The matrix A^{-1} can appear in various forms, which can be derived from the identity

$$(B - UDV)^{-1} = [I_N - B^{-1}UDV]^{-1}B^{-1}$$

where B is assumed nonsingular, through the generic power series expansion

$$(I_N - B^{-1}UDV)^{-1}B^{-1} = [I_N + B^{-1}UDV + (B^{-1}UDV)^2 + \cdots]B^{-1} \tag{6.25}$$

The convergence of the series is obtained if the eigenvalues of $(B^{-1}UDV)$ are less than unity. Expression (6.25) is a generalized matrix inversion lemma [2]. Consider, for example, regrouping and summing all terms but the first in (6.25) to obtain

$$(B - UDV)^{-1} = I_N + B^{-1}U[I_N - DVB^{-1}U]^{-1}DVB^{-1} \tag{6.26}$$

which is another form of (6.24).

This lemma can be applied to the calculation of $R_N^{-1}(n+1)$ in such a way that no matrix inversion is needed, just division by a scalar. Since

$$R_N(n+1) = WR_N(n) + X(n+1)X^t(n+1) \tag{6.27}$$

let us choose

$$B = WR_N(n), \quad C = X(n+1), \quad D = 1$$

then, lemma (6.24) yields

$$R_N^{-1}(n+1) = \frac{1}{W}\left[R_N^{-1}(n) - \frac{R_N^{-1}(n)X(n+1)X^t(n+1)R_N^{-1}(n)}{W + X^t(n+1)R_N^{-1}(n)X(n+1)}\right] \tag{6.28}$$

It is convenient to define the adaptation gain $G(n)$ by

$$G(n) = R_N^{-1}(n)X(n) \tag{6.29}$$

which, using (6.28) and after adequate simplifications, leads to

$$G(n+1) = \frac{1}{W + X^t(n+1)R_N^{-1}(n)X(n+1)}R_N^{-1}(n)X(n+1) \tag{6.30}$$

Now, expression (6.28) and recursion (6.23) can be rewritten as

$$R_N^{-1}(n+1) = \frac{1}{W}[R_N^{-1}(n) - G(n+1)X^t(n+1)R_N^{-1}(n)] \tag{6.31}$$

and

$$H(n+1) = H(n) + G(n+1)[y(n+1) - X^t(n+1)H(n)] \tag{6.32}$$

Relations (6.30)–(6.32) provide a recursive procedure to perform the filter coefficient updating without matrix inversion. Clearly, a nonzero initial value $R_N^{-1}(0)$ is necessary for the procedure to start; that point is discussed in a later section.

The number of arithmetic operations represented by the above procedure is proportional to N^2, because of the matrix multiplications involved. Matrix manipulations can be completely avoided, and the computational complexity made proportional to N only by considering that $R_N(n)$ is a real-time estimate of the input signal AC matrix and that, as shown in Chapter 5, its inverse can be represented by prediction parameters.

Before introducing the corresponding fast algorithms, several useful relations between LS variables are derived.

6.3. RELATIONSHIPS BETWEEN LS VARIABLES

In deriving the recursive least squares (RLS) procedure, the matrix inversion is avoided by the introduction of an appropriate scalar. Let

$$\varphi(n+1) = \frac{W}{W + X^t(n+1)R_N^{-1}(n)X(n+1)} \tag{6.33}$$

It is readily verified, using (6.28), that

$$\varphi(n+1) = 1 - X^t(n+1)R_N^{-1}(n+1)X(n+1)$$

The scalar $\theta(n)$, defined by

$$\theta(n) = X^t(n)R_N^{-1}(n)X(n) \tag{6.34}$$

has a special interpretation in signal processing. First, it is clear from

$$\theta(n+1) = X^t(n+1)[WR_N(n) + X(n+1)X^t(n+1)]^{-1}X(n+1)$$

that, assuming the existence of the inverse matrix

$$\theta(n+1) \leqslant X^t(n+1)[X(n+1)X^t(n+1)]^{-1}X(n+1)$$

Since

$$[X(n+1)X^t(n+1)]X(n+1) = \|X(n+1)\|X(n+1) \tag{6.35}$$

where $\|X\|$ the Euclidean norm of the vector X, the inverse matrix $[X(n+1)X^t(n+1)]^{-1}$ by definition satisfies

$$[X(n+1)X^t(n+1)]^{-1}X(n+1) = \|X(n+1)\|^{-1}X(n+1) \tag{6.36}$$

and the variable $\theta(n)$ is bounded by

$$0 \leqslant \theta(n) \leqslant 1$$

Now, from Section 2.12, it appears that the term in the exponent of the joint density of N zero mean Gaussian variables, has a form similar to $\theta(n)$, which can be interpreted as its sample estimate—hence the name of likelihood variable given to $\theta(n)$ in estimation theory [3]. Thus, $\theta(n)$ is a measure of the likelihood that the N most recent input data samples come from a Gaussian process with AC matrix $R_N(n)$ determined from all the available past observations. A small value of $\theta(n)$ indicates that the recent input data are likely samples of a Gaussian signal, and a value close to unity indicates that the observations are unexpected; in the latter case, $X(n+1)$ is out of the current estimated signal space, which can be due to the time-varying nature of the signal statistics. As a consequence, $\theta(n)$ can be used to detect changes in the signal statistics. If the adaptation gain $G(n)$ is available, as in the fast algorithms presented below, $\theta(n)$ can be readily calculated by

$$\theta(n) = X^t(n)G(n) \tag{6.37}$$

From the definitions, $\varphi(n)$ and $\theta(n)$ have similar properties. Those relevant to LS techniques are presented next.

Postmultiplying both sides of recurrence relation (6.27) by $R_N^{-1}(n)$ yields

$$R_N(n+1)R_N^{-1}(n) = WI_N + X(n+1)X^t(n+1)R_N^{-1}(n) \tag{6.38}$$

Using the identity

$$\det[I_N + V_1V_2^t] = 1 + V_1^tV_2 \tag{6.39}$$

where V_1 and V_2 are N-element vectors, and the definition of $\varphi(n)$, one gets

$$\varphi(n+1) = W^N \frac{\det R_N(n)}{\det R_N(n+1)} \tag{6.40}$$

Because of the definition of $R_N(n)$ and its positiveness and recurrence relation (6.27), the variable $\varphi(n)$ is bounded by

$$0 \leqslant \varphi(n) \leqslant 1 \tag{6.41}$$

which, through a different approach, confirms (6.36). This is a crucial property, which can be used to check that the LS conditions are satisfied in realizations of fast algorithms.

Now, we show that the variable $\varphi(n)$ has a straightforward physical meaning. The RLS procedure applied to forward linear prediction is based on a cost function, which is the prediction error energy

$$E_a(n) = \sum_{p=1}^{n} W^{n-p}[x(p) - A^t(n)X(p-1)]^2 \tag{6.42}$$

The coefficient vector is

$$A(n) = R_N^{-1}(n-1)r_N^a(n) \tag{6.43}$$

with

$$r_N^a(n) = \sum_{p=1}^{n} W^{n-p} x(p) X(p-1) \tag{6.44}$$

The index $n-1$ in (6.43) is typical of forward linear prediction, and the RLS coefficient updating equation is

$$A(n+1) = A(n) + G(n)e_a(n+1) \tag{6.45}$$

where

$$e_a(n+1) = x(n+1) - A^t(n)X(n) \tag{6.46}$$

is the a priori forward prediction error.

The updated coefficients $A(n+1)$ are used to calculate the a posteriori prediction error

$$\varepsilon_a(n+1) = x(n+1) - A^t(n+1)X(n) \tag{6.47}$$

or

$$\varepsilon_a(n+1) = e_a(n+1)[1 - G^t(n)X(n)] \tag{6.48}$$

From definition (6.33) we have

$$\varphi(n) = \frac{\varepsilon_a(n+1)}{e_a(n+1)} \tag{6.49}$$

and $\varphi(n)$ is the ratio of the forward prediction errors at the next time. This result can lead to another direct proof of inequality (6.41).

A similar result can also be obtained for backward linear prediction. The cost function used for the RLS procedure is the backward prediction error energy

$$E_b(n) = \sum_{p=1}^{n} W^{n-p}[x(p-N) - B^t(n)X(p)]^2 \tag{6.50}$$

The backward coefficient vector is

$$B(n) = R_N^{-1}(n)r_N^b(n) \tag{6.51}$$

with

$$r_N^b(n) = \sum_{p=1}^{n} W^{n-p} x(p-N) X(p) \tag{6.52}$$

The coefficient updating equation is now

$$B(n+1) = B(n) + G(n+1)e_b(n+1) \tag{6.53}$$

with

$$e_b(n+1) = x(n+1-N) - B^t(n)X(n+1) \tag{6.54}$$

The backward a posteriori prediction error is

$$\varepsilon_b(n+1) = x(n+1-N) - B^t(n+1)X(n+1) \tag{6.55}$$

Substituting (6.53) into (6.55) gives

$$\varphi(n+1) = \frac{\varepsilon_b(n+1)}{e_b(n+1)} \tag{6.56}$$

which shows that $\varphi(n)$ is the ratio of the backward prediction errors at the same time index.

In fact, this is a general result, which applies to any adaptive filter, and the following equation is obtained in a similar manner:

$$\varphi(n+1) = \frac{\varepsilon(n+1)}{e(n+1)} \tag{6.57}$$

It is worth pointing out that this result can lead to another proof of inequality (6.41). Let us consider the error energy (6.20) at time $n+1$:

$$E_N(n+1) = W \sum_{p=1}^{n} W^{n-p}[y(p) - H^t(n+1)X(p)]^2 + \varepsilon^2(n+1) \tag{6.58}$$

and the variable

$$E'_N(n+1) = W \sum_{p=1}^{n} W^{n-p}[y(p) - H^t(n)X(p)]^2 + e^2(n+1) \tag{6.59}$$

By definition of the optimal set of coefficients, the two following inequalities hold

$$E'_N(n+1) \geqslant E_N(n+1) \tag{6.60}$$

and

$$E_N(n+1) - \varepsilon^2(n+1) \geqslant E'_N(n+1) - e^2(n+1) \tag{6.61}$$

As a consequence,

$$e^2(n+1) \geqslant \varepsilon^2(n+1) \tag{6.62}$$

The above results can be illustrated with the help of simple signals. For example, with $N = 2$ and $x(n)$ a sinusoidal signal, the direct application of the definition of $\varphi(n)$ yields, for large n

$$\varphi(n) \approx 2 - 2(1 - W) = 2W - 1 \tag{6.63}$$

This result can be generalized to any N, if the frequency ω in $x(n) = \sin n\omega$ satisfies the conditions: $\pi/N \leqslant \omega \leqslant \pi - \pi/N$.

Now, for $x(n)$ a white noise and W close to one,

$$E[\varphi(n)] \approx 1 - N(1 - W) \tag{6.64}$$

The forward prediction error energy can be computed recursively. Substituting equation (6.43) into the expression of $E_a(n+1)$ yields

$$E_a(n+1) = \sum_{p=1}^{n+1} W^{n+1-p} x^2(p) - A^t(n+1) r_N^a(n+1) \tag{6.65}$$

The recurrence relations for $A(n+1)$ and $r_N^a(n+1)$, in connection with the definitions for the adaptation gain and the prediction coefficients, yield after simplification

$$E_a(n+1) = WE_a(n) + e_a(n+1)\varepsilon_a(n+1) \tag{6.66}$$

Similarly, the backward prediction error energy can be calculated by

$$E_b(n+1) = WE_b(n) + e_b(n+1)\varepsilon_b(n+1) \tag{6.67}$$

These are fundamental recursive computations which are used in the fast algorithms.

6.4. FAST ALGORITHM BASED ON A PRIORI ERRORS

In the RLS procedure, the adaptation gain $G(n)$ used to update the coefficients is itself updated with the help of the inverse input signal AC matrix. In fast algorithms, prediction parameters are used instead [4].

Let us consider the $(N+1) \times (N+1)$ AC matrix $R_{N+1}(n+1)$; as pointed out in Chapter 5, it can be partitioned in two different manners, exploited in forward and backward prediction equations:

$$R_{N+1}(n+1) = \begin{bmatrix} \sum_{p=1}^{n+1} W^{n+1-p} x^2(p) & [r_N^a(n+1)]^t \\ r_N^a(n+1) & R_N(n) \end{bmatrix} \tag{6.68}$$

and

$$R_{N+1}(n+1) = \begin{bmatrix} R_N(n+1) & r_N^b(n+1) \\ [r_N^b(n+1)]^t & \sum_{p=1}^{n+1} W^{n+1-p} x^2(p-N) \end{bmatrix} \tag{6.69}$$

The objective is to find $G(n+1)$ satisfying

$$R_N(n+1)G(n+1) = X(n+1) \tag{6.70}$$

and it will be reached in two consecutive steps. In the first step, the adaptation gain at order $N+1$, a vector with $N+1$ elements, will be calculated from forward linear prediction parameters. Then, it will be used to derive the desired gain $G(n+1)$ with the help of backward linear prediction parameters.

Since $R_N(n)$ is present in (6.68), let us calculate

$$R_{N+1}(n+1)\begin{bmatrix} 0 \\ G(n) \end{bmatrix} = \begin{bmatrix} [r_N^a(n+1)]^t G(n) \\ X(n) \end{bmatrix} \tag{6.71}$$

From definitions (6.29) for the adaptation gain and (6.43) for the optimal forward prediction coefficients, we have

$$[r_N^a(n+1)]^t G(n) = A^t(n+1)X(n) \tag{6.72}$$

Introducing the a posteriori prediction error, we get

$$R_{N+1}(n+1)\begin{bmatrix} 0 \\ G(n) \end{bmatrix} = X_1(n+1) - \begin{bmatrix} \varepsilon_a(n+1) \\ 0 \end{bmatrix} \tag{6.73}$$

where $X_1(n)$ is the vector of the $N+1$ most recent input data. Similarly, partitioning (6.69) leads to

$$R_{N+1}(n+1)\begin{bmatrix} G(n+1) \\ 0 \end{bmatrix} = \begin{bmatrix} X(n+1) \\ [r_N^b(n+1)]^t G(n+1) \end{bmatrix} \tag{6.74}$$

From definitions (6.70) and (6.51), we have

$$[r_N^b(n+1)]^t G(n+1) = B^t(n+1)X(n+1) \tag{6.75}$$

and

$$R_N(n+1)\begin{bmatrix} G(n+1) \\ 0 \end{bmatrix} = X_1(n+1) - \begin{bmatrix} 0 \\ \varepsilon_b(n+1) \end{bmatrix} \tag{6.76}$$

Now, the adapttion gain at dimension $N+1$, denoted $G_1(n+1)$ with the above notation, is defined by

$$R_{N+1}(n+1)G_1(n+1) = X_1(n+1) \tag{6.77}$$

Then, equation (6.73) can be rewritten as

$$R_{N+1}(n+1)\left[G_1(n+1) - \begin{bmatrix} 0 \\ G(n) \end{bmatrix}\right] = \begin{bmatrix} \varepsilon_a(n+1) \\ 0 \end{bmatrix} \tag{6.78}$$

Equation (6.76) becomes

$$R_{N+1}(n+1)\left[G_1(n+1)-\begin{bmatrix}G(n+1)\\0\end{bmatrix}\right]=\begin{bmatrix}0\\\varepsilon_a(n+1)\end{bmatrix} \tag{6.79}$$

Now, linear prediction matrix equations will be used to compute $G_1(n+1)$ from $G(n)$, and then $G(n+1)$ from $G_1(n+1)$. The forward linear prediction matrix equation, combining (6.43) and (6.65), is

$$R_{N+1}(n+1)\begin{bmatrix}1\\-A(n+1)\end{bmatrix}=\begin{bmatrix}E_a(n+1)\\0\end{bmatrix} \tag{6.80}$$

Identifying factors in (6.80) and (6.78) yields

$$G_1(n+1)=\begin{bmatrix}0\\G(n)\end{bmatrix}+\frac{\varepsilon_a(n+1)}{E_a(n+1)}\begin{bmatrix}1\\-A(n+1)\end{bmatrix} \tag{6.81}$$

The backward linear prediction matrix equation is

$$R_{N+1}(n+1)\begin{bmatrix}-B(n+1)\\1\end{bmatrix}=\begin{bmatrix}0\\E_b(n+1)\end{bmatrix} \tag{6.82}$$

Identifying factors in (6.82) and (6.79) yields

$$G_1(n+1)-\begin{bmatrix}G(n+1)\\0\end{bmatrix}=\frac{\varepsilon_b(n+1)}{E_b(n+1)}\begin{bmatrix}-B(n+1)\\1\end{bmatrix} \tag{6.83}$$

The scalar factor on the right side need not be calculated; it is already available. Let us partition the adaptation gain vector

$$G_1(n+1)=\begin{bmatrix}M(n+1)\\m(n+1)\end{bmatrix} \tag{6.84}$$

with $M(n+1)$ having N elements; the scalar $m(n+1)$ is given by the last line of (6.83):

$$m(n+1)=\frac{\varepsilon_b(n+1)}{E_b(n+1)} \tag{6.85}$$

The N-element adaptation gain is updated by

$$G(n+1)=M(n+1)+m(n+1)B(n+1) \tag{6.86}$$

But the updated adaptation gain is needed to get $B(n+1)$. Substituting (6.53) into (6.86) provides an expression of the gain as a function of available quantities:

$$G(n+1)=\frac{1}{1-m(n+1)e_b(n+1)}[M(n+1)+m(n+1)B(n)] \tag{6.87}$$

Note that, instead, (6.86) can be substituted into the coefficient updating equation, allowing the computation of $B(n+1)$ first:

$$B(n+1) = \frac{1}{1 - m(n+1)e_b(n+1)} [B(n) + M(n+1)e_b(n+1)] \tag{6.88}$$

In these equations, a new scalar is showing up. Since one must always be careful with dividers, it is interesting to investigate its physical interpretation and appreciate its magnitude range. Combining (6.85) and the energy updating equation (6.67) yields

$$1 - m(n+1)e_b(n+1) = 1 - \frac{\varepsilon_b(n+1)e_b(n+1)}{E_b(n+1)} = \frac{WE_b(n)}{E_b(n+1)} \tag{6.89}$$

Thus, the divider $1 - m(n+1)e_b(n+1)$ is the ratio of two consecutive values of the backward prediction error energy, and its theoretical range is

$$0 < 1 - m(n+1)e_b(n+1) \leqslant 1 \tag{6.90}$$

Clearly, as time goes on, its value approaches unity, more so than when the prediction error is small. Incidentally, equation (6.89) is an alternative to (6.67) to update the backward prediction error energy. Overall a fast algorithm is available and the sequence of operations is given in Figure 6.4. The corresponding FORTRAN subroutine is given in Annex 6.1.

It is sometimes called the fast Kalman algorithm [4]. The LS initialization is obtained by taking $A(n) = B(n) = G(n) = 0$ and $E_a(0) = E_0$, a small positive constant, as discussed in a later section.

The adaptation gain updating requires $8N + 4$ multiplications and two divisions in the form of inverse calculations; in the filtering, $2N$ multiplications are involved. Approximately $6N$ memories are needed to store the coefficients and variables. The progress with respect to RLS algorithms is impressive; however, it is still possible to improve these figures.

The above algorithm is mainly based on the a priori errors; for example, the backward a posteriori prediction error is not calculated. If all the prediction errors are exploited, a better balanced and more efficient algorithm is derived [5, 6].

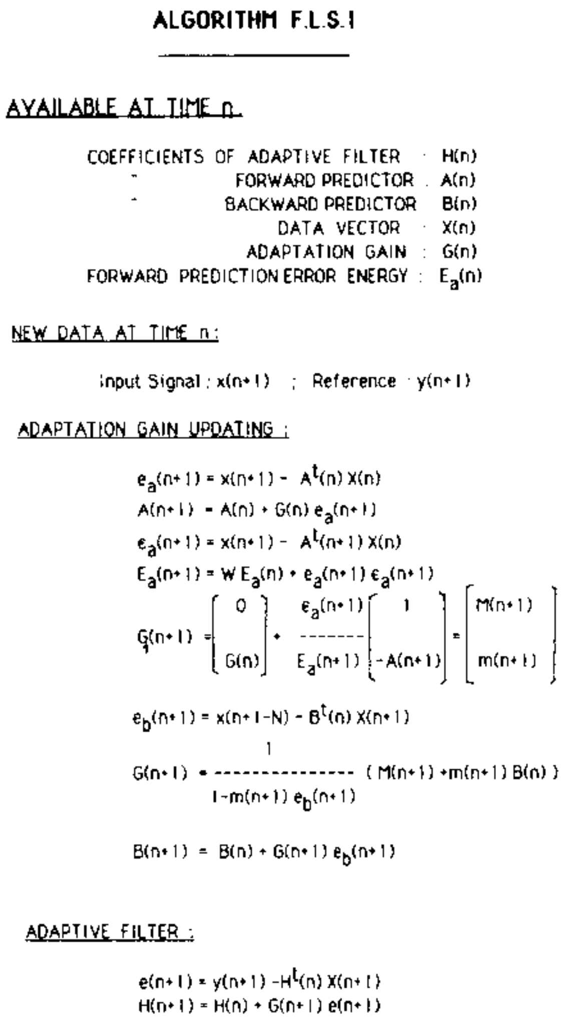
6.5. ALGORITHM BASED ON ALL PREDICTION ERRORS

Let us define an alternative adaptation gain vector with N elements, $G'(n)$, by

$$R_N(n)G'(n+1) = X(n+1) \tag{6.91}$$

Because of the term $R(n)$ in $G'(n+1)$, it is also called the a priori adaptation gain, in contrast with the a posteriori gain $G(n+1)$.

Similarly at order $N + 1$



ALGORITHM F.L.S.1

AVAILABLE AT TIME n.

COEFFICIENTS OF ADAPTIVE FILTER	$H(n)$
" FORWARD PREDICTOR	$A(n)$
" BACKWARD PREDICTOR	$B(n)$
DATA VECTOR	$X(n)$
ADAPTATION GAIN	$G(n)$
FORWARD PREDICTION ERROR ENERGY	$E_a(n)$

NEW DATA AT TIME n:

Input Signal : $x(n+1)$; Reference : $y(n+1)$

ADAPTATION GAIN UPDATING :

$$e_a(n+1) = x(n+1) - A^t(n)\,X(n)$$
$$A(n+1) = A(n) + G(n)\,e_a(n+1)$$
$$\varepsilon_a(n+1) = x(n+1) - A^t(n+1)\,X(n)$$
$$E_a(n+1) = W\,E_a(n) + e_a(n+1)\,\varepsilon_a(n+1)$$
$$G_1(n+1) = \begin{bmatrix} 0 \\ G(n) \end{bmatrix} + \frac{\varepsilon_a(n+1)}{E_a(n+1)} \begin{bmatrix} 1 \\ -A(n+1) \end{bmatrix} = \begin{bmatrix} M(n+1) \\ m(n+1) \end{bmatrix}$$
$$e_b(n+1) = x(n+1-N) - B^t(n)\,X(n+1)$$
$$G(n+1) = \frac{1}{1-m(n+1)\,e_b(n+1)}\,(M(n+1) + m(n+1)\,B(n))$$
$$B(n+1) = B(n) + G(n+1)\,e_b(n+1)$$

ADAPTIVE FILTER :

$$e(n+1) = y(n+1) - H^t(n)\,X(n+1)$$
$$H(n+1) = H(n) + G(n+1)\,e(n+1)$$

FIG. 6.4 Computational organization of the fast algorithm based on a priori errors.

$$R_{N+1}(n)G_1'(n+1) = X_1(n+1) \tag{6.92}$$

Exploiting, as in the previous section, the two different partitionings, (6.68) and (6.69), of the AC matrix estimation $R_{N+1}(n)$, one gets

$$R_{N+1}(n)\begin{bmatrix} G'(n+1) \\ 0 \end{bmatrix} = X_1(n+1) - \begin{bmatrix} 0 \\ e_b(n+1) \end{bmatrix} \tag{6.93}$$

and

$$R_{N+1}(n)\begin{bmatrix} 0 \\ G'(n) \end{bmatrix} = X_1(n+1) - \begin{bmatrix} e_b(n+1) \\ 0 \end{bmatrix} \tag{6.94}$$

Now, substituting definition (6.92) into (6.93) yields

$$R_{N+1}(n)\left[G_1'(n+1) - \begin{bmatrix} G'(n+1) \\ 0 \end{bmatrix} \right] = \begin{bmatrix} 0 \\ e_b(n+1) \end{bmatrix} \tag{6.95}$$

Identifying with the backward prediction matrix equation (6.82) gives a first expression for the order $N+1$ adaptation gain:

$$G_1'(n+1) = \begin{bmatrix} G'(n+1) \\ 0 \end{bmatrix} + \frac{e_b(n+1)}{E_b(n)} \begin{bmatrix} -B(n) \\ 0 \end{bmatrix} \tag{6.96}$$

Similarly (6.94) and (6.92) lead to

$$R_{N+1}(n)\left[G_1'(n+1) - \begin{bmatrix} 0 \\ G'(n) \end{bmatrix} \right] = \begin{bmatrix} e_a(n+1) \\ 0 \end{bmatrix} \tag{6.97}$$

Identifying with the forward prediction matrix equation (6.80) provides another expression for the gain:

$$G_1'(n+1) = \begin{bmatrix} 0 \\ G'(n) \end{bmatrix} + \frac{e_a(n+1)}{E_a(n)} \begin{bmatrix} 1 \\ -A(n) \end{bmatrix} \tag{6.98}$$

The procedure for calculating $G'(n+1)$ consists of calculating $G_1'(n+1)$ from the forward prediction parameters by (6.98) and then using (6.96).

Once the alternative gain $G'(n)$ is updated, it can be used in the filter coefficient recursion, provided it is adequately modified. It is necessary to replace $R_N^{-1}(n+1)$ by $R_N^{-1}(n)$ in equation (6.23). At time $n+1$ the optimal coefficient definition (6.21) is

$$[WR_N(n) + X(n+1)X^t(n+1)]H(n+1) = Wr_{Yx}(n) + y(n+1)X(n+1)$$

which, after some manipulation, leads to

$$H(n+1) = H(n) + W^{-1}R_N^{-1}(n)X(n+1)[y(n+1) - X^t(n+1)H(n+1)] \tag{6.99}$$

The a posteriori error

$$\varepsilon(n+1) = y(n+1) - X^t(n+1)H(n+1) \tag{6.100}$$

has to be calculated from available data; this is achieved with the help of the variable $\varphi(n)$ defined by (6.33), which is the ratio of a posteriori to a priori errors. From (6.33) we have

$$W + X^t(n+1)G'(n+1) = \frac{W}{\varphi(n+1)} = \alpha(n+1) \tag{6.101}$$

The variable $\alpha(n+1)$ is actually calculated in the algorithm.

Substituting $H(n+1)$ from (6.99) into (6.100) yields the kind of relationship already obtained for prediction:

$$\varepsilon(n+1) = \varphi(n+1)e(n+1) \tag{6.102}$$

Now the coefficient updating equation is

$$H(n+1) = H(n) + \frac{e(n+1)}{\alpha(n+1)}G'(n+1) \tag{6.103}$$

Note that, from the above derivations, the two adaptation gains are related by the scalar $\alpha(n+1)$ and an alternative definition of $G'(n+1)$ is

$$\begin{aligned} G'(n+1) &= [W + X^t(n+1)R_N^{-1}(n)X(n+1)]G(n+1) \\ &= \alpha(n+1)G(n+1) \end{aligned} \tag{6.104}$$

The variable $\alpha(n+1)$ can be calculated from its definition (6.101). However, a recursive procedure, similar to the one worked out for the adaptation gain, can be obtained. The variable corresponding to the order $N+1$ is $\alpha_1(n+1)$, defined by

$$\alpha_1(n+1) = W + X_1^t(n+1)G_1'(n+1) \tag{6.105}$$

The two different expressions for $G_1'(n+1)$, (6.96) and (6.98), yield

$$\alpha_1(n+1) = \alpha(n) + \frac{e_a^2(n+1)}{E_a(n)} = \alpha(n+1) + \frac{e_b^2(n+1)}{E_b(n)} \tag{6.106}$$

which provides the recursion for $\alpha(n+1)$ and $\varphi(n+1)$.

Since $\varphi(n+1)$ is available, it can be used to derive the a posteriori prediction errors $\varepsilon_a(n+1)$ and $\varepsilon_b(n+1)$, with only one multiplication instead of the N multiplications and additions required by the definitions.

The backward a priori prediction error can be obtained directly. If the $N+1$ dimension vector gain is partitioned,

$$G_1'(n+1) = \begin{bmatrix} M'(n+1) \\ m'(n+1) \end{bmatrix} \tag{6.107}$$

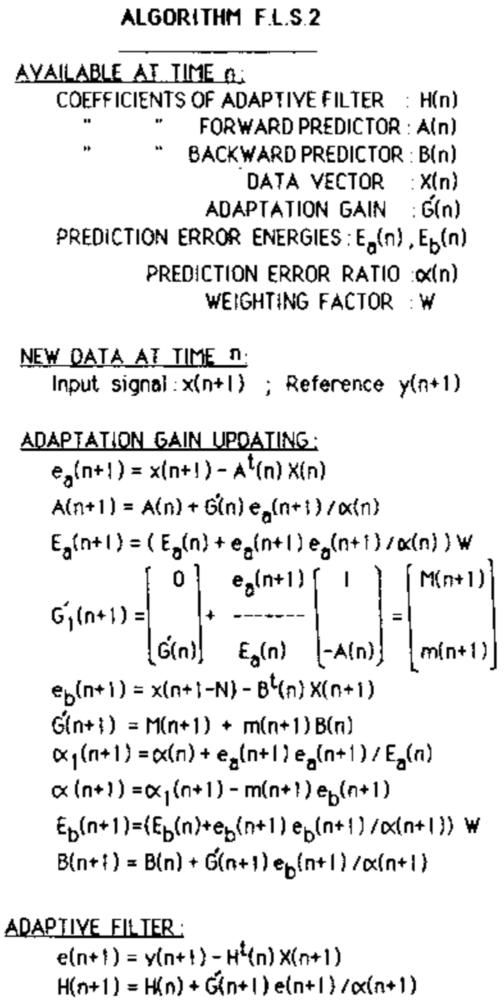
the last line of matrix equation (6.96) is

$$m'(n+1) = \frac{e_b(n+1)}{E_b(n)} \tag{6.108}$$

which provides $e_b(n+1)$ through just a single multiplication. However, due to roundoff problems discussed in a later section, this simplification is not recommended. The overall algorithm is given in Figure 6.5.

The LS initialization corresponds to

$$A(0) = B(0) = G'(0) = 0, \quad E_a(0) = E_0, \quad E_b(0) = W^{-N} E_0 \tag{6.109}$$



ALGORITHM F.L.S.2

AVAILABLE AT TIME n:

COEFFICIENTS OF ADAPTIVE FILTER	: $H(n)$
" " FORWARD PREDICTOR	: $A(n)$
" " BACKWARD PREDICTOR	: $B(n)$
DATA VECTOR	: $X(n)$
ADAPTATION GAIN	: $G'(n)$
PREDICTION ERROR ENERGIES	: $E_a(n)$, $E_b(n)$
PREDICTION ERROR RATIO	: $\alpha(n)$
WEIGHTING FACTOR	: W

NEW DATA AT TIME n:

Input signal : $x(n+1)$; Reference $y(n+1)$

ADAPTATION GAIN UPDATING:

$$e_a(n+1) = x(n+1) - A^t(n)X(n)$$
$$A(n+1) = A(n) + G'(n)e_a(n+1)/\alpha(n)$$
$$E_a(n+1) = (E_a(n) + e_a(n+1)e_a(n+1)/\alpha(n))W$$
$$G_1'(n+1) = \begin{bmatrix} 0 \\ \\ G'(n) \end{bmatrix} + \frac{e_a(n+1)}{E_a(n)} \begin{bmatrix} 1 \\ \\ -A(n) \end{bmatrix} = \begin{bmatrix} M(n+1) \\ \\ m(n+1) \end{bmatrix}$$
$$e_b(n+1) = x(n+1-N) - B^t(n)X(n+1)$$
$$G'(n+1) = M(n+1) + m(n+1)B(n)$$
$$\alpha_1(n+1) = \alpha(n) + e_a(n+1)e_a(n+1)/E_a(n)$$
$$\alpha(n+1) = \alpha_1(n+1) - m(n+1)e_b(n+1)$$
$$E_b(n+1) = (E_b(n) + e_b(n+1)e_b(n+1)/\alpha(n+1))W$$
$$B(n+1) = B(n) + G'(n+1)e_b(n+1)/\alpha(n+1)$$

ADAPTIVE FILTER:

$$e(n+1) = y(n+1) - H^t(n)X(n+1)$$
$$H(n+1) = H(n) + G'(n+1)e(n+1)/\alpha(n+1)$$

FIG. 6.5 Computational organization of the fast algorithm based on all prediction errors.

where E_0 is a small positive constant. Definition (6.101) also yields $\alpha(0) = W$.

The adaptation gain updating section requires $6N + 9$ multiplications and three divisions in the form of inverse calculations. The filtering section has $2N + 1$ multiplications. Approximately $6N + 7$ memories are needed. Overall this second algorithm can bring an appreciable improvement in computational complexity over the first one, particularly for large order N.

6.6. STABILITY CONDITIONS FOR LS RECURSIVE ALGORITHMS

For a nonzero set of signal samples, the LS calculations provide a unique set of prediction coefficients. Recursive algorithms correspond to exact calculations at any time, and, therefore, their stability is guaranteed in theory for any weighting factor W. Since fast algorithms are mathematically equivalent to RLS, they enjoy the same property. Their stability is even guaranteed for a zero signal sequence, provided the initial prediction error energies are greater than zero. This is a very important and attractive theoretical property, which, unfortunately, is lost in realizations because of finite precision effects in implementations [7–10].

Fast algorithms draw their efficiency from a representation of LS parameters, the inverse input signal AC matrix, and cross-correlation estimations, which is reduced to a minimal number of variables. With the finite accuracy of arithmetic operations, that representation can only be approximate. So, the inverse AC matrix estimation $R_N^{-1}(n)$ appears in FLS algorithms through its product by the data vector $X(n)$, which is the adaptation gain $G(n)$. Since the data vector is by definition an exact quantity, the roundoff errors generated in the gain calculation procedure correspond to deviations of the actual inverse AC matrix estimation from its theoretical infinite accuracy value.

In Section 3.11, we showed that random errors on the AC matrix elements do not significantly affect the eigenvalues, but they alter the eigenvector directions. Conversely, a bias in estimating the ACF causes variations of eigenvalues.

When the data vector $X(n)$ is multiplied by the theoretical matrix $R_N^{-1}(n)$, the resulting vector has a limited range because $X(n)$ belongs to the signal space of the matrix.

However, if an approximation of $R_N^{-1}(n)$ is used, the data vector can have a significant projection outside of the matrix signal space; in that case, the norm of the resulting vector is no longer controlled, which can make variables exceed the limits of their magnitude range. Also, the eigenvalues can become negative because of long-term roundoff error accumulation.

Several variables have a limited range in FLS algorithms. A major step in the sequence of operations is the computation of a posteriori errors, from coefficients which have been updated with the adaptation gain and a priori errors. Therefore the accuracy of the representation of $R_N^{-1}(n)X(n)$ by $G(n)$ can be directly controlled by the ratio $\varphi(n)$ of a posteriori to a priori prediction errors. In realizations the variable $\varphi(n)$, introduced in Section 6.3, corresponds to

$$\varphi(n) = 1 - X^t(n)[R_N^q(n)]^{-1}X(n) \tag{6.110}$$

where $R_N^q(n)$ is the matrix used instead of the theoretical $R_N(n)$. The variable $\varphi(n)$ can exceed unity if eigenvalues of $R_N^q(n)$ become negative; $\varphi(n)$ can become negative if the scalar $X^t(n)[R_N^q(n)]^{-1}X(n)$ exceeds unity.

Roundoff error accumulation, if present, takes place in the long run. The first precaution in implementing fast algorithms is to make sure that the scalar $X^t(n)[R_N^q(n)]^{-1}X(n)$ does not exceed unity.

To begin with, let us assume that the input signal is a white zero mean Gaussian noise with power σ_x^2. As seen in Section 3.11, for sufficiently large n one has

$$R_N(n) \approx \frac{\sigma_x^2}{1-W} I_N \tag{6.111}$$

Near the time origin, the actual matrix $R_N^q(n)$ is assumed to differ from $R_N(n)$ only by addition of random errors, which introduces a decoupling between $R_N^q(n)$ and $X(n)$. Hence the following approximation can be justified:

$$X^t(n)[R_N^q(n)]^{-1}X(n) \approx \frac{1-W}{\sigma_x^2} X^t(n)X(n) \tag{6.112}$$

The variable $X^t(n)X(n)$ is Gaussian with mean $N\sigma_x^2$ and variance $2N\sigma_x^4$. If a peak factor of 4 is assumed, a condition for keeping the prediction error ratio above zero is

$$(1-W)(N + 4\sqrt{2N}) < 1 \tag{6.113}$$

This inequality shows that a lower bound is imposed on W. For example, if $N = 10$, then $W > 0.95$.

Now, for a more general input signal, the extreme situation occurs when the data vector $X(n)$ has the direction of the eigenvector associated with the smallest eigenvalue $\lambda_{\min}^q(n)$ of $R_N^q(n)$. Under the hypotheses of zero mean random error addition, neglecting long-term accumulation processes if any, the following approximation can be made:

$$\lambda_{\min}^{q}(n) \approx \frac{\lambda_{\min}}{1-W} \tag{6.114}$$

where $\lambda_{\min}$ is the smallest eigenvalue of the input signal AC matrix. If we further approximate $X^t(n)X(n)$ by $N\sigma_x^2$, the condition on $\varphi(n)$ becomes:

$$(1-W)\frac{N\sigma_x^2}{\lambda_{\min}} < 1 \tag{6.115}$$

This condition may appear extremely restrictive, since the ratio $\sigma_x^2/\lambda_{\min}$ can take on large values. For example, if $x(n)$ is a determinist signal with additive noise and the predictor order N is large enough, $\sigma_x^2/\lambda_{\min}$ is the SNR. Inequalities (6.13) and (6.115) have been derived under restrictive hypotheses on the effects of roundoff errors, and they must be used with care. Nevertheless, they show that the weighting factor W cannot be chosen arbitrarily small.

6.7. INITIAL VALUES OF THE PREDICTION ERROR ENERGIES

The recursive implementations of the weighted LS algorithms require the initialization of the state variables. If the signal is not known before time $n = 0$, it is reasonable to asume that it is zero and the prediction coefficients are zero. However, the forward prediction error energy must be set to a positive value, say E_0. For the algorithm to start on the right track, the initial conditions must correspond to a LS situation.

A positive forward prediction error energy, when the prediction coefficients are zero, can be interpreted as corresponding to a signal whose previous samples are all zero except for one. Moreover, if the gain $G(0)$ is also zero, then the input sequence is

$$\begin{aligned} x(-N) &= (W^{-N}E_0)^{1/2} \\ x(n) &= 0, \quad n \leqslant 0, n \neq -N \end{aligned} \tag{6.116}$$

The corresponding value for the backward prediction error energy is $E_b(0) = x^2(-N) = W^{-N}E_0$—hence the initialization (6.109).

In these conditions the initial value of the AC matrix estimation is

$$R_N(0) = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & W^{-1} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & W^{-(N-1)} \end{bmatrix} E_0 \tag{6.117}$$

and the matrix actually used to estimate the input AC matrix is $R_N^*(n)$, given by

$$R_N^*(n) = R_N(n) + W^n R_N(0) \tag{6.118}$$

The smallest eigenvalue of the expectation of $R_N^*(n)$, denoted $\lambda_{\min}^*(n)$, is obtained, using (6.22), by

$$\lambda_{\min}^*(n) = \frac{1 - W^n}{1 - W}\lambda_{\min} + W^n E_0 \tag{6.119}$$

The first term on the right side is growing with n while the second is decaying. The transient phase and the steady state are put in the same situation as concerns stability if a lower bound is set on E_0. Equation (6.119) can be rewritten as

$$\lambda_{\min}^*(n) = \frac{\lambda_{\min}}{1 - W} + W^n\left(E_0 - \frac{\lambda_{\min}}{1 - W}\right) \tag{6.120}$$

Now, $\lambda_{\min}^*(n)$ is at least equal to $\lambda_{\min}/1 - W$ if E_0 itself is greater or equal to that quantity. From condition (6.115), we obtain

$$E_0 \geqslant N\sigma_x^2 \tag{5.121}$$

This condition has been derived under extremely restrictive hypotheses; it is, in general, overly pessimistic, and smaller values of the initial prediction error energy can work in practice. The representation of the matrix $R_N(n)$ in the system can stay accurate during a period of time longer than the transient phase as soon as the machine word length is sufficiently large. For example, extensive experiments carried out with a 16-bit microprocessor and fixed-point arithmetic have shown that a lower bound for E_0 is about $0.01\sigma_x^2$ [11]. If the word length is smaller, then E_0 must be larger. As an illustration, a unit power AR signal is fed to a predictor with order $N = 4$, and the quadratic deviation of the coefficients from their ideal values is given in Figure 6.6 for several values of E_0. The weighting factor is $W = 0.99$ and a word length of 12 bits in fixed-point arithmetic is simulated. Satisfactory operation of the algorithm is obtained for $E_0 \geqslant 0.1$.

Finally, the above derivations show that the initial error energies cannot be taken arbitrarily small.

6.8. BOUNDING PREDICTION ERROR ENERGIES

The robustness of LS algorithms to roundoff errors can be improved by adding a noise sequence to the input signal. The smallest eigenvalue of the input AC matrix is increased by the additive noise power with that method, which can help satisfy inequality (6.115). However, as mentioned in Chapter 5, a bias is introduced on the prediction coefficients, and it is more desirable to use an approach bearing only on the algorithm operations.

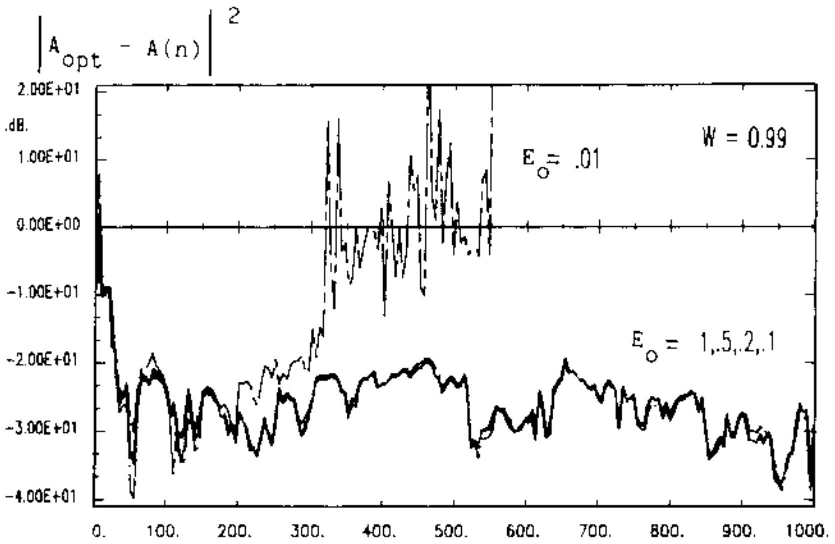


FIG. 6.6 Coefficient deviations for several initial error energy values.

When one considers condition (6.115), one can observe that, for W and N fixed, the only factor which can be manipulated is $\lambda_{\min}$, the minimal eigenvalue of the $N \times N$ input signal AC matrix. That factor is not available in the algorithm. However, it can be related to the prediction error energies, which are available.

From a different point of view, if the input signal is predictable, as seen in Section 2.9, the steady-state prediction error is zero for an order N sufficiently large. Consequently, the variables $E_a(n)$ and $E_b(n)$ can become arbitrarily small, and the rounding process eventually sets them to zero, which is unacceptable since they are used as divisors. Therefore a lower bound has to be imposed on error energies when the FLS algorithm is implemented in finite precision hardware. A simple method is to introduce a positive constant C in the updating equation

$$E_a(n+1) = WE_a(n) + e_a(n+1)\varepsilon_a(n+1) + C \tag{6.122}$$

If σ_e^2 denotes the prediction error power associated with a stationary input signal, the expectation of $E_a(n)$ in the steady state is

$$E[E_a(n)] = \frac{\sigma_e^2 + C}{1 - W} \tag{6.123}$$

The same value would have been obtained with the weighting factor W' satisfying

$$\frac{\sigma_e^2 + C}{1 - W} = \frac{\sigma_e^2}{1 - W'} \tag{6.124}$$

and a first global assessment of the effect of introducing the constant C is that it increases the weighting factor from W to W', which helps satisfy condition (6.115).

As concerns the selection of a value for C, it can be related to the initial error energy E_0 and a reasonable choice can be:

$$C = (1 - W)E_0 \tag{6.125}$$

In fact both E_0 and C depend on the performance objectives and the information available on the input signal characteristics.

A side effect of introducing the constant C is that it produces a leakage in the updating of the backward prediction coefficient vector, which can contribute to counter roundoff error accumulation.

Adding a small constant C to $E_a(n+1)$ leads to the adaptation gain

$$G_1^*(n+1) \approx G_1(n+1) - \frac{\varepsilon_a(n+1)}{E_a^2(n+1)} \begin{bmatrix} 1 \\ -A(n+1) \end{bmatrix} C \tag{6.126}$$

The last element is

$$m^*(n+1) \approx m(n+1) + \frac{\varepsilon_a(n+1)}{E_a^2(n+1)} a_N(n+1)C \tag{6.127}$$

and the backward prediction updating equation in these conditions takes the form

$$B(n+1) \approx (1 - \gamma_b)B(n) + G(n+1)e_b(n+1) \tag{6.128}$$

with

$$E[\gamma_b] \approx C(1 - W)E[a_N^2(n+1)]/E(E_a(n+1)] \tag{6.129}$$

However, it must be pointed out that, with the constant C, the algorithm is no longer in conformity with the LS theory and the theoretical stability is not guaranteed for any signals. The detailed analysis further reveals that the constant C increases the prediction error ratio $\varphi(n)$. Due to the range limitations for $\varphi(n)$ that can lead to the algorithm divergence for some signals. For example, with sinusoids as input signals, it can be seen, using the results given in Section 3.7 of Chapter 3, that $\varphi(n)$ can take on values very close to unity for sinusoids with close frequencies. In those cases the value of the constant C has to be very small and, consequently, a large machine word length is needed.

The roundoff error accumulation process is investigated next.

6.9. ROUNDOFF ERROR ACCUMULATION AND ITS CONTROL

Roundoff errors are generated by the quantization operations which generally take place after the multiplications and divisions. They are thought to come from independent sources, their spectrum is assumed flat, and their variance is $q^2/12$, where q is the quantization step size related to the internal word length of the machine used. The particularity of the FLS algorithms, presented in the previous sections, is that accumulation can take place [6–9]. Basically, the algorithm given in Figure 6.4, for example, consists of three overlapping recursions. The adaptation gain updating recursion makes the connection between forward and backward prediction coefficient recursions, and these recursions can produce roundoff noise accumulation [12].

Let us assume, for example, that an error vector $\Delta B(n)$ is added to the backward prediction coefficient vector $B(n)$ at time n. Then if we neglect the scalar term in (6.87) and consider the algorithm in Figure 6.4, the deviation at time $n+1$ is

$$\begin{aligned}\Delta B(n+1) = {} & [I_N[1+m(n+1)e_b(n+1)] - G(n+1)X^t(n+1)]\Delta B(n)\\ & - \Delta B(n)\Delta B^t(n)m(n+1)X(n+1)\end{aligned} \tag{6.130}$$

If $\Delta B(n)$ is a random vector with zero mean, which is the case for a rounding operation, the mean of $\Delta B(n+1)$ is not zero because of the matrix $\Delta B(n)\Delta B^t(n)$ in (6.130) and because $m(n+1)$ is related to the input signal, the expectation of the product $m(n+1)X(n+1)$ is, in general, not zero. The factor of $\Delta B(n)$ is close to a unity matrix—it can even have eigenvalues greater than 1—thus the introduction of error vectors $\Delta B(n)$ at each time n produces a drift in the coefficients. The effect is a shift of the coefficients from their optimal values, which degrades performance. However, if the minimum eigenvalue $\lambda_{1\,\min}$ of the $(N+1)\times(N+1)$ input AC matrix is close to the signal power σ_x^2, the prediction error power, also close to σ_x^2 because of (5.6), is an almost flat function of the coefficients and the drift can continue to the point where the resulting deviation of the eigenvalues and eigenvectors of the represented matrix $R_N^q(n)$ makes $\varphi(n)$ exceed its limits (6.41). Then, the algorithm is out of the LS situation and generally becomes unstable.

It is important to note that long-term roundoff error accumulation affects the backward prediction coefficients but, except for the case $N=1$, has much less effect on the forward coefficients. This is mainly due to the shift in the elements of the gain vector, which is performed by equation (6.81).

An efficient technique to counter roundoff error accumulation consists of finding a representative control variable and using it to prevent the coefficient drift [13].

Since we have observed that roundoff error accumulation occurs in the backward prediction section of the algorithms, it seems desirable to find an alternative way to compute the backward linear prediction error $e_b(n+1)$.

Combining equations (6.85) and (6.56) yields

$$e_b(n+1) = m(n+1)E_b(n+1)/\varphi(n+1) \tag{6.131}$$

Now, considering the forward linear prediction matrix equation and computing the first row, the equivalent of equation (5.72) is obtained:

$$1 = \frac{\det R_N(n)}{\det R_{N+1}(n+1)} E_a(n+1) \tag{6.132}$$

The same procedure can be applied to backward linear prediction, to yield

$$1 = \frac{\det R_N(n+1)}{\det R_{N+1}(n+1)} E_b(n+1) \tag{6.133}$$

Combining the two above expressions with (6.40) we get

$$\varphi(n+1) = W^N E_b(n+1)/E_a(n+1) \tag{6.134}$$

and finally

$$e_b(n+1) = m(n+1)W^{-N} E_a(n+1) \tag{6.135}$$

Thus, the backward linear prediction error can be computed from variables updated in the forward prediction section of the algorithm, and the variable

$$\xi(n+1) = e_b(n+1) - m(n+1)W^{-N} E_a(n+1) \tag{6.136}$$

can be considered representative of the roundoff error accumulation in algorithm FLS1. It can be minimized by a recursive least squares procedure applied to the backward linear prediction coefficient vector and using adaptation gain $G(n+1)$. In fact, to control the roundoff error accumulation, it is sufficient to update the backward prediction coefficient vector as follows:

$$B(n+1) = B(n) + G(n+1)[e_b(n+1) + \xi(n+1)] \tag{6.137}$$

As concerns algorithm FLS2, a similar procedure can be employed, based on the variable $m'(n+1)$ defined by equation (6.108). The correction variable is

$$\xi'(n+1) = [x(n+1-N) - m'(n+1)E_b(n+1)] - B^t(n)X(n+1) \tag{6.138}$$

and the roundoff error control can be implemented with no additional multiplication if, in Figure 6.5, the backward coefficient updating recursion is replaced by

$$B(n+1) = B(n) + \frac{G'(n+1)[e_b(n+1) + e_b(n+1) - m'(n+1)E_b(n)]}{\alpha(n+1)} \tag{6.139}$$

The FORTRAN program of the corresponding algorithm, including roundoff error accumulation control in the simplest version, is given in Annex 6.2.

It must be pointed out that there is no formal proof that the approaches presented in this section avoid all possible roundoff error accumulation; and, in fact, more sophisticated correction techniques can be devised. However, the above techniques are simple and have been shown to perform satisfactorily under a number of circumstances.

An alternative way of escaping roundoff error accumulation is to avoid using backward prediction coefficients altogether.

6.10. A SIMPLIFIED ALGORITHM

When the input signal is stationary, the steady-state backward prediction coefficients are equal to the forward coefficients, as shown in Chapter 5, and the following equalities hold:

$$B(n) = J_N A(n), \quad E_a(n) = E_b(n) \tag{6.140}$$

This suggests replacing backward coefficients by forward coefficients in FLS algorithms. However, the property of theoretical stability of the LS principle is lost. Therefore it is necessary to have means to detect out-of-range values of LS variables. The variable $\alpha(n) = W/\varphi(n)$ can be used in combination with the gain vector $G'(n)$. The simplified algorithm obtained is given in Figure 6.7. It requires $7N+5$ multiplications and two divisions (inverse calculations). The stability in the initial phase, starting from the idle state, can be critical. Therefore, the magnitude of $\alpha(n)$ is monitored, and if it falls below W the system is reinitialized.

In some cases, particularly with AR input signals when the prediction order exceeds the model order, the simplified algorithm turns out to provide faster convergence than the standard FLS algorithms with the same para-

```
ALGORITHM S.F.L.S.

AVAILABLE AT TIME n:
    COEFFICIENTS OF ADAPTIVE FILTER : $H(n)$
        "        "   FORWARD PREDICTOR : $A(n)$
                        DATA VECTOR : $X(n)$
                    ADAPTATION GAIN : $G'(n)$
            PREDICTION ERROR ENERGY   $E_a(n)$
             PREDICTION ERROR RATIO   $\alpha(n)$
                   WEIGHTING FACTOR   $W$

NEW DATA AT TIME n:
    Input signal : $x(n+1)$ ; Reference $y(n+1)$

ADAPTATION GAIN UPDATING:
    $e_a(n+1) = x(n+1) - A^t(n)\,X(n)$
    $A(n+1) = A(n) + G'(n)\,e_a(n+1)/\alpha(n)$
    $E_a(n+1) = (E_a(n) + e_a(n+1)\,e_a(n+1)/\alpha(n))\,W$
    $G'_1(n+1) = \begin{bmatrix} 0 \\ G'(n) \end{bmatrix} + \frac{e_a(n+1)}{E_a(n)} \begin{bmatrix} 1 \\ -A(n) \end{bmatrix} = \begin{bmatrix} M(n+1) \\ m(n+1) \end{bmatrix}$
    $G'(n+1) = M(n+1) + m(n+1)\,J_N\,A(n)$
    $\alpha(n+1) = W + X^t(n+1)\,G'(n+1)$

ADAPTIVE FILTER:
    $e(n+1) = y(n+1) - H^t(n)\,X(n+1)$
    $H(n+1) = H(n) + G'(n+1)\,e(n+1)/\alpha(n+1)$
```

FIG. 6.7 Computational organization of a simplified LS-type algorithm.

meters because the backward coefficients start with a value which is not zero but may be close to the final one.

6.11. PERFORMANCE OF LS ADAPTIVE FILTERS

The main specifications for adaptive filters concern, as in Section 4.2, the time constant and the system gain. Before investigating the initial transient phase, let us consider the filter operation after the first data have become available.

The set of output errors from time 1 to n constitute the vector

$$\begin{bmatrix} e(1) \\ e(2) \\ e(3) \\ \vdots \\ e(N) \\ \vdots \\ e(n) \end{bmatrix} = \begin{bmatrix} y(1) \\ y(2) \\ y(3) \\ \vdots \\ y(N) \\ \vdots \\ y(n) \end{bmatrix} - \begin{bmatrix} x(1) & 0 & 0 & \cdots & 0 \\ x(2) & x(1) & 0 & \cdots & 0 \\ x(3) & x(2) & x(1) & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ x(N) & x(N-1) & & \cdots & x(1) \\ \vdots & \vdots & & & \vdots \\ x(n) & x(n-1) & & \cdots & x(n+1-N) \end{bmatrix}$$

$$\times \begin{bmatrix} h_0(n) \\ h_1(n) \\ \vdots \\ h_{N-1}(n) \end{bmatrix} \tag{6.141}$$

Recall that the coefficients at time n are calculated to minimize the sum of the squares of the output errors. Clearly, for $n = 1$ the solution is

$$h_0(1) = \frac{y(1)}{x(1)}, \quad h_i(1) = 0, \quad 2 \leqslant i \leqslant N-1 \tag{6.142}$$

For $n = 2$,

$$\begin{aligned} h_0(2) &= \frac{y(1)}{x(1)} \\ h_1(2) &= \frac{y(2)}{x(1)} - h_0(2)\frac{x(2)}{x(1)} \\ h_i(2) &= 0, \quad 3 \leqslant i \leqslant N-1 \end{aligned} \tag{6.143}$$

The output of the adaptive LS filter is zero from time 1 to N, and the coefficients correspond to an exact solution of the minimization problem. In fact, the system of equations becomes overdetermined, and the LS procedure starts only at time $N+1$.

In order to get simple expressions for the transient phase, we first analyze the system identification, shown in Figure 6.8. The reference signal is

$$y(n) = X^t(n)H_{\text{opt}} + b(n) \tag{6.144}$$

where $b(n)$ is a zero mean white observation noise with power $E_{\min}$, uncorrelated with the input $x(n)$. H_{opt} is the vector of coefficients which the adaptive filter has to find.

The coefficient vector of the LS adaptive filter at time n is

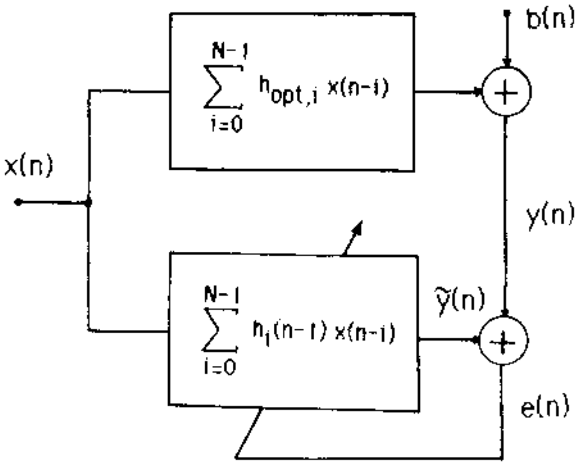

b(n)
$\sum_{i=0}^{N-1} h_{opt,i}\, x(n-i)$
x(n)
y(n)
$\tilde{y}(n)$
$\sum_{i=0}^{N-1} h_i(n-1)\, x(n-i)$
e(n)

FIG. 6.8 Adaptive system identification.

$$H(n) = R_N^{-1}(n) \sum_{p=1}^{n} W^{n-p}[X(p)X^t(p)H_{\text{opt}} + X(p)b(p)] \tag{6.145}$$

or, in concise form,

$$H(n) = H_{\text{opt}} + R_N^{-1}(n) \sum_{p=1}^{n} W^{n-p} X(p)b(p) \tag{6.146}$$

Denoting by $\Delta H(n)$ the coefficient deviation

$$\Delta H(n) = H(n) - H_{\text{opt}} \tag{6.147}$$

and assuming that, for a given sequence $x(p)$, $b(p)$ is the only random variable, we obtain the covariance matrix

$$E[\Delta H(n)\Delta H^t(n)] = E_{\min} R_N^{-1}(n) \left[\sum_{p=1}^{n} W^{2(n-p)} X(p)X^t(p) \right] R_N^{-1}(n) \tag{6.148}$$

For $W = 1$,

$$E[\Delta H(n)\Delta H^t(n)] = E_{\min} R_N^{-1}(n) \tag{6.149}$$

At the output of the adaptive filter the error signal at time n is

$$e(n) = y(n) - X^t(n)H(n-1) = b(n) - X^t(n)\Delta H(n-1) \tag{6.150}$$

The variance is

$$E[e^2(n)] = E_{\min} + X^t(n)E[\Delta H(n-1)\Delta H^t(n-1)]X(n) \tag{6.151}$$

and, for $W = 1$,

$$E[e^2(n)] = E_{\min}[1 + X^t(n)R_N^{-1}(n-1)X(n)] \tag{6.152}$$

Now, the mean residual error power $E_R(n)$ is obtained by averaging over all input signal sequences. If the signal $x(n)$ is a realization of a stationary process with AC matrix R_{xx}, for large n one has

$$R_N(n) \approx nR_{xx} \tag{6.153}$$

Using the matrix equality

$$X^t(n)R_N^{-1}(n)X(n) = \text{trace}[R_N^{-1}(n)X(n)X^t(n)] \tag{6.154}$$

and (6.153), we have

$$E_R(n) = E_{\min}\left(1 + \frac{N}{n-1}\right) \tag{6.155}$$

If the first datum received is $x(1)$, then, since the LS process starts at time $N+1$, the mean residual error power at time n is:

$$E_R(n) = E_{\min}\left(1 + \frac{N}{n-N}\right), \quad n \geqslant N+1 \tag{6.156}$$

Thus, at time $n = 2N$, the mean residual error power is twice, or 3 dB above, the minimal value. This result can be compared with that obained for the LMS algorithm, which, for an input signal close to being a white noise and a step size corresponding to the fastest start, is

$$E(n) - E_{\min} = [E(0) - E_{\min}]\left(1 - \frac{1}{N}\right)^{2n} \tag{6.157}$$

which was derived by applying results obtained in Section 4.4

The corresponding curves in Figure 6.9 show the advantage of the theoretical LS approach over the gradient technique when the system starts from the idle state [14].

Now, when a weighting factor is used, the error variance has to be computed from (6.148). If the matrix $R_N(n)$ is approximated by its expectation as in (6.153), one has

$$\begin{aligned} R_N(n) &\approx \frac{1-W^n}{1-W}R_{xx} \\ \sum_{p=1}^{n} W^{2(n-p)}X(p)X^t(p) &\approx \frac{1-W^{2n}}{1-W^2}R_{xx} \end{aligned} \tag{6.158}$$

which, using identity (6.154) again, gives

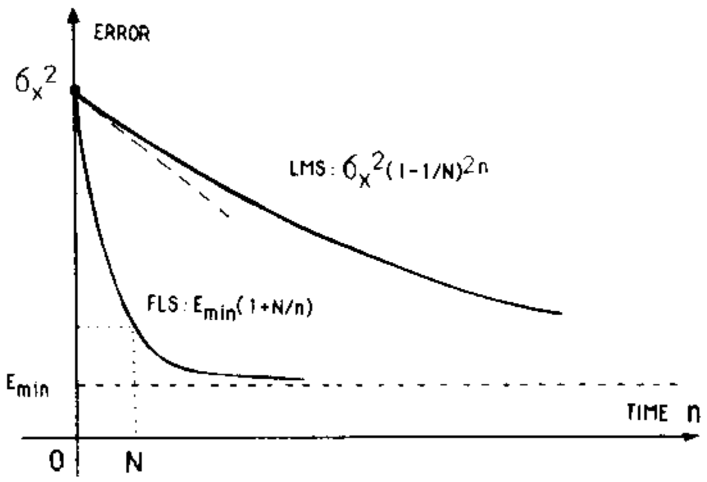


FIG. 6.9 Learning curves for LS and LMS algorithms.

$$E_R(n) \approx E_{\min}\left[1 + N\frac{1-W}{1+W}\frac{1+W^n}{1-W^n}\right] \tag{6.159}$$

For $n \to \infty$,

$$E_R(\infty) = E_{\min}\left[1 + N\frac{1-W}{1+W}\right] \tag{6.160}$$

This expression can be compared to the corresponding relation (4.35) in Chapter 4 for the gradient algorithm. The weighting factor introduces an excess MSE proportional to $1 - W$.

The coefficient learning curve is derived from recursion (6.23), which yields

$$\begin{aligned}\Delta H(n+1) = {} & [I_N - R_N^{-1}(n+1)X(n+1)X^t(n+1)]\Delta H(n) \\ & + R_N^{-1}(n+1)X(n+1)b(n+1)\end{aligned} \tag{6.161}$$

Assuming that $\Delta H(n)$ is independent of the input signal, which is true for large n, and using approximation (6.158), one gets

$$E[\Delta H(n+1)] = \left(1 - \frac{1-W}{1-W^n}\right)E[\Delta H(n)] \tag{6.162}$$

Therefore, the learning curve of the filter of order N is similar to that of the first-order filter analyzed in Section 6.1, and for large n the time constant is $\tau = 1/(1-W)$. It is that long-term time constant which has to be considered when a nonstationary reference signal is applied to the LS adaptive filter. In fact, $1/(1-W)$ can be viewed as the observation time window of the filter, and, as in Section 4.8, its value is chosen to be compatible with the time period over which the signals can be considered as stationary; it is a trade-off between lag error and excess MSE.

6.12. SELECTING FLS PARAMETER VALUES

The performance of adaptive filters based on FLS algorithms differs from that of the theoretical LS filters because of the impact of the additional parameters they require. The value of the initial forward prediction error power E_0 affects the learning curve of the filter.

The matrix $R_N^*(n)$, introduced in Section 6.7, can be expressed by

$$R_N^*(n) = [I_N + W^n R_N(0) R_N^{-1}(n)] R_N(n) \tag{6.163}$$

As soon as n is large enough, we can use (6.25), to obtain its inverse:

$$[R_N^*(n)]^{-1} \approx R_N^{-1}(n)[I_N - W^n R_N(0) R_N^{-1}(n)] \tag{6.164}$$

In these conditions, the deviation $\Delta A(n)$ of the prediction coefficients due to E_0 is

$$\Delta A(n) = W^n R_N^{-1}(n) R_N(0) A(n) \tag{6.165}$$

and the corresponding excess MSE is

$$\Delta E(n) = [\Delta A(n)]^t R_{xx} \Delta A(n) \tag{6.166}$$

Approximating $R_N(n)$ by its expectation and the initial matrix $R_N(0)$ by $E_0 I_N$ gives

$$\Delta E(n) \approx W^{2n} E_0^2 (I-W)^2 A^t(n) R_{xx}^{-1} A(n) \tag{6.167}$$

for W close to 1,

$$\ln[\Delta E(n)] \approx 2\ln[E_0(1-W)] + \ln[A^t(n) R_{xx}^{-1} A(n)] - 2n(1-W) \tag{6.168}$$

For example, the curves $\|\Delta A(n)\|^2$ as a function of n are given in Figure 6.10 for $N=2$, $x(n) = \sin(n\frac{\pi}{4})$, $W = 0.95$, and three different values of the parameter E_0.

The impact of the initial parameter E_0 on the filter performance is clearly apparent from expression (6.168) and the above example. Smaller values of E_0 can be taken if the constant C of Section 6.8 is introduced.

The constant C in (6.122) increases the filter long-term time constant according to (6.124).

The ratio $(1-W')/(1-W)$ is shown in Figure 6.11 as a function of the prediction error σ_e^2. It appears that the starting value $\sigma_x^2/(\sigma_x^2 + C)$ should be

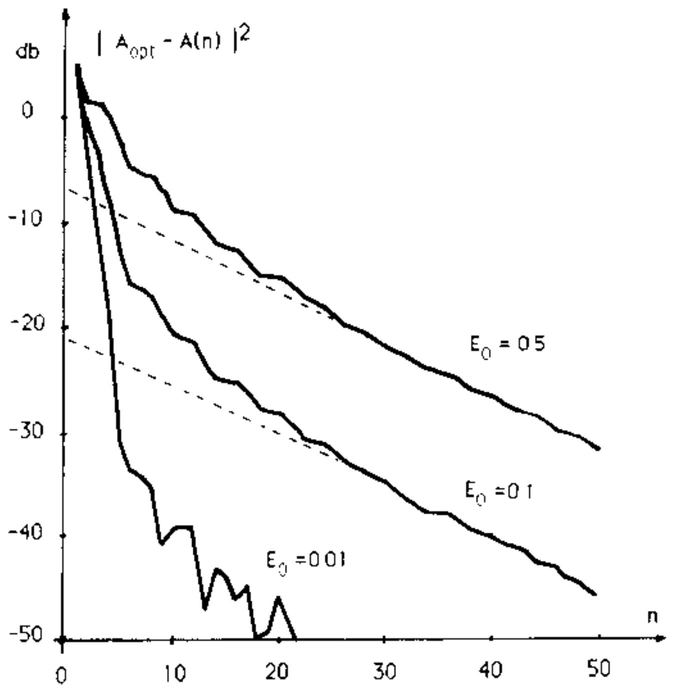


FIG. 6.10 Coefficient deviations for several prediction error energy values with sinusoidal input.

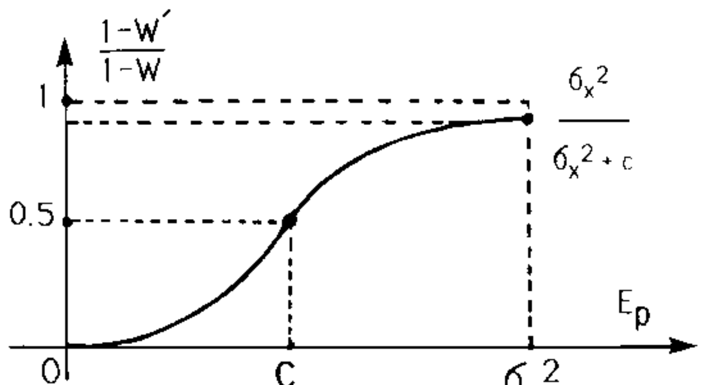


FIG. 6.11 Weighting factor vs. prediction error power with constant *C*.

made as close to unity as possible. So, C should be smaller than the input signal power σ_x^2, which in turn, through (6.115), means that W approaches unity.

If C is significantly smaller than σ_x^2, the algorithm can react quickly to large changes in input signal characteristics, and slowly to small changes. In other words, it has an adjustable time window.

Another effect of C is to modify the excess misadjustment error power, according to equation (6.160), in which W' replaces W.

Nonstationary signals deserve particular attention. The range of values for C depends on E_0 and thus on the signal power. Thus, if the input signal is nonstationary, it can be interesting to use, instead of C, a function of the signal power. For example, the following equation can replace (6.122):

$$E_a(n+1) = E_a(n) + e_a(n+1)\varepsilon_a(n+1) + WN[C_1 + C_2x^2(n+1)] \quad (6.169)$$

where C_1 and C_2 are positive real constants, chosen in accordance with the characteristics of the input signal.

For example, an adequate choice for a speech sentence of unity long-term power has been found to be $C_1 = 1.5$ and $C_2 = 0.5$. The prediction gain obtained is shown in Figure 6.12 for several weighting factor values. As a comparison, the corresponding curve for the normalized LMS algorithm is also shown.

An additional parameter, the coefficient leakage factor, can be useful in FLS algorithms.

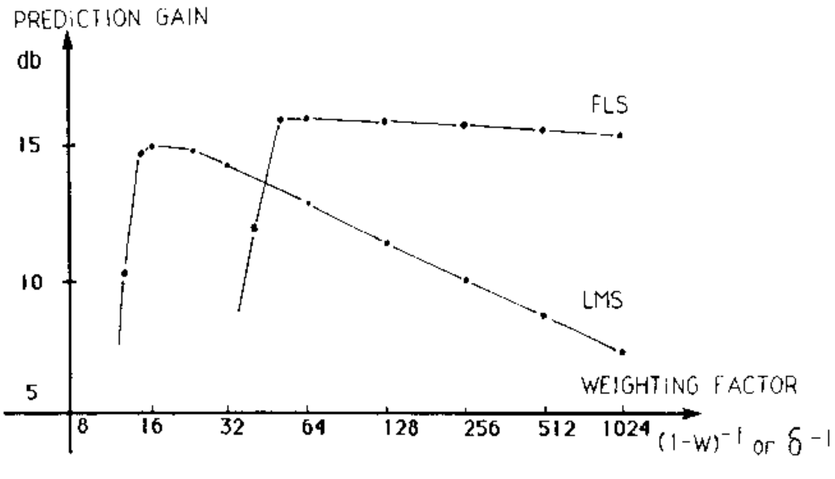


FIG. 6.12 Prediction gain vs. weighting factor or step size for a speech sentence.

From the sequence of operations given in Figures 6.4 and 6.5, it appears that, if the signal $x(n)$ becomes zero, the prediction errors and adaptation gain decay to zero while the coefficients keep a fixed value. The system may be in the initial state considered in the previous sections, when the signal reappears, if a leakage factor is introduced in coefficient updating equations.

Furthermore, such a parameter offers the advantages already mentioned in Section 4.6—namely, it makes the filter more robust to roundoff errors and implementation constraints.

However, the corresponding arithmetic operations have to be introduced with care in FLS algorithms. They have to be performed outside the gain updating loop to preserve the ratio of a posteriori to a priori prediction errors. For example, in Figure 6.4 the two leakage operations

$$\begin{aligned} A(n+1) &= (1-\gamma)A(n+1), \\ B(n+1) &= (1-\gamma)B(n+1), \end{aligned} \qquad 0 < \gamma \ll 1 \tag{6.170}$$

can be placed at the end of the list of equations for the adaptation gain updating. Recall that the leakage factor introduces a bias given by expression (4.69) on the filter coefficients. Note also that, with the leakage factor, the algorithm is no longer complying with the LS theory and theoretical stability cannot be guaranteed for any signals.

6.13. WORD-LENGTH LIMITATIONS AND IMPLEMENTATION

The implementation of transversal FLS adaptive filters can follow the schemes used for gradient filters presented in Chapter 4. The operations in Figure 6.4, for example, correspond roughly to a set of five gradient filters adequately interconnected. However, an important point with FLS is the need for two divisions per iteration, generally implemented as inverse calculations.

The divider $E_a(n)$ is bounded by

$$\min\left\{E_0, \frac{C}{1-W}\right\} \leqslant E_a(n) \leqslant \frac{\sigma_x^2}{1-W} \tag{6.171}$$

and the constant C controls the magnitude range of its inverse. Recall that the other dividers are in the interval [0, 1].

Overall, the estimations of word lengths for FLS filters can be derived using an approach similar to that which is used for LMS filters in Section 4.5. For example, let us consider the prediction coefficients.

In two extreme situations, the FLS algorithm is equivalent to an LMS algorithm with adaptation step sizes:

$$\delta_{\max} = \frac{1}{\lambda_{\min}(n)}, \quad \delta_{\min} = \frac{1}{\lambda_{\max}(n)} \tag{6.172}$$

Now, taking $\lambda_{\max}(n) \approx \lambda_{\max}/(1 - W)$ and recalling that $\lambda_{\max} \leqslant N\sigma_x^2$, we obtain an estimation of the prediction coefficient word length b_c from equation (4.61) in Chapter 4:

$$b_c \approx \log_2\left(\frac{N}{1 - W}\right) + \log_2(G_p) + \log_2(a_{\max}) \tag{6.173}$$

where G_p is the prediction gain and $a_{\max}$ is the magnitude of the largest prediction coefficient, as in Section 4.5. Thus, it can be stated that FLS algorithms require larger word lengths than LMS algorithms, and the difference is about $\log_2 N$.

The implementation is guided by the basic constraint on updating operations, which have to be performed in a sample period. As shown in previous sections, there are different ways of organizing the computations, and that flexibility can be exploited to satisfy given realization conditions. In software, one can be interested in saving on the number of instructions or on the internal data memory capacity. In hardware, it may be important, particularly in high-speed applications using multiprocessor realizations, to rearrange the sequence of operations to introduce delays between internal filter sections and reach some level of pipelining [15]. For example, the algorithm based on a priori errors can be implemented by the following sequence at time $n + 1$:

$$e_a(n+1) \to e_b(n) \to E_a(n) \to G_1(n) \to B(n) \to G(n) \to A(n+1) \to \varepsilon_a(n+1)$$

The corresponding diagram is shown in Figure 6.13 for a prediction coefficient adaptation section. With a single multiplier, the minimum multiply speed is five multiplications per sample period.

6.14. COMPARISON OF FLS AND LMS APPROACHES—SUMMARY

A geometrical illustration of the LS and gradient calculations is given in Figure 6.14. It shows how the inverse input signal AC matrix R_{xx}^{-1} rotates the cost function gradient vector **Grad J** and adjusts its magnitude to reach the optimum coefficient values.

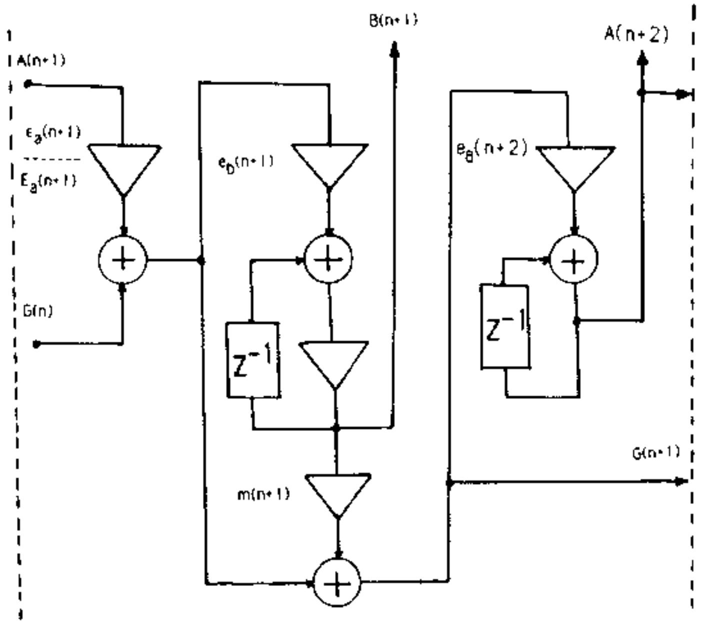


FIG. 6.13 Adaptation section for a prediction coefficient in an FLS algorithm.

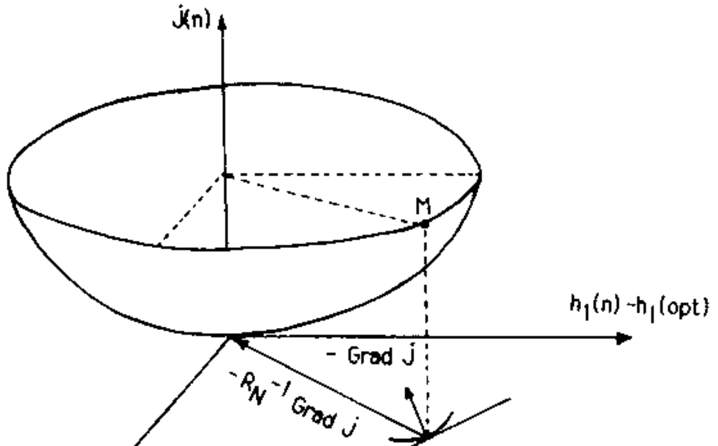


FIG. 6.14 Geometrical illustration of LS and gradient calculations.

In FLS algorithms, real-time estimations of signal statistics are computed and the maximum convergence speed and accuracy can be expected. However, several parameters have to be introduced in realizations, which limit the performance; they are the weighting factor, initial prediction error energies, stabilization constant, and coefficient leakage factor. But if the values of these parameters are properly chosen, the performance can stay reasonably close to the theoretical optimum.

In summary, the advantages of FLS adaptive filters are as follows:

Independence of the spread of the eigenvalues of the input signal AC matrix
Fast start from idle state
High steady-state accuracy

FLS adaptive filters can upgrade the adaptive filter overall performance in various existing applications. However, and perhaps more importantly, they can open up new areas. Consider, for example, spectral analysis, and let us assume that two sinusoids in noise have to be resolved with an order $N = 4$ adaptive predictor. The results obtained with the LMS algorithm are shown in Figure 6.15. Clearly, the prediction coefficient values cannot be used because they indicate the presence of a single sinusoid. Now, the same curves for the FLS algorithm, given in Figure 6.16, allow the correct detection after a few hundred iterations. That simple example shows that FLS algorithms can open new possibilities for adaptive filters in real-time spectral analysis.

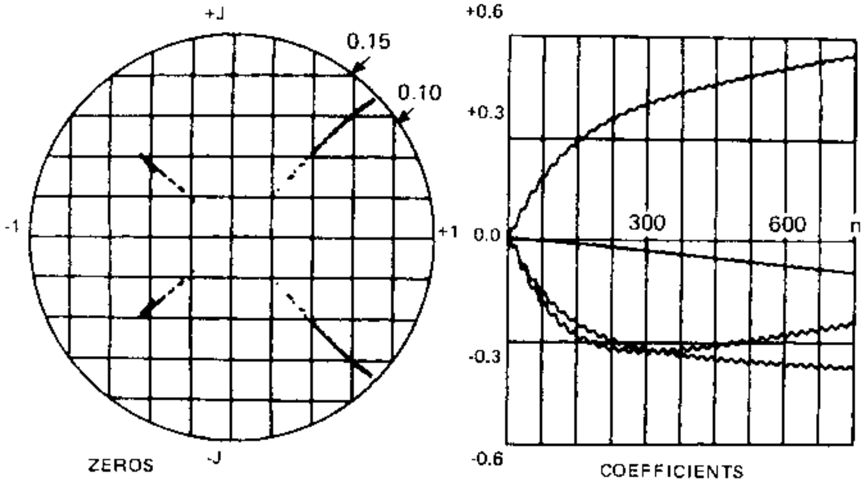


FIG. 6.15 LMS adaptive prediction of two sinusoids with frequencies 0.1 and 0.15.

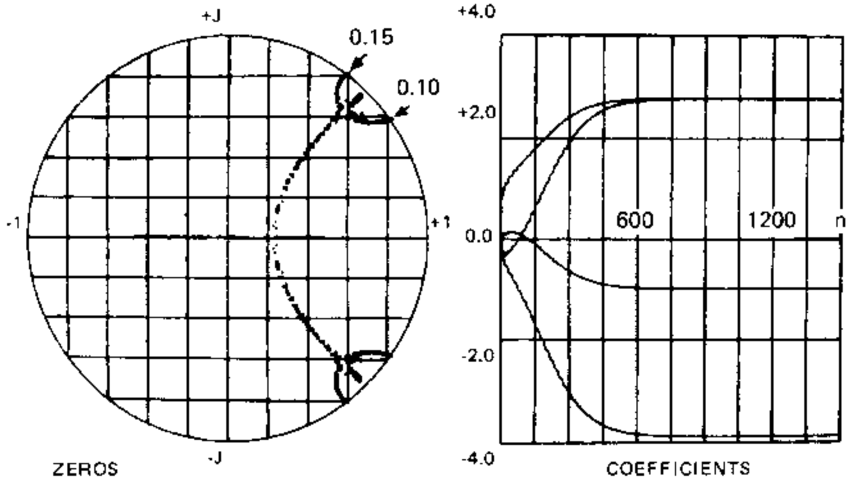


FIG. 6.16 FLS adaptive prediction of two sinusoids.

EXERCISES

1. Verify, through matrix manipulations, the matrix inversion lemma (6.24). Use this lemma to find the inverse M^{-1} of the matrix

$$M = \delta I_N + XX^t$$

where X is an N-element nonzero vector. Give the limit of M^{-1} when $\delta \to 0$. Compare with (6.35).

2. Calculate the matrix $R_2(5)$ for the signal $x(n) = \sin(n\frac{\pi}{4})$ and $W = 0.9$. Compare the results with the signal AC matrix. Calculate the likelihood variable $\theta(5)$. Give bounds for $\theta(n)$ as $n \to \infty$.
3. Use the recurrence relationships for the backward prediction coefficient vector and the correlation vector to demonstrate the backward prediction error energy updating equation (6.67).
4. The signal

$$x(n) = \sin\left(n\frac{\pi}{2}\right), \quad n \geqslant 0$$
$$x(n) = 0, \quad n < 0$$

is fed to an order $N = 4$ FLS adaptive predictor. Assuming initial conditions $A(0) = B(0) = G(0) = 0$, calculate the variables of the algorithm in Figure 6.4 for time $n = 1$ to 5 when $W = 1$ and for initial error energies $E_0 = 0$ and $E_0 = 1$. Compare the coefficient values to optimal values. Comment on the results.

5. In an FLS adaptive filter, the input signal $x(n)$ is set to zero at time N_0 and after. Analyze the evolution of the vectors $A(n)$, $B(n)$, $G(n)$ and the scalars $E_a(n)$ and $\alpha(n)$ for $n \geqslant N_0$.
6. Modify the algorithm of Figure 6.4 to introduce the scalar $\alpha(n)$ with the minimum number of multiplications. Give the computational organization, and count the multiplications, additions, and memories.
7. Study the hardware realization of the algorithm given in Figure 6.5. Find a reordering of the equations which leads to the introduction of sample period delays on the data paths interconnecting separate filter sections. Give the diagram of the coefficient adaptation section. Assuming a single multiplier per coefficient, what is the minimum multiply speed per sample period.

ANNEX 6.1 FLS ALGORITHM BASED ON A PRIORI ERRORS

```
      SUBROUTINE FLS1(N,X,VX,A,B,EA,G,W,IND)
C
C       COMPUTE THE ADAPTATION GAIN (FAST LEAST SQUARES)
C       N   = FILTER ORDER
C       X   = INPUT SIGNAL : x(n+1)
C       VX  = N-ELEMENT DATA VECTOR : X(n)
C       A   = FORWARD PREDICTION COEFFICIENTS
C       B   = BACKWARD PREDICTION COEFFICIENTS
C       EA  = PREDICTION ERROR ENERGY
C       G   = ADAPTATION GAIN
C       W   = WEIGHTING FACTOR
C       IND = TIME INDEX
C
        DIMENSION VX(15),A(15),B(15),G(15),G1(16)
        IF(IND.GT.1)GOTO30
C
C       INITIALIZATION
C
        DO20I=1,15
        A(I)=0.
        B(I)=0.
        G(I)=0.
        VX(I)=0.
   20   CONTINUE
        EA=1.
   30   CONTINUE
C
C       ADAPTATION GAIN CALCULATION
```

```
C
      EAV=X
      EPSA=X
      DO40I=1,N
   40 EAV=EAV-A(I)*VX(I)
      DO50I=1,N
      A(I)=A(I)+G(I)*EAV
      EPSA=EPSA-A(I)*VX(I)
   50 CONTINUE
      EA=W*EA+EAV*EPSA
      G1(1)=EPSA/EA
      DO60I=1,N
   60 G1(I+1)=G(I)-A(I)*G1(1)
      EAB=VX(N)
      DO70I=2,N
      J=N+1-I
   70 VX(J+1)=VX(J)
      VX(1)=X
      DO80I=1,N
   80 EAB=EAB-B(I)*VX(I)
      GG=1.0-EAB*G1(N+1)
      DO90I=1,N
      G(I)=G1(I)+G1(N+1)*B(I)
   90 G(I)=G(I)/GG
      DO100I=1,N
  100 B(I)=B(I)+G(I)*EAB
      RETURN
      END
```

ANNEX 6.2 FLS ALGORITHM BASED ON ALL THE PREDICTION ERRORS AND WITH ROUNDOFF ERROR CONTROL (SIMPLEST VERSION)

```
      SUBROUTINE FLS2(N,X,VX,A,B,EA,EB,GP,ALF,W,IND)
C
C     COMPUTES THE ADAPTATION GAIN (FAST LEAST SQUARES)
C     N   = FILTER ORDER
C     X   = INPUT SIGNAL : x(n+1)
C     VX  = N-ELEMENT DATA VECTOR : X(n)
C     A   = FORWARD PREDICTION COEFFICIENTS
C     B   = BACKWARD PREDICTION COEFFICIENTS
C     EA  = PREDICTION ERROR ENERGY - EB
C     GP  = ''A PRIORI'' ADAPTATION GAIN
```

```
C       ALF = PREDICTION ERROR RATIO
C       W   = WEIGHTING FACTOR
C       IND = TIME INDEX
C
        DIMENSION VX(15),A(15),B(15),G(15),G1(16),GP(15)
        IF(IND.GT.1)GOTO30
C
C       INITIALIZATION
C
        DO20I=1,15
        A(I)=0.
        B(I)=0.
        GP(I)=0.
        VX(I)=0.
   20   CONTINUE
        EA=1.
        EB=1./W**N
        ALF=W
   30   CONTINUE
C
C       ADAPTATION GAIN CALCULATION
C
        EAV=X
        DO40I=1,N
   40   EAV=EAV-A(I)*VX(I)
        EPSA=EAV/ALF
        G1(1)=EAV/EA
        EA=(EA+EAV*EPSA)*W
        DO50I=1,N
   50   G1(I+1)=GP(I)-A(I)*G1(1)
        DO60I=1,N
   60   A(I)=A(I)+GP(I)*EPSA
        EAB1=G1(N+1)*EB
        EAB=VX(N)-B(1)*X
        DO65I=2,N

        EAB=EAB-B(I)*VX(I-1)
   65   CONTINUE
        DO70I=1,N
   70   GP(I)=G1(I)+B(I)*G1(N+1)
        ALF1=ALF+G1(1)*EAV
        ALF=ALF1-G1(N+1)*EAB
        EPSB=(EAB+EAB-EAB1)/ALF
        EB=(EB+EAB*EPSB)*W
        DO80I=1,N
```

```
80   B(I)=B(I)+GP(I)*EPSB
     DO90I=2,N
     J=N+1-I
90   VX(J+1)=VX(J)
     VX(1)=X
     RETURN
     END
```

REFERENCES

1. A. A. Giordano and F. M. Hsu, *Least Squares Estimation With Applications to Digital Signal Processing*, Wiley, New York, 1985.
2. D. J. Tylavsky and G. R. Sohie, ‘‘Generalization of the Matrix Inversion Lemma,’’ *Proc. IEEE* **74**, 1050–1052 (July 1986).
3. J. M. Turner, ‘‘Recursive Least Squares Estimation and Lattice Filters,’’ *Adaptive Filters*, Prentice-Hall, Englewood Cliffs, N.J., 1985, Chap. 5.
4. D. Falconer and L. Ljung, ‘‘Application of Fast Kalman Estimation to Adaptive Equalization,’’ *IEEE Trans.* **COM-26**, 1439–1446 (October 1978).
5. G. Carayannis, D. Manolakis, and N. Kalouptsidis, ‘‘A Fast Sequential Algorithm for LS Filtering and Prediction,’’ *IEEE Trans.* **ASSP-31**, 1394–1402 (December 1983).
6. J. Cioffi and T. Kailath, ‘‘Fast Recursive Least Squares Transversal Filters for Adaptive Filtering,’’ *IEEE Trans.* **ASSP-32**, 304–337 (April 1984).
7. D. Lin, ‘‘On Digital Implementation of the Fast Kalman Algorithms,’’ *IEEE Trans.* **ASSP-32**, 998–1005 (October 1984).
8. S. Ljung and L. Ljung, ‘‘Error Propagation Properties of Recursive Least Squares Adaptation Algorithms,’’ *Automatica* **21**, 157–167 (1985).
9. J. M. Cioffi, ‘‘Limited Precision Effects in Adaptive Filtering,’’ *IEEE Trans.* **CAS-** (1987).
10. S. H. Ardalan and S. T. Alexander, ‘‘Fixed-Point Round-off Error Analysis of the Exponentially Windowed RLS Algorithm,’’ *IEEE Trans.* **ASSP-35**, 770–783 (1987).
11. R. Alcantara, J. Prado, and C. Gueguen, ‘‘Fixed Point Implementation of the Fast Kalman Algorithm Using a TMS 32010 Microprocessor,’’ *Proc. EUSIPCO-86*, North-Holland, The Hague, 1986, pp. 1335–1338.
12. J. L. Botto, ‘‘Stabilization of Fast Recursive Least Squares Transversal Filters,’’ *Proc. IEEE/ICASSP-87*, Dallas, Texas, 1987, pp. 403–406.
13. D. T. M. Slock and T. Kailath, ‘‘Numerically Stable Fast Transversal Filters for Recursive Least-Squares Adaptive Filtering,’’ *IEEE Trans.* **ASSP-39**, 92–114 (1991).
14. M. L. Honig, ‘‘Echo Cancellation of Voice-Band Data Signals Using RLS and Gradient Algorithms,’’ *IEEE Trans.* **COM-33**, 65–73 (January 1985).
15. V. B. Lawrence and S. K. Tewksbury, ‘‘Multiprocessor Implementation of Adaptive Digital Filters,’’ *IEEE Trans.* **COM-31**, 826–835 (June 1983).

7
Other Adaptive Filter Algorithms

The derivation of FLS algorithms for transversal adaptive filters with N coefficients exploits the shifting property of the vector $X(n)$ of the N most recent input data, which is transferred to AC matrix estimations. Therefore, fast algorithms can be worked out whenever the shifting property exists. It means that variations of the basic algorithms can cope with different situations such as nonzero initial state variables and special observation time windows, and also that extensions to complex and multidimensional signals can be obtained.

A large family of algorithms can be constituted and, in this chapter, a selection is presented of those which may be of particular interest in different technical application fields.

If a set of N data $X(1)$ is already available at time $n = 1$, then when the filter is ready to start it may be advantageous to use that information in the algorithm rather than discard it. The so-called covariance algorithm is obtained [1].

7.1. COVARIANCE ALGORITHMS

The essential link in the derivation of the fast algorithms given in the previous chapter is provided by the $(N+1) \times (N+1)$ matrix $R_{N+1}(n+1)$, which relates the adaptation gains $G(n+1)$ and $G(n)$ at two consecutive instants. Here, a slightly different definition of that matrix has to be taken, because the first $(N+1)$-element data vector which is available is $X_1(2)$:

$$[X_1(2)]^t = [x(2), X^t(1)]$$

Thus

$$R_{N+1}(n) = \sum_{p=2}^{n} W^{n-p} X_1(p) X_1^t(p) \tag{7.1}$$

The LS procedure for the prediction filters, because of the definitions, can only start at time $n = 2$, and the correlation vectors are

$$\begin{aligned} r_N^a(n) &= \sum_{p=2}^{n} W^{n-p} x(p) X(p-1) \\ r_N^b(n) &= \sum_{p=2}^{n} W^{n-p} x(p-N) X(p) \end{aligned} \tag{7.2}$$

The matrix $R_{N+1}(n+1)$ can be partitioned in two ways:

$$R_{N+1}(n+1) = \left[\begin{array}{c:c} \sum_{p=2}^{n+1} W^{n+1-p} x^2(p) & [r_N^a(n+1)]^t \\ \hdashline r_N^a(n+1) & R_N(n) \end{array} \right] \tag{7.3}$$

and

$$R_{N+1}(n+1) = \left[\begin{array}{c:c} R_N(n+1) - W^n X(1) X^t(1) & r_N^b(n+1) \\ \hdashline [r_N^b(n+1)]^t & \sum_{p=2}^{n+1} W^{n+1-p} x^2(p-N) \end{array} \right] \tag{7.4}$$

Now the procedure given in Section 6.4 can be applied again. However, several modifications have to be made because of the initial term $W^n X(1) X^t(1)$ in (7.4).

The $(N+1)$-element adaptation gain vector $G_1(n+1)$ can be calculated by equation (6.73) in Chapter 6, which yields $M(n+1)$ and $m(n+1)$. Equation (7.4) leads to

$$[R_N(n+1) - W^n X(1) X^t(1)] M(n+1) + m(n+1) r_N^b(n+1) = X(n+1) \tag{7.5}$$

Similarly the backward prediction matrix equation (6.74) in Chapter 6 combined with partitioning (7.4) leads to

$$[R_N(n+1) - W^n X(1) X^t(1)] B(n+1) = r_N^b(n+1) \tag{7.6}$$

Now the definition of $G(n+1)$ yields

$$G(n+1) = R_N^{-1}(n+1)X(n+1) = [I_N - W^n R_N^{-1}(n+1)X(1)X^t(1)] \times [M(n+1) + m(n+1)B(n+1)] \tag{7.7}$$

The difference with equation (6.86) of Chapter 6 is the initial term, which decays to zero as time elapses. The covariance algorithm, therefore, requires the same computations as the regular FLS algorithm with, in addition, the recursive computation of an initial transient variable. Let us consider the vector

$$D(n+1) = W^n R_N^{-1}(n+1)X(1) \tag{7.8}$$

A recursion is readily obtained by

$$R_N(n+1)D(n+1) = W^n X(1) \tag{7.9}$$

which at time n corresponds to

$$R_N(n)D(n) = W^{n-1}X(1)$$

Taking into account relationship (6.47) in Chapter 6 between $R_N(n)$ and $R_N(n+1)$, one gets

$$D(n+1) = [I_N - R_N^{-1}(n+1)X(n+1)X^t(n+1)]D(n) \tag{7.10}$$

which with (7.7) and some algebraic manipulations yields

$$D(n+1) = \frac{1}{1 - X^t(1)F(n+1)X^t(n+1)D(n)}[I_N - F(n+1)X^t(n+1)]D(n) \tag{7.11}$$

where

$$F(n) = M(n) + m(n)B(n) \tag{7.12}$$

The adaptation gain is obtained by rewriting (7.7) as

$$G(n+1) = [I_N - D(n+1)X^t(1)]F(n+1) \tag{7.13}$$

Finally, the covariance version of the fast algorithm in Section 6.4 is obtained by incorporating equations (7.11) and (7.13) in the sequence of operations. The additional cost in computational complexity amounts to $4N$ multiplications and one division.

Some care has to be exercised in the initialization. If the prediction coefficients are zero, $A(1) = B(1) = 0$, since the initial data vector is nonzero, an initially constrained LS procedure has to be used, which, as mentioned in Section 6.7, corresponds to the following cost function for the filter [1]:

$$J_c(n) = \sum_{p=1}^{n} W^{n-p}[y(p) - X^t(p)H(n)]^2 + E_0 H^t(n)W(n)H(n) \tag{7.14}$$

where

$$W(n) = \operatorname{diag}(W^n, W^{n-1}, \dots, W^{n+1-N})$$

and E_0 is the initial prediction error energy.

In these conditions, the actual AC matrix estimate is

$$R_N^*(n) = \sum_{p=1}^{n} W^{n-p} X(p) X^t(p) + E_0 W(n) \tag{7.15}$$

The value $R_N^{-1}(1)$ is needed because

$$D(1) = R_N^{-1}(1) X(1) = G(1)$$

It can be calculated with the help of the matrix inversion lemma. Finally,

$$D(1) = G(1) = \frac{1}{E_0 + X^t(1) W^{-1}(1) X(1)} W^{-1}(1) X(1) \tag{7.16}$$

and for the prediction error energy $E_a(1) = WE_0$.

The weighting factor W introduces an exponential time observation window on the signal. Instead, it can be advantageous in some applications—for example, when the signal statistics can change abruptly—to use a constant time-limited window. The FLS algorithms can cope with that situation.

7.2. A SLIDING WINDOW ALGORITHM

The sliding window algorithms are characterized by the fact that the cost function $J_{\text{SW}}(n)$ to be minimized bears on the N_0 most recent output error samples:

$$J_{\text{SW}}(n) = \sum_{p=n+1-N_0}^{n} [y(p) - X^t(p) H(n)]^2 \tag{7.17}$$

where N_0 is a fixed number representing the length of the observation time window, which slides on the time axis. In general, no weighting factor is used in that case, $W = 1$. Clearly, the AC matrix and cross-correlation vector estimations are

$$R_N(n) = \sum_{p=n+1-N_0}^{n} X(p) X^t(p), \quad r_{yx}(n) = \sum_{p=n+1-N_0}^{n} y(p) X(p) \tag{7.18}$$

Again the matrix $R_{N+1}(n+1)$ can be partitioned as

$$R_{N+1}(n+1) = \sum_{p=n+2-N_0}^{n+1} \begin{bmatrix} x(p) \\ X(p-1) \end{bmatrix} [x(p), X^t(p-1)]$$
$$= \left[\begin{array}{c:c} \sum\limits_{p=n+2-N_0}^{n+1} x^2(n+1) & [r_N^a(n+1)]^t \\ \hdashline r_N^a(n+1) & R_N(n) \end{array}\right] \tag{7.19}$$

and

$$R_{N+1}(n+1) = \sum_{p=n+2-N_0}^{n} \begin{bmatrix} X(p) \\ X(p-N) \end{bmatrix} [X^t(p), x(p-N)]$$
$$= \begin{bmatrix} R_N(n+1) & r_N^b(n+1) \\ [r_N^b(n+1)]^t & \sum\limits_{p=n+2-N_0}^{n+1} x^2(p-N) \end{bmatrix} \tag{7.20}$$

However, the recurrence relations become more complicated. For the AC matrix estimate, one has

$$R_N(n+1) = R_N(n) + X(n+1)X^t(n+1) - X(n+1-N_0)X^t(n+1-N_0) \tag{7.21}$$

For the cross-correlation vector,

$$r_{yx}(n+1) = r_{yx}(n) + y(n+1)X(n+1) - y(n+1-N_0)X(n+1-N_0) \tag{7.22}$$

The coefficient updating equation is obtained, as before, from

$$R_N(n+1)H(n+1) = r_{yx}(n+1)$$

by substituting (7.22) and then, replacing $R_N(n)$ by its equivalent given by (7.21):

$$\begin{aligned} H(n+1) = H(n) &+ R_N^{-1}(n+1)X(n+1)[y(n+1) - X^t(n+1)H(n)] \\ &- R_N^{-1}(n+1) \equiv (n+1-N_0) \\ &\times [y(n+1-N_0) - X^t(n+1-N_0)H(n)] \end{aligned} \tag{7.23}$$

backward variables are showing up: the backward innovation error is

$$e_0(n+1) = y(n+1-N_0) - X^t(n+1-N_0)H(n) \tag{7.24}$$

and the backward adaptation gain is

$$G_0(n+1) = R_N^{-1}(n+1)X(n+1-N_0) \tag{7.25}$$

In concise form, equation (7.23) is rewritten as

$$H(n+1) = H(n) + G(n+1)e(n+1) - G_0(n+1)e_0(n+1)$$

These variables have to be computed and updated in the sliding window algorithms.

Partitioning (7.19) yields

$$R_{N+1}(n+1)\begin{bmatrix} 0 \\ G_0(n) \end{bmatrix} = \begin{bmatrix} x(n+1-N_0) \\ X(n-N_0) \end{bmatrix} - \begin{bmatrix} \varepsilon_{0a}(n+1) \\ 0 \end{bmatrix} \tag{7.26}$$

with

$$\varepsilon_{0a}(n+1) = x(n+1-N_0) - A^t(n+1)X(n-N_0) \tag{7.27}$$

where the forward prediction coefficient vector is

$$A(n+1) = R_N^{-1}(n) \sum_{p=n+2-N_0}^{n+1} x(p)X(p-1) \tag{7.28}$$

Similarly, the second partitioning (7.20) yields

$$R_{N+1}(n+1)\begin{bmatrix} G_0(n+1) \\ 0 \end{bmatrix} = X_1(n+1-N_0) - \begin{bmatrix} 0 \\ \varepsilon_{0b}(n+1) \end{bmatrix} \tag{7.29}$$

with

$$\varepsilon_{0b}(n+1) = x(n+1-N_0-N) - B^t(n+1)X(n+1-N_0) \tag{7.30}$$

and

$$B(n+1) = R_N^{-1}(n+1)r_N^b(n+1) \tag{7.31}$$

Now, combining the above equations with matrix prediction equations, as in Section 6.4, leads to

$$G_{01}(n+1) = \begin{bmatrix} 0 \\ G(n) \end{bmatrix} - \frac{\varepsilon_{0a}(n+1)}{E_a(n+1)}\begin{bmatrix} 1 \\ -A(n+1) \end{bmatrix} = \begin{bmatrix} M_0(n+1) \\ m_0(n+1) \end{bmatrix} \tag{7.32}$$

and

$$\begin{aligned} G_0(n+1) &= M_0(n+1) + \frac{\varepsilon_{0b}(n+1)}{E_b(n+1)}B(n+1) \\ m_0(n+1) &= \frac{\varepsilon_{0b}(n+1)}{E_b(n+1)} \end{aligned} \tag{7.33}$$

Clearly, the updating technique is the same for both adaptation gains $G(n)$ and $G_0(n)$. The adequate prediction errors have to be employed.

The method used to derive the coefficient recursion (7.23) applies to linear prediction as well; hence

$$\begin{aligned}A(n+1) = A(n) &+ R_N^{-1}(n)X(n)[x(n+1) - X^t(n)A(n)] \\ &- R_N^{-1}(n)X(n-N_0)[x(n+1-N_0) - X^t(n-N_0)A(n)]\end{aligned} \tag{7.34}$$

or, in more concise form,

$$A(n+1) = A(n) + G(n)e_a(n+1) - G_0(n)e_{0a}(n+1)$$

Now, the prediction error energy $E_a(n+1)$, which appears in the matrix prediction equations, is

$$E_a(n+1) = \sum_{p=n+2-N_0}^{n+1} x^2(p) - A^t(n+1)r_N^a(n+1) \tag{7.35}$$

Substituting (7.34) and the recursion for $r_N^a(n+1)$ into the above expression, as in Section 6.3, leads to

$$E_a(n+1) = E_a(n) + e_a(n+1)\varepsilon_a(n+1) - e_{0a}(n+1)\varepsilon_{0a}(n+1) \tag{7.36}$$

The variables needed to perform the calculations in (7.32), and in the same equation for $G_1(n+1)$, are available and the results can be used to get the updated gains.

The backward prediction coefficient vector is updated by

$$B(n+1) = B(n) + G(n+1)e_b(n+1) - G_0(n+1)e_{0b}(n+1) \tag{7.37}$$

which leads to the set of equations:

$$\begin{aligned}&G(n+1)[1 - m(n+1)e_b(n+1)] \\ &\qquad = M(n+1) + m(n+1)B(n) - G_0(n+1)e_{0b}(n+1)m(n+1) \\ &G_0(n+1)[1 + m_0(n+1)e_{0b}(n+1)] \\ &\qquad = M_0(n+1) + m_0(n+1)B(n) + G(n+1)e_b(n+1)m_0(n+1)\end{aligned} \tag{7.38}$$

Finally, letting

$$k = \frac{m(n+1)}{1 + m_0(n+1)e_{0b}(n+1)}, \quad k_0 = \frac{m_0(n+1)}{1 - m(n+1)e_b(n+1)} \tag{7.39}$$

we obtain the adaptation gains

$$\begin{aligned}G(n+1) &= \frac{1}{1 - ke_b(n+1)}[M(n+1) + kB(n) - ke_{0b}(n+1)M_0(n+1)] \\ G_0(n+1) &= \frac{1}{1 + k_0e_{0b}(n+1)}[M_0(n+1) + k_0B(n) + k_0e_b(n+1)M(n+1)]\end{aligned} \tag{7.40}$$

The algorithm is then completed by the backward coefficient updating equation (7.37).

The initial conditions are those of the algorithm in Section 6.4, the extra operations being carried out only when the time index n exceeds the window length N_0.

Overall the sliding window algorithm based on a priori errors has a computational organization which closely follows that of the exponential window algorithm, but it performs the operation twice to update and use its two adaptation gains. The sequence of operations is given in Figure 7.1 and the FORTRAN subroutine is given in Annex 7.1.

More efficient sliding window algorithms, but with a less regular structure, can be worked out by decomposing in two different steps the sequence of operations for each new input signal sample [2].

As concerns the performance, the analysis of Section 6.11 can be reproduced for the sliding window. In system identification, the mean value of the residual error power can be estimated with the help of equation (6.152), which leads to

$$E_R(n) = E_{\min}\left(1 + \frac{N}{N_0}\right), \qquad n > N_0 \tag{7.41}$$

It is interesting to compare with the exponential window and consider equation (6.160). The window length N_0 and the forgetting factor W are related by

$$N_0 = \frac{1+W}{1-W}, \qquad W = \frac{N_0 - 1}{N_0 + 1} \tag{7.42}$$

To study the convergence, let us assume that, at time n_0, the system to be identified undergoes an abrupt change in its coefficients, from vector H_1 to vector H_2. Then the definition of $H(n)$ yields

$$H(n) = R_N^{-1}(n)\left(\sum_{p=n-N_0}^{n_0} y(p)X(p) + \sum_{p=n_0+1}^{n} y(p)X(p)\right) \tag{7.43}$$

For the exponential window, in these conditions one gets

$$E[H(n) - H_2] = W^{n-n_0}[H_1 - H_2] \tag{7.44}$$

and for the sliding window

$$E[H(n) - H_2] = \frac{N_0 - (n - n_0)}{N_0}[H_1 - H_2], \qquad n_0 \leqslant n \leqslant n_0 + N_0 \tag{7.45}$$

In the latter case, the difference vanishes after N_0 samples, as shown in Figure 7.2. It is the main advantage of the approach [3].

ALGORITHM F.L.S.S.W.

AVAILABLE AT TIME n:

COEFFICIENTS OF ADAPTIVE FILTER : $H(n)$
COEFF. OF FORWARD PREDICTOR : $A(n)$
COEFF. OF BACKWARD PREDICTOR : $B(n)$
DATA VECTOR : $X(n)$
WINDOW LENGTH : N_0
DELAYED DATA VECTOR : $X(n-N_0)$
ADAPTATION GAIN : $G(n)$
ADAPTATION GAIN FOR DELAYED DATA : $G_0(n)$
FORWARD PREDICTION ERROR ENERGY : $E_a(n)$

NEW DATA AT TIME n:

Input Signal : $x(n+1)$; Reference : $y(n+1)$

ADAPTATION GAIN UPDATING :

$e_a(n+1) = x(n+1) - A^t(n)\, X(n)$

$e_{0a}(n+1) = x(n+1-N_0) - A^t(n)\, X(n-N_0)$

$A(n+1) = A(n) + G(n)\, e_a(n+1) - G_0(n)\, e_{0a}(n+1)$

$\varepsilon_a(n+1) = x(n+1) - A^t(n+1)\, X(n)$

$\varepsilon_{0a}(n+1) = x(n+1-N_0) - A^t(n+1)\, X(n-N_0)$

$E_a(n+1) = E_a(n) + e_a(n+1)\, \varepsilon_a(n+1) - e_{0a}(n+1)\, \varepsilon_{0a}(n+1)$

$$G_1(n+1) = \begin{bmatrix} 0 \\ G(n) \end{bmatrix} + \frac{e_a(n+1)}{E_a(n+1)} \begin{bmatrix} 1 \\ -A(n+1) \end{bmatrix} = \begin{bmatrix} M(n+1) \\ m(n+1) \end{bmatrix}$$

$$G_{01}(n+1) = \begin{bmatrix} 0 \\ G_0(n) \end{bmatrix} + \frac{\varepsilon_{0a}(n+1)}{E_a(n+1)} \begin{bmatrix} 1 \\ -A(n+1) \end{bmatrix} = \begin{bmatrix} M_0(n+1) \\ m_0(n+1) \end{bmatrix}$$

$e_b(n+1) = x(n+1-N) - B^t(n)\, X(n+1)$

$e_{0b}(n+1) = x(n+1-N-N_0) - B^t(n)\, X(n+1-N_0)$

$k = m(n+1) / [\, 1 + m_0(n+1)\, e_{0b}(n+1)\,]$

$k_0 = m_0(n+1) / [\, 1 - m(n+1)\, e_b(n+1)\,]$

$$G(n+1) = \frac{1}{1 - k\, e_b(n+1)} [\, M(n+1) + k\, B(n) - k\, e_{0b}(n+1)\, M_0(n+1)\,]$$

$$G_0(n+1) = \frac{1}{1 + k_0 e_{0b}(n+1)} [\, M_0(n+1) + k_0\, B(n) + k_0 e_b(n+1)\, M(n+1)\,]$$

$B(n+1) = B(n) + G(n+1)\, e_b(n+1) - G_0(n+1)\, e_{0b}(n+1)$

ADAPTIVE FILTER .

$e(n+1) = y(n+1) - H^t(n)\, X(n+1)$

$e_0(n+1) = y(n+1-N_0) - H^t(n)\, X(n+1-N_0)$

$H(n+1) = H(n) + G(n+1)\, e(n+1) - G_0(n+1)\, e_0(n+1)$

FIG. 7.1 Fast least squares sliding window algorithm.

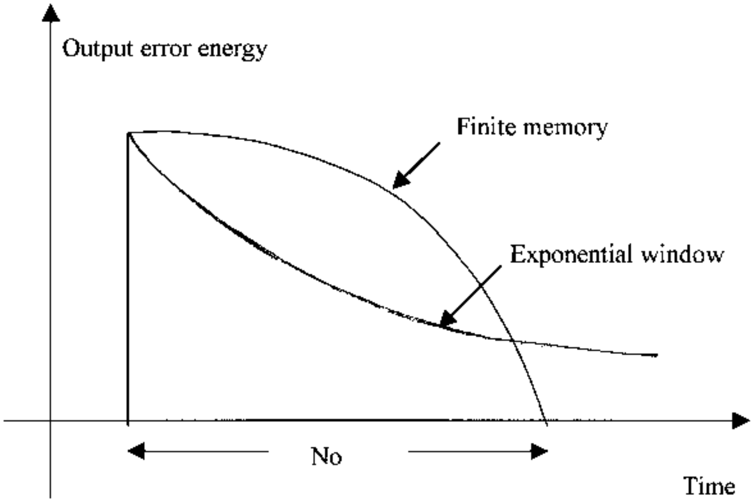


FIG. 7.2 Step responses of exponential and sliding window algorithms.

The sliding window algorithm is subject to roundoff errrors accumulation, and the control procedure of Section 6.9 can be applied.

7.3. ALGORITHMS WITH VARIABLE WEIGHTING FACTORS

The tracking capability of weighted least squares adaptive filters is related to the weighting factor value, which defines the observation time window of the algorithm. In the presence of evolving signals, it may be advantageous, in some circumstances, to continuously adjust the weighting factor, using a priori information on specific parameters or measurement results.

In the derivation of fast algorithms, the varying weighting factor $W(n)$ raises a problem, and it is necessary to introduce the weighting operation on the input signal and the reference signal rather than on the output error sequence, as previously [4]. Accordingly, the data are weighted as follows at time n:

$$y(n),\ W(n-1)y(n-1), \dots, \left[\prod_{i=1}^{n-p} W(n-i)\right] y(p), \dots$$

and the cost function is

$$J(n) = \sum_{p=1}^{n} \left[\left[\prod_{i=1}^{n-p} W(n-i) \right] y(p) - H^t(n) \left[\prod_{i=1}^{n-p} W(n-i) \right] D(p) X(p) \right]^2 \tag{7.46}$$

where $D(p)$ is the diagonal matrix

$$D(p) = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & W(p-1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \prod_{i=1}^{N-1} & W(p-i) \end{bmatrix} \tag{7.47}$$

After factorization, the cost function can be rewritten as

$$J(n) = \sum_{p=1}^{n} \left[\prod_{i=1}^{n-p} W^2(n-i) \right] [y(p) - X^t(p)D(p)H(n)]^2 \tag{7.48}$$

The coefficient vector that minimizes the cost function is obtained through derivation and it is given by the conventional equation

$$H(n) = R_N^{-1}(n) r_{yx}(n) \tag{7.49}$$

but the AC matrix, now, is

$$R_N(n) = \sum_{p=1}^{n} \left[\prod_{i=1}^{n-p} W^2(n-i) \right] D(p)X(p)X^t(p)D(p) \tag{7.50}$$

and the cross-correlation vector is

$$r_{yx}(n) = \sum_{p=1}^{n} \left[\prod_{i=1}^{n-p} W^2(n-i) \right] y(p)D(p)X(p) \tag{7.51}$$

The recurrence relations become

$$\begin{aligned} R_N(n+1) &= W^2(n)R_N(n) + D(n+1)X(n+1)X^t(n+1)D(n+1) \\ r_{yx}(n+1) &= W^2(n)r_{yx}(n) + y(n+1)D(n+1)X(n+1) \end{aligned} \tag{7.52}$$

and, for the coefficient vector

$$H(n+1) = H(n) + G(n+1)e(n+1) \tag{7.53}$$

The adaptation gain is expressed by

$$G(n+1) = R_N^{-1}(n+1)D(n+1)X(n+1) \tag{7.54}$$

and the output error is

$$e(n+1) = y(n+1) - X^t(n+1)D(n+1)H(n) \tag{7.55}$$

The same approach, when applied to forward linear prediction, leads to the cost function

$$E_a(n) = \sum_{p=1}^{n}\left[\left[\prod_{i=1}^{n-p} W^2(n-i)\right]x(p) - X^t(p-1)D(p-1)W(p-1)A(n)\right]^2 \tag{7.56}$$

and the prediction coefficient vector is

$$A(n) = [W^2(n-1)R_N(n-1)]^{-1} r_N^a(n) \tag{7.57}$$

In fact, the delay on the input data vector introduces additional weighting terms in the equations, and the correlation vector is given by

$$r_N^a(n) = \sum_{p=1}^{n}\left[\prod_{i=1}^{n-p} W^2(n-i)\right]x(p)W(p-1)D(p-1)X(p-1) \tag{7.58}$$

Exploiting the recurrence relationships for $R_N(n)$ and $r_N^a(n)$ leads to the following recursion for the prediction coefficient vector:

$$A(n+1) = A(n) + W^{-1}(n)G(n)e_a(n+1) \tag{7.59}$$

where $e_a(n+1)$ is the forward a priori prediction error

$$e_a(n+1) = x(n+1) - X^t(n)D(n)W(n)A(n) \tag{7.60}$$

Now, the adaptation gain can be updated using a partition of the AC matrix $R_N(n+1)$ as follows:

$$R_{N+1}(n+1) = \sum_{p=1}^{n}\left[\prod_{i=1}^{n+1-p} W^2(n+1-i)\right]\left[\begin{matrix} x(p) \\ W(p-1)D(p-1)X(p-1) \end{matrix}\right] [x(p), X^t(p-1)D(p-1)W(p-1) \tag{7.61}$$

and, in a more concise form,

$$R_{N+1}(n+1) = \begin{bmatrix} R_1(n+1) & r_N^a(n+1) \\ r_N^a(n+1) & W^2(n)R_N(n) \end{bmatrix} \tag{7.62}$$

Let us consider the product

$$R_{N+1}(n+1)\begin{bmatrix} 0 \\ W^{-1}(n)G(n) \end{bmatrix} = \begin{bmatrix} x(n) \\ W(n)D(n)X(n) \end{bmatrix} - \begin{bmatrix} \varepsilon_a(n+1) \\ 0 \end{bmatrix} \tag{7.63}$$

where the a posteriori forward prediction error is

$$\varepsilon_a(n+1) = x(n+1) - X^t(n)D(n)W(n)A(n+1) \tag{7.64}$$

The adaptation gain with $N+1$ elements is computed by

$$G_1(n+1) = \begin{bmatrix} 0 \\ W^{-1}(n)G(n) \end{bmatrix} + \frac{\varepsilon_a(n+1)}{E_a(n+1)} \begin{bmatrix} 1 \\ -A(n+1) \end{bmatrix} \tag{7.65}$$

Then, the updated adaptation gain $G(n+1)$ is derived from $G_1(n+1)$ using backward linear prediction equations. The cost function is

$$E_b(n) = \sum_{p=1}^{n} \left[\prod_{i=1}^{n-p} W^2(n-i) \right] \left[x(p-N) \left[\prod_{i=1}^{N} W(p-i) \right] - X^t(p)D(p)B(n) \right]^2 \tag{7.66}$$

and the backward linear prediction coefficient recursion is

$$B(n+1) = B(n) + G(n+1)e_b(n+1) \tag{7.67}$$

with

$$e_b(n+1) = x(n+1-N) \left[\prod_{i=1}^{N} W(n+1-i) \right] - B^t(n)D(n+1)X(n+1) \tag{7.68}$$

As in Section 6.4, the backward linear prediction parameters can be used to compute $G_1(n+1)$, which leads to the determination of $G(n+1)$.

Finally, an algorithm with a variable weighting factor is obtained and it has the same computational organization as the algorithm in Figure 6.4, provided that the equations to compute the variables $e_a(n+1)$, $A(n+1)$, $\varepsilon_a(n+1)$, $G_1(n+1)$, and $e_b(n+1)$ are modified as above. Of course, $W(n+1)$ is a new datum at time n.

The approach can be applied to other fast least squares algorithms, to accommodate variable weighting factors. The crucial option is the weighting of the signals insead of the output error sequence. Another area where the same option is needed is forward–backward linear prediction.

7.4. FORWARD–BACKWARD LINEAR PREDICTION

In some applications, and particularly in spectral analysis, it is advantageous to define linear prediction from a cost function which is the sum of forward and backward prediction error energies [5].

Accordingly, the cost function is the energy of the forward–backward linear prediction error signal, expressed by

$$E(n) = \sum_{p=1}^{n} [W^{n-p} x(p) - B^t(n) J W^{n-p+1} D X(p-1)]^2 \\ + [W^{n-(p-n)} x(p-N) - B^t(n) W^{n-p} D X(p)]^2 \tag{7.69}$$

where J is the coidentity matrix (3.63) and D is the diagonal weighting matrix

$$D = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & W & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & W^{N-1} \end{bmatrix} \tag{7.70}$$

The objective is to compute a coefficient vector which is used for backward linear prediction and, also, with elements in reversed order, for forward linear prediction, which explains the presence of the coidentity matrix J.

The vector of prediction coefficients is expressed by

$$D(n) = [R_N(n) + W^2 J R_N(n-1) J]^{-1} [J r_N^a(n) + r_N^b(n)] \tag{7.71}$$

where

$$\begin{aligned} R_N(n) &= \sum_{p=1}^{n} W^{2(n-p)} D X(p) X^t(p) D \\ r_N^a(n) &= \sum_{p=1}^{n} W^{2(n-p)} x(p) W D X(p-1) \\ r_N^b(n) &= \sum_{p=1}^{n} W^{2(n-p)} x(p-N) W^N D X(p) \end{aligned} \tag{7.72}$$

Due to the particular weighting, the recurrence equations for the variables are

$$\begin{aligned} R_N(n) &= W^2 R_N(n-1) + D X(n) X^t(n) D \\ r_N^a(n) &= W^2 r_N^a(n-1) + x(n) W D X(n-1) \\ r_N^b(n) &= W^2 r_N^b(n-1) + x(n-N) W^N D X(n) \end{aligned} \tag{7.73}$$

The same procedure as in the preceding section leads to the recurrence equation

$$B(n+1) = B(n) + W^{-2} G_1(n) \varepsilon_a(n+1) + W^{-2} G_2(n+1) \varepsilon_b(n+1) \tag{7.74}$$

where the forward adaptation gain is

$$G_1(n) = [R_N(n) + W^2 J R_N(n-1) J]^{-1} J W D X(n) \tag{7.75}$$

the forward a posteriori prediction error is

$$\varepsilon_a(n+1) = x(n+1) - WX^t(n)DJB(n+1) \tag{7.76}$$

the backward adaptation gain is

$$G_2(n+1) = [R_N(n) + W^2 JR_N(n-1)J]^{-1} DX(n+1) \tag{7.77}$$

and the a posteriori backward prediction error is

$$\varepsilon_b(n+1) = x(n+1-N)W^N - X^t(n+1)DB(n+1) \tag{7.78}$$

Since forward prediction and backward prediction are combined, the relationships between a priori and a posteriori errors take a matrix form

$$\begin{bmatrix} \varepsilon_a(n+1) \\ \varepsilon_b(n+1) \end{bmatrix} = \begin{bmatrix} 1 + W^{-1}X^t(n)DJG_1(n) & W^{-1}X^t(n)DJG_2(n+1) \\ W^{-2}X^t(n+1)DG_1(n) & 1 + W^{-2}X^t(n+1)DG_2(n+1) \end{bmatrix}$$

$$\begin{bmatrix} e_a(n+1) \\ e_b(n+1) \end{bmatrix} \tag{7.79}$$

As concerns the error energy, it is computed by

$$E(n) = \sum_{p=1}^{n} W^{2(n-p)}[x^2(p) + W^{2N}x^2(p-N)] - B^t(n)[Jr_N^a(n) + r_N^b(n)] \tag{7.80}$$

or, in a more concise recursive form,

$$E(n+1) = W^2E(n) + e_a(n+1)\varepsilon_a(n+1) + e_b(n+1)\varepsilon_b(n+1) \tag{7.80}$$

Now, in order to obtain a fast algorithm, it is necessary to introduce an intermediate adaptation gain $U(n)$ defined by

$$[R_N(n-1) + JR_N(n-1)J]U(n) = DX(n) \tag{7.81}$$

Exploiting the recursion for $R_N(n-1)$, one gets

$$[R_N(n-1) + W^2JR_N(n-2)J + JDX(n-1)X^t(n-1)DJ]U(n) = DX(n) \tag{7.82}$$

Using (7.75) and (7.77), the intermediate adaptation gain $U(n)$ is expressed in a simple form

$$U(n) = G_2(n) - W^{-1}G_1(n-1)X^t(n-1)DJU(n) \tag{7.83}$$

and more concisely

$$U(n) = G_2(n) - G_1(n-1)\varepsilon_u(n) \tag{7.84}$$

with

$$\varepsilon_u(n) = \frac{X^t(n-1)DJG_2(n)}{W + X^t(n-1)DJG_1(n-1)} \tag{7.85}$$

The intermediate gain can be used to update $G_1(n)$. From definitions (7.75) and (7.81), one gets

$$G_1(n) = W^{-1}JU(n) - W^{-2}U(n)X^t(n)DG_1(n) \tag{7.86}$$

or, as above

$$G_1(n) = W^{-1}JU(n) - U(n)\varepsilon_y(n) \tag{7.87}$$

with

$$\varepsilon_y(n) = \frac{X^t(n)DJU(n)}{W^2 + X^t(n)DU(n)} \tag{7.88}$$

The updating of the backward linear prediction adaptation gain exploits the two decompositions of the matrix $R_{N+1}(n)$ as defined by (7.73). After some algebraic manipulations, one gets

$$[R_{N+1}(n) + JR_{N+1}(n)J] = \begin{bmatrix} R_N(n) + W^2JR_N(n-1)J & r_N^b(n) + Jr_N^a(n) \\ [r_N^b(n) + Jr_N^a(n)]^t & R_1(n) + W^{2N}R_1(n-N) \end{bmatrix} \tag{7.89}$$

Then it is sufficient to proceed, as in Chapter 6, to compute the intermediate adaptation gain with $N+1$ elements, denoted $U_1(n+1)$, from the forward adaptation gain by

$$JU_1(n+1) = \begin{bmatrix} G_1(n) \\ 0 \end{bmatrix} + \frac{e_a(n+1)}{E(n)} \begin{bmatrix} -B(n) \\ 1 \end{bmatrix} = \begin{bmatrix} m(n+1) \\ JM(n+1) \end{bmatrix} \tag{7.90}$$

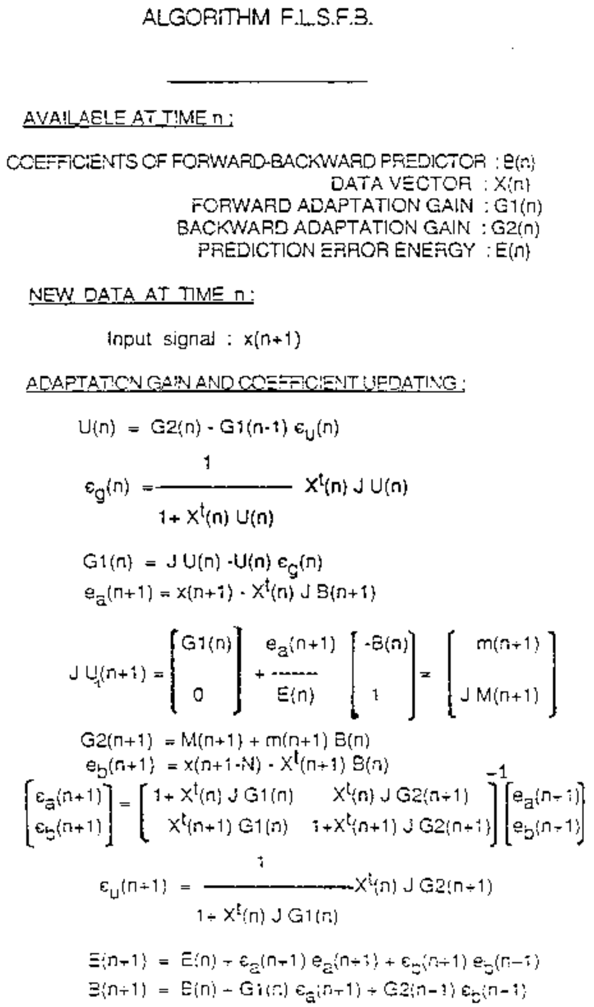
Similarly, with the backward adaptation gain, an alternative expression is obtained

$$U_1(n+1) = \begin{bmatrix} G_2(n+1) \\ 0 \end{bmatrix} + \frac{e_b(n+1)}{E(n)} \begin{bmatrix} -B(n) \\ 1 \end{bmatrix} = \begin{bmatrix} M(n+1) \\ m(n+1) \end{bmatrix} \tag{7.91}$$

And, finally, the backward adaptation gain is given by

$$G_2(n+1) = M(n+1) + m(n+1)B(n), \qquad m(n+1) = \frac{e_b(n+1)}{E(n)} \tag{7.92}$$

This equation completes the algorithm. The list of operations is given in Figure 7.3 and the FORTRAN program is given in Annex 7.2. Applications of the FBLP algorithm can be found in real time signal analysis.



ALGORITHM F.L.S.F.B.

AVAILABLE AT TIME n :

COEFFICIENTS OF FORWARD-BACKWARD PREDICTOR	: $B(n)$
DATA VECTOR	: $X(n)$
FORWARD ADAPTATION GAIN	: $G1(n)$
BACKWARD ADAPTATION GAIN	: $G2(n)$
PREDICTION ERROR ENERGY	: $E(n)$

NEW DATA AT TIME n :

Input signal : $x(n+1)$

ADAPTATION GAIN AND COEFFICIENT UPDATING :

$$U(n) = G2(n) - G1(n-1)\,\varepsilon_u(n)$$

$$\varepsilon_g(n) = \frac{1}{1+X^t(n)\,U(n)}\,X^t(n)\,J\,U(n)$$

$$G1(n) = J\,U(n) - U(n)\,\varepsilon_g(n)$$

$$e_a(n+1) = x(n+1) - X^t(n)\,J\,B(n+1)$$

$$J\,U_1(n+1) = \begin{bmatrix} G1(n) \\ 0 \end{bmatrix} + \frac{e_a(n+1)}{E(n)} \begin{bmatrix} -B(n) \\ 1 \end{bmatrix} = \begin{bmatrix} m(n+1) \\ J\,M(n+1) \end{bmatrix}$$

$$G2(n+1) = M(n+1) + m(n+1)\,B(n)$$

$$e_b(n+1) = x(n+1-N) - X^t(n+1)\,B(n)$$

$$\begin{bmatrix} \varepsilon_a(n+1) \\ \varepsilon_b(n+1) \end{bmatrix} = \begin{bmatrix} 1+X^t(n)\,J\,G1(n) & X^t(n)\,J\,G2(n+1) \\ X^t(n+1)\,G1(n) & 1+X^t(n+1)\,J\,G2(n+1) \end{bmatrix}^{-1} \begin{bmatrix} e_a(n+1) \\ e_b(n+1) \end{bmatrix}$$

$$\varepsilon_u(n+1) = \frac{1}{1+X^t(n)\,J\,G1(n)}\,X^t(n)\,J\,G2(n+1)$$

$$E(n+1) = E(n) + \varepsilon_a(n+1)\,e_a(n+1) + \varepsilon_b(n+1)\,e_b(n+1)$$

$$B(n+1) = B(n) - G1(n)\,\varepsilon_a(n+1) + G2(n+1)\,\varepsilon_b(n+1)$$

FIG. 7.3 Fast least squares forward–backward linear prediction algorithm.

7.5. LINEAR PHASE ADAPTIVE FILTERING

In some applications, like identification and equalization, the symmetry of the filter coefficients is sometimes required. The results of the above section can be applied directly in that case [5].

Let us first consider linear prediction with linear phase. The cost function is

$$E(n) = \sum_{p=1}^{n} \left(\left[x(p) - \frac{B_\ell^t(n)}{2} JX(p-1) \right]^2 + \left[x(p-1-N) - \frac{B_\ell^t(n)}{2} X(p-1) \right]^2 \right) \tag{7.93}$$

and the coefficients of the corresponding linear prediction filter make a vector $B_\ell(n)$ satisfying the equation

$$[R_N(n+1) + JR_N(n-1)J] \frac{B_\ell(n)}{2} = Jr_N^a(n) + r_N^b(n-1) \tag{7.94}$$

For simplification purposes, the weighting factor W has been omitted in the above expressions, which are very close to (7.69) and (7.71) for forward–backward linear prediction. In fact, the difference is a mere delay in the backward terms. Therefore, the intermediate adaptation gain can be used.

The linear phase coefficient vector $B_\ell(n)$ can be updated recursively by

$$\begin{aligned} \frac{B_\ell(n+1)}{2} = \frac{B_\ell(n)}{2} &+ [R_N(n-1) + JR_N(n-1)J]^{-1} \\ &(JX(n)\varepsilon_a(n+1) + X(n)\varepsilon_{0b}(n+1)] \end{aligned} \tag{7.95}$$

where the error signals are defined by

$$\varepsilon_a(n+1) = x(n+1) - X^t(n) \frac{B_\ell(n+1)}{2} \tag{7.96}$$

and

$$\varepsilon_{0b}(n+1) = x(n-N) - X^t(n+1) \frac{B_\ell(n+1)}{2} \tag{7.97}$$

The linear phase constraint, which is the symmetry of the coefficients, is imposed if the error signals are equal:

$$\varepsilon_a(n+1) = \varepsilon_{0b}(n+1) = \tfrac{1}{2}\varepsilon(n+1) = \tfrac{1}{2}[x(n+1) + x(n-N) - X^t(n)B_\ell(n+1)] \tag{7.98}$$

Hence the coefficient updating equation

$$B_\ell(n+1) = B_\ell(n) + [U(n) + JU(n)]\varepsilon(n+1) \tag{7.99}$$

where $U(n)$ is the intermediate adaptation gain defined by (7.81). The “a posteriori” error $\varepsilon(n+1)$ can be computed from the “a priori” error $e(n+1)$. Starting from the definitions of the errors, after some algebraic manipulations, the following proportionality expression is obtained:

$$e(n+1) = \varepsilon(n+1)[1 + X^t(n)DW[U(n) + JU(n)]] \tag{7.100}$$

As concerns the linear phase adaptive filter, it can be handled in very much the same way. The cost function is

$$J(n) = \sum_{p=1}^{n} \left[y(p) - H_p(n) \frac{X^t(p)J + X^t(p)}{2} \right]^2 \tag{7.101}$$

and the coefficient vector $H_\ell(n)$ satisfies

$$\sum_{p=1}^{n} [JX(p)X^t(p)J + X(p)X^t(p)] \frac{H_\ell(n)}{2} = \sum_{p=1}^{n} y(p)[JX(p) + X(p)] \tag{7.102}$$

Hence, the recursion

$$H_\ell(n+1) = H_\ell(n) + [U(n+1) + JU(n+1)]\varepsilon_h(n+1) \tag{7.103}$$

follows, and the error proportionality relationship is

$$e_h(n+1) = \varepsilon_h(n+1)[1 + X^t(n+1)DW[U(n+1) + JU(n+1)]] \tag{7.104}$$

The "a priori" error is computed according to its definition by

$$e_h(n+1) = y(n+1) - X^t(n+1)H_\ell(n) \tag{7.105}$$

Finally, a complete algorithm for least squares linear phase adaptive filtering consists of the equations in Figure 7.3 to update the intermediate gain and the three filter section equations (7.105), (7.104), and (7.103).

The above algorithm is elegant but computationally complex. A simpler approach is obtained directly from the general adaptive filter algorithm, and is presented in a later section, after the case of adaptive filtering with linear constraints has been dealt with.

7.6. CONSTRAINED ADAPTIVE FILTERING

Constrained adaptive filtering can be found in several signal processing techniques like minimum variance spectral analysis and antenna array processing. In fact, many particular situations in adaptive filtering can be viewed as a general case with specific constraints. Therefore it is important to be able to include constraints in adaptive algorithms [6].

The constraints are assumed to be linear, and they are introduced by the linear system

$$C^t H(n) = F \tag{7.106}$$

where C is the $N \times K$ constraint matrix and F is a K-element response vector. The set of filter coefficients that minimizes the cost function

$$J(n) = \sum_{p=1}^{n} W^{n-p}[y(p) - H^t(n)X(p)]^2 \tag{7.107}$$

subject to the constraint (7.83), is obtained through the Lagrange multiplier method.

Let us introduce an alternative cost function

$$J'(n) = \sum_{p=1}^{n} W^{n-p}[y(p) - H^t(n)X(p)]^2 + \alpha^t C^t H(n) \tag{7.108}$$

where α is a k-element vector, the so-called Lagrange multiplier vector. The derivative of the cost function with respect to the coefficient vector is

$$\frac{\partial J'(n)}{\partial H(n)} = -2R_N(n)H(n) + 2r_{yx}(n) + C\alpha \tag{7.109}$$

and it is zero for

$$H(n) = R_N^{-1}(n)[r_{yx}(n) + \tfrac{1}{2}C\alpha] \tag{7.110}$$

Now, this coefficient vector must satisfy the constraint (7.106), which implies

$$C^t[R_N^{-1}(n)[r_{yx}(n) + \tfrac{1}{2}C\alpha]] = F \tag{7.111}$$

and

$$\tfrac{1}{2}\alpha = [C^t R_N^{-1}(n)C]^{-1}[F - C^t R_N^{-1}(n)r_{yx}(n)] \tag{7.112}$$

Substituting (7.112) into (7.110) leads to the constrained least squares solution

$$H(n) = R_N^{-1}(n)r_{yx}(n) + R_N^{-1}(n)C[C^t R_N^{-1}(n)C]^{-1}[F - C^t R_N^{-1}(n)r_{yx}(n)] \tag{7.113}$$

Now, in a recursive approach, the factors which make $H(n)$ have to be updated. First let us define the $N \times k$ matrix $\Gamma(n)$ by

$$\Gamma(n) = R_N^{-1}(n)C \tag{7.114}$$

and show how it can be recursively updated. The basic recursion for the AC matrix yields the following equation, after some manipulation:

$$R_N^{-1}(n+1) = \frac{1}{W}[R_N^{-1}(n) - G(n+1)X^t(n+1)R_N^{-1}(n)] \tag{7.115}$$

Right-multiplying both sides by the constraint matrix C leads to the following equation for the updating of the matrix $\Gamma(n)$:

$$\Gamma(n+1) = \frac{1}{W}[\Gamma(n) + G(n+1)X^t(n+1)\Gamma(n)] \tag{7.116}$$

The second factor to be updated in $H(n)$ as defined by (7.113) is $[C^t\Gamma(n)]^{-1}$, and the matrix inversion lemma can be invoked. The first step in the procedure consists of multiplying both sides of (7.116) by C^t to obtain

$$C^t\Gamma(n+1) = \frac{1}{W}[C^t\Gamma(n) + C^tG(n+1)X^t(n)\Gamma(n)] \tag{7.117}$$

Clearly, the second term in the right-hand side of the above equation is the scalar product of two vectors. Therefore, the inversion formula is obtained with the help of (6.24) as

$$[C^t\Gamma(n+1)]^{-1} = W\{[C^t\Gamma(n)]^{-1} + L(n+1)X^t(n+1)\Gamma(n)[C^t\Gamma(n)]^{-1}\} \tag{7.118}$$

where $L(n+1)$ is the k-element vector defined by

$$L(n+1) = [C^t\Gamma(n)]^{-1}C^tG(n+1)/\{1 + X^t(n+1)\Gamma(n)[C^t\Gamma(n)]^{-1}C^tG(n+1)\} \tag{7.119}$$

or in a more concise form, using (7.118),

$$L(n+1) = \frac{1}{W}[C^t\Gamma(n+1)]^{-1}C^tG(n+1) \tag{7.120}$$

Once $G(n+1)$ is available, the set of equations (7.119), (7.118), (7.116) constitute an algorithm to recursively compute the coefficient vector $H(n+1)$ through equation (7.113). The adaptation gain $G(n+1)$ itself can be obtained with the help of one of the algorithms presented in Chapter 6.

7.7. A ROBUST CONSTRAINED ALGORITHM

In the algorithm derived in the previous section, the constraint vector F does not explicitly show up. In fact, it is only present in the initialization phase which consists of the two equations

$$\Gamma(0) = R_N^{-1}(0)C \tag{7.121}$$

and

$$H(0) = \Gamma(0)[C^t\Gamma(0)]^{-1}F \tag{7.122}$$

Due to the unavoidable roundoff errors, the coefficient vector will deviate from the constraints as time elapses, and a correction procedure is mandatory for long or continuous data sequences. In fact, it is necessary to derive a recursion for the coefficient vector, which is based on the output error signal. The coefficient vector can be rewritten as

$$H(n) = R_N^{-1}(n)r_{yx}(n) + \Gamma(n)[C^t\Gamma(n)]^{-1}[F - C^t R_N^{-1}(n)r_{yx}(n)] \tag{7.123}$$

Now, substituting (7.116) and (7.118) into the above equation at time $n+1$, using expression (7.120) and the regular updating equation for the unconstrained coefficient vector, the following recursion is obtained for the constrained coefficient vector

$$H(n+1) = H(n) + G(n+1)e(n+1) - W\Gamma(n+1)L(n+1)e(n+1) \tag{7.124}$$

with

$$e(n+1) = y(n+1) - H^t(n)X(n+1) \tag{7.125}$$

In simplified form, the equation becomes

$$H(n+1) = H(n) + P(n+1)G(n+1)e(n+1) \tag{7.126}$$

with the projection operator

$$P(n+1) = I_N - \Gamma(n+1)[C^t\Gamma(n+1)]^{-1}C^t \tag{7.127}$$

Robustness to roundoff errors is introduced through an additional term in the recursion, proportional to the deviation from the constraint expressed as $F - C^tH(n)$. Then the recursion becomes

$$\begin{aligned} H(n+1) = H(n) &+ P(n+1)G(n+1)e(n+1) \\ &+ \Gamma(n+1)[C^t\Gamma(n+1)]^{-1}[F - C^tH(n)] \end{aligned} \tag{7.128}$$

and it is readily verified that the coefficient vector satisfies the constraint for any n.

Some factorization can take place, which leads to an alternative expression

$$H(n+1) = P(n+1)[H(n) + G(n+1)e(n+1)] + M(n+1) \tag{7.129}$$

where

$$M(n+1) = \Gamma(n+1)[C^t\Gamma(n+1)]^{-1}F \tag{7.130}$$

At this stage, it is worth pointing out that a similar expression exists for the constrained LMS algorithm as mentioned in Section 4.12. The equations are recalled for convenience:

$$H(n+1) = P[H(n) + \delta X(n+1)e(n+1)] + M \tag{7.131}$$

with

$$M = C(C^tC)^{-1}F, \qquad P = I_N - C(C^tC)^{-1}C^t \tag{7.132}$$

However, in the LMS algorithm, the quantities M and P are fixed, while they are related to the signal autocorrelation in the FLS algorithm.

In order to finalize the robust algorithm, it is convenient to introduce the matrix $Q(n)$, with $N \times k$ elements, as

$$Q(n) = \Gamma(n)[C^t \Gamma(n)]^{-1} \tag{7.132}$$

and compute the updated coefficient vector in two steps as follows:

$$H'(n+1) = H(n) + G(n+1)e(n+1) \tag{7.133}$$

and then

$$H(n+1) = H'(n+1) + Q(n+1)[F - C^t H'(n+1)] \tag{7.134}$$

In the robust algorithm, $Q(n)$ has to be computed recursively and it must be free of roundoff error accumulation. The procedure is a direct combination of (7.116) and (7.118). Let us define the vectors

$$U(n+1) = C^t G(n+1) \tag{7.135}$$

and

$$V(n+1) = X^t(n+1)Q(n) \tag{7.136}$$

Now, the recursion is

$$Q(n+1) = [Q(n) - G(n+1)V^t(n+1)]\left[I_k + \frac{U(n+1)V^t(n+1)}{1 - V^t(n+1)U(n+1)}\right] \tag{7.137}$$

According to the definition (7.132) of the matrix $Q(n)$, in the absence of roundoff error accumulation, the following equality holds:

$$C^t Q(n+1) = I_k \tag{7.138}$$

Therefore, if $Q'(n+1)$ denotes a matrix with roundoff errrors, a correcting term can be introduced in the same manner as above, and the correct matrix is obtained as

$$Q(n+1) = Q'(n+1) + C(C^t C)^{-1}[I_k - C^t Q'(n+1)] \tag{7.139}$$

Finally, the robust constrained FLS algorithm is given in Figure 7.4. The number of multiplies, including error correction, amounts to $Nk^2 + 5Nk + k^2 + k + 2N$. Additionally, k divisions are needed. Some gain in computation is achieved if the term $C(C^t C)^{-1}$ in (7.139) is precomputed.

It is worth pointing out that the case of linear phase filters can be seen as an adaptive constrained filtering problem. The constraint matrix for an odd number of coefficients is taken as

- New data at time $(n+1)$: $x(n+1)$ and $d(n+1)$

- Compute the adaptation gain: $g(n+1)$

  (for example, using a FLS algorithm)

- Update the NxK matrix $Q(n+1)$:

$$u(n+1) = C^t g(n+1)$$

$$v^t(n+1) = x^t(n+1)Q(n)$$

$$Q'(n+1) = \left[Q(n) - g(n+1)v^t(n+1)\right]\left[I_K + \frac{u(n+1)v^t(n+1)}{1 - v^t(n+1)u(n+1)}\right]$$

$$Q(n+1) = Q'(n+1) + C\left[C^tC\right]^{-1}\left[I_K - C^tQ'(n+1)\right]$$

- Update the filter coefficients:

$$e(n+1) = d(n+1) - h^t(n)x(n+1)$$

$$h'(n+1) = h(n) + g(n+1)e(n+1)$$

$$h(n+1) = h'(n+1) + Q(n+1)\left[f - C^t h'(n+1)\right]$$

FIG. 7.4 The robust CFLS algorithm.

$$C = \begin{bmatrix} I_{(N-1)/2} \\ 0 \cdots 0 \\ \pm J_{(N-1)/2} \end{bmatrix} \tag{7.140}$$

and for N even it is

$$C = \begin{bmatrix} I_{N/2} \\ \pm J_{N/2} \end{bmatrix}$$

while the response vector in (7.106) is

$$F = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

The constrained algorithms provide an alternative to those presented in Section 7.5.

7.8. THE CASE OF COMPLEX SIGNALS

Complex signals take the form of sequences of complex numbers and are encountered in many applications, particularly in communications and radar. Adaptive filtering techniques can be applied to complex signals in a straightforward manner, the main peculiarity being that the cost functions used in the optimization process must remain real and therefore moduli are involved.

For reasons of compatibility with the subsequent study of the multidimensional case, the cost function is taken as

$$J_{cX}(n) = \sum_{p=1}^{n} W^{n-p} | y(p) - \bar{H}^t(n)X(p)|^2 \tag{7.141}$$

or

$$J_{cX}(n) = \sum_{p=1}^{n} W^{n-p} e(p)\bar{e}(p)$$

where $\bar{e}(n)$ denotes the complex conjugate of $e(n)$, and the weighting factor W is assume real.

Based on the cost function, FLS algorithms can be derived through the procedures presented in Chapter 6 [7].

The minimization of the cost function leads to

$$H(n) = R_N^{-1}(n) r_{yx}(n) \tag{7.142}$$

where

$$\begin{aligned} R_N(n) &= \sum_{p=1}^{n} W^{n-p} X(p)\bar{X}^t(p) \\ r_{yx}(n) &= \sum_{p=1}^{n} W^{n-p} \bar{y}(p)X(p) \end{aligned} \tag{7.143}$$

Note that $[\bar{R}_N(n)]^t = R_N(n)$, which is the definition of a Hermitian matrix.

The connecting matrix $R_{N+1}(n+1)$ can be partitioned as

$$\begin{aligned} R_{N+1}(n+1) &= \sum_{p=1}^{n+1} W^{n+1-p} \begin{bmatrix} x(p) \\ X(p-1) \end{bmatrix} [\bar{x}(p), \bar{X}^t(p-1)] \\ &= \begin{bmatrix} \sum_{p=1}^{n+1} W^{n+1-p}|x(p)|^2 & [\bar{r}_N^a(n+1)]^t \\ r_N^a(n+1) & R_N(n) \end{bmatrix} \end{aligned} \tag{7.144}$$

and

$$R_{N+1}(n+1) = \sum_{p=1}^{n+1} W^{n+1-p} \begin{bmatrix} X(p) \\ x(p-N) \end{bmatrix} [\bar{X}^t(p), \bar{x}(p-N)]$$
$$= \begin{bmatrix} R_N(n+1) & r_N^b(n+1) \\ [\bar{r}_N^b(n+1)]^t & \sum_{p=1}^{n+1} W^{n+1-p} |x(p-N)|^2 \end{bmatrix} \quad (7.145)$$

Following the definitions (7.42) and (7.43), the forward prediction coefficient vector is updated by

$$A(n+1) = R_N^{-1}(n) r_N^a(n+1) = A(n) + R_N^{-1}(n) X(n) [\bar{x}(n+1) - \bar{X}^t(n) A(n)] \quad (7.146)$$

or

$$A(n+1) = A(n) + G(n) \bar{e}_a(n+1) \quad (7.147)$$

where the adaptation gain has the conventional definition and

$$e_a(n+1) = x(n+1) - \bar{A}^t(n) X(n)$$

Now, using the partitioning (7.44) as before, one gets

$$R_{N+1}(n+1) \begin{bmatrix} 0 \\ G(n) \end{bmatrix} = X_1(n+1) - \begin{bmatrix} \varepsilon_a(n+1) \\ 0 \end{bmatrix} \quad (7.148)$$

which, taking into account the prediction matrix equations, leads to the same equations as for real signals:

$$G_1(n+1) = \begin{bmatrix} 0 \\ G(n) \end{bmatrix} + \frac{\varepsilon_a(n+1)}{E_a(n+1)} \begin{bmatrix} 1 \\ -A(n+1) \end{bmatrix} = \begin{bmatrix} M(n+1) \\ m(n+1) \end{bmatrix}$$

The prediction error energy $E_a(n+1)$ can be updated by the following recursion, which is obtained through the method given in Section 6.3, for $R_N(n)$ Hermitian:

$$E_a(n+1) = W E_a(n) + e_a(n+1) \bar{\varepsilon}_a(n+1) \quad (7.149)$$

The end of the procedure uses the partitioning of $R_{N+1}(n+1)$ given in equation (7.45) to express the order $N+1$ adaptation gain in terms of backward prediction variables. It can be verified that the conjugate of the backward prediction error

$$e_b(n+1) = x(n+1-N) - \bar{B}^t(n) X(n+1)$$

appears in the updated gain

$$G(n+1) = \frac{1}{1 - \bar{e}_b(n+1)m(n+1)}[M(n+1) + B(n)m(n+1)] \tag{7.150}$$

The backward prediction coefficients are updated by

$$B(n+1) = B(n) + G(n+1)\bar{e}_b(n+1) \tag{7.151}$$

Finally the FLS algorithm for complex signals based on a priori errors is similar to the one given in Figure 6.4 for real data.

There is an identity between the complex signals and the two-dimensional signals which are considered in the next section. Algorithms for complex signals are directly obtained from those given for 2-D signals by adding complex conjugation to transposition.

The prediction error ratio

$$\varphi(n) = \frac{\varepsilon_a(n+1)}{e_a(n+1)} = 1 - \bar{X}^t(n)R_N^{-1}(n)X(n) \tag{7.152}$$

is a real number, due to the Hermitian property of the AC matrix estimation $R_N(n)$. It is still limited to the interval [0, 1] and can be used as a reliable checking variable.

7.9. MULTIDIMENSIONAL INPUT SIGNALS

The input and reference signals in adaptive filters can be vectors. To begin with, the case of an input signal consisting of K elements $x_i(n)(1 \leqslant i \leqslant k)$ and a scalar reference is considered. It is illustrated in Figure 7.5. The programmable filter, whose output $\tilde{y}(n)$ is a scalar like the reference $y(n)$, consists of a set of k different filters with coefficient vectors $H_i(n)(1 \leqslant i \leqslant k)$. These coefficients can be calculated to minimize a cost function in real time, through FLS algorithms.

Let $\chi(n)$ denote the k-element input vector

$$\chi^t(n) = [x_1(n), x_2(n), \ldots, x_k(n)]$$

Assuming that each filter coefficient vector $H_i(n)$ has N elements, let $X(n)$ denote the following input vector with KN elements:

$$X^t(n) = [\chi^t(n), \chi^t(n-1), \ldots, \chi^t(n+1-N)]$$

and let $H(n)$ denote the KN element coefficient vector

$$H^t(n) = \vdots\, \overset{\longleftarrow K \longrightarrow}{[h_{11}(n), \ldots, h_{K1}(n),}\, \vdots\, \overset{\longleftarrow K \longrightarrow}{h_{12}(n), \ldots, h_{K2}(n),}\, \vdots \ldots \vdots\, \overset{\longleftarrow K \longrightarrow}{h_{1N}(n), \ldots, h_{KN}(n)]}\, \vdots$$

The output error signal $e(n)$ is

$$e(n) = y(n) - H^t(n)X(n) \tag{7.153}$$

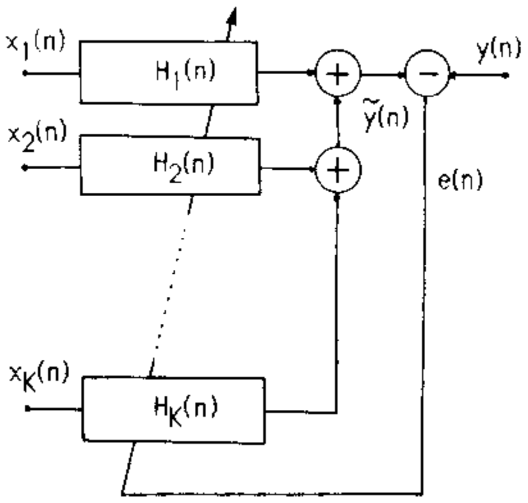


FIG. 7.5 Adaptive filter with multidimensional input and scalar reference.

The minimization of the cost function $J(n)$ associated with an exponential time window,

$$J(n) = \sum_{p=1}^{n} W^{n-p} e^2(p)$$

leads to the set of equations

$$\frac{\partial J(n)}{\partial h_{ij}(n)} = 2 \sum_{p=1}^{n} W^{n-p} [y(p) - H^t(n)X(p)] x_i(p-j) = 0 \tag{7.154}$$

with $1 \leqslant i \leqslant K$, $0 \leqslant j \leqslant N-1$. Hence the optimum coefficient vector at time n is

$$H(n) = R_{KN}^{-1}(n) r_{KN}(n) \tag{7.155}$$

with

$$R_{KN}(n) = \sum_{p=1}^{n} W^{n-p} X(p) X^t(p)$$

$$r_{KN}(n) = \sum_{p=1}^{n} W^{n-p} y(p) X(p)$$

The matrix $R_{KN}(n)$ is a cross-correlation matrix estimation. The updating recursion for the coefficient vector takes the form

$$H(n+1) = H(n) + R_{KN}^{-1}(n+1)X(n+1)e(n+1) \tag{7.156}$$

and the adaptation gain

$$G_K(n) = R_{KN}^{-1}(n)X(n) \tag{7.157}$$

is a KN-element vector, which can be updated through a procedure similar to that of Section 6.4.

The connecting matrix $R_{KN1}(n+1)$ is defined by

$$R_{KN1}(n+1) = \sum_{p=1}^{n+1} W^{n+1-p} \begin{bmatrix} \chi(p) \\ X(p-1) \end{bmatrix} [\chi^t(p), X^t(p-1)] \tag{7.158}$$

and can be partitioned as

$$R_{KN1}(n+1) = \begin{bmatrix} \sum_{p=1}^{n+1} W^{n+1-p}\chi(p)\chi^t(p) & [r_{KN}^a(n+1)]^t \\ r_{KN}^a(n+1) & R_{KN}(n) \end{bmatrix} \tag{7.159}$$

where $r_{KN}^a(n+1)$ is the $KN \times K$ cross-correlation matrix

$$r_{KN}^a(n+1) = \sum_{p=1}^{n+1} W^{n+1-p} X(p-1)\chi^t(p) \tag{7.160}$$

From the alternative definition

$$R_{KN1}(n+1) = \sum_{p=1}^{n+1} W^{n+1-p} \begin{bmatrix} X(p) \\ \chi(p-N) \end{bmatrix} [X^t(p), \chi^t(p-N)] \tag{7.161}$$

a second partitioning is obtained:

$$R_{KN1}(n+1) = \begin{bmatrix} R_{KN}(n+1) & r_{KN}^b(n+1) \\ [r_{KN}^b(n+1)]^t & \sum_{p=1}^{n+1} W^{n+1-p}\chi(n+1-N)\chi^t(n+1-N) \end{bmatrix} \tag{7.162}$$

where $r_{KN}^b(n+1)$ is the $KN \times K$ matrix

$$r_{KN}^b(n+1) = \sum_{p=1}^{n+1} W^{n+1-p} X(p)\chi^t(p-N) \tag{7.163}$$

The fast algorithms use the prediction equations. The forward prediction error takes the form of a K-element vector

$$e_{Ka}(n+1) = \chi(n+1) - A_K^t(n)X(n) \tag{7.164}$$

where the prediction coefficients form a $KN \times K$ matrix, which is computed to minimize the prediction error energy, defined by

$$E_a(n) = \sum_{p=1}^{n} W^{n-p} e_{Ka}^t(p) e_{Ka}(p) = \text{trace}[E_{Ka}(n)] \tag{7.165}$$

with the quadratic error energy matrix defined by

$$E_{Ka}(n) = \sum_{p=1}^{n} W^{n-p} e_{Ka}(p) e_{Ka}^t(p) \tag{7.166}$$

The minimization process yields

$$A_K(n+1) = R_{KN}^{-1}(n) r_{KN}^a(n+1) \tag{7.167}$$

The forward prediction coefficients, updated by

$$A_K(n+1) = A_K(n) + G_K(n) e_{Ka}^t(n+1) \tag{7.168}$$

are used to derive the a posteriori prediction error $\varepsilon_{Ka}(n+1)$, also a K-element vector, by

$$\varepsilon_{Ka}(n+1) = \chi(n+1) - A_K^t(n+1)X(n) \tag{7.169}$$

The quadratic error energy matrix can also be expressed by

$$E_{Ka}(n+1) = \sum_{p=1}^{n+1} W^{n+1-p} \chi(p)\chi^t(p) - A_K^t(n+1) r_{KN}^a(n+1) \tag{7.170}$$

which, by the same approach as in Section 6.3, yields the updating recursion

$$E_{Ka}(n+1) = W E_{Ka}(n) + e_{Ka}(n+1) \varepsilon_{Ka}^t(n+1) \tag{7.171}$$

The a priori adaptation gain $G_K(n)$ can be updated by reproducing the developments given in Section 6.4 and using the two partitioning equations (7.159) and (7.162) for $R_{KN1}(n+1)$. The fast algorithm based on a priori errors is given in Figure 7.6.

If the predictor order N is sufficient, the prediction error elements, in the steady-state phase, approach white noise signals and the matrix $E_{Ka}(n)$ approaches a diagonal matrix. Its initial value can be taken as a diagonal matrix

$$E_{Ka}(0) = E_0 I_K \tag{7.172}$$

where E_0 is a positive scalar; all other initial values can be zero.

A stabilization constant, as in Section 6.8, can be introduced by modifying recursion (7.171) as follows:

ALGORITHM F.L.S. 1-K

AVAILABLE AT TIME n:

COEFFICIENTS OF ADAPTIVE FILTER	: $H(n)$
FORWARD PREDICTION MATRIX	: $A_K(n)$
BACKWARD PREDICTION MATRIX	: $B_K(n)$
DATA VECTOR	: $X(n)$
ADAPTATION GAIN	: $G_K(n)$
QUADRATIC ERROR MATRIX	: $E_{Ka}(n)$
WEIGHTING FACTOR	: W

NEW DATA AT TIME n:

Input signal : $\chi(n+1)$; Reference : $y(n+1)$

ADAPTATION GAIN UPDATING:

$$e_{Ka}(n+1) = \chi(n+1) - A_K^t(n)\,X(n)$$

$$A_K(n+1) = A_K(n) + G_K(n)\,e_{Ka}^t(n+1)$$

$$\varepsilon_{Ka}(n+1) = \chi(n+1) - A_K^t(n+1)\,X(n)$$

$$E_{Ka}(n+1) = W\,E_{Ka}(n) + e_{Ka}(n+1)\,\varepsilon_{Ka}^t(n+1)$$

$$G_{K1}(n+1) = \begin{bmatrix} 0 \\ \\ G_K(n) \end{bmatrix} + \begin{bmatrix} I_K \\ \\ -A_K(n+1) \end{bmatrix} E_{Ka}^{-1}(n+1)\,\varepsilon_{Ka}(n+1) = \begin{bmatrix} M_K(n+1) \\ \\ m_K(n+1) \end{bmatrix}$$

$$e_{Kb}(n+1) = \chi(n+1-N) - B_K^t(n)\,X(n+1)$$

$$G_K(n+1) = \frac{1}{1 - e_{Kb}^t(n+1)\,m_K(n+1)}\left(M_K(n+1) + B_K(n)m_K(n+1)\right)$$

$$B_K(n+1) = B_K(n) + G_K(n+1)\,e_{Kb}^t(n+1)$$

ADAPTIVE FILTER :

$$e(n+1) = y(n+1) - H^t(n)\,X(n+1)$$

$$H(n+1) = H(n) + G_K(n+1)e(n+1)$$

FIG. 7.6 FLS algorithm for multidimensional input signals.

$$E_{Ka}(n+1) = WE_{Ka}(n) + e_{Ka}(n+1)\varepsilon^t_{Ka}(n+1) + CI_K \tag{7.173}$$

where C is a positive scalar.

The matrix inversion in Figure 7.6 is carried out, with the help of the matrix inversion lemma (6.26) of Chapter 6 by updating the inverse quadratic error matrix:

$$E^{-1}_{Ka}(n+1) = W^{-1}\left[E^{-1}_{Ka}(n) - \frac{E^{-1}_{Ka}(n)e_{Ka}(n+1)\varepsilon^t_{Ka}(n+1)E^{-1}_{Ka}(n)}{W + \varepsilon^t_{Ka}(n+1)E^{-1}_{Ka}(n)e_{Ka}(n+1)}\right] \tag{7.174}$$

The computational complexity of that expression amounts to $3K^2 + 2K$ multiplications and one division or inverse calculation.

Note that if $N = 1$, which means that there is no convolution on the input data, then $E^{-1}_{Ka}(n)$ is just the inverse cross-correlation matrix $R^{-1}_{KN}(n)$, and it is updated directly from the input signal data as in conventional RLS techniques.

For the operations related to the filter order N, the algorithm presented in Figure 7.2 requires $7K^2N + KN$ multiplications for the adaptation gain and $2KN$ multiplications for the filter section. The FORTRAN program is given in Annex 7.3.

The ratio $\varphi(n)$ of a posteriori to a priori prediction errors is still a scalar, because

$$\varepsilon_{aK}(n+1) = e_{aK}(n+1)[1 - G^t_K(n)X(n)] \tag{7.175}$$

Therefore it can still serve to check the correct operation of the multidimensional algorithms. Moreover, it allows us to extend to multidimensional input signals the algorithms based on all prediction errors.

7.10. M-D ALGORITHM BASED ON ALL PREDICTION ERRORS

An alternative adaptation gain vector, which leads to exploiting a priori and a posteriori prediction errors is defined by

$$G'_K(n+1) = R^{-1}_{KN}(n)X(n+1) = \frac{G_K(n+1)W}{\varphi(n+1)} \tag{7.176}$$

The updating procedure uses the ratio of a posteriori to a priori prediction errors, under the form of the scalar $\alpha(n)$ defined by

$$\alpha(n) = W + X^t(n)R^{-1}_{KN}(n-1)X(n) = \frac{W}{\varphi(n)} \tag{7.177}$$

The computational organization of the corresponding algorithm is shown in Figure 7.7. Indeed, it follows closely the sequence of operations already given in Figure 6.5, but scalars and vectors have been replaced by vectors and matrices when appropriate.

ALGORITHM F.L.S.2-K

AVAILABLE AT TIME n

COEFFICIENTS OF ADAPTIVE FILTER	$H(n)$
FORWARD PREDICTION MATRIX	$A_K(n)$
BACKWARD PREDICTION MATRIX	$B_K(n)$
DATA VECTOR	$X(n)$
ADAPTATION GAIN	$G'_K(n)$
QUADRATIC ERROR MATRICES	$E_{Ka}(n), E_{Kb}(n)$
PREDICTION ERROR RATIO	$\alpha(n)$
WEIGHTING FACTOR	W

NEW DATA AT TIME n:

Input signal : $x(n+1)$; Reference : $y(n+1)$

ADAPTATION GAIN UPDATING:

$$e_{Ka}(n+1) = x(n+1) - A_K^t(n)X(n)$$

$$A_K(n+1) = A_K(n) + G'_K(n)e_{Ka}^t(n+1)/\alpha(n)$$

$$E_{Ka}(n+1) = (E_{Ka}(n) + e_{Ka}(n+1)e_{Ka}^t(n+1)/\alpha(n))W$$

$$G'_{K1}(n+1) = \begin{bmatrix} 0 \\ \\ G_K(n) \end{bmatrix} + \begin{bmatrix} I_K \\ \\ -A_K(n) \end{bmatrix} E_{Ka}^{-1}(n+1)e_{Ka}(n+1) = \begin{bmatrix} M_K(n+1) \\ \\ m_K(n+1) \end{bmatrix}$$

$$e_{Kb}(n+1) = x(n+1-N) - B_K^t(n)X(n+1)$$

$$G'_K(n+1) = M_K(n+1) + B_K(n)m_K(n+1)$$

$$\alpha_1(n+1) = \alpha(n) + e_{Ka}^t(n+1)E_{Ka}^{-1}(n)e_{Ka}(n+1)$$

$$\alpha(n+1) = \alpha_1(n+1) - e_{Kb}^t(n+1)m_K(n+1)$$

$$E_{Kb}(n+1) = (E_{Kb}(n) + e_{Kb}(n+1)e_{Kb}^t(n+1)/\alpha(n+1))W$$

$$B_K(n+1) = B_K(n) + G'_K(n)e_{Kb}^t(n+1)/\alpha(n+1)$$

ADAPTIVE FILTER:

$$e(n+1) = y(n+1) - H^t(n)X(n+1)$$

$$H(n+1) = H(n) + G'_K(n+1)e(n+1)/\alpha(n+1)$$

FIG. 7.7 Algorithm based on all prediction errors for M-D input signals.

The operations related to the filter order N correspond to $6K^2N$ multiplications for the gain and $2KN$ multiplications for the filter section.

In the above procedure, the backward a priori prediction error vector $e_{Kb}(n+1)$ can also be calculated directly by

$$e_{Kb}(n+1) = E_{Kb}(n)m_K(n+1) \tag{7.178}$$

Again that provides means to control the roundoff error accumulation, through updating the backward prediction coefficients, as in (6.139) of Chapter 6 by

$$B_K(n+1) = B_K(n) + G'_K(n+1) \times [e_{Kb}(n+1) + e_{Kb}(n+1) - E_{Kb}(n)m_K(n+1)]/\alpha(n+1) \tag{7.179}$$

Up to now, the reference signal has been assumed to be a scalar sequence. The adaptation gain calculations which have been carried out only depend on the input signals, and they are valid for multidimensional reference signals as well. The case of K-dimensional (K-D) input and L-dimensional (L-D) reference signals is depicted in Figure 7.8. The only modifications with respect to the previous algorithms concern the filter section. The L-element reference vector $Y_L(n)$ is used to derive the output error vector $e_L(n)$ from the input and the $KN \times L$ coefficient matrix $H_L(n)$ as follows:

$$e_L(n+1) = Y_L(n+1) - H_L^t(n)X(n+1) \tag{7.180}$$

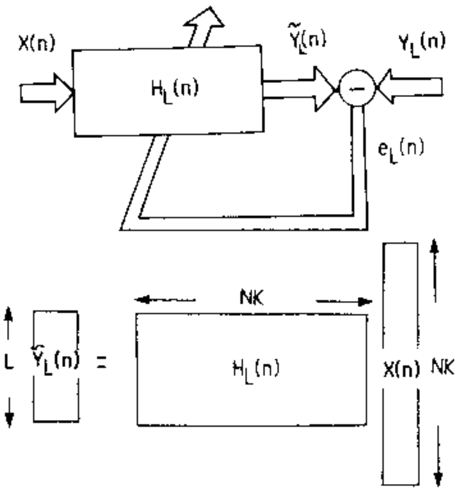


FIG. 7.8 Adaptive filter with M-D input and reference signals.

The coefficient matrix is updated by

$$H_L(n+1) = H_L(n) + \frac{G'_K(n+1)e^t_L(n+1)}{\alpha(n+1)} \tag{7.181}$$

The associated complexity amounts to $2NKL + L$ multiplications.

The developments given in Chapter 6 and the preceding sections have illustrated the flexibility of the procedures used to derive fast algorithms. Another example is provided by filters of nonuniform length [8].

7.11. FILTERS OF NONUNIFORM LENGTH

In practice it is desirable to tailor algorithms to meet the specific needs of applications. The input sequences may be fed to filters with different lengths, and adjusting the fast algorithms accordingly can provide substantial savings.

Assume that the K filters in Figure 7.5 have lengths $N_i (1 \leqslant i \leqslant K)$. The data vector $X(n)$ can be rearranged as follows:

$$X^t(n) = [X^t_1(n), X^t_2(n), \dots, X^t_K(n)] \tag{7.182}$$

where

$$X^t_i(n) = [x_i(n), x_i(n-1), \dots, x_i(n+1-N_i)]$$

The number of elements ΣN is

$$\Sigma N = \sum_{i=1}^{K} N_i \tag{7.183}$$

The connecting $(\Sigma N + K)(\Sigma N + K)$ matrix $R_{\Sigma N1}(n+1)$, defined by

$$R_{\Sigma N1}(n+1) = \sum_{p=1}^{n+1} W^{n+1-p} \begin{bmatrix} x_1(n+1) \\ X_1(n) \\ \vdots \\ x_K(n+1) \\ X_K(n) \end{bmatrix} \begin{bmatrix} x_1(n+1) \\ X_1(n) \\ \vdots \\ x_K(n+1) \\ X_K(n) \end{bmatrix}^t$$

can again be partitioned in two different manners and provide the gain updating operations. The algorithms obtained are those shown in Figures 7.6 and 7.7. The only difference is that the prediction coefficient $\Sigma N \times K$ matrices are organized differently to accommodate the rearrangement of the data vector $X(n)$.

A typical case where filter dimensions can be different is pole-zero modeling.

7.12. FLS POLE-ZERO MODELING

Pole-zero modeling techniques are used in control for parametric system identification.

An adaptive filter with zeros and poles can be viewed as a filter with 2-D input data and 1-D reference signal if the equation error approach is chosen. The filter defined by

$$\tilde{y}(n+1) = A^t(n)X(n+1) + B^t(n)\tilde{Y}(n) \tag{7.184}$$

is equivalent to a filter as in Figure 7.5 with input signal vector

$$\chi(n+1) = \begin{bmatrix} x(n+1) \\ \tilde{y}(n) \end{bmatrix} \tag{7.185}$$

For example, let us consider the pole-zero modeling of a system with output $y(n)$ when fed with $x(n)$. An approach which ensures stability is shown in Figure 4.12(b). A 2-D FLS algorithm can be used to compute the model coefficients with input signal vector

$$\chi(n+1) = \begin{bmatrix} x(n+1) \\ y(n) \end{bmatrix} \tag{7.186}$$

However, as pointed out in Section 4.11, that equation error type of approach is biased when noise is added to the reference signal. It is preferable to use the output error approach in Figure 4.12(a). But stability can only be guaranteed if the smoothing filter with z-transfer function $C(z)$ satisfying strictly positive real (SPR) condition (4.149) in Chapter 4 is introduced on the error signal.

An efficient approach to pole-zero modeling is obtained by incorporating the smoothing filter in the LS process [9]. A 3-D FLS algorithm is employed, and the corresponding diagram is shown in Figure 7.9. The output error signal $f(n)$ used in the adaptation process is

$$f(n) = y(n) - [u_1(n) + u_2(n) + u_3(n)] \tag{7.187}$$

where $u_1(n)$, $u_2(n)$, and $u_3(n)$ are the outputs of the three filters fed by $\tilde{y}(n)$, $x(n)$, and $e(n) = y(n) - \tilde{y}(n)$, respectively. The cost function is

$$J_3(n) = \sum_{p=1}^{n} W^{n-p} f^2(p) \tag{7.188}$$

Let the unknown system output be

$$y(n) = \sum_{i=0}^{N} a_i x(n-i) + \sum_{i=1}^{N} b_i y(n-i) \tag{7.189}$$

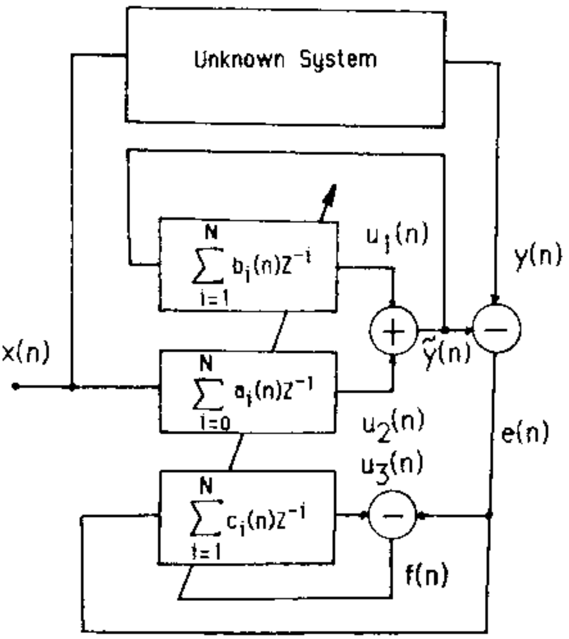

Unknown System
$\sum_{i=1}^{N} b_i(n)Z^{-i}$
$u_1(n)$
$y(n)$
$x(n)$
$\sum_{i=0}^{N} a_i(n)Z^{-1}$
$\tilde{y}(n)$
$u_2(n)$
$e(n)$
$u_3(n)$
$\sum_{i=1}^{N} c_i(n)Z^{-i}$
$f(n)$

FIG. 7.9 Adaptive pole-zero modeling with a 3-D FLS algorithm.

or

$$y(n) = \sum_{i=0}^{N} a_i x(n-i) + \sum_{i=1}^{N} b_i \tilde{y}(n-i) + \sum_{i=1}^{N} b_i e(n-i) \tag{7.190}$$

From (7.187), the error signal is zero in the steady state if

$$a_i(\infty) = a_i, \quad b_i(\infty) = b_i, \quad c_i(\infty) = b_i, \quad 1 \leqslant i \leqslant N$$

Now, assume that a white noise sequence $\eta(n)$ with power σ_η^2 is added to the system output. The cost function to be minimized becomes

$$J_{3\eta}(n) = \sum_{p=1}^{n} W^{n-p}\left[f(p) + \eta(p) - \sum_{i=1}^{N} c_i(n)\eta(p-i)\right]^2 \tag{7.191}$$

which, for sufficiently large n can be approximated by

$$J_{3\eta}(n) \approx \sum_{p=1}^{n} W^{n-p}\left[f^2(p) + \sigma_\eta^2\left[1 + \sum_{i=1}^{N} c_i^2(n)\right]\right] \tag{7.192}$$

The steady-state solution is

$$a_i(\infty) = a_i, \quad b_i(\infty) = b_i, \quad c_i(\infty) = 0, \quad 1 \leqslant i \leqslant N$$

Finally, the correct system identification is achieved, in the presence of noise or not. The smoothing filter coefficients vanish on the long run when additive noise is present. An illustration is provided by the following example.

Example [9]

Let the transfer function of the unknown system be

$$H(z) = \frac{0.05 + 0.1z^{-1} + 0.075z^{-2}}{1 - 0.96z^{-1} + 0.94z^{-2}}$$

and let the input be the first-order AR signal

$$x(n) = e_0(n) + 0.8x(n-1)$$

where $e_0(n)$ is a white Gaussian sequence.

The system gain G_S defined by

$$G_S = 10 \log \frac{E[y^2(n)]}{E[e^2(n)]}$$

is shown in Figure 7.10(a) versus time. The ratio of the system output signal to additive noise power is 30 dB. For comparison the gain obtained with the equation error or series-parallel approach is also given. In accordance with expression (4.154) in Chapter 4, it is bounded by the SNR. The smoothing filter coefficients are shown in Figure 7.10(b). They first reach the b_i values $(i = 1, 2)$ and decay to zero after.

The 3-D parallel approach requires approximately twice the number of multiplications of the 2-D series-parallel approach.

7.13. MULTIRATE ADAPTIVE FILTERS

The sampling frequencies of input and reference signals can be different. In the sample rate reduction case, depicted in Figure 7.11, the input and reference sampling frequencies are f_S and $f_{S/K}$, respectively. The input signal sequence is used to form K sequences with sample rate $f_{S/K}$ which are fed to K filters with coefficient vectors $H_i(n)(0 \leqslant i \leqslant K-1)$. The cost function to be minimized in the adaptive filter, $J_{SRR}(Kn)$, is

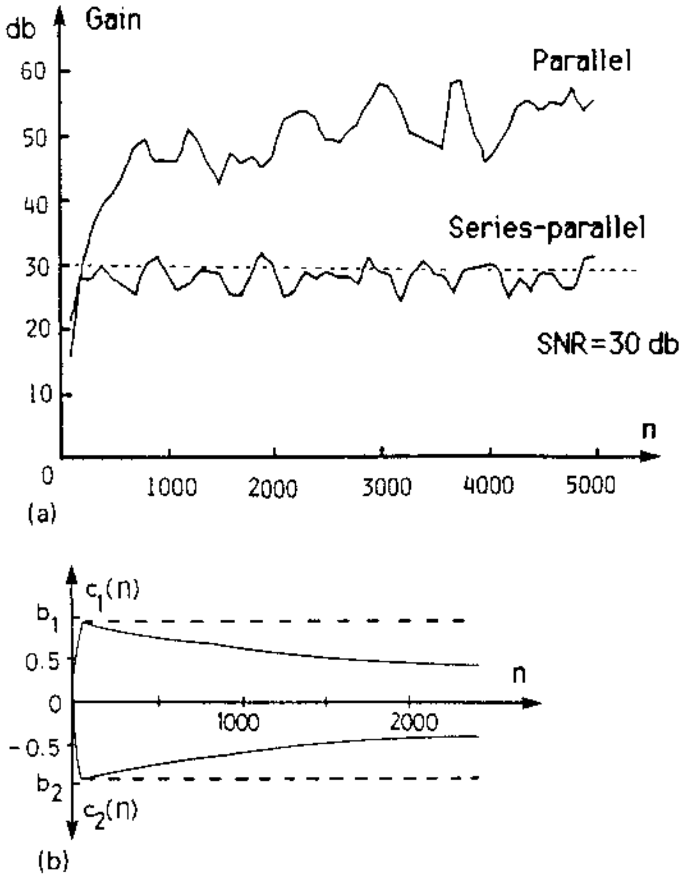


FIG. 7.10 Pole-zero modeling of an unknown system: (a) System gain in FLS identification. (b) Smoothing filter coefficients.

$$J_{SRR}(Kn) = \sum_{p=1}^{n} W^{n-p}[y(Kp) - H^t(Kp)X(Kp)]^2 \tag{7.193}$$

The data vector $X(Kn)$ is the vector of the NK most recent input values. The input may be considered as consisting of K different signals, and the algorithms presented in the preceding sections can be applied. The corresponding calculations are carried out at the frequency fs/k.

An alternative approach takes advantage of the sequential presentation of the input samples and is presented for the particular and important case where $k = 2$.

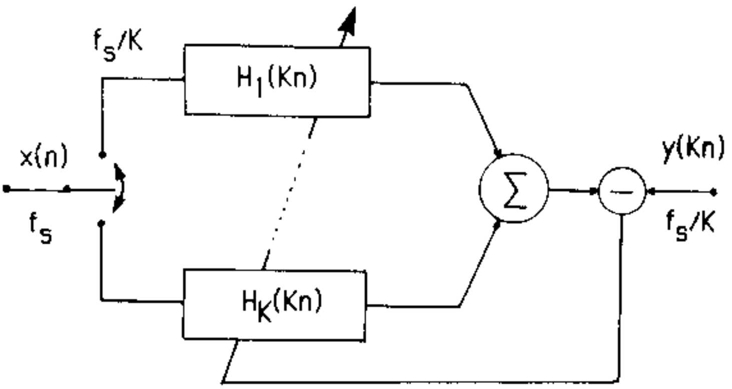


FIG. 7.11 Sample rate reduction adaptive filter.

It is assumed that the input sequence is seen as two interleaved sequences $x_1(n)$ and $x_2(n)$ and two input data vectors, $X_{2N}(n)$ and $X_{1,2N}(n+1)$ are defined as follows:

$$\begin{array}{c} \longleftarrow\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\, X_{2N}(n) \,-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!\longrightarrow \\ x_2(n+1)x_1(n+1)x_2(n)x_1(n) \qquad x_2(n+1-N)x_1(n+1-N) \\ \longleftarrow\!\!-\!\!-\!\!-\!\!-\, X_{1,2N}(n+1) \,-\!\!-\!\!-\!\!-\!\!\longrightarrow \end{array}$$

or in vector form

$$X_{2N}(n) = \begin{bmatrix} x_2(n) \\ x_1(n) \\ \vdots \\ x_1(n+1-N) \end{bmatrix}, \qquad X_{1,2N}(n+1) = \begin{bmatrix} x_1(n+1) \\ x_2(n) \\ \vdots \\ x_2(n+1-N) \end{bmatrix}$$

The cost function is

$$J(n) = \sum_{p=1}^{n} W^{n-p}[y(p) - H_{2N}(n)X_{2N}(p)]^2 \tag{7.194}$$

where $H_{2N}(n)$ is the coefficient vector with $2N$ elements. The multirate adaptive filter section consists of the two following equations:

$$\begin{aligned} e(n+1) &= y(n+1) - H_{2N}^t(n)X_{2N}(n+1) \\ H_{2N}(n+1) &= H_{2N}(n) + G_{2N}(n+1)e(n+1) \end{aligned} \tag{7.195}$$

The adaptation gain vector $G_{2N}(n)$ is itself defined from the AC matrix

$$R_{2N}(n) = \sum_{p=1}^{n} W^{n-p} X_{2N}(p) X_{2N}^{t}(p) \tag{7.196}$$

as follows

$$G_{2N}(n) = R_{2N}^{-1}(n) X_{2N}(n) \tag{7.197}$$

In the multirate fast least squares algorithm, the adaptation gain vector is updated through linear prediction. A first error energy can be defined by

$$E_{1a}(n) = \sum_{p=1}^{n} W^{n-p} [x_1(p) - A_{1,2N}^{t}(n) X_{2N}(p-1)]^2 \tag{7.198}$$

and it leads to the linear prediction matrix equation

$$R_{2N+1}(n+1) \begin{bmatrix} 1 \\ -A_{1,2N}(n+1) \end{bmatrix} = \begin{bmatrix} E_{1a}(n+1) \\ 0 \end{bmatrix} \tag{7.199}$$

where the extended $(2N+1) \times (2N+1)$ matrix is

$$R_{2N+1}(n+1) = \sum_{p=1}^{n+1} W^{n+1-p} \begin{bmatrix} x_1(p) \\ X_{2N}(p-1) \end{bmatrix} [x_1(p), X_{2N}^{t}(p-1)] \tag{7.200}$$

Now, the procedure of Chapter 6 can be applied to compute an extended adaptation gain $G_{1,2N+1}(n+1)$ from forward linear prediction and an updated adaptation gain $G_{1,2N}(n+1)$ from backward linear prediction. The same procedure can be repeated with $x_2(n+1)$ as the new data, leading to another extended adaptation gain $G_{2,2N+1}(n+1)$ and, finally, to the desired updated gain $G_{2N}(n+1)$. The complete computational organization is given in Figure 7.12; in fact, the one-dimensional FLS algorithm is run twice in the prediction section.

The approach can be extended to multidimensional, or multichannel, inputs with K elementary signals. It is sufficient to run K times the prediction section for 1-D signals, and use the proper prediction and adaptation gain vectors each time. There is no gain in computational simplicity with respect to the algorithms presented in Sections 7.9 and 7.10, but the scheme is elegant and easy to implement, particularly in the context of multirate filtering.

As concerns the case of increasing sample rate, it is shown in Figure 7.13. It corresponds to scalar input and multidimensional reference signals.

It is much more economical in terms of computational complexity than the sample rate reduction, because the adaptation gain is computed once for the K interpolating filters. All the calculations are again carried out at frequency $f_{S/K}$, the reference sequence being split into K sequences at that

Available at time n :

Coefficients of the 2-D adaptive filter : $H_{2N}(n)$

" forward predictors : $A_{1,2N}(n)$; $A_{2,2N}(n)$

backward predictors : $B_{1,2N}(n)$; $B_{2,2N}(n)$

Data vector : $X_{2N}(n)$

Adaptation gains : $G_{2N}(n)$; $G_{1,2N}(n)$

Forward prediction error energies : $E_{1a}(n)$; $E_{2a}(n)$

Weighting factor : W

New data at time n :

Input signals : $x_1(n+1)$, $x_2(n+1)$; reference : $y(n+1)$

Prediction Section

$$e_{1a}(n+1) = x_1(n+1) - A_{1,2N}^t(n)\, X_{2N}(n)$$
$$A_{1,2N}(n+1) = A_{1,2N}(n) + G_{2N}(n)\, e_{1a}(n+1)$$
$$\varepsilon_{1a}(n+1) = x_1(n+1) - A_{1,2N}^t(n+1)\, X_{2N}(n)$$
$$E_{1a}(n+1) = W\, E_{1a}(n) + e_{1a}(n+1)\, \varepsilon_{1a}(n+1)$$

$$G_{1,2N+1}(n+1) = \begin{bmatrix} 0 \\ G_{2N}(n) \end{bmatrix} + \frac{\varepsilon_{1a}(n+1)}{E_{1a}(n+1)} \begin{bmatrix} 1 \\ -A_{1,2N}(n+1) \end{bmatrix} = \begin{bmatrix} M_1(n+1) \\ m_1(n+1) \end{bmatrix}$$

$$e_{1b}(n+1) = x_1(n+1-N) - B_{1,2N}^t(n)\, X_{1,2N}(n+1)$$
$$G_{1,2N}(n+1) = \frac{1}{1-m_1(n+1)e_{1b}(n+1)} \left[\ M_1(n+1) + m_1(n+1)\, B_{1,2N}(n)\ \right]$$
$$B_{1,2N}(n+1) = B_{1,2N}(n) + G_{1,2N}(n+1)\, e_{1b}(n+1)$$

$$e_{2a}(n+1) = x_2 n+1) - A_{2,2N}^t(n+1)\, X_{1,2N}(n+1)$$
$$A_{2,2N}(n+1) = A_{2,2N}(n) + G_{1,2N}(n+1)\, e_{2a}(n+1)$$
$$\varepsilon_{2a}(n+1) = x_2(n+1) - A_{2,2N}^t(n+1)\, X_{1,2N}(n+1)$$
$$E_{2a}(n+1) = W\, E_{2a}(n) + e_{2a}(n+1)\, \varepsilon_{2a}(n+1)$$

$$G_{2,2N+1}(n+1) = \begin{bmatrix} 0 \\ G_{1,2N}(n+1) \end{bmatrix} + \frac{\varepsilon_{2a}(n+1)}{E_{2a}(n+1)} \begin{bmatrix} 1 \\ -A_{2,2N}(n+1) \end{bmatrix} = \begin{bmatrix} M_2(n+1) \\ m_2(n+1) \end{bmatrix}$$

$$e_{2b}(n+1) = x_2(n+1-N) - B_{2,2N}^t(n)\, X_{2N}(n+1)$$
$$G_{2N}(n+1) = \frac{1}{1-m_2(n+1)\, e_{2b}(n+1)} \left[\ M_2(n+1) - m_2(n+1)\, B_{2,2N}(n)\ \right]$$
$$B_{2,2N}(n+1) = B_{2,2N}(n) + G_{2n}(n+1)\, e_{2b}(n+1)$$

Filter section

$$e(n+1) = y(n+1) - H_{2N}^t(n)\, X(n+1)$$
$$H_{2N}(n+1) = H_{2n}(n) + G_{2N}(n+1)\, e(n+1)$$

FIG. 7.12 The algorithm FLS 2-D/1-D.

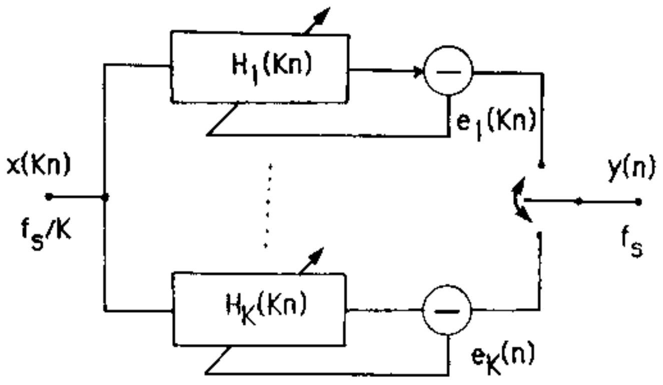


FIG. 7.13 Sample rate increase adaptive filter.

frequency. The system boils down to K different adaptive filters with the same input.

In signal processing, multirate aspects are often linked with DFT applications and filter banks, which correspond to frequency domain conversions.

7.14. FREQUENCY DOMAIN ADAPTIVE FILTERS

The power conservation principle states that the power of a signal in the time domain equals the sum of the powers of its frequency components. Thus, the LS techniques and adaptive methods worked out for time data can be transposed in the frequency domain.

The principle of a frequency domain adaptive filter (FDAF) is depicted in Figure 7.14. The N-point DFTs of the input and reference signals are computed. The complex input data obtained are multiplied by complex coefficients and subtracted from the reference to produce the output error used to adjust the coefficients.

At first glance, the approach may look complicated and farfetched. However, there are two motivations [10, 11]. First, from a theoretical point of view, the DFT computer is actually a filter bank which performs some orthogonalization of the data; thus, an order N adaptive filter becomes a set of N separate order 1 filters. Second, from a practical standpoint, the efficient FFT algorithms to compute the DFT of blocks of N data, particularly for large N, can potentially produce substantial savings in computation speed, because the DFT output sampling frequency can be reduced by the factor N.

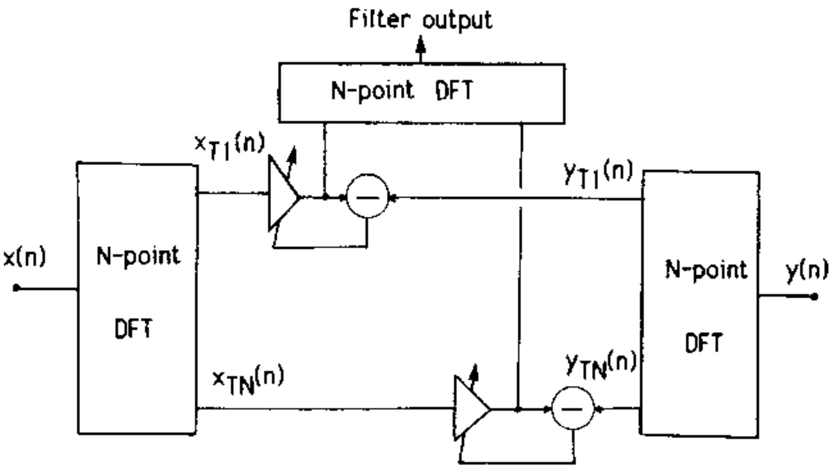


FIG. 7.14 FDAF structure.

Assuming N separate complex filters and combining the results of Sections 6.1 and 7.8, we obtain the LS solution for the coefficients

$$h_i(n) = \frac{\sum_{p=1}^{n} W^{n-p} \bar{y}_{Ti}(p) x_{Ti}(p)}{\sum_{p=1}^{n} W^{n-p} x_{Ti}(p) \bar{x}_{Ti}(p)}, \qquad 0 \leqslant i \leqslant N-1 \tag{7.201}$$

where $x_{Ti}(n)$ and $y_{Ti}(n)$ are the transformed sequences.

For sufficiently large n, the denominator of that equation is an estimate of the input power spectrum, and the numerator is an estimate of the cross-power spectrum between input and reference signals. Overall the FDAF is an approximation of the optimal Wiener filter, itself the frequency domain counterpart of the time domain filter associated with the normal equations. Note that the optimal method along these lines, in case of stationary signals, would be to use the eigentransform of Section 3.12.

The updating equations associated with (7.201) are

$$h_i(n+1) = h_i(n) + r_i^{-1}(n+1) x_{Ti}(n+1) \times [y_{Ti}(n+1) - \bar{h}_i(n) x_{Ti}(n+1)] \tag{7.202}$$

and

$$r_i(n+1) = W r_i(n) + x_{Ti}(n+1) \bar{x}_{Ti}(n+1) \tag{7.203}$$

The FFT algorithms need about $(N/2)\log_2(N/2)$ complex multiplications each, which have to be added to the N order 1 adaptive filter operations. Altogether savings can be significant for large N, with respect to FLS algorithms.

The LMS algorithm can also be used to update the coefficients, and the results given in Chapter 4 can serve to assess complexity and performance.

It must be pointed out that the sample rate reduction by N at the DFT output can alter the adaptive filter operation, due to the circular convolution effects. A scheme without sample rate reduction is shown in Figure 7.15, where a single orthogonal transform is used. If the first row of the transform matrix consists of 1's only, the inverse transformed data are obtained by just summing the transformed data [12]. Note also that the complex operations are avoided if a real transform, such as the DCT [equations (3.160) in Chapter 3], is used.

A general observation about the performance of frequency domain adaptive filters is that they can yield poor results in the presence of nonstationary signals, because the subband decomposition they include can enhance the nonstationary character of the signals.

7.15. SECOND-ORDER NONLINEAR FILTERS

A nonlinear second-order Volterra filter consists of a linear section and a quadratic section in parallel, when the input signal is Gaussian, as mentioned in Section 4.16.

In this structure, FLS algorithms can be used to update the coefficients of the linear section in a straightforward manner. As concerns the quadratic

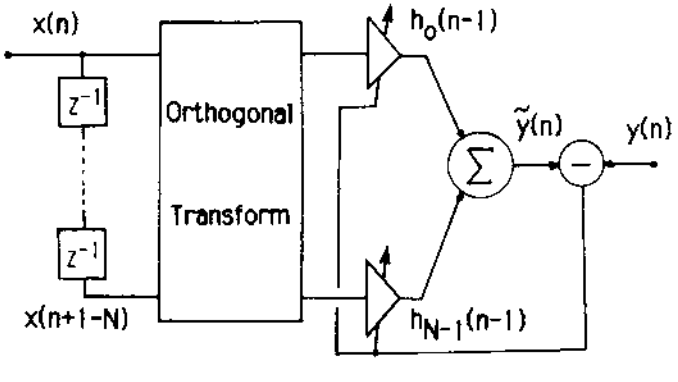


FIG. 7.15 FDAF with a single orthogonal transform.

section, its introduction in the least squares procedure brings a significant increase of the computational complexity. However, it is possible to introduce a simplified iterative procedure, based on the adaptation gain of the linear section [13].

Let us consider the system to be identified in Figure 7.16. The input signal $x(n)$ is assumed to be a white noise, as well as the measurement noise $b(n)$, which is uncorrelated with $x(n)$ and has the power σ_b^2. The cost function at time n is

$$J(n) = \sum_{p=1}^{n} [y(p) - X^t(p)H(n) - X^t(p)M(n)X(p)]^2 \tag{7.204}$$

Due to the Gaussian hypothesis, the third-order moments vanish, and setting to zero the derivatives yields for the linear section with N coefficients

$$\sum_{p=1}^{n} y(p)X(p) - \left[\sum_{p=1}^{n} X(p)X^t(p)\right] H(n) = 0 \tag{7.205}$$

and for the quadratic section with N^2 coefficients

$$\sum_{p=1}^{n} X(p)X^t(p)y(p) - \sum_{p=1}^{n} X(p)X^t(p)M(n)X(p)X^t(p) = 0 \tag{7.206}$$

Since $x(n)$ is a white noise, the coefficients are given by

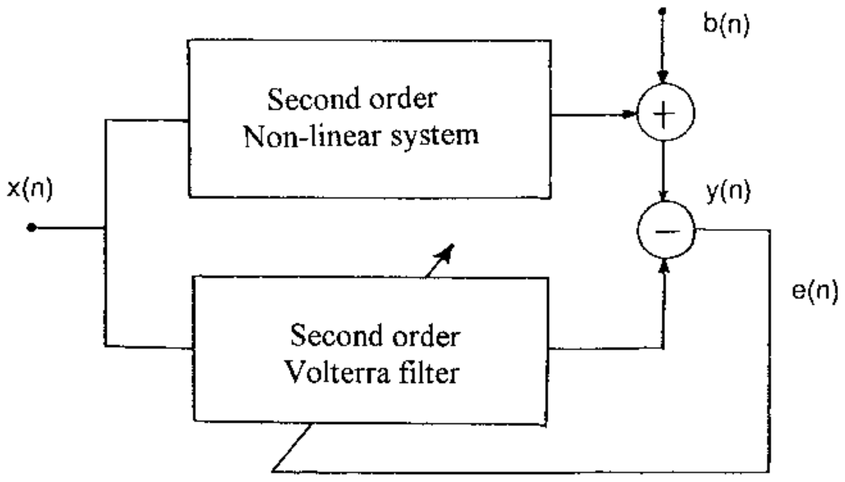


FIG. 7.16 Identification of a second-order nonlinear system.

$$M(n) = R_N^{-1}(n)\left[\sum_{p=1}^{n} X(p)X^t(p)y(p)\right]R_N^{-1}(n) \tag{7.207}$$

The above expressions for $H(n)$ and $M(n)$ are the counterparts of equations (4.162) in the least squares context. Therefore, the coefficients satisfy the following recursion

$$M(n+1) = M(n) + G(n+1)e(n+1)G^t(n+1) \tag{7.208}$$

with

$$e(n+1) = y(n+1) - X^t(n+1)H(n) - X(n+1)M(n)X^t(n+1) \tag{7.209}$$

The same derivation as in Section 4.16 leads to the following expression for the output error power:

$$E[e^2(n+1)] \approx \sigma_b^2\left(1 + \frac{N}{n} + \frac{2N}{n^2}\right) \tag{7.210}$$

where the terms N/n and $2N/n^2$ correspond to the linear and quadratic terms respectively. Obviously, the speed of convergence of the nonlinear section is limited by the speed of convergence of the linear section.

The approach can be extended to cost functions with a weighting factor. In any case, the performance can be significantly enhanced, compared to what the gradient technique achieves.

7.16. UNIFIED GENERAL VIEW AND CONCLUSION

The adaptive filters presented in Chapters 4, 6, and 7, in FIR or IIR direct form, have a strong structural resemblance, illustrated in the following coefficient updating equations:

$$\left|\begin{matrix}\text{new}\\ \text{coefficient}\\ \text{vector}\end{matrix}\right| = \left|\begin{matrix}\text{old}\\ \text{coefficient}\\ \text{vector}\end{matrix}\right| + \left|\begin{matrix}\text{step}\\ \text{size}\end{matrix}\right|\left|\begin{matrix}\text{input}\\ \text{data}\\ \text{vector}\end{matrix}\right|\left|\begin{matrix}\text{innovation}\\ \text{signal}\end{matrix}\right|$$

To determine the terms in that equation, the adaptive filter has only the data vector and reference signal available. All other variables, including the coefficients, are estimated. There are two categories of estimates; those which constitute predictions from the past, termed a priori, and those which incorporate the new information available, termed a posteriori. The final output of the filter is the a posteriori error signal

$$\varepsilon(n+1) = y(n+1) - H^t(n+1)X(n+1) \tag{7.211}$$

which can be interpreted as a measurement noise, a model error, or, in prediction, an excitation signal.

The innovation signal $i(n)$ represents the difference between the reference $y(n+1)$ and a priori estimates which are functions of the past coefficients and output errors:

$$i(n+1) = y(n+1) - F_1[H^t(n), H^t(n-1), \ldots]X(n+1) \\ - F_2[\varepsilon(n), \varepsilon(n-1), \ldots] \tag{7.212}$$

or, in terms of variable deviations

$$i(n+1) = \Delta H^t(n+1)X(n+1) + \Delta\varepsilon(n+1) \tag{7.213}$$

with

$$\Delta H(n+1) = H(n+1) - F_1[H(n), H(n-1), \ldots]$$
$$\Delta\varepsilon(n+1) = \varepsilon(n+1) - F_2[\varepsilon(n), \varepsilon(n-1), \ldots]$$

The derivation of an adaptive algorithm requires the design of predictors to generate the a priori estimates and a criterion defining how to use the innovation $i(n+1)$ to determine the a posteriori estimates from the a priori ones.

When one takes

$$i(n+1) = e(n+1) = y(n+1) - H^t(n)X(n+1) \tag{7.214}$$

one simply assumes that the a priori estimate $H(n)$ for the coefficients is the a posteriori estimate at time n, which is valid for short-term stationary signals, and that the a priori error signal is zero, which is reasonable since the error signal is expected to be a zero mean white noise [14].

Minimizing the deviation between a posteriori and a priori estimates, with the cost function

$$J(n) = \Delta H^t(n)R(n)\Delta H(n) + [\Delta\varepsilon(n)]^2 \tag{7.215}$$

where $R(n)$ is a symmetric positive definite weighting matrix, yields

$$H(n+1) = H(n) + \frac{i(n+1)}{1 + X^t(n+1)R^{-1}(n+1)X(n+1)} R^{-1}(n+1)X(n+1) \tag{7.216}$$

The flow graph of the general direct-form adaptive filter is given in Figure 7.17. It is valid for real, complex, or M-D data. The type of algorithm employed impacts the matrix $R(n)$, which is diagonal for the LSM algorithm and a square symmetric matrix for the LS approaches. Only the IIR filter discussed in Sections 4.15 and 7.12 uses an error prediction calculation to control the stability. The coefficient prediction filter can be usd in a nonstationary environment to exploit the a priori knowledge on the nature of the nonstationarity and perform an appropriate bandlimited extrapolation.

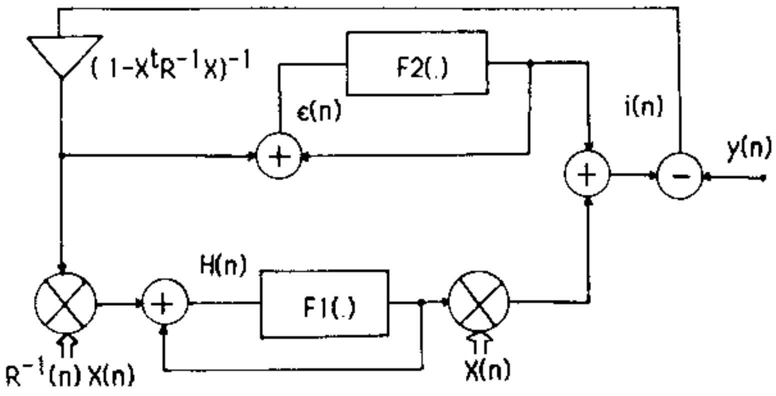


FIG. 7.17 General direct-form adaptive filter.

Finally, the transversal adaptive filters form a large, diverse, and versatile family which can satisfy the requirements of applications in many technical fields. Their complexity can be tailored to the resources of the users, and their performances assessed accordingly. It is particularly remarkable to observe how flexible the FLS algorithms are, since they can provide exact solutions for different kinds of signals, observation conditions, and structures. A further illustration is given in the next chapter.

EXERCISES

1. Use the approach in Section 7.1 to derive an algorithm based on all prediction errors as in Section 6.5, with nonzero initial input data vector. What is the additional computation load?
2. Taking

$$e(n+1) = y(n+1) - H^t(n)X(n+1)$$

instead of (7.41) as the definition for the output error signal, give the computational organization of an alternative FLS algorithm for complex signals. Show that only forward prediction equations are modified by complex conjugation operations. Compare with the equations given in Section 7.3.
3. Give the detailed computational organization of an FLS algorithm for 2-D input signals, the coefficient vectors $H_1(n)$ and $H_2(n)$ having $N_1 = N_2 = 4$ elements. Count the memories needed. Modify the algorithm to achieve the minimum number of operations when $N_1 = 4$ and

$N_2 = 2$. What reduction in number of multiplications and memories is obtained?

4. Extend the algorithm given in Section 7.4 for M-D input signals to the case of a sliding window. Estimate the additional computation load.
5. At the input of an adaptive filter with order $N = 4$, the signal is sampled at 4 kHz. The observed reference signal is available at the sampling frequency 1 kHz. Give the FLS algorithm for this multirate filter. Compare the complexities of the multirate algorithm and the standard algorithm which corresponds to a 4-kHz reference signal sampling rate. Compare also the performance of the two algorithms; what is the penalty in adaptation speed brought by undersampling the reference?
6. Use the technique described in Section 7.7 for pole-zero modeling to design an LS FIR/IIR predictor. Compare the 2-D and 3-D approaches in terms of computational complexity.
7. Consider the FDAF in Figure 7.10. The orthogonal transform of order N is the DCT which produces real outputs; describe the corresponding FLS algorithms. Compare the multiplication speed obtained with that of a direct FLS algorithm of order N. Compare also the performance of the two approaches.

ANNEX 7.1 FLS ALGORITHM WITH SLIDING WINDOW

```
C
      SUBROUTINE FLSSW(N,NO,X,A,EAB,EA,G,GO,IND)
C
C     COMPUTES THE ADAPTATION GAIN (F.L.S. with SLIDING
      WINDOW)
C     N   = FILTER ORDER
C     NO  = WINDOW LENGTH
C     X   = INPUT SIGNAL : x(n+1)
C     VXO = DATA VECTOR : N+NO ELEMENTS
C     A   = FORWARD PREDICTION COEFFICIENTS
C     B   = BACKWARD PREDICTION COEFFICIENTS
C     EA  = PREDICTION ERROR ENERGY
C     G   = ADAPTATION GAIN
C     GO  = BACKWARD ADAPTATION GAIN

C     IND = TIME INDEX
C
      DIMENSION VXO(500),A(15),B(15),G(15),G1(16),
      GO(15),GO1(16) IF(IND.GT.1)GOTO30
```

```
C
C     INITIALIZATION
C
      DO20I=1,15
      A(I)=0.
      B(I)=0.
      G(I)=0.
      GO(I)=0.
   20 CONTINUE
      DO21I=1,500
      VXO(I)=0.
   21 CONTINUE
      EA=0.1
   30 CONTINUE
C
C     ADAPTATION GAIN CALCULATION
C
      EAV=X
      EPSA=X
      EAVO=VXO(NO)
      EPSAO=VXO(NO)
      DO40I=1,N
      EAV=EAV-A(I)*VXO(I)
   40 EAVO=EAVO-A(I)*VXO(NO+I)
      DO50I=1,N
      A(I)=A(I)+G(I)*EAV-GO(I)*EAVO
      EPSA=EPSA-A(I)*VXO(I)
      EPSAO=EPSAO-A(I)*VXO(NO+I)
   50 CONTINUE
      EA=EA+EAV*EPSA-EAVO*EPSAO
      G1(1)=EPSA/EA
      GO1(1)=EPSAO/EA
      DO60I=1,N
      G1(I+1)=G(I)-A(I)*G1(1)
      GO1(I+1)=GO(I)-A(I)*GO1(1)
   60 CONTINUE
      EAB=VXO(N)
      EABO=VXO(N+NO)
      DO70I=2,NO+N
      J=NO+N+1-I
   70 VXO(J+1)=VXO(J)
      VXO(1)=X
      DO80I=1,N
      EAB=EAB-B(I)*VXO(I)
      EABO=EABO-B(I)*VXO(I+NO)
```

```
  80   CONTINUE
       GG1=G1(N+1)/(1.+GO1(N+1)*EABO)
       GGO=GO1(N+1)/(1.-G1(N+1)*EAB)
       DO90I=1,N
       G(I)=G1(I)+GG1*(B(I)-EABO*GO1(I))
       G(I)=G(I)/(1.-GG1*EAB)
       GO(I)=GO1(I)+GGO*(B(I)+EAB*G1(I))
       GO(I)=GO(I)/(1.+GGO*EABO)
  90   CONTINUE
       DO100I=1,N
  100  B(I)=B(I)+G(I)*EAB-GO(I)*EABO
       RETURN
       END
```

ANNEX 7.2 FLS ALGORITHM FOR FORWARD–BACKWARD LINEAR PREDICTION

```
C
       SUBROUTINE FLSFB(N,X,B,EE,U,W,IND)
C
C      COMPUTES THE ADAPTATION GAINS FOR COMBINED
       FORWARD-BACKWARD
C      PREDICTION USING A FAST LEAST SQUARES ALGORITHM
C      N    = FILTER ORDER
C      X    = INPUT SIGNAL : x(n+1)
C      VX   = DATA VECTOR : X(n) ; N elements
C      B    = COEFFICIENT VECTOR ; N elements
C      G1   = FORWARD GAIN VECTOR
C      G2   = BACKWARD GAIN VECTOR
C      U    = SYMMETRIC GAIN VECTOR
C      W    = WEIGHTING FACTOR
C      IND  = TIME INDEX
C
       DIMENSION VX(15),B(15),G1(15),G2(15),U(15),
       U1(16) IF(IND.GT.1)GOTO30
C
C      INITIALIZATION
C
       DO20I=1,N
       B(I)=0.
       G1(I)=0.
       G2(I)=0.
       VX(I)=0.
  20   CONTINUE
       EPSU=0.
```

```
      EE=0.1
   30 CONTINUE
C
C     ADAPTATION GAIN CALCULATION
C
      DO40I=1,N
   40 U(I)=G2(I)-G1(I)*EPSU
      EPSG=0.
      EPSGG=W*W
      DO50I=1,N
      EPSGG=EPSGG+VX(I)*U(I)
   50 EPSG=EPSG+VX(I)*U(N+1-I)
      EPSG1=EPSG
      EPSG=EPSG/EPSGG
      DO60I=1,N
   60 G1(I)=(U(N+1-I)-EPSG*U(I))/W
      EAV=0.
      DO70I=1,N
   70 EAV=EAV+B(N+1-I)*VX(I)
      EAV=X-EAV*W
      U1(1)=EAV/EE
      DO80I=1,N
   80 U1(N+2-I)=G1(I)-U1(1)*B(I)
      DO90I=1,N
   90 G2(I)=U1(I)+U1(N+1)*B(I)
      ALF1=(EPSGG-EPSG*EPSG1)/(W*W)
      EAB=VX(N)*W
      DO100I=1,N-1
  100 VX(N+1-I)=VX(N-I)*W
      VX(1)=X
      ALF2=0.
      DO105I=1,N
  105 ALF2=ALF2+VX(I)*G2(I)
      ALF2=1.+ALF2/(W*W)
      DO110I=1,N
  110 EAB=EAB-B(I)*VX(I)
      ALF12=0.
      DO120I=1,N
  120 ALF12=ALF12+VX(I)*G1(I)
      ALF12=ALF12/(W*W)
      EPSU=ALF12/ALF1
      ALFF=ALF1*ALF2-ALF12*ALF12
      EPSA=(ALF2*EAV-ALF12*EAB)/ALFF
      EPSB=(ALF1*EAB-ALF12*EAV)/ALFF
      EE=W*W*EE+EPSA*EAV+EPSB*EAB
      DO130I=1,N
  130 B(I)=B(I)+(G1(I)*EPSA+G2(I)*EPSB)/(W*W)
      RETURN
      END
C
C
```

ANNEX 7.3 FLS ALGORITHM WITH MULTIDIMENSIONAL INPUT SIGNAL

```
      SUBROUTINE FLS1MD(K,N,EAINV,UA,UB,VU,VU1,
      A,G,B,W)
C
C     COMPUTES THE ADAPTATION GAIN FOR MULTIDIMENSIONAL
      INPUT SIGNAL
C
C     K     = NUMBER OF INPUT SIGNALS (FILTER DIMENSION)
C     N     = NUMBER OF COEFFICIENTS IN EVERY CHANNEL
C     UA    = INPUT VECTOR AT TIME (n+1)
C     UB    = INPUT VECTOR AT TIME (n+1-N)
C     VU    = KN ELEMENT DATA VECTOR AT TIME (n)
C     VU1   = KN ELEMENT DATA VECTOR AT TIME (n+1)
C     A     = FORWARD LINEAR PREDICTION (KNxK) MATRIX
C     B     = BACKWARD LINEAR PREDICTION (KNxK) MATRIX
C     G     = ADAPTATION GAIN VECTOR
C     EAINV = PREDICTION ERROR ENERGY INVERSE (KxK) MATRIX
C     W     = WEIGHTING FACTOR
C
      DIMENSION UA(1),UB(1),VU(1),VU1(1),G(1)
      DIMENSION A(20,10),B(20,10),EAINV(10,10)
      DIMENSION SM(10),RM(20),EKA(10),EKB(10),
      AUX(10,10)
      DIMENSION EPKA(10),P1(10,10),P2(10),P3(10,10),
      P5(10,10)
      KN=K*N
C
C     FORWARD LINEAR PREDICTION ERROR :
C
      DO 1 I=1,K
      PR=0.
      P2(I)=0.
      DO 2 J=1,KN
      PR=PR+A(J,I)*VU(J)
    2 CONTINUE
      EKA(I)=UA(I)-PR
```

```
  1    CONTINUE
C
C      FORWARD PREDICTON MATRIX :
C
       DO 3 I=1,KN
       DO 4 J=1,K
       A(I,J)=A(I,J)+G(I)*EKA(J)
  4    CONTINUE
  3    CONTINUE
C
C      A POSTERIORI PREDICTION ERROR :
C
       DO 5 I=1,K
       PR=0.
       DO 6 J=1,KN
       PR=PR+A(J,I)*VU(J)
  6    CONTINUE
       EPKA(I)=UA(I)-PR
  5    CONTINUE
C
C      UPDATING OF ERROR ENERGY INVERSE MATRIX :
C
       P4=0.
       DO 7 J=1,K
       DO 8 I=1,K
       P1(J,I)=EKA(J)*EPKA(I)
       P2(J)=P2(J)+EPKA(I)*EAINV(I,J)
       P3(I,J)=0.
       P5(I,J)=0.
  8    CONTINUE
  7    CONTINUE
       DO 21 I=1,K
       DO 22 J=1,K
       DO 23 L=1,K
       P3(I,J)=P3(I,J)+EAINV(I,L)*P1(L,J)
  23   CONTINUE
  22   CONTINUE
       P4=P4+P2(I)*EKA(I)
  21   CONTINUE
       P4=P4+W
       DO 24 I=1,K
       DO 25 J=1,K
       DO 26 L=1,K
       P5(I,J)=P5(I,J)+P3(I,L)*EAINV(L,J)
  26   CONTINUE
```

```
      P5(I,J)=P5(I,J)/P4
   25 CONTINUE
   24 CONTINUE
      DO 27 I=1,K
      DO 28 J=1,K
      EAINV(I,J)=(EAINV(I,J)-P5(I,J))/W
      AUX(I,J)=EAINV(I,J)
   28 CONTINUE
   27 CONTINUE
C
C     EAINV IS IN AUX FOR SUBSEQUENT CALCULATIONS
C     KN+K ELEMENT ADAPTATION GAIN (VECTORS RM AND SM) :
C
      DO 9 I=1,K
      EX=0.
      DO 10 J=1,K
      EX=EX+AUX(I,J)*EPKA(J)
   10 CONTINUE
      AUX(I,1)=EX
    9 CONTINUE
      DO 11 I=K+1,KN+K
      EX=0.
      DO 12 J=1,K
      EX=EX-A(I-K,J)*AUX(J,1)
   12 CONTINUE
      AUX(I,1)=EX+G(I-K)
   11 CONTINUE
      DO 13 I=1,KN
      RM(I)=AUX(I,1)
      IF(I.LE.K) SM(I)=AUX(KN+I,1)
   13 CONTINUE
C
C     BACKWARD PREDICTION ERROR :
C
      DO 14 I=1,K
      PR=0.
      DO 15 J=1,KN
      PR=PR+B(J,I)*VU1(J)
   15 CONTINUE
      EKB(I)=UB(I)-PR
   14 CONTINUE
C
C     KN ELEMENT ADAPTATION GAIN :
C
      EX=0.
      DO 16 I=1,K
      EX=EX+EKB(I)*SM(I)
  16  CONTINUE
      EX=1./(1.-EX)
      DO 17 I=1,KN
      PR=0.
      DO 18 J=1,K
      PR=PR+B(I,J)*SM(J)
  18  CONTINUE
      G(I)=EX*(RM(I)+PR)
  17  CONTINUE
C
C     BACKWARD PREDICTION (KNxK) MATRIX :
C
      DO 19 I=1,KN
      DO 20 J=1,K
      B(I,J)=B(I,J)+G(I)*EKB(J)
  20  CONTINUE
  19  CONTINUE
      RETURN
      END
```

REFERENCES

1. D. Lin, "On Digital Implementation of the Fast Kalman Algorithms," *IEEE Trans.* **ASSP-32**, 998–1005 (October 1984).
2. M. L. Honig and D. G. Messerschmitt, *Adaptive Filters: Structures, Algorithms and Applications*, Kluwer Academic, Boston, 1984, Chap. 6.
3. D. Manolakis, F. Ling, and J. Proakis, "Efficient Time Recursive Least Squares Algorithms for Finite Memory Adaptive Filtering," *IEEE Trans.* **CAS-34**, 400–407 (April 1987).
4. B. Toplis and S. Pasupathy, "Tracking Improvements in Fast RLS Algorithms Using a Variable Forgetting Factor," *IEEE Trans.* **ASSP-36**, 206–227 (February 1988).
5. N. Kalouptsidis and S. Theodoridis, "Efficient Structurally Symmetric Algorithms for Least Squares FIR Filters with Linear Phase," *IEEE Trans.* **ASSP-36**, 1454–1465 (September 1988).
6. L. Resende, J. M. T. Romano, and M. Bellanger, "A Fast Least Squares Algorithm for Linearly Constrained Adaptive Filtering," *IEEE Trans.* **SP-44**, 1168–1174 (May 1996).
7. S. T. Alexander, "A Derivation of the Complex Fast Kalman Algorithm," *IEEE Trans.* **ASSP-32**, 1230–1232 (December 1984).
8. D. Falconer and L. Ljung, "Application of Fast Kalman Estimation to Adaptive Equalization," *IEEE Trans.* **COM-26**, 1439–1446 (October 1978).

9. K. Kurosawa and S. Tsujii, "A New IIR Adaptive Algorithm of Parallel Type Structure," *Proc. IEEE/ICASSP-86* Tokyo, 1986, pp. 2091–2094.
10. E. R. Ferrara, "Frequency Domain Adaptive Filtering," in *Adaptive Filters*, Prentice-Hall, Englewood Cliffs, N.J., 1985.
11. J. C. Ogue, T. Saito, and Y. Hoshiko, "A Fast Convergence Frequency Domain Adaptive Filter," *IEEE Trans.* **ASSP-31**, 1312–1314 (October 1983).
12. S. S. Narayan, A. M. Peterson, and M. J. Narasimha, "Transform Domain LMS Algorithm," *IEEE Trans.* **ASSP-31**, 609–615 (June 1983).
13. C. E. Davila, A. J. Welch, and H. G. Rylander, "A Second Order Adaptive Volterra Filter with Rapid Convergence," *IEEE Trans.* **ASSP-35**, 1259–1263 (September 1987).
14. G. Kubin, "Direct Form Adaptive Filter Algorithms: A Unified View," in *Signal Processing III*, Elsevier, 1986, pp. 127–130.

8 Lattice Algorithms and Geometrical Approach

Although FLS algorithms for transversal adaptive sructures are essentially based on time recursions, the algorithms for lattice structures make a joint use of time and order recurrence relationships. For a fixed filter order value N, they require more operations than their transversal counterparts. However, they provide adaptive filters of all the intermediate orders from 1 to N, which is an attractive feature in those applications where the order is not known beforehand and several different values have to be tried [1–3].

The order recurrence relationships introduced in Section 5.6 can be extended to real-time estimates.

8.1. ORDER RECURRENCE RELATIONS FOR PREDICTION COEFFICIENTS

Let $A_N(n)$, $B_N(n)$, $E_{aN}(n)$, $E_{bN}(n)$ and $G_N(n)$ denote the input signal prediction coefficient vectors, the error energies, and the adaptation gain at time n for filter order N. The forward linear prediction matrix equation for order $N-1$ is

$$R_N(n)\begin{bmatrix} 1 \\ -A_{N-1}(n) \end{bmatrix} = \begin{bmatrix} E_{a(N-1)}(n) \\ 0 \end{bmatrix} \tag{8.1}$$

Similarly, the backward prediction equation is

$$R_N(n)\begin{bmatrix} -B_{N-1}(n) \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ E_{b(N-1)}(n) \end{bmatrix} \tag{8.2}$$

Now, partitioning equation (6.61) in Chapter 6 for $R_{N+1}(n)$ yields

$$\begin{bmatrix} R_N(n) & r_N^b(n) \\ \cdots\cdots & \cdots\cdots \\ [r_N^b(n)]^t & R_1(n-N) \end{bmatrix} \begin{bmatrix} 1 \\ -A_{N-1}(n) \\ \cdots\cdots \\ 0 \end{bmatrix} = \begin{bmatrix} E_{a(N-1)}(n) \\ 0 \\ \cdots\cdots \\ K_N(n) \end{bmatrix} \tag{8.3}$$

where the variable $K_N(n)$, corresponding to the last row, is

$$K_N(n) = \sum_{p=1}^{n} W^{n-p} x(p)x(p-N) - A_{N-1}^t(n) R_{N-1}(n-1) B_{N-1}(n-1) \tag{8.4}$$

In (8.4), forward and backward prediction coefficients appear in a balanced manner. Therefore the same variable $K_N(n)$ appears in the backward prediction matrix equation as well:

$$\begin{bmatrix} R_1(n) & [r_N^a(n)]^t \\ \cdots\cdots & \cdots\cdots \\ r_N^a(n) & R_N(n-1) \end{bmatrix} \begin{bmatrix} 0 \\ \cdots\cdots \\ -B_{N-1}(n-1) \\ 1 \end{bmatrix} = \begin{bmatrix} K_N(n) \\ \cdots\cdots \\ 0 \\ E_{b(N-1)}(n-1) \end{bmatrix} \tag{8.5}$$

as can be readily verified by analyzing the first row. Multiplying both sides by the scalar $K_N(n)/E_{b(N-1)}(n-1)$ gives

$$R_{N+1}(n) \begin{bmatrix} 0 \\ \begin{bmatrix} -B_{N-1}(n-1) \\ 1 \end{bmatrix} \dfrac{K_N(n)}{E_{b(N-1)}(n-1)} \end{bmatrix} = \begin{bmatrix} \dfrac{K_N^2(n)}{E_{b(N-1)}(n-1)} \\ 0 \\ K_N(n) \end{bmatrix} \tag{8.6}$$

Now, subtracting equation (8.6) from equation (8.3) and identifying with the forward prediction matrix equation (8.1) for order N, we botain the following recursion for the forward prediction coefficient vectors:

$$A_N(n) = \begin{bmatrix} A_{N-1}(n) \\ 0 \end{bmatrix} - \frac{K_N(n)}{E_{b(N-1)}(n-1)} \begin{bmatrix} B_{N-1}(n-1) \\ -1 \end{bmatrix} \tag{8.7}$$

The first row yields a recursion for the forward prediction error energies:

$$E_{aN}(n) = E_{a(N-1)}(n) - \frac{K_N^2(n)}{E_{b(N-1)}(n-1)} \tag{8.8}$$

The same method can be applied to backward prediction equations. Matrix equation (8.3) can be rewritten as

$$R_{N+1}(n) \begin{bmatrix} \begin{bmatrix} 1 \\ -A_{N-1}(n) \end{bmatrix} \dfrac{K_N(n)}{E_{a(N-1)}(n)} \\ 0 \end{bmatrix} = \begin{bmatrix} K_N(n) \\ 0 \\ \dfrac{K_N^2(n)}{E_{a(N-1)}(n)} \end{bmatrix} \tag{8.9}$$

Subtracting equation (8.9) from equation (8.5) and identifying with the backward prediction matrix equation (8.2) for order N lead to recurrence relations for the backward prediction coefficients vectors

$$B_N(n) = \begin{bmatrix} 0 \\ B_{N-1}(n-1) \end{bmatrix} - \frac{K_N(n)}{E_{a(N-1)}(n)} \begin{bmatrix} -1 \\ A_{N-1}(n) \end{bmatrix} \tag{8.10}$$

and for the backward prediction error energy

$$E_{bN}(n) = E_{b(N-1)}(n-1) - \frac{K_N^2(n)}{E_{a(N-1)}(n)} \tag{8.11}$$

The definitions of the backward prediction a priori error

$$e_{aN}(n+1) = x(n+1) - A_N^t(n)X(n)$$

and backward prediction error

$$e_{bN}(n+1) = x(n+1-N) - B_N^t(n)X(n+1)$$

in connection with recursions (8.7) and (8.10), lead to the lattice predictor structure, which relates errors for orders N and $N-1$:

$$e_{aN}(n+1) = e_{a(N-1)}(n+1) - \frac{K_N(n)}{E_{b(N-1)}(n-1)} e_{b(N-1)}(n) \tag{8.12}$$

and

$$e_{bN}(n+1) = e_{b(N-1)}(n) - \frac{K_N(n)}{E_{a(N-1)}(n)} e_{a(N-1)}(n+1) \tag{8.13}$$

Similarly, for a posteriori errors,

$$\varepsilon_{aN}(n+1) = x(n+1) - A_N^t(n+1)X(n)$$

and

$$\varepsilon_{bN}(n+1) = x(n+1-N) - B_N^t(n+1)X(n+1)$$

The lattice structure operations are

$$\varepsilon_{aN}(n+1) = \varepsilon_{a(N-1)}(n+1) - k_{bN}(n+1)\varepsilon_{b(N-1)}(n) \tag{8.14a}$$

$$\varepsilon_{bN}(n+1) = \varepsilon_{b(N-1)}(n) - k_{aN}(n+1)\varepsilon_{a(N-1)}(n+1) \tag{8.14b}$$

where

$$k_{aN}(n+1) = \frac{K_N(n+1)}{E_{a(N-1)}(n+1)}, \qquad k_{bN}(n+1) = \frac{K_N(n+1)}{E_{b(N-1)}(n)} \tag{8.15}$$

are the estimates of the PARCOR or reflection coefficients introduced in Section 5.5.

The flow diagram of the corresponding lattice filter section is shown in Figure 8.1. The same structure can be used for a priori and a posteriori errors. A prediction error filter of order N is obtained by cascading N such sections.

Similar order recursions can be derived for the coefficients of adaptive filters, the adaptation gain, and the ratio of a posteriori to a priori errors.

8.2. ORDER RECURRENCE RELATIONS FOR THE FILTER COEFFICIENTS

An adaptive filter with N coefficients produces an output error signal $e_N(n)$:

$$e_N(n+1) = y(n+1) - H_N^t(n)X(n+1) \tag{8.16}$$

The coefficient vector $H_N(n)$, which minimizes the error energy at time n, is obtained by

$$H_N(n) = R_N^{-1}(n)r_{yxN}(n) \tag{8.17}$$

with

$$r_{yxN}(n) = \sum_{p=1}^{n} W^{n-p}y(p)X_N(p)$$

For a filter with $N+1$ coefficients, the equations are

$$e_{N+1}(n+1) = y(n+1) - H_{N+1}^t(n)X_{N+1}(n+1) \tag{8.18a}$$

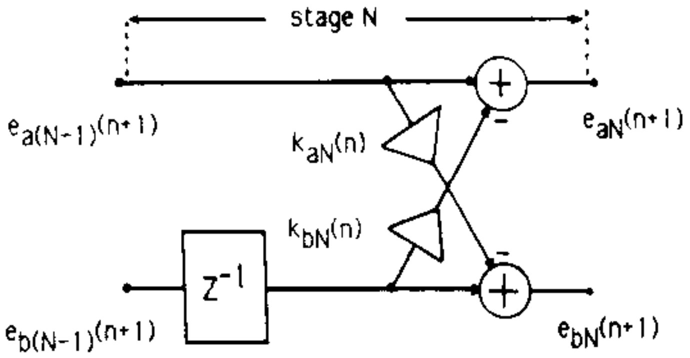


FIG. 8.1 Adaptive lattice prediction error filter section.

$$R_{N+1}(n)H_{N+1}(n) = \begin{bmatrix} r_{yxN}(n) \\ \sum_{p=1}^{n} W^{n-p} y(p)x(p-N) \end{bmatrix} \tag{8.18b}$$

The coefficient vector $H_{N+1}(n)$ can be obtained from $H_N(n)$ with the help of the partitioning (6.61) of Chapter 6 of the input signal AC matrix. As in the preceding section, consider the equation

$$\begin{aligned} R_{N+1}(n)\begin{bmatrix} H_N(n) \\ 0 \end{bmatrix} &= \begin{bmatrix} R_N(n) & r_N^b(n) \\ [r_N^b(n)]^t & R_1(n-N) \end{bmatrix}\begin{bmatrix} H_N(n) \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} r_{yxN}(n) \\ [r_N^b(n)]^t H_N(n) \end{bmatrix} \end{aligned} \tag{8.19}$$

The last row can also be written as

$$[r_N^b(n)]^t H_N(n) = B_N^t(n)R_N(n)H_N(n) = B_N^t(n)r_{yxN}(n) \tag{8.20}$$

Subtracting equation (8.19) from (8.18a) yields

$$R_{N+1}(n)\left[H_{N+1}(n) - \begin{bmatrix} H_N(n) \\ 0 \end{bmatrix}\right] = \begin{bmatrix} 0 \\ K_{fN}(n) \end{bmatrix} \tag{8.21}$$

where

$$K_{fN}(n) = \sum_{p=1}^{n} W^{n-p} y(p)[x(p-N) - B_N^t(n)X(p)] \tag{8.22}$$

Now, identifying equation (8.21) with the backward linear prediction matrix equation leads to the following recurrence equation for the filter coefficients:

$$H_{N+1}(n) = \begin{bmatrix} H_N(n) \\ 0 \end{bmatrix} - \frac{K_{fN}(n)}{E_{bN}(n)}\begin{bmatrix} B_N(n) \\ -1 \end{bmatrix} \tag{8.23}$$

Substituting (8.23) into definition (8.17a) yields the relation for a priori output errors

$$e_{N+1}(n+1) = e_N(n+1) - \frac{K_{fN}(n)}{E_{bN}(n)} e_{bN}(n+1) \tag{8.24}$$

The corresponding equation for a posteriori errors is

$$\varepsilon_{N+1}(n+1) = \varepsilon_N(n+1) - \frac{K_{fN}(n+1)}{E_{bN}(n+1)} \varepsilon_{bN}(n+1) \tag{8.25}$$

Altogether, equations (8.12), (8.13), and (8.24) constitute the set of a priori equations for the lattice filter, while equations (8.14a,b) and (8.25) give the a posteriori version.

The error energy can also be computed recursively. According to the definition of the filter error output, we have

$$E_{N+1}(n) = \sum_{p=1}^{n} W^{n-p} y^2(p) - H_{N+1}^t(n) R_{N+1}(n) H_{N+1}(n) \tag{8.26}$$

Substituting recurrence relation (8.23) into (8.26) and using the backward prediction matrix equation, we obtain the order recursion

$$E_{N+1}(n) = E_N(n) - \frac{K_{fN}^2(n)}{E_{bN}(n)} \tag{8.27}$$

Obviously $E_{N+1}(n) \leqslant E_N(n)$, and the error power decreases as the filter order increases, which is a logical result.

Recall from Section 6.4 that the adaptation gain can be computed in a similar way. The derivation is repeated here for convenience. From the definition relation

$$R_N(n) G_N(n) = X_N(n) \tag{8.28}$$

we have

$$\begin{aligned} R_N(n) \begin{bmatrix} G_{N-1}(n) \\ 0 \end{bmatrix} &= \begin{bmatrix} R_{N-1}(n) & r_{N-1}^b(n) \\ [r_{N-1}^b(n)]^t & R_1(n+1-N) \end{bmatrix} \begin{bmatrix} G_{N-1}(n) \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} X_{N-1}(n) \\ [r_{N-1}^b(n)]^t G_{N-1}(n) \end{bmatrix} \end{aligned} \tag{8.29}$$

The last row can be expressed by

$$[r_{N-1}^b(n)]^t G_{N-1}(n) = B_{N-1}^t(n) X_{N-1}(n) = x(n+1-N) - \varepsilon_{b(N-1)}(n) \tag{8.30}$$

and equation (8.29) can be rewritten as

$$\begin{bmatrix} G_{N-1}(n) \\ 0 \end{bmatrix} = G_N(n) - R_N^{-1}(n) \begin{bmatrix} 0 \\ \varepsilon_{b(N-1)}(n) \end{bmatrix} \tag{8.31}$$

But the last row of the inverse AC matrix is proportional to the backward prediction coefficient vector; hence

$$G_N(n) = \begin{bmatrix} G_{N-1}(n) \\ 0 \end{bmatrix} + \frac{\varepsilon_{b(N-1)}(n)}{E_{b(N-1)}(n)} \begin{bmatrix} -B_{N-1}(n) \\ 1 \end{bmatrix} \tag{8.32}$$

This is equation (6.75) in Section 6.4. Recall that the other partitioning of $R_N(n)$ and the use of forward variables led to equation (6.73) in Chapter 6, which is a mixture of time and order recursions.

This expression is useful to recursively compute the ratio $\varphi_N(n)$ of a posteriori to a priori errors, defined by

$$\varphi_N(n) = \frac{\varepsilon_N(n)}{e_N(n)} = 1 - X_N^t(n)R_N^{-1}(n)X_N(n) = 1 - X_N^t(n)G_N(n)$$

Direct substitution yields

$$\varphi_N(n) = \varphi_{N-1}(n) - \frac{\varepsilon^2_{b(N-1)}(n)}{E_{b(N-1)}(n)} \tag{8.33}$$

The initial stage $N = 1$ is worth considering:

$$\varphi_1(n) = \varphi_0(n) - \frac{\varepsilon^2_{b0}(n)}{E_{b0}(n)} = 1 - \frac{x^2(n)}{\sum\limits_{p=1}^{n} W^{n-p}x^2(p)}$$

Thus, in order to compute $\varphi_N(n)$ recursively, it is sufficient to take $\varphi_0(n) = 1$ and repeatedly use equation (8.33).

We reemphasize that $\varphi_N(n)$ is a crucial variable in FLS algorithms. It is of particular importance in lattice algorithms because it forms an essential link between order and time recursions.

8.3. TIME RECURRENCE RELATIONS

For a fixed filter order N, the lattice variable $K_N(n)$ can be computed recursively in time. According to definition (8.4), we have

$$\begin{aligned} K_{N+1}(n+1) = {} & W\sum_{p=1}^{n} W^{n-p}x(p)x(p-N-1) + x(n+1)x(n-N) \\ & - A_N^t(n+1)R_N(n)B_N(n) \end{aligned} \tag{8.34}$$

Now, from the time recurrence relations (6.45), (6.26), and (6.53) in Chapter 6 for $A_N(n+1)$, $R_N(n)$, and $B_N(n)$, respectively, the following updating relation is obtained after some algebraic manipulations:

$$K_{N+1}(n+1) = WK_{N+1}(n) + e_{aN}(n+1)\varepsilon_{bN}(n) \tag{8.35}$$

Due to relations (6.49) and (6.56) of Chapter 6 between a priori and a posteriori errors, an alternative updating equation is

$$K_{N+1}(n+1) = WK_{N+1}(n) + \varepsilon_{aN}(n+1)e_{bN}(n) \tag{8.36}$$

Clearly, the variable $K_{N+1}(n)$ represents an estimation of the cross-correlation between forward and backward order N prediction errors. Indeed, equation (8.35) is similar to the prediction error energy updating relations (6.58) and (6.59) derived and used in Chapter 6.

A similar relation can be derived for the filter output error energy $E_N(n)$. Equation (8.26) for order N and time $n+1$ corresponds to

$$E_N(n+1) = \sum_{p=1}^{n+1} W^{n+1-p} y^2(p) - H_N^t(n+1) R_N(n+1) H_N(n+1) \tag{8.37}$$

Substituting the coefficient updating relation

$$H_N(n+1) = H_N(n) + G_N(n+1) e_N(n+1) \tag{8.38}$$

into (8.37), again yields after simplification

$$E_N(n+1) = W E_N(n) + e_N(n+1) \varepsilon_N(n+1) \tag{8.39}$$

For the filter section variable $K_{fN}(n+1)$, definition (8.22) can be rewritten as

$$\begin{aligned} K_{fN}(n+1) = & \sum_{p=1}^{n+1} W^{n+1-p} y(p) x(p-N) \\ & - [B_N^t(n) + G_N^t(n+1) e_{bN}(n+1)] \\ & \times [W r_{yxN}(n) + y(n+1) X_N(n+1)] \end{aligned} \tag{8.40}$$

which, after simplification, leads to

$$K_{fN}(n+1) = W K_{fN}(n) + \varepsilon_{bN}(n+1) e_N(n+1) \tag{8.41}$$

Note that the variable $K_{fN}(n+1)$, which according to definition (8.22) is an estimate of the cross-correlation between the reference signal and the backward prediction error, can be calculated as an estimate of the cross-correlation between the filter output error and the backward prediction error. This is due to the property of noncorrelation between the prediction errors and the data vector.

The recurrence relations derived so far can be used to build FLS algorithms for filters in lattice structures.

8.4. FLS ALGORITHMS FOR LATTICE STRUCTURES

The algorithms combine time and order recurrence relations to compute, for each set of new values of input and reference signals which become available, the lattice coefficients, the prediction and filter errors, their energies, and their cross-correlations. For a filter of order N, the operations are divided into prediction and filter operations.

To begin with, let us consider the initialization procedure. Since there are two types of recursions, two types of initializations have to be distinguished. The initializations for the order recursions are obtained in a straightforward manner: the prediction errors are initialized by the new input signal sample, the prediction error energies are set equal to the input signal power, and the variable $\varphi_0(n)$ is set to 1.

For time recursions, an approach to initialize the state variables of the order N lattice filter can be obtained as an extension of that given in Section 6.7. The input signal for $n \leqslant 0$ is assumed to consist of a single pulse at time $-N$, which leads to

$$\begin{aligned} e_{ai}(0) = e_{bi}(0) = \varepsilon_{ai}(0) = \varepsilon_{bi}(0) = 0, &\quad 0 \leqslant i \leqslant N-1 \\ E_{ai}(0) = W^N E_0, &\quad 0 \leqslant i \leqslant N-1 \\ E_{bi}(0) = W^{N-i} E_0, &\quad 0 \leqslant i \leqslant N-1 \\ K_i(0) = 0, &\quad 1 \leqslant i \leqslant N \end{aligned} \tag{8.42}$$

where E_0 is a real positive scalar. It can be verified that the prediction order recursions, and particularly energy relations (8.8) and (8.11), are satisfied for $n = 0$. Indeed, in these conditions, the impact of the choice of the initial error energy value E_0 on the filter performance is the same as for the transversal structure, and the relevant results given in Chapter 6 are still valid.

Many more or less different algorithms can be worked out from the basic time and order recursions, depending on the selection of internal variables and on whether the emphasis is on a priori or a posteriori error calculations and on time or order recurrence relations.

There are general rules to design efficient and robust algorithms, some of which can be stated as follows:

Minimize the number of state variables.
Give precedence to time recurrence whenever possible.
Make sure that reliable control variables are available to check the proper functioning of the adaptive filter.

Accordingly, the lattice algorithm given below avoids using the cross-correlation variable $K_i(n)$ and is based on a direct time updating of the reflection coefficients [4].

Substituting the time recursion (8.36) and the error energy updating equation into definition (8.15) gives

$$[E_{ai}(n+1) - e_{ai}(n+1)\varepsilon_{ai}(n+1)]k_{a(i+1)}(n) = K_{i+1}(n+1) - \varepsilon_{ai}(n+1)e_{bi}(n) \tag{8.43}$$

Hence, using again (8.15) at time $n+1$ gives

$$k_{a(i+1)}(n+1) - k_{a(i+1)}(n) + \frac{\varepsilon_{ai}(n+1)}{E_{ai}(n+1)}[e_{bi}(n) - k_{a(i+1)}(n)e_{ai}(n+1)] \tag{8.44}$$

Now, the time recursion (8.13) yields

$$k_{a(i+1)}(n+1) = k_{a(i+1)}(n) + \frac{\varepsilon_{ai}(n+1)e_{b(i+1)}(n+1)}{E_{ai}(n+1)} \tag{8.45}$$

which provides a time updating for the reflection coefficients involving only error variables.

The same procedure, using time recursions (8.35) and (8.12), leads to the time updating equation for the other reflection coefficients in the prediction section:

$$k_{b(i+1)}(n+1) = k_{b(i+1)}(n) + \frac{\varepsilon_{bi}(n)e_{a(i+1)}(n+1)}{E_{bi}(n)} \tag{8.46}$$

For the filter section, let

$$k_{fN}(n) = \frac{K_{fN}(n)}{E_{bN}(n)} \tag{8.47}$$

The same procedure again, using time recursion (8.21) and the filter error energy updating relation, yields

$$k_{fi}(n+1) = k_{fi}(n) + \frac{\varepsilon_{bi}(n+1)e_{i+1}(n+1)}{E_{bi}(n+1)} \tag{8.48}$$

The computational organization of the lattice adaptive filter based on a priori errors is given in Figure 8.2. The initial conditions are

$$\begin{aligned} &e_{bi}(0) = k_{ai}(0) = k_{bi}(0) = k_{fi}(0) = 0, \quad 0 \leqslant i \leqslant N-1 \\ &\varphi_i(0) = 1, \quad E_{ai}(0) = W^N E_0, \quad E_{bi}(0) = W^{N-i}E_0, \quad 0 \leqslant i \leqslant N-1 \end{aligned} \tag{8.49}$$

and the FORTRAN program is given in Annex 8.1.

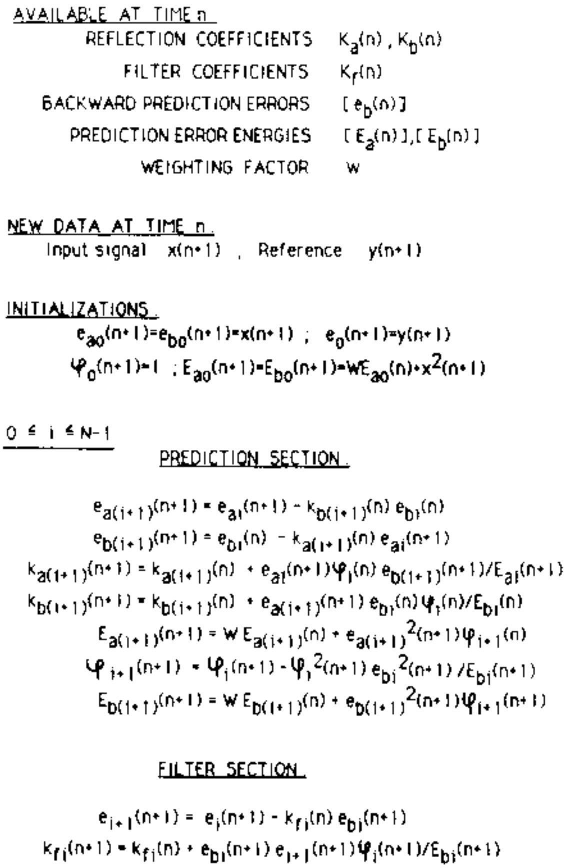
A lattice algorithm based on a posteriori errors can be derived in a similar manner.

The computational complexity of the algorithm in Figure 8.2 amounts to $16N+2$ multiplications and $3N$ divisions in the form of inverse calculations. About $7N$ memories are required.

The block diagram of the adaptive filter is shown in Figure 8.3. The filter section is sometimes called the ladder section, and the complete system is called a lattice-ladder adaptive filter.

Since it has been shown in Section 5.3 that the backward prediction errors are uncorrelated, the filter can be viewed as a decorrelation processor followed by a set of N first-order separate adaptive filters.

In the presence of stationary signals, the two sets of lattice coefficients, like the forward and backward prediction coefficients, take on similar values in the steady state. Algorithms which use only one set of coefficients, and thus are potentially simpler, can be obtained with normalized variables [5].



AVAILABLE AT TIME n

REFLECTION COEFFICIENTS $K_a(n)$, $K_b(n)$

FILTER COEFFICIENTS $K_f(n)$

BACKWARD PREDICTION ERRORS $[e_b(n)]$

PREDICTION ERROR ENERGIES $[E_a(n)],[E_b(n)]$

WEIGHTING FACTOR w

NEW DATA AT TIME n.

Input signal $x(n+1)$, Reference $y(n+1)$

INITIALIZATIONS.

$$e_{a0}(n+1)=e_{b0}(n+1)=x(n+1)\ ;\quad e_0(n+1)=y(n+1)$$

$$\varphi_0(n+1)=1\ ;\ E_{a0}(n+1)=E_{b0}(n+1)=WE_{a0}(n)+x^2(n+1)$$

$0 \le i \le N-1$

PREDICTION SECTION.

$$e_{a(i+1)}(n+1) = e_{ai}(n+1) - k_{b(i+1)}(n)\, e_{bi}(n)$$

$$e_{b(i+1)}(n+1) = e_{bi}(n) - k_{a(i+1)}(n)\, e_{ai}(n+1)$$

$$k_{a(i+1)}(n+1) = k_{a(i+1)}(n) + e_{ai}(n+1)\varphi_i(n)\, e_{b(i+1)}(n+1)/E_{ai}(n+1)$$

$$k_{b(i+1)}(n+1) = k_{b(i+1)}(n) + e_{a(i+1)}(n+1)\, e_{bi}(n)\varphi_i(n)/E_{bi}(n)$$

$$E_{a(i+1)}(n+1) = W E_{a(i+1)}(n) + e_{a(i+1)}^2(n+1)\varphi_{i+1}(n)$$

$$\varphi_{i+1}(n+1) = \varphi_i(n+1) - \varphi_i^2(n+1)\, e_{bi}^2(n+1)/E_{bi}(n+1)$$

$$E_{b(i+1)}(n+1) = W E_{b(i+1)}(n) + e_{b(i+1)}^2(n+1)\varphi_{i+1}(n+1)$$

FILTER SECTION.

$$e_{i+1}(n+1) = e_i(n+1) - k_{fi}(n)\, e_{bi}(n+1)$$

$$k_{fi}(n+1) = k_{fi}(n) + e_{bi}(n+1)\, e_{i+1}(n+1)\varphi_i(n+1)/E_{bi}(n+1)$$

FIG. 8.2 Computational organization of a lattice adaptive filter.

8.5. NORMALIZED LATTICE ALGORITHMS

The variable $K_i(n)$ defined by equation (8.4) and updated by (8.35) corresponds to a cross-correlation calculation. A true cross-correlation coefficient, with magnitude range $[-1, 1]$, is obtained by scaling that variable with the energies of the error signals, which leads to the normalized variable, $k_i(n)$, defined by

$$k_{i+1}(n) = \frac{K_{i+1}(n)}{\sqrt{E_{ai}(n)E_{bi}(n-1)}} \tag{8.50}$$

A time recurrence relation can be derived, using (8.36) to get

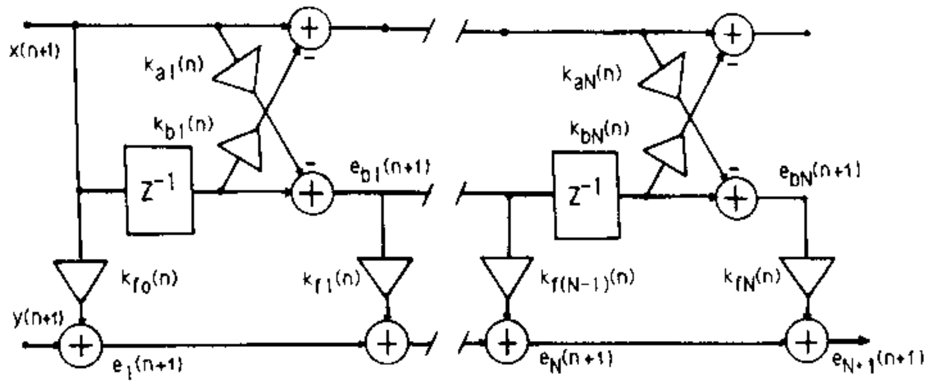


FIG. 8.3 The lattice adaptive filter.

$$k_{i+1}(n+1) = [E_{ai}(n+1)]^{-1/2}[WK_{i+1}(n) + e_{ai}(n+1)\varepsilon_{bi}(n)][E_{bi}(n)]^{-1/2} \tag{8.51}$$

In order to make $k_{i+1}(n)$ appear in (8.51), we have to consider the ratios of the error energies. The time updating equations can be rewritten as

$$W\frac{E_{ai}(n)}{E_{ai}(n+1)} = 1 - \frac{e_{ai}^2(n+1)}{E_{ai}(n+1)}\varphi_i(n) \tag{8.52}$$

and

$$W\frac{E_{bi}(n-1)}{E_{bi}(n)} = 1 - \frac{\varepsilon_{bi}^2(n)}{E_{bi}(n)}\frac{1}{\varphi_i(n)} \tag{8.53}$$

If the normalized forward prediction error is defined by

$$e_{nai}(n+1) = e_{ai}(n+1)\sqrt{\frac{\varphi_i(n)}{E_{ai}(n+1)}} = \varepsilon_{ai}(n+1)[\varphi_i(n)E_{ai}(n+1)]^{-1/2} \tag{8.54}$$

and the normalized backward prediction error by

$$e_{nbi}(n) = \varepsilon_{bi}(n)[\varphi_i(n)E_{bi}(n)]^{-1/2} \tag{8.55}$$

then, the recurrence equation (8.51) becomes

$$k_{i+1}(n+1) = k_{i+1}(n)[(1 - e_{nai}^2(n+1))(1 - e_{nbi}^2(n))]^{1/2} + e_{nai}(n+1)e_{nbi}(n) \tag{8.56}$$

Clearly, with the above definitions, the normalized error variables are intermediates between a priori and a posteriori errors.

To obtain an algorithm, we must derive recursions for the normalized prediction errors. The order recursion (8.14a) for forward a posteriori errors can be rewritten as

$$\varphi_{i+1}(n)e_{a(i+1)}(n+1) = \varphi_i(n)e_{ai}(n+1) - \frac{K_{i+1}(n+1)}{E_{bi}(n)}\varepsilon_{bi}(n) \tag{8.57}$$

Substitution of the normalized errors in that expression leads to

$$e_{na(i+1)}(n+1) = \left[\frac{E_{ai}(n+1)}{E_{a(i+1)}(n+1)}\right]^{1/2}\left[\frac{\varphi_i(n)}{\varphi_{i+1}(n)}\right]^{1/2} e_{nai}(n+1) \tag{8.58}$$

The normalized variables can be introduced into the order recursions (8.8) and (8.33) to yield

$$E_{a(i+1)}(n+1) = E_{ai}(n+1)[1 - k_{i+1}^2(n+1)] \tag{8.59}$$

and

$$\varphi_{i+1}(n) = \varphi_i(n)[1 - e_{nbi}^2(n)] \tag{8.60}$$

Substituting into (8.58) leads to the final form of the time recurrence relation for the normalized forward prediction error:

$$\begin{aligned} e_{na(i+1)}(n+1) &= [1 - k_{i+1}^2(n+1)]^{-1/2}[1 - e_{nbi}^2(n)]^{-1/2} \\ &\quad \times (e_{nai}(n+1) - k_{i+1}(n+1)e_{nbi}(n)) \end{aligned} \tag{8.61}$$

The same method can be applied to backward prediction errors. Order recursion (8.14b) is expressed in terms of normalized variables by

$$\begin{aligned} e_{nb(i+1)}(n+1) &= \left[\frac{E_{bi}(n)}{E_{b(i+1)}(n+1)}\right]^{1/2}\left[\frac{\varphi_i(n)}{\varphi_{i+1}(n+1)}\right]^{1/2} \\ &\quad \times (e_{nbi}(n) - k_{i+1}(n+1)e_{nai}(n+1)) \end{aligned} \tag{8.62}$$

Equation (8.11) for the energy can be written

$$E_{b(i+1)}(n+1) = E_{bi}(n)[1 - k_{i+1}^2(n+1)] \tag{8.63}$$

An equation relating $\varphi_{i+1}(n+1)$ and $\varphi_i(n)$ can be obtained with the help of adaptation gain recurrence relation (6.73) in Chapter 6, which yields

$$\varphi_{i+1}(n+1) = \varphi_i(n) - \frac{\varepsilon_{ai}^2(n+1)}{E_{ai}(n+1)} \tag{8.64}$$

and thus

$$\varphi_{i+1}(n+1) = \varphi_i(n)[1 - e_{nai}^2(n+1)] \tag{8.65}$$

Hence the final form of the time recurrence relation for the normalized backward prediction error is

$$e_{nb(i+1)}(n+1) = [1 - k_{i+1}^2(n+1)]^{-1/2}[1 - e_{nai}^2(n+1)]^{-1/2} \times (e_{nbi}(n) - k_{i+1}(n+1)e_{nai}(n+1)) \tag{8.66}$$

Finally equations (8.56), (8.61), and (8.66) make an algorithm for the normalized lattice adaptive predictor.

Normalized variables can be introduced as well in the filter section. The normalized filter output errors are defined by

$$e_{ni}(n) = e_i(n)\left[\frac{\varphi_i(n)}{E_i(n)}\right]^{1/2} = \varepsilon_i(n)[\varphi)_i(n)E_i(n)]^{-1/2} \tag{8.67}$$

Then order recursion (8.25) yields

$$e_{n(i+1)}(n) = \left[\frac{E_i(n)}{E_{i+1}(n)}\right]^{1/2}\left[\frac{\varphi_i(n)}{\varphi_{i+1}(n)}\right]^{1/2}\left(e_{ni}(n) - \frac{K_{fi}(n)}{\sqrt{E_i(n)\varphi_i(n)}}\frac{\varepsilon_{bi}(n)}{E_{bi}(n)}\right) \tag{8.68}$$

Defining the normalized coefficients by

$$k_{fi}(n) = \frac{K_{fi}(n)}{\sqrt{E_{bi}(n)E_i(n)}} \tag{8.69}$$

We can write the order recursion (8.27) for error energies as

$$E_{i+1}(n) = E_i(n)[1 - k_{fi}^2(n)] \tag{8.70}$$

Substituting (8.60) and (8.70) into (8.68) leads to the order recursion for filter output errors:

$$e_{n(i+1)}(n) = [1 - k_{fi}^2(n)]^{-1/2}[1 - e_{nbi}^2(n)]^{-1/2}[e_{ni}(n) - k_{fi}(n)e_{nbi}(n)] \tag{8.71}$$

Now the normalized coefficients themselves have to be calculated. Once the normalized variables are introduced in time recursion (8.41), one gets

$$k_{fi}(n+1) = \left[\frac{E_{bi}(n)}{E_{bi}(n+1)}\right]^{1/2}\left[\frac{E_i(n)}{E_i(n+1)}\right]^{1/2} Wk_{fi}(n) + e_{nbi}(n+1)e_{ni}(n+1) \tag{8.72}$$

The time recursion for filter output error energies can be rewritten as

$$W\frac{E_i(n)}{E_i(n+1)} = 1 - \frac{e_i^2(n+1)\varphi_i(n+1)}{E_i(n+1)} = 1 - e_{ni}^2(n+1) \tag{8.73}$$

Substituting (8.53) and (8.73) into (8.72), we obtain the time recursion for the normalized filter coefficients:

$$k_{fi}(n+1) = k_{fi}(n)[1 - e_{nbi}^2(n+1)]^{1/2}[1 - e_{ni}^2(n+1)]^{1/2} + e_{nbi}(n+1)e_{ni}(n+1) \tag{8.74}$$

which completes the normalized lattice filter algorithm. The initializations follow the definition of the normalized variables, which implies for the prediction

$$e_{na0}(n+1) = \frac{x(n+1)}{\sqrt{E_{a0}(n+1)}} = e_{nb0}(n+1) \tag{8.75}$$

and for the filter section

$$e_{n0}(n+1) = \frac{y(n+1)}{\sqrt{E_{f0}(n+1)}}, \quad E_{f0}(n+1) = WE_{f0}(n) + y^2(n+1) \tag{8.76}$$

Other initializations are in accordance with (8.49), with the additional equation $E_{f0}(0) = E_0$.

The computational organization of the normalized lattice adaptive filter is shown in Figure 8.4, and a filter section is depicted in Figure 8.5.

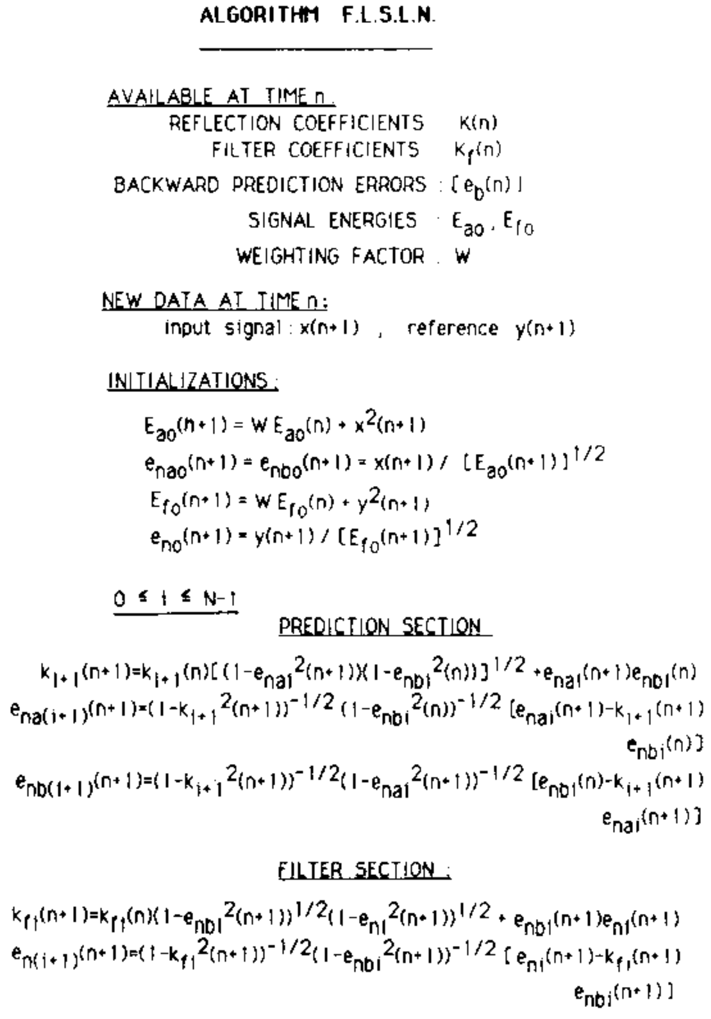
In spite of its conciseness, this algorithm requires more calculations than its unnormalized counterpart. The prediction section needs $10N + 2$ multiplications, $2N + 1$ divisions, and $3N + 1$ square roots, whereas the filter section requires $6N + 2$ multiplications, $N + 1$ divisions, and $2N + 1$ square roots. Altogether, the algorithm complexity amounts to $16N + 4$ multiplications, $3N + 2$ divisions, and $5N + 2$ square roots. An important point is the need for square-root calculations, which are a significant burden in implementations. The number of memories needed is about $3N$.

Overall, the normalized algorithm may be attractive for handling nonstationary signals with fixed-point arithmetic because it has a built-in magnitude scaling of its variables. The resulting robustness to roundoff errors is enhanced by the fact that only one set of prediction coefficients is calculated [5–7].

The main advantage of the lattice approach is that it constitutes a set of N adaptive filters with all orders from 1 to N. Therefore it may be interesting to calculate the coefficients and adaptation gains of the corresponding transversal filters.

8.6. CALCULATION OF TRANSVERSAL FILTER COEFFICIENTS

The conversion from lattice to transversal prediction coefficients is performed with the help of the order recursions (8.7) and (8.10), which can be written as



ALGORITHM F.L.S.L.N.

AVAILABLE AT TIME n.

REFLECTION COEFFICIENTS $K(n)$

FILTER COEFFICIENTS $K_f(n)$

BACKWARD PREDICTION ERRORS : $[e_b(n)]$

SIGNAL ENERGIES : E_{ao}, E_{fo}

WEIGHTING FACTOR . W

NEW DATA AT TIME n:

input signal : $x(n+1)$, reference $y(n+1)$

INITIALIZATIONS:

$E_{ao}(n+1) = W E_{ao}(n) + x^2(n+1)$

$e_{nao}(n+1) = e_{nbo}(n+1) = x(n+1) / [E_{ao}(n+1)]^{1/2}$

$E_{fo}(n+1) = W E_{fo}(n) + y^2(n+1)$

$e_{no}(n+1) = y(n+1) / [E_{fo}(n+1)]^{1/2}$

$0 \leq i \leq N-1$

PREDICTION SECTION

$k_{i+1}(n+1)=k_{i+1}(n)[(1-e_{nai}^2(n+1))(1-e_{nbi}^2(n))]^{1/2} + e_{nai}(n+1)e_{nbi}(n)$

$e_{na(i+1)}(n+1)=(1-k_{i+1}^2(n+1))^{-1/2}(1-e_{nbi}^2(n))^{-1/2}[e_{nai}(n+1)-k_{i+1}(n+1)\,e_{nbi}(n)]$

$e_{nb(i+1)}(n+1)=(1-k_{i+1}^2(n+1))^{-1/2}(1-e_{nai}^2(n+1))^{-1/2}[e_{nbi}(n)-k_{i+1}(n+1)\,e_{nai}(n+1)]$

FILTER SECTION:

$k_{fi}(n+1)=k_{fi}(n)(1-e_{nbi}^2(n+1))^{1/2}(1-e_{ni}^2(n+1))^{1/2} + e_{nbi}(n+1)e_{ni}(n+1)$

$e_{n(i+1)}(n+1)=(1-k_{fi}^2(n+1))^{-1/2}(1-e_{nbi}^2(n+1))^{-1/2}[e_{ni}(n+1)-k_{fi}(n+1)\,e_{nbi}(n+1)]$

FIG. 8.4 Computational organization of the normalized lattice adaptive filter.

$$A_{i+1}(n+1) = \begin{bmatrix} A_i(n+1) \\ 0 \end{bmatrix} - k_{b(i+1)}(n+1)\begin{bmatrix} B_i(n) \\ -1 \end{bmatrix}$$

$$B_{i+1}(n+1) = \begin{bmatrix} 0 \\ B_i(n) \end{bmatrix} - k_{a(i+1)}(n+1)\begin{bmatrix} -1 \\ A_i(n+1) \end{bmatrix} \tag{8.77}$$

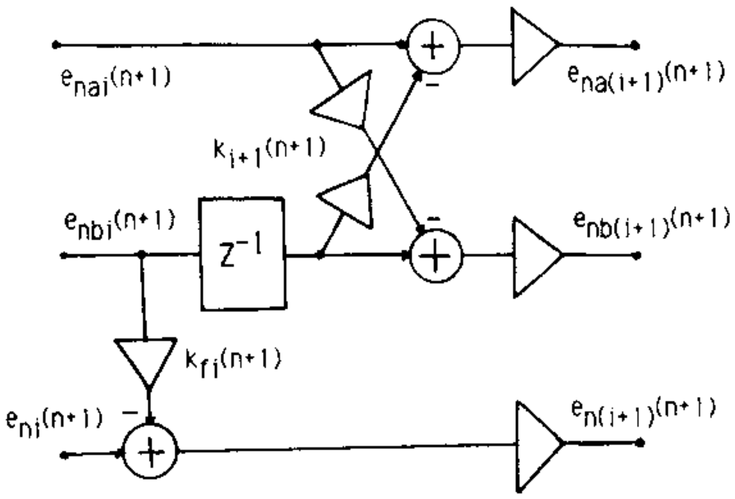


FIG. 8.5 A section of normalized lattice adaptive filter.

The coefficients of the transversal filters can be recursively computed from order 2 to order N. However, it may be more convenient to replace $B_i(n)$ by $B_i(n+1)$ in order to deal with a set of variables homogeneous in time.

Substituting the time recursions of the forward and backward prediction coefficients into (8.77) and adding the order recursion (8.32) for the adaptation gain, the conversion set becomes

$$\begin{aligned}
A_{i+1}(n+1) &= \begin{bmatrix} A_i(n+1) \\ 0 \end{bmatrix} - k_{b(i+1)}(n+1)\begin{bmatrix} B_i(n+1) \\ -1 \end{bmatrix} \\
&\quad + k_{b(i+1)}(n+1)e_{bi}(n+1)\begin{bmatrix} G_i(n+1) \\ 0 \end{bmatrix} \\
B_{i+1}(n+1) &= \begin{bmatrix} 0 \\ B_i(n+1) \end{bmatrix} - e_{bi}(n+1)\begin{bmatrix} 0 \\ G_i(n+1) \end{bmatrix} \\
&\quad - k_{a(i+1)}(n+1)\begin{bmatrix} -1 \\ A_i(n+1) \end{bmatrix} \\
G_{i+1}(n+1) &= \begin{bmatrix} G_i(n+1) \\ 0 \end{bmatrix} + \frac{\varepsilon_{bi}(n+1)}{E_{bi}(n+1)}\begin{bmatrix} -B_i(n+1) \\ 1 \end{bmatrix}
\end{aligned} \tag{8.78}$$

The corresponding flow graph is shown in Figure 8.6. The implementation requires some care in handling the coefficient vectors. The operator Z^{-1} in the flow graph represents a one-element shift of an $(i+1)$-element vector in an $(i+2)$-element register. The input of the first section, corresponding to

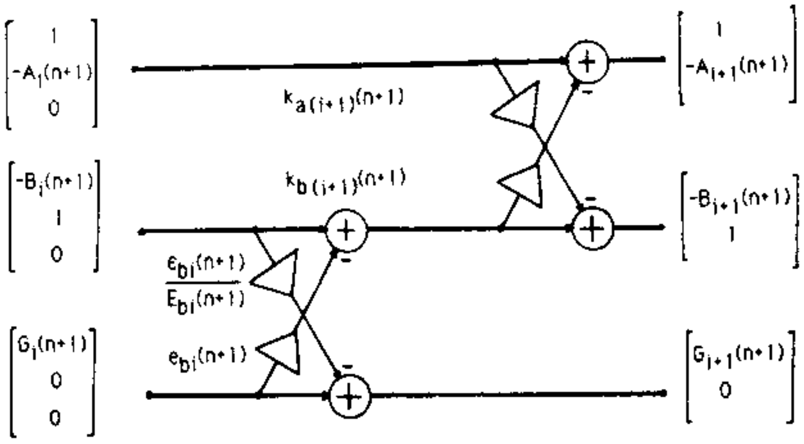


FIG. 8.6 A section for calculating the transversal predictor coefficients.

$i = 0$, is (1,1,0), and the output of the last section, corresponding to $i = N - 1$, yields the prediction coefficients.

The transversal coefficients $H_i(n)$ of the filter section are obtained recursively from equation (8.23).

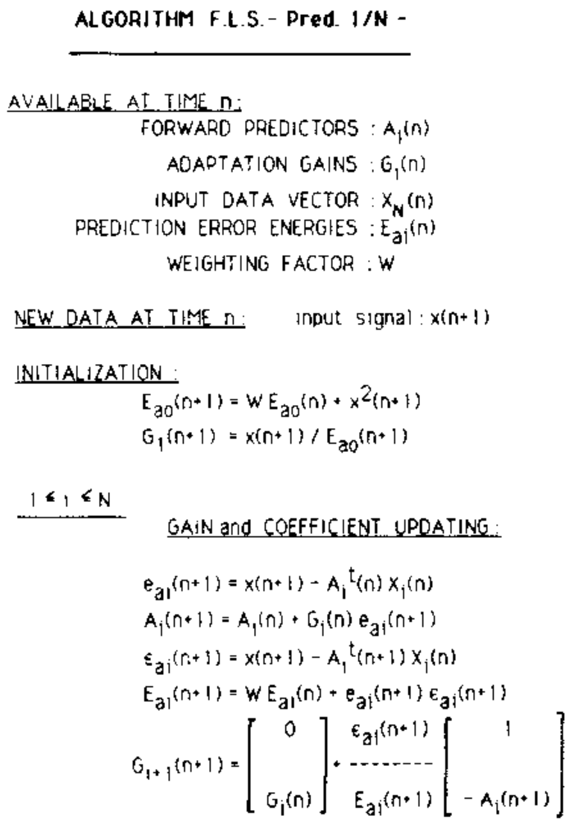
Note that a similar computational complexity can be obtained through the direct calculation of the forward prediction transversal coefficients. Suppose we want to calculate all the coefficients from order 1 to order N: since the adaptation gain updating can use only forward variables, backward variables are no longer needed, and the algorithm obtained by simplifying the algorithms in Chapter 6 is shown in Figure 8.7. The computational complexity is about $2N(N+1)$ multiplications and N divisions per time sample.

8.7. MULTIDIMENSIONAL LATTICE ALGORITHMS

The lattice algorithms for scalar input and reference signals can be extended to vector signals. As shown in Section 7.5, for a K-element input signal the prediction errors become a K-element vector, the lattice coefficients and error energies become $K \times K$ matrices, and the prediction error ratios remain scalars. It is sufficient to change accordingly the equations in Figure 8.2 to obtain a multidimensional lattice algorithm.

As an example, let us consider the 2-D input signals $\chi^t(n) = [x_1(n), x_2(n)]$ and scalar reference $y(n)$, the notations being as in Section 7.4.

The $2i$-element filter coefficient vector $H_{2i}(n)$ which minimizes the cost function



ALGORITHM F.L.S.- Pred. 1/N -

AVAILABLE AT TIME n:

FORWARD PREDICTORS	: $A_i(n)$
ADAPTATION GAINS	: $G_i(n)$
INPUT DATA VECTOR	: $X_N(n)$
PREDICTION ERROR ENERGIES	: $E_{ai}(n)$
WEIGHTING FACTOR	: W

NEW DATA AT TIME n: input signal : $x(n+1)$

INITIALIZATION :

$$E_{a0}(n+1) = W E_{a0}(n) + x^2(n+1)$$
$$G_1(n+1) = x(n+1) / E_{a0}(n+1)$$

$1 \leq i \leq N$

GAIN and COEFFICIENT UPDATING:

$$e_{ai}(n+1) = x(n+1) - A_i^t(n) X_i(n)$$
$$A_i(n+1) = A_i(n) + G_i(n) e_{ai}(n+1)$$
$$\varepsilon_{ai}(n+1) = x(n+1) - A_i^t(n+1) X_i(n)$$
$$E_{ai}(n+1) = W E_{ai}(n) + e_{ai}(n+1) \varepsilon_{ai}(n+1)$$
$$G_{i+1}(n+1) = \begin{bmatrix} 0 \\ \\ G_i(n) \end{bmatrix} + \frac{\varepsilon_{ai}(n+1)}{E_{ai}(n+1)} \begin{bmatrix} 1 \\ \\ -A_i(n+1) \end{bmatrix}$$

FIG. 8.7 Direct calculation of forward prediction transversal coefficients for orders 1 to N.

$$J_{2i}(n) = \sum_{p=1}^{n} W^{n-p}[y(p) - H_{2i}(n)X_{2i}^t(p)]^2 \tag{8.79}$$

satisfies the relation

$$R_{2i}(n)H_{2i}(n) = r_{2i}(n)$$

The same relation at order $i + 1$ is

$$\sum_{p=1}^{n} W^{n-p}\begin{bmatrix} X_{2i}(p) \\ \chi(p-i) \end{bmatrix}[X_{2i}^t(p), \chi^t(p-i)]H_{2(i+1)}(n) = \begin{bmatrix} r_{2i}(n) \\ \sum_{p=1}^{n} W^{n-p} y(p)\chi(p-i) \end{bmatrix} \tag{8.80}$$

The partitioning of the matrix $R_{2(i+1)}(n)$ leads to

$$\begin{bmatrix} R_{2i}(n) & r_{2i}^b(n) \\ \cdots\cdots & \cdots\cdots \\ [r_{2i}^b(n)]^t & \sum_{p=1}^{n} W^{n-p} \chi(p-i)\chi^t(p-i) \end{bmatrix} \begin{bmatrix} H_{2i}(n) \\ \cdots \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} r_{2i}(n) \\ \cdots\cdots \\ [r_{2i}^b(n)]^t H_{2i}(n) \end{bmatrix} \tag{8.81}$$

Hence

$$H_{2(i+1)}(n) = \begin{bmatrix} H_{2i}(n) \\ 0 \\ 0 \end{bmatrix} + R_{2(i+1)}^{-1}(n) \begin{bmatrix} 0 \\ K_i(n) \end{bmatrix} \tag{8.82}$$

with

$$K_i(n) = \sum_{p=1}^{n} W^{n-p} y(p)[\chi(p-i) - B_{2i}^t(n) X_{2i}(p)] \tag{8.83}$$

the $2i \times 2$ backward prediction coefficient matrix being expressed by

$$B_{2i}(n) = R_{2i}^{-1}(n) r_{2i}^b(n)$$

The backward prediction matrix equation is

$$R_{2(i+1)}(n) \begin{bmatrix} -B_{2i}(n) \\ I_2 \end{bmatrix} = \begin{bmatrix} 0 \\ E_{2bi}(n) \end{bmatrix} \tag{8.84}$$

where $E_{2bi}(n)$ is the 2×2 backward error energy matrix. From the output error definition

$$e_{i+1}(n+1) = y(n+1) - H_{2(i+1)}^t(n) X_{2(i+1)}(n+1) \tag{8.85}$$

the following order recursion is obtained, from (8.82) and (8.84):

$$e_{i+1}(n+1) = e_i(n+1) - K_i^t(n) E_{2bi}^{-1}(n) e_{2bi}(n+1) \tag{8.86}$$

It is the extension of (8.24) to the 2-D input signal case.

Consequently, for each order, the filter output error is computed with the help of the backward prediction errors, which are themselves computed recursively with the forward prediction errors. The filter block diagram is in Figure 8.3.

Simplifications can be made when the lengths of the two corresponding adaptive filters, as shown in Figure 7.1, are different, say M and $N + M$. Then the overall filter appears as a combination of a 1-D section with N stages and a 2-D section with M stages. These two different sections have to be carefully interconnected. It is simpler to make the 1-D section come first [8].

At order N, the elements of the forward prediction error vector are

$$\begin{aligned} e_{aN}^{(1)}(n+1) &= x_1(n+1) - [x_1(n), \dots, x_1(n+1-N)]A_{11}(n) \\ e_{aN}^{(2)}(n+1) &= x_2(n+1) - [x_1(n), \dots, x_1(n+1-N)]A_{21}(n) \end{aligned} \tag{8.87}$$

and those of the backward prediction error vector are

$$\begin{aligned} e_{bN}^{(1)}(n+1) &= x_1(n+1-N) - [x_1(n+1), \dots, x_1(n+2-N)]B_{11}(n) \\ e_{bN}^{(2)}(n+1) &= x_2(n+1) - [x_1(n+1), \dots, x_1(n+2-N)]A_{21}(n) \end{aligned} \tag{8.88}$$

where the prediction coefficient matrices are partitioned as

$$A_{2N}(n) = \begin{bmatrix} A_{11}(n) & A_{12}(n) \\ A_{21}(n) & A_{22}(n) \end{bmatrix}, \quad B_{2N}(n) = \begin{bmatrix} B_{11}(n) & B_{12}(n) \\ B_{21}(n) & B_{22}(n) \end{bmatrix}$$

Clearly, $e_{aN}^{(1)}(n+1)$ and $e_{bN}^{(1)}(n+1)$ are the forward and backward prediction errors of the 1-D process, as expected. They are provided by the last stage of the 1-D lattice section. The two other errors $e_{aN}^{(2)}(n+1)$ and $e_{bN}^{(2)}(n+1)$ turn out to be the outputs of 1-D filters whose reference signal is $x_2(n)$.

Therefore, they can be computed recursively as shown in Section 8.2, using equatins similar to (8.24) for the error signal and (8.41) for the cross-correlation estimation; the initial values are $e_{a0}^{(2)}(n+1) = e_{b0}^{(2)}(n+1) = x_2(n+1)$.

Definition (8.88) and the procedure in Section 8.2 lead to

$$e_{aN}^{(2)}(n+1) = e_{a(N-1)}^{(2)}(n+1) - \frac{K_{a(N-1)}(n)}{E_{b(N-1)}(n-1)} e_{b(N-1)}^{(1)}(n) \tag{8.89}$$

and for a posteriori errors

$$\varepsilon_{aN}^{(2)}(n+1) = \varepsilon_{a(N-1)}^{(2)}(n+1) - \frac{K_{a(N-1)}(n+1)}{E_{b(N-1)}(n)} \varepsilon_{b(N-1)}^{(1)}(n) \tag{8.90}$$

with

$$K_{a(N-1)}(n+1) = WK_{a(N-1)}(n) + \varepsilon_{b(N-1)}^{(1)}(n) e_{a(N-1)}^{(2)}(n+1) \tag{8.91}$$

We can obtain $e_{bN}^{(2)}(n+1)$ directly from the forward prediction errors, because it has the same definitoin as $e_{aN}^{(2)}(n+1)$ except for the shift of the data vector. Therefore the order recursive procedure can be applied again to yield

$$\varepsilon_{bN}^{(2)}(n+1) = \varepsilon_{a(N-1)}^{(2)}(n+1) - \frac{K_{bN}(n+1)}{E_{a(N-1)}(n+1)} \varepsilon_{a(N-1)}^{(1)}(n+1) \tag{8.92}$$

and

$$K_{bN}(n+1) = WK_{bN}(n) + \varepsilon_{a(N-1)}^{(2)}(n+1) e_{a(N-1)}^{(1)}(n+1) \tag{8.93}$$

Finally, the 1-D/2-D lattice filter for nonuniform lengths is depicted in Figure 8.8.

The above technique can be extended to higher dimensions to produce cascades of lattice sections with increasing dimensions.

8.8. BLOCK PROCESSING

The algorithms considered so far assume that updating the coefficient is needed whenever new data become available. However, in a number of applications the coefficient values are used only when a set or block of n data has been received. Updating at each time index is adequate in that case too, but it may require an excessive number of arithmetic operations.

The problem is to compute the N elements of the coefficient vector $H_N(n)$ which minimizes the cost function $J_N(n)$ given by

$$J_N(n) = \sum_{p=1}^{n} [y(p) - H_N^t(n) X_N(p)]^2 \tag{8.94}$$

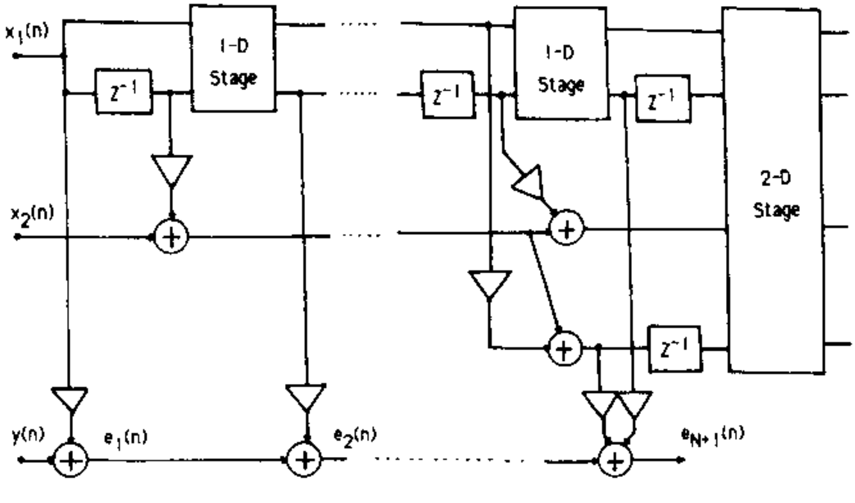


FIG. 8.8 The 1-D/2-D lattice structure for nonuniform length filters.

where the block length n is usually significantly larger than the filter order N.

As seen before, the solution is

$$H_N(n) = \left[\sum_{p=1}^{n} X_N(p)X_N^t(p)\right]^{-1} \sum_{p=1}^{n} y(p)X_N(p) \tag{8.95}$$

If the initial data vector is null, $X(0) = 0$, it is recommended to carry out the calculation up to the time $n + N - 1$ while taking $X(n+1) = 0$, because the input signal AC matrix so obtained is Toeplitz. The computation of its N different elements requires nN multiplications and additions. The same amount is required by the cross-correlation vector. Once the correlation data have been calculated, the prediction coefficients are obtained through the Levinson algorithm given in Section 5.4, which requires N divisions and $N(N+1)$ multiplications. The filter coefficients are then calculated recursively through (8.23), where the variable $k_{fi}(n)$ $(0 \leqslant i \leqslant N-1)$ can be obtained directly from its definition (8.22), because the cross-correlation coefficients $r_{yxN}(n)$ are available; again N divisions are required as well as $N(N-1)$ multiplications. The corresponding FORTRAN subroutine is given in Annex 5.1.

For arbitrary initial vectors or for zero initial input vector and summation stopping at n, the AC matrix estimation in (8.95) is no longer Toeplitz, and order recursive algorithms can be worked out to obtain the coefficient vector $H_N(n)$. They begin with calculating the cross-correlation variables $K_i(n)$ and $K_{fi}(n)$ from their definitions (8.3) and (8.22), and they use the recursions given in the previous sections. They are relatively complex, in terms of number of equations [9]. For example, the computational requirements are about $nN + 4.5N^2$ for prediction and $2nN + 5.5N^2$ for the filter, in the algorithm given in [10].

8.9. GEOMETRICAL DESCRIPTION

The procedure used to derive the FLS algorithms in the previous chapters consists of matrix manipulations. A vector space viewpoint is introduced below, which provides an opportunity to unify the derivations of the different algorithms [3, 11–14].

The vector space considered is defined over real numbers, and its vectors have M elements; it is denoted R^M. The vector of the N most recent input data is

$$X_M(n) = [x(n), x(n-1), \ldots, x(1), 0, \ldots, 0]^t$$

and the data matrix containing the N most recent input vectors is

$$X_{MN}(n) = [X_M(n), X_M(n-1), \ldots, X_M(n+1-N)]$$

The column vectors form a basis of the corresponding N dimensional subspace.

An essential operator is the projection matrix, which for a subspace U is defined by

$$P_U = U(U^tU)^{-1}U^t \tag{8.96}$$

It is readily verified that $P_UU = U$. If U and Y are vectors, P_UY is the projection of Y on U as shown in Figure 8.9. The following are useful relationships:

$$P_U^t = P_U, \quad (P_UY)^t(P_UY) = Y^tP_UY, \quad P_UP_U = P_U \tag{8.97}$$

The orthogonal projection operator is defined by

$$P_U^o = I - U(U^tU)^{-1}U^t \tag{8.98}$$

Indeed the sum of the projections is the vector itself:

$$P_UY + P_U^oY = Y \tag{8.99}$$

Let us consider as a particular case the operator $P^o_{\{X_{MN}(n-1)\}}$ applied to the M-element vector $X_M(n)$:

$$\begin{aligned} P^o_{\{X_{MN}(n-1)\}}X_M(n) = X_M(n) - X_{MN}(n-1) \\ \times [X_{MN}^t(n-1)X_{MN}(n-1)]^{-1}X_{MN}^t(n-1)X_M(n) \end{aligned}$$

The product of the last two terms is

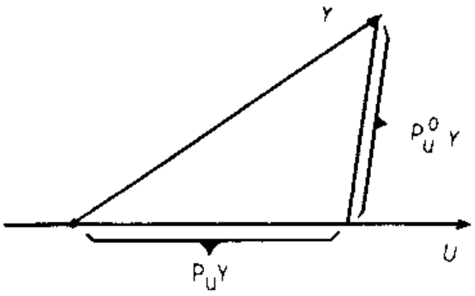


FIG. 8.9 Projection operator.

$$X_{MN}^t(n-1)X_M(n)$$
$$= \begin{bmatrix} x(n-1) & x(n-2), & \cdots & x(2) & x(1) & 0 & \cdots & 0 \\ x(n-2) & x(n-3) & \cdots & x(1) & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & & & & & \\ x(n+1-N) & x(n-N) & \cdots & \cdots & \cdots & & \cdots & 0 \end{bmatrix} \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(1) \\ \vdots \\ 0 \end{bmatrix} \tag{8.100}$$

With the relations of the previous chapters, we have

$$X_{MN}^t(n-1)X_M(n) = \sum_{p=1}^{n} X_N(p-1)x(p) = r_N^a(n) \tag{8.101}$$

Similarly

$$[X_{MN}^t(n-1)X_{MN}(n-1)] = \sum_{p=1}^{n} X_N(p)X_N^t(p) = R_N(n-1) \tag{8.102}$$

Hence

$$[X_{MN}^t(n-1)X_{MN}(n-1)]^{-1}X_{MN}^t(n-1)X_M(n) = R_N^{-1}(n-1)r_N^a(n) = A_N(n) \tag{8.103}$$

Thus, the M-element forward prediction error vector is obtained:

$$P_{\{X_{MN}(n-1)\}}^o X_M(n) = e_M(n) = X_M(n) - X_{MN}(n-1)A_N(n) \tag{8.104}$$

It is such that

$$e_M^t(n)e_M(n) = \sum_{p=1}^{n}[x(p) - X^t(p-1)A_N(n)]^2 = E_{aN}(n) \tag{8.105}$$

and the forward prediction error energy is the squared norm of the orthogonal projection of the new vector $X_M(n)$ on the subspace spanned by the N most recent input vectors.

Finally, the operator $P_{\{X_{MN}(n-1)\}}^o$, denoted in a shorter form by $P_x^o(n-1)$, is a prediction operator. Note that the first element in the error vector $e_M(n)$ is the a posteriori forward prediction error

$$\varepsilon_{aN}(n) = x(n) - X_N^t(n-1)A_N(n) \tag{8.106}$$

It is useful to define a dual prediction operator $Q_x^o(n-1)$ which produces the a priori forward prediction error as the first element of the error vector. It is defined by

$$Q_U^o = I - U(U^tS^tSU)^{-1}U^tS^tS \tag{8.107}$$

where S is the $M \times M$ shifting matrix

$$S = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

The product of S with a time-dependent $M \times 1$ vector shifts this vector one sample back. Therefore one has

$$SX_M(n) = X_M(n-1), \quad SX_{MN}(n) = X_{MN}(n-1) \tag{8.108}$$

The $M \times M$ matrix S^tS is a diagonal matrix with 0 as the first diagonal element and 1's as the other elements.

As before, the operator $Q^o_{\{X_{MN}(n-1)\}}$ is denoted by $Q_X^o(n-1)$. Let us consider the product $Q_X^o(n-1)X_M(n)$. Clearly,

$$X_{MN}^t(n-1)S^tSX_M(n) = \sum_{p=1}^{n-1} X_N(p-1)x(p) = r_N^a(n-1) \tag{8.109}$$

and

$$X_{MN}^t(n-1)S^tSX_{MN}(n-1) = \sum_{p=1}^{n-2} X_N(p)X_N^t(p) = R_N(n-2) \tag{8.110}$$

which leads to

$$Q_X^o(n-1)X_M(n) = e'_M(n) = X_M(n) - X_{MN}(n-1)A_N(n-1) \tag{8.111}$$

The first element of the vector $e'_M(n)$ is

$$e_{aN}(n) = x(n) - X_N^t(n-1)A_N(n-1) \tag{8.112}$$

That operation itself can be expressed in terms of operators. In order to single out the first element of a vector, we use the so-called $M \times 1$ pinning vector Π:

$$\Pi = [1, 0, \ldots, 0]^t$$

Therefore the forward prediction errors are expressed by

$$\varepsilon_{aN}(n) = \Pi^t P_X^o(n-1)X_M(n) = X_M^t(n)P_X^o(n-1)\Pi \tag{8.113}$$

and

$$e_{aN}(n) = \Pi^t Q_X^o(n-1)X_M(n) = X_M^t(n)Q_X^o(n-1)\Pi \tag{8.114}$$

These two errors are related by the factor $\varphi_N(n-1)$, which is expressed in terms of the space operators as follows:

$$\Pi^t P_X^o(n)\Pi = 1 - X_N^t(n)R_N^{-1}(n)X_N(n) = \varphi_N(n)$$

Hence, we have the relationship beween P_X^o and Q_X^2:

$$\Pi^t Q_X^o = (\Pi^t P_X^o \Pi)^{-1}\Pi^t P_X^o \tag{8.115}$$

Fast algorithms are based on order and time recursions, and it is necessary to determine the relationship between the corresponding projection operators.

8.10. ORDER AND TIME RECURSIONS

Incrementing the filter order amounts to adding a vector to the matrix $X_{MN}(n)$ and thus expanding the dimensionality of the associated subspace. A new projection operator is obtained.

Assume U is a matrix and V a vector; then for any vector Y the following equality is valid for the orthogonal projection operators:

$$P_U^o Y = P_{U,V}^o Y + P_U^o V(V^t P_U^o V)^{-1} V^t P_U^o Y \tag{8.116}$$

It is the combined projection theorem illustrated in Figure 8.10. Clearly, if U and V are orthogonal—that is, $P_U V = 0$ and $P_U^o V = V$—then equation (8.116) reduces to

$$P_U^o Y = P_{U,V}^o Y + P_V Y \tag{8.117}$$

For the operators one gets

$$P_{U,V} = P_U - P_U^o V(V^t P_U^o V)^{-1} V^t P_U^o \tag{8.118}$$

In Chapter 6, order recursions are involved in the adaptation gain updating process. The adaptation gain $G_N(n)$ can be viewed as the first vector of an $N \times M$ matrix

$$G_X = (X_{MN}^t X_{MN})^{-1} X_{MN}^t \tag{8.119}$$

and

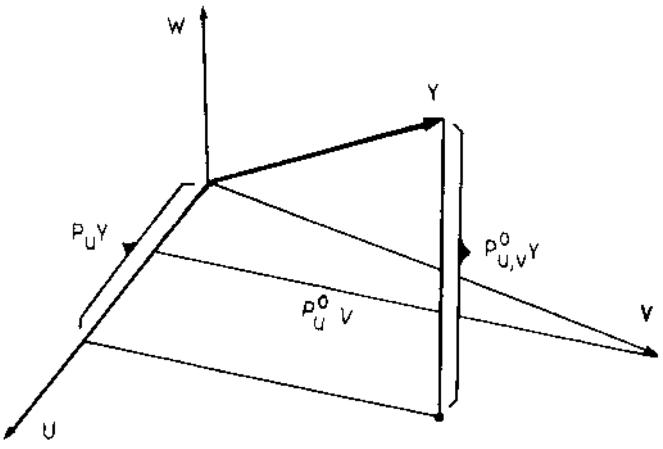


FIG. 8.10 Illustration of the combined projection theorem.

$$G_N(n) = R_N^{-1}(n) X_N(n) = G_X(n)\Pi \tag{8.120}$$

In order to determine the operator associated with an expanded subspace, it is useful to notice that $X_{MN} G_X$ is the projection operator P_X. For U a matrix and V a vector, equations (8.118) and (8.99) lead to

$$[U, V]G_{U,V} = [U, V]\begin{bmatrix} G_U \\ 0 \end{bmatrix} + (V - UG_U V)(V^t P^o_U V)^{-1} V^t P^o_U$$

Hence

$$G_{U,V} = \begin{bmatrix} G_U \\ 0 \end{bmatrix} + \begin{bmatrix} -G_U V \\ 1 \end{bmatrix} (V^t P^o_U V)^{-1} V^t P^o_U \tag{8.121}$$

Similarly, if U and V are permuted, one gets

$$G_{V,U} = \begin{bmatrix} 0 \\ G_U \end{bmatrix} + \begin{bmatrix} 1 \\ -G_U V \end{bmatrix} (V^t P^o_U V)^{-1} V^t P^o_U \tag{8.122}$$

These are the basic order recursive equations exploited in the algorithms in Chapter 6.

The time recursions can be described in terms of geometrical operators as well. Instead of adding a column to the data matrix $X_{MN}(n)$, we add a row to the matrix $X_{MN}(n-1)$ after a backward shift. Let us consider the matrices

$$S^t SX = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ x(n-1) & x(n-2) & \cdots & x(n-M) \\ x(n-2) & x(n-3) & \cdots & x(n-1-M) \\ \vdots & \vdots & & \vdots \\ x(1) & 0 & & 0 \end{bmatrix} \tag{8.123}$$

$$\Pi\Pi^t X = \begin{bmatrix} x(n) & x(n-1) & \cdots & x(n+1-N) \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Clearly, their column vectors are orthogonal and they span orthogonal subspaces. The following equality is valid for the projectors:

$$P_X = P_{S^t SX} + P_{\Pi\Pi^t X} \tag{8.124}$$

Due to the definition of the shifting matrix, we have

$$S^t SS^t = S^t, \quad SS^t S = S, \quad S^t S + \Pi\Pi^t = I \tag{8.125}$$

Thus

$$P_{S^t SX} = S^t P_{SX} S \tag{8.126}$$

The time recursions useful in the algorithms involve the error signals, and, therefore, the orthogonal projectors are considered. Definition (8.98) yields

$$S^t P^o_{SX} S = S^t S - S^t SX(X^t S^t SX)^{-1} X^t S^t S \tag{8.127}$$

As time advances, the objective is to update the orthogonal projection operator associated with the data matrix $X_{MN}(n)$, and an equation linking P^o_{SX} and P^o_X is looked for. Definitions (8.123) lead to

$$X^t X = X^t S^t SX + X^t \Pi\Pi^t X \tag{8.128}$$

Now, using the matrix inversion lemma (6.24) of Chapter 6, one gets

$$(X^t S^t SX)^{-1} = (X^t X)^{-1} + (X^t X)^{-1} X^t \Pi(\Pi^t P^o_Y \Pi)^{-1} \Pi^t X(X^t X)^{-1} \tag{8.129}$$

Substituting into (8.127) yields, in concise form,

$$S^t P^o_{SX} S = S^t S[P^o_X - P^o_X \Pi(\Pi^t P^o_X)^{-1} \Pi^t P^o_X] S^t S$$

Using the property (8.125), we obtain the time recursion equation

$$P^o_X = S^t P^o_{SX} S + P^o_X \Pi(\Pi^t P^o_X \Pi)^{-1} \Pi^t P^o_X \tag{8.130}$$

To illustrate that result, let us postmultiply both sides by the reference signal vector $Y_M(n)$, defined by

$$Y)M(n) = [y(n), y(n-1), \dots, y(1), 0, \dots, 0]^t$$

Clearly

$$P_X^o Y_M(n) = \begin{bmatrix} y(n) - X_N^t(n)H_N(n) \\ y(n-1) - X_N^t(n-1)H_N(n) \\ \vdots \\ y(1) - X_N^t(1)H_N(n) \\ 0 \\ \vdots \\ 0 \end{bmatrix}; H_N(n) = R_N^{-1}(n)r_{yxN}(n) \tag{8.131}$$

The same operation at time $n-1$ leads to

$$S^t P_{SX}^o S Y_M(n) = \begin{bmatrix} 0 \\ y(n-1) - X_N^t(n-1)H_N(n-1) \\ \vdots \\ y(1) - X_N^t(1)H_N(n-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{8.132}$$

Now

$$\Pi^t P_X^o \Pi = 1 - X_N^t(n)R_N(n)X_N(n) = \varphi_N(n) \tag{8.133}$$

and the last term of the right side of the recursion equation (8.130) is

$$P_X^o \Pi(\Pi^t P_X^o \Pi)^{-1} \Pi^t P_X^o Y_M(n) = \begin{bmatrix} \varphi_N(n) \\ -X_N^t(n-1)G_N(n) \\ \vdots \\ -X_N^t(1)G_N(n) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \frac{\varepsilon_N(n)}{\varphi_N(n)} \tag{8.134}$$

The filter coefficient time updating equation

$$H_N(n) = H_N(n-1) + \frac{G_N(n)\varepsilon_N(n)}{\varphi_N(n)}$$

leads to the verifications of the result

$$P_X^o Y_M(n) = S^t P_{SX}^o S Y_M(n) + P_X^o \Pi(\Pi^t P_X^o \Pi)^{-1} \Pi^t P_X^o Y_M(n) \tag{8.135}$$

It is important to consider the application of the time updating formula (8.130) to the gain operator G_X. Definition (8.119) and equation (8.115) lead to

$$I - XG_X = S^t(I - SXG_{SX})S + (I - XG_X)\Pi\Pi^t Q_X^o \tag{8.136}$$

Then, the properties of the shifting matrix S and pinning vector Π yield, after simplification, the following time updating formula for the gain operator:

$$G_X = G_{SX}S + G_X\Pi\Pi^t Q_X^o \tag{8.137}$$

With the geometrical operators presented so far, all sorts of algorithms can be derived.

8.11. UNIFIED DERIVATION OF FLS ALGORITHMS

The FLS algorithms are obtained by applying the basic order and time recursions with different choices of signal matrices and vectors.

In order to derive the transversal algorithm based on a priori errors and presented in Section 6.4, one takes $U = X_{MN}(n-1)$ and $V = X_M(n)$. The following equalities are readily verified:

$$\begin{aligned} V^t P_U^o \Pi &= \varepsilon_{aN}(n), \quad & V^t Q_U^o \Pi &= e_{aN}(n) \\ G_U V &= A_N(n), \quad & VtP_U^o V &= E_{aN}(n) \end{aligned} \tag{8.138}$$

Therefore, the time updating of the forward prediction coefficients is obtained by postmultiplying (8.137) by $X_M(n)$. The time and order updating equation for the adaptation gain is obtained by postmultiplying (8.122) by Π. The recursion for the error energy $E_{aN}(n)$ corresponds to premultiplying the time updating formula (8.130) by $X_M^t(n)$ and postmultiplying by $X_M(n)$. The backward variables are obtained in the same manner as the forward variables, $X_M(n-N)$ replacing $X_M(n)$.

The algorithm based on all prediction errors and given in Section 6.5 uses the error ratio $\varphi_N(n) = \Pi^t P_X^o(n)\Pi$, which is calculated through a time and order updating equation.

Postmultiplying (8.118) by Π and premultiplying by Π^t yields after simplification

$$\varphi_{N+1}(n) = \varphi_N(n-1) - \frac{\varepsilon_{aN}^2(n)}{E_{aN}(n)} \tag{8.139}$$

Now, substituting (6.49) of Chapter 6 and the time recursion for the error energy into (8.139) gives

$$\varphi_{N+1}(n) = \varphi_N(n-1)\frac{E_{aN}(n-1)}{E_{aN}(n)} \tag{8.140}$$

A similar relation can be derived for the backward prediction error energies, taking $U = X_{MN}(n)$ and $V = X_M(n-N)$. It is

$$\varphi_N(n) = \frac{\varphi_{N+1}(n)E_{bN}(n)}{E_{bN}(n-1)} \tag{8.141}$$

In order to get a sequential algorithm, we must calculate the updated energy $E_{bN}(n)$. Postmultiplying (8.121) by $U = X_{MN}(n)$ and $V = X_M(n-N)$ yields the adaptation gain recursion (6.75) of Chapter 6, which shows that the last element of $G_{N+1}(n)$ is

$$m(n) = \frac{\varepsilon_{bN}(n)}{E_{bN}(n)}$$

Hence

$$\varphi_N(n) = \frac{\varphi_{N+1}(n)}{1 - e_{bN}(n)m(n)} \tag{8.142}$$

Finally, the error ratio $\varphi_N(n)$ can be updated by equations (8.140) and (8.142). The algorithm is completed by taking into account the backward coefficient time updating equation and rewriting (6.75) of Chapter 6 as

$$G_{N+1}(n) = \begin{bmatrix} G_N(n)[1 - e_{bN}(n)m(n)] \\ 0 \end{bmatrix} + \begin{bmatrix} -B_N(n-1) \\ 1 \end{bmatrix} m(n) \tag{8.143}$$

Dividing both sides by $\varphi_{N+1}(n)$ and substituting (8.142) lead to

$$\frac{G_{N+1}(n)}{\varphi_{N+1}(n)} = \begin{bmatrix} \dfrac{G_N(n)}{\varphi_N(n)} \\ 0 \end{bmatrix} + \begin{bmatrix} -B_N(n-1) \\ 1 \end{bmatrix} \frac{m(n)}{\varphi_{N+1}(n)} \tag{8.144}$$

Therefore the a priori adaptation gain $G'_N(n) = G_N(n)/\varphi_N(n)$ can be used instead of $G_N(n)$, and the algorithm of Section 6.5 is obtained. In Figure 6.5 $\varphi_N^{-1}(n)$ is updated.

The geometrical approach can also be employed to derive the lattice structure equations. The lattice approach consists of computing the forward and backward prediction errors recursively in order. The forward a posteriori prediction error for order i is

$$\varepsilon_{ai}(n) = X_M^t(n)P_{U,V}^o\Pi \tag{8.145}$$

where

$$U = X_{M(i-1)}(n-1), \quad V = X_M(n-i)$$

Substituting projection equation (8.118) into (8.145) yields

$$\varepsilon_{ai}(n) = \varepsilon_{a(i-1)}(n) - X_M^t(n) P_U^o V (V^t P_U^o V)^{-1} V^t P_U^o \Pi \tag{8.146}$$

The factors in the second term on the right side are

$$V^t P_U^o \Pi = \varepsilon_{b(i-1)}(n-1), \qquad V^t P_U^o V = E_{b(i-1)}(n-1)$$
$$X_M^t(n) P_U^o V = \sum_{p=1}^{n} x(p)x(p-i) - A_{i-1}^t(n) R_{i-1}(n-1) B_{i-1}(n-1) = K_i(n) \tag{8.147}$$

Hence

$$\varepsilon_{ai}(n) = \varepsilon_{a(i-1)}(n) - \frac{K_i(n)}{E_{b(i-1)}(n-1)} \varepsilon_{b(i-1)}(n-1)$$

which is equation (8.14a). The corresponding backward equation (8.14b) is

$$\varepsilon_{bi}(n) = X_M^t(n-i) P_{U,V}^o \Pi \tag{8.148}$$

with $U = X_{M(i-1)}(n-1)$, $V = X_M(n)$.

The a priori equations are obtained by using the operator $Q_{U,V}^o$ instead of $P_{U,V}^o$.

Algorithms with nonzero initial conditions in either transversal or lattice structures are obtained in the same manner; block processing algorithms are also obtained similarly.

8.12. SUMMARY AND CONCLUSION

The flexibility of LS techniques has been further illustrated by the derivation of order recurrence relationships for prediction and filter coefficients and their combination with time recurrence relationships to make fast algorithms. The lattice structures obtained are based on reflection coefficients which represent a real-time estimation of the cross-correlation between forward and backward prediction errors. A great many different algorithms can be worked out by varying the types and arrangements of the recursive equations. However, if the general rules for designing efficient and robust algorithms are enforced, the actual choice reduces to a few options, and an algorithm based on direct time updating of the reflection coefficients has been presented.

The LS variables can be normalized in such a way that time and order recursions be kept. For the lattice structure, a concise and robust algorithm can be obtained, which uses a single set of reflection coefficients. However, the computational complexity is significantly increased by the square-root operations involved.

The lattice approach can be extended to M-D signals with uniform and nonuniform filter lengths. The 1-D/2-D case has been investigated.

Overall, the lattice approach requires more computations than the transversal method. However, besides its academic interest, it provides all the filters with orders from 1 to N and can be attractive in those applications where the filter order is not known beforehand and when the user can be satisfied with reflection coefficients.

A vector space viewpoint provides an elegant description of the fast algorithms and their computational mechanisms. The calculation of errors corresponds to a projection operation in a signal vector space. Order and time updating formulae can be worked out for the projection operators. By choosing properly the matrices and vectors for these projection operators, one can derive all sorts of algorithms in a simple and concise way. The method applies to transversal or lattice structures, with or without initial conditions, with exponential or sliding time windows. Overall, the geometric description offers a unified derivation of the FLS algorithms.

EXERCISES

1. The signal

   $$x(n) = \sin(n\pi/3) + \sin(n\pi/4)$$

   is fed to an order 4 adaptive FIR lattice predictor. Give the values of the four optimal reflection coefficients. The weighting factor in the adaptive algorithm is $W = 0.98$; give upper bounds for the magnitudes of the variables $K_i(n)$. What are their steady-state values?
2. Give the computational organization of an FLS lattice algorithm in which the cross-correlation estimation variables $K_i(n)$ are updated in time and the a priori and a posteriori forward and backward prediction errors are calculated. Count the multiplications, divisions, and memories needed.
3. Consider the filter section in the block diagram in Figure 8.3. Calculate the coefficient $h_i(n)$ of an order 1 LS adaptive filter whose input sequence is $e_{bi}(n+1)$ and whose reference signal is $e_i(n+1)$. Compare with the expression of $k_{fi}(n)$ and comment on the difference.
4. Derive the lattice algorithm with direct time updating of the coefficients as in Figure 8.2, but with a posteriori errors. *Hint*: Use the error ratios $\varphi_i(n)$ to get the a posteriori errors and then find the updating equations for the reflection coefficients.
5. Let X_N be an N-element vector such that $0 < X_N^t X_N < 1$. Prove the identities

$$(I_N - X_N X_N^t)^{1/2} = I_N - \frac{1-(1-X_N^t X_N)^{1/2}}{X_N^t X_N} X_N X_N^t$$
$$(I_N - X_N X_N^t)^{-1/2} = I_N + \frac{(1-X_N^t X_N)^{-1/2}-1}{X_N^t X_N} X_N X_N^t$$
$$(I_N - X_N X_N^t)^{1/2} X_N = X_N(1-X_N^t X_N)^{1/2}$$

Show that the square roots of these matrices can be obtained with $\frac{N^2}{2} + \frac{N}{2}$ multiplications and one square-root calculation.

6. In order to derive normalized versions of the transversal FLS algorithms, we define the normalized variables

$$e_{an}(n) = \frac{e_a(n)}{E_a(n-1)}, \quad \varepsilon_{an}(n) = \frac{\varepsilon_a(n)}{E_a(n)}$$

Define normalized versions of the prediction coefficients and the adaptation gain. Give the corresponding time updating relationships. Give the updating equations for the error energies. Give the computational organization of a normalized transversal FLS algorithm and compare the complexity with that of the standaard algorithm.

7. In order to visualize the vector space approach, consider the case where $M = 3$, $N = 2$ and the signal input sequence is

$$x(n) = 0, \quad n \leqslant 0$$
$$x(1) = 4, \quad x(2) = 2, \quad x(3) = 4$$

In the 3-D space $(0_x, 0_y, 0_z)$, draw the vectors $X_M(1), X_M(2), X_M(3)$, and the vector Π. Calculate and show the vector $P_X^o(2)X_M(3)$. Show the forward and backward prediction errors at time $n = 3$. Show how the adaptation gains $G_2(2)$ and $G_2(3)$ are formed.

8. Find the order updating equation for the prediction opertor Q. Use it to geometrically derive the lattice equations for a priori prediction errors.

ANNEX 8.1 FLS ALGORITHM FOR A PREDICTOR IN LATTICE STRUCTURE

```
      SUBROUTINE FLSL(N,X,EAB,EA,EB,KA,KB,W,IND)
C
C     COMPUTES THE PARAMETERS OF A LATTICE PREDICTOR
C     N   = FILTER ORDER
C     X   = INPUT SIGNAL
C     EAB = VECTOR OF BACKWARD PREDICTION ERRORS
      (A PRIORI)
```

```
C      EA   = VECTOR OF FORWARD PREDICTION ERROR ENERGIES
C      EB   = VECTOR OF BACKWARD PREDICTION ERROR ENERGIES
C      KA,KB = LATTICE COEFFICIENTS
C      W    = WEIGHTING FACTOR
C      IND  = TIME INDEX
C
       REAL KA,KB
       DIMENSION
       EAB(1),EA(1),EB(1),KA(1),KB(1),EAV(15),PHI(15)
C
C      INITIALIZATION
C
C      IF(IND.GT.1)GOTO30
       X1=0
       EO=1.
       DO20I=1,N
       EAB(I)=0.
       EA(I)=EO*W**N
       EB(I)=EO*W**(N-I)
       KA(I)=0.
       KB(I)=0.
       PHI(I)=1.
   20  CONTINUE
   30  CONTINUE
C
C      ORDER : 1
C
       EO1=EO
       EO=W*EO+X*X
       EAV(1)=X-KB(1)*X1
       EAB1=EAB(1)
       EAB(1)=X1-KA(1)*X
       KA(1)=KA(1)+X*EAB(1)/EO
       KB(1)=KB(1)+EAV(1)*X1/EO1
       EA(1)*W*EA(1)+EAV(1)*EAV(1)*PHI(1)
       PHI1=PHI(1)
       PHI(1)=1-X*X/EO
       EB1=EB(1)
       EB(1)=W*EB(1)+EAB(1)*EAB(1)*PHI(1)
       X1=X
C
C      ORDERS > 1
C
       N1=N-1
       DO50I=1,N1
```

```
      EAV(I+1)=EAV(I)-KB(I+1)*EAB1
      EAB2=EAB(I+1)
      EAB(I+1)=EAB1-KA(I+1)*EAV(I)
      KA(I+1)=KA(I+1)+EAV(I)*PHI1*EAB(I+1)/EA(I)
      KB(I+1)=KB(I+1)+EAV(I+1)*EAB1*PHI1/EB1
      EA(I+1)=W*EA(I+1)+EAV(I+1)*EAV(I+1)*PHI(I+1)
      PHI1=PHI(I+1)
      PHI(I+1)=PHI(I)*(1-PHI(I)*EAB(I)*EAB(I)/EB(I))
      EB1=EB(I+1)
      EB(I+1)=W*EB(I+1)+EAB(I+1)*EAB(I+1)*PHI(I+1)
      EAB1=EAB2
   50 CONTINUE
      RETURN
      END
```

REFERENCES

1. B. Friedlander, ‘‘Lattice Filters for Adaptive Processing,’’ *Proc. IEEE* **70**, 829–867 (August 1982).
2. J. M. Turner, ‘‘Recursive Least Squares Estimation and Lattice Fitlers,’’ in *Adaptive Filters*, Prentice-Hall, Englewood Cliffs, N.J., 1985, Chap. 5.
3. M. Honig and D. Messerschmitt, ‘‘Recursive Least Squares,’’ in *Adaptive Filters: Structures, Algorithms and Applications*, Kluwer Academic, Boston, 1984, Chap. 6.
4. F. Ling, D. Manolakis, and J. Proakis, ‘‘Numerically Robust L.S. Lattice-Ladder Algorithms with Direct Updating of the Coefficients,’’ *IEEE Trans.* **ASSP-34**, 837–845 (August 1986).
5. D. Lee, B. Friedlander, and M. Morf, ‘‘Recursive Square Root Ladder Estimation Algorithms,’’ *IEEE Trans.* **ASSP-29**, 627–641 (June 1981).
6. C. Samson and V. U. Reddy, ‘‘Fixed Point Error Analysis of the Normalized Ladder Algorithm,’’ *IEEE Trans.* **ASSP-31**, 1177–1191 (October 1983).
7. P. Fabre and C. Gueguen, ‘‘Improvement of the Fast Recursive Least Squares Algorithms via Normalization: A Comparative Study,’’ *IEEE Trans.* **ASSP-34**, 296–308 (April 1986).
8. F. Lin and J. Proakis, ‘‘A Generalized Multichannel Least Squares Lattice Algorithm Based on Sequential Processing Stages,’’ *IEEE Trans.* **ASSP-32**, 381–389 (April 1984).
9. N. Kaloupsidis, G. Carayannis, and D. Manolakis, ‘‘Efficient Recursive in Order L.S. FIR Filtering and Prediction,’’ *IEEE Trans.* **ASSP-33**, 1175–1187 (October 1985).
10. J. Cioffi, ‘‘The Block-Processing FTF Adaptive Algorithm,’’ *IEEE Trans.* **ASSP-34**, 77–90 (February 1986).
11. C. Samson, ‘‘A Unified Treatment of Fast Algorithms for Identification,’’ *Int. J. Control* **35**, 909–934 (1982).

12. H. Lev-Ari, T. Kailath, and J. Cioffi, ‘‘Least Squares Adaptive Lattice and Transversal Filters: A Unified Geometric Theory,’’ *IEEE Trans.* **IT-30**, 222–236 (March 1984).
13. Jin-Der Wang and H. J. Trussel, ‘‘A Unified Derivation of the Fast RLS Algorithms,’’ *Proc. ICASSP-86, Tokyo*, 1986, pp. 261–264.
14. S. T. Alexander, *Adaptive Signal Processing: Theory and Applications*, Springer-Verlag, New York, 1986.

9
Rotation-Based Algorithms

The rotation operation has a unique feature: namely it does not affect the norm of vectors. Therefore, with algorithms using exclusively rotations, no increases or doublings of the dynamic range can be expected, like those occurring in other algorithms, when data are squared or multiplied by one another. Moreover, it is well known in digital filtering that rotations can provide for optimal processing and ensure stability.

As pointed out in the previous chapter, the family of lattice algorithms essentially relies on a triangular decomposition of the input signal autocorrelation matrix, which is presented in Chapter 5. In fact, a similar decomposition can be performed on the matrix of the input samples, namely the so-called QR decomposition [1].

Using the QR decomposition technique, the solution to a least squares problem is obtained in two steps. First, an orthogonal matrix is used to transform the input signal sample matrix into another matrix whose elements are zeros except for a triangular submatrix. Then, the optimal set of coefficients is derived through solving a linear triangular system of equations [2,3].

In this chapter, the QR decomposition is presented, and fast least squares algorithms are derived for one-dimensional and multidimensional signals. But, to begin with, the basic data rotation operation is analyzed.

9.1. THE ROTATION OPERATION

Let us consider a real data vector X with 3 elements,

$$X = \begin{bmatrix} x_2 \\ x_1 \\ x_0 \end{bmatrix} \tag{9.1}$$

and two rotations R_1 and R_2 defined by the matrices

$$R_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & -\sin\theta_1 \\ 0 & \sin\theta_1 & \cos\theta_1 \end{bmatrix}, \qquad R_2 = \begin{bmatrix} \cos\theta_2 & 0 & -\sin\theta_2 \\ 0 & 1 & 0 \\ \sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix} \tag{9.2}$$

Now let us design rotations R_1 and R_2 to cancel the middle and the top elements respectively in vector X. This is achieved by choosing the rotation angle θ_1 such that

$$\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_0 \end{bmatrix} = \begin{bmatrix} 0 \\ s_1 \end{bmatrix} \tag{9.3}$$

where the variable $s_1 = \sqrt{x_1^2 + x_0^2}$ is the norm of the two-element data vector, which is rotation-invariant. Then

$$\cos\theta_1 = \frac{x_0}{s_1}, \qquad \sin\theta_1 = \frac{x_1}{s_1} \tag{9.4}$$

The same procedure can be applied again, using rotation angle θ_2, leading to:

$$\begin{bmatrix} \cos\theta_2 & -\sin\theta_2 \\ \sin\theta_2 & \cos\theta_2 \end{bmatrix} \begin{bmatrix} x_2 \\ s_1 \end{bmatrix} = \begin{bmatrix} 0 \\ s_2 \end{bmatrix} \tag{9.5}$$

where $s_2 = \sqrt{s_1^2 + x_0^2}$ and

$$\cos\theta_2 = \frac{s_1}{s_2}, \qquad \sin\theta_2 = \frac{x_2}{s_2} \tag{9.6}$$

The new variable s_2 is the norm of the vector X. The entire operation is described by the matrix equation

$$\begin{bmatrix} 0 \\ 0 \\ s_2 \end{bmatrix} = R_2 R_1 \begin{bmatrix} x_2 \\ x_1 \\ x_0 \end{bmatrix} \tag{9.7}$$

Therefore, the norm of any vector can be computed iteratively through a set of rotations, defined by the elements of that vector. In the iterations the cosine contains the previous value of the norm, while the sine introduces the new data. As concerns initial values, the first element must be positive or null. If $x_0 = 0$ and $x_1 \neq 0$, then $\theta_1 = \pi/2$ and all the subsequent angles θ_i satisfy the inequalities

$$-\pi/2 < \theta_i < \pi/2 \tag{9.8}$$

Next, the procedure is applied to the QR decomposition.

9.2. THE QR DECOMPOSITION

In transversal least squares adaptive filtering with N coefficients, the cost function is defined, as in the previous chapters, by

$$J(n) = \sum_{p=0}^{n} W^{n-p}[y(p) - X^t(p)H(n)]^2 \tag{9.9}$$

where $y(p)$ is the reference sequence, $X(p)$ is the vector of the N most recent input samples at time p and W is the weighting factor. The N-coefficient vector at time n, $H(n)$, minimizes the cost function $J(n)$. It is assumed that the input sequence is $x(n) = 0$ for $n < 0$.

Now, the input data matrix $X_N(n)$ is introduced. It is an $(n+1) \times N$-element matrix defined by

$$X_N(n) = \begin{bmatrix} x(n) & x(n-1) & \cdots & x(n+1-N) \\ W^{1/2}x(n-1) & W^{1/2}x(n-2) & \cdots & W^{1/2}x(n-N) \\ \vdots & \vdots & & \vdots \\ W^{n/2}x(0) & 0 & \cdots & 0 \end{bmatrix} \tag{9.10}$$

The first step in deriving algorithms consists of multiplying this matrix by a rotation matrix $Q_N(n)$, in order to reduce the nonzero elements to a triangular submatrix. In fact, there are two possible approaches to reach that goal. The first one is based on forward linear prediction [2]. It yields the following result, where $S_N(n)$ is an upper-left-triangular $N \times N$ matrix:

$$Q_N(n)X_N(n) = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \\ \multicolumn{2}{c}{S_N(n)} & \vdots \\ & \cdots & 0 \end{bmatrix} \tag{9.11}$$

The dual approach exploits backward linear prediction, leading to

$$Q_N(n)X_N(n) = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \\ \vdots & & T_N(n) \\ 0 & \cdots & \cdots \end{bmatrix} \tag{9.12}$$

where $T_N(n)$ is a lower-right-triangular $N \times N$ matrix [3].

Both approaches lead to fast algorithms, with similar computation complexities. However, the backward technique has a number of advantages in terms of simplicity of derivation, interpretation of internal variables, flexibility of implementation and conciseness. It is the one which is adopted here.

As concerns the rotation matrix $Q_N(n)$, later on, it will be shown how it is constructed iteratively from the input data sequence.

Once the above triangularization has been completed, the solution of the least squares problem in adaptive filtering is readily obtained. In fact, the cost function $J(n)$ given by (9.9) is the square of the norm of the error vector expressed by

$$\begin{bmatrix} e_n(n) \\ W^{1/2}e_{n-1}(n) \\ \vdots \\ W^{n/2}e_0(n) \end{bmatrix} = \begin{bmatrix} y(n) \\ W^{1/2}y(n-1) \\ \vdots \\ W^{n/2}y(0) \end{bmatrix} - X_N(n)H(n) \tag{9.13}$$

Now both sides are multiplied by the rotation matrix $Q_N(n)$, and equations (9.11) and (9.12) are used. The optimal coefficient vector is the vector $H(n)$ which cancels the last N elements of the vector obtained after rotation, leading to

$$Q_N(n)\begin{bmatrix} e_n(n) \\ W^{1/2}e_{n-1}(n) \\ \vdots \\ W^{1/2}e_0(n) \end{bmatrix} = \begin{bmatrix} e_{qn}(n) \\ e_{qn-1}(n) \\ \vdots \\ e_{qN}(n) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \end{matrix}}\right\} n+1-N \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} N \end{matrix} \tag{9.14}$$

Since rotations conserve the norms of the vectors, the cost function is equivalently expressed by

$$J(n) = \sum_{p=N}^{n} e_{qp}^2(n) \tag{9.15}$$

Finally, the error elements involved have been obtained by rotating the reference vector by the matrix $Q_N(n)$.

Coming back to the triangular matrix $T_N(n)$ in (9.12), it is used to derive a fast algorithm. We need to make clear the physical interpretation of its elements, and the backward linear prediction problem serves that purpose.

9.3. ROTATIONS IN BACKWARD LINEAR PREDICTION

The backward linear prediction energy at time n is denoted $E_{bN}(n)$ and expressed by

$$E_{bN}(n) = \sum_{p=0}^{n} W^{m-p}[x(p-N) - X^t(p)B_N(n)]^2 \tag{9.16}$$

where $B_N(n)$ is the N-element vector of the transversal backward linear prediction coefficients. Following the same path as in the previous section, one can observe that $E_{bn}(n)$ is the square of the norm of the error vector expressed by

$$\begin{bmatrix} e_{bn}(n) \\ W^{1/2}e_{b\,n-1}(n) \\ \vdots \\ W^{n/2}e_{b0}(n) \end{bmatrix} = \begin{bmatrix} x(n-N) \\ W^{1/2}x(n-1-N) \\ \vdots \\ W^{n/2}x(-N) \end{bmatrix} - X_N(n)B_N(n) \tag{9.17}$$

Again, since rotations preserve the norms of the vectors, an equivalent problem can be formulated after both sides of equation (9.17) have been multiplied by the rotation matrix $Q_N(n)$. In fact, let us extend the input data matrix by one column to the right and perform a multiplication of the matrix $X_{N+1}(n)$ obtained by the rotation matrix $Q_N(n)$. In that operation, the right column of the extended matrix $X_{N+1}(n)$ produces the following vector

$$Q_N(n)\begin{bmatrix} x(n-N) \\ W^{1/2}x(n-1-N) \\ \vdots \\ W^{1/2}x(-N) \end{bmatrix} = \begin{bmatrix} e_{bqn}(n) \\ e_{bqn-1}(n) \\ \vdots \\ e_{bqN+1}(n) \\ X_{bqN}(n) \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} n+1-N \\ \left.\vphantom{\begin{matrix} a \\ a \end{matrix}}\right\} N \end{matrix} \tag{9.18}$$

As in equation (9.14), the top $n+1-N$ elements of the vector obtained are the rotated backward linear prediction errors, which make up the error energy by

$$E_{bN}(n) = \sum_{p=N}^{n} e_{bqp}^2(n) \tag{9.19}$$

The remaining N bottom elements, $X_{bqn}(n)$, are the result of rotating the input data vector, and the backward linear prediction is the solution of the system involving the triangular matrix

$$X_{bqN}(n) = T_N(n)B_N(n) \tag{9.20}$$

Finally, rotating the extended matrix yields

$$Q_N(n)\begin{bmatrix} & x(n-N) \\ & W^{1/2}x(n-1-N) \\ X_N(n) & \vdots \\ & W^{1/2}x(-N) \end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 & e_{bqn}(n) \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & e_{bq\,N+1}(n) \\ & T_N(n) & & X_{bqN}(n) \end{bmatrix} \tag{9.21}$$

Now, from the right-hand-side matrix in the above equation, it is possible to obtain the triangular matrix of order $N+1$, namely $T_{N+1}(n)$. A set of $n-N$ rotations, as those described in Section 9.1, can be used to accumulate the prediction errors, leading to the order recursion (9.22). Obviously, the recursion is valid for any order N and, through induction, the following expression (9.23) is obtained for the triangular matrix.

$$T_{N+1}(n) = \begin{bmatrix} 0 & \cdots & 0 & E_{bN}^{1/2}(n) \\ 0 & \cdots & & \\ \vdots & & & X_{bqN}(n) \\ & T_N(n) & & \end{bmatrix} \tag{9.22}$$

$$T_{N+1}(n) = \begin{bmatrix} 0 & 0 & \cdots & E_{bN}^{1/2}(n) \\ \vdots & \vdots & & \\ 0 & E_{b1}^{1/2}(n) & \cdots & X_{bqN}(n) \\ E_{b0}^{1/2}(n) & X_{bq1}(n) & \cdots & \end{bmatrix} \tag{9.23}$$

Thus, the triangular matrix $T_{N+1}(n)$ is made of the square roots of the backward prediction error energies for its diagonal and it is made of the rotated input data vectors, with orders 1 through N, for the remaining nonzero elements.

At this point, no consideration has been given as to how the rotation matrix $Q_N(n)$ can be obtained. Actually, it can be constructed recursively for each new input data sample. Let us assume that the rotation matrix is available at time $n-1$. It is denoted by $Q_N(n+1)$ and produces the following result, when it multiplies the input data matrix $X_N(n+1)$, defined by equation (9.10):

$$Q_N(n-1)X_N(n-1) = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \\ 0 & \cdots & 0 \\ & T_N(n-1) & \end{bmatrix} \tag{9.24}$$

The input data matrix satisfies the relation

$$X_N(n) = \begin{bmatrix} x(n) & \cdots & x(n+1-N) \\ & W^{1/2}X_N(n-1) & \end{bmatrix} \tag{9.25}$$

Next, the following product is considered:

$$\begin{bmatrix} 1 & 0 \\ 0 & Q_N(n-1) \end{bmatrix} X_N(n) = \begin{bmatrix} x(n) & \cdots & x(n+1-N) \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \\ & W^{1/2}T_N(n-1) & \end{bmatrix} \tag{9.26}$$

and the rotation matrix at time n is obtained from $Q_N(n-1)$ by a set of N rotations, which cancel the first row of the right-hand side of (9.26).

The first of these N rotations is R_1, defined by

$$R_1 = \begin{bmatrix} \cos\theta_1 & 0 & \cdots & 0 & -\sin\theta_1 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ \sin\theta_1 & 0 & \cdots & 0 & \cos\theta_1 \end{bmatrix} \tag{9.27}$$

It cancels $x(n)$ in (9.26) if the angle θ_1 is chosen, using (9.23), such that

$$\cos\theta_1 = \frac{W^{1/2}E_{b0}^{1/2}(n-1)}{E_{b0}^{1/2}(n)}, \qquad \sin\theta_1 = \frac{x(n)}{E_{b0}^{1/2}(n)} \tag{9.28}$$

Actually $E_{b0}(n)$ is the input signal energy. The next element in the first row was $x(n-1)$, but it has been changed by the first rotation. Let us denote it by u. It is cancelled by the rotation matrix R_2 expressed by

$$R_2 = \begin{bmatrix} \cos\theta_2 & 0 & \cdots & -\sin\theta_2 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & & & \vdots & \vdots \\ \sin\theta_2 & 0 & \cdots & \cos\theta_2 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \tag{9.29}$$

Because the rotations lead to the triangular matrix at time n, the angle θ_2 is such that

$$u \sin\theta_2 + W^{1/2} E_{b1}^{1/2}(n-1) \cos\theta_2 = E_{b1}^{1/2}(n) \tag{9.30}$$

and, therefore

$$\cos\theta_2 = \frac{W^{1/2} E_{b1}^{1/2}(n-1)}{E_{b1}^{1/2}(n)}, \qquad \sin\theta_2 = \frac{u}{E_{b1}^{1/2}(n)} \tag{9.31}$$

Now, considering the time recursion for the backward prediction error energy

$$E_{b1}(n) = W E_{b1}(n-1) + e_{b1q}^2(n) \tag{9.32}$$

where $e_{b1q}(n)$ is the rotation linear prediction error, and identifying with (9.31), it becomes clear that the scalar u is actually the rotation linear prediction error $e_{b1q}(n)$.

After N such steps, the operation is complete and the following recursion is obtained:

$$Q_N(n) = R_N \cdots R_2 R_1 \begin{bmatrix} 1 & 0 \\ 0 & Q_N(n-1) \end{bmatrix} \tag{9.33}$$

The angles are defined by

$$\cos\theta_{i+1} = \frac{W^{1/2} E_{bi}^{1/2}(n-1)}{E_{bi}^{1/2}(n)}, \qquad \sin\theta_{i+1} = \frac{e_{biq}(n)}{E_{bi}^{1/2}(n)} \tag{9.34}$$

It is worth emphasizing that such a factorization is possible only because the matrix $T_N(n-1)$ is triangular.

9.4. ROTATION IN FORWARD LINEAR PREDICTION

A dual path to reach the matrix $T_{N+1}(n)$ from $T_N(n)$ is provided by forward linear prediction. The forward linear prediction energy, $E_{aN}(n)$, is defined by

$$E_{aN}(n) = \sum_{p=0}^{n} W^{n-p} [x(p) - X^t(p-1) A_N(n)]^2 \tag{9.35}$$

and it is the square of the norm of the error vector

$$\begin{bmatrix} e_{aN}(n) \\ W^{1/2}e_{aN-1}(n) \\ \vdots \\ W^{n/2}e_{a0}(n) \end{bmatrix} = \begin{bmatrix} x(n) \\ W^{1/2}x(n-1) \\ \vdots \\ W^{n/2}x(0) \end{bmatrix} - \begin{bmatrix} & X_N(n-1) & \\ 0 & \cdots & 0 \end{bmatrix} A_N(n) \tag{9.36}$$

The input data matrix $X_N(n-1)$ has to be completed with a bottom row of zeros, to satisfy the time shift in forward linear prediction, and the rotation matrix used for the triangularization must be modified accordingly. In the following product, the top $n-N$ elements of the resulting vector are the rotated forward prediction errors, the next N elements, denoted $X_{aqN}(n)$, are the rotated input data vector

$$\begin{bmatrix} Q_N(n-1) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(n) \\ W^{1]2}x(n-1) \\ \vdots \\ W^{n/2}x(0) \end{bmatrix} = \begin{bmatrix} e_{aqn}(n) \\ \vdots \\ e_{aqN+1}(n) \\ X_{aqN}(n) \\ W^{n/2}x(0) \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ b \end{matrix}}\right\} n-N \\ \left.\vphantom{\begin{matrix} a \\ b \end{matrix}}\right\} N \\ \end{matrix} \tag{9.37}$$

As mentioned before, the linear prediction vector $A_N(n)$ is the solution of the system

$$X_{aqN}(n) = T_N(n-1)A_N(n) \tag{9.38}$$

Since the objective is to derive the triangular matrix $T_{N+1}(n)$ from $T_N(n-1)$, it is useful to consider the extended matrix $X_{N+1}(n)$. The same rotation as in expression (9.37) above yields

$$\begin{bmatrix} Q_N(n-1) & 0 \\ 0 & 1 \end{bmatrix} X_{N+1}(n) = \begin{bmatrix} e_{aqn}(n) & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ e_{aqN+1}(n) & 0 & \cdots & 0 \\ X_{aqN}(n) & \vdots & T_N(n-1) & \\ W^{n/2}x(0) & 0 & \cdots & 0 \end{bmatrix} \tag{9.39}$$

Now, it is sufficient to accumulate the forward prediction error values $e_{aq\ N+1}(n)$ through $e_{aqn}(n)$ on the bottom element $W^{n/2}x(0)$ to achieve triangularization. In the process, the bottom element of the first column is replaced by the square root of the forward linear prediction error energy. As shown in Section 9.1, this accumulation is performed by a set of appropriate rotations, which have no bearing on the desired least squares algorithm and are not required in explicit form. In fact, the next step consists of the operation

$$Q_\alpha(n)\begin{bmatrix} X_{aqN}(n) & 0 & \cdots & \\ \vdots & & & T_N(n-1) \\ E_{aN}^{1/2}(n) & 0 & \cdots & 0 \end{bmatrix} = T_{N+1}(n) \tag{9.40}$$

The rotation matrix $Q_\alpha(n)$ accumulates the vector $X_{aqN}(n)$ on the bottom element and is the product of N rotations $R_{\alpha i}$ defined by

$$R_{\alpha i} = \begin{bmatrix} 1 & \cdots & \cos\alpha_i & \cdots & -\sin\alpha_i \\ \vdots & \ddots & \vdots & & \\ 0 & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & \sin\alpha_i & \cdots & \cos\alpha_i \end{bmatrix} \tag{9.41}$$

The angles α_i can be determined in a straightforward manner from the elements of the triangular matrices $T_{N+1}(n)$ and $T_N(n-1)$ involved in equation (9.40). From expression (9.23) for a triangular matrix, and considering the diagonal elements, combined with the fact that $T_N(n-1)$ has been completed with zeros in the bottom row, the following relationships appear for 1pi2} 4 N:

$$E_{b\,i+1}^{1/2}(n) = \cos\alpha_{i+1} E_{bi}^{1/2}(n-1) \tag{9.42}$$

In order to illustrate the physical meaning of the angles, it is interesting to compare the above equation with the following order recursion of the normalized lattice algorithm derived in Chapter 8:

$$E_{b\,i+1}^{1/2}(n) = [1 - k_{i+1}^2(n)] E_{bi}^{1/2}(n-1) \tag{9.43}$$

As a consequence, the following equivalence can be established

$$\sin\alpha_i = k_i(n), \qquad \text{1pi2 4 } N \tag{9.44}$$

The procedure to calculate the angles α_i is defined by the simplified relation

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \\ E_{b0}^{1/2}(n) \end{bmatrix} = Q_\alpha(n) \begin{bmatrix} X_{aqN}(n) \\ E_{aN}^{1/2}(n) \end{bmatrix} \tag{9.45}$$

Note the relation obtained by reversing the above equation:

$$\prod_{i=1}^{N} \cos\alpha_i = E_{aN}^{1/2}(n) / E_{b0}^{1/2}(n) \tag{9.46}$$

Additionally, an explicit expression for $\sin\alpha_i$ is

$$\sin\alpha_i = \frac{x_{aqNi}(n)}{\left[E_{aN}(n) + \sum_{j=1}^{N} x_{aqNj}^2(n)\right]^{1/2}} = k_i(n) \tag{9.47}$$

It provides useful information about the rotated input data: the elements $x_{aqNi}(n)$ of the vector obtained by rotation of the input signal samples are linked to the normalized lattice coefficients $k_i(n)$.

9.5. THE FAST LEAST SQUARES QR ALGORITHM

In a recursive algorithm, the rotation matrix $Q_N(n)$ has to be computed from $Q_N(n-1)$. The new input datum at time n, $x(n)$, sits at the upper left corner in the matrix $X_N(n)$. Now, in the triangularization operation, it is processed by the first column of the rotation matrix. Therefore, it is sufficient, in the search for a recursive algorithm, to concentrate on the first column of the matrix.

Picking up the first column in a matrix is expressed in equations by a multiplication of the matrix by a vector with a one at the top and zero elsewhere. Such a vector is sometimes called a pinning vector. Taking into account the recursion already obtained for $Q_N(n)$, namely expression (9.33), the operation leads to

$$Q_N(n)\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix} = \begin{bmatrix}\gamma_N(n)\\0\\\vdots\\0\\G_N(n)\end{bmatrix} \tag{9.48}$$

where the new variables $\gamma_N(n)$ and $G_N(n)$ are defined in terms of the angles involved in the rotations R_1 through R_N.

As concerns $\gamma_N(n)$, it is expressed by

$$\gamma_N(n) = \prod_{i=1}^{N} \cos\theta_i \tag{9.49}$$

and it has a simple physical meaning. From equation (9.14) for example, the a posteriori error can be written as

$$e_n(n) = [e_{qn}(n), e_{qn-1}(n), \ldots, e_{qN}(n), 0\cdots 0]Q_N^t(n)\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix} \tag{9.50}$$

which, using (9.48) yields

$$e_n(n) = \gamma_N(n) e_{qn}(n) \tag{9.51}$$

Thus, the scalar $\gamma_N(n)$ is the ratio of the a posteriori error at time n to the rotation error. It is the square root of the prediction error ratio $\varphi_N(n)$.

The elements of the vector $G_N(n)$ are given by

$$\begin{aligned} g_1(n) &= \sin\theta_1 = x(n)/E_{b0}^{1/2}(n) \\ g_2(n) &= \cos\theta_1 \sin\theta_2 = \varepsilon_{b1}(n)/E_{b1}^{1/2}(n) \\ g_N(n) &= \varepsilon_{b\,N-1}(n)/E_{b\,N-1}^{1/2}(n) \end{aligned} \tag{9.52}$$

where $\varepsilon_{bi}(n)$ is the a posteriori backward prediction error for order i, connected to the rotation error by

$$\varepsilon_{bi}(n) = \gamma_i(n) e_{biq}(n) \tag{9.53}$$

From the algorithmic point of view, the vector $G_N(n)$ provides the link between forward and backward linear prediction and it leads to a fast algorithm.

The forward linear prediction procedure described above can be summarized, combining (9.40) and (9.41), by the following factorization of the rotation matrix $Q_{N+1}(n)$

$$Q_{N+1}(n) = \begin{bmatrix} I_{n-N} & 0 \\ 0 & Q_\alpha(n) \end{bmatrix} Q_{ea}(n) \begin{bmatrix} Q_N(n-1) & 0 \\ 0 & 1 \end{bmatrix} \tag{9.54}$$

where $Q_{ea}(n)$ stands for the combination of rotations used to accumulate the forward prediction error values and produce the prediction error energy $E_{aN}(n)$.

Multiplying both sides by the pinning vector yields

$$\begin{bmatrix} I_{n-N} & 0 \\ 0 & Q_\alpha(n) \end{bmatrix} Q_{ea}(n) \begin{bmatrix} \gamma_N(n-1) \\ 0 \\ \vdots \\ 0 \\ G_N(n-1) \\ 0 \end{bmatrix} = \begin{bmatrix} \gamma_{N+1}(n) \\ 0 \\ \vdots \\ 0 \\ G_{N+1}(n) \end{bmatrix} \tag{9.55}$$

The action of the matrix $Q_{ea}(n)$ is resticted to the last rotation, which projects the top element in the vector, $\gamma_N(n-1)$, at the bottom with the multiplying factor $e_{aqn}(n)/E_{aN}^{1/2}(n)$. Therefore, keeping just the bottom $N+1$ rows in (9.55), one gets

$$Q_\alpha(n) \begin{bmatrix} G_N(n-1) \\ \gamma_N(n-1) e_{aqn}(n)/E_{aN}^{1/2}(n) \end{bmatrix} = \begin{bmatrix} g_{N+1}(n) \\ G_N(n) \end{bmatrix} \tag{9.56}$$

A fast algorithm is obtained by combining equations (9.33), (9.45), (9.48), (9.56). The sequence of operations is shown in Figure 9.1, where the unnecessary variables have been dropped and rotation matrices are restricted to the size $(N+1) \times (N+1)$. To prevent any confusion in notations the matrix product $R_1 R_2 \cdots R_N$ in recursion (9.33) has been denoted $Q_a(n)$ in its restricted form.

The filter section is derived through application of recursion (9.33) to the vector of the reference signal samples. The a posteriori output error, and the a priori error as well, can be provided from the rotation output error denoted $e_q(n+1)$.

The FORTRAN subroutine for the FLS-QR algorithm is given in Annex 9.1, including the initialization of the internal variables.

9.6. IMPLEMENTATION ASPECTS

From an implementation point of view, there are two main motivations to develop rotation-based algorithms. First, the adaptive filter variables can keep the same dynamic range as the input variables. Second, numerical stability can be ensured. The good numerical behavior of the QR-based algorithm stems from the fact that rotations conserve the norms of the vector. Thus, when computing with finite arithmetic, the following constraints must be satisfied for the limited precision variables, for $1 \leqslant i \leqslant n$:

$$\cos^2\theta_i + \sin^2\theta_i \leqslant 1, \qquad \cos^2\alpha_i + \sin^2\alpha_i \leqslant 1 \tag{9.57}$$

Moreover, it is interesting to notice in the algorithm of Figure 9.1 the existence of a reduction mechanism for roundoff errors. The vector $G_N(n)$ is a key variable. Assume that its square norm is increased by Δ at time n. The last rotation in the prediction section yields

$$\gamma_N^2(n) = 1 - \|G_N(n)\|^2 - \Delta \tag{9.58}$$

Then, at time $n+1$, the norm of the updated vector $G_N(n+1)$ is changed approximatively by $\Delta[1 - e_{aq}^2(n+1)/E_a(n+1)]$. A detailed analysis of the stability of QR-based algorithms can be found in reference [4].

As concerns the numerical calculations, they can be derived in two categories, namely rotations and angle calculations. The algorithm of Figure 9.1 comprises $3N$ rotations and $2N$ angle calculations, for the elements of matrices Q_a and Q_α respectively. Each rotation is equivalent to a complex multiplication, while the angle calculations are preferably implemented with the help of look-up tables.

The block diagram of the prediction section is given in Figure 9.2, emphasizing the sets of rotations in the prediction section and the feedback

Available at time n

Forward Transformed input data N-element vector : $Xq(n)$
Transformed reference N-element vector : $Yq(n)$
Transformed unit vector : $G_N(n)$
Forward prediction error energy : $Ea(n)$
Prediction error ratio : $\gamma_N(n)$
Weighting factor : W

New Data at Time n :

Input Signal : $x(n+1)$; Reference : $y(n+1)$

Prediction Section :

$$\begin{vmatrix} e_{aq}(n+1) \\ Xq(n+1) \end{vmatrix} = Qa(n) \begin{vmatrix} x(n+1) \\ W^{1/2}\,Xq(n) \end{vmatrix}$$

$$Ea(n+1) = W\,Ea(n) + e_{aq}{}^{2}(n+1)$$

$$\begin{vmatrix} 0 \\ E_{b0}{}^{1/2}(n+1) \end{vmatrix} = Q\alpha(n+1) \begin{vmatrix} Xq(n+1) \\ Ea^{1/2}(n+1) \end{vmatrix}$$

$$u = \gamma_N(n)\, e_{aq}(n+1) \,/\, Ea^{1/2}(n+1)$$

$$\begin{vmatrix} g_{N+1}(n+1) \\ G_N(n+1) \end{vmatrix} = Q\alpha(n+1) \begin{vmatrix} G_N(n) \\ u \end{vmatrix}$$

$$\begin{vmatrix} \gamma_N(n+1) \\ G_N(n+1) \end{vmatrix} = Qa(n+1) \begin{vmatrix} 1 \\ 0 \end{vmatrix}$$

Filter Section

$$\begin{vmatrix} e_q(n+1) \\ Yq(n+1) \end{vmatrix} = Qa(n+1) \begin{vmatrix} y(n+1) \\ W^{1/2}\,Yq(n) \end{vmatrix}$$

$$\varepsilon(n+1) = \quad \gamma_N(n+1) \qquad e_q(n+1)$$

FIG. 9.1 Computational organization of the fast least squares QR algorithm.

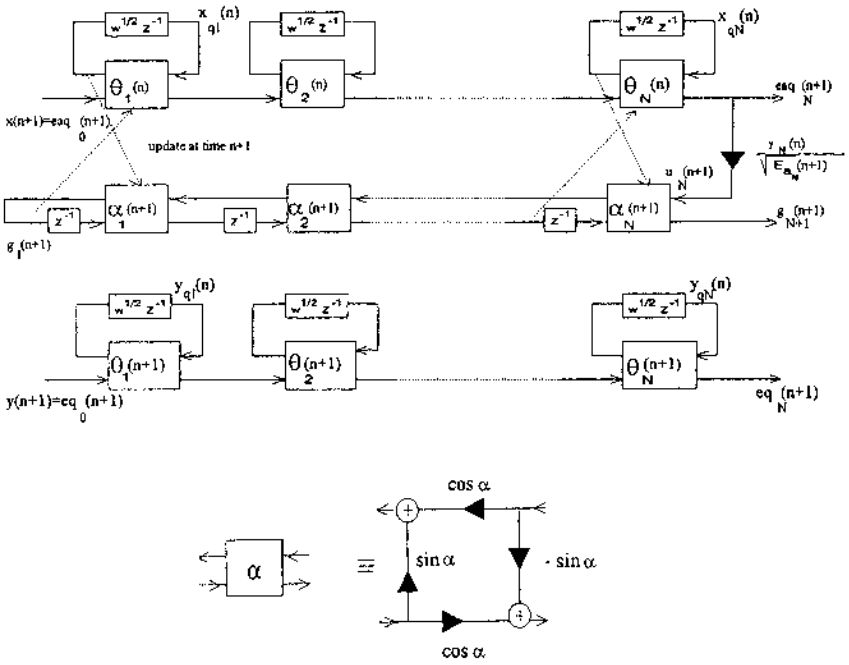


FIG. 9.2 Block diagram of the FLSQR adaptive filter.

brought by the variable u. The two different directions for the propagation of the calculations are worth pointing out. Note that the numerical accuracy in the implementation can be monitored. The last rotation in the $Q_\alpha(n+1)$ matrix provides, according to equation (9.55), the variable $g_{N+1}(n+1)$, which is the last element of $G_{N+1}(n)$, and also the first element of the same vector, i.e., $g_1(n+1)$, which can be computed directly from the input data by

$$g_1(n+1) = \frac{x(n)}{E_{b0}^{1/2}(n)} \tag{9.59}$$

At each time, the difference between the value given by the algorithm and the direct computation reflects the numerical accuracy of the algorithm. A VLSI implementation of the FLSQR algorithm is described in reference [5]. It is based on the CORDIC approach, which is particularly efficient at performing signal processing functions in which rotations are involved.

9.7. THE CASE OF COMPLEX SIGNALS

Rotations suitable for complex data can be derived from the general form of the elementary unitary matrix

$$Q_\theta = \begin{bmatrix} e^{j\alpha}\cos\theta & -e^{j\beta}\sin\theta \\ e^{j\gamma}\sin\theta & e^{j\delta}\cos\theta \end{bmatrix} \tag{9.60}$$

with $\alpha - \beta - \gamma + \delta = 0$ (modulo 2π). However, the FLS-QR algorithm exhibits several specificities which lead to a unique option for the rotation matrix.

To begin with, the rotations involved in the algorithm lead to a real vector, because its single nonzero element is the square root of the error energy, the backward prediction error energy for the rotations denoted Q_α and the forward prediction error energy for Q_a. In the matrices, every rotation has to cancel the first element of a two-element vector and make the second element real. Additionally, the second element in the rotated vector is always real, because of order recursions and because it is the square root of a prediction error energy. Finally, rotation matrices in the present case can only take the form

$$Q_\theta = \begin{bmatrix} c\theta & -s\theta \\ \bar{s}\theta & c\theta \end{bmatrix} \tag{9.61}$$

with $(c\theta)^2 + s\theta\bar{s}\theta = 1$, c$\theta$ being real and sθ complex.

Thus, in order to process complex data, it is sufficient to modify the angle calculations accordingly.

The error ratio $\gamma_N(n)$ is a product of $c\theta$ variables and remains real. The forward linear prediction error energy is computed by

$$E_a(n+1) = WE_a(n) + e_{aq}(n+1)\bar{e}_{aq}(n+1) \tag{9.62}$$

As an illustration, the following results have been obtained in the case of system modelling. The reference signal is

$$y(n) = [0.5 + 0.8j]x(n) + (0.4 + 0.3j)x(n+1) + (0.1 + 0.7j)x(n-2)$$

and $x(n)$ is a white Gaussian noise whose power is unity. After 10^5 iterations, one gets

$$Y_q(n) = [1.0 + 7.0j, 4.1 + 3.2j, 5.1 + 8.2j]^t$$

The theoretical vector is derived from Section 2 and the equation

$$Y_{qN}(n) = T_N(n)H(n) \tag{9.63}$$

The weighting factor in the simulation is $W = 0.99$, and one gets

$$Y_{q3} = \frac{1}{\sqrt{1-W}}[0.1 + 0.7j, 0.4 + 0.3j, 0.5 + 0.8j]^t$$

The rotation algorithm for complex data finds application in communication receivers, for equalization.

9.8. MULTIDIMENSIONAL SIGNALS

The procedure described in the previous sections can be applied to multidimensional (MD) signals, leading to the FLS-QR algorithm for MD signals.

The case of two signals $x_1(n)$ and $x_2(n)$ is considered first. The $(n+1) \times 2N$ input data matrix is defined by

$$X_{2N} = \begin{bmatrix} x_1(n) & x_2(n) & \cdots & x_2(n+1-N) \\ W^{1/2}x_1(n-1) & W^{1/2}x_2(n-1) & \cdots & W^{1/2}x_2(n-N) \\ \vdots & \vdots & \ddots & \vdots \\ W^{n/2}x_1(0) & W^{n/2}x_2(0) & \cdots & 0 \end{bmatrix} \tag{9.64}$$

Following the developments in the previous sections, a fast algorithm is derived from the path to the matrix $T_{2N+2}(n)$ from $T_{2N}(n)$ using forward linear prediction. By definition,

$$Q_{2N}(n)X_{2N}(n) = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \\ 0 & \cdots & 0 \\ 0 & \cdots & \\ \vdots & & T_{2N}(n) \end{bmatrix} \tag{9.65}$$

A time recurrence relation for the rotation matrix $Q_{2N}(n)$ is obtained by considering the equation

$$\begin{bmatrix} 1 & 0 \\ 0 & Q_{2N}(n-1) \end{bmatrix} X_{2N(n)=} \begin{bmatrix} x_1(n) & x_2(n) & \cdots & x_2(n+1-N) \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \\ & W^{1/2} & T_{2N}(n-1) & \end{bmatrix} \tag{9.66}$$

and canceling the first row of the right-hand-side matrix. Now let us introduce forward linear prediction, as in (9.39), by the equation

$$\begin{bmatrix} Q_{2N}(n-1) & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{2N+2}(n) \\ 0 \cdots 0 \end{bmatrix}$$
$$= \begin{bmatrix} e_{1aqn}(n) & e_{2aqn}(n) & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ e_{1aqN}(n) & e_{2aqN}(n) & 0 & \cdots & 0 \\ X_{1aqN}(n) & X_{2aqN}(n) & & T_{2N(n-1)} & \\ W^{n/2}x_1(0) & W^{n/2}x_2(0) & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{9.67}$$

By using two sets of rotations, $Q_{1ea}(n)$ and $Q_{2ea}(n)$, the prediction errors can be accumulated to produce the matrix

$$\begin{bmatrix} X_{1aqN}(n) & X_{2aqN}(n) & T_{2N}(n-1) \\ 0 & E_{22aN}^{1/2}(n) & 0 \cdots 0 \\ E_{11aN}^{1/2}(n) & E_{12aN}^{1/2}(n) & 0 \cdots 0 \end{bmatrix} \tag{9.68}$$

It appears that $E_{11aN}^{1/2}(n)$ is the prediction error energy for the signal $x_1(n)$. As concerns the signal $x_2(n)$, its prediction error energy is the sum of two terms, namely $E_{12aN}(n)$ which is linked to the signal $x_1(n)$ and an orthogonal component $E_{22aN}(n)$.

The same triangularization procedure can be applied to the input signals themselves, yielding

$$Q_\psi \begin{bmatrix} x_1(n) & x_2(n) \\ 0 & W^{1/2}E_{22b0}^{1/2}(n-1) \\ W^{1/2}E_{11b0}^{1/2}(n-1) & W^{1/2}E_{12b0}^{1/2}(n-1) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & E_{22b0}^{1/2}(n) \\ E_{11b0}^{1/2}(n) & E_{12b0}^{1/2}(n) \end{bmatrix} \tag{9.67}$$

where Q_ψ is a product of two rotations. The matrix obtained is included in the triangular matrices expressed by

$$T_{2N+2}(n) = \begin{bmatrix} 0 & 0 & \cdots & E_{22bN}^{1/2}(n) \\ \vdots & \vdots & \iddots & \\ 0 & E_{22b0}^{1/2}(n) & \cdots & X_{2BqN}(n) \\ E_{11b0}^{1/2}(n) & E_{12b0}^{1/2}(n) & \cdots & \end{bmatrix} \tag{9.68}$$

The procedure to derive a fast algorithm starts from (9.67). The matrix $T_{2N+2}(n)$ is obtained by a set of two rotation matrices $Q_{1\alpha}(n)$ and $Q_{2\alpha}(n)$, and the 2-D version of the factorization (9.54) is

$$Q_{2N+2}(n) = Q_{2\alpha}(n)Q_{1\alpha}(n)Q_{2ea}(n)Q_{1ea}(n)\begin{bmatrix} Q_{2N}(n-1) & 0 \\ 0 & I_2 \end{bmatrix} \tag{9.69}$$

The restriction to the first column is obtained by multiplying both sides by the pinning vector, which yields

$$\begin{bmatrix} \gamma_{2N+2}(n) \\ 0 \\ \vdots \\ 0 \\ G_{2N+2}(n) \end{bmatrix} = \begin{bmatrix} \gamma_{2N+2}(n) \\ 0 \\ \vdots \\ 0 \\ g_{2N+2}(n) \\ g_{2N+1}(n) \\ G_{2N}(n) \end{bmatrix} \tag{9.70}$$

and

$$\begin{bmatrix} \gamma_{2N+2}(n) \\ 0 \\ \vdots \\ 0 \\ y_{2N+2}(n) \\ y_{2N+1}(n) \\ G_{2N}(n) \end{bmatrix} = Q_{2\alpha}(n)Q_{1\alpha}(n)Q_{2ea}(n)Q_{1ea}(n)\begin{bmatrix} \gamma_{2N}(n-1) \\ 0 \\ \vdots \\ 0 \\ G_{2N}(n-1) \\ 0 \\ 0 \end{bmatrix} \tag{9.71}$$

Finally, the matrices which cancel the first row of the right hand side matrix in (9.66) are obtained by

$$\begin{bmatrix} \gamma_{2N}(n) \\ 0 \\ \vdots \\ 0 \\ G_{2N}(n) \end{bmatrix} = R_{2N} \cdots R_1 \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{9.72}$$

The algorithm for 2-D signals closely follows the FLS-QR algorithm for 1-D signals, the size of the vectors for internal data being doubled.

In fact, it is even possible to use the 1-D algorithm to perform multidimensional adaptive filtering, as already shown in Section 7.13 for transversal filters. Let us consider the rotated input data vector

$$\begin{bmatrix} e_{2aq}(n+1) \\ X_{2q}(n+1) \end{bmatrix} = Q_{2a}(n)\begin{bmatrix} x_2(n+1) \\ W^{1/2}X_{2q}(n) \end{bmatrix} \tag{9.73}$$

and the prediction error energy

$$E_{2a}(n+1) = WE_{2a}(n) + e_{2aq}^2(n+1) \tag{9.74}$$

Next, the matrix $Q_{2\alpha}(n+1)$ can be computed according to equation (9.55), and the following vector can be obtained, repeating (9.56):

$$G'_{2N+1}(n+1) = Q_{2\alpha}(n+1)\begin{bmatrix} G_{2N}(n) \\ u \end{bmatrix} \tag{9.75}$$

Continuing with the sequence of operations in FLS-QR, a $(2N+1) \times (2N+1)$ rotation matrix $Q'_{2a}(n+1)$ is computed by

$$\begin{bmatrix} \gamma'_{2N+1}(n+1) \\ G'_{2N+1}(n+1) \end{bmatrix} = Q'_{2a}(n+1)\begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{9.76}$$

With the help of this matix, the new input data vector associated with $x_1(n+1)$ can be computed

$$\begin{bmatrix} e'_{1aq}(n+1) \\ X'_{1q}(n+1) \end{bmatrix} = Q'_{2a}(n+1)\begin{bmatrix} x_1(n+1) \\ W^{1/2}X'_{1q}(n) \end{bmatrix} \tag{9.77}$$

and repetition of the above sequence of operations leads to the vector $G_{2N+2}(n+1)$.

Since only the $2N \times 2N$ matrix $Q_{2a}(n+1)$ is needed, the last two elements in $G_{2N+2}(n+1)$ can be dropped to obtain $G_{2N}(n+1)$. Therefore, in the second path also, the computations can be carried out on vectors with $2N$ elements. Finally, the rotation matrix $Q_{2a}(n)$ is updated in two paths of the 1-D linear prediction algorithm, fed with the data set $[x_2(n+1), X_{2q}(n)]$ and $[x_1(n+1), X'_{1q}(n+1)]$, respectively.

The above approaches can be extended to multidimensional signals. However, it is worth emphasizing that a specific triangularization operation has to be included in the direct approach, according to (9.68), for each of the K dimensions of the input signal. In contrast, the standard 1-D algorithm can be used K times to achieve the same result, and is therefore simpler to implement.

9.9. NORMALIZATION AND EQUIVALENCE WITH LATTICE

The FLS-QR algorithm in Figure 9.1 can be made more regular by a normalization operation, performed by the rotation

$$\begin{bmatrix} 0 \\ E_{b0}^{1/2}(n+1) \end{bmatrix} = \begin{bmatrix} \cos\psi(n+1) & -\sin\psi(n+1) \\ \sin\psi(n+1) & \cos\psi(n+1) \end{bmatrix}\begin{bmatrix} x(n+1) \\ W^{1/2}E_{b0}^{1/2}(n) \end{bmatrix} \tag{9.78}$$

with

$$\sin\psi(n+1) = \frac{x(n+1)}{E_{b0}^{1/2}(n+1)}; \cos\psi(n+1) = \frac{W_{b0}^{1/2}E^{1/2}(n)}{E_{b0}^{1/2}(n+1)}$$

In the algorithm, $x(n+1)$ is replaced by $\sin\psi(n+1)$, and $W^{1/2}$ is replaced by $\cos\psi(n+1)$. Then, the first angle calculation equation becomes

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} = Q_\alpha(n+1)\begin{bmatrix} X_q(n+1) \\ E_a^{1/2}(n+1) \end{bmatrix} \tag{9.79}$$

and it is similar to the second angle calculation. In fact, the above equation (9.79) can be reversed and expressed as

$$\begin{bmatrix} E_a^{1/2}(n+1) \\ X_q(n+1) \end{bmatrix} = J_N Q_\alpha^{-1}(n+1)J_N\begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{9.80}$$

Now, an order recursion algorithm can be derived which turns out to be equivalent to the normalized lattice algorithm.

Relationships between QR and lattice approaches have been pointed out in the previous sections. According to the definitions in Section 8.5, the normalized backward prediction error is

$$e_{nbi}(n+1) = \frac{\varepsilon_{bi}(n+1)}{\varphi_i^{1/2}(n+1)E_{bi}^{1/2}(n+1)} = \sin\theta_{i+1}(n+1) \tag{9.81}$$

and similarly,

$$e_{nai}(n+1) = \frac{e_{aqi}(n+1)}{E_{ai}^{1/2}(n+1)} \tag{9.82}$$

The lattice coefficients are linked to the angles by equation (9.47): $\sin\alpha_i(n+1) = k(n+1)$. Finally, the same variables appear in both normalized QR and normalized lattice algorithms.

9.10. CONCLUSION

Of all the algorithms for adaptive filtering, the FLS-QR given in Figure 9.1 is probably the best in many respects. It combines the performance of least squares with the robustness of gradient techniques. It is a square-root type of algorithm, and no expansion of the data dynamic range is necessary. The angle calculations can be efficiently performed with the help of look-up tables.

As concerns flexibility, it can cope with different situations: complex signal, multidimensional signals, and temporary absence of data. Different types of errors can be accommodated, and the internal variables can be exploited for signal analysis. It is also possible, whenever it is necessary, to deliver the transversal coefficients, using an iterative procedure similar to the method given in Section 8.6 for the lattice structure.

EXERCISES

1. Consider the data vector $X = [1, 0.7, 0.2, -0.5]^t$. Use a set of rotations to compute its norm.
2. Study the relationships between the autocorrelation matrix $R_N(n)$ and the triangular matrices $S_N(n)$ and $T_N(n)$. Discuss the connections with other triangular decompositions of $R_N(n)$ as in Chapter 5.
3. The input signal to a QR algorithm is $x(n) = \sin n\pi/3$. After 10^3 iterations, the rotated data vector is

   $$X_{aq8} = [-0.520, 0.854, 0.018, 0.003, -0.004, -0.007, -0.005, 0.001]^t$$

   Give the value of the weighting factor W. Discuss the relative values of the elements of the vector X_{aq8}.
4. In a modeling case, the input signal $x(n)$ is a unit-power white noise and the reference signal is

   $$y(n) = x(n) + 0.7x(n-1) + 0.4x(n-2)$$

   For $N = 5$ and $W = 0.99$, give the theoretical value of the rotated reference signal vector. Verify by simulation using the algorithm of Figure 9.1.
5. Give the complete list of operations for the complex FLS-QR algorithm of Section 9.7.
6. Give the complete list of operations for the normalized FLS-QR algorithm of Section 9.9, in systolic form.
7. In the equalizer of a communication receiver, the reference signal is derived from the adaptive filter output, in the tracking mode. Thus, the filter output $\tilde{y}(n)$ has to be provided, and the following equation can be used

   $$\begin{bmatrix} 0 \\ Y_q'(n+1) \end{bmatrix} = Q_a(n+) \begin{bmatrix} \tilde{y}(n+1) \\ W^{1/2} Y_q(n) \end{bmatrix}$$

   $Q_a(n+1)$ and $Y_q(n+1)$ being known, justify the above equation and explain how $\tilde{y}(n+1)$ and $Y_q'(n+1)$ are calculated. Then show that $Y_q(n+1)$ is obtained by the equation

   $$Y_q(n+1) = Y_q'(n+1) + G_N(n+1)e(n+1)$$

ANNEX 9.1 FORTRAN SUBROUTINE FOR THE FLSQR ALGORITHM

```
C
      SUBROUTINE FLSQR2(N,X,Y,EPSA,EPS,EA,GAMA,W,WD,
      IND,YAQ)
C
C     COMPUTES THE PREDICTION ERROR, THE ERROR RATIO AND
      THE FILTER
C     OUTPUT ERROR (TRIANGULAR FACTORIZATION QR - TYPE 2)
C     N    = FILTER ORDER
C     X    = INPUT SIGNAL : x(n+1)
C     Xaq  = TRANSFORMED FORWARD DATA VECTOR : N elements
             (Reflection coefficients)
C     Y    = REFERENCE SIGNAL
C     G    = TRANSFORMED UNIT VECTOR
             (Backward ''a posteriori'' prediction error
             vector)
C     EPSA = A POSTERIORI PREDICTION ERROR
C     EA   = PREDICTION ERROR ENERGY
C     GAMA = ERROR RATIO
C     W    = WEIGHTING FACTOR
C
      DIMENSION Xaq(15),G(16),CALF(15),SALF(15),
      CTET(15),STET(15)
      DIMENSION YAQ(15)
      IF(IND.GT.1)GOTO30
C
C     INITIALIZATION
C
      DO20I=1,15
      Xaq(I)=0.
      Yaq(I)=0.
      G(I)=0.
      CALF(I)=1.
      SALF(I)=0.
      CTET(I)=1.
      STET(I)=0.
   20 CONTINUE
      EA=1.
      GAMA=1.
   30 CONTINUE
C
C     CALCULATION OF ROTATION ANGLES AND PREDICTION ERRORS
C
```

```
      EPSAQ=X
      DO50I=1,N
      EPAQ=EPSAQ
      EPSAQ=EPAQ*CTET(I)-WD*XAQ(I)*STET(I)
      XAQ(I)=EPAQ*STET(I)+WD*XAQ(I)*CTET(I)
   50 CONTINUE
      EA=W*EA+EPSAQ*EPSAQ
      E1=EA
      EQ=SQRT(EA)
      ALF2=GAMA*EPSAQ/EQ
      DO60I=1,N
      EB=XAQ(N+1-I)*XAQ(N+1-I)+E1
      EBQ=SQRT(EB)
      SALF(N+1-I)=XAQ(N+1-I)/EBQ
      CALF(N+1-I)=EQ/EBQ
      EQ=EBQ
      E1=EB
   60 CONTINUE
      EN=G(N)*SALF(N)+ALF2*CALF(N)
      DO70I=2,N
      G(N+2-I)=G(N+1-I)*CALF(N+1-I)-EN*SALF(N+1-I)
      EN1=G(N+1-I)*SALF(N+1-I)+EN*CALF(N+1-I)
      EN=EN1
   70 CONTINUE
      G(1)=EN
      GAMA=1.
      DO80I=1,N
      STET(I)=G(I)/GAMA
      CTET(I)=SQRT(1.-STET(I)*STET(I))
      GAMA=GAMA*CTET(I)
   80 CONTINUE
      EPSA=GAMA*EPSAQ
C
C     FILTER SECTION
C
      EPSQ=Y
      DO90I=1,N
      EPQ=EPSQ
      EPSQ=EPQ*CTET(I)-WD*YAQ(I)*STET(I)
      YAQ(I)=EPQ*STET(I)+WD*YAQ(I)*CTET(I)
   90 CONTINUE
      EPS=EPSQ*GAMA
      RETURN
      END
```

REFERENCES

1. J. G. F. Francis, ‘‘The QR Transformation: A Unitary Analog to the LR Transformation,’’ *Computer Journal* **4**, 265–271 (1961).
2. J. Cioffi, ‘‘The Fast Adaptive Rotors RLS Algorithm,’’ *IEEE Trans.* **ASSP-38**, 631–653 (April 1990).
3. I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, ‘‘Fast QRD-Based Algorithms for Least-Squares Linear Prediction,’’ *Proceedings of the IMA Conference on Mathematics in Signal Processing*, Warwick, U.K. Dec. 1988.
4. P. Regalia, ‘‘Numerical Stability Properties of QR-Based Fast Least Squares Algorithms,’’ *IEEE Trans.* on Signal Processing, **41**, 2096–2109 (June 1993).
5. M. Terré, ‘‘Study of FLS-QR Algorithms Principles and Implementation,’’ Ph.D. Thesis, CNAM, Paris, Feb. 1995.
6. M. Terré and M. Bellanger, ‘‘A Fast Least Squares QRD-Based Algorithm for Complex Data,’’ *IEEE Trans.* **SP-42**, 3272–3273 (November 1999).
7. M. Bellanger and P. Regalia, ‘‘The FLS-QR Algorithm for Adaptive Filtering: the Case of Multichannel Signals,’’ *Signal Processing*, **22**, 115–126 (1991).

10
Spectral Analysis

The estimation of prediction coefficients which is performed in the adaptation gain updating section of FLS algorithms corresponds to a real-time analysis of the input signal spectrum. Therefore, in order to make correct decisions when choosing the algorithm parameter values, we need a good knowledge of the signal characteristics, particularly its spectral parameters.

Independently of FLS algorithms, adaptive filters in general are often used to perform signal analysis. Thus, it is clear that the fields of adaptive filtering and spectral analysis are tightly interconnected.

In this chapter, the major spectrum estimation techniques are reviewed, with emphasis on the links with adaptive filtering. To begin with, the objectives are stated [1].

10.1. DEFINITION AND OBJECTIVES

In theory the spectral analysis of a stationary signal $x(n)$ consists of computing the Fourier transform $X(f)$ defined by

$$X(f) = \sum_{n=-\infty}^{\infty} x(n)e^{-j2\pi nf} \tag{10.1}$$

The function $X(f)$ consists of a set of pulses, or spectral lines, if the signal is periodic or predictable. It has a continuous component if it is random. These aspects are discussed in Chapter 2.

In practical situations, for many different reasons, only a finite set, or record, of input data is available. and it is an estimate of the true spectrum which is obtained. The set of N data, $x(n)(0 \leqslant n \leqslant N-1)$, is considered as a realization of a random process whose power spectral density, or spectrum, $S(f)$ is defined, as stated in Section 2.4, by

$$S(f) = \sum_{p=-\infty}^{\infty} r(p)e^{-j2\pi pf} \tag{10.2}$$

where the ACF values for complex signals are defined by

$$r(p) = E[x(n)\bar{x}(n-p)]$$

The spectral analysis techniques aim at providing estimates of that true spectrum $S(f)$. To judge the performance, we envisage three criteria: resolution, fidelity, and variance.

The limitation of the data record length produces a smoothing effect in the frequency domain which distorts and obscures details. If the estimated spectrum is smoothed to the degree that two spectral lines of interest cannot be distinguished, the estimator is said to have inadequate or low resolution. The resolution is often judged subjectively [2]. Here, it is taken as the minimum frequency interval necessary to separate two lines.

The fidelity can be measured by the distance of the estimated spectrum from the true spectrum. It takes into account the error or bias, when estimating the frequency of a line, as well as its amplitude.

The variance, as usual, measures the confidence one can have in the estimator.

An ideal spectrum estimator would equally well, with respect to specified criteria as above, represent the true spectrum, irrespective of its characteristics. Unfortunately, it is not possible, and the different methods are in general linked to particular signals and emphasize a specific criterion. The presentation given below corresponds, to a certain degree, to an order of increasing resolution. Therefore emphasis is put on line spectra, which represent a significant share of the applications and permit simple and clear comparisons.

10.2. THE PERIODOGRAM METHOD

From the available set of N_0 data, an estimate $S_{\text{pd}}(f)$ of the spectrum is obtained from

$$S_{\text{pd}}(f) = \frac{1}{N_0}\left|\sum_{n=0}^{N_0-1} x(n)e^{-j2\pi nf}\right|^2 \tag{10.3}$$

In order to relate $S_{\text{pd}}(f)$ t othe true spectrum $S(f)$, let us expand the right side and rearrange the summation:

$$S_{\text{pd}}(f) = \sum_{p=-(N_0-1)}^{N_0-1} \left[\frac{1}{N_0} \sum_{n=p}^{N_0-1} x(n)\bar{x}(n-p) \right] e^{-j2\pi pf} \tag{10.4}$$

The expression in brackets is the estimate $r_1(p)$ of the ACF studied in Section 3.2. Taking the expectation of both sides of (10.4) yields

$$E[S_{\text{pd}}(f)] = \sum_{p=-(N_0-1)}^{N_0-1} r(p) \frac{N_0 - |p|}{N_0} e^{-j2\pi pf} \tag{10.5}$$

which, due to the properties of the Fourier transform, leads to

$$E[S_{\text{pd}}(f)] = S(f) * \frac{\sin^2 \pi f N_0}{N_0 \sin^2 \pi f} \tag{10.6}$$

where * denotes the convolution operation, which corresponds to a filtering operation in the frequency domain. The filtering function is shown in Figure 10.1.

When $N_0 \to \infty$, one gets

$$\lim_{N_0 \to \infty} E[S_{\text{pd}}(f)] = S(f) \tag{10.7}$$

Thus, the estimate is unbiased. If the spectrum consists of just a line, it is correctly found by that approach, as the peak of the estimate $S_{\text{pd}}(f)$. However, two lines associated with two sinusoids in the signal can only be distinguished if they are at least separated by the interval $\Delta f = 1/N_0$,

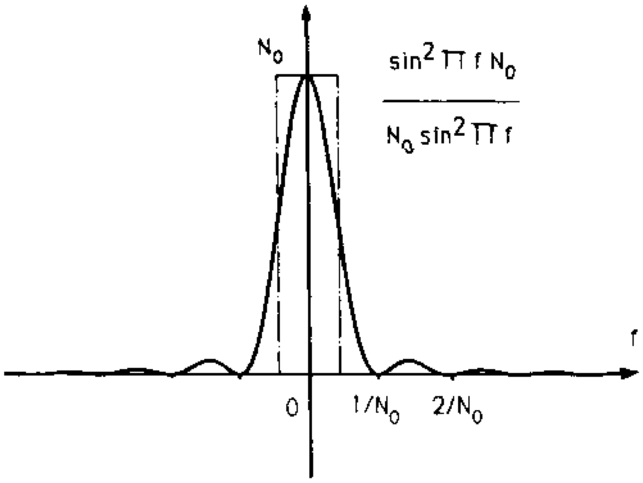


FIG. 10.1 The frequency domain filtering function in the periodogram method.

which, therefore, is the frequency resolution of the analysis.

The variance can be calculated. The following simple expression is obtained for Gaussian signals:

$$\mathrm{var}\{S_{\mathrm{pd}}(f)\} = S^2(f)\left[1 + \frac{\sin^2 2\pi f N_0}{N_0 \sin^2 2\pi f}\right] \tag{10.8}$$

Equation (10.8) shows that for large N_0 the standard deviation of the estimator is equal to its expectation. The estimator is not consistent.

In order to reduce the variance, we can divide the set of N_0 data into K subsets, with or without overlap. The periodogram is computed on each subset, and the spectrum estimate is taken as the average of the values obtained. By so doing, the variance is approximately $S^2(f)/K$, and if K is made proportional to the record length N_0, then the estimator is consistent. However, the counterpart is a decrease in resolution by the factor K.

Data windowing can be incorporated in the method to counter the sidelobe leakage effect. Let $w(n)$ denote the weighting function, the estimate is

$$S_{\mathrm{pd}w}(f) = \frac{1}{K}\sum_{i=1}^{K} \frac{1}{\sum_{n=0}^{M-1} w^2(n)} \left| \sum_{n=0}^{M-1} x(iM+n) w(n) e^{-j2\pi n f} \right|^2 \tag{10.9}$$

where $M = N_0/K$ is the number of data per section. In practice, some degree of overlap is generally taken in the sectioning process to gain on the estimator variance. For example, with $M = 2N_0/K$ and a square cosine window, $w(n) = \cos^2(\pi/M)(n - M/2)$, the variance is reduced by alsmot 25% with respect to the case $M = N_0/K$ and $w(n) = 1$.

The above technique is also called the weighted periodogram or Welch method [3]. It is made computationally efficient by using the FFT algorithm to compute the periodograms of the data subsets. In that case, the spectrum is estimated at discrete frequencies, which are interger multiples of f_s/M, f_s being the input signal sampling frequency.

10.3. THE CORRELOGRAM METHOD

A critical point in the previous approach is the choice of the sectioning parameter K value. It has to be a trade-off between resolution and variance, but the information for making the decision is not readily available. Consider equation (10.6); the effect of the convolution operation is negligible if the following approximation holds:

$$S(f) * \frac{\sin^2 \pi f M}{M \sin^2 \pi f} \approx S(f) \tag{10.10}$$

or, in the time domain,

$$r(p)\left[1 - \frac{|p|}{M}\right] \approx r(p) \tag{10.11}$$

Consequently, the length M of each section of the data record should be significantly greater than the range of index values P_0, over which the correlation function is not negligible.

The correlogram method is in a better position in that respect. It consists of the direct computation of the spectrum according to its definition (10.2). If P ACF values are available, the estimate is

$$S_{\mathrm{CR}}(f) = \sum_{p=-(P-1)}^{P-1} r(p) e^{-j2\pi pf} \tag{10.12}$$

or, as a function of the true spectrum,

$$S_{\mathrm{CR}}(f) = S(f) * \frac{\sin \pi f (2P-1)}{\sin \pi f} \tag{10.13}$$

If the correlation values are computed using N_0 data, as shown in Section 3.2, an estimate of the true correlation function is obtained, which in turn is reflected in the spectrum estimation $S_{\mathrm{CR}}(f)$. It can be shown that the variance is approximately

$$\mathrm{var}\{S_{\mathrm{CR}}(f)\} \approx \frac{2P-1}{N_0} S^2(f) \tag{10.14}$$

Therefore the number of correlation values must be taken as small as possible. The optimal conditions are obtained if P is chosen as P_0, assuming the AC function can be neglected for $P > P_0$, as shown in Figure 10.2. The estimation, according to (10.13), can become negative. In real applications, the ACF values have to be estimated; another window $w(p)$ is used instead of the rectangular window, leading to the estimate

$$S_{\mathrm{CR}w}(f) = \sum_{p=-(P-1)}^{P-1} w(p) r(p) e^{-j2\pi pf} \tag{10.15}$$

With the triangular window, $w(p) = 1 - |p|/P$, the estimate is positive, as in the previous section, and the information is available for choosing the summation range P. The variance is

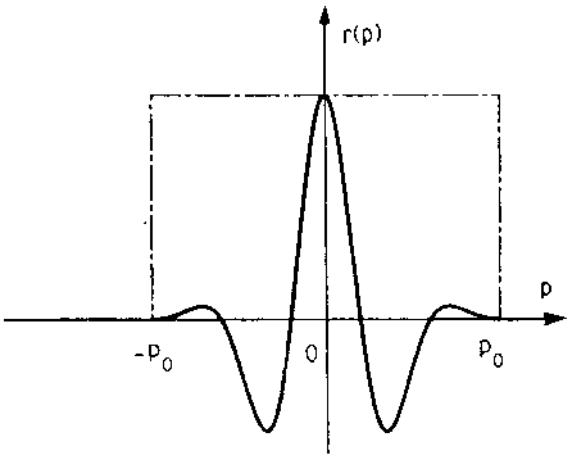


FIG. 10.2 Optimal number of AC values in the correlogram method.

$$\operatorname{var}\{S_{\mathrm{CR}w}(f)\} \approx S^2(f)\frac{1}{N_0}\sum_{p=-(P-1)}^{P-1} w^2(p) \tag{10.16}$$

The correlogram is also called the Blackman–Tukey method [4]. Concerning the computational complexity, the calculation of the ACF values has a significant impact [5]. However, if the simplified techniques described in Section 3.2 can be used, the need for multiplications is avoided and the approach is made efficient.

In the methods described above, and particularly the periodogram, the Fourier transform operates as a bank of filters whose coefficients are the same for all the frequency values. Instead, the filter coefficients can be adjusted for each frequency to minimize the estimation variance.

10.4. THE MINIMUM VARIANCE (MV) METHOD

The principle of that approach, also called the maximum likelihood or Capon method [6], is to calculate the coefficients of a filter matched to the frequency under consideration and take the filter output signal power as the value of the power spectrum. Consequently, a sinusoid at that frequency is undistorted, and the variance of the output is minimized.

The filter output is

$$y(n) = \sum_{i=0}^{N} h_i x(n-i) = H^t X(n) \tag{10.17}$$

The $N+1$ coefficients are subject to the constraint

$$\sum_{i=0}^{N} \bar{h}_i e^{-j2\pi if} = 1 \tag{10.18}$$

to preserve a cisoid at frequency f. Let

$$F = [1, e^{-j2\pi f}, \ldots, e^{-j2\pi Nf}]^t$$

be the vector with complex elements. The filter coefficients which minimize the output power minimize the expression

$$E[\bar{H}^t[X(n)X^t(n)]H] + \alpha(1 - \bar{H}^t F) \tag{10.19}$$

where α is a Lagrange multiplier.

The optimum coefficients are

$$H_{\text{opt}} = \frac{\alpha}{2} R_{N+1}^{-1} F \tag{10.20}$$

Using equation (9.18) to get the value of α and substituting into the above expression yields

$$H_{\text{opt}} = \frac{1}{\bar{F}^t R_{N+1}^{-1} F} R_{N+1}^{-1} F \tag{10.21}$$

The output signal power $S_{\text{MV}}(f)$ is

$$S_{\text{MV}}(f) = \bar{H}_{\text{opt}}^t R_{N+1} H_{\text{opt}} = \frac{1}{\bar{F}^t R_{N+1}^{-1} F} \tag{10.22}$$

If such a filter is calculated for every frequency value, an estimate of the power spectrum is

$$S_{\text{MV}}(f) = \frac{1}{\sum_{k=0}^{N} \sum_{l=0}^{N} \rho_{kl} e^{-j2\pi(k-l)f}} \tag{10.23}$$

where the values ρ_{kl} are the elements of the inverse input signal AC matrix, which have to be estimated from the input data set.

The function $S_{\text{MW}}(f)$ can be related to the prediction filter frequency responses, denoted by $A_i(f)$ and defined by

$$A_i(f) = 1 - \sum_{k=1}^{i} a_{kl} e^{-j2\pi kf}, \quad 0 \leqslant i \leqslant N \tag{10.24}$$

The triangular decomposition of the inverse AC matrix [equation (5.81) of Chapter 5] yields

$$\frac{1}{S_{\mathrm{MV}}(f)} = \sum_{i=0}^{N} |A_i(f)|^2 \tag{10.25}$$

or

$$\frac{1}{S_{\mathrm{MV}}(f)} = \sum_{i=0}^{N} \frac{1}{S_{\mathrm{AR}i}(f)} \tag{10.26}$$

where

$$S_{\mathrm{AR}i}(f) = \frac{1}{\left|1 - \sum\limits_{k=1}^{i} a_{ki} e^{-j2\pi kf}\right|^2} \tag{10.27}$$

is the AR spectrum estimate, taken as the inverse of the squared prediction error filter response.

Therefore, $S_{\mathrm{MV}}(f)$ turns out to be the harmonic average of the AR estimations for all orders from 0 to N; consequently it exhibits less resolution than the AR estimate with the highest order N. The emphasis with that approach is on minimizing the variance.

The resolution of the MV method can be significantly improved. According to the definition, it provides an estimate of the signal power at each frequency. A better resolution is obtained by techniques which estimate the power spectral density instead. As seen in Chapter 3 with eigenfilters, the power spectral density is kept if the filter is unit norm, which leads to the minimum variance with normalization estimate [7]

$$S_{\mathrm{MV}N}(f) = \frac{\bar{H}_{\mathrm{opt}}^t R_{N+1} H_{\mathrm{opt}}}{\bar{H}_{\mathrm{opt}}^t H_{\mathrm{opt}}} \tag{10.28}$$

Using (9.21), we have

$$S_{\mathrm{MV}N}(f) = \frac{\bar{F}^t R_{N+1}^{-1} F}{\bar{F}^t R_{N+1}^{-2} F} \tag{10.29}$$

As an illustration, the functions $S_{\mathrm{MV}}(f)$ and $S_{\mathrm{MV}N}(f)$ are shown in Figure 10.3 for a signal consisting of two sinusoids in white noise. The data record length is 64; the filters have $N + 1 = 10$ coefficients; the power of each sinusoid is 10 dB above the noise power. Clearly, the resolution of the normalized method is significantly improved, since it can distinguish the two sine waves, whereas the standard method cannot.

The MV method with normalization comes closer to the AR method, as far as resolution is concerned. The price to be paid is a significant increase in computations.

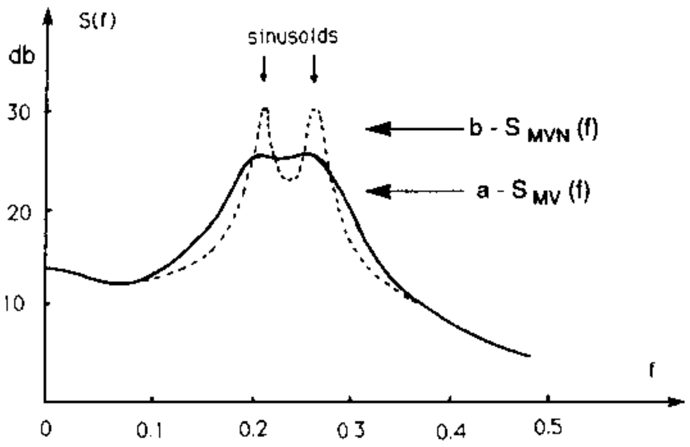


FIG. 10.3 Minimum variance spectral estimation of two sinusoids in white noise: (a) standard method, (b) normalized method.

The methods presented so far are based on filtering the data with a filter matched to a single sinusoid. Consequently, they are not optimal to resolve several sinusoids. Methods for that specific case are presented next.

10.5. HARMONIC RETRIEVAL TECHNIQUES

The principle of harmonic decomposition has been introduced in Section 2.11 as an illustration of the fundamental decomposition of signals. It is based on the assumption that the signal consists of real sinusoids with uncorrelated random phases in white noise, and the spectrum is calculated as

$$S_{\mathrm{HR}}(f) = \sigma_e^2 + \sum_{i=1}^{M} |S_i|^2 [\delta(f - f_i) + \delta(f + f_i)] \tag{10.30}$$

The corresponding ACF is

$$r(p) = 2 \sum_{i=1}^{M} |S_i|^2 \cos(2\pi p f_i) + \sigma_e^2 \delta(p) \tag{10.31}$$

where σ_e^2 is the white noise power, f_i are the sinusoid frequencies, and S_i are their amplitudes.

According to the results of Chapters 2 and 3, the noise power corresponds to the smallest eigenvalue of the $(2M+1) \times (2M+1)$ AC matrix R_{2M+1}:

$$\sigma_e^2 = \lambda_{\min,\ 2M+1} \tag{10.32}$$

and the sinusoid frequencies are the zeros of the associated eigenvector polynomial

$$H_{2M+1}(z) = 1 + \sum_{i=1}^{2M} v_{\min,\ i} z^{-i} \tag{10.33}$$

Once the M sinusoid frequencies have been obtained, the powers are calculated by solving the system of equations (10.31) for $p = 0, 1, \ldots, M-1$, corresponding to the matrix equation (2.41) of Chapter 2.

If it is not known a priori, the order M is determined by calculating the minimum eigenvalues $\lambda_{\min,\ 2M+1}$ for increasing orders until they become virtually constant.

This method, called the *Pisarenko* method, provides, when the ACF values are known, unbiased spectral estimates of signals consisting of the sum of M sinusoids with uncorrelated random phases in white noise [8]. It is very elegant and appealing from a theoretical point of view. However, with real data, when the ACF has to be estimated, it exhibits a number of limitations.

First of all, there is a fundamental practical difficulty related to the hypothesis on the noise. Estimation bounds and experiments show that it takes numerous samples for the characteristics of a white noise to appear. It has been shown in Section 3.11 that errors on the AC matrix elements affect the eigendirections; therefore, a bias is introduced on the sinusoid frequencies which becomes significant for low SNRs and small data records. That bias can be reduced by taking the order in the procedure greater than the number of sinusoids M. In these conditions, the occurrence of spurious lines can be prevented by choosing the minimum eigenvector in the noise space in such a way that the zeros of the eigenpolynomial in excess be not on the unit circle, as discussed in Section 10.8.

Example

To illustrate the impact of the number N_0 of data available, let us take the signal

$$x(n) = \sin(n\omega_0 + \varphi)$$

Estimator (3.23) for the AC function yields

$$r_2(p) = \tfrac{1}{2}\cos p\omega_0 + \tfrac{1}{2}\frac{\sin(N_0 - p)\omega_0}{(N_0 - p)\sin\omega_0}\cos[(N_0 + 1)\omega_0 + 2\varphi]$$

Taking the order $2M + 1 = 3$ in the procedure, the estimated frequency is

$$\tilde{\omega}_0 = \cos^{-1}\left[\frac{r(2) + \text{sign}[r(1)]\sqrt{r^2(2) + 8r^2(1)}}{4r(1)}\right]$$

To judge the sensitivity of this estimator with respect to the AC function, it is useful to calculate the derivatives:

$$\left|\frac{\partial\tilde{\omega}_0}{\partial r(1)}\right| = \left|\frac{r(2)}{r(1)}\right|\frac{1}{\sqrt{r^2(2) + 8r^2(1)}}\left|\tan^{-1}\tilde{\omega}_0\right|$$
$$\left|\frac{\partial\tilde{\omega}_0}{\partial r(2)}\right| = \frac{1}{\sqrt{r^2(2) + 8r^2(1)}}\left|\tan^{-1}\tilde{\omega}_0\right|$$

Clearly, for $\tilde{\omega}_0$ small or close to π, the estimation is very sensitive to perturbations of $r(1)$ or $r(2)$, which can come from the second term in the above estimation of $r_2(p)$ or from a noise component.

Concerning computational complexity, the procedure contains two demanding operations: the eigenparameter calculation, and the extraction of the roots of the eigenpolynomial. The determination of the frequencies f_i becomes inaccurate for large order M. It can be avoided if only the shape of the spectrum is of interest by calculating

$$S'_{\text{HR}}(f) = \frac{|H_m|^2}{\left|1 + \sum_{i=1}^{2M} v_{i,\min}e^{-j2\pi if}\right|^2} \tag{10.34}$$

where H_m is the peak of the modulus of $H_{2M+1}(e^{j2\pi if})$.

For the eigenparameter calculation, the conjugate gradient technique given in Annex 3.2 is an efficient approach. Iterative gradient techniques, mentioned in Section 3.10, can also be used; they permit an adaptive realization.

The hypothesis on the noise is avoided in the damped sinusoid decomposition method, called the *Prony* method [9]. The principle consists of fitting the set of $N_0 = 2P$ data to P damped cisoids:

$$x(n) = \sum_{i=1}^{P} S_i z_i^n \tag{10.35}$$

with

$$z_i = e^{-(\alpha_i + j2\pi f^i)}$$

As pointed out in Chapter 2, this is equivalent to assuming that the data satisfy the recurrence relationship

$$x(n) = \sum_{i=1}^{P} a_i x(n-i) \tag{10.36}$$

The coefficients can be obtained by solving the system

$$\begin{bmatrix} x(p) & x(p-1) & \cdots & x(1) \\ x(p+1) & x(p) & \cdots & x(2) \\ \vdots & \vdots & & \vdots \\ x(2p-1) & x(2p-2) & \cdots & x(p) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} x(p+1) \\ x(p+2) \\ \vdots \\ x(N_0) \end{bmatrix} \tag{10.37}$$

An iterative solution is given in Section 5.14, and the algorithm is presented in Figure 5.13. The values z_i are computed as the roots of the equation

$$1 - \sum_{i=1}^{P} a_i z^{-i} = 0 = \prod_{i=1}^{P} (1 - z_i z^{-1}) \tag{10.38}$$

Finally the amplitudes S_i are obtained by solving the system

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ z_1 & z_2 & \cdots & z_P \\ \vdots & \vdots & & \vdots \\ z_1^{P-1} & z_2^{P-1} & \cdots & z_P^{P-1} \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_P \end{bmatrix} = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(P) \end{bmatrix} \tag{10.39}$$

The spectral estimate corresponds to the Fourier transform of equation (10.35); if we assume symmetry, $x(n) = x(-n)$, the result, for continuous signals, is

$$S_{\mathrm{DS}}(f) = \left| \sum_{i=1}^{P} \frac{2\alpha_i S_i}{\alpha_i^2 + 4\pi^2 (f - f_i)^2} \right|^2 \tag{10.40}$$

The method can be extended to real undamped sinusoids. It is well suited to finding out the modes in a vibration transient.

System (10.37) may be under- or overdetermined and solved as indicated in Section 3.5. The overdetermination case corresponds to the AR approach.

10.6. AUTOREGRESSIVE MODELING

The method is associated with the calculation of the linear prediction coefficients $a_i(1 \leqslant i \leqslant N)$ through the normal equations

$$R_{N+1} \begin{bmatrix} 1 \\ -a_1 \\ \vdots \\ -a_N \end{bmatrix} = \begin{bmatrix} E_N \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{10.41}$$

where E_N is the prediction error power and R_{N+1} is an estimate of the signal AC matrix. The spectrum is derived from the coefficients by

$$S_{\text{AR}}(f) = \frac{E_N}{\left|1 - \sum_{i=1}^{N} a_i e^{-j2\pi if}\right|^2} \tag{10.42}$$

The resolution capability of that approach is illustrated in Figure 10.4, which shows the spectrum estimated from $N = 64$ samples of a signal consisting of two sinusoids separated by $\frac{1}{5}(\frac{1}{N_0})$, the SNR being 50 dB. Clearly, the AR method provides a good analysis, but the Fourier transform approach cannot distinguish the two components; it is a high-resolution technique.

The matrix R_{N+1} used in (10.41) can be calculated from the set of N_0 data in various ways [10]. Let us consider the $(N_0 + N) \times (N + 1)$ input signal matrix

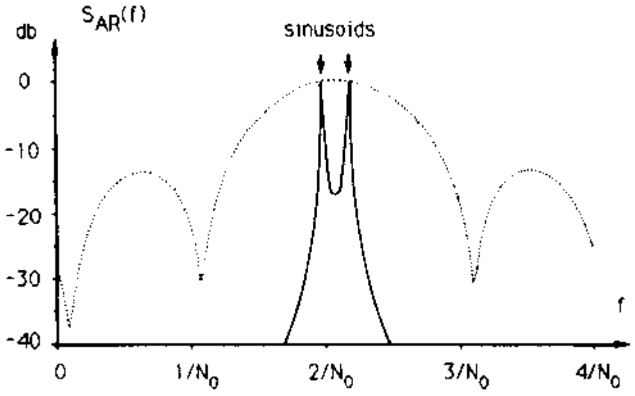


FIG. 10.4 AR spectral estimation of two sinusoids.

$$X_{N_0(N+1)} = \left[\begin{array}{ccc} x(N_0) & \cdots & 0 \\ x(N_0-1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ x(N_0-N+1) & \cdots & 0 \\ x(N_0-N) & \cdots & x(N_0) \\ \vdots & \cdots & \vdots \\ x(1) & \cdots & x(N+1) \\ 0 & \cdots & x(N) \\ \vdots & \ddots & \vdots \\ 0 & \cdots & x(1) \end{array}\right] \begin{array}{l} \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\} U \\ \left.\begin{array}{l} \\ \\ \\ \end{array}\right\} X_a \\ \left.\begin{array}{l} \\ \\ \\ \end{array}\right\} L \end{array} \tag{10.43}$$

and denote by U, X_a, and L the upper, center, and lower sections, respectively, as indicated. The choice

$$R_{N+1} = \bar{X}^t X = \bar{U}^t U + \bar{X}_a^t X_a + \bar{L}^t L \tag{10.44}$$

corresponds to the so-called AC equations, because the matrix obtained is Hermitian and Toeplitz, like the theoretical AC matrix.

Another choice is

$$R_{N+1} = \bar{X}_a^t X_a \tag{10.45}$$

which leads to the so-called covariance equations. The matrix is near Toeplitz. To complete the picture, the prewindowed equations correspond to

$$R_{N+1} = \bar{X}_a^t X_a + \bar{L}^t L = \bar{X}^t X - \bar{U}^t U \tag{10.46}$$

and the postwindowed equations to

$$R_{N+1} = \bar{X}_a^t X_a + \bar{U}^t U = \bar{X}^t X - \bar{L}^t L \tag{10.47}$$

Of these four types of equations, the covariance type is, in general, the most efficient for resolution with short data records because it does not imply any assumptions on the data outside the observation interval. The method is well suited to adaptive implementation, as emphasized in the previous chapters.

The values taken above for the matrix R_{N+1} are based on forward linear prediction. If the signal is stationary, the coefficients of the backward prediction filter are identical to the coefficients of the forward prediction filter, but reversed in time and conjugated, in the complex case. Therefore, the most complete estimation procedure is based on minimizing the sum of the forward and backward prediction error powers, and the corresponding expression for the matrix R_{N+1} is

$$R_{N+1} = \bar{X}_a^t X_a + \bar{X}_b^t X_b \tag{10.48}$$

where X_b is the backward $(N_0 - N) \times (N+1)$ matrix

$$X_b = \begin{bmatrix} \bar{x}(N_0) & \cdots & \bar{x}(N_0 - N) \\ \vdots & \cdots & \vdots \\ \bar{x}(N+1) & \cdots & \bar{x}(1) \end{bmatrix}$$

An efficient approach to solve the normal equations consists of calculating the reflection coefficients. The LS lattice structure analysis given in Section 8.1 leads to the calculation of two sets of reflection coefficients $k_{ai}(n)$ and $k_{bi}(n)$ for forward and backward prediction, respectively, defined by equation (8.16). For stationary signals, the PARCOR coefficient k_i is unique and given by expression (5.61) in Chapter 5. Estimates for k_i from $k_{ai}(n)$ and $k_{bi}(n)$ can be obtained in different ways. A first estimate is the geometric mean

$$|k_i| = \sqrt{|k_{ai}(n)||k_{bi}(n-1)|} \tag{10.49}$$

which, according to (8.50), corresponds to the normalized lattice structure.

Another estimate is the harmonic mean, corresponding to the so-called *Burg* method [1]

$$\frac{1}{k_i} = \frac{1}{k_{ai}(n)} + \frac{1}{k_{bi}(n)} \tag{10.50}$$

Accordingly, the coefficients are calculated for complex data by

$$k_i = \frac{2 \sum_{j=i+1}^{N_0} e_{a,(i-1)}(j)\bar{e}_{b,(i-1)}(j-1)}{\sum_{j=i+1}^{N_0} |e_{a,(i-1)}(j)|^2 + |e_{b,(i-1)}(j-1)|^2} \tag{10.51}$$

Their absolute values are bounded by unity, which corresponds to a stable prediction error filter and a finite spectrum estimate at all frequencies.

The Burg procedure is summarized as follows:

1. Calculate k_1 by

$$k_1 = \frac{2 \sum_{j=2}^{N_0} x(j)\bar{x}(j-1)}{\sum_{j=2}^{N_0} |x(j)|^2 + |x(j-1)|^2} \tag{10.52}$$

2. For $1 \leqslant i \leqslant N$:
   a. Calculate the prediction errors by

$$e_{ai}(j) = e_{a(i-1)}(j) - k_i e_{b(i-1)}(j-1)$$
$$e_{bi}(j) = e_{b(i-1)}(j-1) - k_i e_{a(i-1)}(j) \tag{10.53}$$

b. Calculate the reflection coefficients k_{i+1} by equation (9.51)
c. Calculate the prediction coefficients by

$$a_{ii} = k_i$$
$$a_{ji} = a_{j(i-1)} - k_i a_{(i-j)(i-1)}; \quad 1 \leqslant j \leqslant i-1 \tag{10.54}$$

Independently of adaptive LS lattice filters, the procedure itself can be made adaptive by the introduction of a weighting factor W in the summations of the numerator and denominator of equation (9.51). The updating equations are

$$\begin{aligned} D_i(n+1) &= WD_i(n) + |e_{a(i-1)}(n+1)|^2 + |e_{b(i-1)}(n)|^2 \\ k_i(n+1) &= k_i(n) + \frac{1}{D_i(n+1)}[e_{ai}(n+1)\bar{e}_{b(i-1)}(n) \\ &\quad + e_{a(i-1)}(n+1)\bar{e}_{bi}(n+1)] \end{aligned} \tag{10.55}$$

The above updating technique can be simplified by making constant the variable $D_i(n+1)$, which leads to the gradient approach defined by expression (5.116) of Chapter 5.

The adaptive technique associated with the geometric mean approach (10.49) is based on the adaptive normalized lattice algorithms described in Section 8.5.

An interesting aspect of the lattice approach is that it provides the linear prediction coefficients for all orders from 1 to N, and, consequently, the corresponding spectral estimations.

Given a data record of length N_0, the selection of the optimal predictor order N is not straightforward. It if is too small, a smoothed spectrum is obtained, which produces a poor resolution. On the contrary, if it is too large, spurious peaks may appear in the spectrum.

The results given in Section 5.7 indicate that the predictor order N can be chosen as the value which corresponds to the maximum of the reflection coefficients $|k_i|$. Another choice is based on the prediction error power; it minimizes the final prediction error (FPE) [11]

$$\text{FPE}(N) = \frac{N_0 + N + 1}{N_0 - N_{-1}} E_N \tag{10.56}$$

Experience has shown that a reasonable upper bound for N is [12]

$$N \leqslant \frac{N_0}{3} \tag{10.57}$$

The resolution of the AR method strongly depends on the noise level. Extending the results in Section 5.2, we can state that it is not possible to distinguish between two real sinusoids separated by Δf if the noise power exceeds the limit σ_L^2 given approximately by

$$\sigma_L^2 \approx 40(\Delta f)^3 \tag{10.58}$$

when the predictor order N is twice the number M of real sinusoids in the signal. When N increases, the results of Section 5.8 show that the limit σ_L^2 increases as a function of $N - 2M$.

For a single sinusoid in noise, the 3-dB bandwidth of the prediction error filter can be roughly estimated by

$$B_{3\mathrm{AR}} \approx \frac{1}{N^2 \cdot \mathrm{SNR}} \tag{10.59}$$

where SNR is the signal-to-noise ratio and the order N is assumed to be large. The derivation is based on the hypothesis that the signal power can be approximated by the product of the maximum value of $S_{\mathrm{AR}}(f)$ by $B_{3\mathrm{AR}}$. Note that, for N large, the prediction error filter closely approximates the notch filter of Section 5.7.

The parameter $B_{3\mathrm{AR}}$ can represent the resolution of the method. The comparison with the periodogram is illustrated by the ratio

$$\frac{B_{3\mathrm{AR}}}{B_{3\mathrm{N}}} = \frac{1}{\mathrm{N} \cdot \mathrm{SNR}} \tag{10.60}$$

which clearly shows the advantage of the AR method in resolution for large SNR values.

The variance of the AR spectrum estimate is shown to be proportional to $\frac{1}{N^2 N_0}$ for an AR signal and to $\frac{1}{N^2 N_0^2}$ for a signal composed of sinusoids in noise.

A noisy signal expressed by

$$x(n) = x_p(n) + e(n) \tag{10.61}$$

where $x_p(n)$ is a predictable signal satisfying the recursion

$$x_p(n) = \sum_{i=1}^{M} a_i x_p(n-i) \tag{10.62}$$

and $e(n)$ a noise, can be viewed as an ARMA signal because

$$x(n) = \sum_{i=1}^{M} a_i x(n-i) + e(n) - \sum_{i=1}^{M} a_i e(n-i) \tag{10.63}$$

The gain obtained by increasing the predictor order corresponds to an approximation of the MA section of the model [13]. However, a direct ARMA modeling approach can be more efficient.

10.7. ARMA MODELING

The spectrum estimation is

$$S_{\mathrm{ARMA}}(f) = E_N \frac{\left|1 + \sum_{i=1}^{N} b_i e^{-j2\pi i f}\right|^2}{\left|1 + \sum_{i=1}^{N} a_i e^{-j2\pi i f}\right|^2} \tag{10.64}$$

where E_N is an error power and the signal $x(n)$ is assumed to follow the model

$$x(n) = \sum_{i=1}^{N} a_i x(n-i) + e(n) + \sum_{i=1}^{N} b_i e(n-i) \tag{10.65}$$

A detailed analysis of ARMA signals is provided in Section 2.6.

The results can be used to calculate the model coefficients from an estimate of the $2N+1$ first values of the signal ACF. The spectrum is then calculated from (10.64). Recall, that the spectrum can be calculated without explicitly determining the MA coefficients. The AR coefficients are found from the extended normal equations (2.79), the ACF of the auxiliary signal is derived from (2.89), and the spectrum is obtained from (2.92). All equations are from Chapter 2.

Adaptive aspects of ARMA modeling are dealt with in Section 4.15, where the application of the LMS algorithm is discussed, and in Section 7.12, which covers FLS techniques. A particular simplified case worth pointing out is the notch filter presented in Section 5.8, whose transfer function is

$$H_N(z) = \frac{1 + \sum_{i=1}^{N} a_i z^{-i}}{1 + \sum_{i=1}^{N} a_i (1-\varepsilon)^i z^{-i}} \tag{10.66}$$

where the notch parameter ε is a small positive scalar ($0 \leqslant \varepsilon \ll 1$). When predictable signals are analyzed, because of the respective locations of its zeros and poles in the z-plane, it can be a useful intermediate between the prediction error filter, whose zeros are prevented from reaching the unit circle by the noise, and the minimum eigenfilter, whose zeros are on the unit circle.

The coefficients can be derived from a set of N_0 data through iterative techniques. The filter can also be made adaptive using gradient-type algorithms [14]. With LS the nonlinear character of the problem can be overcome in a way which illustrates the flexibility of that technique [15]. Consider the diagram in Figure 10.5. The input signal is fed to the recursive section of the filter first. Then the output sequence obtained is fed to a prediction filter, whose coefficients are updated using an FLS algorithm.

If the same coefficients at each time are also used in the recursive section, an FLS adaptive notch filter is achieved. The value of the fixed-notch parameter ε reflects the a priori information available about the signal: for sinusoids in noise it can be close to zero, whereas noiselike signals lead to choosing ε close to unity. The results obtained for two sinusoids in noise are shown in Figure 10.6. The SNR is 3 dB and $\varepsilon = 0.1$. The coefficient learning curves and the locations of the corresponding filter zeros in the complex plane demonstrate that the two sinusoids are clearly identified by an order 4 filter.

In that approach, which can be used for efficient real-time analysis, the recursive section placed in front of the predictor enhances the lines in the spectrum and helps the predictor operation. A similar effect can be obtained with signal and noise space methods [16].

10.8. SIGNAL AND NOISE SPACE METHODS

The signal AC matrix estimates used in AR methods are degraded by noise. Improvements can be expected from attempts to remove that degradation [17].

Assume that M real sinusoids are searched in a signal. From the set of N_0 data an estimate R_N of the $N \times N$ AC matrix is computed. Its eigendecomposition, as seen in Chapter 3, is

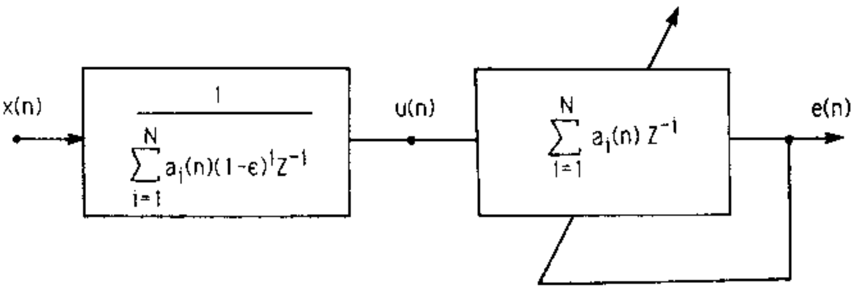

x(n)
$\dfrac{1}{\sum_{i=1}^{N} a_i(n)(1-\varepsilon)^i z^{-i}}$
u(n)
$\sum_{i=1}^{N} a_i(n) z^{-i}$
e(n)

FIG. 10.5 FLS adaptive notch filter.

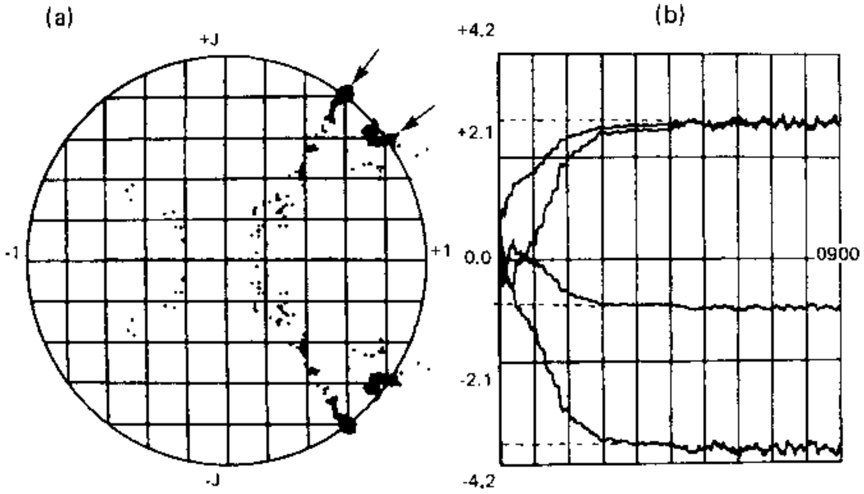


FIG. 10.6 Identification of two sinusoids in noise by an adaptive notch filter: (a) filter zeros in the complex plane, (b) coefficient learning curves.

$$R_N = \sum_{i=1}^{N} \lambda_i U_i U_i^t, \quad \lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_N \tag{10.67}$$

and the prediction coefficient vector is

$$A_N = R_N^{-1} r_N^a = \sum_{i=1}^{N} \frac{1}{\lambda_i} U_i U_i^t r_N^a \tag{10.68}$$

For $N > 2M$, if R_N were the true AC matrix, the $N - 2M$ last eigenvalues would be zero. Therefore the $N - 2M$ smallest eigenvalues of R_N can be assumed to be associated with the noise space, and the optimum approximation of the true AC matrix is R'_N obtained by

$$R'_N = \sum_{i=1}^{2M} \lambda_i U_i U_i^t \tag{10.69}$$

Thus, an improved estimate of the prediction coefficients is

$$A'_N = \sum_{i=1}^{2M} \frac{1}{\lambda_i} U_i (U_i^t r_N^a) \tag{10.70}$$

If not known, the number of sinusoids can be determined from the observation of the eigenvalues λ_i. Concerning the computational complexity, the eigenvalue decomposition is a significant load.

Another estimate of the prediction coefficient vector can be found by noticing that, for predictable signals in the absence of noise, the prediction error is zero and the filter coefficient vector is orthogonal to all signal vectors [18]. Therefore the estimate A''_N is a vector satisfying

$$U_i^t \begin{bmatrix} 1 \\ -A''_N \end{bmatrix} = 0, \quad 1 \leqslant i \leqslant 2M \tag{10.71}$$

where U_i now denotes an $(N+1)$-element eigenvector of the estimated $(N+1) \times (N+1)$ AC matrix R_{N+1}. The eigenvectors $U_i (1 \leqslant i \leqslant 2M)$ span the signal space.

In matrix form the system (10.71) is

$$\begin{bmatrix} u_{12} & \cdots & u_{1(N+1)} \\ u_{22} & \cdots & u_{2(N+1)} \\ \vdots & \cdots & \\ u_{2M2} & \cdots & u_{2M(N+1)} \end{bmatrix} \begin{bmatrix} a''_1 \\ a''_2 \\ \vdots \\ a''_N \end{bmatrix} = \begin{bmatrix} u_{11} \\ u_{22} \\ \vdots \\ u_{2M1} \end{bmatrix} \tag{10.72}$$

or, in concise form,

$$U_{2M,N} A''_N = U_{2M,1}$$

The system is underdetermined, since there are $2M$ equations in N unknowns and $N > 2M$. The minimum norm solution is given by expression (3.73) in Chapter 3, which here yields

$$A''_N = U^t_{2M,N} [U_{2M,N} U^t_{2M,N}]^{-1} U_{2M,1} \tag{10.73}$$

Because the eigenvectors are orthogonal and have unit norm, the above equation simplifies to

$$A''_N = (1 - U^t_{2M,1} U_{2M,1})^{-1} U^t_{2M,N} U_{2M,1} \tag{10.74}$$

Once the estimation of the prediction coefficients has been calculated, the spectrum is obtained by (10.42).

A further, efficient, approach to exploit the orthogonality of signal and noise spaces consists of searching the set of M cisoids which are the most closely orthogonal to the noise space, spanned by the $N - M$ eigenvectors $U_i (M + 1 \leqslant i \leqslant N)$ [19]. The M frequencies are taken as the peaks of the function

$$S(\omega) = \frac{1}{\sum_{i=M+1}^{N} |\bar{F}^t(\omega) U_i|^2} \tag{10.75}$$

where $F(\omega)$ is the cisoid vector

$$\bar{F}^t(\omega) = [1, e^{j\omega}, \dots, e^{j(N-1)\omega}]$$

Weighting factors can also be introduced in the above summation.

The above methods, like those described in the previous sections, have a resolution which is linked to the number N_0 of the signal values available and the signal-to-noise ratio SNR. An estimation is given by the expression

$$E[(\Delta f)^2] = \frac{1}{N_0^2 \cdot \text{SNR}} \tag{10.76}$$

The signal and noise space methods are essentially suited to batch processing. Nevertheless, they provide a deep insight into the structures of real signals and therefore are useful for adaptive filter design.

10.9. ESTIMATION BOUNDS

Adaptive techniques can be used for estimation purposes, and their performance can be tested against estimation bounds in these cases [12–20, 22].

Let us consider an indirect estimation procedure in which a set of N parameters θ_i is derived from a set of N measurements γ_i. These measurements are related to the parameters by

$$\gamma_i = f_i(\theta_1, \dots, \theta_N), \quad 1 \leqslant i \leqslant N \tag{10.77}$$

The absence of perturbations corresponds to optimal values, and a Taylor series expansion in the vicinity of the optimum yields

$$\gamma_i - \gamma_{i,\text{opt}} = \frac{\partial f_i}{\partial \theta_1}(\theta_{1,\text{opt}})[\theta_1 - \theta_{1,\text{opt}}] + \cdots + \frac{\partial f_i}{\partial \theta_N}(\theta_{N,\text{opt}})[\theta_N - \theta_{N,\text{opt}}] \tag{10.78}$$

or, in matrix form

$$(\Gamma - \Gamma_{\text{opt}}) = M_G^t(\theta - \theta_{\text{opt}}) \tag{10.79}$$

where M_G is the parameter measurement function gradient matrix.

The measurement covariance matrix is

$$E[(\Gamma - \Gamma_{\text{opt}})(\Gamma - \Gamma_{\text{opt}})^t] = M_G^t E[(\theta - \theta_{\text{opt}})(\theta - \theta_{\text{opt}})^t] M_G \tag{10.80}$$

Assuming unbiased estimation, the parameter covariance matrix, denoted var$\{\theta\}$, is

$$\mathrm{var}\{\theta\} = (M_G^{-1})^t \mathrm{var}\{\Gamma\} M_G^{-1} \tag{10.81}$$

This provides a lower bound for the variance of the parameter estimation θ, obtained from the measurement vector Γ in the presence of perturbations. If complex functions are involved, the transpose operation is replaced by conjugate transpose.

If the numbers of parameters and measurements are different, pseudoinverses can be used. For example, with more measurements than parameters, equation (10.81) can be written as

$$\mathrm{var}\{\theta\} = [M_G M_G^t]^{-1} M_G \mathrm{var}\{\Gamma\} M_G^t [M_G M_G^t]^{-1} \tag{10.82}$$

An important simplification occurs when the perturbation in the measurements is just caused by a white noise with power σ_b^2, because equation (10.82) becomes

$$\mathrm{var}\{\theta\} = [M_G M_G^t]^{-1} \sigma_b^2 \tag{10.83}$$

For example, let us consider the cisoid in noise

$$x(n) = ae^{jn\omega} + b(n) \tag{10.84}$$

and assume the amplitude a and the angular frequency ω have to be estimated from N samples ($1 \leqslant n \leqslant N$). The $2 \times N$ matrix M_G is

$$M_G = \begin{bmatrix} \dfrac{\partial f_1}{\partial a} & \cdots & \dfrac{\partial f_N}{\partial a} \\ \dfrac{\partial f_1}{\partial \omega} & \cdots & \dfrac{\partial f_N}{\partial \omega} \end{bmatrix} = \begin{bmatrix} e^{j\omega} & \cdots & e^{jN\omega} \\ jae^{j\omega} & \cdots & jNae^{jN\omega} \end{bmatrix}$$

and therefore

$$[M_G \bar{M}_G^t] = \begin{bmatrix} N & -ja\dfrac{N(N+1)}{2} \\ ja\dfrac{N(N+1)}{2} & a^2\dfrac{N(N+1)(2N+1)}{6} \end{bmatrix} \tag{10.85}$$

The variances of the estimates are

$$\mathrm{var}\{a\} = 2\sigma_b^2 \frac{2N+1}{N(N-1)}, \quad \mathrm{var}\{\omega\} = 2\sigma_b^2 \frac{6}{a^2 N(N^2-1)} \tag{10.86}$$

The bound in equation (10.81) can also be expressed as

$$\mathrm{var}\{\theta\} = [M_G \mathrm{var}\{\Gamma\}^{-1} M_G^t]^{-1}$$

This expression is reminiscent of the definition of joint probability densities of Gaussian variables. In fact, the above procedure can be generalized in estimation theory, using the log-likelihood function [20, 23].

Let $\Pr(\Gamma|\theta)$ denote the conditional joint probability density function of the random vector represented by the measurement vector Γ. The log-likelihood function $L(\theta)$ is defined by

$$L(\theta) = \ln[\Pr(\Gamma|\theta)] \tag{10.87}$$

The matrix I_{nf} of the derivatives of the log-likelihood function is called Fisher's information matrix:

$$I_{\mathrm{nf}} = -E\begin{bmatrix} \left(\dfrac{\partial L}{\partial \theta_1}\right)^2 & \cdots & \dfrac{\partial L}{\partial \theta_1}\dfrac{\partial L}{\partial \theta_N} \\ \vdots & & \vdots \\ \dfrac{\partial L}{\partial \theta_N}\dfrac{\partial L}{\partial \theta_1} & \cdots & \left(\dfrac{\partial L}{\partial \theta_N}\right)^2 \end{bmatrix} \tag{10.88}$$

For example, let us consider the case of M-D Gaussian signals with probability density $p(x)$ expressed by [equation (2.135) of Chapter 2]

$$P(X) = \frac{1}{(2\pi)^{n/2}}\frac{1}{(\det R)^{1/2}}\exp[-\tfrac{1}{2}(X-m)^t R^{-1}(X-m)]$$

where the AC matrix R and the mean vector m are functions of a set of variables θ. The information matrix elements are

$$\inf(k,l) = \frac{1}{2}\operatorname{trace}\left[R^{-1}\frac{\partial R}{\partial \theta_k}R^{-1}\frac{\partial R}{\partial \theta_l}\right] + \frac{\partial m^t}{\partial \theta_k}R^{-1}\frac{\partial m}{\partial \theta_l} \tag{10.89}$$

The lower bound of the variance of the parameter vector estimation is called the Cramer–Rao bound, and it is defined by

$$\mathrm{CRB}(\theta) = \operatorname{diag}[I_{\mathrm{nf}}^{-1}] \tag{10.90}$$

When the functional form of the log-likelihood function is known, for unbiased estimates, a lower bound of the parameter estimates can be calculated, and the following set of inequalities hold:

$$\operatorname{var}\{\theta_i\} \geqslant \mathrm{CRB}(\theta_i), \quad 1 \leqslant i \leqslant N \tag{10.91}$$

An unbiased estimator is said to be efficient if its variance equals the bound.

10.10. CONCLUSION

The analysis techniques discussed in this chapter provide a set of varied and useful tools to investigate the characteristics of signals. These characteristics are helpful in studying, designing, and implementing adaptive filters. In particular, they can provide pertinent information on how to select filter parameter values or to assess the dynamic range of the variables, which is necessary for word-length determination.

As an illustration, consider the initial value E_0 of the prediction error energy in FLS algorithms, used for example in prediction applications. As pointed out in Chapter 6, it controls the initial adaptation speed of the filter. If the SNR is poor, it does not help to take a small value, and E_0 can be chosen close to the signal power; on the contrary, with high SNRs, small values of the initial error energy make the filter fast and can lead to quick detection of sinusoids for example.

Several analysis techniques, particularly the AR method, are well suited to adaptive approaches, which lead to real-time signal analysis.

EXERCISES

1. Calculate the order $N = 16$ DFT of the sequences

$$x_1(n) = \sin\left(2\pi \frac{2.5}{16} n\right), \quad 0 \leqslant n \leqslant 15$$
$$x_2(n) = \sin\left(2\pi \frac{3.5}{16} n\right), \quad 0 \leqslant n \leqslant 15$$
$$x_3(n) = x_1(n) + x_2(n)$$

   Discuss the possibility of recognizing the sinusoids in the spectrum.

2. The real signal $x(n)$ is analyzed with an N-point DFT operator. Show that the signal power spectrum can be estimated by

$$S(k) = X(k)X(N-k)$$

   where $X(k)$ is the DFT output with index $k (0 \leqslant k \leqslant N-1)$. If $x(n) = b(n)$, where $b(n)$ is a white noise with power σ_b^2, calculate the mean and variance of the estimator $S(k)$.

   Now assume that the signal is

$$x(n) = \sin\left(2\pi \frac{k_0}{N} n\right) + b(n), \quad 1 < k_0 < \frac{N}{2}$$

   with k_0 integer. Calculate the mean and variance of the estimator and comment on the results. Is the analysis technique efficient?

3. A signal has ACF $r(0) = 1.0$, $r(1) = 0.866$, $r(2) = 0.5$. Perform the eigenvalue decomposition of the 3×3 AC matrix and give the harmonic decomposition of the signal. How is it modified if
   (a) a white noise with power σ^2 is added to the signal:
   (b) The ACF $r(p)$ is replaced by $0.9^P r(p)$?
   Give the shape of the spectrum using expression (10.34).

4. Consider the signal sequence

$n =$	1	2	3	4	5	6
$x(n) =$	1.05	0.72	0.45	-0.32	-0.61	-0.95

   Perform the damped sinusoid decomposition following the procedure in Section 10.5 and calculate the spectrum $S_{\mathrm{DS}}(f)$.

5. For the signal

$n =$	1	2	3	4	5	6	7	8	9	10
$x(n) =$	1.41	1.43	1.35	1.22	1.14	0.91	0.84	0.67	0.51	0.31

   calculate the matrix R_4 according to expression (10.45) and use it to derive three forward prediction coefficients. Calculate the AR spectrum and draw the curve $S_{\mathrm{AR}}(f)$ versus frequency.

   Repeat the above operations with R_4 calculated according to the forward-backward technique (10.48). Compare the spectra obtained with both approaches.

6. Consider the cisoids in noise

$$x(n) = e^{jn\omega_1} + e^{jn\omega_2} + b(n)$$

   and assume the angular frequencies have to be estimated from N samples. Calculate the variance estimation bounds and show the importance of the quantity $\omega_2 - \omega_1$.

   Perform the same calculations when a phase parameter φ_1 is introduced

$$x(n) = e^{j(n\omega_1+\varphi_1)} + e^{jn\omega_2} + b(n)$$

   Comment on the impact of the phase parameter.

REFERENCES

1. S. L. Marple, *Digital Spectral Analysis with Applications*, Prentice-Hall, Englewood Cliffs, N.J., 1986.
2. S. M. Kay and C. Demeure, "The High Resolution Spectrum Estimator—A Subjective Entity," *Proc. IEEE* No. 72, 1815–1816 (December 1984).
3. P. Welch, "The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging of Short, Modified Periodograms," *IEEE Trans.* **AU-15**, 70–73 (June 1967).
4. R. B. B. Blackman and J. W. Tukey, *The Measurement of Power Spectra from the Point of View of Communications Engineering*. Dover, New York, 1959.
5. H. Clergeot, "Choice between Quadratic Estimators of Power Spectra," *Ann. Telecomm.* **39**, 113–128 (1984).

6. J. Capon, ‘‘High Resolution Frequency Wave Number Spectrum Analysis,’’ *Proc. IEEE* **57**, 1408–1418 (August 1969).
7. M. A. Lagunas-Hernandez and A. Gazull-Llampallas, ‘‘An Improved Maximum Likelihood Method for Power Spectral Density Estimation,’’ *IEEE Trans.* **ASSP-32**, 170–173 (February 1984).
8. V. F. Pisarenko, ‘‘The Retrieval of Harmonics from a Covariance Function,’’ *Geophys. J. Roy. Astronom. Soc.* **33**, 347–366 (1967).
9. F. B. Hildebrand, *Introduction to Numerical Analysis*, McGraw-Hill, New York, 1956, Chap. 9.
10. S. Haykin, ed., *Nonlinear Methods of Spectral Analysis*, Springer-Verlag, New York, 1983.
11. H. Akaike, ‘‘A New Look at the Statistical Model Identification,’’ *IEEE Trans.* **AC-19**, 716–723 (1974).
12. S. W. Lang and J. H. McClellan, ‘‘Frequency Estimation with Maximum Entropy Spectral Estimators,’’ *IEEE Trans.* **ASSP-28**, 716–724 (December 1980).
13. J. A. Cadzow, ‘‘Spectral Estimation: An Overdetermined Rational Model Equation Approach,’’ *Proc. IEEE* **70**, 907–939 (September 1982).
14. D. V. Bhaskar Rao and S. Y. Kung, ‘‘Adaptive Notch Filtering for the Retrieval of Sinusoids in Noise,’’ *IEEE Trans.* **ASSP-32**, 791–802 (August 1984).
15. J. M. Travassos-Romano and M. Bellanger, ‘‘FLS Adaptive Notch Filtering, *Proc.* ECCTD Conf., Paris, 1987.
16. G. Bienvenu and L. Kopp, ‘‘Optimality of High Resolution Array Processing Using the Eigensystem Approach,’’ *IEEE Trans.* **ASSP-31**, 1235–1243 (October 1983).
17. D. W. Tufts and R. Kumaresan, ‘‘Estimation of Frequencies of Multiple Sinusoids: Making Linear Prediction Perform Like Maximum Likelihood,’’ *Proc. IEEE* **70**, 975–989 (1983).
18. R. Kumaresan and D. W. Tufts, ‘‘Estimating the Angles of Arrival of Multiple Plane Waves,’’ *IEEE Trans.* **AES-19**, 134–139 (1983).
19. M. Kaveh and A. Barabell, ‘‘The Statistical Performance of the Music and Minimum Norm Algorithms in Resolving Plane Waves in Noise,’’ *IEEE Trans.* **ASSP-34**, 331–341 (April 1986).
20. H. L. Van Trees, *Detection, Estimation and Modulation Theory*, Wiley, New York, 1971.
21. J. Casar and G. Cybenko, ‘‘An SVD Analysis of Resolution Limits for Harmonic Retrieval Problem,’’ in F. Depretene, *SVD and Signal Processing*, North Holland, Amsterdam, 1988, pp. 461–465.
22. S. Kay and J. Makhoul, ‘‘On the Statistics of the Estimated Reflection Coefficients of an Autoregressive Process,’’ *IEEE Trans.* **ASSP-31**, 1447–1455 (December 1983).
23. Y. X. Yao and S. M. Pandit, ‘‘Cramér–Rao Lower Bounds for a Damped Sinusoidal Process,’’ *IEEE Trans.* on Signal Processing **43**, 878–885 (April 1995).

11
Circuits and Miscellaneous Applications

The integrated circuits, programmable microprocessors, and array processors designed for general signal processing can be used for the implementation of adaptive filters. However, several specific aspects can make the realization of dedicated architectures worthwhile. A major feature of adaptive techniques is the real-time updating of a set of internal variables, with the related checking operations. The optimization of some of the functions included in that process may be justified; a typical function is the multiplication of the elements of a vector by a scalar. An important point also is the introduction of arithmetic operations which are not widely used in other areas of signal processing, namely division and, to a lesser extent, square root.

11.1. DIVISION AND SQUARE ROOT

Division can be implemented in arithmetic logic units (ALUs) as a sequence of shifts, subtractions, and tests. The procedure is time consuming, and a more efficient approach is obtained by dedicated circuitry.

Let n and d be two positive numbers satisfying $0 < n < d$. To calculate n/d, use the following algorithm [1]:

$$t_0 = 2n - d, \qquad \begin{matrix} t_0 < 0, q_1 = 0 \\ t_0 \geqslant 0, q_1 = 1 \end{matrix}$$

$$t_1 = 2t_0 - (2q_1 - 1)d; \qquad \begin{matrix} t_1 < 0, q_2 = 0 \\ t_1 \geqslant 0, q_2 = 1 \end{matrix} \tag{11.1}$$

$$t_i = 2t_{i-1} - (2q_i - 1)d; \qquad \begin{array}{l} t_i < 0, q_{i+1} = 0 \\ t_i \geqslant 0, q_{i+1} = 1 \end{array}$$

It is readily verified that the following equations hold:

$$n = d\left(\sum_{j=1}^{i} q_j 2^{-j}\right) + 2^{-i}\frac{t_i + d}{2}, \qquad |t_i| < d \tag{11.2}$$

The bits q_j are the most significant bits of the quotient $q = n/d$. The algorithm can be implemented in a sequential manner as shown in Figure 11.1, assuming the same word length for n and d.

The number of full adders equals the number of bits of the factors plus one. A parallel realization leads to an array of adders as with multipliers.

As an illustration, consider the following example: $n = 9 = 001001$; $d = 21 = 010101$.

$$\begin{array}{rrl}
 & 001001 & (n) \\
 & 010010 & (2n) \\
 & \underline{+\ 101011} & (-d;\ 2\text{'s complement}) \\
 & 111101 & (t_0 = 2n - d) \\
q_1 = 0 & (1)\ 111010 & (2t_0) \\
 & \underline{+\ 010101} & \\
 & 001111 & (t_1) \\
q_2 = 1 & (0)\ 011110 & (2t_1) \\
 & \underline{+\ 101011} & (-d) \\
 & 001001 & (t_2) \\
q_3 = 1 & (0)\ 010010 &
\end{array} \tag{11.3}$$

The result is $q = \frac{3}{7} = 0.011\ldots$.

When the word length of the divider d is smaller than the word length of the dividend n, it may be advantageous to perform the operation as $q = (\frac{1}{d})Xn$, which corresponds to an inverse calculation followed by a multiplication.

The square-root extraction can be viewed as a division by an unknown divider. The two operations have many similarities, and in both cases the most significant bits of the result are obtained first.

In order to show how the square-root extraction can be performed recursively, let us assume that the square root of a given number X has P bits and that i bits $(s_0, \ldots s_{i-1})$ are available after stage i of the extraction procedure. The remainder R_i is expressed by

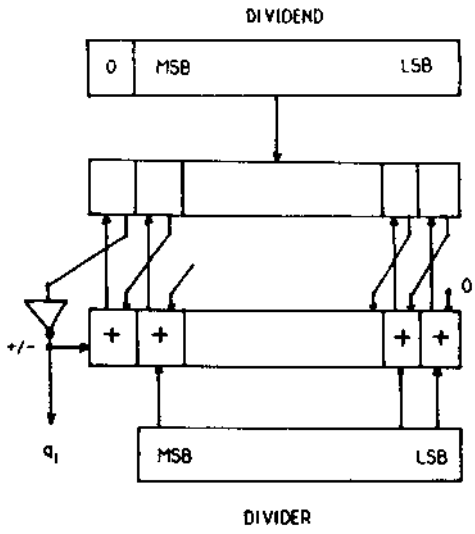


FIG. 11.1 A sequential divider.

$$R_i = X - (S_i 2^{P-i})^2 \tag{11.4}$$

where

$$S_i = \sum_{j=0}^{i-1} s_{i-1-j} 2^j$$

At the next stage, the remainder R_{i+1} is

$$R_{i+1} = X - [(2S_i + s_i)2^{P-(i+1)}]^2 \tag{11.5}$$

The unknown s_i is a binary number, and thus $s_i^2 = s_i$. Now, expanding the product on the right side of (11.5) yields

$$R_{i+1} = R_i - (4S_i + 1)s_i 2^{2(P-i-1)} \tag{11.6}$$

Consequently, to obtain s_i it suffices to calculate the quantity

$$Q_{i+1} = R_i - (4s_i + 1)2^{2(P-i-1)} \tag{11.7}$$

and take its sign bit.

Hence the procedure to perform the square-root extraction for a number X having N bits ($N \leqslant 2P$) is as follows:

Initialization: $R_0 = X$, $S_0 = 0$
For $i = 0, \ldots, P-1$,

$$
\begin{aligned}
&Q_{i+1} = R_i - (4S_i + 1)2^{2(P-i-1)} \\
&Q_{i+1} \geqslant 0, \quad s_i = 1, \quad R_{i+1} = Q_{i+1}, \quad S_{i+1} = 2S_i + 1 \\
&Q_{i+1} < \ 0, \quad s_i = 0, \quad R_{i+1} = R_i, \quad S_{i+1} = 2S_i
\end{aligned}
\tag{11.8}
$$

The desired square root is S_p.

Example

$$
\begin{aligned}
&X = 25 = 011001, \quad N = 6 = 2P \\
&Q_1 = 001001, \quad s_0 = 1, \quad R_1 = 001001, \quad S_1 = 000001 \\
&Q_2 = 001001 - 010100, \quad s_1 = 0, \quad R_2 = R_1, \quad S_2 = 000010 \\
&Q_3 = 0, \quad s_2 = 1, \quad R_3 = 0, \quad S_3 = 000101 = 5
\end{aligned}
$$

The procedure can be implemented on a standard ALU as a sequence of shift, additions, subtractions, and tests. Dedicated circuits can also be worked out for sequential or parallel realization.

Overall, considering the algorithms (11.1) and (11.8) it appears that, using a standard ALU, the division is more complex than the multiplication because it requires a test to decide between addition and subtraction at each stage; the square root is more complex than the division because it requires an addition, a subtraction, and two shifts at each stage. However, if a dedicated circuit is built, the test needed in the division can be included in the logic circuitry, and the division becomes equivalent to the multiplication. The square-root extraction is still more complex than the division.

11.2. A MULTIBUS ARCHITECTURE

Signal processing machines are characterized by the separation of the data and control paths. For adaptive processing, additional flexibility is desirable, due to the real-time updating of the internal variables. Three data paths can be distinguished: two for the factors of the arithmetic operations, and one for the results. Therefore, a high level of efficiency and speed is obtained with the four-bus programmable architecture sketched in Figure 11.2.

The data buses A and B are used for the arithmetic factors, bus C for the results. The various system units, ALU, multiplier, and memories interact with these buses in an adequate manner. An interface unit handles the data exchanges with the external world. The control unit is connected through the instruction bus I to the system units and external control signal sources.

The multibus approach brings a certain level of complexity in hardware and software as well. However, the parallelism introduced that way offers a pipelining capacity which leads to fast and efficient realizations [2, 3].

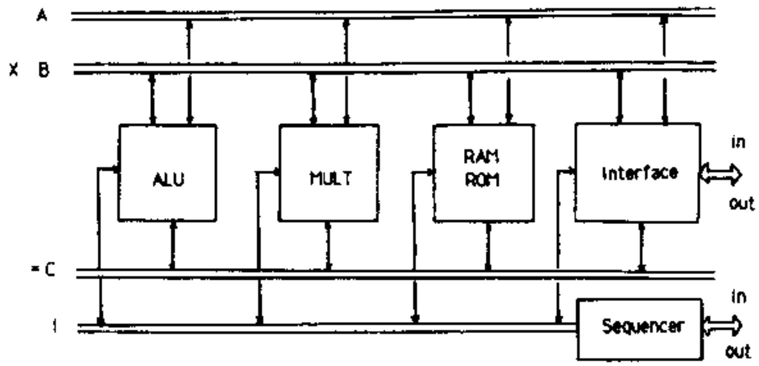


FIG. 11.2 A four-bus architecture for adaptive processing.

A wide range of programmable integrated microsignal processors is now available, as well as specific parts to build multichip architectures. Machines can now be designed for any kind of application. A selection of applications from various fields is given below.

11.3. LINE CANCELING AND ENHANCEMENT

A frequently encountered problem is the canceling of a line while preserving the rest of the spectrum. As mentioned in Section 5.8, the notch filter is the appropriate structure. If the frequency of the line is not known or changing with time, an adaptive version can be used, with gradient or FLS algorithms as pointed out in Section 10.7. Good performance can be obtained under a wide range of conditions [4].

The recursive section of the notch filter actually performs a line enhancement. The general approach is based on linear prediction, as shown in Figure 11.3.

Let us assume that the signal $x(n)$ consists of M sinusoids in noise. The output $\tilde{x}(n)$ of the adaptive prediction filter $A(z)$ contains the same spectral lines, with virtually no deviations in amplitudes and phases, provided the filter order N exceeds $2M$ with a sufficient margin. However, as seen in Section 5.2, the noise component power is reduced in $\tilde{x}(n)$ since the power of the output $e(n)$ is minimized. The delay Δ in front of the prediction filter is chosen as a function of the correlation radius p_0 of the noise ($\Delta \geqslant p_0$); in case of white noise, a one-step predictor is adequate [5].

The improvement in SNR for the enhancer output $\tilde{x}(n)$ is the enhancement gain G_e, which can be calculated using the results in Chapter 5; it is proportional to the prediction filter order.

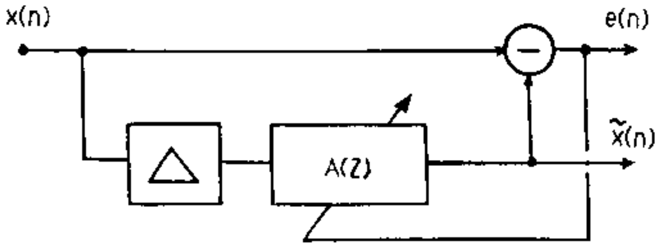


FIG. 11.3 Adaptive line enhancement.

11.4. ADAPTIVE DIFFERENTIAL CODING

Besides signal analysis, linear prediction techniques can be used to condense the representation of signals. The information in a signal is essentially contained in the unpredictable components. Therefore, if the predictable components are attenuated, the amplitude range of the samples is reduced, fewer bits are needed to encode them, and a denser representation is obtained. In practice, for the sake of simplicity and ease of manipulation, it is generally desirable that the original signal be retrievable from the prediction error sequence only. Therefore, in an adaptive approach the filter has to be implemented in a loop configuration as shown in Figure 11.4, in order to take into account the effects of output sequence quantization.

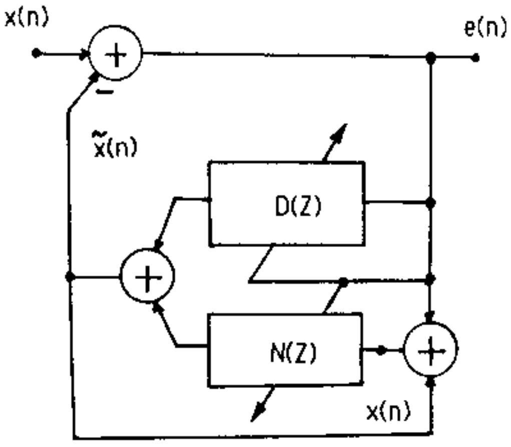


FIG. 11.4 Adaptive differential encoding to condense information.

The prediction error filter is of the FIR/IIR type.The coefficient update section can use LMS algorithms as in Section 4.15, or the FLS algorithms as pointed out in Section 7.12 on pole-zero modeling. Typical uses of the above scheme are for signal storage or efficient transmission [6]. For example, in communications the technique is known as ADPCM (adaptive differential pulse code modulation), and it permits a telephone conversation to be transmitted with unnoticeable degradations through a digital link with 32 kbit/s capacity [7].

The CCITT recommendation G.721 uses for prediction the diagram of Figure 11.4. The transversal part, shown as $D(Z)$ in Figure 11.4, has six coefficients, and its output is defined by

$$s_{ez}(k) = \sum_{i=1}^{6} b_i(k-1)d_q(k-i) \tag{11.9}$$

where $d_q(k)$ is the signal resulting from the quantization of the error signal $e(n)$.

The autoregressive section, shown as $N(Z)$ in Figure 11.4, has two coefficients only, for easy stability control, and the signal estimate $s_e(k)$ is

$$s_e(k) = s_{ez}(k) + \sum_{i=1}^{2} a_i(k-1)s_r(k-i) \tag{11.10}$$

The reconstructed signal $s_r(k)$ is defined by

$$s_r(k) = s_e(k) + d_q(k)$$

Both sets of predictor coefficients are updated using a simplified gradient algorithm. For the second-order predictor,

$$\begin{aligned} a_1(k) &= (1-2^{-8})a_1(k-1) + 3\cdot 2^{-8}\text{sign}[p(k)p(k-1)] \\ a_2(k) &= (1-2^{-7})a_2(k-1) + 2^{-7}\text{sign}[p(k)p(k-2)] \\ &\quad - f[a_1(k-1)]\text{sign}[p(k)p(k-1)] \end{aligned} \tag{11.11}$$

with

$$p(k) = d_q(k) + s_{ez}(k) \tag{11.12}$$

and

$$\begin{aligned} f(a_1) &= 4a_1 \text{ if } |a_1| \leqslant 2^{-1} \\ f(a_1) &= 2\ \text{sign}(a_1) \text{ if } |a_1| > 2^{-1} \end{aligned} \tag{11.13}$$

The reason for introducing the variable $p(k)$ is to make the adaptation more robust to transmission errors by subtracting from the reconstructed signal the portion coming out of the autoregressive section. As concerns the nonlinearity, i.e., the function f, it has been shown to prevent blocking in some circumstances with tone signals.

The stability is guaranteed by the constraints

$$|a_2(k)| \leqslant 0.75, \qquad a_1(k) \leqslant 1 - 2^{-4} - a_2(k) \tag{11.14}$$

Note that, with the use of sign algorithms, the coefficients are implicitly limited to ± 2. Additionally, the encoder includes a single tone detector defined by

$$\begin{aligned} t_d(k) &= 1 \quad \text{if } a_2(k) < -0.71875 \\ t_d(k) &= 0 \quad \text{otherwise} \end{aligned} \tag{11.15}$$

and used in the quantization step adaptation procedure, itself based on adaptive techniques.

11.5. ADAPTIVE DECONVOLUTION

Deconvolution is applied to experimental data in order to remove distortions caused by a measurement system to a desired inaccessible signal. Let us assume that the experimental sequence $y(p)(1 \leqslant p \leqslant n)$ is generated by filtering the desired signal $x(p)$ as follows:

$$y(p) = \sum_{i=0}^{N-1} h_i x(p-i) \tag{11.16}$$

The operation is described in matrix notation by

$$\begin{bmatrix} y(n) \\ y(n-1) \\ \vdots \\ y(1) \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & h_2 & \cdots & 0 & 0 \\ 0 & h_0 & h_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{N-2} & h_{N-1} \end{bmatrix} \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(2-N) \end{bmatrix} \tag{11.17}$$

or

$$y(n) = H^t X(n) \tag{11.18}$$

According to the results in Section 3.5, an LS solution is obtained by

$$X(n) = H(H^t H)^{-1} Y(n) \tag{11.19}$$

The desired sequence $x(n)$ can be retrieved in an adaptive manner through the technique depicted in Figure 11.5.

The estimated data $\tilde{x}(n)$ are fed to the distorting FIR filter, whose coefficients are assumed to be known, and the output $\hat{y}(n)$ is subtracted from the

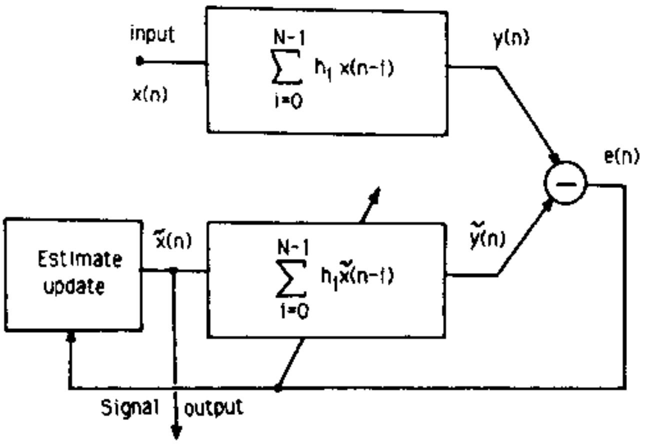


FIG. 11.5 Signal restoration by adaptive deconvolution.

experimental data to produce an error $e(n)$ used to update the estimate at time $n+1$.

The simplest approach one can think of consists of calculating $\tilde{x}(n+1)$ by

$$\tilde{x}(n+1) = \frac{1}{h_0}\left[y(n+1) - \sum_{i=1}^{N-1} h_i \tilde{x}(n+1-i) \right] \tag{11.20}$$

However, it is unrealistic, due to initial conditions and the presence of noise added in the mesurement process.

The gradient method corresponds to the updating equation

$$\begin{bmatrix} x_1(n+1) \\ x_2(n+1) \\ \vdots \\ x_N(n+1) \end{bmatrix} = \begin{bmatrix} 0 \\ x_1(n) \\ \vdots \\ x_{N-1}(n) \end{bmatrix} + \delta \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-1} \end{bmatrix} \left[y(n+1) - \sum_{i=1}^{N-1} h_i x_i(n) \right] \tag{11.21}$$

where δ is the adaptation step, the $x_i(n)(1 \leqslant i \leqslant N)$ are state variables, and the restored signal at the system output is

$$\tilde{x}(n+1-N) = x_N(n) \tag{11.22}$$

The technique can be refined by using a more sophisticated adaptation gain. If matrix manipulations can be afforded, LS techniques based on equation (11.19) can be worked out, associated with recursive procedures to efficiently perform the computations [8, 9].

11.6. ADAPTIVE PROCESSING IN RADAR

Signal processing techniques are employed in radar for target detection by whitening filters, separation of targets by high-resolution spectral analysis methods, and target or terrain recognition by comparison with models or by inverse filtering [10].

The target detection method is depicted in Figure 11.6.

When the signal $s(t)$ is emitted, the signal received can be expressed by

$$y(t) = Gs(t - t_0) + P(t) \tag{11.23}$$

where G is a complex parameter representing the propagation conditions, t_0 is the delay of the signal reflected on the target, and $P(t)$ is a perturbation representing the multiple undesired reflections on various obstacles, or clutter. The useful signal $s(t - t_0)$ can be much smaller than the perturbation, which can be modeled by a colored and evolving noise or, in other words, by an AR signal with variable parameters.

The perturbation can be significantly attenuated by an adaptive prediction error filter, which performs a whitening operation and delivers the signal $y_w(t)$. The signal $s(t)$ is fed to a filter with the same coefficients, and the output is $s_w(t)$. Now the detection process consists of comparing to a threshold the quantity

$$c(\tau) = \frac{|\int y_w(t)\bar{s}_w(t-\tau)dt|^2}{\int |s_w(t)|^2 dt} \tag{11.24}$$

The corresponding operations are the correlation, squared modulus calculations, normalization, and decision.

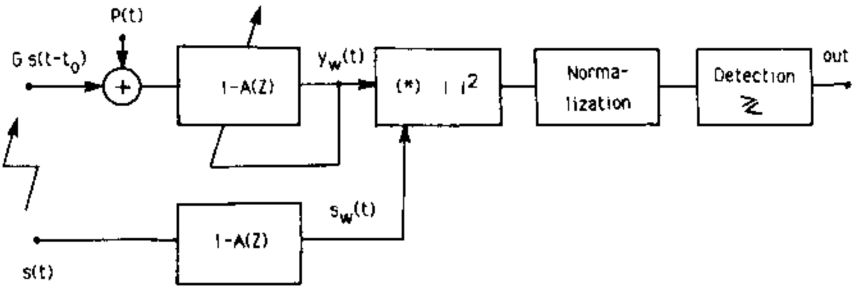


FIG. 11.6 Target detection in the presence of colored evolving noise.

The order N of the adaptive whitening filter is typically close to 10, and the sampling frequency is several megahertz.

The applications reviewed so far deal with one-dimensional signals. In antenna arrays and image processing, M-D signals are involved.

11.7. ADAPTIVE ANTENNAS

The outputs from the elements of an antenna array can be combined to produce a far field beam pattern which optimizes the reception of a desired signal [11]. The beam can be directed towards the bearing of the signal, and it can be configured to have sidelobes which attenuate jamming signals. Moreover, an equalization of the transmission channel can be achieved. In the block diagram in Figure 11.7, the N elements collect delayed versions of the useful signal $x(t)$. For a linear array whose elements are separated by a distance D, the basic delay is

$$\Delta t = \frac{d \sin\theta}{v} \tag{11.25}$$

where θ is the incidence angle and v is the signal velocity. The delays are compensated through N interpolators whose outputs are summed.

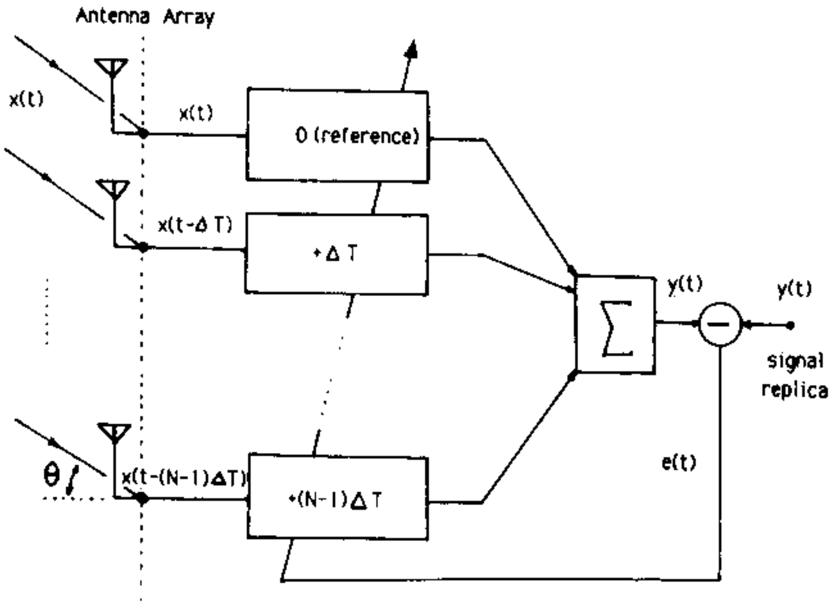


FIG. 11.7 Adaptive antenna array.

However, the filters connected to the antenna elements can do more than just compensate the delays. If a replica of the signal is available, as in digital transmission, and used as a reference, an error can be derived and the system can become an M-D adaptive filter employing the algorithms presented in Chapter 7.

An interesting simplification occurs if the antenna elements are narrowband, because the received signal can be considered as a sine wave. The equation of a propagating wave is

$$s(x, t) = S \exp 2\pi\left(ft - \frac{x}{\lambda}\right) \tag{11.26}$$

where $\lambda = v/f$ is the wavelength associated to the frequency f. From equation (11.26) it appears that adequate beamforming can take place only if

$$\frac{D \sin\theta}{\lambda} < \frac{1}{2} \qquad \text{or} \qquad D < \frac{\lambda}{2} \tag{11.27}$$

Therefore $\lambda/2$ is an upper bound for the spatial sampling interval. The filtering paths reduce to multiplications by weights $w_i = \exp(j\omega\Delta T i)$ with $0 \leqslant i \leqslant N-1$ and $\Delta T = (D\sin\theta)/v$. The coefficients w_i can be calculated to minimize the cross-correlation between the output $y(n)$ and the inputs in the absence of the desired signal. The corresponding equation is

$$R_N \begin{bmatrix} 1 \\ -W_{N-1} \end{bmatrix} = \begin{bmatrix} E \\ 0 \end{bmatrix} \tag{11.28}$$

where R_N is the input covariance matrix, W_{N-1} and $(N-1)$-element coefficient vector, and E the output error power. The coefficients can be found and updated through gradient techniques.

Another approach consists of maximizing the output SNR, which leads to the N-coefficient vector

$$W_N = \frac{1}{\bar{F}^t R^{-1} F} R^{-1} F \tag{11.29}$$

with

$$\bar{F} = [1, e^{j\omega\Delta T}, \ldots, e^{(N-1)j\omega\Delta T}]^t$$

The similarity with linear prediction of time sequences is worth pointing out.

11.8. IMAGE SIGNAL PREDICTION

Linear models are useful in image processing for region classification and segmentation and also for the detection of small regions which differ from their surroundings [12].

A picture element (pixel) of a 2-D image can be represented by a white-noise-driven linear model as

$$x(n,m) = \sum_{l,k}\sum_{\in M} a(l,k)x(n-l, m-k) + e(n,m) \tag{11.30}$$

where $e(n,m)$ is a white noise with power σ_e^2 and M represents the support region for the filter coefficients, called the mask. The mask M can take on several forms. In prediction, it is associated with the past of the point (n,m). The past of a point (n_0, m_0) is related to causality and defined as the set of points

$$\{(n,m) | n = n_0, m < m_0; n < n_0, -\infty < m < \infty\}$$

The model equation (11.30) can also be considered as a prediction operation in which the signal $x(n,m)$ is predicted by the summation $\tilde{x}(n,m)$ and $e(n,m)$ is the prediction error. The prediction coefficients $a(l,k)$ can be calculated from the 2-D normal equations

$$r(l,k) - \sum\sum_{p,q\in M} a(p,q)r(l-p, k-q) = \sigma_e^2\delta(l,k) \tag{11.31}$$

where $r(l,k)$ is the correlation function

$$r(l,k) = E[x(n,m)x(n-l, m-k)] \tag{11.32}$$

The image predictor can be made adaptive by using either gradient or LS algorithms.

A third dimension can be introduced with television signals, considering the sequence of frames.

A pixel is predicted from its past in the same frame and from elements of the previous frames. Applications are for reduced rate encoding, for analysis such as edge detection, and for noise reduction. The complexity issue is crucial in that case, since the sampling rate is 13.5 MHz. Filters with only a small number of coefficients can actually be implemented in real-time hardware.

11.9. ARTIFICIAL INTELLIGENCE AND NEURAL NETWORKS

Artificial intelligence (AI) techniques attempt to reproduce and automatize simple learning and reasoning processes in order to give machines the ability to reason at a level approaching human performance in limited domains. Another of their goals is to extend and structure human-machine interaction [14].

Successful achievements in that field are the so-called knowledge-based or expert systems. A description of such a system is given in Figure 11.8. It is essentially a structured software technique of performing logical interference on symbolic data.

The expert system efficiency rests on the quality of its inference rules, exploited by the inference engine, and of the information stored in the data base.

In some areas, signal processing is involved in the process of constituting the data bases on which AI works. For example, it can be needed to convert the real data, which carry the relevant information into symbolic data. Adaptive techniques can be useful in that process because of the improvements in accuracy and speed achieved in analyzing signals and extracting the parameters or features to condense the information.

In automation, it is increasingly important to equip machines or robots with the capability to communicate with their environment in real time, through acoustic and visual signals, like humans. To that purpose, signal generation and recognition are fundamental operations.

Recognition can be defined as the automatic assignment of a signal to a predetermined category in order to stimulate a predetermined subsequent action. Clearly, adaptive processing methods, in one-dimensional and M-D forms are instrumental in accurately and efficiently performing that task.

An approach which is particularly appealing for recognition and classification is that of neural networks. The basic idea is to model the brain and its neurons, connected by synapses, by nonlinear elements linked by connections which include weighting factors. Accordingly, the output $\tilde{y}(n)$ of a nonlinear element is a function of the vector of the N input values x_i given by

$$\tilde{y} = f\left(\sum_{i=0}^{N} w_i x_i\right); \; x_0 = 1 \tag{11.33}$$

where w_i $(0 \leqslant i \leqslant N)$ are the weighting factors and $f(x)$ the nonlinear function. This function, which allows the decision to be made, can be the step associated with a comparator or a more regular or smoother function, like the sigmoid associated with a saturable amplifier. The system defined by equation (11.33) allows the classification in two categories of the input vectors, depending on whether they produce a positive or negative output, the boundary being given by the equation

$$w_0 + \sum_{i=1}^{N} w_i x_i = 0 \tag{11.34}$$

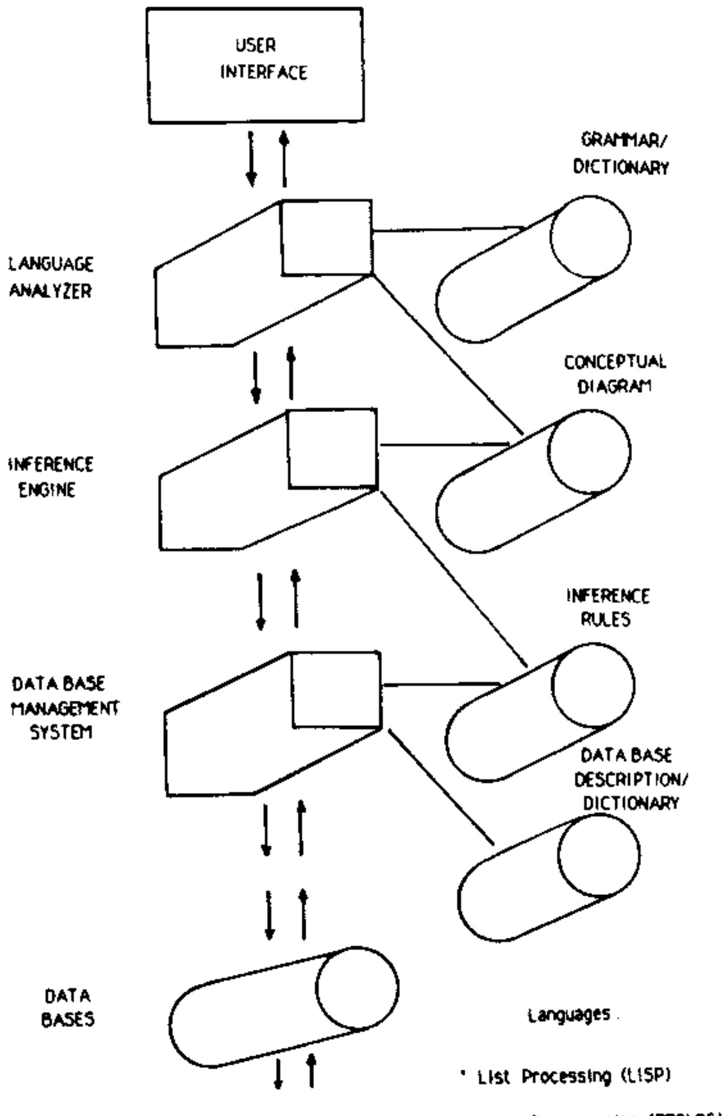


FIG. 11.8 A knowledge-based system.

For $N = 2$, this is the equation of a line. The system is called a perceptron [11]. Its coefficients can be determined through a learning procedure exploiting a set of N_0 input vectors $X(n)$, for which the output $d(n)$ is known. The gradient algorithm can be used in that operation

$$W(n+1) = W(n) + X(n)[d(n) - \tilde{y}(n)], \qquad 1 \leqslant n \leqslant N_0 \tag{11.35}$$

The separation of a set of elements of types A and B by a 3-coefficient perceptron is illustrated in Figure 11.9.

The system can be made more sophisticated through the combination of elements, like logic circuits. A network with 3 levels, or layers, is shown in Figure 11.10. Basically, a 2-layer network can delimit convex domains, associated with the outputs of the lower layer being zero. The 3-layer network allows the combination of convex domains and can therefore reach domains of any shape: it is a general classifier.

The learning process in a 3-layer perceptron can still exploit gradient techniques, under the condition that the nonlinearity is a derivable function, like the sigmoid for example. The cost function associated with an M-output system is

$$J = \sum_{i=1}^{M} (d_i - \tilde{y}_i)^2 \tag{11.36}$$

The iterative determination of the coefficients is based on the derivatives of the output terms $\tilde{y}_i$ with respect to these coefficients. In the computation, it is very simple to verify that the output errors propagate backwards in the same linearized network. The procedure is thus known as the method of backpropagation. The corresponding equations and sequence of operations form the backpropagation algorithm shown in Figure 11.11. It is also possible to use perceptrons with a feedback loop from output to input, like IIR filtering.

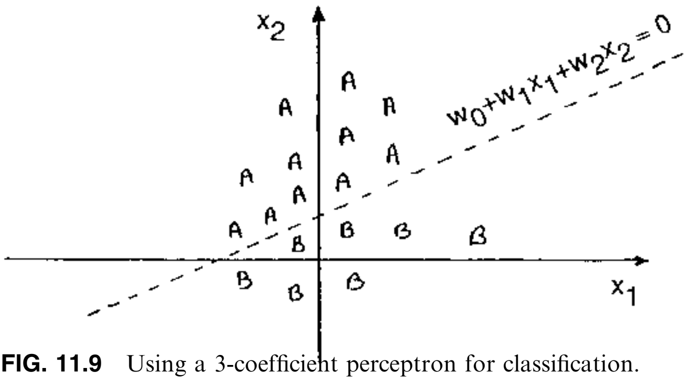


FIG. 11.9 Using a 3-coefficient perceptron for classification.

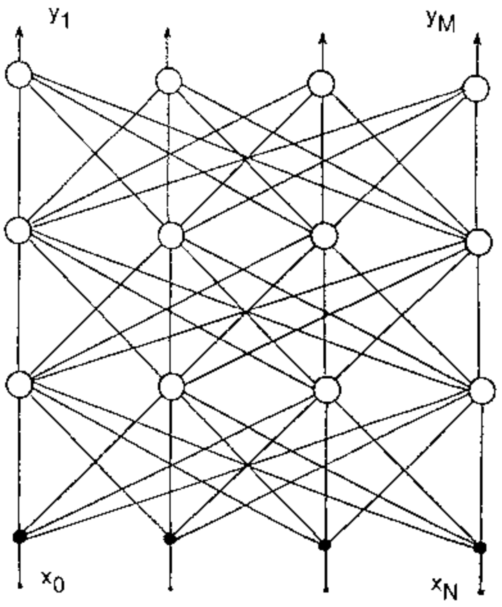


FIG. 11.10 Three-layer perception for general classification.

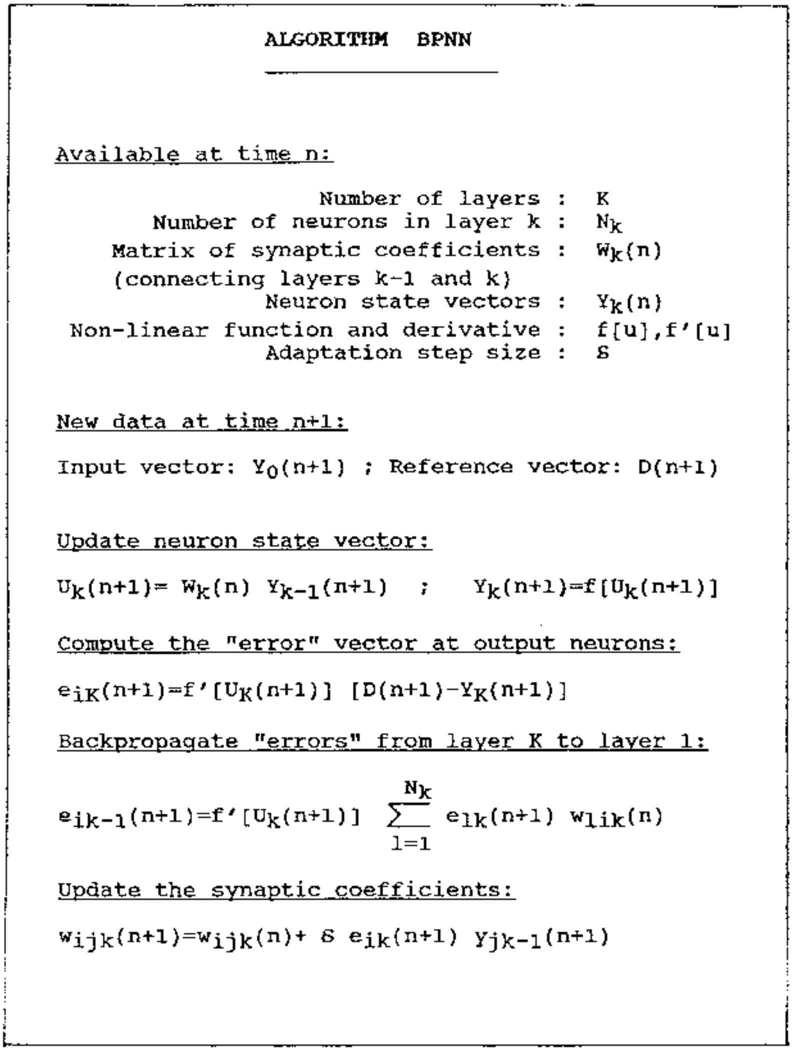
The case of binary inputs leads to the so-called Hopfield network, which consists of a set of N elements, whose outputs are ± 1, interconnected as shown in Figure 11.12. The weighting coefficients are computed to minimize, on a set of N_0 reference vectors, the quadratic cost function

$$J = \sum_{j=1}^{N} \sum_{n=1}^{N_0} [x_j(n) - \sum_{\substack{i=1 \\ i \neq j}}^{N} w_{ij} x_i(n)]^2 \tag{11.37}$$

Under the hypothesis of approximately orthogonal reference data vectors, one gets for the coefficients

$$\begin{aligned} W_{ij} &= \sum_{n=1}^{N_0} x_j(n) x_i(n) \quad (i \neq j) \\ W_{ii} &= 0 \quad (i = 1, \ldots, N) \end{aligned} \tag{11.38}$$

The technique is reminiscent of linear prediction. The cost function has a set of N local minima and, in order to determine the class to which an input



ALGORITHM BPNN

Available at time n:

Number of layers :	K
Number of neurons in layer k :	N_k
Matrix of synaptic coefficients : (connecting layers k-1 and k)	$W_k(n)$
Neuron state vectors :	$Y_k(n)$
Non-linear function and derivative :	$f[u], f'[u]$
Adaptation step size :	δ

New data at time n+1:

Input vector: $Y_0(n+1)$; Reference vector: $D(n+1)$

Update neuron state vector:

$U_k(n+1)= W_k(n)\ Y_{k-1}(n+1)$; $Y_k(n+1)=f[U_k(n+1)]$

Compute the "error" vector at output neurons:

$e_{iK}(n+1)=f'[U_K(n+1)]\ [D(n+1)-Y_K(n+1)]$

Backpropagate "errors" from layer K to layer 1:

$$e_{ik-1}(n+1)=f'[U_k(n+1)] \sum_{l=1}^{N_k} e_{lk}(n+1)\ w_{lik}(n)$$

Update the synaptic coefficients:

$w_{ijk}(n+1)=w_{ijk}(n)+ \delta\ e_{ik}(n+1)\ y_{jk-1}(n+1)$

FIG. 11.11 The backpropagation algorithm for multilayer neural networks.

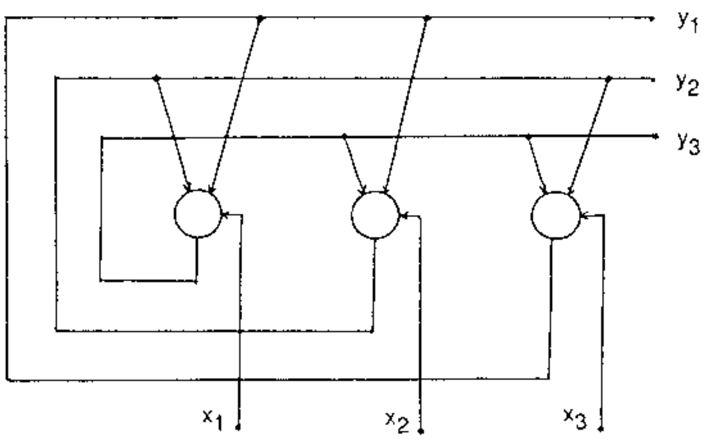


FIG. 11.12 Hopfield network for the classification of binary data.

vector belongs, it is sufficient to feed it to the network and let each output evolve according to the equation

$$y_j(n+1) = f\left[\sum_{i=1}^{N} w_{ij} y_i(n)\right] \tag{11.39}$$

When the system has reached its steady state, the output vector gives the class to which the input vector belongs.

The neural network algorithms can be implemented in software and hardware, as illustrated in Reference [12].

Overall, adaptive methods can contribute to the advances of AI techniques. In return, AI techniques can contribute to the diffusion of adaptive methods. For example, expert systems can be dedicated to adaptive filtering and signal analysis and exploited by practitioners as a valuable help in their efforts to optimize their realizations.

11.10. CONCLUSION

The range of applications which have been briefly introduced illustrate the versatility of adaptive signal processing techniques and the universality of their principles. In practice, it turns out that, for particular cases, variations and adequate refinements are often introduced in the methods to tailor them to the applications and enhance their efficiency. Therefore, these applications may look as many different fields. To a certain extent, it is true, however, there is a common ground, and it is the corresponding common knowledge which is presented in the previous chapters.

The diffusion of adaptive techniques and the extension of their application fields are highly dependent on the advances in technology. Considering the successive generations of large and very large-scale integrated circuits, considering the growing family of integrated signal microprocessor chips, some of which are even specially designed to suit the needs of adaptive filtering, it appears that more and more technological resources are available.

Finally, it can be stated that adaptive methods are bound to have an increasing impact in science and industry.

EXERCISES

1. Let us consider the adaptive line enhancer shown in Figure 11.3. The input signal is $x(n) = \sin(n\pi/4)$ and the delay is $\Delta = 1$. The adaptive prediction filter $A(Z)$ has $N = 4$ coefficients.

   A white noise with power $\sigma_b^2 = 0.1$ is added to the input signal. Compute the four coefficients of the optimal prediction filter, using the results of Section 5.12. What is the magnitude of the sinusoidal component in the output $\tilde{x}(n)$ of the prediction filter?

   Compute the noise power and give the value of the enhancement gain G_e.

   For large N, give the asymptotic value of the enhancement gain.
2. For speech, the following long-term autocorrelation coefficients are considered:

   $$r_0 = 1, r_1 = 0.86, r_2 = 0.56.$$

   Compute the optimal coefficients of a linear predictor with $N = 2$ coefficients. Give the value of the prediction gain G_p.

   Give the block diagram of an ADPCM coder based on such a predictor. Propose a simple scheme to ensure stability in the decoder, in the presence of transmission errors.
3. In a radar system, the useful signal $x(n) = \sin(n\pi/3)$ is sent by the emitter. The receiver captures the following signal: $y(n) = \sin(n\pi/12) + 0.01\sin(n-2)\pi/3)$. A linear predictor with two coefficients is used to attenuate the disturbing clutter component.

   Give the optimal value of the $N = 2$ predictor coefficients. Compute the amplitude of the two sinusoidal components at the output of the linear predictor.

   Assume that a flat noise with power $\sigma_b^2 = 0.01$ is added to the received signal. What is the impact on the sinusoidal components at the output of the linear predictor? Give the value of the signal-to-noise ratio.

4. A perceptron with $N = 3$ coefficients and two inputs uses a step as its nonlinear function. The training sequence $X(n)$ and the known corresponding output $d(n)$ are as follows:

n =	1	2	3	4	5	6	7	8	9	10
$x_1(n)$ =	−0.5	5	2.5	0.5	1.5	−3	2	−1	−1.5	3.5
$x_2(n)$ =	1	2.5	0.5	1	4.5	0.5	2.5	−0.5	2	2
$d(n)$ =	1	−1	−1	−1	1	1	1	−1	1	−1

Compute the optimal values of the $N = 3$ coefficients, using least squares techniques. Now, the perceptron is used as a classifier and $d(n) = 1$ is associated with class A and $d(n) = -1$ is associated with class B. Give a global and direct way to determine coefficient values. Compare the results with those of the least squares approach.

REFERENCES

1. M. David, J. P. Deschamps, and A. Thayse, *Digital Systems with Algorithm Implementation*, Wiley, London, 1983.
2. J. Du, G. Warner, E. Vallow, and T. Hollenbach, "High Performance DSPs," *Signal Processing Magazine* (Special Issue on Industry DSP Technologies) **17**, 16–26 (March 2000).
3. K. K. Parhi, *VLSI Signal Processing Systems*, John Wiley, Inc, New York, N.Y., 1999.
4. A. Nehorai, "A Minimal Parameter Adaptive Notch Filter with Constrained Poles and Zeros," *IEEE Trans.* **ASSP-3**, 983–996 (August 1985).
5. J. R. Zeidler, E. H. Satorius, D. M. Chabries, and H. T. Wexler, "Adaptive Enhancement of Multiple Sinusoids in Uncorrelated Noise," *IEEE Trans.* **ASSP-26**, 240–254 (June 1978).
6. T. Claasen and W. Mecklenbrauker, "Adaptive Techniques for Signal Processing in Communications," *IEEE Communications Mag.* **23**, 8–19 (November 1985).
7. CCITT Rec. G.721, "32 kbit/s Adaptive Differential Pulse Code Modulation," Blue Book, Fascicle 3.4, Melbourne 1988, pp. 231–268.
8. D. Commenges, "The Deconvolution Problem: Fast Algorithms Including the Preconditioned Conjugate-Gradient to Compute MAP Estimator," *IEEE Trans.* **AC-29**, 229–243 (1984).
9. G. Demoment and R. Reynaud, "Fast Minimum Deconvolution," *IEEE Trans.* **ASSP-33**, 1324–1326 (October 1985).
10. B. L. Lewis, F. K. Kretschmer, and W. Shelton, *Aspects of Radar Signal Processing*, Artech House, London, 1986.
11. J. A. Anderson and E. Rosenfeld, Eds., *Neuro-Computing: Foundations of Research*, MIT Press, Cambridge, MA, 1988.
12. M. Duranton and J. A. Sirat, "Learing on VLSI: A General Purpose Digital Neuro-Chip," *Philips Journal of Research* **45**, 1–17 (1990).

12
Adaptive Techniques in Communications

Communications are a major field of appliction for adaptive techniques, and the corresponding algorithms can be found everywhere in networks, terminals, line equipment, switching machines, and human interfaces. All sorts of conditions, constraints, and challenges are encountered. The applications selected below illustrate this diversity, and focus on the flexibility of the algorithms and their ability to cope with many different environments.

12.1. AUTOMATIC GAIN CONTROL

Automatic gain control (AGC) is a very common function in digital receivers. For example, an AGC is generally associated with an analog-to-digital (A/D) converter, to optimize the dynamic range. In that case, the gain of the amplifier placed in front of the A/D converter is controlled by digital means. In some cases, also, the amplifier is digital as well [1].

The objective is to make the signal power constant. If that constant is unity, the gain G is just the inverse of the square root of the power of the input signal $x(n)$. An implementation of a digital AGC is shown in Figure 12.1.

Estimation of the input signal power is performed by a narrow-band low-pass filter $H(Z)$, whose parameter ε defines the estimation time constant τ by the relation

$$\tau = \frac{1}{\varepsilon} \tag{12.1}$$

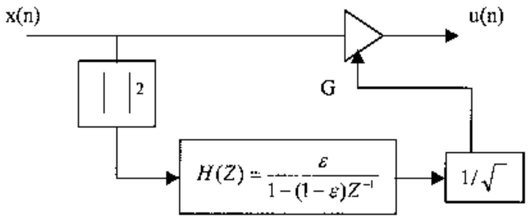


FIG. 12.1 Digital automatic gain control.

according to classical results in digital filtering. The input to the filter is the signal $s(n) = x^2(n)$, whose variance is expressed by

$$\mathrm{var}\{s(n)\} = E[s^2(n) - (E[s(n)])^2] \tag{12.2}$$

Assuming the input samples to be uncorrelated, the estimate $P_x(n)$ of the input signal power available at the filter output has the variance

$$\mathrm{var}\{P_x(n) = [E[x^4(n)] - (E[x^2(n)])^2]\frac{\varepsilon}{2} \tag{12.3}$$

For example, if $x(n)$ is Gaussian with power σ_x^2, one gets

$$\mathrm{var}\{P_x(n)\} = 2\sigma_x^4\frac{\varepsilon}{2} = \varepsilon\sigma_x^4 \tag{12.4}$$

Now, it must be pointed out that the power estimation induces a multiplicative noise at the output of the AGC. The estimate $P_x(n)$ can be considered as the sum of the true value and a noise $b(n)$

$$P_x(n) = E[x^2(n)] + b(n) \tag{12.5}$$

Then, if the parameter ε is sufficiently small,

$$G = \frac{1}{\sqrt{E[x^2(n)] + b(n)}} \approx \frac{1}{\sqrt{E[x^2(n)]}}\left(1 - \frac{b(n)}{2E[x^2(n)]}\right) \tag{12.6}$$

and the AGC output is

$$u(n) = \frac{x(n)}{\sqrt{E[x^2(n)]}} - \frac{x(n)b(n)}{2(E[x^2(n)])^{3/2}} \tag{12.7}$$

In the Gaussian case, the power of the multiplicative noise component in the output $u(n)$ is

$$\frac{1}{4\sigma_x^6}E[x^2(n)b^2(n)] = \frac{\varepsilon}{4} \tag{12.8}$$

using (12.4) and assuming $x(n)$ and $b(n)$ to be uncorrelated. This equation clearly shows the importance of the integrator parameter ε The square-root calculation in Figure 12.1 can be avoided if a magnitude estimation is performed and $|x(n)|$ is fed to the integrator. However, a bias is introduced in the estimation and, for a Gaussian input signal, it is given by

$$E[|x(n)|] = \sqrt{\frac{2}{\pi}}\sigma_x \approx 0.8\sigma_x \tag{12.9}$$

The approach can be extended to complex input signals in a straightforward manner.

12.2. ECHO CANCELLATION

In transmission networks, echoes occur when a delayed and attenuated version of the signal sent by the local emitter to the distant receiver reaches the local receiver. These echo signals have their origin in the hybrid transformers which perform the two/four-wire conversion, in the impedance mismatches along the two-wire lines, and, in some specific cases like hands-free telephony, in acoustic couplings between loudspeakers and microphones in the subscriber sets.

Echo cancellation or (more accurately) echo control, consists in modeling these unwanted couplings between local emitters and receivers and subtracting a synthetic echo from the real echo. Actually, it is a straight application of adaptive filtering concepts and algorithms. However, the problem may become extremely complex and challenging, depending on the environment, the operational constraints, and the user requirements.

Two different situations can be distinguished, according to the nature of the signals involved, typically data or voice. The case of data modems is dealt with first, since it is easier to handle [2].

12.2.1. Data Echo Canceller

The most efficient exploitation of two-wire lines is achieved when data signals are transmitted simultaneously in the two directions and in the same frequency bands. This is called bidirectional or full-duplex transmission, in contrast to half-duplex transmission, when only one direction is used at a time. The principle is illustrated in Figure 12.2

The signal $x_A(n)$ is sent from terminal A to terminal B through a two-wire line. The signal $y(n)$ at the input of the receiver of terminal A consists of two components, namely the signal $y_B(n)$ coming from berminal B, which is the useful data signal, and the returned unwanted echo generated from $x_A(n)$

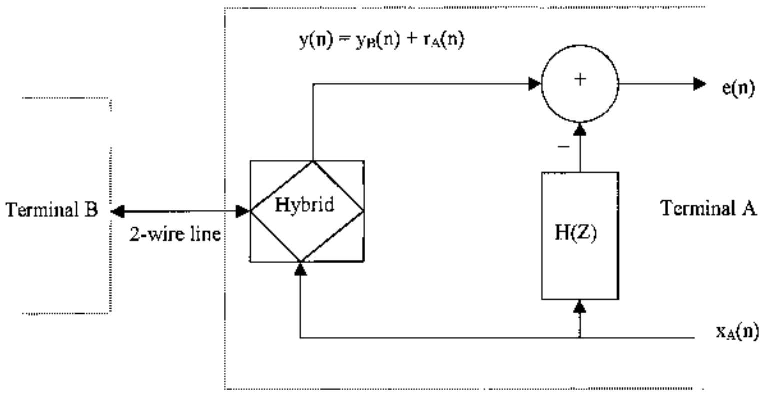


FIG. 12.2 Echo cancellation for full-duplex data transmission.

and denoted $r_A(n)$. The task of the filter $H(Z)$ is to generate a synthetic echo $\tilde{y}(n)$ as close as possible to $r_{\mathrm{A}}(n)$, so that, after subtraction, the output error $e(n)$ is kept sufficiently close to $y_{\mathrm{B}}(n)$ to make the transmission of data from terminal B to terminal A satisfactory.

The selection of the parameters of the adaptive filter is guided by the context. The number N of the coefficients is derived from the duration of the echo impulse response that has to be compensated, taking into account the sampling frequency. For example, let us consider a subscriber line of length $D = 3$ km, an electric signal velocity over this subscriber line $v = 2 \times 10^8$ m/s and a sampling frequency $f_s = 2$ MHz. If a single reflection on the distant hybrid is taken into account, the number of coefficients of the echo canceller is calculated by

$$N = \frac{2D}{v} f_S \tag{12.10}$$

which amounts to $N = 60$ with the figures given above.

It is necessary to make the filter adaptive, because the characteristics of the transmission line may change with time. It has been emphasized in Chapter 4 that the nature of the input signal is critical to the performance of any adaptive filter. Here, according to Figure 12.2, the input to the filter is the sent data signal $x_{\mathrm{A}}(n)$, which generally is uncorrelated, has unit power, and therefore has the autocorrelation matrix $R_N = I_N$. This is the most favorable situation for adaptive filtering, since LMS algorithms perform as well as RLS algorithms; the adaptation step size δ is bounded by $2/N$ and the adaptation time constant $\tau = 1/\delta$, as shown in Chapters 4 and 6.

In the learning phase, according to Section (4.4), the average output error power is

$$E_r(n) = \|H_{\text{opt}}\|_2^2(1-\delta)^{2n} \tag{12.11}$$

The L_2 norm of the echo coefficient vector, $\|H_{\text{opt}}\|_2^2$, represents the power of the echo signal.

Now, in full-duplex transmission, the useful signal $y_{\text{B}}(n)$ in the reference is smaller, and even generally much smaller, than the echo signal $r_{\text{A}}(n)$. If A_{s} designates the ratio of the echo to the useful signal, SNR and the desired signal to noise ratio at receiver input, then the echo attenuation A_{e} must satisfy the inequality

$$A_{\text{e}} > A_{\text{s}} + \text{SNR} \quad \text{(dB)} \tag{12.12}$$

For example, assume SNR $=$ 40 dB and $A_s = 20$ dB; then $A_e = 60$ dB, which obviously implies a high level of performance for the echo canceller, in terms of residual error in the steady state after convergence.

In the adaptation process, the useful signal creates a misadjustment of the coefficients resulting in an excess output error. Denoting by σ_y^2 the power of the useful received data signal and using equation (4.32), the variance of each filter coefficient after convergence is $\sigma_y^2\delta/2$, and the excess mean square error is N times greater, namely $N\sigma_y^2\delta/2$. In order to reach the target SNR, the step size δ must satisfy the inequality

$$N\frac{\delta}{2} < \frac{1}{\text{SNR}} \tag{12.13}$$

In the above derivation, it is assumed that the output error power is very close to the power of the useful signal σ_y^2. For example, if SNR $= 10^4$ (40 dB) and $N = 60$, one gets $\delta < 3.3 \times 10^{-6}$. This is a very small value, leading to a very long learning phase. Therefore, whenever practical, the echo canceller is trained in the half-duplex mode with a large step size. Then, once a specified level of convergence has been observed, the step size is switched to the appropriate small value.

The impact on the coefficient wordlength is worth emhasizing. The analysis given in Section 4.5 can be carried out in the present context, and equation (4.61) simplifies to

$$b_c \approx \log_2\frac{1}{\delta} + \frac{1}{2}\log_2 A_e \tag{12.14}$$

With $\delta = 3.3 \times 10^{-6}$ and $A_e = 10^6$, the estimated number of bits of the coefficients is $b_c \approx 29$. As pointed out in Section 4,5, it is not necessary to implement the multiplications with such accuracy in the filter. The full accuracy is just needed for the coeffiient updating.

12.2.2. Voice Echo Canceller

Voice Echo Cancellation, or control, turns out to be a somewhat different problem, due to the specificities of the speech signal. First, in order to cope with the high level of nonstationarity of speech, the algorithms have to be properly designed. Second, due to the signal bandwidth and the velocity of acoustic waves in the open air, very long filters are necessary in some applications. For example, using equation (12.10) with $\nu = 330$ m/s, $f_s = 8$ kHz, and $2D = 100$ m yields $N = 2400$. This kind of filter length is encountered in audioconferencing applications, for example. Some specific techniques, like coefficient interpolation, can be employed to limit the complexity of the device. Third, in order to reach a high level of performance and meet the expectations of the users, the voice echo canceller may have to include several other functions, like speech detection and denoising. Overall, a very sophisticated device may result in the end [3].

A typical algorithm for adaptive acoustic echo cancellation is the so-called affine projection (AP) algorithm, which in an extension of the normalized LMS algorithm (NLMS) [4]. Let us consider a correlated input signal $x(n)$. For first-order correlation, the vector $X(n+1)$ of the most recent input samples at time $n+1$ can be expressed as a linear combination of two orthogonal vectors by

$$X(n+1) = \alpha X(n) + Z(n+1) \tag{12.15}$$

where α is a scalar and $Z(n+1)$ is orthogonal to $X(n)$, i.e.,

$$E[X(n)^t Z(n+1)] = 0 \tag{12.16}$$

Now, in the NLMS algorithm, the coefficient vector is updated by

$$H(n+1) = H(n) + \delta \frac{X(n+1)e(n+1)}{X^t(n+1)X(n+1)} \tag{12.17}$$

as discussed in Section (4.8). With that algorithm and equation (12.15), the updating vector at time $n+1$ is related to the updating vector at time n. Some redundancy can be removed if the projection

$$\alpha(n+1) = \frac{X^t(n+1)X(n)}{X^t(n)X(n)} \tag{12.18}$$

is calculated and used to derive $Z(n+1)$. In fact, $\alpha(n+1)$ is just the first-order prediction coefficient for the vector $X(n)$, since it minimizes the cost function

$$J = \|X(n+1) - \alpha(n+1)X(n)\|^2 \tag{12.19}$$

The AP algorithm consists of the equations

$$H(n+1) = H(n) + \delta \frac{e(n+1)Z(n+1)}{Z^t(n+1)X(n+1)} \tag{12.20}$$

and

$$Z(n+1) = X(n+1) - \alpha(n+1)X(n) \tag{12.21}$$

completed by (12.18)

The updating of the coefficients is performed in a direction orthogonal to $X(n)$. The output error $e(n+1)$ is given, as before, by

$$e(n+1) = y(n+1) - H^t(n)X(n+1) \tag{12.22}$$

It is readily verified that the a posteriori error $\varepsilon(n+1)$ is zero if $\delta = 1$. As concerns complexity, the number of multiplications is twice that of the NLMS. The convergence analysis is similar to that of the NLMS. The procedure can be extended beyond first-order prediction, leading to higher-order AP algorithms. With speech, experimental results show that the AP approach accelerates the algorithms, with respect to NLMS algorithms. As concerns the echo attenuation, realistic objectives can be set in the range 30–50 dB.

During conversation, it may happen that both users talk at the same time, and simultaneous bidirectional transmission takes place: the so-called double-talk situation. It is particularly disturbing because it produces misalignment of the coefficients and a drop in echo attenuation. In fact, it is generally recognized that the adaptive filter coefficients have to be frozen during double talk, and the problem arises of double-talk detection [5].

The performance of double-talk detectors is crucial for the comfort of the users. Several techniques, more or less complicated, can be employed, and they can be found in the current literature. A simple and reasonably efficient technique is given in Figure 12.3. The idea is to compare the level of the received signal $r(n)$ with the level of the signal $e(n)$ after subtraction of the synthetic echo. In the absence of a distant talker signal in $r(n)$, the levels will exhibit large differences, assuming that the echo canceller works properly. On the other hand, during double talk, the levels come closer. Based on that information, it is possible to decide on the presence or absence of double talk and freeze or adjust the coefficients. Obviously, the parameters for level detection and decision have to be chosen carefully to avoid false decision and excessive delays. The level detectors can be based on power or amplitude measurements, as in Section 12.1.

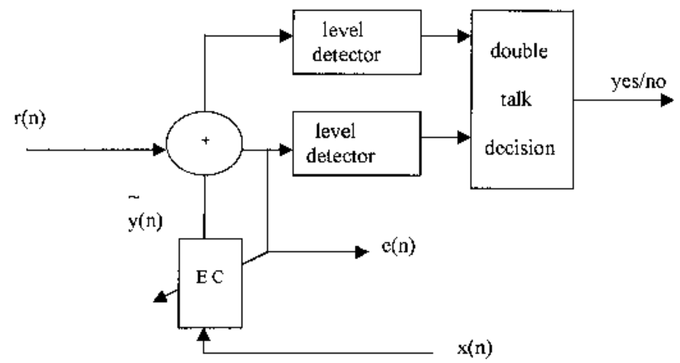


FIG. 12.3 Principle of a simple scheme for double talk detection.

12.3. MODELS FOR CHANNEL EQUALIZATION AND TARGET FILTERS

The transmission of digital data is based on the assumption that the channel is linear-phase and satisfies the Nyquist criterion, namely that its transfer function $C(f)$ satisfies the relation

$$C(f_b + f) + C(f_b - f) = 1 \tag{12.23}$$

where f_b is the symbol rate. This equation is the condition for the impulse response to be zero at all the instants that are nonzero integer multiples of $T_b = 1/f_b$, and consequently the condition for the absence of interference between consecutive symbols. Obviously, practical channels do not meet the criterion, and equalization must take place in the data receiver [6, 7].

In a given situation, the relevant equalizer structure is selected according to the model of the channel, and it is important to review the models associated with different transmission configurations. Ostensibly, in the absence of noise and other disturbing signals, the optimal equalizer is just the inverse of the channel model.

12.3.1. Channel Models

A common case is a channel that introduces amplitude and minimum-phase distortion, with the transfer function $C_1(Z)$ given by

$$C_1(Z) = \frac{A}{1 - \sum_{i=1}^{N} a_i Z^{-1}} \tag{12.24}$$

It is an all-pole model, and all the poles are inside the unit circle, which ensures stability. Typical of such channels are telephone cables. The optimal equalizer $H(Z)$, in the absence of noise, is the inverse of $C_1(Z)$ and is of the FIR type, generally called the transversal equalizer.

In radiocommunications, the signals radiated by the antenna of the emitter can reach the receiver antenna through several propagation paths. Generally, a few paths for microwave links and many paths for mobile radiocommunications. The model is

$$C_2(Z) = \left[1 - \sum_{i=1}^{M} b_k Z^{-1}\right] \tag{12.25}$$

This is an all-zero model, and the inverse is an all-pole equalizer. If all the zeros of the model are inside the unit circle, the function is minimum-phase and the equalizer is generally implemented as a decision-feedback equalizer.

In many circumstances, the channel is not minimum-phase and, for equalization purposes, the model transfer function is represented by:

$$C_3(Z) = A\Phi(Z)\left(1 - \sum_{i=1}^{M} b_i Z^{-1}\right) \tag{12.26}$$

where the first factor $\Phi(Z)$ is a pure phase shifter and the second factor is minimum-phase. In fact, $\Phi(Z)$ is a phase corrector which shrinks the zeros outside the unit circle into the unit circle. Such a phase corrector is not equalizable. However, if a delay can be tolerated, then a suitably specified transversal filter can perform the task. Therefore the channel $C_3(Z)$ can be equalized by the combination of a transversal equalizer and a decision-feedback equalizer.

The presence of noise in the transmission channel complicates the equalization issues. First of all, if a white noise is added at the channel output, like a measurement noise, the filter that maximizes the signal-to-noise ratio at a given time is the so-called *matched* filter.

12.3.2. The Matched Filter

The matched filter is derived from the following objective: find the filter frequency response $H(f)$ such that, at a given time θ, the output signal-to-noise ratio is maximized. The input to the filter is assumed to be $x(t)$ given by

$$x(t) = s(t) + b(t) \tag{12.27}$$

where $s(t)$ is the useful signal with Fourier transform $S(f)$, and $b(t)$ is a white noise with spectral power density β. The useful filter output is

$$y(t) = \int_{-\infty}^{\infty} S(f)H(f)e^{j2\pi ft}dt \tag{12.28}$$

The output noise power is

$$P_B = \beta \int_{-\infty}^{\infty} |H(f)|^2 df \tag{12.29}$$

and the signal-to-noise ratio SNR at time θ is expressed by

$$\mathrm{SNR} = \frac{|y(\theta)|^2}{P_B} = \frac{\left| \int_{-\infty}^{\infty} S(f)H(f)e^{j2\pi f\theta}df \right|^2}{\beta \int_{-\infty}^{\infty} |H(f)|^2 df} \tag{12.30}$$

Now, the Schwarz inequality can be invoked, namely

$$\left| \int_{-\infty}^{\infty} S(f)H(f)e^{j2\pi f\theta}df \right|^2 \leq \int_{-\infty}^{\infty} |S(f)|^2 df \int_{-\infty}^{\infty} |H(f)|^2 df \tag{12.31}$$

Equality is reached for

$$H(f)e^{j2\pi f\theta} = k\overline{S(f)} \tag{12.32}$$

where k is a real constant. Therefore the maximum of SNR is obtained for

$$H(f) = k\overline{S(f)}e^{-j2\pi f\theta} \tag{12.33}$$

and, in terms of impulse responses,

$$h(t) = ks(\theta - t) \tag{12.34}$$

Finally, the impulse response of the optimal filter is proportional to the time-inverted useful signal. The optimal filter is matched to the useful signal.

The meaning in the digital transmission context is that the optimal filter coefficients, in the presence of white noise, are given by the channel-impulse-response elements, in reverse order. Assuming that the data signal fed to the channel is of unit power, the channel-impulse-response vector is $C^t = [c_0, c_1, \ldots, c_{N-1}]$, the noise power is σ_b^2, the delay at the receiver output is $\theta = N - 1$, and the signal-to-noise ratio is

$$\mathrm{SNR} = \frac{\sum_{i=0}^{N-1} c_i^2}{\sigma_b^2} \tag{12.35}$$

In terms of propagation, this result indicates that all the paths bring their contribution to the final SNR. However, noise is not the only obstacle in digital transmission: intersymbol interference has to be minimized.

12.3.3. The Wiener Filter

The problem of data transmission can also be appreciated from a signal reconstruction perspective, using least squares. The situation is depicted in Figure 12.4.

The objective is to minimize the power of the difference between output and input signals, expressed by

$$J = \int_{-\infty}^{\infty} |s(t) - \tilde{s}(t)|^2 dt \tag{12.36}$$

in the presence of the additive colored noise $b(t)$ with power-spectral density $|B(f)|^2$. According to the least squares principle, the optimal filter is obtained when the error $e(t) = s(t) - \tilde{s}(t)$ is uncorrelated with the filter input $x(t)$, which implies

$$r_{ex}(\tau) = \int_{-\infty}^{\infty} [s(t) - \tilde{A}(t)]\, x(t - \tau)\, dt = 0 \tag{12.37}$$

In the spectral domain, using the results in Section 3.1 and the assumption that the noise $b(t)$ is uncorrelated with the input signal $s(t)$, the following filter frequency response is obtained:

$$H(f) = \frac{|S(f)|^2 \overline{C(f)}}{|S(f)|^2 |C(f)|^2 + |B(f)^2} \tag{12.38}$$

with $|S(f)|^2$ the power spectral density of the input signal. The above expression can be rewritten in different forms. First, the inverse of the channel response can be made apparent:

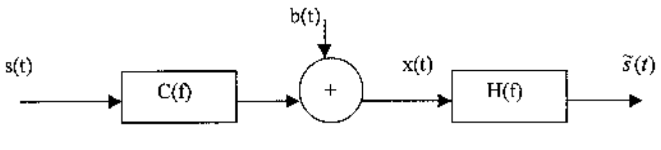


FIG. 12.4 Additive noise and Wiener filtering.

$$H(f) = \frac{1}{C(f)} \frac{|C(f)|^2}{|C(f)|^2 + |B(f)|^2/|S(f)|^2} \tag{12.39}$$

Clearly, in the digital transmission context, in the absence of noise, the optimal equalizer frequency response is the inverse of the channel frequency response. However, in the presence of colored noise, the spectral density of the noise affects the equalizer frequency response. Expression (12.39) can also be related to the matched filter, considering the factorization

$$H(f) = \frac{\overline{C(f)}}{|C(f)|^2 + |B(f)|^2/|S(f)|^2} \tag{12.40}$$

In the presence of white noise with power σ_b^2 and a unit-power uncorrelated input data signal, this equation simplifies to

$$H(f) = \frac{\overline{C(f)}}{|C(f)|^2 + \sigma_b^2} \tag{12.41}$$

A comparison with equation (12.33) shows that the Wiener filter is suboptimal for digital transmission. The difference between the matched and Wiener filters is discussed in a later section.

To conclude this section, an additional important aspect must be introduced. The signals are actually transmitted between emitters and receivers in analog form. A digital-to-analog conversion takes place in the emitter while an analog-to-digital conversion is carried out in the receiver; in between, the signal propagates in analog form. The sampling times in emitter and receiver are not necessarily the same, and fractional delays are introduced. Thus, an interpolating filter is part of the overall model and has to be included in the equalizer function. It can severely degrade the performance of the equalizer, because an accurate interpolation requires a large number of FIR coefficients, and this is why synchronization of the timing instants is generally carried out separately in digital receivers.

12.4. TRANSVERSAL EQUALIZER

The adaptive transversal equalizer can cope with all the situations, but with various degrees in performance. The principle is shown in Figure 12.5. In an actual modem, the reference signal can have two origins. It can be the data signal itself, $d(n)$, adequately delayed to reflect the delays incurred during transmission and equalization. This is typical of the learning phase, with a learning sequence known at the receiver, in a transmission procedure. The received data $\tilde{d}(n)$ can also be used as the reference, and the equalizer is said

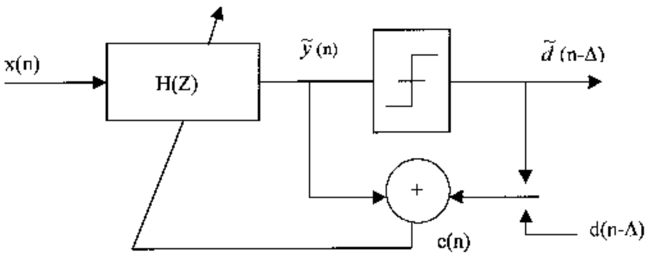


FIG. 12.5 The transversal equalizer.

to be "decision-directed"; it is the tracking phase and comes after the learning or training phase, when real data are transmitted.

With the LMS algorithm, the equations are:

$$\begin{aligned} e(n+1) &= d(n+1-\Delta) - H^t(n)X(n+1) \\ H(n+1) &= H(n) + \delta e(n+1)X(n+1) \end{aligned} \tag{12.42}$$

where Δ is the delay in the total link. The received signal itself is related to the data by

$$x(n) = \sum_{i=0}^{\infty} c_i d(n-i) \tag{12.43}$$

the channel transfer function being expressed by

$$C(Z) = \sum_{i=0}^{\infty} c_i Z^{-1} \tag{12.44}$$

The optimal coefficient vector is

$$H_{\text{opt}} = R_{xx}^{-1}\ E[d(n+1-\Delta)X(n+1)] \tag{12.45}$$

If the data sequence is uncorrelated, then

$$H_{\text{opt}} = R_{xx}^{-1} \begin{bmatrix} c_\Delta \\ c_{\Delta-1} \\ \vdots \\ c_{\Delta+1-N} \end{bmatrix} = R_{xx}^{-1} C_\Delta \tag{12.46}$$

Assuming that the data sequence has unit power, the minimal residual error power after equalization is expressed by

$$E_r = 1 - C_\Delta^t R_{xx}^{-1} C_\Delta \tag{12.47}$$

The importance of the delay is apparent, and the delay Δ has to be included in the search for optimality.

Another crucial issue is the level of white noise included in the input signal $x(n)$. If the noise power is σ_b^2 at the input, the noise power at the decision point is

$$\sigma_{\text{dec}}^2 = H_{\text{opt}}^t H_{\text{opt}} \sigma_b^2 \tag{12.48}$$

Therefore noise is amplified in the transversal equalization process. This can be a serious limitation, since it degrades the performance of the transmission system.

Implementation using the LMS algorithm is simple and robust. Other algorithms like RLS, Lattice, or QR can also be used. Due to the FIR structure, there is no stability issue.

The extension to complex signals is straightforwawrd, as pointed out in Section 7.8. Computing the gradient of the squared norm of the complex output error $e(n)$ leads to the coefficient updating equation

$$H(n+1) = H(n) + \delta e(n+1)\bar{X}(n+1) \tag{12.49}$$

Indeed, the computational complexity increases four times

The length of the learning sequence is an important parameter, because it affects the efficiency of the transmission link. It is determined from a study of convergence in the decision-directed mode. First, the case of two-level binary data, $d(n) = \pm 1$, is considered. Due to the decision device, there are two optimal coefficient vectors, namely $\pm H_{\text{opt}}$, and zero is a saddle point on the error surface. Once convergence has started the speed depends on the bit error rate BER. Assuming that a wrong decision reverses the sign of the output error, the impact of the bit error rate can be seen as a reduction of the adaptation step size from δ to δ(1-2BER). Thus, a short learning sequence suffices with binary data.

With multilevel signals, if d is the maximum distance between neighboring levels or neighboring points in the complex-plane constellation, the output error is smaller than $d/2$. The error surface as a function of the coefficient vector exhibits local minima, whose number is related to the number of levels. In order to reach the global minimum in the decision-directed mode, the coefficient vector must be sufficiently close to the optimum vector when the decision-directed mode starts. Therefore, the learning sequence must be long enough to bring the output error power below $(d/2)^2$.

As an illustration, let us consider real data and L level symbols, with uniform amplitude and probability distributions, and distance $d = 1$. The signal power is $P_s = L^2/12$. If an LMS algorithm with maximum speed ($\delta = 1.N\sigma_x^2$) is used in the learning phase, according to equatin (4.44) in Section 4.4, the length N_s of the learning sequence must be such that

$$\frac{L^2}{12}\left(1-\frac{1}{N}\right)^{2N_s} < \frac{1}{4} \tag{12.50}$$

where N is the number of equalizer coefficients. For example, if $L = 16$ and $N = 32$, the length of the learning sequence is $N_s > 70$. Now, if the adaptation step size is made smaller by a factor k, the length is approximately multiplied by the same factor. Therefore it is essential for the efficiency of the transmission link to get the fastest possible initial convergence. To that purpose, fast least squares algorithms can be used to advantage.

12.5. DECISION FEEDBACK EQUALIZER—DFE

The objective of the DFE is to equalize FIR channels without inverting the transfer function, in order to avoid the need for complicated stability control. The issue is also raised in Section 4.16, and the equation error technique is introduced. In the context of equalization, the interpretation is as follows: the channel impulse response is split into three sections: namely the sample with maximal magnitude; the samples before, called precursors; and the samples after, called postcursors. The DFE compensates the postcursors.

From a signal-processing perspective, the problem is how to deal with the zeros of the channel transfer function $C(Z)$ that are out of the unit circle. Let Z_0 be such a zero, implying

$$|Z_0| > 1 \tag{12.51}$$

The corresponding factor in the equalizer transfer function should be

$$H_0(z) = \frac{1}{1 - Z_0 Z^{-1}} \tag{12.52}$$

which can be developed in series as

$$H_0(z) = \frac{-Z}{Z_0}\frac{1}{1-\dfrac{Z}{Z_0}} = \frac{-Z}{Z_0}\sum_{i=0}^{\infty}\left(\frac{Z}{Z_0}\right)^i \tag{12.53}$$

Due to inequality (12.51), for a suitable integer P, an approximation of H_0 (z) is

$$H_0(z) \approx \frac{-Z}{Z_0}\sum_{i=0}^{P}\left(\frac{Z}{Z_0}\right)^i \tag{12.54}$$

This approximation cannot be implemented. However, if a delay is introduced as

$$Z^{-(P+1)}H_0(Z) = -\frac{1}{Z_0}\left(\frac{1}{Z_0^P} + \frac{Z^{-1}}{Z_0^{P-1}} + \cdots + Z^{-P}\right) \tag{12.55}$$

a realizable transfer function is obtained. Therefore, a maximum-phase zero in the channel can be approximately equalized if a delay is introduced. The greater the delay, the better the approximation.

The structure of the approximation obtained is worth pointing out. The equation (12.55) is the cascade of a gain and a backward linear predictor, and it can be implemented in that way, if blind equalization is considered, for example.

Coming back to the equalizer, its block diagram is shown in Figure 12.6. It consists of a transversal section $A(z)$, also called a feedforward filter, and a recursive section $B(z)$, also called a feedback filter, whose coefficients can be updated with the LMS algorithm. The optimum values of the coefficients A_{opt} and B_{opt} can be computed according to the least squares principle. The output error is expressed by

$$e(n+1) = d(n+1-\Delta) - B^t(n)D(n-\Delta) - A^t(n)X(n+1) \tag{12.56}$$

where Δ is the delay of the system. The least squares solution is readily obtained by

$$\begin{bmatrix} B_{\text{opt}} \\ A_{\text{opt}} \end{bmatrix} = E\left[\begin{bmatrix} D(n-\Delta) \\ X(n+1) \end{bmatrix}[D^t(n-\Delta)X^t(n+1)]\right]^{-1} E\left[d(n+1-\Delta)\begin{bmatrix} D(n-\Delta) \\ X(n+1) \end{bmatrix}\right] \tag{12.57}$$

The input $x(n)$ satisfies equation (12.44) and, assuming uncorrelated unit power data, one gets

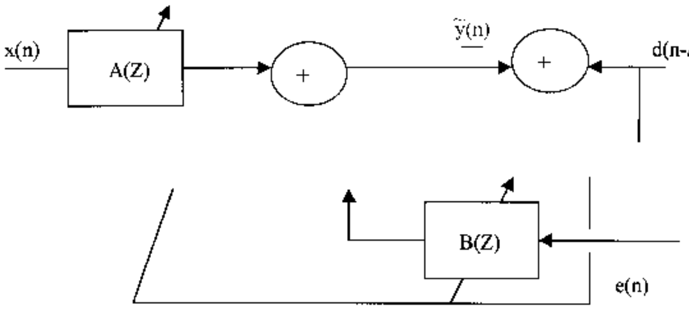


FIG. 12.6 Principle of the adaptive DFE.

$$\begin{bmatrix} B_{\text{opt}} \\ A_{\text{opt}} \end{bmatrix} = \begin{bmatrix} I_{N_b} & M \\ M^t & R_{N_a} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ C_\Delta \end{bmatrix} \tag{12.58}$$

The transversal and recursive sections have N_a and N_b coefficients respectively. R_{N_a} is the $N_a \times N_a$ autocorrelation matrix of the input signal, I_{N_b} is the $N_b \times N_b$ identity matrix. The $N_b \times N_a$ matrix M is defined by:

$$M = E\left[\begin{bmatrix} d(n-\Delta) \\ d(n-1-\Delta) \\ \vdots \\ d(n+1-N_b-\Delta) \end{bmatrix} [x(n+1), x(n), \dots, x(n+2-N_a)]\right] \tag{12.59}$$

Its elements are the coefficients c_i of the channel impulse response. For example, if $N_a = N_b = 3$ and $\Delta = 2$, the matrix M is

$$M = \begin{bmatrix} c_3 & c_2 & c_1 \\ c_4 & c_3 & c_2 \\ c_5 & c_4 & c_3 \end{bmatrix} \tag{12.60}$$

The N_a-element vector C_Δ is

$$C_\Delta^t = [c_\Delta, c_{\Delta-1}, \dots, c_{\Delta+1-N_a}] \tag{12.61}$$

An efficient way to compute the solution of equation (12.58) is to notice the relationships between A_{opt} and B_{opt} vectors. Equation (12.56) can be rewritten as

$$\begin{aligned} B_{\text{opt}} + MA_{\text{opt}} &= 0 \\ M^t B_{\text{opt}} + R_{N_a} A_{\text{opt}} &= C_\Delta \end{aligned} \tag{12.62}$$

And, finally, the optimum values for the DFE coefficients are:

$$\begin{aligned} A_{\text{opt}} &= [R_{N_a} - M^t M]^{-1} C_\Delta \\ B_{\text{opt}} &= -MA_{\text{opt}} \end{aligned} \tag{12.63}$$

Now, the minimum mean square error (MMSE) at the output of the equalizer is, according to Section 4.3,

$$E_{\min} = E[d^2(n-\Delta)] - [B_{\text{opt}}^t, A_{\text{opt}}^t] \begin{bmatrix} 0 \\ C_\Delta \end{bmatrix} \tag{12.64}$$

which yields

$$E_{\min} = E[d^2(n-\Delta)] - A_{\text{opt}}^t C_\Delta \tag{12.65}$$

It is worth pointing out that only the transversal section is involved. The recursive section does not contribute to noise enhancement, which is one of the key advantages of this approach [8, 9].

Example 1

Let us consider the input signal

$$x(n) = 0.5d(n) + 0.3d(n-1) + 0.1d(n-2)$$

and assume the following parameter values: $N_a = 2, N_b = 1, \Delta = 1$. Then the terms in (12.63) are

$$R_{N_a} = \begin{bmatrix} 0.35 & 0.18 \\ 0.18 & 0.35 \end{bmatrix}, \qquad M = [0.1 \quad 0.3], \qquad C_\Delta = \begin{bmatrix} 0.3 \\ 0.5 \end{bmatrix}$$

which yields

$$A_{\text{opt}} = \begin{bmatrix} 0.045 \\ 1.899 \end{bmatrix}, \qquad b_{\text{opt}} = -0.5745$$

The output MMSE is $E_{\min} = 1 - A^t_{\text{opt}} C_\Delta = 0.0365$. Now, let us change the parameter values to $N_a = 1, N_b = 2, \Delta = 0$. The optimal coefficient values are

$$\begin{aligned} A_{\text{opt}} &= \left[0.3 - [0.3 \quad 0.1] \begin{bmatrix} 0.3 \\ 0.1 \end{bmatrix} \right]^{-1} 0.5 = 2 \\ B_{\text{opt}} &= -\begin{bmatrix} 0.3 \\ 0.1 \end{bmatrix} \; 2 = \begin{bmatrix} -0.6 \\ -0.2 \end{bmatrix} \\ E_{\min} &= 0 \end{aligned}$$

In fact, the channel transfer function is

$$C(Z) = 0.5(1 + 0.6Z^{-1} + 0.2Z^{-2})$$

and the equalizer has the same coefficients, with opposite signs, to cancel the multipath signals.

Example 2

Consider a transmission channel with the maximum-phase transfer function

$$C(Z) = 0.5 + Z^{-1} + 0.8Z^{-2}$$

The input signal to the channel is assumed to be of unit power and uncorrelated, and an additive white noise with power $\sigma_b^2 = 0.1$ is present at the channel output. The parameter values are

$$N_a = N_b = 2, \Delta = 1$$

Then

$$R_{N_a} = \begin{bmatrix} 1.99 & 1.30 \\ 1.30 & 1.99 \end{bmatrix}, \quad M = \begin{bmatrix} 0.8 & 1.0 \\ 0.0 & 0.8 \end{bmatrix}, \quad C_\Delta = \begin{bmatrix} 1.0 \\ 0.5 \end{bmatrix}$$

The equalizer coefficients are

$$A_{\text{opt}} = \begin{bmatrix} 0.4494 \\ 0.7865 \end{bmatrix}, \quad B_{\text{opt}} \begin{bmatrix} 1.1460 \\ 0.6292 \end{bmatrix}$$

The output mean square error is

$$E_{\min} = 0.1573$$

It is interesting to observe that the coefficients of the feedback section B_{opt} are close to those of the minimum-phase function

$$C(Z^{-1}) = 0.8(1 + 1.25Z^{-1} + 0.625Z^{-2})$$

which shows how transversal and feedback sections share the equalization task. The system output is

$$y(n) = 0.2247d(n) + 0.8425d(n-1)$$

and the output mean square error can be decomposed as

$$E_{\min} = [0.2247^2 + (1 - 0.8426)^2] + 0.1[0.4494^2 + 0.7865^2] = 0.1573$$

to show the respective contributions of noise and intersymbol interference. Obviously, with more transversal coefficients, and more system delay, the output mean square error can be reduced.

An important point concerning the DFE is its sensitivity to decision errors. In the tracking mode, the decision errors propagate in the feedback section, and multiple errors can be created. In fact, the DFE, in the decision-directed mode, is an IIR adaptive filter. Its poles should be inside the unit circle, since it is supposed to compensate minimum-phase functions. However, in the adaptive process and in the presence of decision errors, they can go out of the unit circle. As discussed in section 5.11, the exponentially growing output signal eventually brings the poles back within the unit circle, but a burst of errors results whose length is linked to the adaptation time constant. Therefore, the DFE can only work properly if the error rate is small; otherwise, specific measures have to be taken [9].

12.6. FRACTIONALLY SPACED EQUALIZER

In the receiver, decisions are taken at the symbol rate f_b to retrieve the data. So far, it has been assumed that all the functions in the receiver are carried out at that rate, including the equalization. However, it is known that the signal spectrum exceeds the symbol rate by the amount of the roll-off of the Nyquist filter, as shown in Figure 12.7.

Therefore sampling at the symbol rate generates aliasing, and the image of the base band spectrum occurs around the frequency f_b. According to sampling theory, the phase of this image is linked to the sampling times. A shift in timing produces a rotation of the phase of the image, and the baseband spectrum and the image no longer add up in phase, in the filter transition band Δf. Therefore the equalizer is sensitive to the sampling times, and equalization may become impossible for frequencies in the vicinity of half the symbol rate. As is well known in multirate filtering, the solution to the problem is to increase the sampling rate sufficiently to avoid aliasing, which leads to the so-called fractionally spaced equalizer [7, 10–12].

An equalizer with double sampling is shown in Figure 12.8. The input signal sample sequence is split into two sequences which are fed to two separate equalizers $H_1(z)$ and $H_2(z)$ operating at the symbol rate.

The output error is

$$e(n) = d(n - \Delta) - H_1^t X_1(n) - H_2^t X_2(n) \tag{12.66}$$

The optimal coefficient vectors are given by

$$\begin{bmatrix} H_{1\text{opt}} \\ H_{2\text{opt}} \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}^{-1} \begin{bmatrix} r_{d1} \\ d_{d2} \end{bmatrix} \tag{12.67}$$

with, for $i = 1, 2$ and $j = 1, 2$,

$$R_{ij} = E[X_i X_j^t], \qquad r_{di} = E[d(n - \Delta)X_i(n)]$$

The updating of the coefficients is carried out at the symbol rate. It is worth pointing out that the input-signal spectrum, except for the noise, goes to zero in the vicinity of the symbol frequency f_b, as shown in Figure 12.7. Depending on the filter roll-off, the eigenvalues of the input signal AC matrix may be widely spread, which may justify using RLS algorithms instead of LMS for coefficient updating. In any case, if LMS is employed,

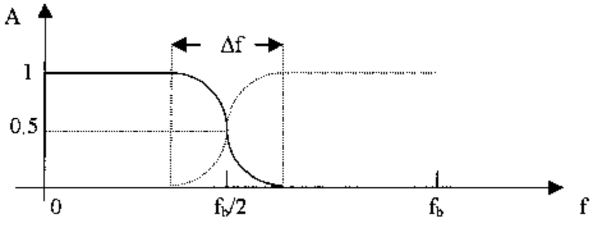


FIG. 12.7 Frequency response of the Nyquist filter.

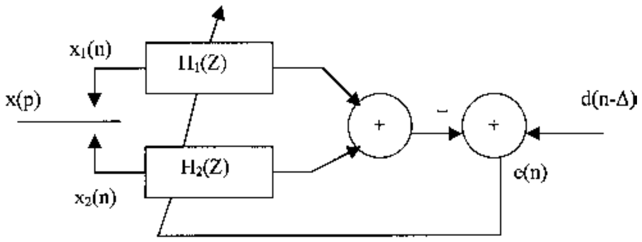


FIG. 12.8 Double sampling equalizer.

it is advisable to use the leakage algorithm described in Section 4.6, to avoid undesirable effects due to insufficient persistent excitation in some frequency bands. In fact, the leakage factor prevents the uncontrolled growth of the coefficients which may occur when no signal is present in some frequency bands.

As concerns noise amplification, it takes place as in any transversal equalizer. The fractionally spaced equalizer can be completed by a feedback section, to make the so-called fractionally spaced DFE. This combination is generally recognized to be the most efficient approach to adaptive equalization.

12.7. MAXIMUM-LIKELIHOOD ALGORITHMS (MLA)

An alternative approach to channel equalization in receivers is maximum-likelihood decoding. Instead of equalizing first and then decoding symbol by symbol to retrieve the data, the idea is to take a block of samples and, assuming that the channel transfer function is known, find the symbol sequence which is the most likely to have produced the block of samples considered [13, 14].

The principle, sketched in Figure 12.9, is the following: assuming that the emitted data $d(n)$ are known or have been decided for $n < 0$, find $d(0)$ by the maximum-likelihood method, using the channel transfer function $C(Z)$,

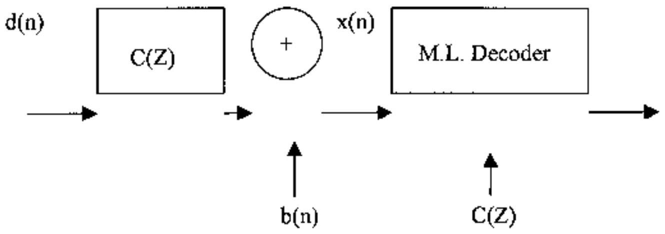


FIG. 12.9 Maximum-likelihood decoding.

which is known or has been estimated. If the channel is FIR with N coefficients, the input signal $x(n)$ is expressed by

$$x(n) = \sum_{i=0}^{N-1} c_i d(n-i) + b(n) = C^t D(n) + b(n) \tag{12.68}$$

where $b(n)$ is a white Gaussian noise with power σ_b^2. The sought datum $d(0)$ is involved explicitly in N received samples:

$$\begin{aligned} x(0) &= c_0 d(0) + c_1 d(-1) + \cdots + c_{N-1} d(-N+1) + b(0) \\ x(1) &= c_0 d(1) + c_1 d(0) + \cdots + c_{N-1} d(-N+2) + b(1) \\ &\vdots \\ x(N-1) &= c_0 d(N-1) + c_1 d(N-2) + \cdots + c_{N-1} d(0) + b(N-1) \end{aligned} \tag{12.69}$$

Now an error vector $E(0)$ can be built with the M elements ($M \geqslant N$)

$$\begin{aligned} e(0) &= x(0) - C^t D(0) \\ e(1) &= x(1) - C^t D(1) \\ &\vdots \\ e(M-1) &= x(M-1) - C^t D(M-1) \end{aligned} \tag{12.70}$$

The maximum-likelihood technique consists in searching for the minimum $E_{\min}$ of the L_2 norm of the vector $E(0)$:

$$\|E(0)\|_2^2 = \sum_{i=0}^{M-1} e^2(i) \tag{12.71}$$

for all possible emitted data values: $d(0), d(1), \ldots, d(M-1)$. The decision for $d(0)$ is the value that is involved in $E_{\min}$.

The following remarks are relevant.

- the channel coefficients are used in the computation of the error vector.
- the decision is taken with the delay $\Delta = M - 1$.
- if each symbol carries K bits, $d(n)$ can take 2^K values, and 2^{KM} vector norms have to be computed.

Clearly, the computational complexity is a major issue, and the parameters M and K have to be kept small. In fact, the scheme is practical for radio channels with a few transmission paths and binary modulation.

Significant reductions in computational complexity can be obtained with the Viterbi algorithm. The principle is as follows:

- initial conditions: vector $D(-1)$ is known.
- time 0: for each of the 2^K possible transitions from vector $D(-1)$ to vector $D(0)$, $e^2(0)$ is calculated.
- time 1: from each of the 2^K vectors, or states, reached at time 0, 2^K transitions are possible to reach a state of time 1. The transitions define paths starting from the known state $D(-1)$ and ending at state $D(1)$. To each path, a weight is attached which is the L_2 norm of the error vector, namely: $e^2(0) + e^2(1)$.
- time $M-1$: the path that has the smallest weight is selected, and the value $d(0)$ involved in that path is selected as the decoder output.

The geometrical representation of the states and the transitions at each time is called a trellis. Note that it is sufficient to maintain $2^{K(N-1)}$ states in the trellis. An example is shown in Figure 12.10, for $K = 1$ and $N = 3$.

The simplifications in computations come from the following observations:

- at a given time n, of all the paths ending at state $D(n)$, it is sufficient to keep only the path with the smallest weight.
- at time n, only the error values $e^2(n)$ attached to the 2^K transitions for each of the states have to be calculated. Since there are $2^{K(N-1)}$ states, 2^{KN} error values have to be calculated.
- each time, it might be sufficient to keep only a few paths, the so-called survivors: those which have the smallest weights.

The performance of the scheme is sensitive to the parameter values, particularly the block size M and the number of survivors, but the most critical input is the channel estimate which has to be carefully monitored.

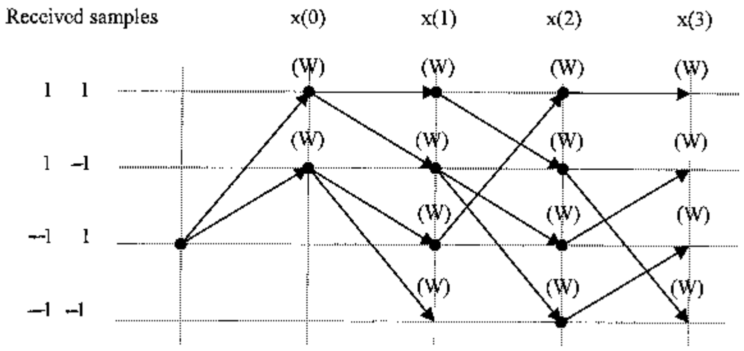


FIG. 12.10 Trellis for the Viterbi algorithm.

12.8. COMPARISON OF EQUALIZATION APPROACHES

In order to assess the relative performance of the various techniques considered so far for optimal reception, it is useful to provide some figures of merit. The noise level that leads to a specified bit error rate can be a good basis for comparison. The target bit error rate retained is BER 10^{-3}.

With the maximum-likelihood (ML) technique, when the correct data vector is picked, the minimum $E_{\min}$ of the norm of the error vector is given by:

$$E_{\min} = \sum_{i=0}^{M-1} b^2(n) \tag{12.72}$$

when expressions (12.68), (12.70), and (12.71) are combined. Assuming binary data, a false $d(0)$ yields the norm

$$E_f = \sum_{i=0}^{N-1} [b(i) + 2c_i]^2 + \sum_{i=N}^{M-1} b^2(n) \tag{12.73}$$

The difference is

$$E_f - E_{\min} = 4\left[\sum_{i=0}^{N-1} c_i b(i) + \sum_{i=0}^{N-1} c_i^2\right] \tag{12.74}$$

The variable

$$u = \sum_{i=0}^{N-1} c_i b(i) \tag{12.75}$$

is Gaussian, and its variance is

$$\sigma_u^2 = \sigma_b^2 \sum_{i=0}^{N-1} c_i^2 \tag{12.76}$$

An error occurs whenever $E_f - E_{\min} < 0$. The probability of this situation is smaller than 10^{-3} if

$$\sigma_u < \frac{1}{3} \sum_{i=0}^{N-1} c_i^2 \tag{12.77}$$

Then the condition on the square root of the noise variance is

$$\sigma_b < \frac{c_0}{3}\left[1 + \sum_{i=1}^{N-1} \frac{c_i^2}{c_0^2}\right]^{\frac{1}{2}} \tag{12.78}$$

Now, let us consider a DFE, with a recursive section only and $N-1$ coefficients, as shown in Figure 12.11. The signal at the input of the decision device is

$$y(n) = x(n) - \sum_{i=1}^{N-1} c_i d(n-i) \tag{12.79}$$

and, if the coefficients take on the exact values, then

$$y(n) = c_0 d(n) + b(n) \tag{12.80}$$

Therefore, assuming that $b(n)$ is a Gaussian noise, the condition BER $< 10^{-3}$ implies

$$\sigma_b < \frac{c_0}{3} \tag{12.81}$$

As concerns the transversal equalizer, it implements the inverse of the channel transfer function or an approximation as follows:

$$C^{-1}(Z) = \frac{1}{c_0\left[1 + \sum\limits_{i=1}^{N-1} \frac{c_i}{c_0} Z^{-i}\right]} \approx \frac{1}{c_0}\left[1 + \sum_{i=1}^{N-1} h_i Z^{-i}\right] = H(z) \tag{12.82}$$

The condition on the bit error rate leads to

$$\sigma_b < \frac{c_0}{3} \frac{1}{\left[1 + \sum\limits_{i=1}^{N-1} h_i^2\right]^{\frac{1}{2}}} \tag{12.83}$$

Expressions (12.78), (12.81), and (12.83) provide a ranking of the three techniques. One can say that, with respect to the input noise, the DFE is neutral, since it keeps the input value of the signal-to-noise ratio, and the transversal equalizer degrades the SNR, while the MLA improves the SNR.

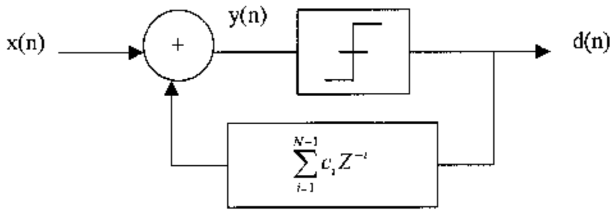


FIG. 12.11 Purely recursive DFE.

In fact, in the maximum-likelihood approach, the SNRs of the various propagation paths add up.

Example

Let us consider the simple case where

$$C(Z) = \frac{1}{1 - aZ^{-1}} \tag{12.84}$$

The noise-level limits for the three approaches are given in table 12.1.

It must be pointed out that the above comparison implies many assumptions and, in practical situations, the comparison is not so clear-cut. In many applications, the transversal equalizer is still the preferred technique, because of its flexibility and ease of implementation, its robustness to decision errors, and its relative computational simplicity. However, if the system has to operate close to the SNR limit in a very noisy environment, and if long delays are acceptable, the MLA might be the right approach.

12.9. CARRIER FREQUENCY ESTIMATION

Shifts in carrier frequency can occur during the transmission of modulated signals, due to local oscillators or, in mobile radio environments, to the so-called Doppler effect. Assuming that a mobile station is moving at speed v and it is receiving a signal emitted from a fixed base station with frequency f, then the received frequency is shifted by Δf, such that

$$\Delta f = f\frac{v}{c}\cos\theta \tag{12.85}$$

where c is the radiowave velocity and θ is the angle between the mobile travelling direction and the direction of the source.

In the receiver, the frequency shift has to be compensated, and a conventional technique is the phase-locked loop. However, when the transmission takes place in bursts, like in some satellite communications, faster techni-

Table 12.1 Comparison of Equalization Techniques

	MLA	DFE	Transversal
$3\sigma_b <$	$\frac{1}{\sqrt{1-a^2}}$	1	$\frac{1}{\sqrt{1+a^2}}$
$a = 0.8$	1/0.6	1	1.3

ques are required. An approach is to estimate the carrier frequency from the signal samples in the burst, then cancel it and proceed with the operations of the optimal receiver [15, 16].

A preliminary step is to get rid of the modulation, and the methods used to reach that goal are closely related to the modulation itself. For example, with 4-QPSK modulation, it is sufficient to raise the signal samples to the 4th power. A 4-QPSK signal is expressed by:

$$x(n) = Ae^{j\left(\pm\frac{\pi}{4}\pm\frac{\pi}{2}+\varphi\right)} \tag{12.86}$$

and

$$x^4(n) = -A^4 e^{j4\varphi} \tag{12.87}$$

Indeed, a high-level noise component must be added to $x(n)$ to reflect practical conditions, and it is enhanced by the operation. In practice, it is recommended to multiply the phase of the signal $x(n)$ by 4 and keep the magnitude or, even, set it to unity.

In any case, the problem of compensating the carrier frequency in a burst boils down to finding the frequency of a single sinusoid in noise, using N samples. Let us assume the signal to be

$$x(n) = Ae^{j(n\omega+\varphi)} + b(n) \quad (0 \le n \le N-1) \tag{12.88}$$

A target accuracy has to be specified first. With 4-QPSK, the residual frequency error $\Delta\omega$ must be small enough, so that after N samples, data extraction is still unambiguous, which implies

$$N\Delta\omega < \frac{1}{2}\frac{\pi}{2} \tag{12.89}$$

Now, the issue is whether this accuracy can be met, given the signal-to-noise ratio SNR. The answer is provided by the estimation bounds.

The results of section (10.9) can be applied in a straightforward manner. There are three unknown parameters: namely the amplitude A, the angular frequency ω, and the phase φ. Using definition (12.88) for the signal samples, the gradient matrix M_G is such that

$$M_G\overline{M}_G^t = \begin{bmatrix} N & -jA\frac{N(N-1)}{2} & -jAN \\ jA\frac{N(N-1)}{2} & A^2\frac{N(N-1)(2N-1)}{6} & A^2\frac{N(N-1)}{2} \\ jAN & A^2\frac{N(N-1)}{2} & A^2N \end{bmatrix} \tag{12.90}$$

Since the three parameters to be estimated are real, the variances are obtained using only the real part of the matrix $M_G\overline{M}_G^t$. Assuming the noise to be complex with power σ_b^2, the inverse of the real part of the matrix $M_G\overline{M}_G^t$ is multiplied by $\sigma_b^2/2$ to produce the variances

$$\begin{aligned}&\mathrm{var}\{A\} = \frac{\sigma_b^2}{2N}, \qquad \mathrm{var}\{\omega\} = \frac{1}{\mathrm{SNR}}\frac{6}{N(N^2-1)},\\ &\mathrm{var}\{\varphi\} = \frac{1}{\mathrm{SNR}}\frac{(N-1)}{N(4N^2-3N+1)}\end{aligned} \tag{12.91}$$

Then, for a given SNR, the number of samples must satisfy the inequality

$$\frac{1}{\mathrm{SNR}}\frac{6}{N(N^2-1)} < \frac{1}{N^2}\left(\frac{\pi}{4}\right)^2 \tag{12.92}$$

or

$$N > \frac{96}{\pi^2 \mathrm{SNR}} \tag{12.93}$$

For example, if SNR = 0.1 (−10 dB), then $N > 97$. Note that, if a peak factor is introduced in inequality (12.89) to reflect a probability limit, then the number of samples is multiplied by the same factor.

Once equality (12.93) has been checked, the next step in to find an estimation algorithm able to meet the bound. A classical approach consists of performing an FFT on the input samples, taking the squared modulus of the transformed values, and looking for the maximum of the signal spectral power density through interpolation. The frequency corresponding to the maximum is the desired estimate. One can also exploit the following property: the autocorrelation function of a cissoid in noise is still a cissoid in noise, as shown by the computation

$$r(p) = \frac{1}{N-p}\sum_{n=p}^{N-1} x(n)\bar{x}(n-p) = A^2 e^{jp\omega} + b'(p) \tag{12.94}$$

with $b'(p)$ a noise component given by

$$b'(p) = \frac{1}{N-p}\left[\sum_{n=p}^{N-1} Ae^{j(n\omega+\varphi)}\bar{b}(n-p) + Ae^{j[(n-p)\omega+\varphi]}b(n) + b(n)\bar{b}(n-p)\right] \tag{12.95}$$

Now the autocorrelation function of the sequence $r(p)$ can be computed. After several such iterations, the element with index $p = 1$ in the autocorrelation function gives the desired estimate, after division by the element with index 0, which is the corresponding signal power. It is worth pointing out that the autocorrelation function of a given sequence is computed efficiently through a cascade of three operations: calculating an FFT, taking the square of the modulus of the transform outputs, and calculating the inverse FFT.

Once the estimate $e^{j\tilde{\omega}}$ of the true carrier frequency $e^{j\omega}$ has been obtained, the compensation can be carried out as shown in Figure 12.12.

The optimal receiver can use the maximum-likelihood algorithm described in Section 12.7 to retrieve the data from the N signal samples.

12.10. ALGORITHMS FOR RADIO COMMUNICATIONS

Signal processing algorithms have been essential for all kinds of radio transmission systems, e.g., microwave fixed links, satellites, and cellular wireless networks. Some have been developed specially, like the DFE equalizer which was introduced to cope with the high level of noise present in some radio channels, while the transversal equalizer gave satisfactory performance in wired communications, where accurate equalization is the main priority and noise is less of a problem.

A typical example of the importance of some basic algorithms, like those given in section 12.1, is given by the power control in cellular systems, particularly those based on code division multiple access (CDMA) [17]. In CDMA networks, several users share the same transmission channel, and it is essential that the corresponding signals reach the base station receiver with similar power levels, in order for the receiver to reliably separate and decode all the signals. However, in a mobility context, the distance between a user and the base station can vary considerably in a short time. Therefore, nearby emitters must have their power reduced while distant emitters have to push up their power levels. In fact, for a CDMA-based cellular system to work properly, a global strategy to adaptively control the power levels of the mobile users has to be implemented. To some extent, the same applies to the base station itself, if it has the capability to perform a space separation of the emitted signals.

In this section, several algorithms which find application in radio receivers are described.

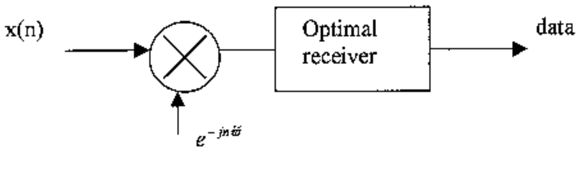


FIG. 12.12 Compensation of the carrier frequency.

12.10.1. Zero Forcing (ZF) Algorithm

The updating of the coefficients in a transversal equalizer can be simplified if the data vector is employed instead of the input signal vector. The equations are

$$\begin{aligned} e(n+1) &= d(n+1) - H^t(n)X(n+1) \\ H(n+1) &= H(n) + \delta e(n+1)D(n+1) \end{aligned} \tag{12.96}$$

where $D(n+1)$ is the vector of the N most recent output data at time $n+1$, and the filter has N coefficients. The scheme is particularly advantageous with binary data. The updating equation leads to the cancellation of the expectation

$$E[e(n)D(n)] = E[d(n)D(n) - D(n)X^t(n)H_{\text{opt}}] = 0 \tag{12.97}$$

when the optimal coefficient vector H_{opt} has been obtained. Now, the input $x(n)$ is related to the data by

$$x(n) = \sum_{i=0}^{\infty} c_i d(n-i) \tag{12.98}$$

Assuming uncorrelated data, equation (12.97) is rewritten as

$$\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} - \begin{bmatrix} c_0 & 0 & \cdots & 0 \\ c_1 & c_0 & \cdots & 0 \\ \vdots & \ddots & & \\ c_{N-1} & c_{N-2} & \cdots & c_0 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-1} \end{bmatrix} = 0 \tag{12.99}$$

which means that the coefficients of the equalizer are such that, on average, the $N-1$ first terms of the impulse response of the channel and equalizer in cascade are forced to zero. This is apparent if the product $C(Z)H(Z)$ is computed. In fact, the coefficients H_{opt} are the N first values of the inverse of the channel impulse response.

The ZF algorithm can be analyzed as the conventional LMS algorithm. For example, the stability condition is derived from the a posteriori output error $\varepsilon(n+1)$ by:

$$\varepsilon(n+1) = e(n+1)[1 - \delta D^t(n+1)X(n+1)] \tag{12.100}$$

Following the reasoning of Section 4.2 and taking the expectations of the absolute values of both sides yields the stability condition

$$0 < \delta < \frac{2}{Nc_0} \tag{12.101}$$

The data are assumed to have unit power. It is worth pointing out that, since the channel coefficients are not available beforehand, some initial guess is necessary to select the adaptation step size δ. In practice, it might be convenient to use the square root of the input signal power.

12.10.2. The Constant Modulus Algorithm (CMA)

The CMA algorithm belongs to the family of the so-called blind algorithms, which means that no reference signal is available to guide the adaptation process, which has to rely on a priori known characteristics of the signal.

A typical characteristic of many radio signals is the constant modulus, which is a basic requirement to fully exploit the dynamic range of the amplifiers and achieve the best protection against noise at a given signal power level. Frequency modulation and phase modulation have this property [18].

The CMA algorithm is commonly used in digital phase-locked loops, and in channel equalizers when appropriate. Since there is no need to transmit an initial learning sequence, the approach is particularly useful in circumstances where it is impossible or impractical to use a learning sequence, as in broadcasting for example. The principle is shown in Figure 12.13.

The cost function J_{CM} is defined by:

$$J_{CM} = \frac{1}{4} E[A^2 - |\tilde{y}(n)|^2]^2 \tag{12.102}$$

The instantaneous gradient is the derivative, with respect to the coefficients, of the quantity in brackets. For real signals, one gets

$$\mathrm{Grad}(n) = -[A^2 - |\tilde{y}(n)|^2]\tilde{y}(n)X(n) \tag{12.103}$$

The filter coefficients are updated by:

$$H(n+1) = H(n) + \delta[A^2 - |\tilde{y}(n+1)|^2]\tilde{y}(n+1)X(n+1) \tag{12.104}$$

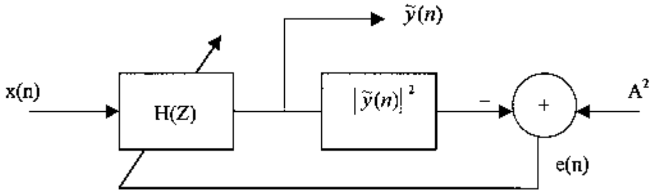


FIG. 12.13 Principle of the CMA equalizer.

For complex signals, the derivative of the quantity $|\tilde{y}(n)|^2$ with respect to the coefficients is the vector $2\tilde{y}(n)\bar{X}(n)$, and the updating equation becomes

$$H(n+1) = H(n) + \delta[A^2 - |\tilde{y}(n+1)|^2]\tilde{y}(n+1)\bar{X}(n+1) \tag{12.105}$$

With the same approach, sign algorithms can also be derived.

A critical issue with the above algorithm is convergence. Since the cost function is not quadratic, and the algorithm includes nonlinear operations, local minima exist. In addition, the performance, particularly the convergence time, cannot be predicted.

12.10.3. Optimal Multipath Equalization

Multipath propagation is a typical feature of most radio channels. Rather than mitigate the effect, the optimal receiver should exploit it and make the various propagation paths contribute to the transmission. In fact, signals carrying the same information should be combined in such a way that the corresponding signal-to-noise ratios add up. The possibility has already been demonstrated for the matched filter and the maximum-likelihood algorithm, and it is considered below from a direct-weighting perspective.

Let us consider a set of K signals x_k, such that

$$x_k = 1 + b_k \quad (0 \leq k \leq K-1) \tag{12.106}$$

The desired quantity is 1, and b_k is a white noise, of power B_k, in the measurement. The K noise components are assumed uncorrelated. Next, let us consider the weighted summation

$$y = x_0 + \sum_{k=1}^{K-1} h_k x_k \tag{12.107}$$

The issue is to find the $K-1$ coefficients h_k that maximize the signal-to-noise ratio in the signal y. Substituting (12.106) into (12.107) leads to

$$y = \left(1 + \sum_{k=1}^{K-1} h_k\right) + b_0 + \sum_{k=1}^{K-1} h_k b_k \tag{12.108}$$

The signal-to-noise ratio is expressed by

$$\mathrm{SNRy} = \frac{\left(1 + \sum_{k=1}^{K-1} h_k\right)^2}{B_0 + \sum_{k=1}^{K-1} h_k^2 B_k} \tag{12.109}$$

The maximum of the function is obtained when all the derivatives with respect to the coefficients are zero. That condition for coefficient h_i can be shown to be

$$B_0 + \sum_{k=1}^{K-1} h_k^2 B_k = \left(1 + \sum_{k=1}^{K-1} h_k\right) h_i B_i \tag{12.110}$$

Thus $K - 1$ such equations are obtained, and they form a system that provides the coefficient values. It is readily verified by mere substitution that the solution to the system is

$$h_k = \frac{B_0}{B_k} \quad (0 \le k \le K-1) \tag{12.111}$$

The corresponding SNR is obtained by substituting (12.111) into (12.109), which yields, after simple algebraic manipulations,

$$\mathrm{SNRy} = \frac{1}{B_0} + \sum_{k=1}^{K-1} \frac{1}{B_k} \tag{12.112}$$

The signal-to-noise ratio of the weighted sum y is just the sum of the individual SNRs. Coming back to the radio-transmission context, the output of a multipath channel can be expressed by

$$x(n) = \sum_{i=0}^{K-1} [c_i d(n-i) + b_i(n)] \tag{12.113}$$

where each of the K paths has an attenuation c_i, a delay i and a noise component $b_i(n)$ with power B_i. If the paths can be separated, and if the signal with index i is multiplied by c_i^{-1}, the noise power for that signal becomes B_i/C_i^2. Now, if the signals are properly delayed and summed with the coefficients given by (12.111), namely

$$h_i = \frac{c_i^2}{B_i} \tag{12.114}$$

the signal-to-noise ratio of the signal $y(n-K)$ so obtained is given by

$$\mathrm{SNRy} = \sum_{i=0}^{K-1} \frac{c_i^2}{B_i} \tag{12.115}$$

This expression becomes identical to (12.36) if all the paths have the same noise power. In fact, a matched filter has been obtained.

From an application viewpoint, the above developments might look like an academic exercise, since the paths generally cannot be separated and the SNRs measured individually. However, the approach can be applied as such in code-division multiple-access (CDMA) systems, because of the decorrelation brought by the coding to delayed versions of the signal, and it is called the "RAKE" receiver [17]. The structure is shown in Figure 12.14.

The channel impulse response is measured, and the M most significant paths are identified, with their delays. The input multipath signal $x(n)$ is fed to a set of M branches, called the fingers, which consist of delays, correlators with the spreading sequence, and multipliers by the weighting coefficients. The signal obtained by summation of the outputs of the branches is fed to the detector which supplies the data. Actually the scheme is a direct implementation of the matched filter.

12.10.4. Adaptive Antennas for Cellular Systems

Adaptive antenna arrays have the potential to enhance the capacity of cellular systems by exploiting spatial separation between subscribers. The base station concentrates the signal power in the directions of the mobile units to be reached and reduces the power radiated in the directions of the other units. The same principle is applied to the reception of the signals emitted by the mobiles. In fact, in cellular systems, the base stations are in a position to perform a global optimization of the whole transmission system, by combining send-and-receive information and by exploiting jointly the estimation and equalization techniques for the paths of the mobiles they are managing. However, in practice, a step-by-step introduction of these concepts is necessary.

The block diagram of an adaptive array and the combination of space diversity and equalization is shown in Figure 12.15.

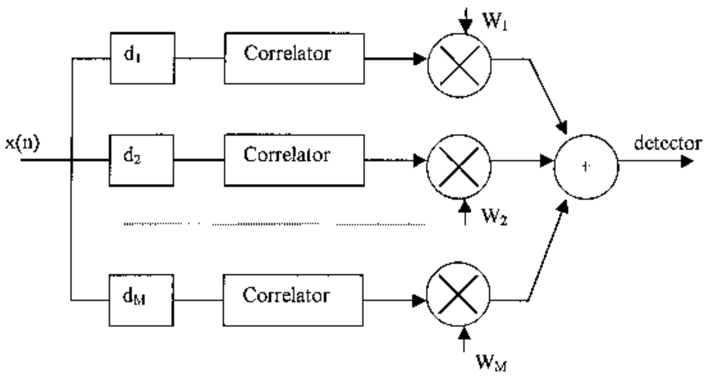


FIG. 12.14 Structure of the RAKE receiver.

In fact, it is a DFE with a multichannel transversal section which can be fractionally spaced. The algorithms presented in Sections 12.5 and 12.6 can be used, as well as the results of Section 11.7 on adaptive antennas. If the received signals are narrow-band, the transversal sections can be simplified to single coefficients.

The scheme can direct the main antenna beam towards the desired mobile unit and position nulls in the directions of interfering units, while equalizing the transmission channel and minimizing the received noise. In fact, it is shown to be optimal in the presence of multipath propagation and jammers [19]. The problem with this general approach resides in the implementation, with the practical limitations in terms of number of array elements, geometry, calibration, and filter dimensioning. A long learning phase is likely to be necessary.

In order to speed up the process, a cascade approach can be contemplated, consisting of two steps. First an estimation of the directions of arrival of the signals is performed. Then constrained equalization is employed, with the appropriate algorithms of Sections 4.12 or 7.7. Direction of arrival estimation and tracking is efficiently achieved by subspace methods, as in Section 10.8, and techniques are available to efficiently carry out the singular-value decomposition of the relevant covariance matrices [20].

At this point, the similarity between neural networks and equalizers is worth pointing out. In fact, the adaptive equalizer in the decision-directed mode is a classifier, like the perceptron described in Section 11.9. The results

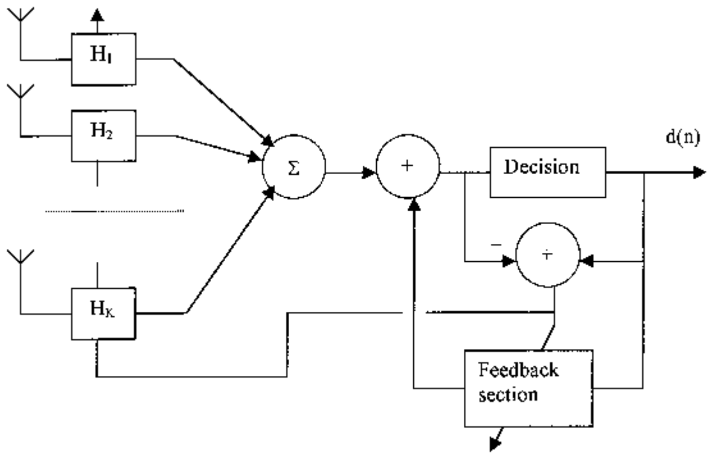


FIG. 12.15 Adaptive multichannel equalizer.

obtained in equalization can be applied to perceptrons which use the same nonlinear operator, in particular concerning convergence speed and residual error power. Conversely, multilayer perceptrons can be used to equalize nonlinear channels.

12.11. CONCLUSION

A wide range of adaptive techniques has been presented for equalization and data retrieval in digital receivers. They are intented to cope with the many different contexts encountered in communications, and they illustrate the flexibility and efficiency of the general concept. For each specific application, a technique or a combination of different techniques can be identified to meet the objectives, while satisfying the constraints.

The trend in communications is towards using more powerful algorithms and sophisticated structures. A typical example in wireless cellular communications is the use of multiple antenna elements at both the base station and the mobile set. A multiple-input multiple-output system results, which should permit a better exploitation of the radio channel and an increase in data throughput [21]. In fact, transmission channels—and particularly radio channels, because of the usable spectrum limitations—constitute a limited resource, in contrast with an ever-growing demand for higher bit rates and mobility.

Another beneficial effect of the progress in signal processing algorithms is that it is now possible to reliably exploit channels that were judged until recently as unusable for high-rate communication. A remarkable example is the copper wire subscriber loop, which can now support bit rates of several Mbit/s with the xDSL (Digital Subscriber Loop) systems [22–23].

EXERCISES

1. The scheme of Figure 12.1 is applied to the signal $x(n) = \sin \frac{1}{4} n\pi$. Give the expression of the sequence $P_x(n)$, an approximate expression for the gain $G(n)$, and the output $u(n)$. Assuming that the output harmonic distortion must be smaller than one percent, what is the maximum value of the filter parameter ε? What is the accuracy of the power control in that case?
2. In order to reduce the computational complexity and speed up long echo cancellers, use is sometimes made of interpolated adaptive filters, in which a fraction only of the coefficient set is actually adaptively computed, and the remainder is derived through linear interpolation. Show that, in such a scheme, interpolation can be carried out on the

input signal, and adaptive filtering is performed on the interpolated signals with a reduced set of coefficients.

Give the details of the operations for the simple case of $N = 5$ coefficients and interpolation factor $K = 2$. Discuss the performance of the interpolated adaptive filter in that case.

3. Schwarz inequality: prove the inequality

$$[a_1 b_1 + a_2 b_2]^2 \leq [a_1^2 + a_2^2][b_1^2 + b_2^2]$$

and give the conditions for equality. Use this result to prove inequality (12.31).

4. A propagation channel has the coefficient vector $C^t = [0.8, 1.0, 0.5]$, and the power of the additive noise is $\sigma_b^2 = 0.1$. Give the coefficients of the filter matched to the channel and the impulse response of the channel-equalizer cascade. What is the overall delay and the maximal value of the signal-to-noise ratio?

5. A transversal filter is considered to equalize a channel with the transfer function

$$C(Z) = \frac{0.5}{1 - 0.5Z^{-1}}$$

Compute the power of the received signal $x(n)$, assuming unit-power uncorrelated input data. For $N = 2$ coefficients, give the coefficient values.

A white noise with power $\sigma_b^2 = 0.1$ is added to the received signal. Compute the values of the coefficients and compare with the previous result. What is the noise enhancement factor?

From the impulse response of the channel-equalizer cascade, derive the intersymbol interference power.

6. A two-path microwave channel is equalized by a transversal filter with three coefficients. The model is

$$C(Z) = 1 - 0.7e^{-j\pi/4}Z^{-1}$$

Assuming that the input data signal is uncorrelated and of unit power, compute the equalizer coefficient vector H_{opt} and the noise enhancement factor.

An alternative transversal equalizer $F(Z)$ is obtained by taking the first three terms in the series development of $C^{-1}(Z)$. Give the noise enhancement factor in that case.

By computing the products $C(Z)H_{\text{opt}}(Z)$ and $C(Z)F(Z)$, show the intersymbol interference in both cases.

Give the coefficient updating equation in the LMS approach. What is the stability limit for the adaptation step size δ?

7. Consider a transmission channel with the transfer function

$$C(Z) = 0.5 + 0.25Z^{-1} + 0.1Z^{-2} + 0.05Z^{-3}$$

The input signal $d(n)$ to the channel is assumed to be of unit power and uncorrelated. Give the values of the coefficients of a decision-feedback equalizer with two coefficients in the transversal section and a single coefficient in the feedback section ($N_a = 2, N_b = 1$), taking $\Delta = 0$ as the total system delay. Compute the output mean square error, and give the expression of the system output to show the residual intersymbol interference.

Next, an additive white noise with power $\sigma_b^2 = 0.1$ is present at the channel output. Compute the equalizer coefficients in that case, and show the relative contributions of noise and intersymbol interference in the output mean square error.

8. At the input of a double sampling equalizer, the received signals are

$$x_1(n) = d(n) + 0.7d(n-1) + 0.2d(n-2) + b_1(n)$$
$$x_2(n) = 0.9d(n) + 0.5d(n-1) + b_2(n)$$

where $b_1(n)$ and $b_2(n)$ are independant white noise signals with power $\sigma_b^2 = 0.05$ and the data $d(n)$ are uncorrelated and of unit power.

Compute the optimal coefficients of the equalizer, with $N = 2$ coefficients in each branch.

Give the output error power. Compare with the regular nonfractionally spaced equalizer, which uses $x_1(n)$ only.

9. A transmission channel has the transfer function

$$CZ) = 1 + 0.7Z^{-1} + 0.2Z^{-2}$$

The input data are $d(n) = \pm 1$ and the following sequence is received from time step 0 to time step 11:

$$x(n) = \{0.6, 1.2, 1.0, -1.2, 0.4, 0.2, 0.6, -0.2, -1.2, -0.5, 0.7, 1.4\}$$

Knowing that $d(-1) = 1$ and $d(-2) = -1$, use the MLA algorithm to find the sequence which has been transmitted.

Give the trellis representation with the states, transitions, paths, and weights. Discuss the computational complexity.

10. In a radio receiver, the antenna consists of a linear array of three uniformly spaced isotropic sensors, separated by the distance $d = \lambda/2$, where λ is the wavelength of the carrier frequency f_0, i.e., $\lambda = c/f_0$ and c is the velocity of light. The useful signal is arriving from the incidence angle $\theta = \pi/6$ with respect to the array. The base-band signal

received by the center element in the array, after frequency shift and sampling, is expressed by

$$x_1(n) = s(n) + b_1(n)$$

where $s(n)$ is the useful source signal and $b_1(n)$ is a white noise with power σ_b^2.

Give the expressions of the signals $x_0(n)$ and $x_2(n)$ received by the other antenna elements, denoting by $b_0(n)$ and $b_2(n)$ the corresponding noise signals assumed to have the power σ_b^2.

After weighted summation, the system output is expressed by

$$y(n) = \alpha x_0(n) + x_1(n) + \beta x_2(n)$$

Give the relationship between the coefficients α and β necessary to cancel a jammer arriving from incidence angle 0 (orthogonal to the array). Compute the coefficients that maximize the output signal-to-noise ratio.

REFERENCES

1. I. A. Glover and P. M. Grant, *Digital Communications*, Prentice-Hall Europe, 1998.
2. D. W. Lin, "MMSE Echo Cancellation and Equalization for DSL Transmission," *IEEE Trans.* **COM-38**, 31–38 (January 1990).
3. C. Breining, E. Hänsler, et al. "Acoustic Echo Control—An Application of Very High Order Adaptive Filters," *IEEE Signal Processing Magazine* **16**, 42–69 (July 1999).
4. G. O. Glentis, K. Berberidis and S. Theodoridis, "Efficient Least Squares Adaptive Algorithms for FIR Transversal Filtering," *IEEE Signal Processing Magazine* **16**, 13–41 (July 1999).
5. ITU-T, Recommendation G.168, *Echo Cancellers for Digital Networks*, Geneva, 1997.
6. R. W. Lucky, Techniques for Adaptive Equalization of Digital Communication Systems, *BSTJ* 45, 255–286 (February 1966).
7. O. Macchi, *Adaptive Processing, the LMS Approach with Application in Transmission*, John Wiley, Chichester, 1994.
8. C. A. Belfiore and J. H. Park, "Decision Feedback Equalization," *Proc. IEEE* **67**, No. 8 (August 1979).
9. D. L. Duttweiler, J. E. Mazo, and D. G. Messerschmitt, "Error Propagation in Decision Feedback Equalizer," *IEEE Trans.* **IT-20**, 490–497 (July 1974).
10. J. R. Treichler, I. Fijalkow, and C. R. Johnson, "Fractionally Spaced Equalizers," *IEEE Signal Processing Magazine* **13**, No. 3 (May 1996).
11. G. Ungerboek, "Fractional Tap Spacing Equalizer and Consequences for Clock Recovery in Data Modems," *IEEE Trans.* **COM-2**, 856–864 (August 1976).

12. R. D. Gitlin and S. B. Weinstein, ‘‘Fractionally Spaced Equalization: An Improved Digital Transversal Equalizer,’’ *BSTJ* **60**, 275–296 (February 1981).
13. G. D. Forney, ‘‘Maximum Likelihood Sequence Estimation in the Presence of Intersymbol Interference,’’ *IEEE Trans.* **IT-18**, 363–378 (May 1972).
14. J. Proakis, *Digital Communications* (3rd edn), McGraw-Hill, 1995.
15. H. Meyr, M. Moeneclay, and S. A. Fechtel, *Digital Communication Receivers: Synchronization, Channel Estimation and Signal Processing*, John Wiley, Chichester, 1998.
16. U. Mengali and A. N. D'Andrea, *Synchronization Techniques for Digital Receivers*, Plenum Press, 1997.
17. A. J. Viterbi, *CDMA: Principles of Spread Spectrum Communications*, Addison Wesley, Reading, MA, 1995.
18. J. R. Treichler and M. G. Agee, ‘‘A New Approach to Multipath Correction of Constant Modulus Signals,’’ *IEEE Trans.* **ASSP-31**, 349–372 (April 1983).
19. P. Balaban and J. Salz, ‘‘Optimum Diversity Combining and Equalization in Cellular Mobile Radio,’’ *IEEE Trans.* **COM-40**, No. 5 (May 1992).
20. B. Yang, A. Perez-Nera, and M. Lagunas, ‘‘Adaptive Arrays for Communications,’’ in *DSP in Telecommunications*, Edited by A. Figueiras-Vidal, Springer Verlag, London, 1996, pp. 93–170.
21. A. F. Naguib, N. Seshadri, and A. R. Calderbank, ‘‘Increasing Data Rates over Wireless Channels,’’ *IEEE Signal Processing Magazine* **17**, 77–92 (May 2000).
22. W. Y. Chen, *DSL: Simulation Techniques and Standards Development for Digital Subscriber Line Systems*, Macmillan, Indianapolis, 1998.
23. T. Starr, J. Cioffi, and P. Silverman, *Understanding Digital Subscriber Technology*, Prentice Hall, Englewood Cliffs, N.J., 1999.