Multirate Adaptive Filtering Algorithms: Analysis and Applications

Thesis by

Vinay Padmakar Sathe

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1991

(Submitted May 6, 1991)

Acknowledgments

I would like to acknowledge the brilliance, sincerity and helping nature of my advisor, Dr. P. P. Vaidyanathan. Without his patience and guidance, this work would not have taken place. I hope that his wisdom guides me through the rest of my life also.

My stay at Caltech has been a wonderful experience. Everyone here, from the faculty to the janitorial staff, was helpful and courteous. I am proud to be a part of this group. In particular, I want to thank my colleagues: Dr. Truong Nguyen, Dr. David Koilpillai, Dr. Vincent Liu, Dr. Zinnur Doganata, Quan Hoang, Tsu-Han Chen, Anand Soman and Igor Djokovic for the interaction we had through group meetings and otherwise. There technical feedback has helped improve the quality of the work presented here.

I would also like to thank the members of my thesis examination committee: Dr. Goodman, Dr. Posner, Dr. Sideris and Dr. Simon. My interaction with them and other faculty members at Caltech, both on the personal and academic level has been most satisfying.

Abstract

In this thesis, we discuss the application of multirate signal processing concepts to adaptive filtering to achieve low computational complexity and speed. To be able to analyze systems involving multirate building blocks, we have studied effects of multirate filters on the statistics of random inputs. As an example of the multirate adaptive filtering concepts, we study the problem of adaptive identification of an unknown bandlimited channel. We show that the bandlimited property can be very efficiently exploited to reduce both the speed and number of computations. The new method embeds an adaptive filter into multirate filters to reduce complexity and speed of computation.

We have applied the theoretical results obtained for the effects of multirate building blocks on stationary inputs to the adaptive identification scheme above and shown that the optimal filter is a matrix filter. We have shown through simulations that for a practical setup, a scalar adaptive filter performs almost as well if the fixed filters in the scheme are designed to have good stopband attenuation.

In a practical implementation of adaptive algorithms, computational noise is of concern. Most of the current analysis focuses on deriving the worst case upper bound on the roundoff errors. We analyze some basic signal processing steps by introducing a statistical flavor to it. This analysis answers questions such as "what is a typical value of the roundoff error?" In particular, for the case of dot product computation, we obtain expressions for the roundoff noise variance for the floating point case, and compare the results with the fixed point noise roundoff noise analysis. We also perform error variance analysis of Givens rotation

and Householder transformation. These two algorithms are used in the upper triangularization of matrices. We have compared the results obtained for these cases and shown that error variance for the Householder case is lower, meaning that the Householder transformation adds lower roundoff error "on an average".

We also address the problem of bandlimited extrapolation of discrete-time signals. We have explained why the term "best solution" does not have a unique answer. Several new techniques for bandlimited extrapolation of discrete-time segments are explored. These methods apply to a wide range of situations (including multiple-burst interpolation of multiband signals). A closed form expression for the optimal solution (for a given value of the energy of extrapolated sequence) has been obtained and evaluated for various values of the final energy. The various methods are compared on the basis of out-of-band energy of the extrapolated signal, total energy of the extrapolated signal (in relation to that of the given segment), and numerical robustness.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Adaptive filtering is a branch of signal processing which deals with the design and implementation of filters and algorithms which "learn" about the problem at hand as new data comes in. It is suitable for applications where very little is known about properties of the input or where the systems involved vary with time. In a typical adaptive filtering implementation, an error signal is computed to judge the performance of the adaptation scheme. The filter parameters are adjusted so as to minimize some appropriate measure of the error. In a real time signal processing application, the adaptive computations have to be performed in the time spacing between two consecutive input samples. For the emerging high-speed applications such as ghost image cancellation for High Definition Television (HDTV) where the sampling rate is about 128 MHz (Jur91), this means that the complexity of the adaptive algorithm should be as low as possible. The research effort to meet the high-speed requirements can be broadly classifed in two main areas:

1. Developing adaptive algorithms which are low in complexity, can be implemented efficiently and are numerically robust. This involves issues such as the number of computations that need to be performed, the speed at which these computations are performed in relation to the speed of the input signal, whether the algorithm can be implemented in parallel and so on.
2. Developing faster hardware and better ways of implementing the algorithms. This includes development of signal processing integreted circuits (ICs) and

special purpose signal processing hardware.

It is however true that both of these aspects of real-time implementation go hand in hand. In this thesis, we will deal with the first apsect: Developing algorithms that are computationally efficient. We will also address the issue of roundoff errors associated with the implementation of these algorithms. We present an adaptive filtering scheme which makes use of multirate filtering concepts to reduce the computational complexity. The issues involved in the analysis of such systems for optimal filtering solution etc. are also discussed.

Multirate signal processing has emerged as an answer to the high-speed signal processing requirements. In this technique, signals are split into M sub-bands and computations are performed in each sub-band at a reduced speed. Apart from the speed reduction advantage, this technique offers benefits such as data compression, data encryption etc. An excellent overview of multirate signal processing and related issues is presented in [Vai90]. The existing multirate signal processing theory deals with design and implementation of filtering schemes for splitting the input in sub-bands, processing the signals in sub-bands, reconstructing the processed signals and various issues involved with this. These multirate filters are not designed to sense the changing input properties. Hence it is important to investigate if the advantages of multirate filtering can be used in adaptive filtering to improve the performance of these algorithms.

Recently, there have been efforts to incorporate adaptive filtering techniques with multirate signal processing [Gil88], [Sat90b], [Sat91a]. While application of such techniques to real life problems has been shown to improve performance, there is still the need for a theoretical framework to analyze these multirate adaptive filtering methods. In this thesis, we will address this issue. We will study the effects of multirate building blocks on wide sense stationary (WSS) processes as they are passed through various multirate filtering configurations. These results

can in turn be used to derive optimal filtering solutions for filtering setups involving multirate building blocks.

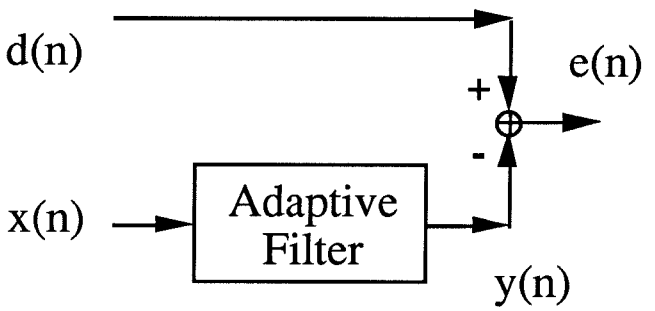
Any algorithm implemented in practice is associated with quantization errors. Signal processing hardware can represent all signals only with a finite bit precision. It is therefore important to study the effect of roundoff errors on the performance of an algorithm. Such analysis is even more crucial for adaptive filtering algorithms because of the iterative nature of the computations. One would like to perform such an analysis beforehand to be able to determine if the algorithm will converge or not. In real-time applications it is important to make sure that the algorithm does not diverge. The instability problems can gravely limit the usefulness of the algorithms in practice. Another important aspect of the analysis is determining the correctness of the solution. One would like to get a quantitative expression for the effects of roundoff errors on the final solution. Researchers have performed roundoff error analysis of various algorithms on the lines mentioned above [Gol89], [Wil65]. Methods have been suggested to remove numerical unstabilities present in certain algorithms like the RLS or the fast RLS algorithm [Slo88].

Such an analysis is inadequate to answer questions about "typical" behavior of algorithms, i.e., typical values of errors incurred. To put it in other words, one would like to know the expected value and variance of roundoff error at a particular stage. This gives us an idea about the energy of roundoff error generated at a certain stage in the computations. In this thesis and in [Sat90a], we have presented some basic results for such an analysis of signal processing algorithms.

1.1 Introduction To Adaptive Filtering

Some signal processing applications arise in situations where very little information about frequency characteristics or stationarity of signals is available. In such cases, it is difficult to design filters a priori to achieve desired results. Instead, one would

want to design filters in such a way that they *adapt* themselves to the changing properties of the input signals. Systems that vary with time, or systems where very little is known about input-output properties are a few such examples where these *adaptive filters* are suitable. Consider a typical adaptive filtering setup as shown in Fig. 1.1.



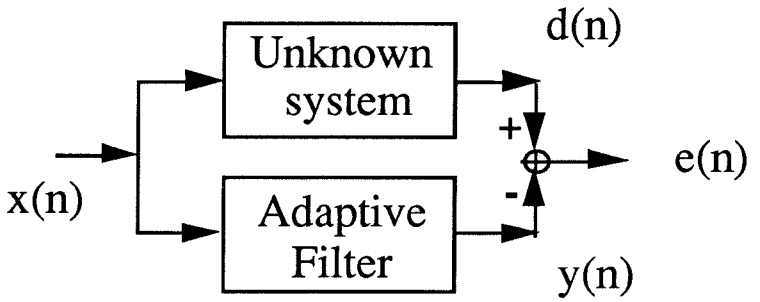
x(n) : input to the adaptive filter
y(n) : output of the adaptive filter
d(n) : desired signal
e(n) : error signal

Fig. 1.1 Adaptive filtering setup.

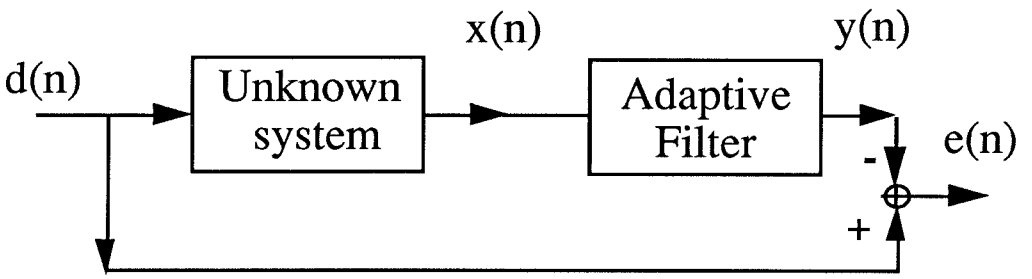
The adaptive filter is adapted so that the output $y(n)$ to input $x(n)$ matches the desired signal $d(n)$ as closely as possible. This is done by minimizing energy of the error $e(n)$. The adaptive filter can have one of the many possible structures, for example an FIR transversal filter or a lattice structure etc. However one should keep in mind that an adaptive filter is not a time-invariant system hence "the adaptive filter is an FIR filter" means that the adaptive filter is implemented in a non-recursive manner.

Depending upon the relation between $x(n)$ and $d(n)$, the adaptive filter can

be employed to perform a variety of applications. The two most common configurations used in this context are shown in Fig. 1.2 (a) and (b). In the cancellation



(a)



(b)

Fig. 1.2 Schemes for (a) adaptive cancellation
(b) adaptive equalization.

configuration, the adaptive filter is employed in parallel to an unknown channel to match its output. This is useful in the identification of unknown systems, or echo cancellation etc. [Hay86]. In the equalizer configuration, the adaptive filter is

used in cascade with the unknown channel so that the output of the adaptive filter matches the input to the unknown system. This configuration is to compensate the unwanted distortion of a channel.

Adaptive algorithms are the exact computational steps used to update coefficients of the adaptive filter from time n to time $(n+1)$. These algorithms are of two types:

(1) Stochastic algorithms: those which assume some statistical model for the input $x(n)$. These algorithms try to minimize the error variance $E[e^2(n)]$. The LMS algorithm is an example of this type of algorithm.

(2) Those algorithms where no assumptions about the input statistics are made. These algorithms are called deterministic or least-squares-type algorithms. The objective function E used here is a windowed (weighted) average of error squares, given by $E = \sum_{i=0}^{n} \lambda^{n-i} e^2(i)$ where n is the current time, and λ is the forgetting factor. The RLS algorithm belongs to this category.

Examples Of Adaptive Algorithms

We present the LMS and RLS algorithms here as they are mentioned throughout the thesis. The exact derivation and properties of these algorithms are available in standard adaptive filtering textbooks such as [Hay86], [Wid85], [Hon84]. The adaptive filter is assumed to have an FIR-type structure. The filter coefficients are represented by an $N \times 1$ vector $\mathbf{a}(n)$. The vector $\mathbf{x}(n) = \big(x(n)\ x(n-1)\ \ldots x(n-N+1)\big)^T$ is the $N \times 1$ data vector.

LMS algorithm

This algorithm uses a one-term approximation of the gradient of the objective function with respect to $\mathbf{a}(n)$. The updating is performed as follows

$$y(n) = \mathbf{a}^T(n)\mathbf{x} \tag{1.1a}$$

$$e(n) = d(n) - y(n) \tag{1.1b}$$

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \mu e(n)\mathbf{x}(n) \tag{1.1c}$$

The constant μ is called the step size, and is crucial in determining the speed and stability of the algorithm.

RLS Algorithm

We use the following steps to update $\mathbf{a}(n)$.

$$\mathbf{k}(n+1) = \frac{\lambda^{-1}\mathbf{P}(n)\ \mathbf{x}(n+1)}{1+\lambda^{-1}\mathbf{x}^T(n+1)\mathbf{P}(n)\ \mathbf{x}(n+1)} \tag{1.2a}$$

$$e(n+1) = d(n+1) - \mathbf{a}^T(n)\mathbf{x}(n+1) \tag{1.2b}$$

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \mathbf{k}(n+1)\ e(n+1) \tag{1.2c}$$

$$\mathbf{P}(n+1) = \lambda^{-1}\mathbf{P}(n) - \lambda^{-1}\mathbf{k}(n+1)\mathbf{x}^T(n+1)\mathbf{P}(n) \tag{1.2d}$$

The vector $\mathbf{k}(n)$ is called Kalman vector. It can be shown that the matrix $\mathbf{P}(n)$ approximates the inverse of the input data autocorrelation matrix.

Outline of Thesis

In Chapter 2, we study the problem of bandlimited extrapolation of finite length sequences. We discuss similarities and differences between discrete-time and continuous-time extrapolation problems. Since any extrapolation result that we compute is finite in time, it can not be exactly bandlimited. We have discussed some methods to make the extrapolation result as bandlimited as possible. The natural error criteria to use is the out-of-band energy of the extrapolated sequence. An expression for the theoretically optimal solution is derived. Results of other extrapolation schemes are compared with the optimal solution. Extrapolation methods based on FIR and IIR filtering techniques or the RLS and fast RLS algorithms are proposed. Simulations are included for each method. All of these methods implicitly involve the inversion of a matrix. Depending on the method,

the size of this matrix is equal to either the length of the known signal or the length of the unknown part. A combined interpolation/extrapolation problem where the missing samples are interlaced with the known samples is also discussed. A method based on the multichannel RLS algorithm is proposed for this problem. All of the proposed methods are compared on the basis of their noise performance. The noise performance is judged by the difference in the extrapolation result when the given segment is perturbed with some noise.

Analysis of the effects of multirate filters on statistical inputs is performed in Chapter 3. We review some useful basic definitions. Using these, we derive conditions to maintain the WSS nature at the output of some multirate building blocks such as an interpolator, a decimator and a time-domain modulator. Similar results are derived subsequently for more complicated filters. As an application, we mention a new multirate adaptive filtering scheme for the identification of bandlimited channels. This scheme involves use of multirate building blocks in an adaptive filtering setup. An analysis of this scheme for the Wiener filtering solution gives the result that the optimal filter is a linear periodically time varying (LPTV) system. This forms the topic of Chapter 4.

In Chapter 4, first we justify the importance of studying the bandlimited channel identification problem. Some practical applications involving bandlimited channels are explained. The new method is then derived using the fact that since a signal bandlimited in the discrete-time domain is an oversampled signal, we can discard some of its samples without losing any information. The Wiener filter solution for the new method is shown to be a matrix filter in general. However, we have shown through simulations that a scalar adaptive filter still matches the performance of a matrix adaptive filter if the lowpass filters in the scheme have good stopband attenutation.

The topic of floating point error variance analysis is discussed in Chapter 5.

We present a discussion stressing the importance of such an analysis. The three algorithms considered in this chapter are dot product computation, Givens Rotation (GR), and Householder Transformation (HT). These basic algorithms are used repetitively in a variety of more complex signal processing algorithms. Expressions for error variance and signal-to-noise ratio (SNR) for dot product computation are derived. After a brief discussion about QR-factorization of matrices, error variance analysis results for GR and HT are derived. These results are compared to show that HT offers an 8 DB improvement in SNR. All the results derived in this chapter are supported by simulations performed on a floating point computer.

Notations used in the Thesis

The letter Ω represents the frequency variable for the continuous-time case. The units are radians/sec. The variable ω is used to denote discrete-time frequency (radians). The frequency response of a discrete-time transfer function $H(z)$ is expressed as $H(e^{j\omega}) = |H(e^{j\omega})|e^{j\phi(\omega)}$, where $|H(e^{j\omega})|$ is the magnitude response and $\phi(\omega)$ is the phase response. In all the plots, we use the "normalized frequency" which is $f = \omega/2\pi$. Unless mentioned otherwise, the impulse response coefficients of $H(z)$ are assumed to be real. If $H(z)$ has real coefficients, then $|H(e^{j\omega})|$ is always plotted for $0 \leq f \leq 0.5$ (because the magnitude response is symmetric with respect to $\omega = 0$).

Bold-faced letters indicate vectors and matrices (except for the letter $\mathbf{u}$ which is a scalar and represents unit roundoff of a machine). Superscript T and † denote transposition and transposed conjugation respectively. The operator $E[\,]$ represents the expected value operator and $fl(\,)$ represents floating point quantizer. Finally, we list all the abbreviations used in the thesis.

Abbreviations

MNLS: Minimum Norm Least Squares

LMS: Least Mean Squares

RLS: Recursive Least Squares

LS: Least Squares

FIR: Finite Impulse Response

IIR: Infinite Impulse Response

σ-*BL:* Bandlimited to the frequency σ

GR: Givens Roations

HT: Householder Transformation

FRLS: Fast Recursive Least Squares

SNR: Signal-to-Noise Ratio

WSS: Wide Sense Stationary

CWSS: Cyclo Wide Sense Stationary

LTI: Linear Time Invariant

LPTV: Linear Periodically Time Varying

gcd: greatest common divisor

MPU: Multiplication Per Unit input time

APU: Additions Per Unit input time

IFIR: Interpolatd Finite Impulse Response

Chapter 2

Fast Methods for Bandlimited Extrapolation of Finite Length Sequences

An important problem in signal processing is bandlimited extrapolation, i.e., finding a bandlimited signal by extrapolation of a finite-length segment of the signal. This problem has been considered in the continuous-time domain [Pap75], [Sab78], [Cad79], and in the discrete-time domain [Jai81], [Hay83], [Sul84] separately, and has also been studied in a unified framework [San83] which deals with four types of signals (continuous and discrete; periodic and aperiodic).

First consider the continuous-time case. Let $x_a(t)$ be the finite duration segment to be extrapolated with support in the region $0 \leq t \leq T$. We wish to obtain a signal $y_a(t)$ bandlimited to $-\sigma \leq \Omega \leq \sigma$ (i.e., a σ-BL signal [Pap77]) such that $y_a(t) = x_a(t)$ for $0 \leq t \leq T$. Given the segment $x_a(t)$, such an extrapolation may or may not exist, i.e., an arbitrary $x_a(t)$ may not be a segment of a σ-BL waveform. For example, if $x_a(t)$ is a nonzero constant in $0 \leq t \leq T$, then $y_a(t)$ does not exist. For a given $x_a(t)$, if $y_a(t)$ does exist, it is unique. Indeed if there were two σ-BL extrapolations $y_{a,1}(t)$ and $y_{a,2}(t)$ for the same $x_a(t)$, then their difference would be a σ-BL waveform which is identically zero in $0 \leq t \leq T$ which implies $y_{a,1}(t) = y_{a,2}(t)$.

Several algorithms have been proposed for the construction of the extrapolation $y_a(t)$ from $x_a(t)$ when it exists. A unified derivation of many of these algo-

rithms based on the theory of alternating projections has been given in [You78]. Moreover it is well-known [Pap75], [You78] that the extrapolation problem is ill-conditioned in the sense that if we add a small amount of noise to the given segment then it may not remain a bandlimited segment and the result of executing an extrapolation algorithm either does not converge or gives rise to waveforms that differ considerably from the noise-free case. Noise can also be introduced into an algorithm due to computational roundoff, which makes the continuous-time extrapolation problem fairly challenging.

For the case of discrete-time signals (which is the topic of this paper) there are some fundamental differences as compared to the continuous case. These are particularly evident from the results reported in [Jai81], [Hay83] (and do not conflict the unified treatment given in [San83]). First, given any finite-length segment $x(n)$, $0 \leq n \leq N-1$, there always exists a σ-BL extrapolation $y(n)$ with

$$y(n) = x(n), \quad 0 \leq n \leq N-1. \tag{2.1}$$

Second, this extrapolation is not unique. In fact there exist an infinite number of σ-BL extrapolations for the same segment. These statements are briefly reviewed in Sec. 2.3. One commonly used technique to obtain a unique answer would be to impose the constraint that the extrapolation $y(n)$ should have the smallest possible energy. Such a unique extrapolation has been obtained in [Jai81] by formulating the problem as a least-squares approximation problem and computing the mininum-norm least squares (MNLS) solution by use of the Moore-Penrose Pseudo Inverse technique [And79].

Even though the matrix involved in the above MNLS problem is nonsingular, it is ill-conditioned particularly for large segment size N. As a result, if the given segment is noisy, the extrapolated sequence tends to differ substantially from the ideal MNLS solution. One obvious technique to reduce this effect is to add a

constant to the diagonal elements of the ill-conditioned matrix. A second technique based on singular value decomposition is proposed in [Sul84]. In this technique an appropriate subset of singular values which are smaller than a threshold are set to zero, and the pseudo inverse (rather than the inverse) is computed.

In order to judge the performance of a particular extrapolation algorithm, one has to find meaningful performance measures. These measures should be weighed against the computational complexity and other cost factors in the implementation of the algorithm. In the continuous-time case, the difference between the original signal $y_o(t)$ and the extrapolated version $y(t)$ can be used to define a performance measure (such as the mean-square error). In the discrete-time case this is not meaningful because, for a given segment, an infinite number of "equally correct" solutions $y(n)$ exist. With the solution constrained to be of minimum-norm, it is still not meaningful to compare it with the "ideal" solution because no such solution exists. Accordingly, computational complexity and robustness appear to be the only remaining considerations. ‡ The tradeoff between these two is evident by comparing the method in [Jai81] with the more robust method in [Sul84] which involves the overhead of singular value decomposition.

It is a well-known fact that if a σ-BL signal $y(n)$ is passed through an ideal lowpass filter with impulse response $\sin(\sigma n)/\pi n$ then the output is equal to $y(n)$ (i.e., $y(n)$ is an eigenfunction of the system with unit eigenvalue). This is the key observation which leads to equation (39) in [Jai81]. In Sec. II of this report we shall take a different approach to discrete-time σ-BL extrapolation. This is based on two modifications of [Jai81]. First, we replace the ideal lowpass filter with a practical

‡ It is possible to find an extrapolation $y(n)$ such that its distance from another possible extrapolation $y_1(n)$ normalized by the norm of $y_1(n)$ (and maximized over all possible extrapolations $y_1(n)$) is minimized, as elaborated in [Kol78]. We shall not pursue this here.

filter. This filter can be FIR or IIR as the application demands. We obtain the extrapolation $y(n)$ as the output of such a filter in response to a finite length input. Both the input and the filter are causal so that the extrapolation $y(n)$ is causal. The input $u(n)$ is designed so that the segment $y(M), y(M+1), \ldots, y(M+N-1)$ of the output agrees with the given segment $x(0), x(1), \ldots x(N-1)$ to be extrapolated. Here M is a non-negative integer whose choice will be discussed later. So the extrapolation is two-sided as usual (with respect to the given segment) but is causal for convenience. If the filter is FIR, we obtain a finite-length extrapolation $y(n)$ (which is perhaps more practical than an infinitely long extrapolation). This is different from merely truncating the extrapolation obtained in [Jai81] because truncation would lead to effects associated with Gibbs phenomenon. Secondly, if we consider the extrapolated signal to be the output of an FIR filter, then several of the well-known techniques for efficient implementation of FIR filters (such as the multirate/multistage methods [Cro83], prefilter-equalizer method [Ada83], and the IFIR method [Neu84]) can be applied to improve the computational efficiency of the method as we shall see in Sec. 2.2. In particular if we consider the lowpass filter to be IIR, then the inherent computational advantage of IIR filters over FIR are available in the extrapolation problem.

With finite-cost filters we can obtain only approximately bandlimited extrapolations. In this paper, the term "σ-BL extrapolation" stands for an extrapolation which exactly agrees with the given segement of N samples even though it may be only approximately bandlimited for practical reasons.

2.1 The σ-BL Extrapolation Problem As A Filtering Problem

Consider a transfer function $H(z) = \sum_{n=0}^{\infty} h(n)z^{-n}$ which represents a causal stable lowpass transfer function with stopband edge equal to σ radians. Without

loss of generality, assume $h(0) \neq 0$; $H(z)$ could be FIR or IIR. Assume that the coefficients $h(n)$ are known. Our aim is to find a finite-length causal input $u(n)$ such that the causal output sequence $y(n)$ satisfies the property

$$y(M+n) = x(n), \quad 0 \leq n \leq N-1, \tag{2.2}$$

where $x(n), 0 \leq n \leq N-1$ is the given length N segment to be extrapolated. Here M is some non-negative integer to be determined.

The qualitative argument (which we shall treat more carefully) for using this scheme is as follows: $y(n)$ is the output of a lowpass filter so that its energy in the frequency range $\sigma \leq \omega \leq \pi$ is expected to be small. At the same time $u(n)$ is such that a set of designated samples of $y(n)$ agree with the given segment $x(n)$ so that $y(n)$ is a σ-BL extrapolation of $x(n)$ except for a shift by M.

The reasons why the above argument is "qualitative" are several. First, the input $u(n)$ which leads to satisfaction of the condition (2.2) may be such that the output $y(n)$ might be a poor extrapolation, i.e., its stopband energy normalized by the total energy may not be "small enough." Second, the resulting extrapolation might be such that the energy of the samples of $y(n)$ outside the range $M \leq n \leq N+M-1$ may substantially dominate the energy of the segment, so that the extrapolation is not useful any more. Later on we shall use these two considerations (among others) as factors that judge the extrapolation quality.

Continuing the above trend of thinking, suppose $M = 0$ so that we are interested in matching the first N samples $y(n)$, $0 \leq n \leq N-1$ with the N samples $x(n)$, $0 \leq n \leq N-1$. We are thus required to find $u(n)$ such that

$$\begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix} = \begin{bmatrix} h(0) & 0 & \dots & 0 \\ h(1) & h(0) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h(N-1) & h(N-2) & \dots & h(0) \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N-1) \end{bmatrix}. \tag{2.3}$$

With $h(0) \neq 0$ it is obvious that such an input can be found trivially. Note that the extrapolation is "one-sided" in the sense that the segment $x(n)$ of length N

has been extrapolated only to the right. This is unlike any of the other schemes reported in the literature. Moreover as we shall substantiate in Example 1, the quality of the extrapolation is not very good. Part of the reason for this is that for a good lowpass filter $h(0)$ is usually very small so that $u(n)$ has much higher energy than required. The integer M introduced in (2.2) helps us to overcome the problem as explained next. For arbitrary M, we can rewrite condition (2.2) in terms of $h(n)$ and $u(n)$ in the following form:

$$\begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix} =$$

$$\begin{bmatrix} h(M) & h(M-1) & \dots & 0 & \dots & 0 \\ h(M+1) & h(M) & \dots & h(0) & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h(M+N-1) & h(M+N-2) & \dots & \dots & \dots & h(0) \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(M+N-1) \end{bmatrix} \tag{2.4}$$

There are N equations here. The last N columns of the matrix are linearly independent so that for any $M > 0$ there exists an input $u(n)$ satisfying (2.4) (simply delay the solution $u(n)$ for (2.3) by M units). However, this is not the only possible solution. Let the $N \times (N+M)$ matrix in (2.4) be denoted by $\mathbf{A}_M$ and let the column vectors on the left and right sides of (2.4) be denoted respectively by $\mathbf{x}$ and $\mathbf{u}_M$ so that (2.4) can be abbreviated as

$$\mathbf{x} = \mathbf{A}_M \mathbf{u}_M. \tag{2.5}$$

Let $\mathbf{A}^{\#}$ denote the Moore-Penrose pseudo-inverse † of a matrix $\mathbf{A}$. We can then obtain a solution to (2.5) of the form

$$\hat{\mathbf{u}}_M = \mathbf{A}_M^{\#} \mathbf{x}. \tag{2.6}$$

† The Moore-Penrose pseudo-inverse is discussed in [And79], [Hay86], [Jai81]. We shall refer to this as just the *pseudo inverse* for the rest of this paper.

This solution will have the smallest norm (i.e., the energy $\hat{\mathbf{u}}_M^\dagger \hat{\mathbf{u}}_M$ will be the smallest) among all possible solutions because of minimum-norm least-squares (MNLS) property of this pseudoinverse [And79].

Next suppose we replace M with $M+1$ in (4) and seek a solution to $\mathbf{u}_{M+1}$. It is clear that $\mathbf{A}_{M+1} = [\,\mathbf{c} \quad \mathbf{A}_M\,]$ (where $\mathbf{c}$ is a new column) so that $\begin{bmatrix} 0 \\ \hat{\mathbf{u}}_M \end{bmatrix}$ is a valid solution. The minimum-norm solution $\hat{\mathbf{u}}_{M+1} = \mathbf{A}_{M+1}^{\#}\mathbf{x}$ therefore has norm less than (or equal to) that of $\hat{\mathbf{u}}_M$. Starting with $M = 0$ we can recursively solve for $\hat{\mathbf{u}}_{M+1}$ from $\hat{\mathbf{u}}_M$ by using recursive least squares (RLS) techniques as elaborated in standard texts such as [Hay86]. As M increases from $M = 0$, the energy of the input that produces the extrapolated output decreases (even though the length of the input increases), until we can perceive no further substantial decrease. We then stop the iteration and the value of M is then determined. The motivation for obtaining a minimum-energy input $\hat{u}(n)$ is explained as follows: since the energy of $\hat{u}(n)$ is minimum, the solution $\hat{u}(n)$ will tend *not* to have any components in the range $\sigma \leq \omega \leq \pi$ (because the filter would attentuate them and make them relatively unperceivable at the output). As a result, $y(n)$ which is a convolution of lowpass $\hat{u}(n)$ with lowpass $h(n)$ tends to be a good lowpass extrapolation.

Thus, there are two key steps in this method. The first is to identify an appropriate input $u(n)$ to the filter $H(z)$ such that the given segment $x(n)$ is matched by a portion of the output. The second step is to compute the output of the filter $H(z)$ in response to this input. We can use efficient implementation methods to reduce the cost of step 2. For example if $H(z)$ is IIR, the cost of step 2 can be negligible compared to that of step 1. This method should be contrasted with earlier methods [Jai81], where the energy of the *extrapolated signal* $y(n)$ was minimized by seeking a minimum-norm least squares solution to a different formulation.

Example 1 : In the rest of the chapter, unless otherwise mentioned, the extrapo-

lation problem to be considered is that of extrapolating a 15-point sequence to 55 points, with the original segment at the center of the extrapolated output. Hence $N = 15$ for this segment. We have included a time domain plot of the segment and the magnitude of its Fourier transform in Fig. 2.1(a) and Fig. 2.1(b) respectively. The problem is to extrapolate the given segment such that the extrapolated signal has small energy in the region $\sigma \leq \omega \leq \pi$ with $\sigma = \pi/3$. To compare relative performance of different algorithms, we consider the following two quantities as performance criteria :

$$\text{Time Domain Energy Ratio(TDE)} = \frac{\text{Energy of the original segment}}{\text{Total energy of the extrapolated signal}} \tag{2.7}$$

$$\text{Stopband Energy Ratio(SE)} = \frac{\text{Stopband energy of the extrapolated signal}}{\text{Total energy of the extrapolated signal}} \tag{2.8}$$

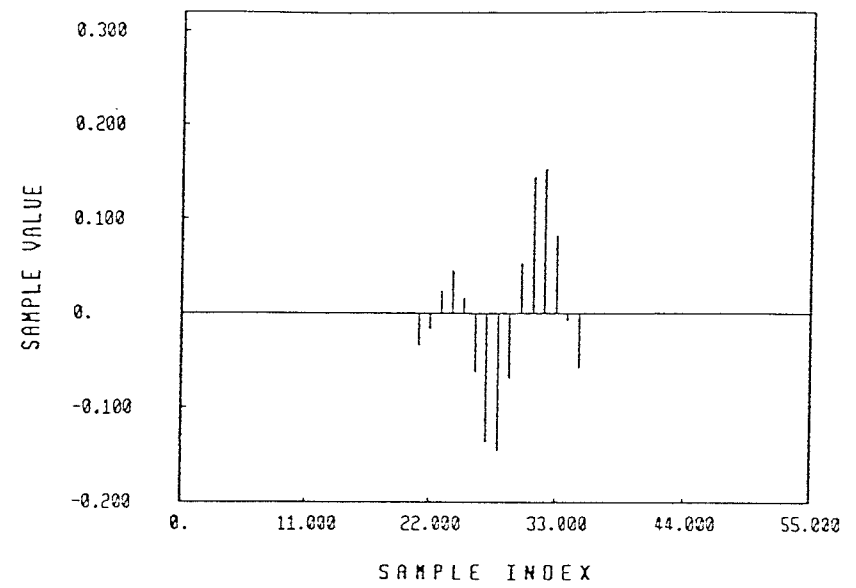
Roughly speaking, a good method should have large TDE and small SE. We will first mention the results of running the extrapolation algorithm with an IIR filter $H(z)$. Then we describe the results obtained with FIR filter and compare it with the pseudoinverse technique as suggested in [Jai81].

Extrapolation through IIR filter

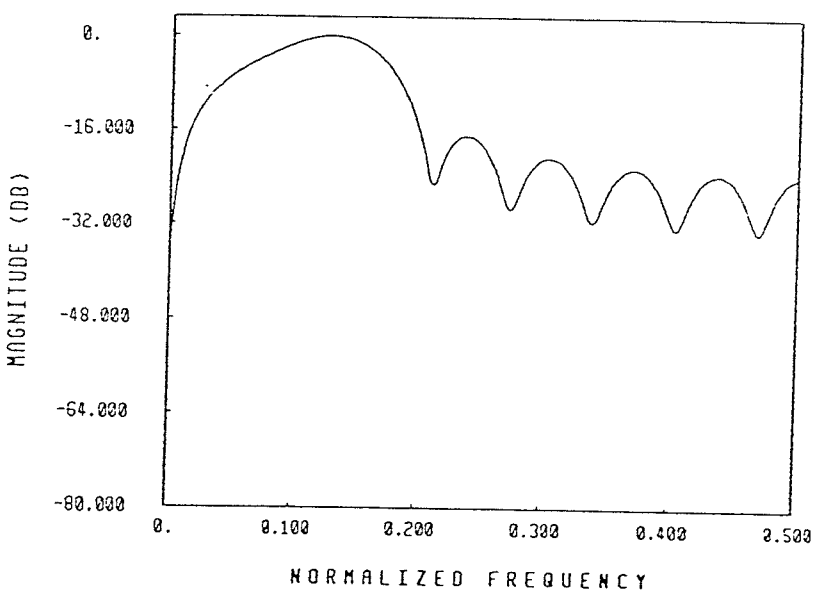
IIR filters are known for their increased computational efficiency as compared to FIR filters. In particular there exists a class of transfer functions [Vai86] which can be implemented as a sum of two allpass filters. For example, it is well-known that a fifth order elliptic filter $H(z)$ can be written in this form i.e.,

$$H(z) = \frac{A_0(z) + A_1(z)}{2} \tag{2.9}$$

where $A_0(z)$ and $A_1(z)$ are allpass filters of orders three and two respectively. These allpass filters can be implemented with three and two multipliers respectively so that we have a total of only five multipliers.



(a)



(b)

Fig. 2.1 The original 15-point sequence (a) time domain (b) magnitude of the Fourier transform.

It is also known that elliptic filters have all their zeros on the unit circle. The presence of these zeros makes the impulse response matrices in (2.3), (2.4) ill-conditioned. The condition number of these matrices can be improved by using a modified form of elliptic response in which the zeros are positioned slightly away from the unit-circle.

In order to do this, recall the mechanism by which (2.9) works. We have $A_0(e^{j\omega}) = e^{j\phi_0(\omega)}$ and $A_1(e^{j\omega}) = e^{j\phi_1(\omega)}$. In the passband the phases $\phi_0(\omega)$ and $\phi_1(\omega)$ are aligned so that the magnitude of (2.9) is close to unity. In the stop-band these phases are offest by about π so that the magnitude of (2.9) is very small. In particular, at the unit-circle zeros of $H(z)$, the phases $\phi_0(\omega)$ and $\phi_1(\omega)$ differ exactly by (an odd integral multiple of) π.

If we wish to "lift" the stop-band response away from zero, this can be accomplished [Ans86], [Koi89] by modifying (2.9) into

$$H(z) = \frac{\cos\theta A_0(z) + \sin\theta A_1(z)}{\cos\theta + \sin\theta} \tag{2.10a}$$

with $0 \leq \theta \leq \pi/2$. Now the maximum passband response is still unity, but the minimum value of $|H(e^{j\omega})|$ in the stopband would be $|(\cos\theta - \sin\theta)/(\cos\theta + \sin\theta)|$. We shall refer to this as a raised filter.

The IIR filter used for our example is a raised elliptic filter with

$$A_0(z) = \frac{-0.7136 + 2.044z^{-1} - 2.23z^{-2} + z^{-3}}{1.0 - 2.23z^{-1} + 2.044z^{-2} - 0.7136z^{-3}}$$
$$A_1(z) = \frac{0.7274 - 1.5139z^{-1} + z^{-2}}{1.0 - 1.5139z^{-1} + 0.7274z^{-2}} \tag{2.10b}$$

with $\theta = 45.1$ degrees. The frequency response of the filter is shown in Fig. 2.2. Since we want to extrapolate given 15 points to 55 points, we have to add 20 points before and 20 points after the given segment, so that $M = 20$. So the final input $u(n)$ to $H(z)$ will be of length $M + N = 35$. Total number of pseudoinverse update steps is $M = 20$.

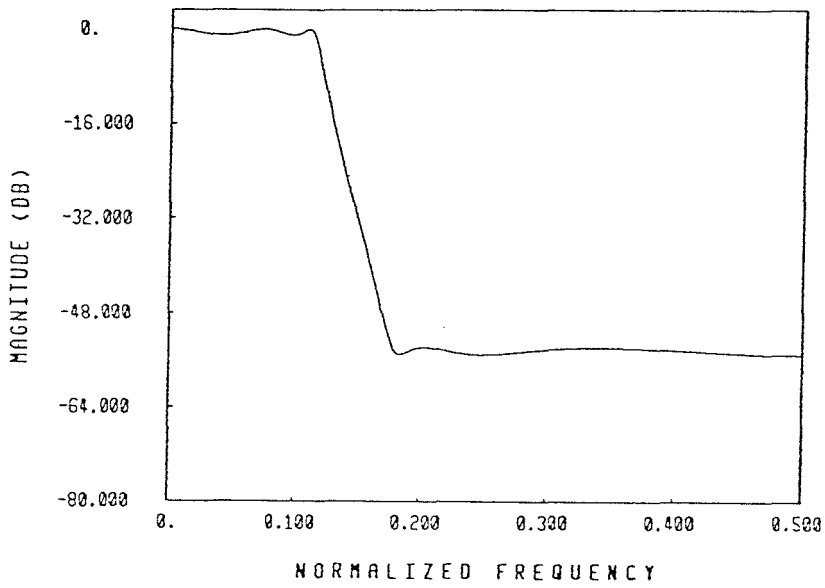


Fig. 2.2 Raised elliptic filter used for the extrapolation.

Since theoretically the IIR filter will produce an extrapolation output which is infinite in length, to get a length 55 extrapolation, we have to use some window function. The window used in this example is an exponentially decaying window which has magnitude unity at the time indices of the given segment, and decays out exponentially on both sides of the given segment. Fig. 2.3(a) shows the extrapolated signal after completion of the update procedure. Fig. 2.3(b) shows the magnitude of its Fourier transform.

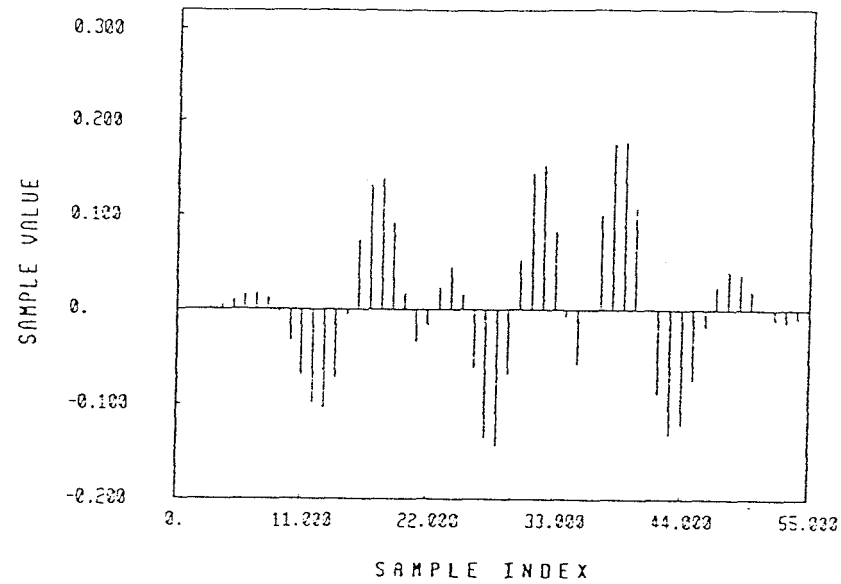
Fig. 2.4(a) and Fig. 2.4(b) show the corresponding graphs for the extrapolation obtained just by the inversion of the triangular matrix obtained at the beginning. Note that this extrapolation is of very poor quality because the extrapolated points greatly dominate the original segment. In fact, the original segment (the first 15 points in Fig. 2.4(a)), can not even be seen in the plot!

Since the input to filter at each update has a minimum norm which is non-increasing, it is of interest to see how the norm of the input decreases with iterations. This is plotted in Fig. 2.5. A plot of how the SE parameter of the input signal changes as iterations progress is shown in Fig. 2.6. This supports the conjecture that the input starts to look more and more lowpass as iterations progress. The Fourier transform magnitude of the final input $\hat{u}(n)$ is shown in Fig. 2.7. In Fig. 2.8, we have plotted energy of the output signal versus the iteration number to see how the decrease in input-norm affects the output-norm. Fig. 2.9 shows plot of output SE as iterations progress.

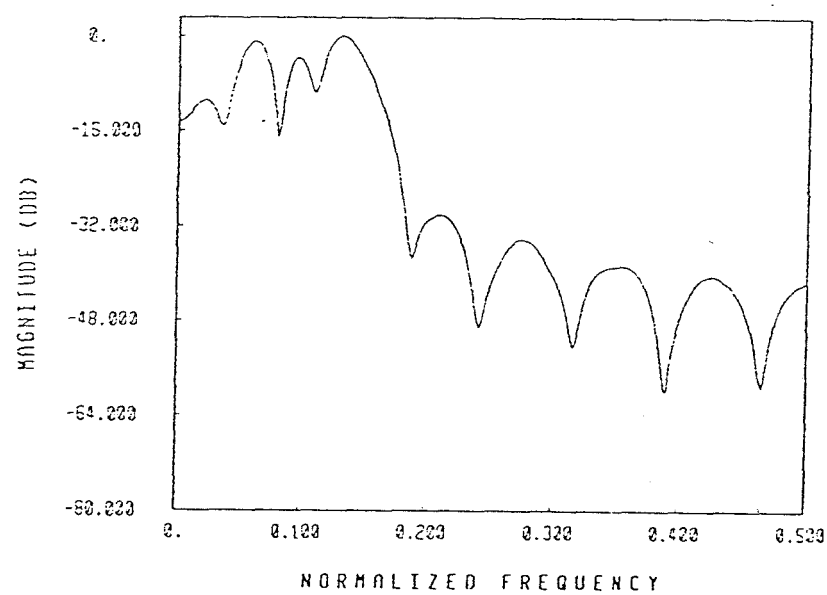
In Table 2.6 the condition numbers are compared for the traditional and raised elliptic filters. For a given attenuation (50 dB) it is clear that the raised filter has a smaller condition number (by a factor of about 3). The magnitude responses of the relevant filters are shown in Figs 2.44, 2.45.

Extrapolation through FIR filter

We can use the method described above to carry out extrapolation with the

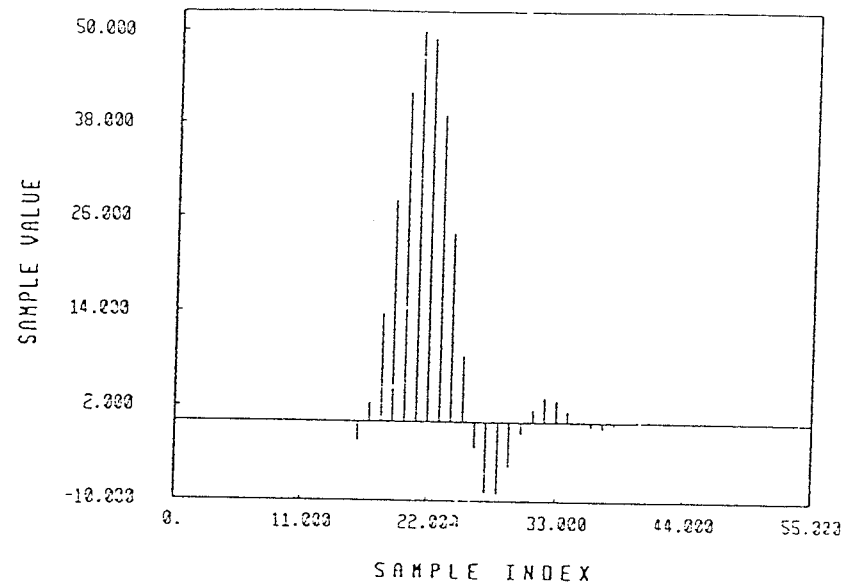


(a)

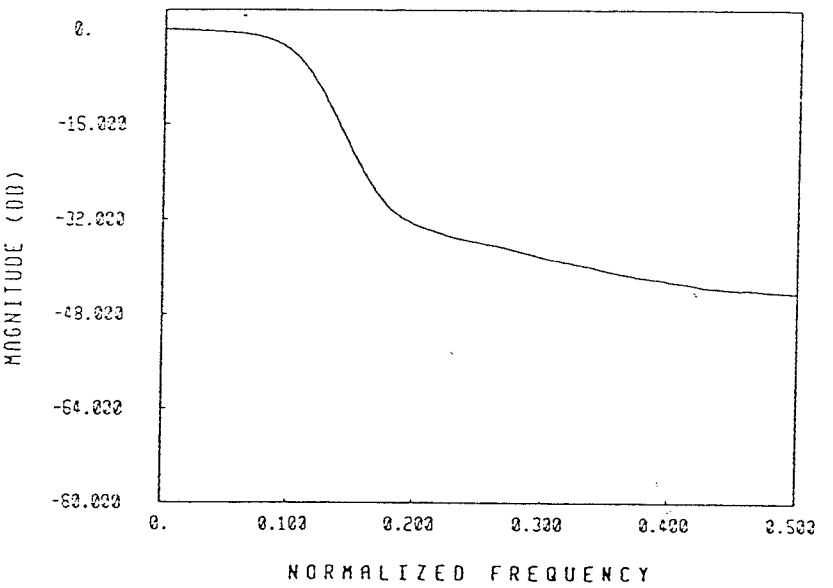


(b)

Fig. 2.3 IIR extrapolation method. The final extrapolation (a) time domain (b) magnitude of the Fourier transform.



(a)



(b)

Fig. 2.4 IIR extrapolation method. The extrapolation result at the 0^{th} iteration (a) time domain (b) the magnitude of the Fourier transform.

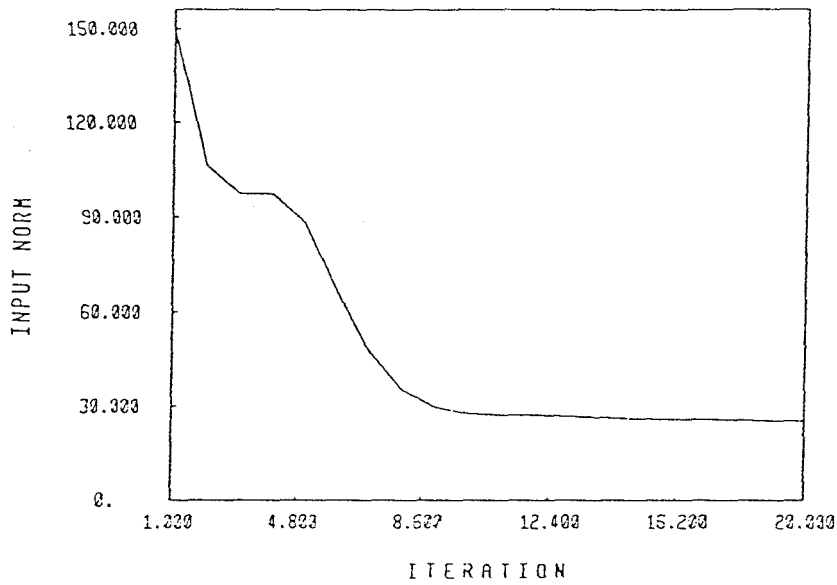


Fig. 2.5 IIR extrapolation method. Input norm v/s iterations.

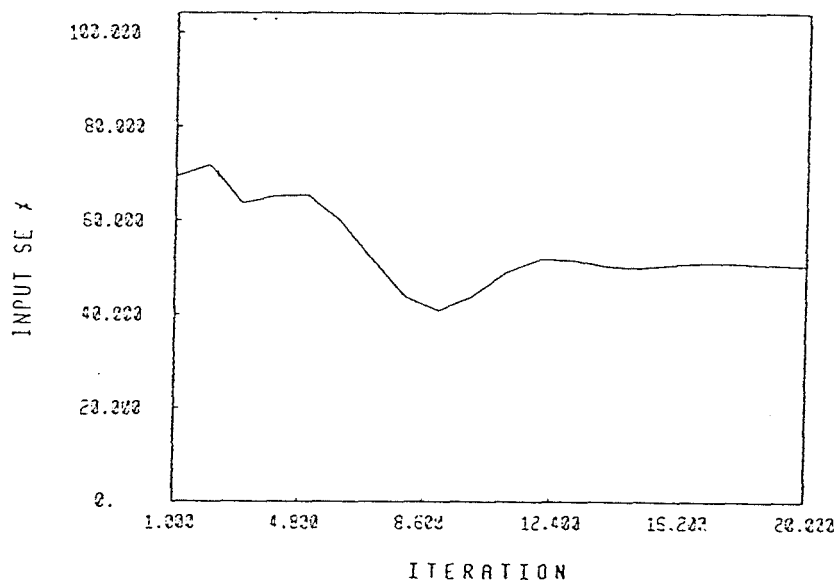


Fig. 2.6 IIR extrapolation method. Stopband energy percent (SE) v/s iterations.

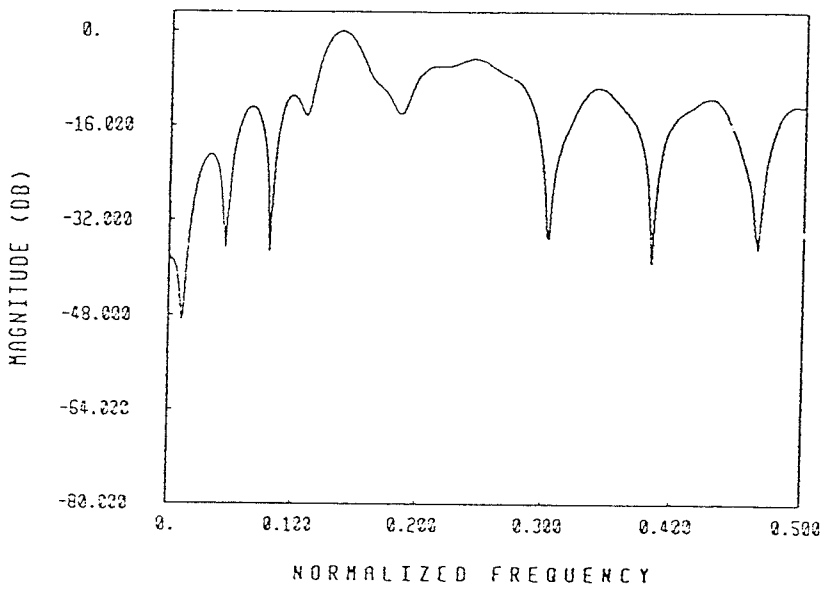


Fig. 2.7 IIR extrapolation method. Magnitude of the Fourier transform of the final input.

impulse response $h(n)$ corresponding to an FIR filter. There are two ways one can set up this extrapolation algorithm. We can either start with a causal FIR filter and corresponding lower triangular matrix as in Eq. (2.3) and add columns or start with a non-causal FIR filter and a symmetric Toeplitz matrix with 0^{th} element of the impulse response $h(0)$ as the diagonal element . In the second method, the square matrix will have entries different from the matrix in eq. (2.3) . The square matrix will not be lower triangular but will be a full matrix, the elements in the upper triangle being $h(-1), h(-2)$ etc. The first method is anologous to the IIR method. The second method is similar to the Pseudoinverse Extrapolation Method [Jai81] because the square matrix inverted at the beginning has a similar structure and interpretation.

FIR Method 1

The FIR filter used for this example has passband edge 0.28π and stopband edge 0.332π and is designed using the McClellan-Parks program [McC73]. It is easy to see that since we need to extrapolate 15 points to 55 points, the length of the filter is 21. The frequency response of the filter is shown in Fig. 2.10. The total number of multipliers needed for implementation is 10. Fig. 2.11(a) shows the extrapolation obtained by using the first FIR method and Fig. 2.11(b) shows the magnitude of its Fourier transform. Fig. 2.12(a) shows the result of extrapolation for the first iteration (i.e. by inverting the lower triangular matrix only). Fig. 2.12(b) shows the magnitude of its Fourier transform. Once again Fig. 2.12(a) represents a poor extrapolation because the extrapolated samples substantially dominate the original signal. Fig. 2.13 shows how the norm of the input changes with iterations and Fig. 2.14 shows plot of the input SE v/s iterations. It can be seen that the stopband energy of the input decreases as iterations progress. Fig. 2.15 shows the Fourier transform of the final input. Fig. 2.16 shows the effect of iterations on the output energy. Fig. 2.17 shows how the stopband energy of the

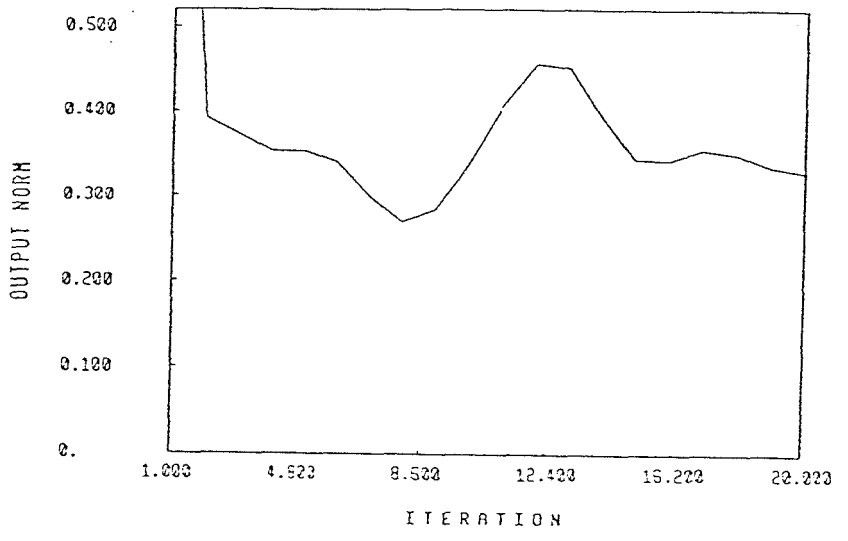


Fig. 2.8 IIR extrapolation method. The energy of extrapolation result v/s iterations.

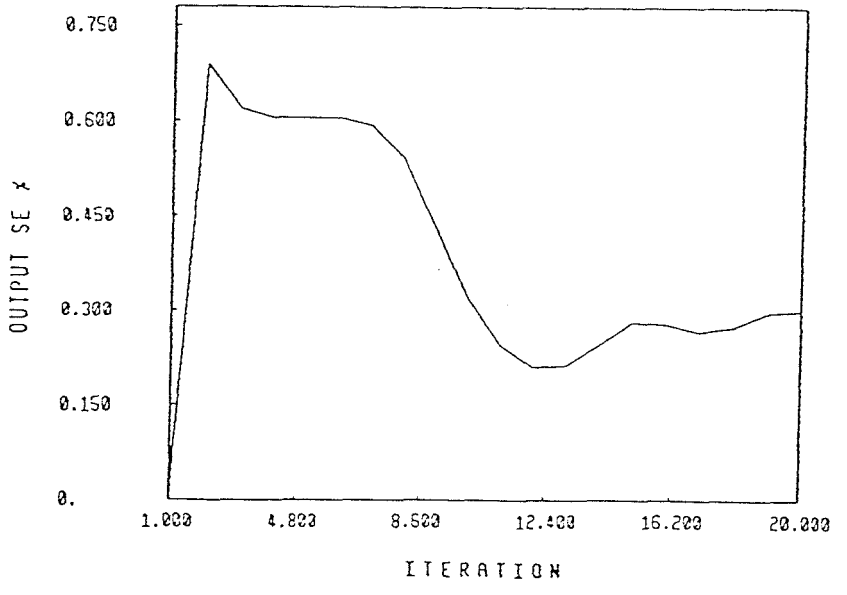


Fig. 2.9 IIR extrapolation method. Stopband energy percent (SE) of extrapolation result v/s iterations.

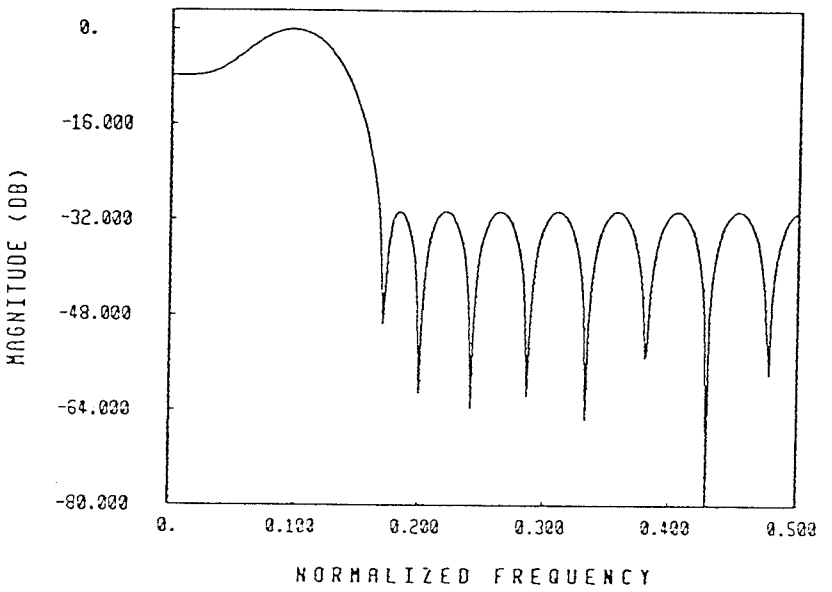
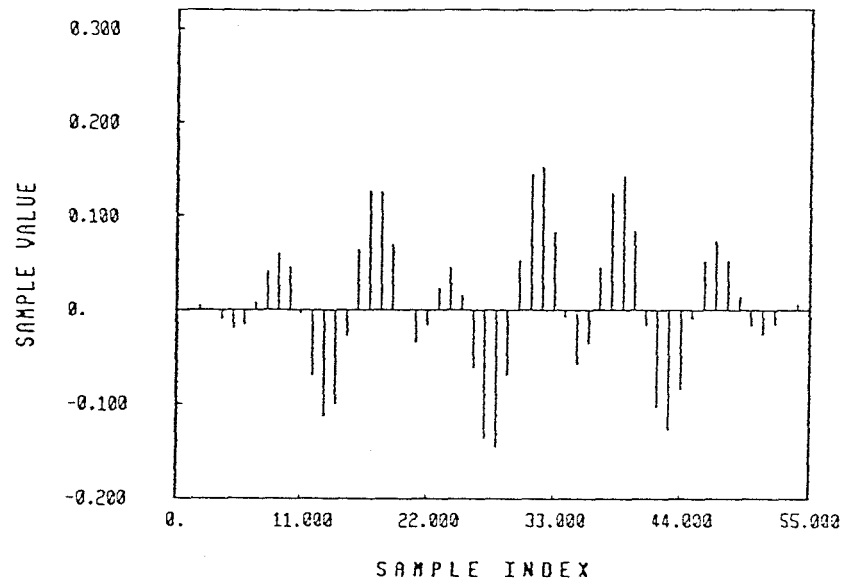
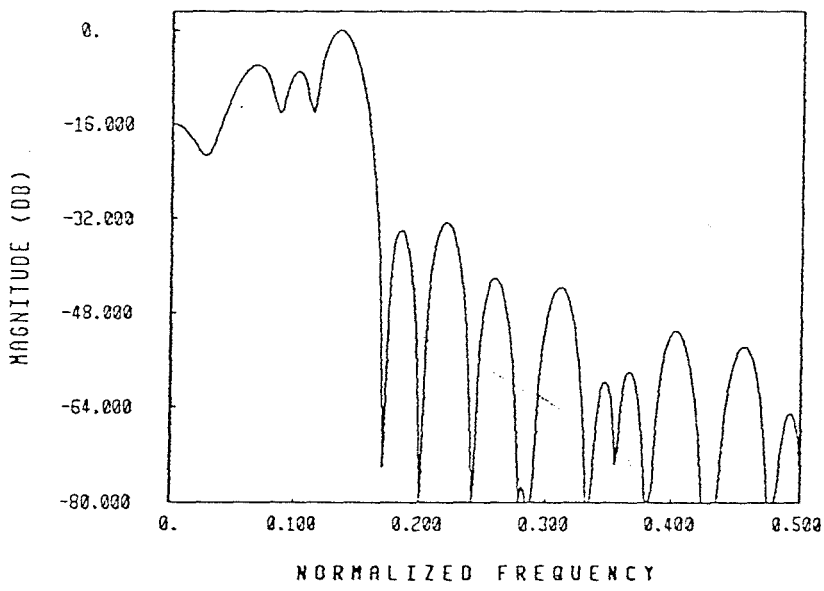


Fig. 2.10 FIR extrapolation method 1. Frequency response of the input filter.



(a)



(b)

Fig. 2.11 FIR extrapolation method 1. The final extrapolation (a) time domain (b) magnitude of the Fourier transform.

output changes with iterations.

FIR Method 2

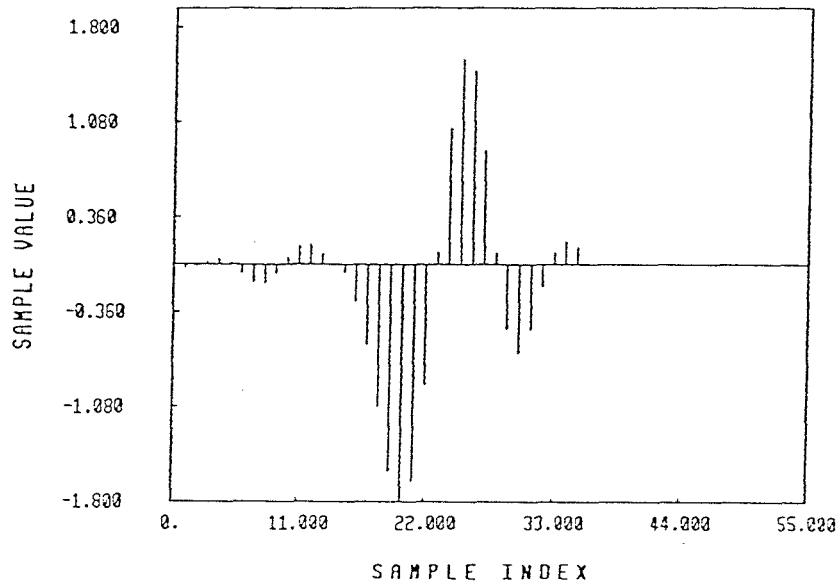
For the second method, after inverting the Toeplitz matrix and getting an input, the only motivation to carry out the iterations is that the input norm decreases and we hope that the input starts looking more and more like a lowpass signal. However, performing iterations like this produces longer and longer input. This in turn means that we start getting more output points than desired. We have chosen not to perform the iteration in this example. Hence this method will be similar to the pseudoinverse method [Jai81]. The only difference is that instead of using the ideal lowpass filter as suggested in [Jai81], we can use any lowpass filter here. In this particular example, we have used the impulse response of an IFIR filter [Neu84]. The reason to choose an IFIR is that it can be implemented with a lesser number of multipliers. This does not change the computational cost of the algorithm, but makes the implementation of the filter more efficient. The IFIR has a length of 41. It is implemented as

$$H(z) = G(z^2)F(z) \tag{2.11}$$

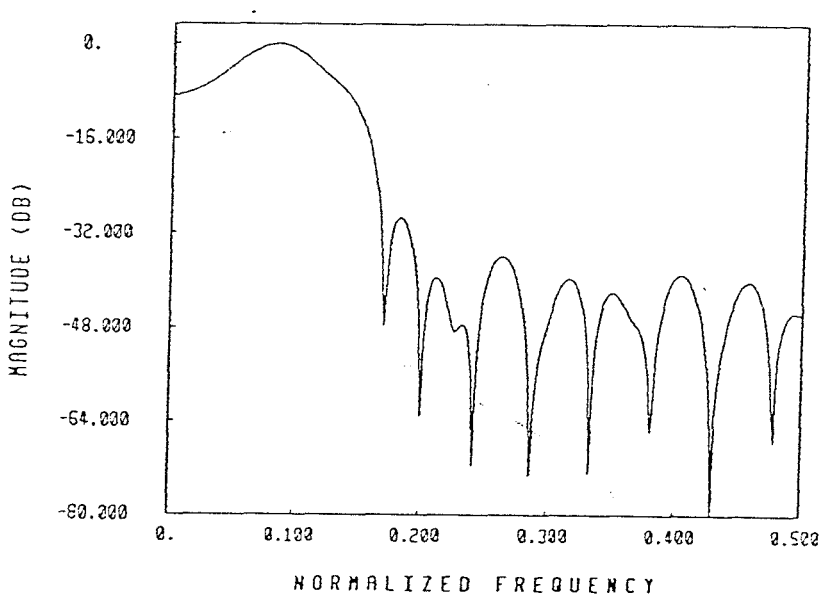
where $G(z)$ and $F(z)$ are lowpass filters with cutoff $2\pi/3$ and lengths 15 and 13 respectively. Hence the number of multipliers needed for the implementation is 7+6 = 13 instead of 20. Fig. 2.18 shows the frequency response of $H(z)$. Figs. 2.19(a) and (b) show the time and the frequency domain plots of extrapolated output obtained by inverting the square matrix.

Extrapolation Through Pseudoinverse Extrapolation Filter [Jai81]

In their paper on extrapolation algorithms, Jain and Ranganath mention an extrapolation method, which is very similar to the FIR method we have described. We have included the results of applying this method to the extrapolation example above. The ideal lowpass filter matrix that is suggested in the method was replaced by a Kaiser windowed lowpass matrix to reduce errors due to rectangular



(a)



(b)

Fig. 2.12 FIR extrapolation method 1. Extrapolation result at the 0^{th} iteration (a) time domain (b) magnitude of the Fourier transform.

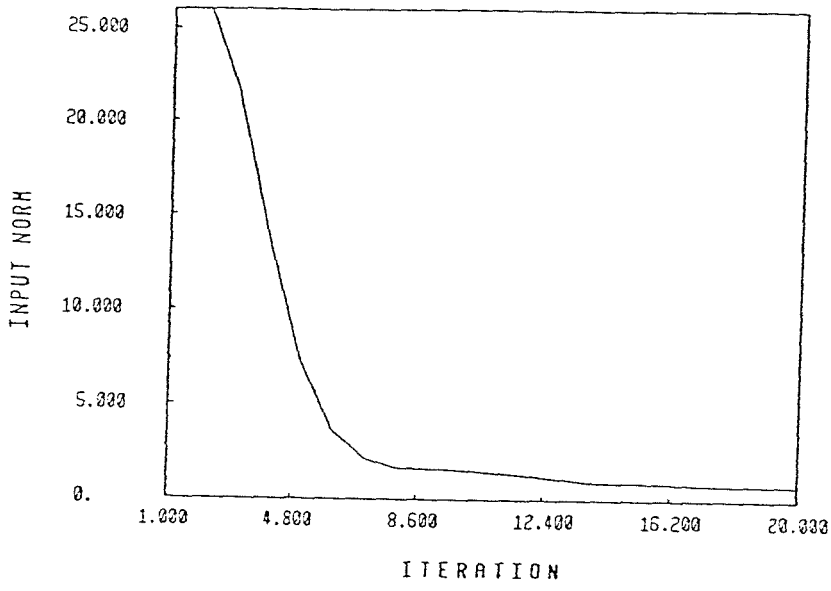


Fig. 2.13 FIR extrapolation method 1. Input norm v/s iterations.

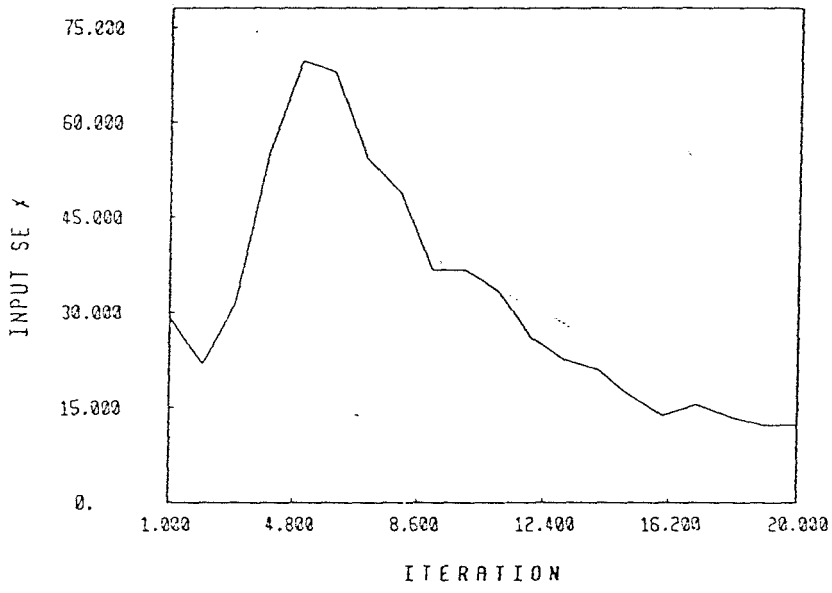


Fig. 2.14 FIR extrapolation method 1. Stopband energy percent (SE) of input v/s iterations.

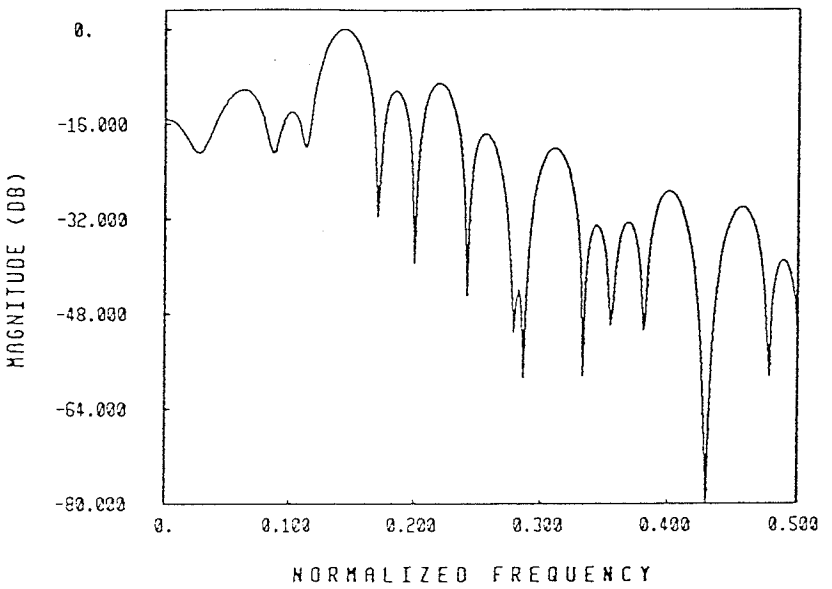


Fig. 2.15 FIR extrapolation method 1. Magnitude of the Fourier transform; the final input.

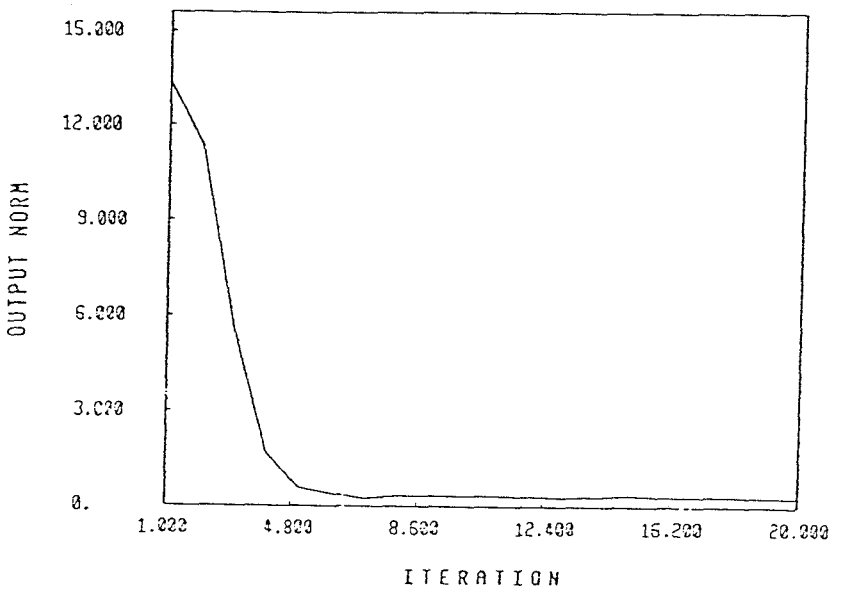


Fig. 2.16 FIR extrapolation method 1. Energy of extrapolation result v/s iterations.

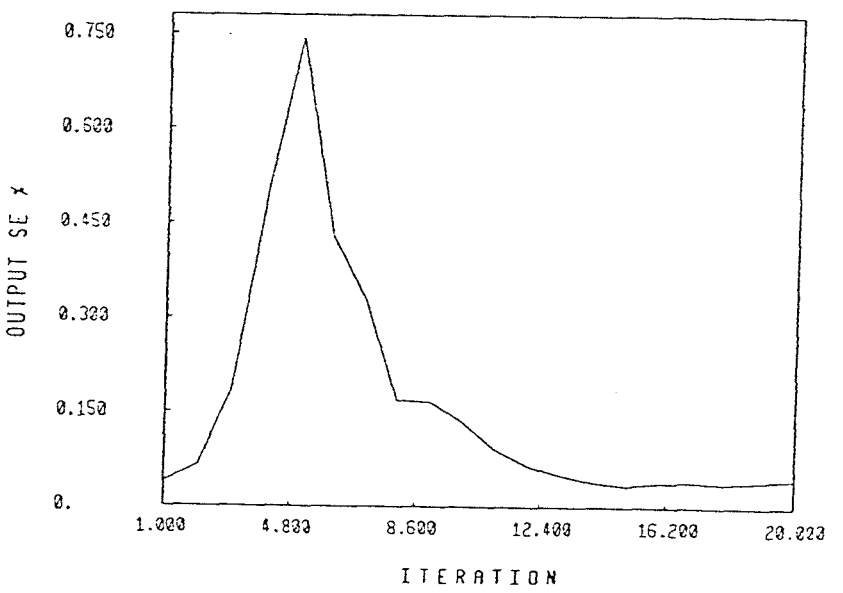


Fig. 2.17 FIR extrapolation method 1. Stopband energy percent (SE) of extrapolation result v/s iterations.

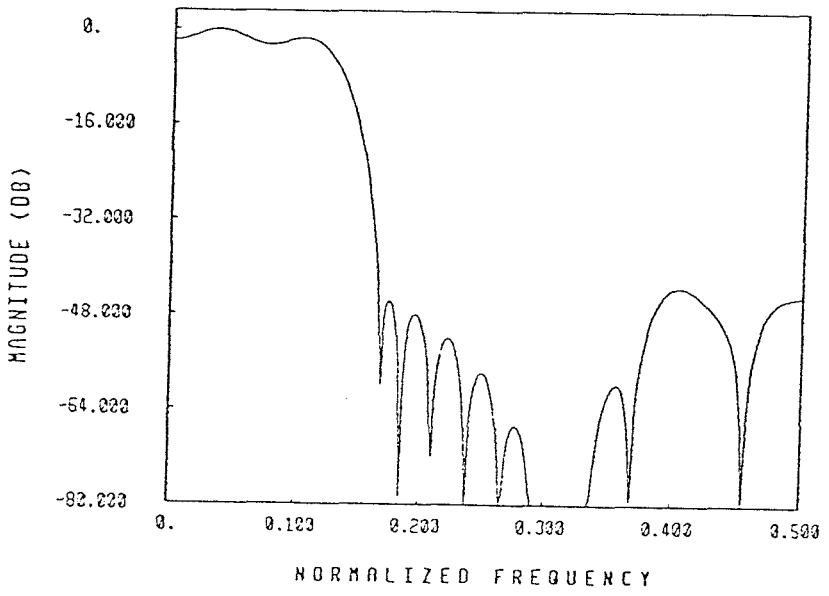


Fig. 2.18 FIR extrapolation method 2. Frequency response of the input length 41 IFIR filter.

windowing. Fig. 2.20(a) shows the resulting extrapolation signal, and (b) shows the corresponding frequency domain plot.

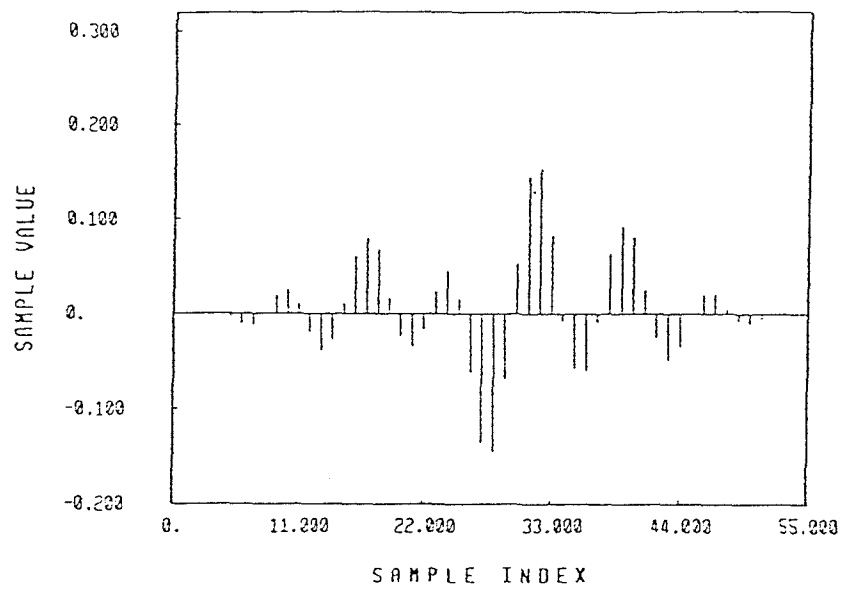
Noise performance of the extrapolation schemes

To study robustness of these extrapolation schemes, the following simulation was performed. We took the original 15-length segment and added a certain amount of random noise to it. The energy of the difference between the original extrapolation result and the new result was used as a measure of error. Fig. 2.21 shows the extrapolation of the noisy original segment using the IIR method. Fig. 2.22 shows the noisy extrapolation using the FIR Method 1. Fig. 2.23 shows the noisy extrapolation obtained using the FIR Method 2. Fig. 2.24 shows the extrapolation for the pseudoinverse method. To get a relative feel of the noise performance, we have listed the energy of difference of each of the extrapolation schemes and schemes we will see later on, in Table 2.1.

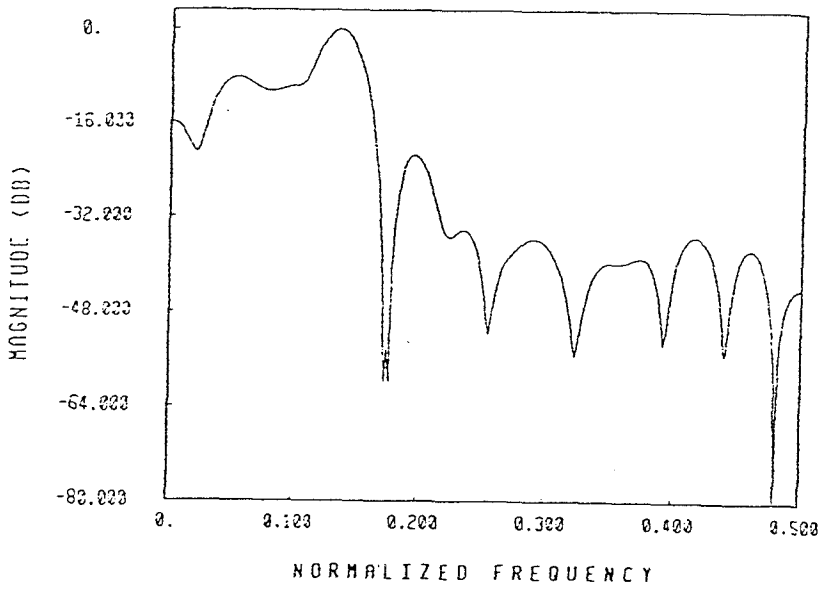
All the above methods involve inversion of a convolution matrix. Hence numerical stability and noise performance of these schemes depends upon the condition-number of the convolution matrix. In the IIR method and the FIR Method 1, we have to update the inverse at each iteration. Hence, we should not only be interested in the condition number at the final iteration, but also in how the condition number changes with the iterations, because this affects the roundoff noise propagation. This is shown in Table 2.2. We compare the final condition numbers for these methods in Table 2.3.

2.2 A First Principles Approach To σ-BL Extrapolation

In this section we shall discuss a procedure for generating a σ-BL extrapolation based on a very simple set of linear equations. The approach consists of two steps. The first step is independent of the given segment $x(n)$ (except for its length N) and has to be performed only once for a given N. The second step depends on the

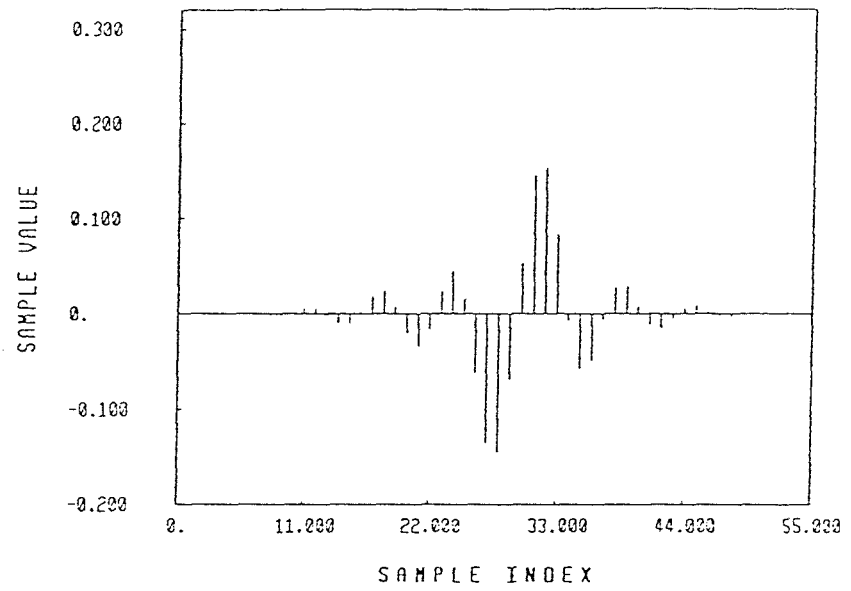


(a)

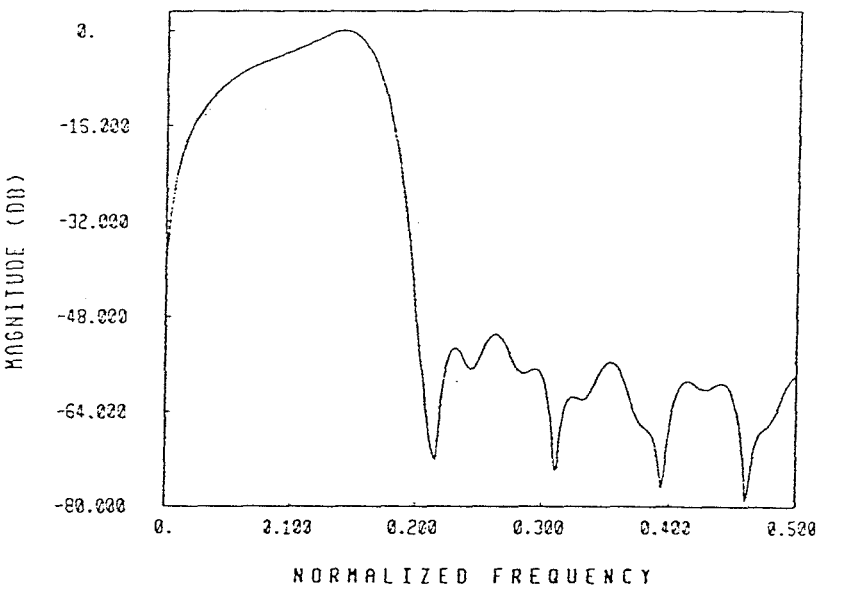


(b)

Fig. 2.19 FIR extrapolation method 2. The extrapolation result (a) time domain (b) magnitude of the Fourier transform.



(a)



(b)

Fig. 2.20 Pseudoinverse extrapolation method. The extrapolation result (a) time domain (b) magnitude of the Fourier transform.

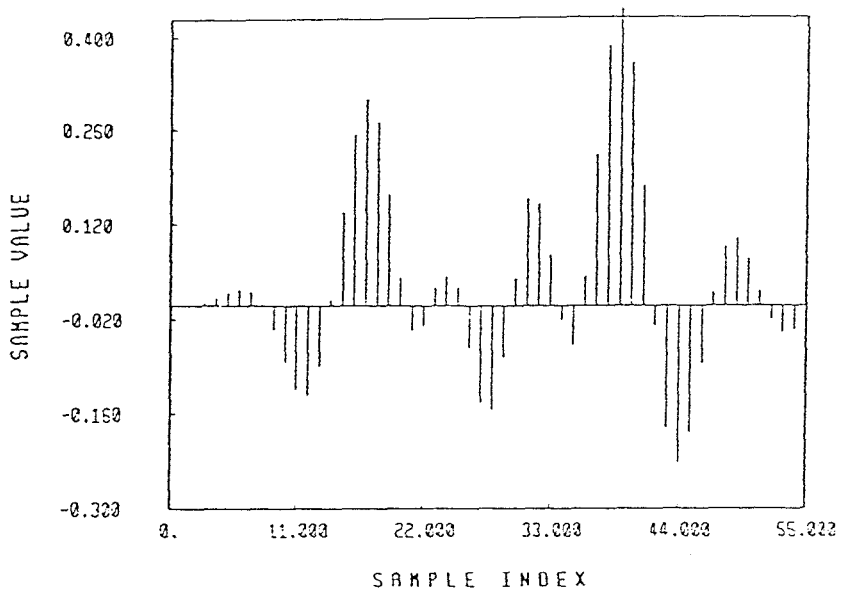


Fig. 2.21 IIR extrapolation method. The extrapolation result with 1% noise added to the original segment.

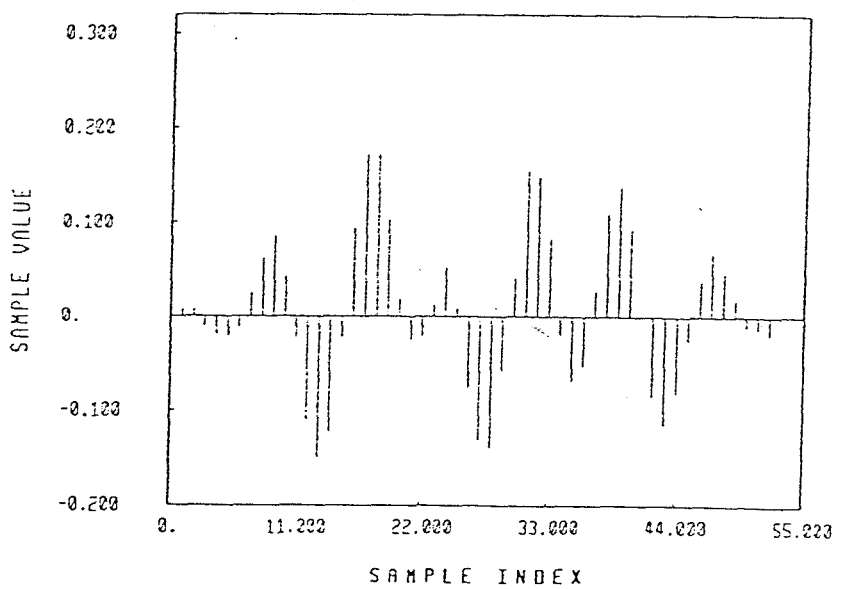


Fig. 2.22 FIR extrapolation method 1. The extrapolation result with 1% noise added to the original segment.

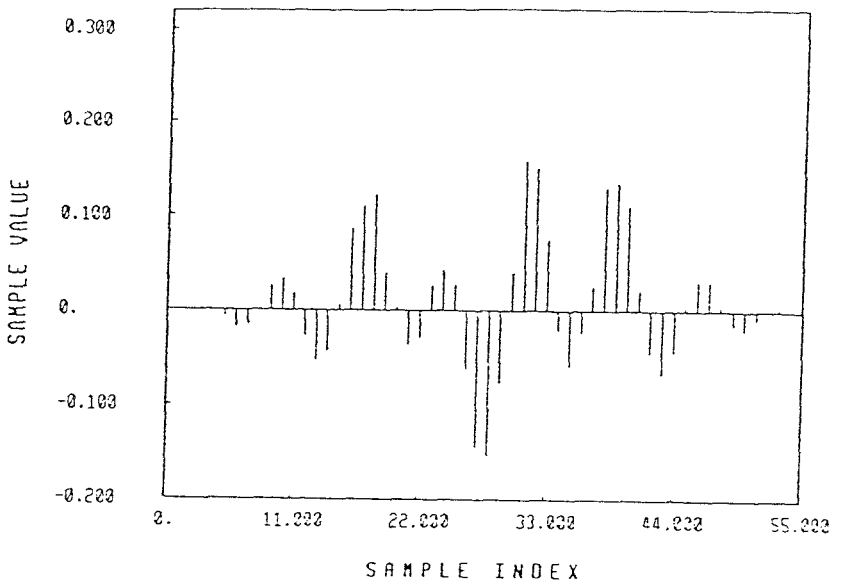


Fig. 2.23 FIR extrapolation method 2. The extrapolation result with 1% noise added to the original segment.

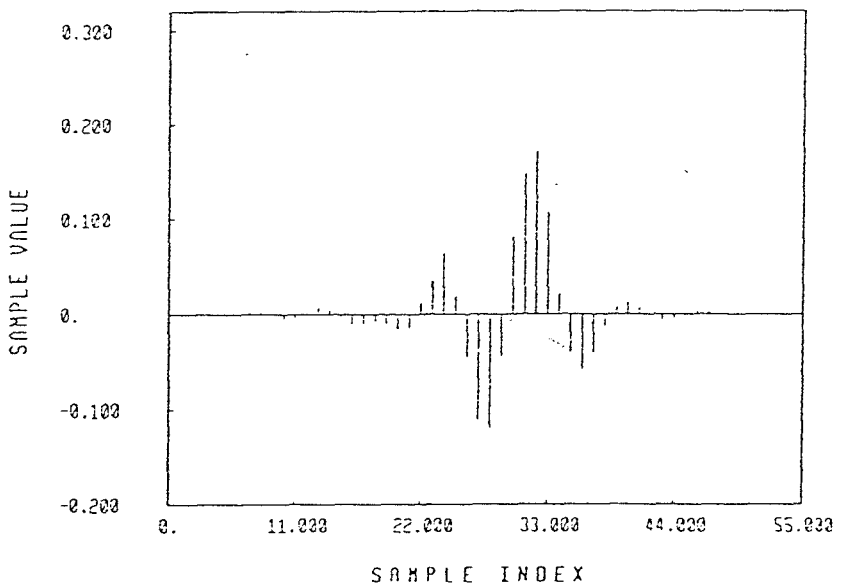


Fig. 2.24 Pseudoinverse extrapolation method. The extrapolation result with 1% noise added to the original segment.

Extrapolation Scheme	The energy of the difference (normalized to the energy of the original extrapolation)%
IIR	53.8
FIR Method 1	5.59
FIR Method 2	11.83
Pseudoinverse	11.77
LPC	22.85
Optimal	111.1
RLS	12.95
Fast RLS	10.55

Table 2.1 Relative noise performance of different extrapolation schemes with 1% noise added to the original segment.

Iteration	IIR Filter	FIR Filter
0	8.825×10^8	3.79×10^4
1	8.388×10^8	6.175×10^3
2	3.648×10^5	7.132×10^3
3	2.850×10^5	5.971×10^3
4	2.851×10^5	3.829×10^3
5	2.854×10^5	2.916×10^3
6	2.855×10^5	2.752×10^3
7	2.854×10^5	2.893×10^3
8	2.853×10^5	3.070×10^3
9	2.851×10^5	3.185×10^3
10	2.851×10^5	3.233×10^3
11	2.851×10^5	3.236×10^3
12	2.852×10^5	3.427×10^3
13	2.853×10^5	3.275×10^3
14	2.854×10^5	3.294×10^3
15	2.854×10^5	3.298×10^3
16	2.855×10^5	3.287×10^3
17	2.855×10^5	3.252×10^3
18	2.855×10^5	2.986×10^3
19	2.854×10^5	1.803×10^3
20	2.854×10^5	1.653×10^3

Table 2.2 Change in the condition number as iterations progress for the IIR extrapolation method and the FIR extrapolation method 1.

Extrapolation Scheme	Condition Number
IIR	285448.26
FIR Method 1	1652.94
FIR Method 2	742624.31
Pseudoinverse	250993.17

Table 2.3 Final condition number for different extrapolation schemes.

segment $x(n)$. These steps are:

1. Generate a set of N σ-BL sequences $y_k(n)$, $0 \leq k \leq N-1$ such that

$$y_k(n) \begin{cases} = 0 & 0 \leq n \leq k-1 \\ \neq 0 & n = k \end{cases} . \tag{2.12}$$

   Note in particular that $y_0(n)$ is not required to be zero for any n in $0 \leq n \leq N-1$.

2. Now define $y(n)$ to be a linear combination of $y_k(n)$, i.e.,

$$y(n) = \sum_{k=0}^{N-1} \alpha_k y_k(n) \tag{2.13}$$

   and choose the N constants α_k such that $y(n)$ matches the given segment, i.e., such that (2.1) holds.

Before elaborating on Step 1, note that Step 2 is particularly easy to perform because of the zero-valued samples introduced in (2.12). This is evident if we write (2.13) in matrix-vector form

$$\begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix} = \begin{bmatrix} y_0(0) & 0 & \dots & 0 \\ y_0(1) & y_1(1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ y_0(N-1) & y_1(N-1) & \dots & y_{N-1}(N-1) \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-1} \end{bmatrix} \tag{2.14}$$

which reveals the triangular nature of the equations to be solved. Given the waveforms $y_k(n)$ for $0 \leq n \leq N-1$ and the segment to be extrapolated, we can compute the α's one at a time in the order α_1, α_2, ... which requires a total of $N(N-1)/2$ addition and multiplication operations and N division operations. Since the diagonal elements of the matrix in (2.14) are chosen to be nonzero, the $\alpha's$ can always be found in this manner.

It remains to generate the N σ-BL sequences $y_k(n)$ satisfying (2.12). This is indeed always possible as first shown in [Hay83]. For this, let $s(n)$ denote some σ-BL waveform (for example $s(n) = \sin(\sigma n)/\pi n$) and assume that $s(n)$ is passed

through an FIR filter $G(z) = \sum_{n=0}^{N-1} g(n)z^{-n}$. The output sequence $t(n)$ which is the convolution of $s(n)$ with $g(n)$ is clearly σ-BL. The samples of $t(n)$ in the duration $0 \leq n \leq N-1$ are given by

$$\begin{bmatrix} t(0) \\ t(1) \\ \vdots \\ t(N-1) \end{bmatrix} = \begin{bmatrix} s(0) & s(-1) & \dots & s(-N+1) \\ s(1) & s(0) & \dots & s(-N+2) \\ \vdots & \vdots & \ddots & \vdots \\ s(N-1) & s(N-2) & \dots & s(0) \end{bmatrix} \begin{bmatrix} g(0) \\ g(1) \\ \vdots \\ g(N-1) \end{bmatrix}. \quad (2.15)$$

We can always specify $t(n)$ to be such that the first $N-1$ components $t(0), \dots t(N-2)$ on the left-hand side of (2.15) are zero, and specify $t(N-1) \neq 0$ (say $t(N-1) = 1$). We can then solve for the N coefficients $g(n)$ from (2.15). The resulting sequence $t(n)$ is σ-BL with a prescibed number of consecutive zeros. In this manner all the waveforms $y_k(n)$ in (2.12) can be generated. In fact, once $t(n)$ is generated as above, one procedure to obtain $y_k(n)$ would be to define

$$y_k(n) = t(n + N - 1 - k). \quad (2.16)$$

The above method will succeed as long as the $N \times N$ matrix in (2.15) is nonsingular. This is easy to ensure. For example, with $s(n) = \sin(\sigma n)/\pi n$ it is well-known [Jai81] that this matrix is nonsingular. The extrapolated sequence $y(n)$ can now be generated for as many values of n as desired by use of (2.13).

Summarizing, we can always obtain an ideal σ-BL extrapolation $y(n)$ for any given segment $x(n)$ simply by executing steps 1 and 2 above. The result is non-unique because the prototype ideal σ-BL waveform is arbitrary. Moreover, given a σ-BL extrapolation $y(n)$, we can always generate another valid extrapolation simply by adding an arbitrary σ-BL sequence $y_N(n)$ whose samples are equal to zero for $0 \leq n \leq N-1$, as pointed out in [Hay83].

From a mathematical viewpoint the above approach for extrapolation can be interpreted as a special case of *backward* linear predictive coding (LPC) [Hay86], [And79] of a wide-sense stationary random process. As a result, it is possible to use

Levinson's recursion [And79] in order to speed up the computation of $g(n)$ in (2.15). In order to see this notice that the $N \times N$ matrix in (2.15) is a Toeplitz matrix, i.e., all the elements along any line parallel to the main diagonal are identical. Furthermore, by choice of $s(n)$ it is easy to make this matrix symmetric as well as positive definite (just by ensuring that $S(e^{j\omega})$ is real and non-negative for all ω). For example, if $s(n) = \sin(\sigma n)/\pi n$, then this matrix is indeed a real symmetric positive definite Toeplitz matrix, i.e., it is a valid autocorrelation matrix of a WSS random process.

Recall that the desired left-hand side in (2.15) has all elements equal to zero except the last element which is required to be non-zero even though its exact value is not critical. In other words we can obtain a valid set of $g(n)'s$ simply by solving the following equations

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \mathcal{E} \end{bmatrix} = \begin{bmatrix} s(0) & s(1) & \dots & s(N-2) & s(N-1) \\ s(1) & s(0) & \dots & s(N-3) & s(N-2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ s(N-2) & s(N-3) & \dots & s(0) & s(1) \\ s(N-1) & s(N-2) & \dots & s(1) & s(0) \end{bmatrix} \begin{bmatrix} b_{N-1} \\ b_{N-2} \\ \vdots \\ b_1 \\ 1 \end{bmatrix} \tag{2.17}$$

for the unknown coefficients $b_n, 1 \leq n \leq N-1$, and where $\mathcal{E}$ is some non-zero quantity. But the set of equations is precisely the *augmented set of* normal equations which arise in the backward LPC problem [And79]. We can solve for b_n from the standard set of normal equations

$$\begin{bmatrix} s(0) & s(1) & \dots & s(N-2) \\ s(1) & s(0) & \dots & s(N-3) \\ \dots & \dots & \ddots & \dots \\ s(N-2) & s(N-1) & \dots & s(0) \end{bmatrix} \begin{bmatrix} b_{N-1} \\ b_{N-2} \\ \vdots \\ b_1 \end{bmatrix} = - \begin{bmatrix} s(N-1) \\ s(N-2) \\ \vdots \\ s(1) \end{bmatrix} \tag{2.18}$$

The quantity $\mathcal{E}$ which should be interpreted as the minimized mean-square prediction error, is guaranteed to be nonzero unless the matrix in (2.12) is singular which is not the case here.

Lattice-structure representation for the extrapolator.

In order to find the coefficients of the $(N-1)$th order backward predictor, Levinson's recursion computes predictors of lower orders in succession, starting from a first order predictor. Let the predictor coefficients for the mth order predictor be denoted $b_{m,i}$, $1 \leq i \leq m$ so that $b_{N-1,i} = b_i$ where b_i are the coefficients in (2.18). Let $v(n)$ represent the WSS random process being predicted (so that $s(n)$ in (2.17) is its autocorrelation). Then the mth order backward predictor produces the estimate $\hat{v}(n-m-1) = -\sum_{k=1}^{m} b_{m,k} v(n-k)$ of the sample $v(n-m-1)$. The prediction error is defined as $e_m(n) = v(n-m-1) - \hat{v}(n-m-1)$ so that $e_m(n)$ can be considered to be the output sequence of the FIR filter $z^{-1}B_m(z) \triangleq \sum_{k=1}^{m} b_{m,k} z^{-k} + z^{-m-1}$ in response to the input sequence $v(n)$. Recall that (2.15) represents a convolution of the sequence $s(n)$ with the sequence $g(n)$. By comparing this with (2.17) we see that a set of N sequences $y_m(n)$, $0 \leq m \leq N-1$ which satisfy (2.12) can be generated as outputs of the system $B_m(z)$ in response to the input $s(n)$. Now it is well-known that the transfer functions $B_m(z)$, $1 \leq m \leq N-1$ can be realized in the form of a cascaded lattice (called the LPC lattice; see [Mar76]. The extrapolated sequence $y(n)$ given by (2.13) can therefore be represented as the output of a tapped cascaded lattice (Fig. 2.25). The input to this lattice is the σ-BL sequence $s(n)$. The tap coefficients α_m are computed according to (2.14). The signals $y_m(n)$ which are being tapped are the outputs of the intermediate transfer functions $B_m(z)$.

In practice if the segment length N is large, the inputs to the taps α_m (which are mean square prediction errors $\mathcal{E}_m$) become very small as m increases. In order to obtain a numerically more robust scheme, the well-known normalized lattice [Mar76] can be used with the taps appropriately modified.

Example 2.2 : Again, we use the same 15-point sequence as used in Example 1 and run the LPC extrapolation method to get the extrapolation output.

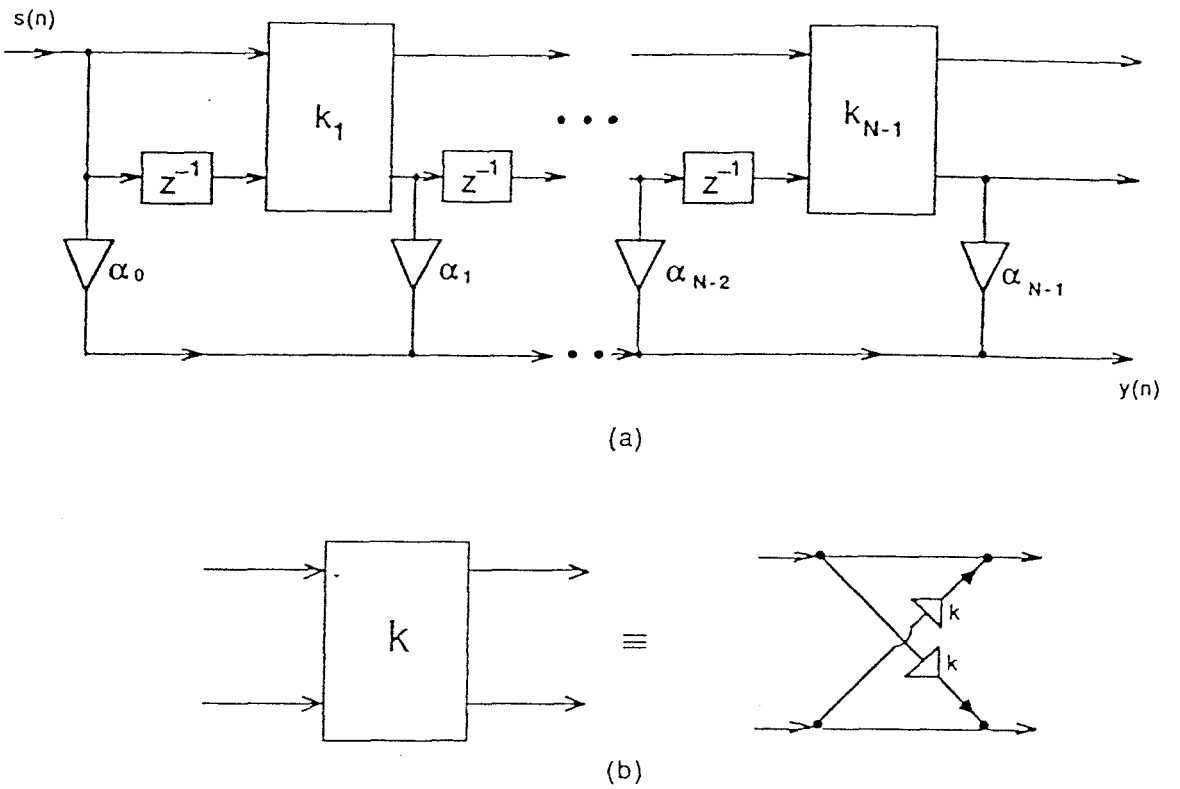


Fig. 2.25 The LPC lattice and σ - BL extrapolation.

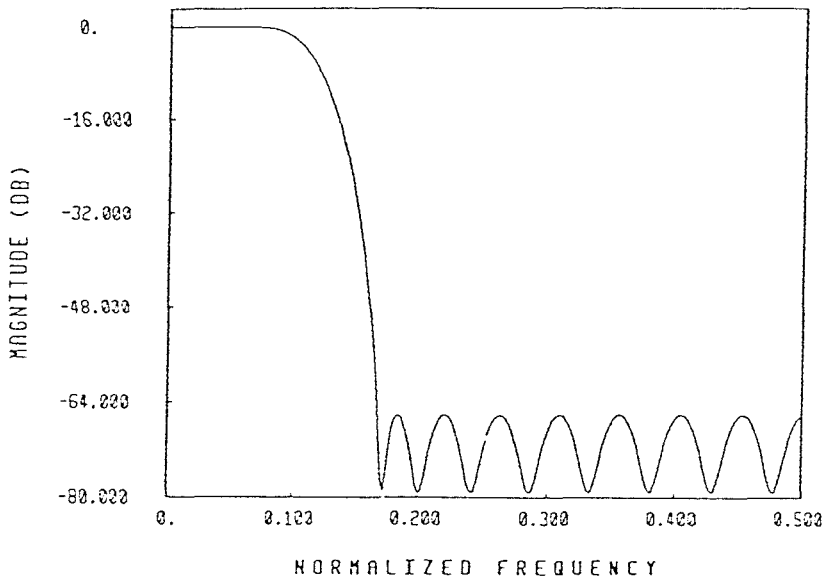
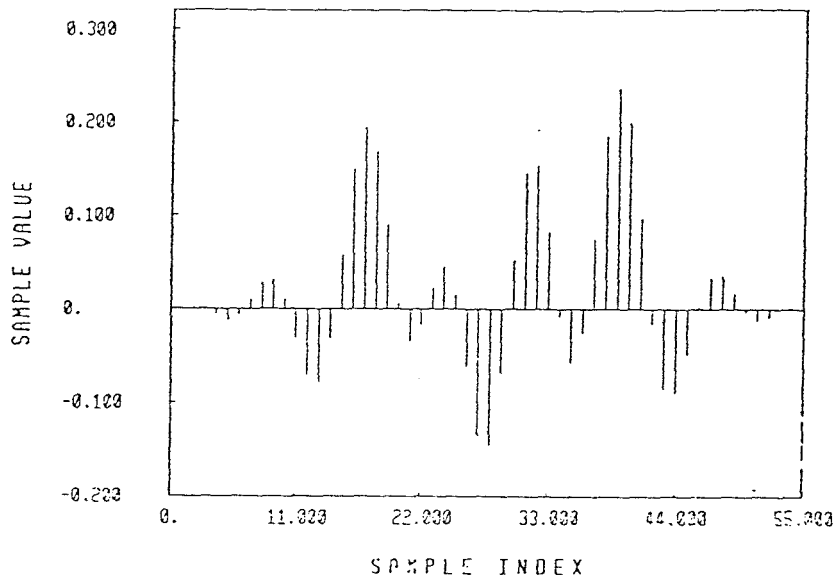
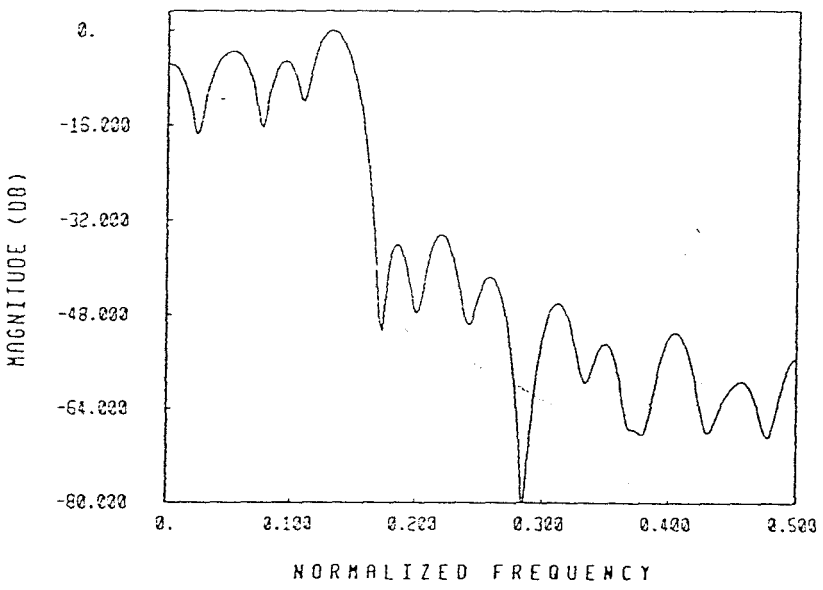


Fig. 2.26 LPC extrapolation method. (a) The input filter with positive power spectrum.



(b)



(c)

Fig. 2.26 The extrapolation result (b) time domain (c) the magnitude of Fourier transform.

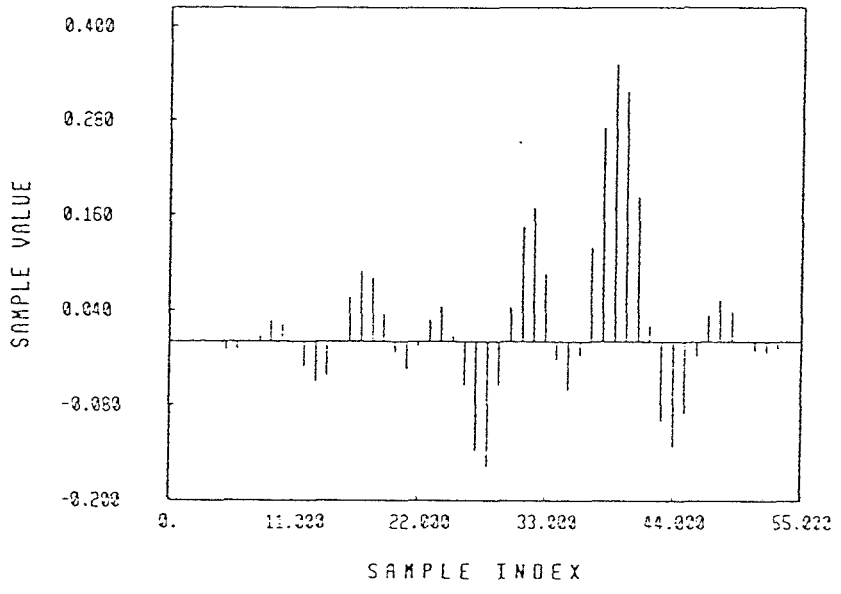


Fig. 2.27 LPC extrapolation method. The extrapolation result with 1% noise added to the original segment.

Fig. 2.26(a) shows the quantity $S(e^{j\omega})$ which was used in the algorithm. Note that $S(e^{j\omega}) > 0$. Fig. 2.26(b) shows the extrapolated signal $y(n)$ and Fig. 2.26(c) the corresponding Fourier transform magnitude plot. We have shown the values of the coefficients $\alpha_i's$ for the lattice structure in Table 2.4. It has been observed that magnitudes of $\alpha_i's$ depend on how positive the spectrum $S(e^{j\omega})$ is. To highlight this, we have also included the values of the α coefficients when $S(e^{j\omega})$ has double zeros for some ω. It can be seen that although individual coefficients α_i do not increase, the coefficients increase overall. This is shown by increase in the sum of squares of alphas.

To study the noise performance, we follow the same procedure as in the Example 2.1, and the resulting extrapolation is shown in Fig. 2.27. We have included the result of this example in Table 2.1.

2.3 Optimal Finite-Length σ-BL Extrapolation

Given a segment $x(n)$ of finite length N, it is often more appropriate to find a longer sequence $y(n)$ of *finite* length $L > N$ such that a subset of samples of $y(n)$ match the segment (as in (2.2)) and the remaining $L - N$ samples of $y(n)$ are such that $y(n)$ is "as bandlimited as possible." Such an approach has a practical advantage because in practice one prefers to do only a finite amount of computation. If we obtain such $y(n)$ by truncating (or windowing) one of the known methods (such as the one in [Jai81], or the one in Sec. 2.3 of this paper) then the result is not necessarily optimal in any sense. In this section we formulate a direct optimization for this. Some of these results have been reported first in a conferecne paper [Liu89]. The following is a review of the method reported in [Liu89].

Without loss of generality we assume that $y(n)$ is causal with possibly nonzero values in $0 \leq n \leq L-1$ and that the N samples $y(M)$ ... $y(M+N-1)$ match

the given segment. Define the following vectors:

$$\mathbf{x} = [x(0)\ x(1)\ \ldots\ x(N-1)]^T, \tag{2.19}$$

and

$$\mathbf{y} = [y(0)\ y(1)\ \ldots\ y(M-1)\ y(M+N)\ \ldots\ y(L-1)]^T. \tag{2.20}$$

The Fourier transform $Y(e^{j\omega}) = \sum_{n=0}^{L-1} y(n)e^{-j\omega n}$ of the finite-length extrapolation can be written as

$$Y(e^{j\omega}) = \mathbf{f}^\dagger(\omega)\mathbf{y} + \mathbf{e}^\dagger(\omega)\mathbf{x} \tag{2.21}$$

where

$$\mathbf{e}^\dagger(\omega) = e^{-j\omega M}[1\ e^{-j\omega}\ \ldots\ e^{-j\omega(N-1)}], \tag{2.22}$$

and

$$\mathbf{f}^\dagger(\omega) = [1\ e^{-j\omega}\ \ldots\ e^{-j\omega(M-1)}\ e^{-j\omega(M+N)}\ \ldots\ e^{-j\omega(L-1)}]. \tag{2.23}$$

We can then express the out-of-band energy as

$$\int_\sigma^\pi |Y(e^{j\omega})|^2 \frac{d\omega}{\pi} = \mathbf{y}^\dagger\mathbf{Q}\mathbf{y} + \mathbf{y}^\dagger\mathbf{P}\mathbf{x} + \mathbf{x}^\dagger\mathbf{P}^\dagger\mathbf{y} + \mathbf{x}^\dagger\mathbf{T}\mathbf{x} \tag{2.24}$$

where

$$\mathbf{Q} = \int_\sigma^\pi \mathbf{f}(\omega)\mathbf{f}^\dagger(\omega)\frac{d\omega}{\pi},\ \mathbf{P} = \int_\sigma^\pi \mathbf{f}(\omega)\mathbf{e}^\dagger(\omega)\frac{d\omega}{\pi},\ \mathbf{T} = \int_\sigma^\pi \mathbf{e}(\omega)\mathbf{e}^\dagger(\omega)\frac{d\omega}{\pi}. \tag{2.25}$$

The fourth term in (2.24) is fixed since it depends only on $\mathbf{x}$. The appropriate objective function representing the stopband energy is

$$\phi_s = \mathbf{y}^\dagger\mathbf{Q}\mathbf{y} + \mathbf{y}^\dagger\mathbf{P}\mathbf{x} + \mathbf{x}^\dagger\mathbf{P}^\dagger\mathbf{y}. \tag{2.26}$$

If we minimize this quantity then we obtain a finite-length extrapolation (which satisfies (2.2) exactly) and which has the smallest out-of-band energy. However, it is often desirable to add a second term to the objective function so that the energy

of $y(n)$ does not unduly dominate that of the given segment. The energy of $y(n)$ is clearly given in the time domain by $\mathbf{y}^\dagger\mathbf{y} + \mathbf{x}^\dagger\mathbf{x}$ of which the portion $\mathbf{x}^\dagger\mathbf{x}$ is fixed. The first term $\mathbf{y}^\dagger\mathbf{y}$ represents the increase in energy due to extrapolation. We shall define the composite objective function

$$\phi = \alpha\phi_s + (1-\alpha)\mathbf{y}^\dagger\mathbf{y}, \tag{2.27}$$

where $0 < \alpha \leq 1$ is a weighting factor for the relative importance of the out-of-band energy and the total energy of the extrapolation $y(n)$. The value of this factor is entirely at the discretion of the designer. Eqn. (2.27) can be rewritten in the form

$$\phi = \mathbf{y}^\dagger\mathbf{R}\mathbf{y} + \mathbf{y}^\dagger\mathbf{S}\mathbf{x} + \mathbf{x}^\dagger\mathbf{S}^\dagger\mathbf{y} \tag{2.28}$$

where $\mathbf{R} = \alpha\mathbf{Q} + (1-\alpha)\mathbf{I}$, and $\mathbf{S} = \alpha\mathbf{P}$. By completion of squares, (2.28) can be written as

$$\phi = (\mathbf{y} + \mathbf{R}^{-1}\mathbf{S}\mathbf{x})^\dagger\mathbf{R}(\mathbf{y} + \mathbf{R}^{-1}\mathbf{S}\mathbf{x}) - (\mathbf{S}\mathbf{x})^\dagger\mathbf{R}^{-1}\mathbf{S}\mathbf{x}. \tag{2.29}$$

The second term in (2.29) is independent of $\mathbf{y}$. Since $\mathbf{R}$ is positive definite, the first term (and hence the objective function ϕ) is minimized by the unique choice $\mathbf{y} = -\mathbf{R}^{-1}\mathbf{S}\mathbf{x}$.

Interpretation as an FIR filter design problem with time-domain constraint

This method for extrapolation can be interpreted as a FIR filter-design problem as follows: we are interested in designing a "lowpass" impulse response $y(n)$ of length L with stopband edge σ. The specification does not include the passband, but contains a *time-domain* constraint on the impulse response given by (2.2).

Comments on complexity of the method.

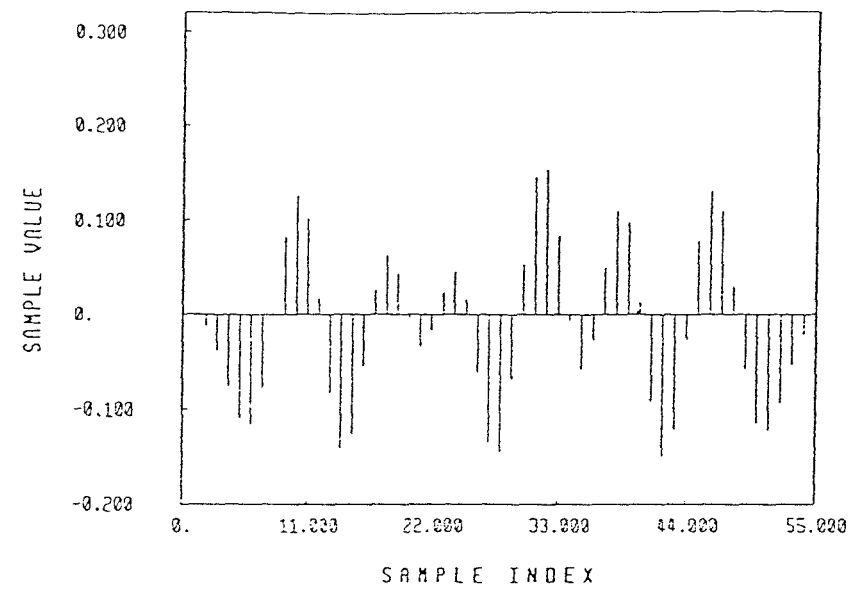
In this method the solution vector $\mathbf{y}$ of length $K = L - N$ is obtained by solving the set of K equations

$$\mathbf{R}\mathbf{y} = -\mathbf{S}\mathbf{x}. \tag{2.30}$$

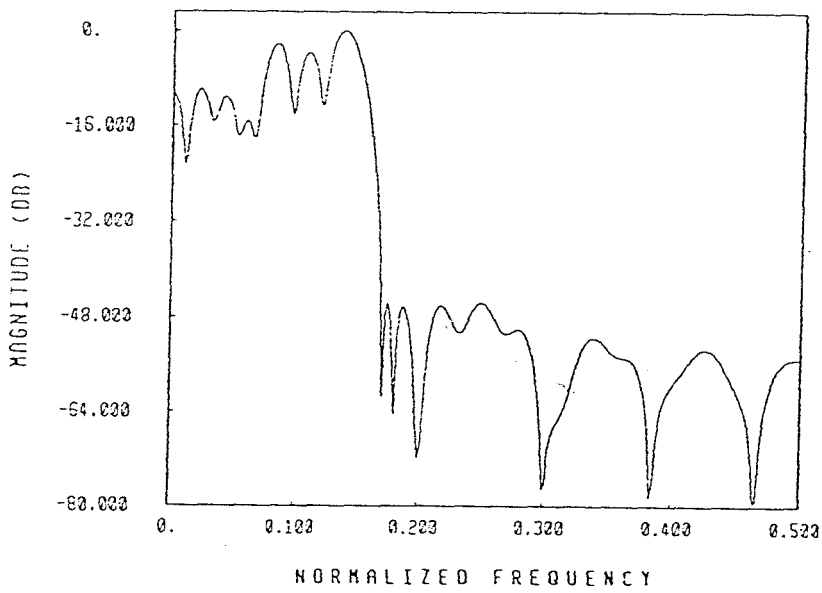
The matrix $\mathbf{R}$ is Hermitian and positive definite. However, it is $K \times K$ rather than $N \times N$ where N is the length of the given segment and $N + K$ is the total length of the extrapolation. For large K, therefore, this method is computationally more expensive than the methods in the previous sections. For the case of real signals (i.e., real $x(n)$ and $y(n)$, we can replace (2.30) with a set of equations involving real matrices, i.e., as $\mathbf{R}_1\mathbf{y} = \mathbf{c}$ where $\mathbf{R}_1$ and $\mathbf{c}$ are real. The matrix $\mathbf{R}_1$, however, is not Toeplitz. It is in fact obtained from a $L \times L$ Toeplitz matrix by dropping a set of N rows and the corresponding set of N columns. Standard fast-techniques for solving Toeplitz equations cannot therefore be directly applied. For these reasons, this method for σ-BL extrapolation is more expensive than most of the techniques described earlier, even though it gives a theoretically optimum finite-length extrapolation. The results of this technique can therefore be used as a standard for judging the performance of other methods.

Example 2.3: Although computationally expensive because of matrix inversion, the method described above gives us the best extrapolation result (in the sense of minimizing the stopband energy) for a given α. So, for the given segment, we can obtain the extrapolation for any α in the range $0 \leq \alpha \leq 1$ and then plot TDE versus SE for the extrapolated result. In this plot α is the parameter that varies along the curve. This plot gives the upper limit on performance for any extrapolation algorithm. We can interpret this curve as follows. For a given TDE, the point on the curve gives a lower bound on SE that can ever be accomplished by any extrapolation method which produces a length L extrapolation from the given length N sequence by any means whatsoever.

Fig. 2.28(a) shows extrapolated segment for $\alpha = 0.0003$ and Fig. 2.28(b) shows magnitude of its Fourier transform. The plot in Fig. 2.29 shows how the parameter SE varies with α. Fig. 2.30 shows the effect of changing α on the parameter TDE. The plot in Fig. 2.31 exhibits the trade off between the TDE and



(a)



(b)

Fig. 2.28 Optimal extrapolation. The extrapolation result for $\alpha = 0.0003$. (a) time domain (b) magnitude of the Fourier transform.

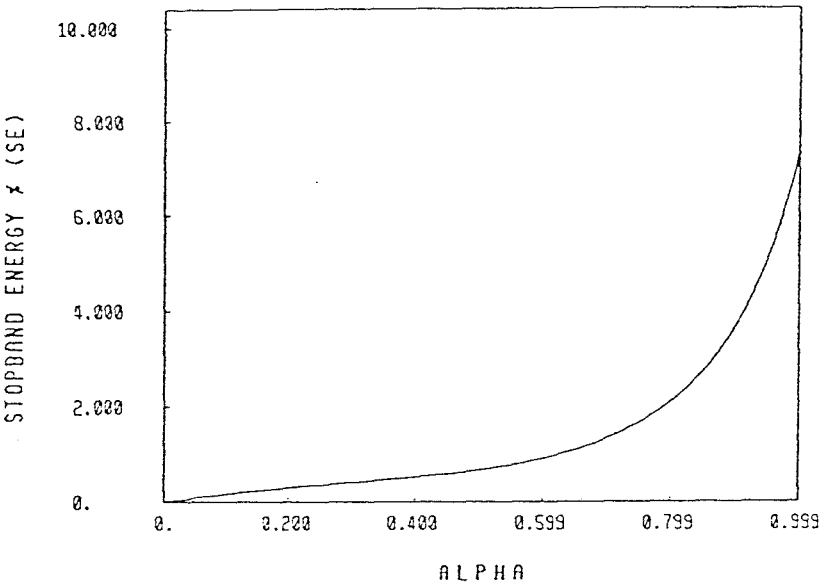


Fig. 2.29 Optimal extrapolation. Stopband energy % (SE) v/s α

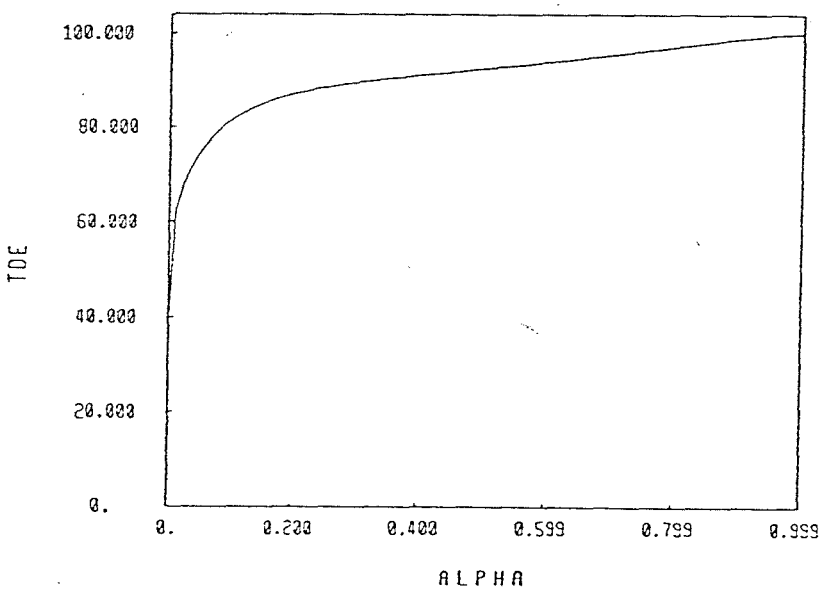


Fig. 2.30 Optimal extrapolation. time domain energy (TDE) v/s α

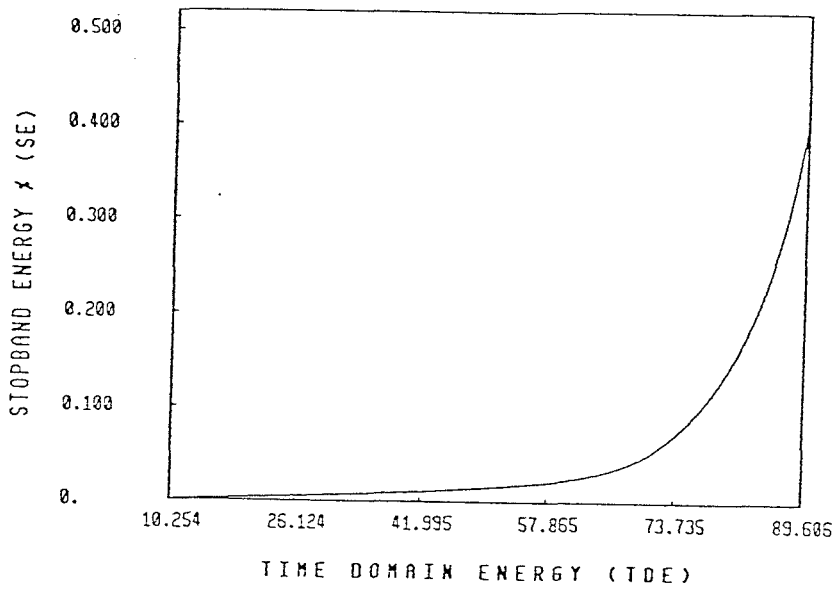


Fig. 2.31 Optimal extrapolation. Stopband energy % v/s time domain energy.

SE parameters as α varies in the range $0 \leq \alpha \leq 1$.

2.4 Bandlimited Extrapolation Of Periodic Signals

If we impose the constraint that the extrapolation $y(n)$ be periodic (hence infinitely long) with period L then the problem takes a particularly simple form. Even though this is handled in [Jai81], we shall present a solution which we believe to be more direct and simpler.

Given the segment $x(n)$, $0 \leq n \leq N-1$ of length N our aim is to find a periodic signal $y(n)$ with period $L > N$ such that $y(n) = x(n)$ for $0 \leq n \leq N-1$ and $y(n)$ is appropriately bandlimited. Because of periodicity, the Fourier transform of $y(n)$ is zero for all ω except $\omega = 2\pi k/L$ where $0 \leq k \leq L-1$. Equivalently if we define $Y(k)$ to be the L-point DFT of $y(n)$, i.e., $Y(k) = \sum_{n=0}^{L-1} y(n)W^{nk}$ where $W = e^{-j2\pi/L}$, then $y(n)$ is completely determined by $Y(k)$ as $y(n) = \frac{1}{L}\sum_{k=0}^{L-1} Y(k)W^{-kn}$. We shall say that $y(n)$ is K-BL if $Y(k) = 0$ for $K \leq k \leq L-K$. For such a sequence the samples $y(n)$ can be expressed as

$$Ly(n) = Y(0) + \sum_{k=1}^{K-1} Y(k)W^{-nk} + \sum_{k=L-K+1}^{L-1} Y(k)W^{-nk}, \quad 0 \leq n \leq L-1. \quad (2.31)$$

We are given the segment $x(n)$ of length N to be extrapolated. The first N samples of $y(n)$ are set equal to $x(n)$ so that $y(n) = x(n)$, $0 \leq n \leq N-1$. Defining

$$\mathbf{x} = [x(0)\ x(1)\ \ldots\ x(N-1)]^T \quad (2.32)$$

and

$$\mathbf{Y} = [Y(0)\ Y(1)\ \ldots\ Y(K-1)\ Y(L-K+1)\ \ldots\ Y(L-1)]^T \quad (2.33)$$

we have from (31)

$$\mathbf{x} = \mathbf{A}\mathbf{Y} \quad (2.34)$$

where $\mathbf{A}$ is a $N \times (2K-1)$ submatrix of the Inverse DFT matrix. Given the N samples in $\mathbf{x}$, if we can find the $2K-1$ components of $\mathbf{Y}$ by solving (2.34) then all the remaining samples $y(n), N \leq n \leq L-1$ can be computed using (2.31). Clearly $y(n)$ is a periodic extrapolation of $x(n)$ and is in addition K-BL.

The $N \times (2K-1)$ matrix $\mathbf{A}$ in (34) can be expressed (see Appendix 2.A) as $\mathbf{A} = \mathbf{DVP}$ where $\mathbf{D}$ is a $N \times N$ nonsingular diagonal matrix, $\mathbf{V}$ is a $N \times (2K-1)$ Vandermonde matrix and $\mathbf{P}$ is a $(2K-1) \times (2K-1)$ permutation matrix (hence nonsingular). The nth row of $\mathbf{V}$ is given by

$$[1 \;\; W^{-n} \;\; W^{-2n} \;\; \ldots \;\; W^{-(2K-2)n}], \quad 0 \leq n \leq N-1. \tag{2.35}$$

Moreover the rows are distinct so that for $N \leq 2K-1$, the rank of $\mathbf{A}$ is equal to N. We can then find an exact solution for (34) which in addition is unique if $N = 2K-1$. If $N < 2K-1$ we can exploit it by finding the minimum-norm solution

$$\hat{\mathbf{Y}} = \mathbf{A}^{\#}\mathbf{x} \tag{2.36}$$

where $\mathbf{A}^{\#}$ is the Moore-Penrose psuedoinverse of $\mathbf{A}$. This continues to be an exact solution since the rank of $\mathbf{A}$ is still N. So for $N \leq 2K-1$ we can always find an exact (and unique) solution $Y(k), 0 \leq k \leq L-1$ which has minimum norm so that the extrapolation $y(n)$, $0 \leq n \leq L-1$ has minimum energy. Moreover the extrapolation satisfies (2.2) exactly with $M = 0$, and is K-BL.

On the other hand if $N > 2K-1$, we can find only an approximate solution for (2.34). We can take this to be the MNLS solution again. Such a solution gives rise to an extrapolation which is bandlimited as desired, but the time-domain segment is matched only approximately and in the least squares sense.

Unlike the MNLS problems formulated in Sec. II, the matrix $\mathbf{A}$ is not Toeplitz. RLS techniques for solving the system (2.34) are therefore not as efficient as in the

Toeplitz case. However, for the case where $N = 2K-1$ there exists a efficient procedure to find the extrapolated samples as discussed next.

Consider a sequence $t(n)$ which is equal to one period of $y(n)$, i.e., $t(n) = y(n)$ for $0 \leq n \leq L-1$ and zero outside this range. From (2.31) the z-transform of $t(n)$ is given by

$$T(z) = \left(\frac{1-z^{-L}}{L}\right)\left(\frac{Y(0)}{1-z^{-1}} + \sum_{k=1}^{K-1} \frac{Y(k)}{1-z^{-1}W^{-k}} + \sum_{k=1}^{K-1} \frac{Y(L-k)}{1-z^{-1}W^{k}}\right) \quad (2.37)$$

The first L samples of the inverse transform of the following quantity

$$S(z) = \frac{1}{L}\left(\frac{Y(0)}{1-z^{-1}} + \sum_{k=1}^{K-1} \frac{Y(k)}{1-z^{-1}W^{-k}} + \sum_{k=1}^{K-1} \frac{Y(L-k)}{1-z^{-1}W^{k}}\right) \quad (2.38)$$

clearly coincide with the L samples of $y(n)$ in $0 \leq n \leq L-1$. We can consider $s(n)$ to be the impulse response of a causal IIR filter $S(z)$ with numerator of degree $2K-2$ and denominator $B(z) = 1 + \sum_{k=1}^{2K-1} b_k z^{-k}$ of degree $2K-1$ so that $s(n)$ satisfies the difference equation $s(n) = -\sum_{k=1}^{2K-1} b_k s(n-k)$ for $n \geq 2K-1$. Clearly the sequence $y(n)$ satisfies this same difference equation, i.e.,

$$y(n) = -\sum_{k=1}^{2K-1} b_k y(n-k), \quad 2K-1 \leq n \leq L-1. \quad (2.39)$$

Summarizing, any periodic K-BL sequence $y(n)$ with period L satisfies (39) so that the first $2K-1$ samples $y(n), 0 \leq n \leq 2K-2$ determine the remaining samples in the period. As a result, if the given number N of samples is such that $N > 2K-1$ then there may not exist a K-BL extrapolation. If $N \leq 2K-1$ we can always find such an extrapolation merely by setting $y(N) = y(N+1) = \ldots = y(2K-2) = 0$ and then using the difference equation (2.39). The coefficients b_k of $B(z)$ are obtained by noting from (2.38) that

$$B(z) =$$

$$(1-z^{-1})\prod_{k=1}^{K-1}(1-W^{k}z^{-1})(1-W^{-k}z^{-1}) = (1-z^{-1})\prod_{k=1}^{K-1}(1-2z^{-1}\cos\theta_k + z^{-2}) \quad (2.40)$$

where $\theta_k = 2\pi k/L$.

2.5 The σ-BL Extrapolation Problem As A Recursive Least Squares Problem

In this section, we formulate the bandlimited extrapolation problem as a least squares problem which can be solved by standard recursive techniques like the RLS method (see Haykin's text [Hay86] and papers cited therein). In the standard least squares filtering problem, we have to adapt the coefficients of a filter such that for a given input $x(n)$, its output approximates a signal $d(n)$ in the least squares sense. The problem is schematically shown in Fig. 2.32. We can make use of this setup as follows. We want the extrapolated signal to be as good a lowpass signal as possible. This means that the extrapolated signal passed through a highpass filter with a *complementary* frequency response should give an output with small energy. Thus, with the extrapolated signal as the input and the highpass filter as the transfer function, we should have zero output as the desired response. This is shown in Fig. 2.33. This problem is not exactly in the conventional adaptive filtering framework. We don't know all the input data, yet we know all the filter coefficients. In fact the problem is to determine the missing samples from the input. To make the unknown quantities resemble the coefficients of an adaptive filter, we reverse the role of the highpass filter and the extrapolated signal and restate our problem by saying that with the impulse response of the highpass filter as input to a system whose impulse response is the extrapolated signal, we should have zero output as the desired signal.

Let *h(n)* be the impulse response of the highpass filter. We split the extrapolated signal as $\mathbf{y} = [\mathbf{y}_1^T, \mathbf{y}_2^T, \mathbf{y}_3^T]^T$, where $\mathbf{y}_1$ and $\mathbf{y}_3$ are the unknown parts respectively, of the extrapolated signal before and after the known part $\mathbf{y}_2$. The problem now is to identify an FIR filter with impulse response coefficients denoted

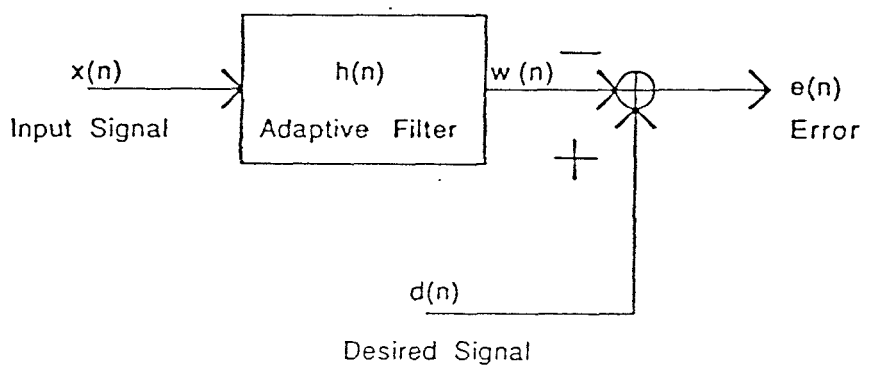


Fig. 2.32 A general adaptive filtering setup.

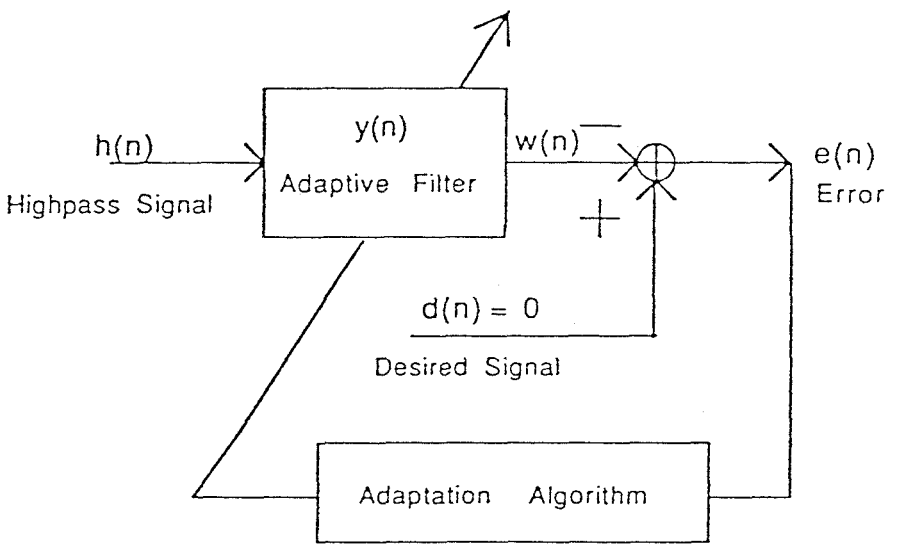


Fig. 2.33 An adaptive filtering scheme equivalent to the RLS bandlimited extrapolation problem.

by $\mathbf{y}$, such that the input sequence $h(n)$ to this filter produces an output as close to the desired signal as possible, where the desired signal is *identically zero.* Let N be the length of the given signal $\mathbf{y_2}$ to be extrapolated and L be the length of the total extrapolated signal. For simplicity we assume that the length of the extrpolated signal before and after the given signal samples is the same. Hence, lengths of $\mathbf{y}_1$ and $\mathbf{y}_3$ are $M = (L-N)/2$ each. Hence

$$\mathbf{y} = [\, y_0 \quad \cdots \quad y_{M-1} \quad y_M \quad \cdots \quad y_{N+M-1} \quad y_{N+M} \quad \cdots \quad y_{L-1} \,]^T \tag{2.41}$$

where the values $[y_M, \ldots\ldots, y_{N+M-1}]$ are known. We thus have an FIR system identification problem as shown in Fig. 2.33. If we define

$$\mathbf{h}(n) \;=\; [\, h(n), h(n-1), \ldots\ldots, h(n-L+1) \,]^T \tag{2.42}$$

then we can write

$$\begin{aligned} w'(n) &= \mathbf{h}^T(n)\, \mathbf{y} \\ &= [\mathbf{h}_1^T(n) \;\; \mathbf{h}_2^T(n) \;\; \mathbf{h}_3^T(n) \;] \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{bmatrix} \end{aligned} \tag{2.43}$$

where

$$\mathbf{h}_1(n) \;=\; [\, h(n) \ldots\ldots h(n-M+1) \,]^T \tag{2.44}$$

$$\mathbf{h}_2(n) \;=\; [\, h(n-M) \ldots\ldots h(n-M-N+1) \,]^T \tag{2.45}$$

$$\mathbf{h}_3(n) \;=\; [\, h(n-M-N) \ldots\ldots h(n-L+1) \,]^T. \tag{2.46}$$

This gives us

$$\begin{aligned} w'(n) &= \mathbf{h}_1^T(n)\mathbf{y}_1 \;+\; \mathbf{h}_2^T(n)\mathbf{y}_2 \;+\; \mathbf{h}_3^T(n)\mathbf{y}_3 \\ e(n) &= d'(n) \;-\; w'(n) \\ &= -w'(n), \qquad\qquad \text{since } \; d'(n) \equiv \; 0 \end{aligned} \tag{2.47}$$

Observe that we know $\mathbf{y}_2$ already. So we don't have to keep it as a part of the unknown coefficient vector. Hence we modify the setup by defining

$$d(n) = -\mathbf{h}_2^T(n)\mathbf{y}_2 \tag{2.48}$$

$$w(n) = \mathbf{h}_1^T(n)\mathbf{y}_1 + \mathbf{h}_3^T(n)\mathbf{y}_3 \tag{2.49}$$

which gives us $e(n)$ as before, but now output $w(n)$ depends only on the unknown vectors $\mathbf{y}_1$ and $\mathbf{y}_3$. This least squares filtering problem is shown in Fig. 2.34. The problem is solved using standard recursive least squares techniques as in [Hay86].

Example 4: We consider the same problem of extrapolating the length 15 sequence to a length 55 sequence, as in the previous examples. Fig. 2.35 shows the frequency response of the highpass filter (of length 41) used as input to the RLS algorithm. Figs. 2.36(a) and (b) show the time domain plot and the magnitude of the Fourier transform of the extrapolated signal respectively.

2.6 Fast Algorithm For Least Squares Bandlimited Extrapolation

In section 2.5, we saw how we could formulate the bandlimited extrapolation problem as a least squares problem, and solve it recursively. Naturally, the next thing to do is to get a fast algorithm for the RLS method described. The problem, the way it is written in section 2.5, is not directly implementable as a fast algorithm because the data matrix involved is not Toeplitz [Hay86]. To be able to get a fast algorithm, we rewrite the problem as a multichannel least squares problem. We can rewrite (2.49) as

$$w(n) \;=\; [\mathbf{h}_1^T(n) \;\; \mathbf{h}_3^T(n)\,] \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_3 \end{bmatrix} \tag{2.50}$$

This equation shows that $w(n)$ can be looked upon as the output of a two-input system. Thus, for the bandlimited extrapolation problem described in section V,

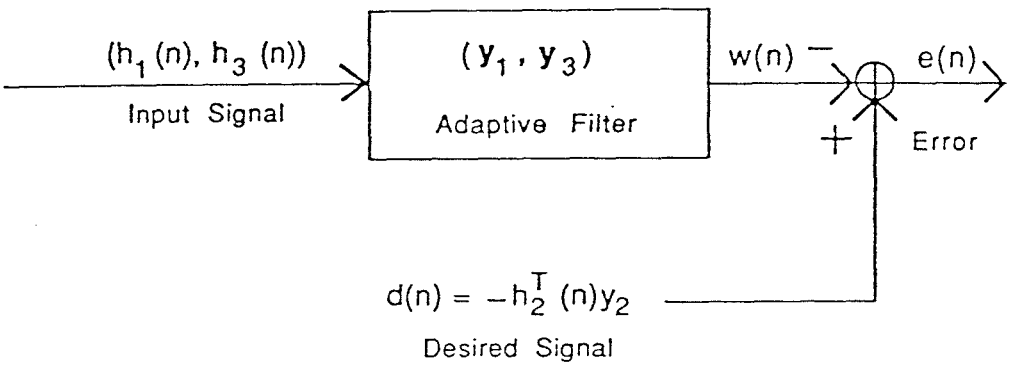


Fig. 2.34 A modification of Fig. 2.33 such that the adaptive filter consists of only the unknown samples.

we have the following *two-input one-output* system to be identified by the least squares technique –

$$w(n) = \mathbf{h}_{2,1}^T(n)\ \mathbf{y}_{2,1} \tag{2.51}$$

$$d(n) = -\mathbf{h}_2^T(n)\ \mathbf{y}_2 \tag{2.52}$$

where

$$\mathbf{h}_{2,1}(n) = [\mathbf{h}_1^T(n)\ \mathbf{h}_3^T(n)]^T \tag{2.53}$$

$$\mathbf{y}_{2,1} = [\mathbf{y}_1^T\ \mathbf{y}_3^T]^T \tag{2.54}$$

We can rearrange the columns of $\mathbf{h}_{2,1}^T$ and the rows of $\mathbf{y}_{2,1}$ so that the set of equations represented by (2.51) for the specified range of n can be written as

$$\mathbf{w} = \mathbf{H}\hat{\mathbf{y}}$$

where $\mathbf{H}$ is a *block Toeplitz* matrix (with 1×2 block size). The vector $\mathbf{w}$ contains the samples $w(n)$ for the appropriate range of n. Similarly we can define the vector $\mathbf{d}$ of the samples $d(n)$. The problem now is to solve for $\hat{\mathbf{y}}$ such that $\mathbf{H}\hat{\mathbf{y}}$ best approximates $\mathbf{d}$ in the least squares sense. This problem can now be solved using a fast multichannel algorithm as in [Kal84]. In the next section, we will see how this idea of converting the extrapolation problem into a multichannel least squares FIR system identification problem can be used to solve multiple burst interpolation/extrapolation problems.

Example 2.5: We take the same extrapolation problem as in Example 2.4 and use the multichannel algorithm [Kal84] to get the extrapolation. For simplicity, it was decided *not* to perform an exact initialization of the fast RLS algorithm (see [Hay86] for meaning of exact initialization).

Fig. 2.37(a) and Fig. 2.37(b) show the time domain plot and the magnitude of the Fourier transform of the resulting extrapolation respectively.

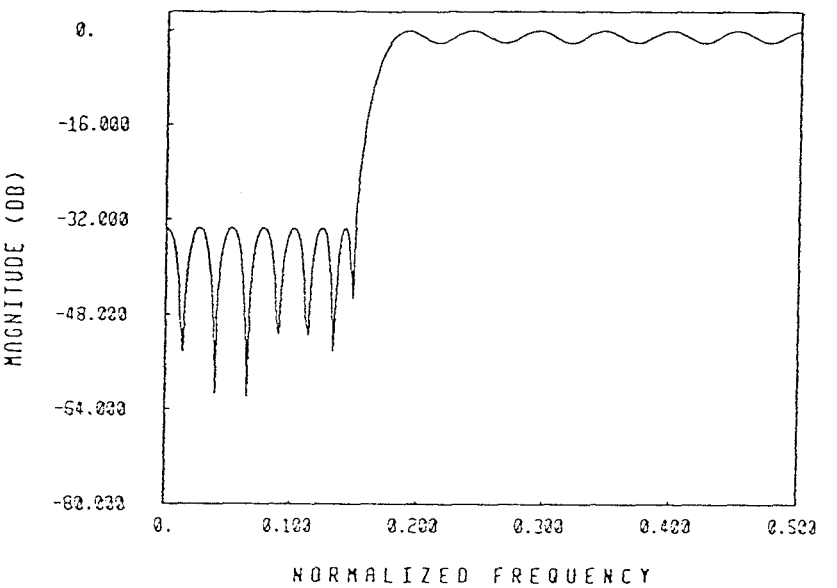
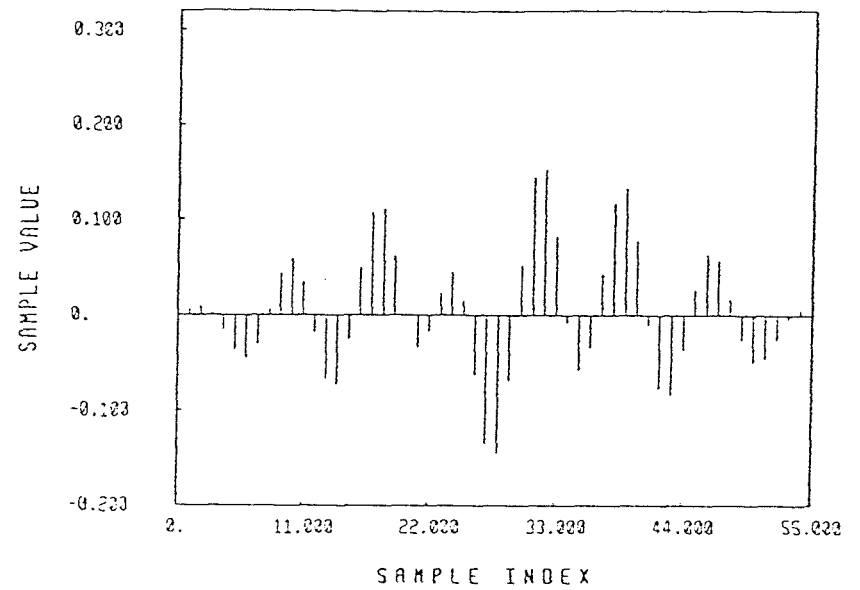
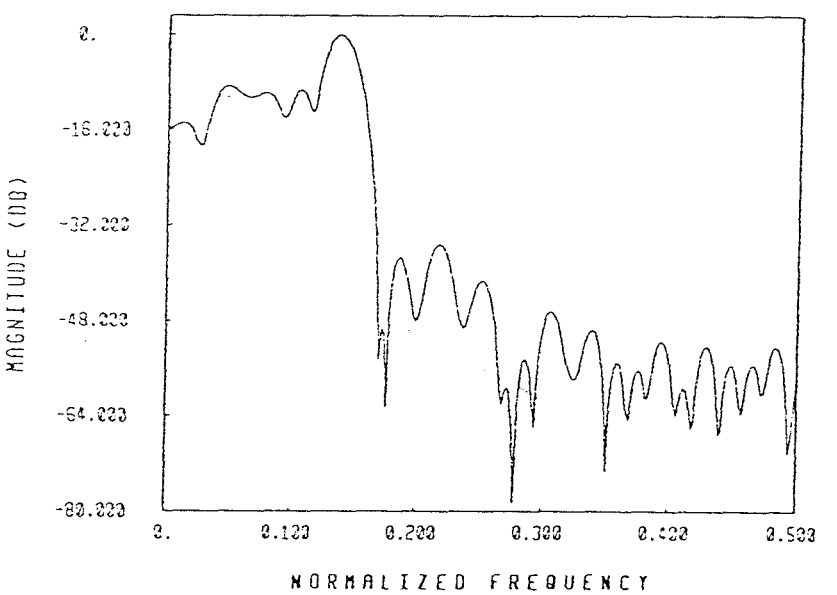


Fig. 2.35 RLS extrapolation method. Frequency response of the input highpass filter.



(a)



(b)

Fig. 2.36 RLS extrapolation method. The extrapolation result (a) time domain (b) magnitude of the Fourier transform.

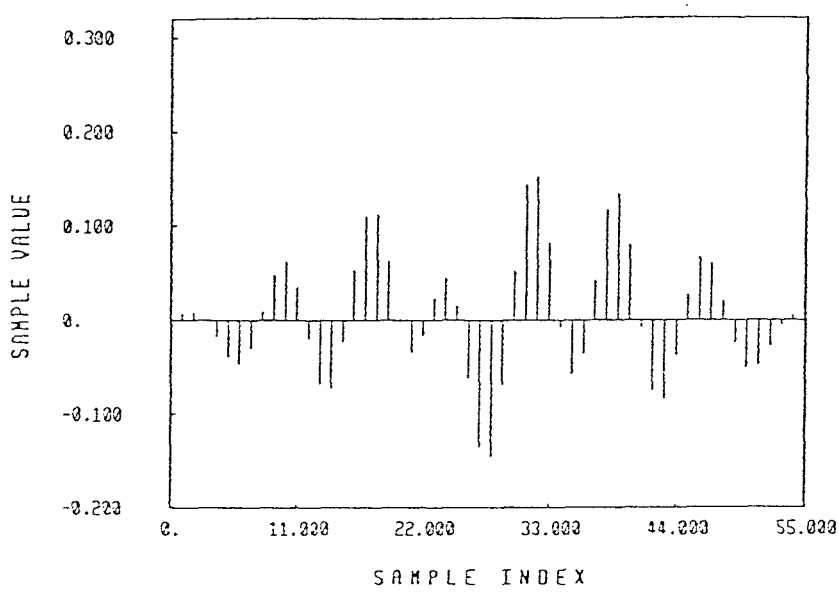
2.7 Multichannel LS-FIR Algorithm For Multiple Burst Interpolation/Extrapolation Problem

The bandlimited extrapolation problem we dealt with so far was to extrapolate a given segment equally on both sides such that the resulting extrapolation was as bandlimited as possible. Another problem of interest is completing a given signal when some of the intermediate samples are erased. Depending on the number of known and unknown samples, we can call this problem either an interpolation or an extrapolation problem. This problem can also be solved as a least squares problem on the same lines as in section 2.6. The basic idea again is to use the impulse response of a filter with the complementary frequency response as the input. The combined output of the system which has the total signal as its impulse response and the impulse response of the complementary response filter as the input is divided into two parts; one due to known signal samples and the other due to unknown signal samples. (In fact, the aim is to calculate the values of these samples.) The part of output due to the known samples is now treated as negative of the desired response. Each unknown segment is treated as an unknown filter response of a channel and a properly delayed input is used for adaptation in each channel.

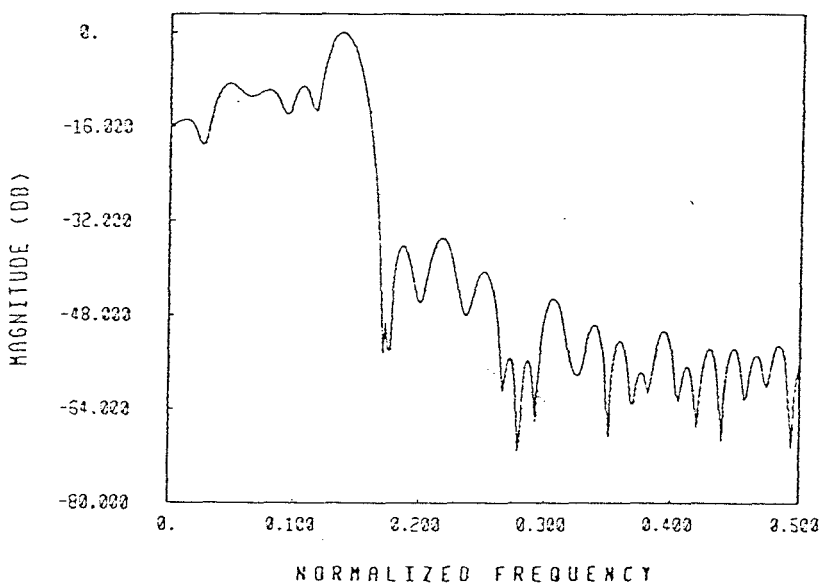
Thus, let $\mathbf{y} = [\, y_0, \ldots\ldots, y_{L-1} \,]^T$ be the total signal of interest of which we don't know some samples. We can write the signal vector in a more convenient form as

$$\mathbf{y} = [\mathbf{y}_0^T \ldots\ldots \mathbf{y}_{K-1}^T \,]^T \tag{2.55}$$

where each vector $\mathbf{y}_i, (\ i = 0, \ldots, K-1\ )$ now corresponds to consecutive known or unknown data samples. We can define an integer function f such that if there are K_1 unknown data vectors in $\mathbf{x}$, then $f(i)$ gives the position of the i^{th} unknown data vector (i.e., the i^{th} erasure) for $1 \leq i \leq K_1$. Correspondingly, we can define a



(a)



(b)

Fig. 2.37 Fast RLS extrapolation method. The extrapolation result. (a) time domain (b) magnitude of the Fourier transform.

complementary integer function f' which gives positions of $K_2 = K - K_1$ known data vectors. For example, for the extrapolation problem described before, we have $K = 3$, $K_1 = 2$, $K_2 = 1$ and $f(1) = 0$, $f(2) = 2$, $f'(1) = 1$.

We split the impulse response vector $h(n)$ defined in eq. (2.42) as

$$\mathbf{h} = [\mathbf{h}_0^T \ldots\ldots \mathbf{h}_{K-1}^T]^T \tag{2.56}$$

such that the dimensions of the vectors $\mathbf{h}_i'$s match those of $\mathbf{y}_i'$s above. We now have a K_1-*input, one-output* unknown system as follows–

$$w(n) = \mathbf{h}_{K_1,1}^T(n)\, \mathbf{y}_{k_1,1} \tag{2.57}$$

$$d(n) = -\sum_{i=1}^{K_2} \mathbf{h}_{f'(i)}^T(n)\, \mathbf{y}_{f'(i)} \tag{2.58}$$

where

$$\mathbf{h}_{K_1,1} = [\mathbf{h}_{f(1)}^T \cdots\cdots \mathbf{h}_{f(K_1)}^T]^T \tag{2.59}$$

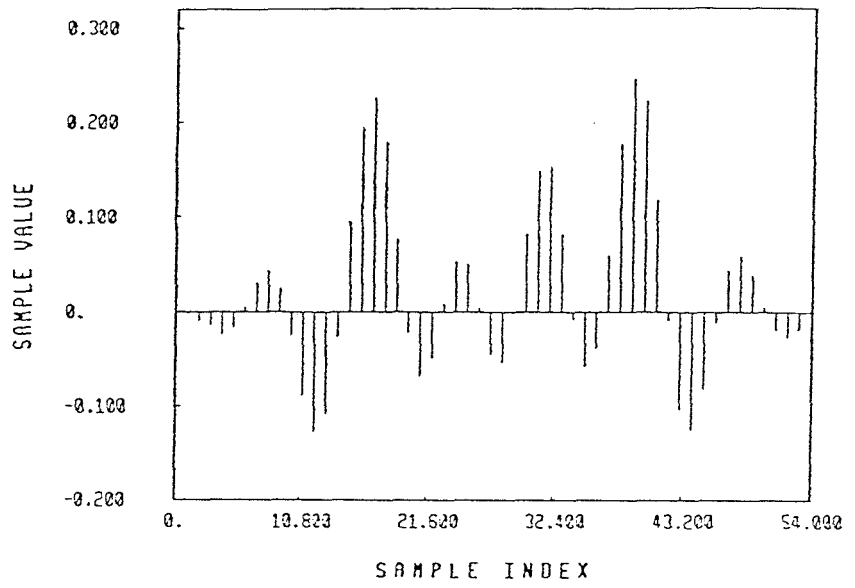
$$\mathbf{y}_{K_1,1} = [\mathbf{y}_{f(1)}^T \cdots\cdots \mathbf{y}_{f(K_1)}^T]^T \tag{2.60}$$

We can rearrange the matrix-vector product in (2.57) by renumbering the columns of $\mathbf{h}_{K_1,1}^T$ and the rows of $\mathbf{y}_{K_1,1}$ so that the set of equations represented by (2.57) (for all values of n in the range of interest) can be expressed in the form

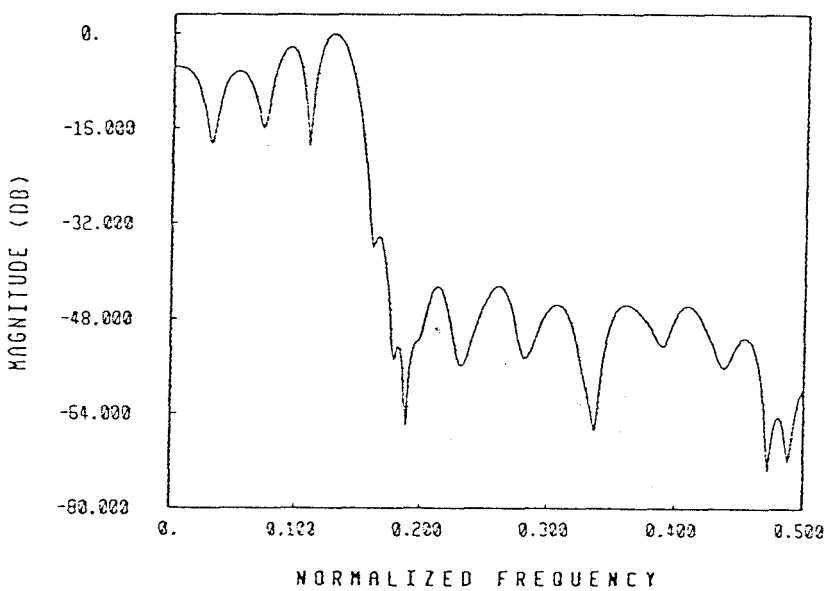
$$\mathbf{w} = \mathbf{H}\mathbf{y} \tag{2.61}$$

where $\mathbf{H}$ is again *block Toeplitz* with block size $1 \times K_1$. Once again we can use the fast multichannel LS FIR identification algorithm [Kal84] to identify the best vector $\hat{\mathbf{y}}$ which matches $\mathbf{d}$ in the least-squares sense. Once we have found out $\mathbf{y}_{K_1,1}$, we can go back and complete the missing samples from the signal $\mathbf{y}$.

Example 2.6: This example considers the problem of extrapolating given 15 points to 54 points. We have a segment , whose first 13 points are unknown, next 5 points



(a)



(b)

Fig. 2.38 Multichannel fast RLS extrapolation method. The extrapolation result. (a) time domain (b) magnitude of the Fourier transform.

are known, next 13 points are unknown, followed by 10 known points followed by 13 unknown points. Hence we have $K = 5, K_1 = 3$ and $K_2 = 2$. The resulting $\pi/3$ bandlimited extrapolation sequence is shown in Fig. 2.38(a), and the corresponding magnitude of the Fourier transform in Fig. 2.38(b).

Example 2.7 : To demonstrate the usefulness of formulating the extrapolation problem as an RLS problem, we give an example where the RLS method is used for the bandpass bandlimited extrapolation. The given segment has 15 samples. Its Fourier transform magnitude is shown in Fig. 2.39. We want to extrapolate this segment equally on both sides so that the result has 55 samples, and we want the result to be bandlimited to a band of width $\pi/3$ symmetrically placed around the frequency $\pi/2$. The bandstop filter used as the input to the RLS method is shown in Fig. 2.40. The filter was designed using the the McClellan-Parks program [McC73] with the passband and the stopband edges at $0.13\pi, 0.38\pi$, 0.21π and 0.29π respectively. The resulting extrapolation and its Fourier transform magnitude plot are shown in Fig. 2.41(a) and Fig. 2.41(b) respectively. The example clearly demonstrates the usefulness of this method. We can get the extrapolation to fit any frequency specifications, as long as we choose the input filter with the complementary specifications.

Noise Performance of RLS and Fast RLS Schemes.

We perform the same simulation as mentioned in Example 2.1. The extrapolation obtained after contaminating the original segment for the RLS scheme is shown in Fig. 2.42. The corresponding extrapolation result for the Fast RLS method is shown in Fig. 2.43. The energy of difference is tabulated in Table 2.1. From these results, we see that both these methods have good noise performance.

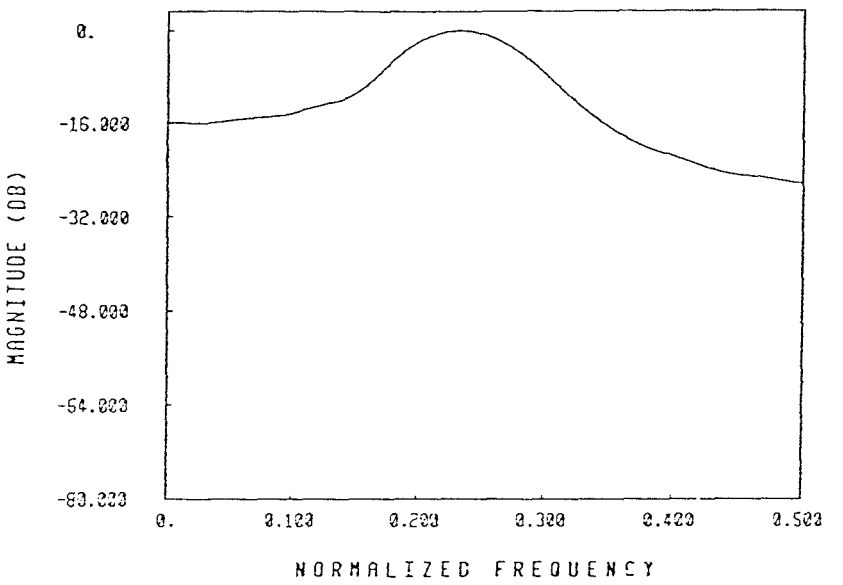


Fig. 2.39 Magnitude of the Fourier transform of the input sequence for bandpass bandlimited extrapolation using the RLS extrapolation method.

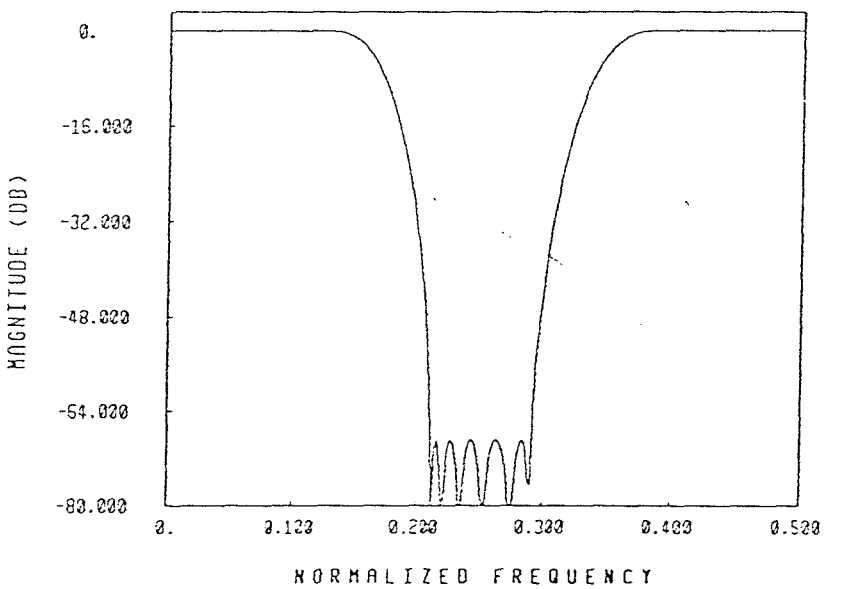
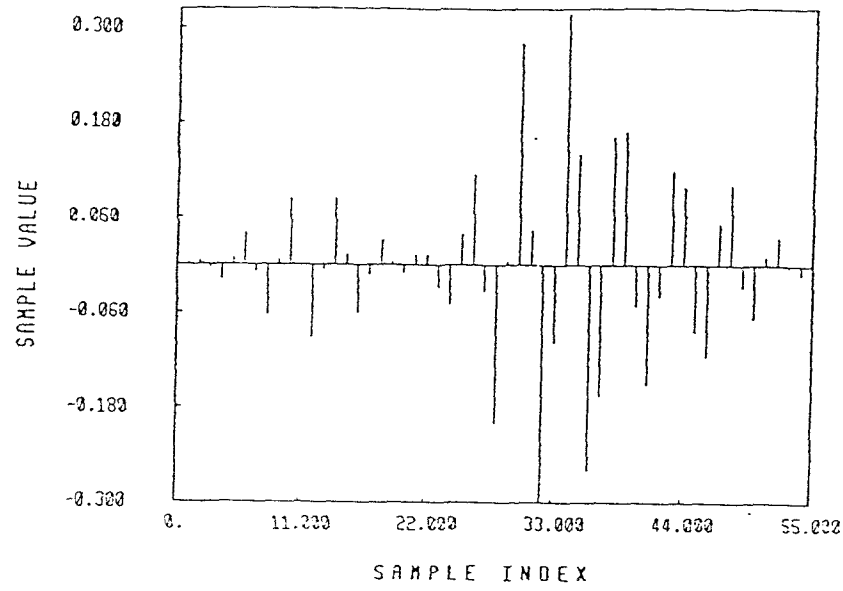
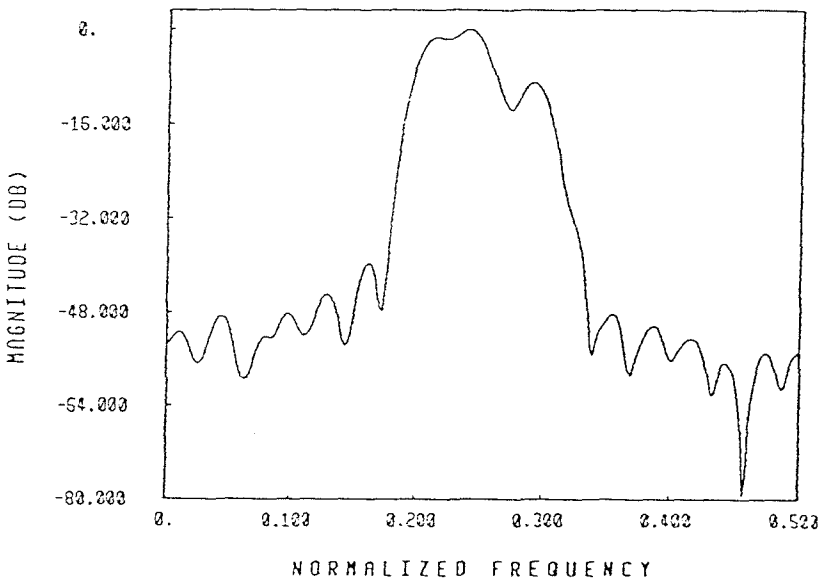


Fig. 2.40 Bandpass bandlimited extrapolation. Frequency response of the input bandstop filter.



(a)



(b)

Fig. 2.41 Bandpass bandlimited extrapolation. The extrapolation result (a) time domain (b) magnitude of the Fourier transform.

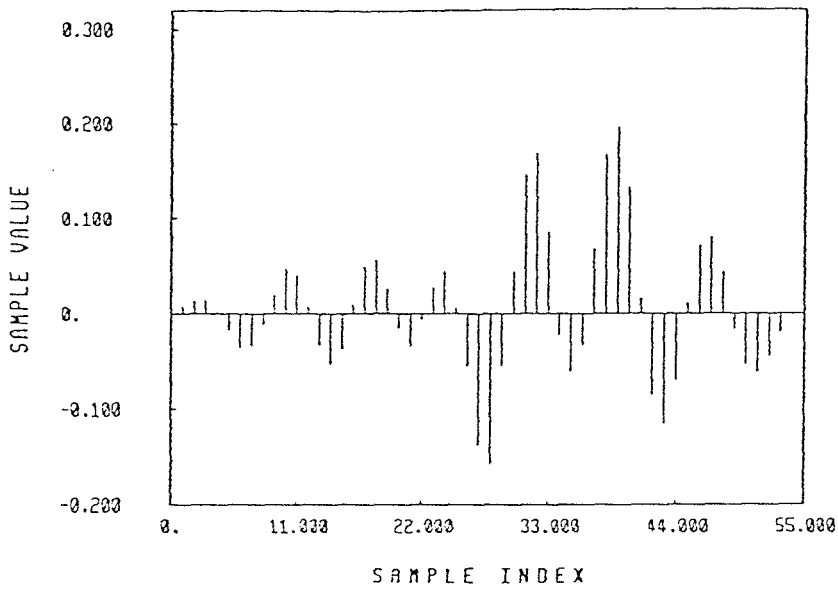


Fig. 2.42 RLS extrapolation method. The extrapolation result with 1% noise added to the original segment.

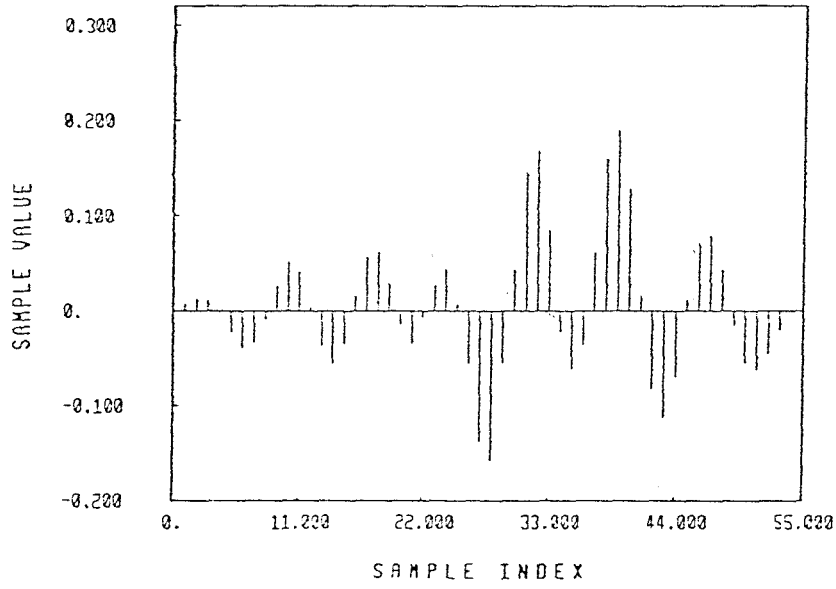


Fig. 2.43 Fast RLS extrapolation method. The extrapolation result with 1% noise added to the original segment.

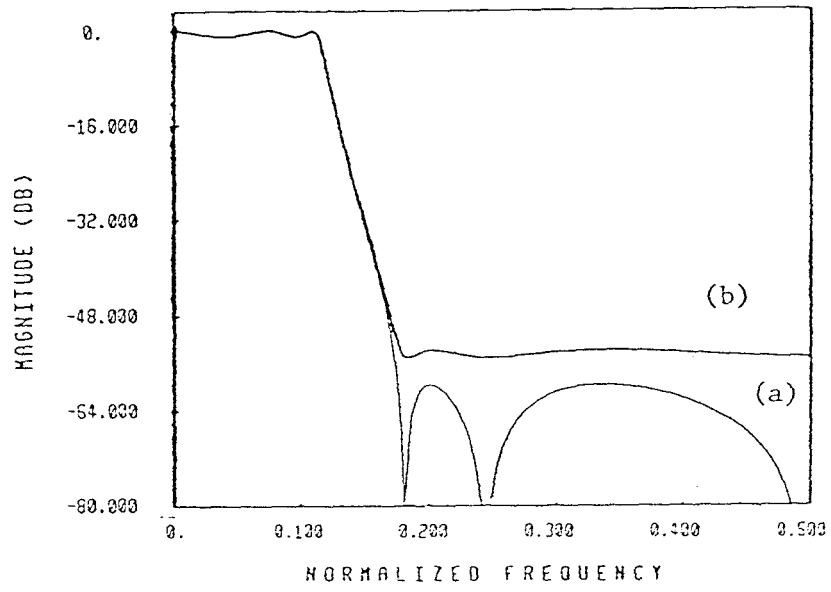


Fig. 2.44 Magnitude of the Fourier transform (a) 60 db attenuation 5^{th} order elliptic filter (b) raised filter, $\theta = 45.1^0$

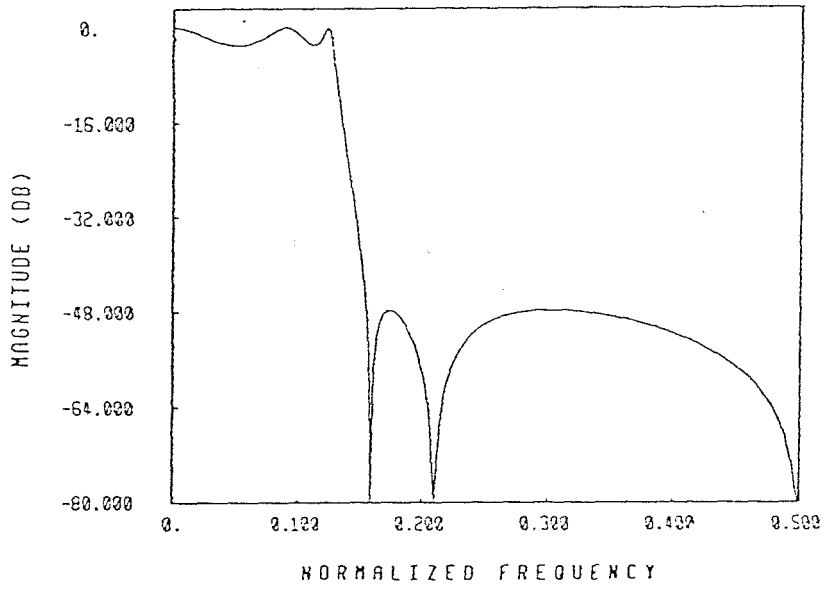


Fig. 2.45 Magnitude of the Fourier transform for the 5^{th} order elliptic filter with the same attenuation as the raised elliptic filter in Fig. 2.44.

i	Strictly Positive Spectrum α_i	Positive Spectrum (Double Zeroes On Unit Circle) α_i
1	-0.1493	-0.1494
2	0.3489	0.3508
3	2.557	2.700
4	-3.685	-5.122
5	-8.165	-9.141
6	-3.758	-3.188
7	5.479	7.404
8	10.08	11.519
9	5.409	4.868
10	-4.489	-6.695
11	-9.583	-11.232
12	-4.670	-3.769
13	5.464	8.198
14	10.292	12.083
15	4.704	3.470
	$\Sigma\alpha_i^2 = 553.61$	$\Sigma\alpha_i^2 = 749.03$

Table 2.4 LPC extrapolation method. α coefficients for power spectrums with different positiveness.

Extrapolation Method	Stopband Energy (SE)%	Optimal SE for same TDE	Time Domain Energy (TDE)%	Optimal TDE for same SE
IIR	0.30	0.0069	33.0	87.3
FIR Method 1	0.0445	0.0092	39.16	69.0
FIR Method 2	0.453	0.0456	69.36	90.4
Pseudoinverse	1.6	1.27	95.3	96.15
LPC	0.025	0.0052	28.0	61.0
Optimal	0.0053	0.0053	28.42	28.42
RLS	0.0297	0.01	47.11	64.0
Fast RLS	0.0349	0.01	46.3	66.2

Table 2.5 Comparison of performance of different extrapolation methods.

Filter	Attenuation	Condition Number
5^{th} order elliptic	60 dB	1.043×10^7
raised 5^{th} order elliptic	50 dB	2.854×10^5
5^{th} order elliptic	50 dB	1.04×10^6

Table 2.6 Condition numbers for 15×15 autocorrelation matrix for elliptic and raised elliptic filters.

2.8 Discussion

In this chapter, we have considered the problem of bandlimited extrapolation of discrete-time sequences. Through our discussions, we have pointed out that the term "best solution" does not have a unique answer here. The reason being that in the digital domain, there exist infinitely many possible extrapolations of a given segment which are bandlimited. However, if we put the additional constraint that the norm of the result be minimum, then the extrapolation is unique. Various methods have been described to obtain a finite-length extrapolation. Since any finite-length extrapolated signal can not be exactly bandlimited, it is obtained by minimizing the out-of-band energy. A technique based on linear predictive lattice is proposed. Similarly, a new technique is proposed for the combined interpolation/extrapolation problem. Two quantities SE and TDE have been proposed to evaluate performance of these schemes. Since it is possible to obtain an "optimal solution" (as described in Section 2.3), the solutions obtained by other methods are compared with the optimal solution by comparing SE and TDE. The methods based on RLS and FRLS algorithms offer good performance and are relatively robust to noise. Another advantage of these methods is that the out-of-band attenuation can be controlled by changing the attenuation of the highpass filter used. The method based on IIR filtering gives extrapolation result that extends to infinite time.

In a noise-free situation, if computational complexity is not the issue, then the optimal solution is the best option. However, in practice, one may want to choose some other algorithm. The choice of a suitable extrapolation scheme will depend on the computational burden that can be handled, the noise performance that is needed or the need for a particular structure that is more convenient for implementation. For example, the LPC extrapolation scheme is convenient to im-

plement if lattice implementation hardware/software is available. In some specific applications like the multiple burst interpolation/extrapolation scheme, only the multichannel RLS extrapolation scheme can be used.

Appendix 2.A Factorization of A = DVP.

The $N \times (2K-1)$ matrix $\mathbf{A}$ in (2.34) has nth row equal to

$$\frac{1}{L}\begin{bmatrix} 1 & a_n & \dots & a_n^{K-1} & a_n^{-(K-1)} & \dots & a_n^{-1} \end{bmatrix} \tag{2.A1}$$

where $a_n = W^{-n}$. This can be rewritten as

$$\frac{1}{L}a_n^{-(K-1)}\begin{bmatrix} 1 & a_n & \dots a_n^{2(K-1)} \end{bmatrix}\mathbf{P} \tag{2.A2}$$

where $\mathbf{P}$ is the permutation matrix given by

$$\mathbf{P} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{K-1} \\ \mathbf{I}_K & \mathbf{0} \end{bmatrix} \tag{2.A3}$$

As a result, the matrix $\mathbf{A}$ can be written as

$$\mathbf{A} = \mathbf{DVP} \tag{2.A4}$$

where $\mathbf{D}$ is $N \times N$ diagonal with ith diagonal entry $a_i^{-(K-1)}/L$, and $\mathbf{V}$ is $N \times (2K-1)$ with elements $\mathbf{V}_{im} = a_i^m$, $0 \le i \le N-1$, $0 \le m \le 2K-2$.

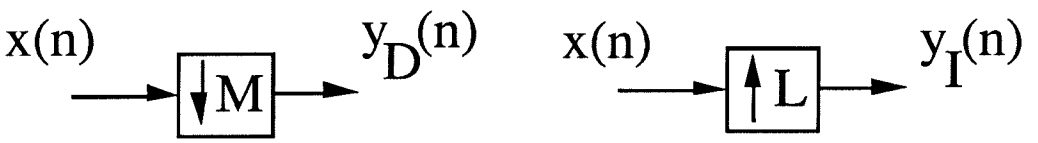
Chapter 3

Analysis Of Effects Of Multirate Systems On The Statistical Properties Of Random Inputs

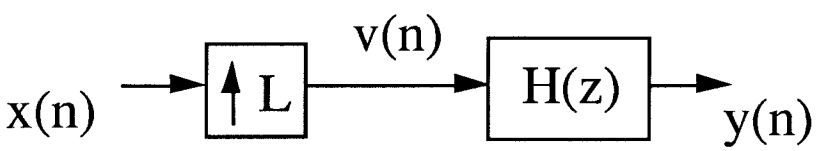
3.1 Introduction

Multirate digital filtering is used in a variety of applications such as sub-band coding, voice privacy systems [Cro83], [Vai90], and adaptive filtering [Gil88], [Sat91a], [Kel88] to name a few. In multirate digital signal processing, we encounter time varying linear systems such as decimators, interpolators, and modulators [Vai90]. In many applications, these building blocks are interconnected together with linear filters to form more complicated systems. Consider for example some of the simple interconnections shown in Fig. 3.1. The M-fold decimator is shown in Fig. 3.1(a). Fig. 3.1(b) shows the L-fold interpolator. Typically, a lowpass filter is used after an interpolator to suppress the images created by passing a signal through the interpolator. This is shown in Fig. 3.1(c). The operation of fractional decimation (or sampling rate conversion) is shown in Fig. 3.1(d).

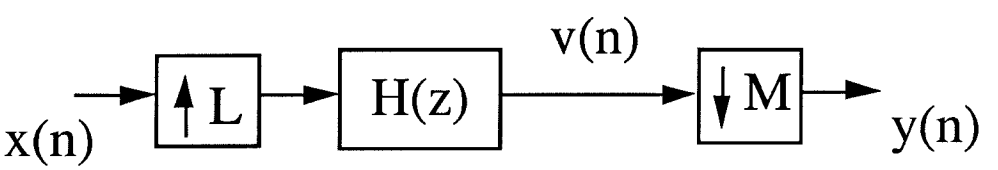
It is often necessary to understand the way in which the statistical behavior of a signal changes as it passes through such systems. While some issues in this context have an obvious answer, the analysis becomes more involved with complicated interconnections. For example, it is easy to see that the decimated version $x(nM)$ of a wide-sense-stationary (WSS) signal $x(n)$ remains WSS. But the following question is more complicated: if we pass a cyclo-wide-sense-stationary (CWSS)



(a) The M-fold decimator (b) The L-fold interpolator



(c) Interpolation filter



(d) Fractional sampling rate changer

Fig. 3.1 Some typical interconnections analyzed in the chapter.

signal [Dav58] with period K through a fractional sampling rate changing device (Fig. 3.1(d)), then what can we say about the stationarity (or otherwise) of the output? Is the answer to this question dependent on whether the lowpass filter is ideal or not, and if so, how?

In this chapter, we answer questions of this nature, starting from a small set of elementary observations. When we make transition from single rate to multirate systems, the assumption that signals are WSS is not valid even in theoretical study. We shall find it more natural to assume that the signals are CWSS. For example, the output of an L-fold interpolation filter to a WSS input is CWSS with period L unless the filter is ideally bandlimited, in which case the output is WSS also. Occurrence of CWSS signals in signal processing and communications applications has been recently discussed in [Gar91]. In Section 3.4 we shall therefore study the effects of multirate filters with reference to CWSS signals.

As an application we consider a novel adaptive filtering structure for identification of bandlimited channels. This scheme is shown in Fig. 4.12. We include a derivation of this scheme in Chapter 4. As we shall see in the next chapter, this structure exploits the bandlimited nature of the channel and embeds the adaptive filter into a multirate system. The advantages are that the adaptive filter has a smaller length and the adaptation as well as the filtering take place at a lower speed resulting in improved computational efficiency. In the theoretical analysis of this system, due to its multirate nature we can not assume that the input to the adaptive filter is WSS (even if the primary input $x(n)$ to the channel is WSS).

Using the theory developed in this chapter, we show that the input to the adaptive filter is CWSS and a matrix adaptive filter gives better performance than a traditional scalar filter. The fact that a matrix adaptive filter is computationally more expensive clearly places in evidence the tradeoff involved when we switch from single rate to multirate systems. The matrix adaptive filter offers a theoretical

performance bound which can not be exceeded by any scalar filter of comparable complexity. Finally, it is shown that if the non-adaptive filters in this system are close to ideal, then the matrix adaptive filter can be replaced by a scalar adaptive filter without a significant loss of performance.

Outline of the Chapter

The chapter is organized as follows. In Section 3.2, we include definitions of various statistical and multirate concepts we use in the paper. The effects of the basic multirate building blocks (decimator, interpolator, modulation) are investigated in Section 3.3. In Section 3.4, we derive similar results for some useful interconnections of the basic building blocks. We conclude the discussion by indicating application of this theory to the adaptation scheme discussed in Chapter 4. Throughout the chapter, all WSS and CWSS processes are assumed to be zero mean.

3.2 Preliminaries

Some basic concepts and definitions from multirate signal processing and system theory are presented in this section.

3.1. M-fold decimator: A decimator is a device which takes an input sequence $x(n)$ and produces the output sequence

$$y_D(n) = x(nM). \tag{3.1}$$

This means that only those samples of $x(n)$ that occur at time equal to multiple of M are retained. In the transform domain, the transfer functions are related as

$$Y_D(e^{j\omega}) = \frac{1}{M}\sum_{k=0}^{M-1} X(e^{-j\omega/M}W_M^{-k}) \tag{3.2}$$

where $W_M = e^{-2j\pi/M}$. Thus, in general passing a signal through a decimator causes aliasing.

3.2. L-fold interpolator: The interpolator takes an input sequence $x(n)$ and produces an output sequence

$$y_I(n) = \begin{cases} x(n/L), & \text{if } n \text{ is an integer multiple of } L; \\ 0, & \text{otherwise.} \end{cases} \tag{3.3}$$

In the frequency domain, we can write

$$Y_I(e^{j\omega}) = X(e^{j\omega L}). \tag{3.4}$$

3.3. Blocking a signal: We call an $M \times 1$ vector signal $\mathbf{x}(n)$ the "M-fold blocked version" of a signal $x(n)$ if they are related by

$$\mathbf{x}(n) = [x(nM)\ x(nM-1) \ldots x(nM-M+1)]^T. \tag{3.5}$$

Using decimators, the blocking mechanism can be shown as in Fig. 3.2(a). The signal $x(n)$ is called the *unblocked version* of the vector process $\mathbf{x}(n)$. The unblocking operation can be represented in terms of multirate building blocks as in Fig. 3.2(b).

3.4. Wide Sense Stationary (WSS) process: A vector stochastic process $\mathbf{x}(n)$ is said to be a wide-sense-stationary process if **(1)** $E[\mathbf{x}(n)] = E[\mathbf{x}(n+k)]$ for all integers n and k and **(2)** the autocorrelation function depends only on the time difference between the two samples, i.e.,

$$E[\mathbf{x}(n)\mathbf{x}^\dagger(n-k)] = \mathbf{R}_{\mathbf{xx}}(k), \qquad \forall n \ \ \forall k. \tag{3.6}$$

3.5. Jointly WSS processes: Two processes $\mathbf{v}(n)$ and $\mathbf{w}(n)$ are said to be *jointly WSS* if the process $\mathbf{u}(n) = [\mathbf{v}^T(n)\ \ \mathbf{w}^T(n)]^T$ is WSS.

3.6. Power Spectral Density: The power spectral density $\mathbf{S}_{\mathbf{xx}}(z)$ of a WSS process $\mathbf{x}(n)$ is defined as the z-transform of its autocorrelation matrix defined in (3.6), i.e.,

$$\mathbf{S}_{\mathbf{xx}}(z) = \sum_{k=-\infty}^{\infty} \mathbf{R}_{\mathbf{xx}}(k) z^{-k}. \tag{3.7}$$

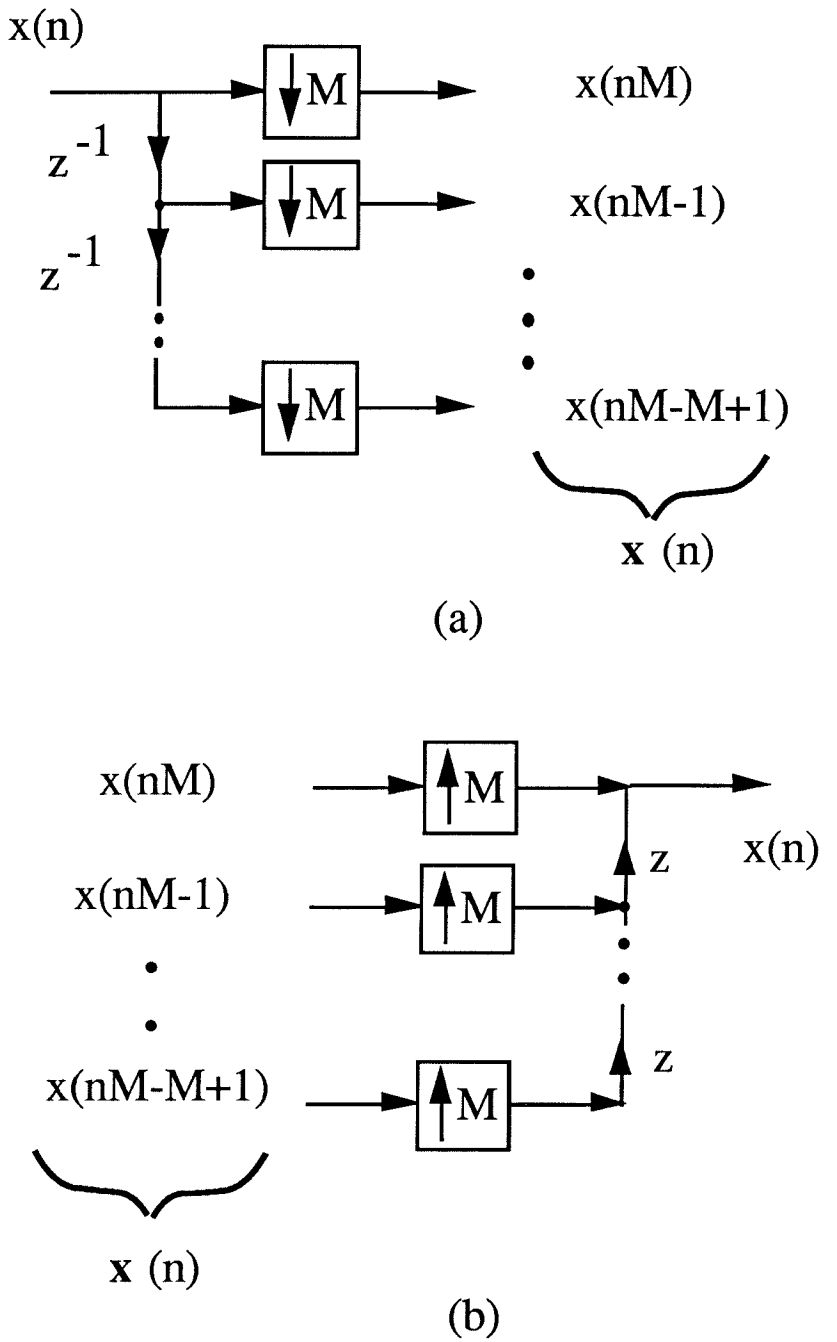


Fig. 3.2 (a) M- fold blocking of a signal
(b) Unblocking of an M x 1 vector signal.

Thus, each entry of this matrix is the z-transform of the corresponding entry of $\mathbf{R}_{xx}(k)$.

3.7. Cyclo-WSS process:

Definition 1: A stochastic process $x(n)$ is said to be a cyclo-WSS process with period L (abbreviated $(\text{CWSS})_\text{L}$), if the L-fold blocked version $\mathbf{x}(n)$ is WSS.

Definition 2: Let $R_{xx}(n,k) = E[x(n)x^*(n-k)]$ denote the autocorrelation function of a process $x(n)$. The process is said to be $(\text{CWSS})_\text{L}$ if $E[x(n)] = E[x(n+kL)]$ for all integers n and k and

$$R_{xx}(n,k) = R_{xx}(n+L,k), \qquad \forall n, \ \forall k. \tag{3.8}$$

A proof of the equivalence of these definitions is given in Appendix 3.A.

3.8. Linear periodically time varying (LPTV) system: A system is said to be LPTV with period L (denoted as $(\text{LPTV})_L$) if the output $y(n)$ in response to input $x(n)$ can be written as

$$y(n) = \sum_{-\infty}^{\infty} h(n,k)x(n-k) \tag{3.9}$$

where

$$h(n,k) = h(n+L,k), \qquad \forall n, \ \forall k. \tag{3.10}$$

An implementation of an $(\text{LPTV})_L$ system is shown in Fig. 3.3. The output at time n is the output of one of L filters depending on the value of (n modulo L).

3.9. Pesudocirculant: An $M \times M$ matrix $\mathbf{A}(e^{j\omega})$ is said to be pseudocirculant if the entries $a_{i,l}(e^{j\omega})$ $(i = 0,..,M-1, \ l = 0,..,M-1)$ satisfy the following relation

$$a_{i,l}(e^{j\omega}) = \begin{cases} a_{0,l-i}(e^{j\omega}) & 0 \le i \le l \\ e^{-j\omega}a_{0,l-i+M}(z) & l < i \le M-1. \end{cases} \tag{3.11}$$

In words, a pseudocirculant matrix is a circulant matrix with elements under the diagonal multiplied by $e^{-j\omega}$. Here is an example of a 3×3 pseudocirculant matrix,

$$\mathbf{A}(e^{j\omega}) = \begin{pmatrix} a_0(e^{j\omega}) & a_1(e^{j\omega}) & a_2(e^{j\omega}) \\ e^{-j\omega}a_2(e^{j\omega}) & a_0(e^{j\omega}) & a_1(e^{j\omega}) \\ e^{-j\omega}a_1(e^{j\omega}) & e^{-j\omega}a_2(e^{j\omega}) & a_0(e^{j\omega}) \end{pmatrix}. \tag{3.12}$$

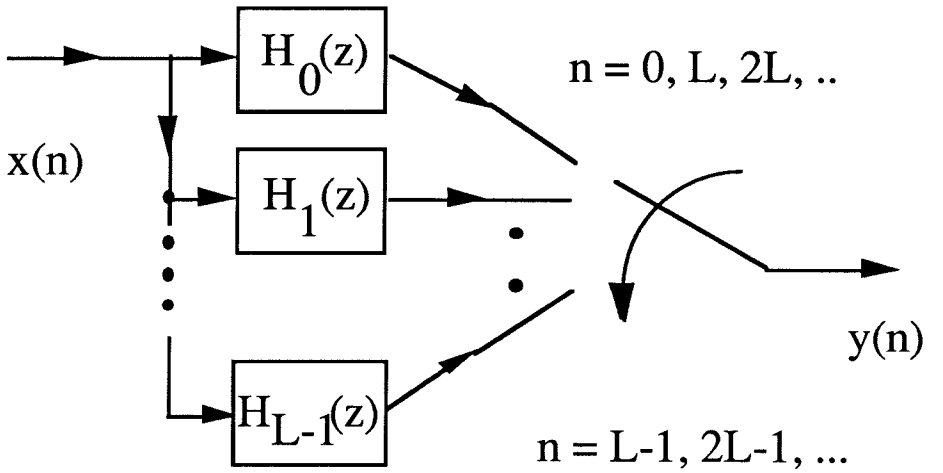


Fig. 3.3 Implementation of an $(\text{LPTV})_L$ system.

We will be using the following two properties of pseudocirculants [Vai88]:

1. If $\mathbf{A}(e^{j\omega})$ is pseudocirculant, then so is $\widetilde{\mathbf{A}}(e^{j\omega})$.
2. If $\mathbf{A}_1(e^{j\omega})$ and $\mathbf{A}_2(e^{j\omega})$ are pseudocirculants, then $\mathbf{A}(e^{j\omega}) = \mathbf{A}_1(e^{j\omega})\mathbf{A}_2(e^{j\omega})$ is also pseudocirculant.

The above definitions and properties hold true in the z-domain using the substitution $z = e^{j\omega}$ if all the z-transforms exist.

3.10. Polyphase Decomposition: Let $X(z)$ be the z-transform of a signal $x(n)$. The polyphase decomposition with respect to M is

$$X(z) = z^{-(M-1)}R_0(z^M) + z^{-(M-2)}R_1(z^M) + \ldots + R_{M-1}(z^M). \qquad (3.13)$$

Each function $R_i(z), 0 \leq i \leq M-1$ is called a polyphase component of $X(z)$. In

the time domain, the kth polyphase component is obtained as

$$r_k(n) = x(nM + M - 1 - k). \tag{3.14}$$

From Fig. 3.2(a), which shows an implementation of blocking the signal $x(n)$ by factor M, it is easy to see that output of each decimator is a polyphase component of the signal with an appropriate shift.

We now tie together seemingly unrelated concepts such as pseudocirculant matrices and wide sense stationarity by proving some interesting relations. These result also bring out the importance of pseudocirculants in the analysis involving WSS signals.

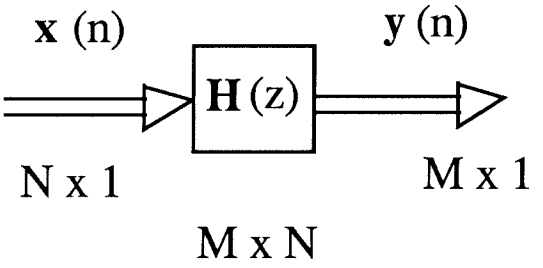


Fig. 3.4 A multi-input, multi-output system.

Fact 3.1 Let $\mathbf{x}(n)$ be an $N \times 1$ vector WSS input to an $M \times N$ transfer matrix $\mathbf{H}(z)$ as shown in Fig. 3.4. Then the power spectral density of the $M \times 1$ vector WSS process $\mathbf{y}(n)$ is given by

$$\mathbf{S_{yy}}(z) = \mathbf{H}(z)\mathbf{S_{xx}}(z)\widetilde{\mathbf{H}}(z). \tag{3.15}$$

Proof: This follows using the convolution expression and the definition of $\mathbf{S}_{yy}(z)$.

Now we prove an important result which relates the pseudocirculant property with wide sense stationarity. The result is complete in the sense that it gives both necessary and sufficient condition.

Fact 3.2: Let $\mathbf{x}(n)$ be the M-fold blocked version of a zero mean stochastic process $x(n)$. Then the following statements are true:

(a) If $x(n)$ is WSS, then the power spectral density $\mathbf{S_{xx}}(z)$ of $\mathbf{x}(n)$ is pseudocirculant.

(b) If, for some M, $\mathbf{x}(n)$ is WSS and $\mathbf{S_{xx}}(z)$ pseudocirculant, then $x(n)$ is a WSS process.

Proof: (a) Let $x(n)$ be WSS. Then, the $(i,l)^{th}$ element of $\mathbf{S_{xx}}(z)$ can be written as

$$[\mathbf{S_{xx}}(z)]_{i,l} = \sum_{k=-\infty}^{\infty} E\big[x(nM-i)x^*(nM-kM-l)\big]z^{-k}. \tag{3.16}$$

For $0 \leq i \leq l$, we can write (3.16) as

$$\begin{aligned} &= \sum_{k=-\infty}^{\infty} E\big[x(nM)x^*(nM-kM-(l-i))\big]z^{-k}. \\ &= [\mathbf{S_{xx}}(z)]_{0,l-i}. \end{aligned} \tag{3.17}$$

For $l < i \leq M-1$, $i-l$ is positive, but $i-l-M$ is negative, hence we can write (3.16) as

$$\begin{aligned} &= \sum_{k=-\infty}^{\infty} E\big[x(nM)x^*(nM-kM+M-(l-i+M))\big]z^{-k} \\ &= \sum_{k=-\infty}^{\infty} E\big[x(nM)x^*(nM-(k-1)M-(l-i+M))\big]z^{-k} \\ &= [z^{-1}\mathbf{S_{xx}}(z)]_{0,l-i+M}. \end{aligned} \tag{3.18}$$

From the definition of a pseudocirculant, we conclude that $\mathbf{S_{xx}}(z)$ is pseudocirculant. We prove in Appendix 3.B that the $(0,m)th$ entry of $\mathbf{S_{xx}}(z)$ is the

$(M-1-m)th$ polyphase component of $S_{xx}(z)$. This implies that we can write down $S_{xx}(z)$ from the $0th$ row of $\mathbf{S_{xx}}(z)$.

(b) If $\mathbf{x}(n)$ is a WSS process, then we can indeed write a valid autocorrelation matrix as in (3.6). The entries of this matrix are

$$\begin{aligned}[\mathbf{R_{xx}}(k)]_{i,l} &= E\big[x(nM-l)x^*(nM-kM-l)\big] \\ &= E\big[x(-i)x^*(-kM-l)\big] \end{aligned} \tag{3.19}$$

The pseudocirculant property of $\mathbf{S_{xx}}(z)$ implies

$$[\mathbf{R_{xx}}(k)]_{i,l} = \begin{cases} [\mathbf{R_{xx}}(k)]_{0,l-i} & l \geq i \geq 0 \\ [\mathbf{R_{xx}}(k-1)]_{0,l-i+M} & l < i \leq M-1. \end{cases} \tag{3.20}$$

Hence we can write,

$$E\big[x(-i)x^*(-kM-l)\big] = E\big[x(0)x^*(-kM+i-l)\big] \quad 0 \leq i,l \leq M-1. \tag{3.21}$$

Now consider $E\big[x(n)x^*(m)\big]$. We can write the time indices as $n = n_0M - i$ and $m = m_0M - l$ where $0 \leq i,l \leq M-1$. So,

$$\begin{aligned} E\big[x(n)x^*(m)\big] &= E\big[x(n_0M-i)x^*(m_0M-l)\big] \\ &= E\big[x(-i)x^*((m_0-n_0)M-l)\big] \\ &= E\big[x(0)x^*((m_0-n_0)M+i-l)\big] \\ &= E\big[x(0)x^*(m-n)\big] \end{aligned} \tag{3.22}$$

Hence $x(n)$ is indeed a WSS process.

Remarks: The above proof holds in particular replacing $z = e^{j\omega}$ for signals for which the z-transform does not exist, but the Fourier transform exists. It is known that if the input to an LTI system is WSS, then the output is also WSS. The right hand side of (3.15) is a product of three matrices. From Fact 3.2, we know that if $x(n)$ is WSS, then both $\mathbf{S_{xx}}(z)$ and $\mathbf{S_{yy}}(z)$ are pseudocirculants. If $\mathbf{H}(z)$ is

pseudocirculant, then it is consistent with (3.15), because $\widetilde{\mathbf{H}}(z)$ is pseudocirculant, and so is the product. The natural question to ask is what does it mean for $\mathbf{H}(z)$ to be pseudocirculant? The next result answers this question.

Fact 3.3: Consider the linear filtering system of Fig. 3.5(a). The signal $y(n)$ is output of the system, to which $x(n)$ is applied as input. Let $\mathbf{x}(n)$ and $\mathbf{y}(n)$ be the corresponding blocked versions. Let us rewrite the input-output relation of the system with $\mathbf{x}(n)$ as input and $\mathbf{y}(n)$ as output (Fig. 3.5(b)) as,

$$\mathbf{y}(n) = \sum_{k=0}^{\infty} \mathbf{h}(k)\mathbf{x}(n-k). \tag{3.23}$$

Let matrix $\mathbf{H}(z)$ be defined as the z-transform of the sequence $\mathbf{h}(k)$. Then $\mathbf{H}(z)$ is pseudocirculant if and only if the original scalar system is LTI.

Proof: See [Vai88].

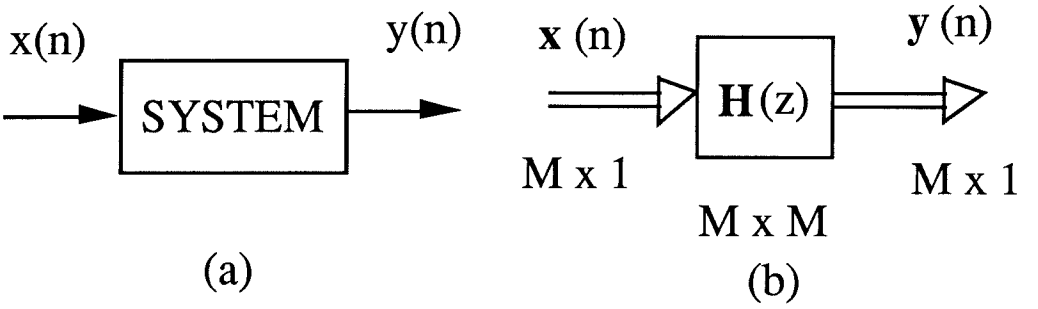


Fig. 3.5 (a) A scalar system
(b) Corresponding blocked version.

This result together with (3.15) is consistent with the fact that the output of an LTI system is WSS if the input is WSS. Finally, we mention a result that gives the solution of an optimal filtering problem involving WSS signals.

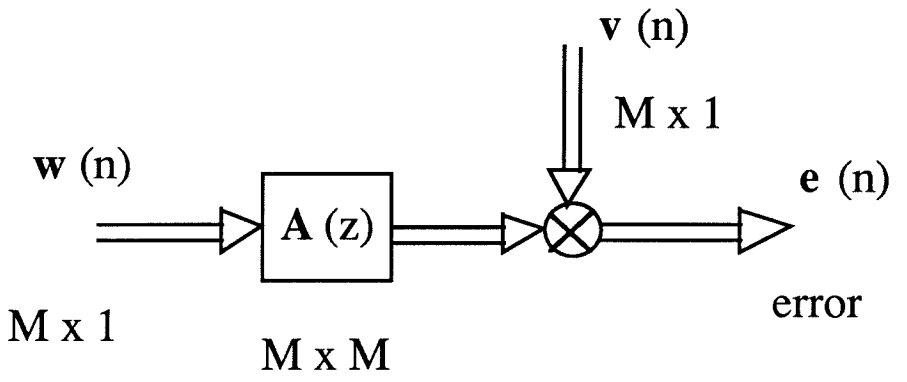


Fig. 3.6 Optimal filtering setup.

Fact 3.4 : Consider the system shown in Fig. 3.6. If the signals $\mathbf{w}(n)$ and $\mathbf{v}(n)$ are jointly-WSS, then the best filter $\mathbf{A}(z)$ in terms of minimizing error variance $E[\mathbf{e}^\dagger(n)\mathbf{e}(n)]$ is given by

$$\mathbf{A}(z) = \widetilde{\mathbf{S}}_{\mathbf{wv}}(z)\mathbf{S}_{\mathbf{ww}}^{-1}(z). \tag{3.24}$$

The matrix $\mathbf{S}_{\mathbf{wv}}(z)$ is the Z-transform of crosscorrelation $\mathbf{R}_{\mathbf{wv}}(k) = E[\mathbf{w}(n)\mathbf{v}^*(n-k)]$. This solution is called the Wiener solution to the problem mentioned above.

Proof: See [Mar86].

We know that if a matrix filter $\mathbf{A}(z)$ is pseudocirculant, then in fact there exists a scalar transfer function corresponding to the unblocked input-output signal description. On the other hand, if the blocked (matrix) transfer function is not pseudocirculant then the corresponding unblocked transfer function is an LPTV system. So for an optimal matrix-filtering problem if the solution given by (3.24) is pseudocirculant, then it is in fact a scalar LTI system. The optimal solution is an LPTV system otherwise. We will use this fact later when we discuss the adaptive filtering application.

3.3 Basic Results

Using the concepts defined in the previous section, we derive some useful results in this section. These pertain to the basic multirate building blocks; an interpolator, a decimator, modulation, and a linear system. First, we prove two results about modulation of a stationary signal by a deterministic signal.

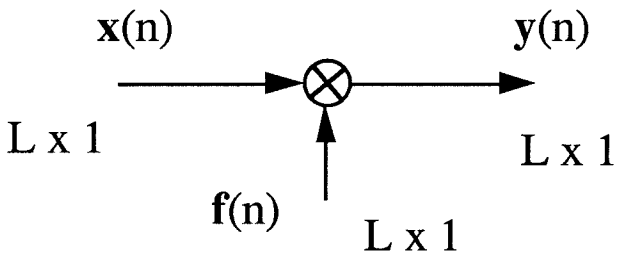


Fig. 3.7 Modulation of signal x(n).

Fact 3.5: Let $\mathbf{x}(n)$ be an $M \times 1$ vector WSS process, modulated by a vector function $\mathbf{f}(n)$ as shown in Fig. 3.7. Then the modulated output $\mathbf{y}(n)$ is WSS if and only if $\mathbf{f}(n)$ is of the type

$$\mathbf{f}(n) = \mathbf{d}e^{j\theta n}, \quad \mathbf{d} \text{ (possibly a complex) constant, } \theta \text{ real.} \tag{3.25}$$

Proof: The "if" part can be easily verified by direct substitution. We prove the "only if" part. Let us represent all the vector quantities in term of their individual components as

$$\mathbf{x}(n) = \begin{pmatrix} x_0(n) \\ \vdots \\ x_{M-1}(n) \end{pmatrix}, \mathbf{f}(n) = \begin{pmatrix} f_0(n) \\ \vdots \\ f_{M-1}(n) \end{pmatrix}, \mathbf{y}(n) = \begin{pmatrix} y_0(n) \\ \vdots \\ y_{M-1}(n) \end{pmatrix}. \tag{3.26}$$

We can write the input output relation as

$$\mathbf{y}(n) = \mathbf{\Lambda}(n)\mathbf{x}(n) \tag{3.27}$$

where $\mathbf{\Lambda}(n)$ is an $M \times M$ diagonal matrix with entries $(f_0(n) \ldots f_{M-1}(n))$ on the diagonal. The autocorrelation function for $\mathbf{y}(n)$ can thus be written as

$$E[\mathbf{y}(n)\mathbf{y}^\dagger(n-k)] = \mathbf{\Lambda}(n)E[\mathbf{x}(n)\mathbf{x}^\dagger(n-k)]\mathbf{\Lambda}^\dagger(n-k). \tag{3.28}$$

This gives the autocorrelation matrix as

$$\mathbf{R_{yy}}(n,k) = \mathbf{\Lambda}(n)\mathbf{R_{xx}}(k)\mathbf{\Lambda}^\dagger(n-k), \tag{3.29}$$

whose $(i,l)^{th}$ entry is therefore

$$[\mathbf{R_{yy}}(n,k)]_{i,l} = f_i(n)[\mathbf{R_{xx}}(k)]_{i,l}f_l^*(n-k). \tag{3.30}$$

For $\mathbf{y}(n)$ to be WSS, we want all the above entries to be free from the time index n. Consider the diagonal elements first. The i^{th} diagonal element will not be a function of n, if and only if $f_i(n)f_i^*(n-k)$ is independent of n, $\forall\ k$. Consider $k = 0$. This implies that $|f_i(n)|^2$ is independent of n. Hence $f_i(n)$ must have the form

$$f_i(n) = c_i e^{j\alpha_i(n)}, \quad \alpha_i(n)\ \ real. \tag{3.31}$$

The $(i,i)^{th}$ element of (3.30) thus becomes

$$[\mathbf{R_{xx}}(k)]_{i,i}f_i(n)f_i^*(n-k) = [\mathbf{R_{xx}}(k)]_{i,i}|c_i|^2 e^{j(\alpha_i(n)-\alpha_i(n-k))}. \tag{3.32}$$

Using the fact that this has to be independent of n, and using a particular value of k ($k = 1$), we get a recursion of the type

$$\alpha_i(n) = \alpha_i(0) + n\beta_i, \quad \beta_i\ \ constant. \tag{3.33}$$

Hence we can rewrite (3.31) as

$$f_i(n) = d_i e^{j\theta_i n}, \quad \theta_i\ \ real, \quad d_i\ \ constant. \tag{3.34}$$

Now, if we use the expression in (3.24) for the $(i,l)^{th}$ element, we get

$$f_i(n)f_l^*(n-k) = d_i d_l^* e^{j(\theta_i n - \theta_l n + \theta_l k)}. \tag{3.35}$$

For this to be independent of n, we should have $\theta_i = \theta_l$ modulo 2π. Summarizing, $f_i(n) = d_i e^{j\theta n}$, so that (3.25) follows.

Remarks: This result implies that translating the power spectrum of a WSS process by different amounts generates processes which are WSS themselves, but are not jointly WSS.

Fact 3.6: Let $x(n)$ be a $(\text{CWSS})_M$ signal. Then the signal $y(n) = f(n)x(n)$ is $(\text{CWSS})_M$ if and only if each polyphase component of the modulating function $f(n)$ with respect to M has the form $\alpha_i e^{j\theta n}$ (α_i possibly complex).

Proof: If $x(n)$ is $(\text{CWSS})_M$, then its blocked version of length M will be a vector WSS process. Similarly, if we block the modulating function $f(n)$, this problem reduces to the setup mentioned in Fact 3.5. Since for the output $y(n)$ to be $(\text{CWSS})_M$ it is necessary and sufficient that the blocked version be vector WSS, the result follows.

Remarks: These results about modulation imply that if we modulate a WSS signal by a cosine wave ($y(n) = x(n)cos\omega_0 n$), then the output $y(n)$ is not WSS even if $x(n)$ is WSS (unless $\omega_0 = 0, \pi$).

We now turn attention to the remaining multirate building blocks.

Fact 3.7: Let $y(n) = x(nM)$, where $x(n)$ is $(\text{CWSS})_L$. Then $y(n)$ is CWSS with period K, where $K = L/gcd(L,M)$. Note the following special cases:

(1) $L = M$. Then $K = 1$ so $y(n)$ is WSS.

(2) $L = 1$ (i.e., $x(n)$ is WSS). Then $K = 1$, and $y(n)$ is WSS.

(3) L and M relatively prime. Then $gcd(L,M) = 1$ and $K = L$ regardless of M.

Proof of Fact 3.7: Using the input-output relation of a decimator, we can write

the autocorrelation for $y(n)$ as

$$E\big[y(n)y^*(n-n_0)\big] = E\big[x(nM)x^*(nM-n_0M)\big]. \tag{3.36}$$

If K is the period of cyclo-wide-sense-stationarity, then

$$E\big[y(n+K)y^*(n+K-n_0)\big] = E\big[y(n)y^*(n-n_0)\big]. \tag{3.37}$$

Thus, from (3.36) we get

$$E\big[x(nM)x^*(nM-n_0M)\big] = E\big[x(nM+KM)x^*(nM+KM-n_0M)\big]. \tag{3.38}$$

Since $x(n)$ is $(\mathrm{CWSS})_L$, (3.38) is satisfied if $KM = lL$ for some integer l. The smallest K is such that all prime factors of L are accounted for by the left hand side. Since M has $\gcd(M,L)$ as the largest factor common with L, we get $K = L/gcd(M,L)$.

Fact 3.8: Passing a $(\mathrm{CWSS})_L$ signal $x(n)$ through an $(\mathrm{LPTV})_\mathrm{L}$ system gives a signal $y(n)$ which is $(\mathrm{CWSS})_L$.

Proof: To prove the result, consider a corresponding blocked system obtained by applying the L-fold blocked input $\mathbf{x}(n)$ to the L-fold blocked version of the $(\mathrm{LPTV})_\mathrm{L}$ system above. We know that blocked version of the $(\mathrm{LPTV})_\mathrm{L}$ system is an LTI system and $\mathbf{x}(n)$ is WSS by definition of cyclo-wide-sense-stationarity. Hence the output of the blocked system will be a WSS signal, and the corresponding unblocked version $y(n)$ will be $(\mathrm{CWSS})_\mathrm{L}$. The result thus follows.

Fact 3.9: If we pass a WSS signal $x(n)$ through an L–fold interpolator, the output $y(n)$ is $(\mathrm{CWSS})_L$. This setup is shown in Fig. 3.1(b).

Proof: Let us block the output $y(n)$ into the vector process $\mathbf{y}(n) = \big[y(nL)\ldots y(nL-L+1)\big]^T$. Since $y(n)$ is interpolated version of $x(n)$, we get $\mathbf{y}(n) =$

$(x(n)\ \ 0 \ldots 0)^T$. Hence

$$E[\mathbf{y}(n)\mathbf{y}^\dagger(n-k)] = \begin{pmatrix} R_{xx}(k) & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{pmatrix}. \tag{3.39}$$

The right hand side is independent of n. This means that $\mathbf{y}(n)$ is a vector WSS process. From the definition of a CWSS process, it follows that $y(n)$ is a $(\text{CWSS})_L$ process.

So far we have seen the effects of basic multirate building blocks on stationary random inputs. Since in the multirate filtering applications, these building blocks are interconnected to form more complex systems, it is of interest to study similar properties for some standard interconnections of these building blocks.

3.4 Results For Interconnections Of The Basic Multirate Building Blocks.

We first consider the operation of L-fold interpolation. We prove that in general the output of this multirate interconnection is not WSS even if the input is.

Fact 3.10: Consider the L-fold interpolation filter shown in Fig. 3.1(c). If $x(n)$ is WSS, then $y(n)$ is $(\text{CWSS})_L$.

Proof: From Fact 3.9, we know that $v(n)$ is $(\text{CWSS})_L$. Hence from Fact 3.8, we can conclude that $y(n)$ is $(\text{CWSS})_L$.

Another important multirate operation is fractional decimation or sampling rate conversion. This is used in a variety of multirate applications. We prove that this operation in general does not produce a WSS output for a WSS input.

Fact 3.11: For the multirate filter shown in Fig. 3.1(d), if input $x(n)$ is WSS, then the output $y(n)$ is $(\text{CWSS})_\text{K}$ where $K = L/gcd(L, M)$.

Proof: From Fact 3.10, we know that $v(n)$ is $(\text{CWSS})_L$. Hence the result follows from Fact 3.6.

Necessary and sufficient condition for wide-sense-stationarity of $y(n)$

We now prove an important result. From Fact 3.10, we know that the output of an interpolation filter in response to a WSS input is CWSS in general. We now find out the necessary and sufficient conditions on the interpolation filter for the output to be WSS.

Let us split $H(e^{j\omega})$ into its polyphase components as follows

$$H(e^{j\omega}) = e^{-j\omega(L-1)}R_0(e^{j\omega L}) + \ldots + e^{-j\omega}R_{L-2}(e^{j\omega L}) + R_{L-1}(e^{j\omega L}). \quad (3.40)$$

We have explained how to obtain the above representation in Definition 3.10.

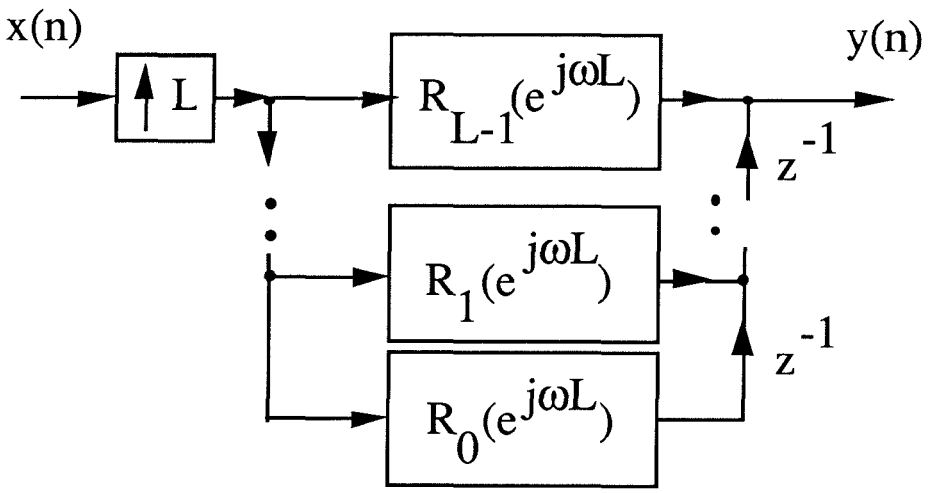


Fig. 3.8 Redrawing Fig. 3.1(c) using polyphase components.

Hence, we can redraw Fig. 3.1(c) as Fig. 3.8. Then using the multirate identity of Fig. 3.9, we can simplify the implementation as shown in Fig. 3.10. Consider the signal $t(n) \triangleq y(n+L-1)$. Fig. 3.11 shows the generation of $t(n)$ from

$x(n)$. It is trivially true that $y(n)$ is WSS if and only if $t(n)$ is WSS. From Fig. 3.11

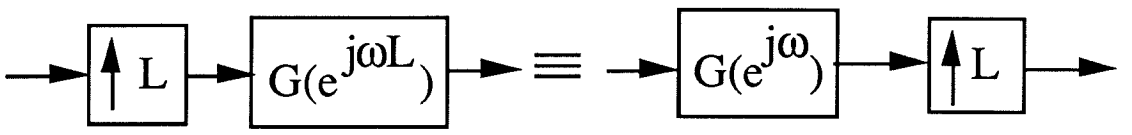


Fig. 3.9 A well-known multirate identity.

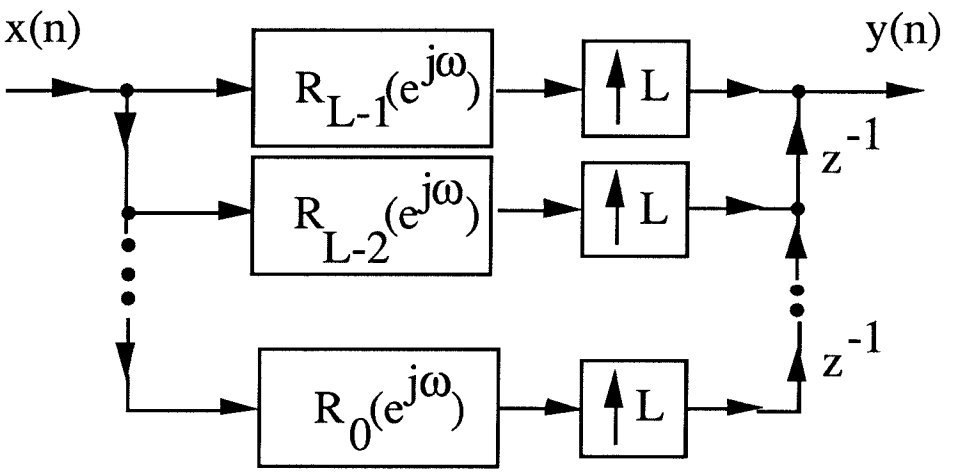


Fig. 3.10 Simplification of the implementation in Fig. 3.8 using the multirate identity.

and Fact 3.2, we know that $t(n)$ is WSS if and only if the blocked version $\mathbf{t}(n) = [t_0(n) \ldots\ t_{L-1}(n)]^T$ is WSS and $\mathbf{S_{tt}}(z)$ is pseudocirculant. Let us define the matrix transfer function $\mathbf{G}(e^{j\omega}) \triangleq \left[R_0(e^{j\omega}) \ \ldots\ R_{L-1}(e^{j\omega})\right]^T$. We now derive the necessary and sufficient conditions for the wide sense stationarity of $t(n)$ in the following steps.

(1) The signal $t(n)$ is WSS if and only if $\mathbf{G}(e^{j\omega})\mathbf{G}^\dagger(e^{j\omega})$ is pseudocirculant.

(2) $\mathbf{G}(e^{j\omega})\mathbf{G}^{\dagger}(e^{j\omega})$ is pseudocirculant if and only if the polyphase components can be represented as

$$R_k(e^{j\omega}) = R_{L-1}(e^{j\omega})e^{j(L-1-k)(\frac{\omega}{L}-\frac{2\pi P(\omega)}{L})} \quad \forall\omega,$$

with the function $P(\omega)$ satisfying the property that $P(\omega + 2\pi) = P(\omega) + 1$ *modulo* L, $\forall\, \omega$.

(3) The above representation is possible if and only if the frequency response $H(e^{j\omega})$ has the property that no aliasing occurs if we perform L-fold decimation of the impulse response $h(n)$. This is equivalent to the following condition (Appendix 3.C): The frequency regions where $H(e^{j\omega})$ is nonzero do not overlap if drawn modulo frequency $2\pi/L$.

Remark: For the interpolation scheme of Fig. 3.1(c), the condition for wide-sense-stationarity of the output for the statistical case is the same as the condition for image-free interpolation for the deterministic case (Appendix 3.C).

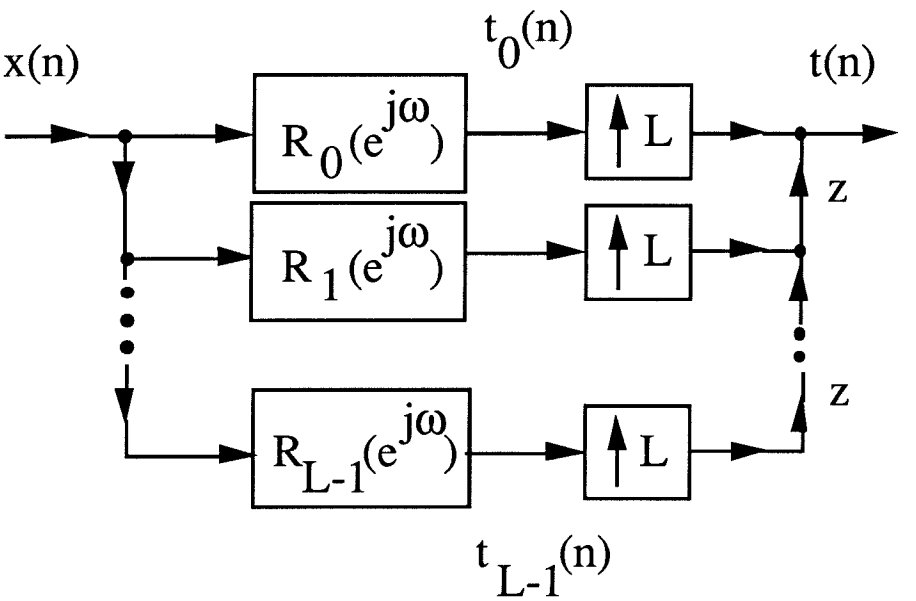


Fig. 3.11 The generation of t(n) from x(n).

Proof of Step 1 :

Since $\mathbf{t}(n)$ is the output of a linear system to which $x(n)$ is the input, it is WSS. Using (3.15) we can write

$$\begin{aligned}\mathbf{S_{tt}}(e^{j\omega}) &= \mathbf{G}(e^{j\omega})S_{xx}(e^{j\omega})\mathbf{G}^{\dagger}(e^{j\omega}) \\ &= S_{xx}(e^{j\omega})\mathbf{G}(e^{j\omega})\mathbf{G}^{\dagger}(e^{j\omega}). \end{aligned} \tag{3.41}$$

Thus, $t(n)$ (and $y(n)$) is WSS if and only if $\mathbf{G}(e^{j\omega})\mathbf{G}^{\dagger}(e^{j\omega})$ is pseudocirculant.

Proof of Step 2 :

The "necessary" part: We first assume that $\mathbf{G}(e^{j\omega})\mathbf{G}^{\dagger}(e^{j\omega})$ is pseudocirculant and prove that the polyphase components are related as above. Using the decimation relation (3.2) we can write

$$R_k(e^{j\omega}) = \frac{1}{L}\sum_{m=0}^{L-1} e^{j(\frac{\omega-2\pi m}{L})(L-1-k)}H(e^{j(\frac{\omega-2\pi m}{L})}),\ k = 0,\ldots,L-1. \tag{3.42}$$

Let us represent the polyphase components in their magnitude-phase form as

$$R_k(e^{j\omega}) = |R_k(e^{j\omega})|e^{j\phi_k(\omega)},\ \ \forall\omega,\ k = 0,\ldots,L-1. \tag{3.43}$$

The $(i,l)^{th}$ entry of $\mathbf{G}(e^{j\omega})\mathbf{G}^{\dagger}(e^{j\omega})$ is given by

$$[\mathbf{G}(e^{j\omega})\mathbf{G}^{\dagger}(e^{j\omega})]_{i,l} = S_{xx}(e^{j\omega})|R_i(e^{j\omega})|\ |R_l(e^{j\omega})|e^{j(\phi_i(\omega)-\phi_l(\omega))}. \tag{3.44}$$

If $\mathbf{G}(e^{j\omega})\mathbf{G}^{\dagger}(e^{j\omega})$ is pseudocirculant all the elements on the main diagonal have to be equal. This gives,

$$|R_k(e^{j\omega})| = |R_{L-1}(e^{j\omega})|,\ \ \forall\omega,\ \ k = 0,\ldots L-1. \tag{3.45}$$

Using the fact that a pseudocirculant is Toeplitz in nature, we can say that the $(k,k+1)st$ elements have to be equal for $k = 0,\ldots,L-2$. This gives the following condition on the phases,

$$\phi_k(\omega) - \phi_{k+1}(\omega) = \phi(\omega)\ \textit{modulo}\ 2\pi,\ \ \forall\omega,\ \ k = 0,\ldots,L-2. \tag{3.46}$$

Using the property of pseudocirculants that the $(1,0)th$ entry is obtained by multiplying the $(0,L-1)st$ entry by $e^{-j\omega}$, we get

$$\phi_0(\omega) - \phi_{L-1}(\omega) = \omega + \phi_1(\omega) - \phi_0(\omega) \quad modulo\ 2\pi \ \ \forall\omega. \tag{3.47}$$

Using the recursion (4.7) we get,

$$\phi_k(\omega) = \phi_{L-1}(\omega) + (L-1-k)\phi(\omega) \ \ modulo\ 2\pi \ \ \forall\omega, \ k = 0,\ldots,L-2. \tag{3.48}$$

Setting $k = 0$ in (3.48) and subtracting form (3.47), we get

$$L\phi(\omega) = \omega \ \ modulo\ 2\pi, \ \ \forall\omega. \tag{3.49}$$

This can be written as an exact functional equality as:

$$L\phi(\omega) = \omega - 2\pi P(\omega) \ \ \forall\omega, \tag{3.50}$$

for an appropriate function $P(\omega)$. Substituting (3.50) in (3.48) we get the following relation among the polyphase components

$$R_k(e^{j\omega}) = R_{L-1}(e^{j\omega})e^{j(L-1-k)(\frac{\omega}{L} - \frac{2\pi P(\omega)}{L})} \ \ \forall\omega, \ k = 0,\ldots,L-1. \tag{3.51}$$

Since $(\frac{\omega}{L} - \frac{2\pi P(\omega)}{L})$ is a phase function, it has to be periodic modulo 2π with period 2π. This gives the following condition on $P(\omega)$

$$P(\omega + 2\pi) = P(\omega) + 1 \ \ modulo\ L, \ \forall\ \omega. \tag{3.52}$$

This proves the "necessary" part.

An example of $P(\omega)$ is shown in Fig. 3.12 for $L = 4$. It can be seen that for this example, $P(\omega)$ takes three different integer values in the interval $0 \leq \omega < 2\pi$. This pattern repeats as per (3.52).

The "sufficient" part: We now prove that (3.51) implies that $\mathbf{G}(e^{j\omega})\mathbf{G}^{\dagger}(e^{j\omega})$ is pseudocirculant. If we assume that (3.51) holds for the polyphase components, then the $(i,l)th$ entry is given by

$$[\mathbf{G}(e^{j\omega})\mathbf{G}^{\dagger}(e^{j\omega})]_{i,l} = S_{xx}(e^{j\omega})|R_{L-1}(e^{j\omega})|^2 e^{j(l-i)(\frac{\omega - 2\pi P(\omega)}{L})}. \tag{3.53}$$

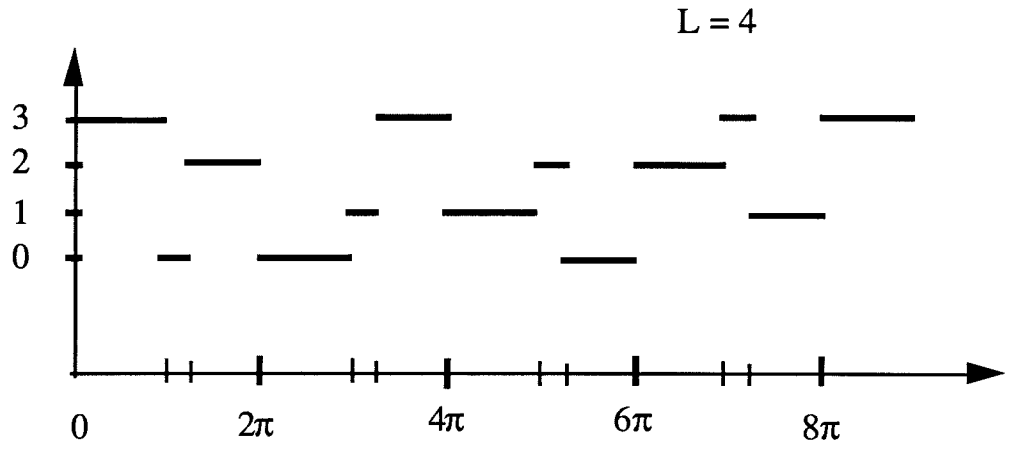


Fig. 3.12 An example of the function P(ω).

It is easy to see that this satisfies the pseudocirculant conditions (3.11).

Proof of Step 3 :

The "necessary" part: We first prove that (3.51) implies the spectral properties mentioned in step 3. Substituting the relation (3.51) in (3.40), we get

$$H(e^{j\omega}) = \sum_{k=0}^{L-1} R_k(e^{j\omega L})e^{-j\omega(L-1-k)}$$

$$= R_{L-1}(e^{j\omega L})e^{-\frac{j2\pi(L-1)P(\omega L)}{L}} \sum_{k=0}^{L-1} W_L^{-kP(\omega L)}. \qquad (3.54)$$

Consider the sum

$$\sum_{k=0}^{L-1} W_L^{-kP(\omega L)} \qquad (3.55)$$

in (3.54). This sum will be nonzero for a frequency ω if and only if $P(\omega L) = 0 \ mod \ L$. Let us study the behavior of $P(\omega L)$ in the interval $0 \leq \omega < 2\pi$. Consider a set of L distinct frequencies $\widehat{\omega}_l$, $l = 0, \ldots, L-1$ in this interval, seperated from each other by integer multiples of $2\pi/L$. Due to the property (3.52), the function P will take on the value (0 *modulo* L) once and only once among the frequencies

$\widehat{\omega}_l$, $l = 0, \ldots, L-1$. The transfer function $H(e^{j\omega})$ can therefore be nonzero only at one such frequency. This means that if for two distinct frequencies ω_1 and ω_2 ($\omega_1 \neq \omega_2$ *modulo* 2π), $H(e^{j\omega_1})$ and $H(e^{j\omega_2})$ are nonzero, then $\omega_1 - \omega_2 \neq \frac{2\pi k}{L}$ unless $k = 0$ *modulo* L. This proves the necessary condition.

A possible $H(e^{j\omega})$ corresponding to the function $P(\omega)$ of Fig. 12 is shown in Fig. 3.13. The filter has nonzero frequency response only in the interval where $P(\omega L)$ has the value (0 *modulo* L).

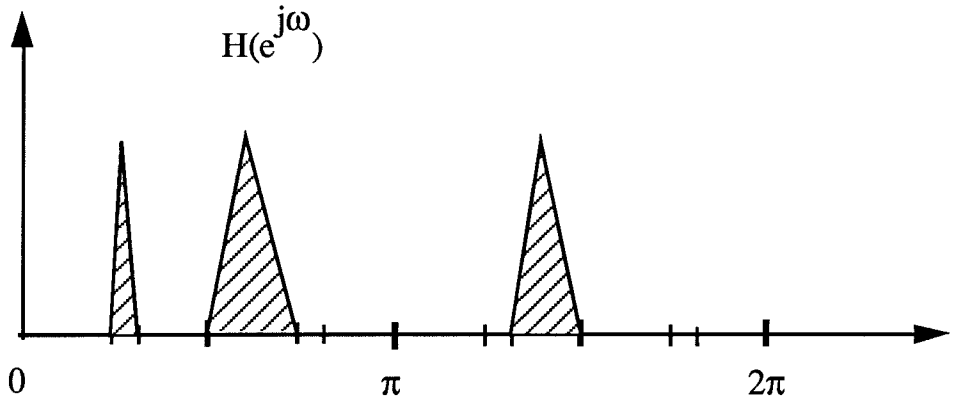


Fig. 3.13 A corresponding possible $H(e^{j\omega})$.

The "sufficient" part: We assume that $H(e^{j\omega})$ has the above-mentioned frequency characteristics and prove that the relation (3.42) holds.

Consider equation (3.42) which shows how each polyphase component $R_k(e^{j\omega})$ is obtained from $H(e^{j\omega})$. Each polyphase component is obtained by adding L images of $H(e^{j\omega})$ stretched in the frequency domain by the factor L and shifted by amount $2\pi m$, $m = 0, \ldots, L-1$. For the particular frequency characteristics of $H(e^{j\omega})$, the stretched and shifted versions of $H(e^{j\omega})$ will not overlap. Hence for each frequency ω, there will only be one nonzero term in the summation (3.42). We can represent the index of that term as a function $m(\omega)$ of the frequency ω.

Furthermore, $m(\omega)$ is independent of k. Hence (3.42) can be rewritten as

$$R_k(e^{j\omega}) = \frac{1}{L} e^{j(\frac{\omega - 2\pi m(\omega)}{L})(L-1-k)} H(e^{j(\frac{\omega - 2\pi m(\omega)}{L})}), \quad k = 0, \ldots, L-1. \tag{3.56}$$

This shows that the polyphase components satisfy the relation (3.51) with $P(\omega) = m(\omega)$. This completes the proof.

We can summarize the results proved above in the following theorem.

Theorem 3.1: The output of Fig. 3.1(c) is WSS for a WSS input $x(n)$ if and only if $H(e^{j\omega})$ is such that the L-fold decimation of its impulse response does not create aliasing.

Remarks: We have shown that for the interpolation filter shown in Fig. 3.1(c), it is possible for $y(n)$ to be WSS. We show in Appendix 3.D that the power spectrum for the process $y(n)$ is given by

$$S_{yy}(e^{j\omega}) = \frac{1}{L} S_{xx}(e^{j\omega L}) \; |H(e^{j\omega})|^2. \tag{3.57}$$

Thus, the power spectrum is in general an L-fold compressed version of $S_{xx}(e^{j\omega})$ possibly piecewise shifted in the frequency domain by frequency $2\pi m/L$ (m integer and possibly different for each piece). This gives rise to an apparent contradiction as follows: Suppose $y_h(n)$ and $y_g(n)$ are two WSS signals generated due the interpolation operation using filters $H(e^{j\omega})$ and $G(e^{j\omega})$. Suppose the filter $H(e^{j\omega}) + G(e^{j\omega})$ does not satisfy the frequency occupancy conditions of Theorem 3.1. Then could it be true that the signal $y(n) = y_h(n) + y_g(n)$, which is a sum of two WSS signals is WSS (in spite of the fact that $H(e^{j\omega}) + G(e^{j\omega})$ does not satisfy the conditions of Theorem 3.1) ? The answer is NO! It can be proved that $y_h(n)$ and $y_g(n)$ are not jointly WSS if the filter $H(e^{j\omega}) + G(e^{j\omega})$ violates the conditions of Theorem 3.1. A proof is included in Appendix 3.E.

Summarizing, we have shown that for a scalar WSS process $x(n)$ and corresponding power spectrum $S_{xx}(e^{j\omega})$, we can perform the following operations on $S_{xx}(e^{j\omega})$ without changing the WSS property at the output:

1. filtering (or "distortion" of the power spectrum),
2. translation by any amount (time domain modulation)
3. stretching (decimation in time domain)
4. compression, and piecewise translation as explained in Theorem 3.1.

Since ideally bandlimited filters can not be implemented in practice, Theorem 3.1 implies that if we use interpolators in a multirate filtering scheme, the subsequent signals will not be WSS (their appropriately blocked versions will be WSS). This fact has some interesting implications. One of these is illustrated in the next chapter.

3.5 Discussion

In this chapter, we have addressed the question of the effects of multirate systems on some statistical properties of random inputs. Starting from the basic multirate building blocks, viz., decimator, interpolator, and modulation, we have derived results for more complex interconnections. We have seen that when we start analyzing multirate systems, the assumption that signals are wide sense stationary is not valid even in theoretical study. It is more natural to assume that signals are CWSS. We proved that output of an interpolator to a WSS input is WSS if and only if it is filtered by an image-free interpolation filter. The output is CWSS otherwise. Similarly, for time-domain modulation of a WSS signal, we have proved that the output is WSS if and only if the signal is modulated by a harmonic signal.

These results are useful when one analyzes systems involving multirate filters. For example, for the case of multirate adaptive fitlers, one needs these results to be able to write down the optimal filter solution. In the next chapter, we illustrate an application of the theoretical analysis to a novel multirate adaptive filtering scheme for identification of bandlimited channels. This scheme exploits the fact that the

channel is bandlimited and embeds the adaptive filter in a multirate configuration. Using the results derived in this chapter, we prove that the optimal filter for this scheme is a matrix filter. Using a matrix adaptive filter is computationally more expensive. However, performance of a scalar adaptive filter (which is computationally less expensive) tends to be close if the fixed lowpass filters in the multirate scheme are designed to have good stopband attenuation.

Appendix 3.A

A proof of the equivalence of the definitions:

1) Definition 1 $\Rightarrow$ Definition 2: From Definition 1, the $(i,j)^{th}$ entry of the autocorrelation matrix is indepedent of n. Hence

$$E\big[x(n_0L-i)x^*(n_0L-k_0L-j)\big] \tag{3.A1}$$

is independent of n_0, for all k_0, for $0 \leq i,j \leq L-1$. Now let $n = n_0L - i$ and $k = k_0L + j$. Then

$$E\big[x(n)x^*(n-k)\big]$$

$$= E\big[x(n_0L-i)x^*(n_0L-k_0L-i-j)\big]$$

$$= E\big[x((n_0+1)L-i)x^*((n_0+1)L-k_0L-i-j)\big]. \tag{3.A2}$$

(since the expression is independent of n_0)

$$= E\big[x(n+L)x^*(n+L-k)\big] \tag{3.A3}$$

so that $R_{xx}(n,k) = R_{xx}(n+L,k)$.

1) Definition 2 $\Rightarrow$ Definition 1: The $(i,j)^{th}$ entry of the autocorrelation matrix in Definition 1.

$$E\big[x(nL-i)x^*(nL-kL-j)\big]$$

$$= E\big[x(nL+L-i)x^*(nL+L-kL-j)\big]. \ (by\ Definition\ 2) \tag{3.A4}$$

$$= E\big[x((n+1)L-i)x^*((n+1)L-kL-j)\big]. \tag{3.A5}$$

Clearly, the $(i,j)^{th}$ entry is independent of n.

Equivalence of Definition 2 and Definition 3 can be established by a similar proof by replacing the variables i and j in the above proof by $-i$ and $-j$ respectively.

Appendix 3.B

Reproducing (3.17) for the $(0,m)th$ entry of $\mathbf{S_{xx}}(z)$ we get

$$[\mathbf{S_{xx}}]_{0,m} = \sum_{k=-\infty}^{\infty} E\big[x(nM)x^*(nM - kM - m)\big]z^{-k}. \qquad (3.B1)$$

If $R_{xx}(k)$ is the autocorrelation function of $x(n)$, then (3.B1) can be written as

$$[\mathbf{S_{xx}}]_{0,m} = \sum_{k=-\infty}^{\infty} R_{xx}(kM + m)z^{-k}. \qquad (3.B2)$$

From (3.14) we can clearly see that this is the z-transform of the $(M-1-m)th$ polyphase component. Thus, the $(0,m)th$ entry of $\mathbf{S_{xx}}(z)$ is the $(M-1-m)th$ polyphase component of $S_{xx}(z)$.

Appendix 3.C

We first prove that if the signal $h(n)$ has the spectral characteristics mentioned in Theorem 3.1, then L-fold decimation of the signal does not result in aliasing. In the frequency domain, the decimation process creates L images of the original function stretched by the factor L, shifted by $2\pi m$, $m = 0,\ldots,L-1$ and added. Clearly, no aliasing results for the spectral conditions of Theorem 3.1.

Converesely, if no aliasing results after L-fold decimation, then the L-fold stretched and $2\pi m$ shifted versions do not overlap. This in turn implies that the frequency response has maximum $2\pi/L$ spectral occupancy and its $2\pi m/L$, $m = 0,\ldots,L-1$ frequency shifted versions do not overlap. This is precisely the condition in Theorem 3.1.

Remarks: A filter $H(e^{j\omega})$ with the frequency characteristics as above has some special properties in the deterministic multirate filtering case. The filter can be used before the L-fold decimator as shown in Fig. 3.14(a) to ensure that no aliasing takes place after decimation. Similarly, filtering output of the L-fold interpolator by $H(e^{j\omega})$ as shown in Fig. 3.14(b) suppresses unwanted images. In a traditional image suppressing filter, this is achieved by filtering out $L-1$ images and retaining one of the L images. However, for the filter $H(e^{j\omega})$, the output power spectrum will in general consist of different pieces from different images such that these pieces can be put together by frequency translation and an L-fold compressed version of the input power spectrum can be obtained. Thus, $H(e^{j\omega})$ is a generalized image suppressing filter.

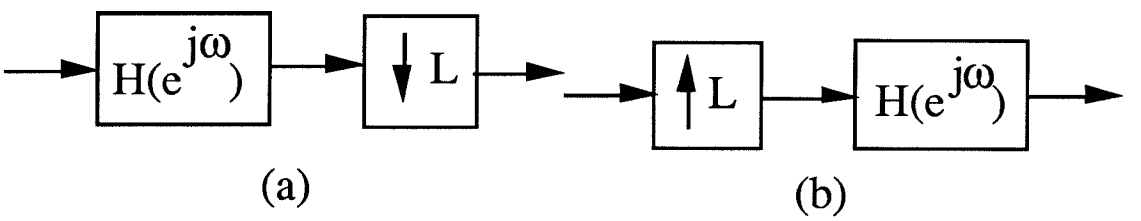


Fig. 3.14 Deterministic case.
(a) Alias-free decimation.
(b) Image-free interpolation.

Appendix 3.D

Derivation of the output power spectral density expression:

We assume that $y(n)$ (and hence $t(n)$) is WSS. We derive the expression for $S_{tt}(e^{j\omega})$ $(= S_{yy}(e^{j\omega}))$. In Appendix 3.B, we have proved a result which implies that the entries of the $0th$ row of the matrix $\mathbf{S_{tt}}(e^{j\omega})$ are the polyphase components of

$S_{tt}(e^{j\omega})$. Using this and the expression (3.44) we can write,

$$\begin{aligned} S_{tt}(e^{j\omega}) &= \sum_{l=0}^{L-1} R_0(e^{j\omega L}) R_l^*(e^{j\omega L}) S_{xx}(e^{j\omega L}) e^{-j\omega l} \\ &= R_0(e^{j\omega L}) S_{xx}(e^{j\omega L}) \Big(\sum_{l=0}^{L-1} R_l(e^{j\omega L}) e^{j\omega l} \Big)^* \\ &= R_0(e^{j\omega L}) S_{xx}(e^{j\omega L}) e^{-j\omega(L-1)} \Big(\sum_{l=0}^{L-1} R_l(e^{j\omega L}) e^{-j\omega(L-1-l)} \Big)^* \\ &= R_0(e^{j\omega L}) S_{xx}(e^{j\omega L}) e^{-j\omega(L-1)} H^*(e^{j\omega}). \end{aligned} \tag{3.D1}$$

Since

$$R_0(e^{j\omega}) = R_{L-1}(e^{j\omega}) e^{j(L-1)\omega} \; W_L^{P(\omega)(L-1)}, \tag{3.D2}$$

we can rewrite (3.D1) as

$$S_{tt}(e^{j\omega}) = S_{xx}(e^{j\omega L}) H^*(e^{j\omega}) R_{L-1}(e^{j\omega}) W_L^{P(\omega)(L-1)}. \tag{3.D3}$$

Using (3.53) we can write

$$S_{tt}(e^{j\omega}) = S_{xx}(e^{j\omega L}) \; |R_{L-1}(e^{j\omega L})|^2 \sum_{l=0}^{L-1} W_L^{-lP(\omega L)}. \tag{3.D4}$$

Using the fact that the sum (3.55) that appears in the above equation can only take two values (0 and L), we can write this down as

$$S_{tt}(e^{j\omega}) = S_{yy}(e^{j\omega}) = \frac{1}{L} S_{xx}(e^{j\omega L}) |H(e^{j\omega})|^2. \tag{3.D5}$$

Appendix 3.E

We have to prove that if the filters $H(e^{j\omega})$ and $G(e^{j\omega})$ are such that the filter $H(e^{j\omega}) + G(e^{j\omega})$ does not satisfy the frequency characteristics necessary for wide-sense-stationarity in Theorem 4.1, then the outputs $y_h(n)$ and $y_g(n)$ are not jointly

WSS. If $H(e^{j\omega})+G(e^{j\omega})$ does not satisfy the frequency conditions, then at least one of the frequency components of $G(e^{j\omega})$ is obtained by translating an appropriate frequency component of $H(e^{j\omega})$ in the frequency domain by an integer multiple of $2\pi/L$ frequency (and passing it through a bandpass filter for a possible change in shape). Hence it is enough to prove that for Fig. 3.1(c), a $2\pi/L$ frequency translated version of $H(e^{j\omega})$ gives an output which is not jointly WSS with the output of $H(e^{j\omega})$.

To prove this, let us assume that $G(e^{j\omega})$ is obtained by frequency shifting $H(e^{j\omega})$ by $2\pi\ell/L$ (M integer). Then the polyphase components $F_k(e^{j\omega})$ of $G(e^{j\omega})$ obtained as in Definition 3.10 satisfy

$$F_k(e^{j\omega}) = W_L^{-M(L-1-k)} R_k(e^{j\omega}). \qquad (3.E1)$$

We prove by contradiction that $y_h(n)$ and $y_g(n)$ are not jointly WSS. If the signals are jointly WSS, then in particular

$$E[y_h(nL)y_g^*(nL-mL+k)] = E[y_h(nL+1)y_g^*(nL-mL+k+1)],\ \forall\ m. \quad (3.E2)$$

Taking the Fourier transform of both the sides (with respect to the index m) and using the appropriate polyphase components for the power spectral density expressions,we get

$$R_0(e^{j\omega})F_k^*(e^{j\omega}) = R_1(e^{j\omega})F_{k+1}^*(e^{j\omega}). \qquad (3.E3)$$

If we substitute $(3.E1)$ in this we get

$$R_0(e^{j\omega})R_k^*(e^{j\omega})W_L^{\ell(L-1-k)} = R_1(e^{j\omega})R_{k+1}^*(e^{j\omega})W_L^{\ell(L-1-k)}\ W^{-\ell}. \qquad (3.E4)$$

Using (3.51) we can show that $(3.E4)$ is not true, hence $(3.E2)$ does not hold. This implies that the signals $y_h(n)$ and $y_g(n)$ can not be jointly WSS. This proves the result.

Chapter 4

Adaptive Identification of Bandlimited Channels Using Multirate/Multistage FIR Filters

System identification is an important problem in digital signal processing [Hya86]. In adaptive system identification, we assume that the system is an FIR or an IIR filter and try to identify the system parameters. In our discussion, we are going to assume that the unknown system can be represented as an FIR filter. A typical system identification setup is shown in Fig. 4.1. From the known input $x(n)$ and output $y(n)$ up to a certain time n, we try to estimate the system parameters. In applications like echo cancellation, rather than identifying the system, one is interested in cancelling the effect of the output of the unknown system by setting up an adaptive filter in parallel [Lee85]. There are well-known adaptive algorithms such as the LMS algorithm [Wid85] and the fast RLS (FRLS) algorithm [Cio84] which minimize the sum of error squares at the output. The computational count per input data sample is about $2L_a$ for the LMS and $7L_a$ for the FRLS algorithm where L_a is the length of the adaptive FIR filter used in the identification/cancellation scheme. In real-time applications, the adaptive computations have to be performed in the time between two input samples. Some applications such as High Definition Television (HDTV) [Jur91] or the teleconferencing application discussed later need the adaptive filter to have long lengths. This requires the signal processing hardware to be fast to handle high speed input data. A natural question to ask is if we can somehow reduce the speed and amount of computations if we have some additional

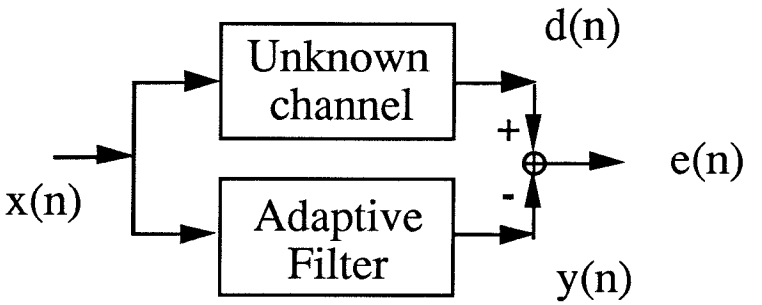


Fig. 4.1 Adaptive Channel Identification Scheme

information about the system we are trying to identify. A scheme to reduce speed of computations by blocking the input has been reported in [Cla81]. In this method, the input signal is split in M sub-bands by passing it through a filter bank or by taking the FFT of the blocked input signal. Adaptation is performed at a reduced speed on separate adaptive filters in each sub-band. For long lengths of adaptive filters, some improvement in the computational count has also been reported in [Cla81]. This method however does not use any information about the unknown system. Can we achieve any further savings if we know some more information about the unknown system? In this chapter, we show that we can indeed achieve computational savings if we know that the unknown system is bandlimited. We can make use of *a priori* knowledge about bandlimitedness of the channel to achieve this reduction which can be very high if the system is bandlimited to a small frequency.

If a signal is bandlimited in the digital domain, it means that the signal is over-sampled. So we can discard some of the samples and still retain all the information about the signal. We can use this fact to reduce the computational complexity for the identification/cancellation of an unknown bandlimited channel. The implemen-

tation naturally leads to splitting up the adapative filter as an adaptive part and a nonadaptive part. The adaptive part is embeded in multirate filters. This not only reduces the number of computations but also the speed at which the adaptive algorithm is performed.

In Section 4.1, we take a look at some applications where the unknown system is bandlimited. We briefly discuss the current techniques which reduce the computational complexity of the adaptive identification using the bandlimitedness information. In Section 4.2, we develop the new method assuming that the unknown channel is bandlimited to $\pi L/M$ (L and M integers). We will see how fractional decimation is used to reduce the speed of computation. A brief discussion on the implementation aspects is also included. The Wiener filter solution for the scheme presented is derived in section 4.3. We show that the optimal filter is an LPTV system. However, simulations presented in section 4.4 indicate that a transversal adaptive filtering structure results in no perceptible degradation in the performance over a more general matrix adaptive filter, if the lowpass filters are designed to have good stopband attenuation.

4.1 Examples Of Bandlimited Systems

Bandlimited systems have been mentioned in the literature in a variety of applications. It is instructive to study some the adaptive filtering applications involving bandlimited systems. In this section, we describe two such applications. Some other applications are mentioned in [Wid85, Chapter 13].

4.1.1 Acoustic Echo Cancellation in Teleconferencing

In the teleconferencing application, a local room and a distant room are connected through audio and video transmission. We are interested in the audio transmission part of the application. This is schematically shown in Fig. 4.2. We have not included such implementation details as the analog-to-digital converters in this

not included such implementation details as the analog-to-digital converters in this

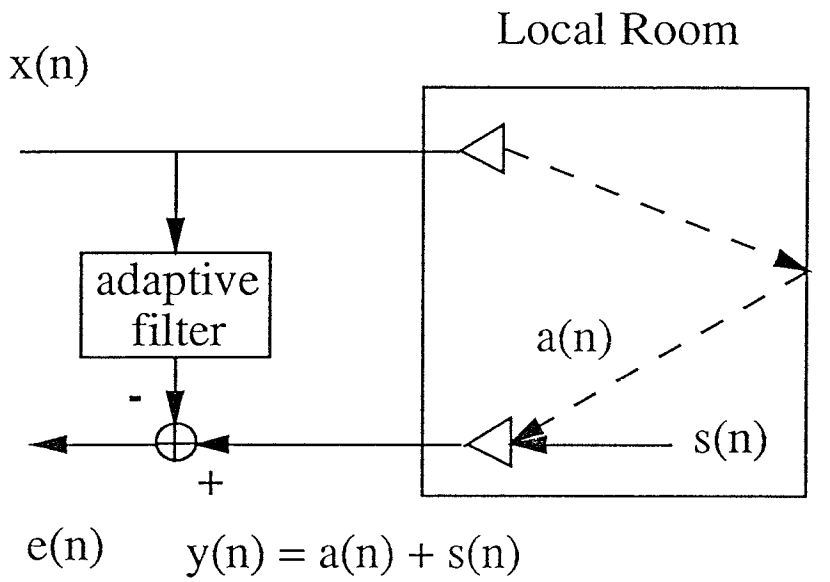


Fig. 4.2 A schematic diagram showing the teleconferencing application.

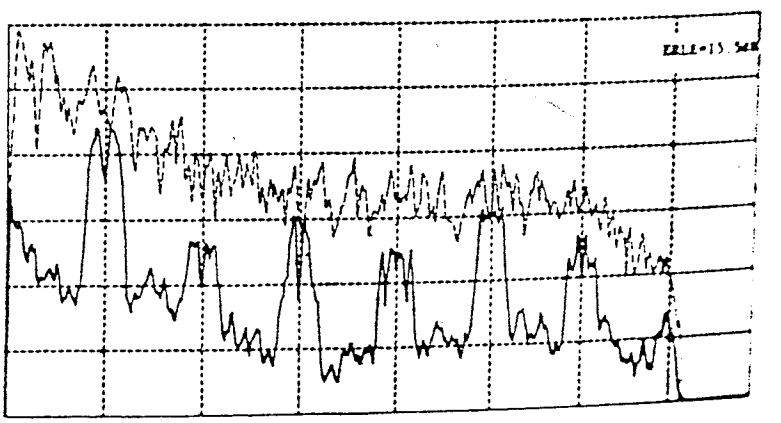


Fig. 4.3 Frequency characteristics of the echo $a(n)$. Scales: X 1KHz /Div, Y 10DB/Div (from [Gil88]).

figure, because these are not pertinent to the discussion below. An exact scheme can be found in [Gil88]. The audio signal $x(n)$ from the distant room is amplified through a loudspeaker into the local room. The speech signal $s(n)$ from the local room is collected through a microphone and transmitted to the distant room. The signal picked up by the microphone consists of local speech as well as the speech from the distant room amplified through the loudspeakers (the signal $a(n)$ in Fig. 4.2). The problem is to eliminate this unwanted feedback of speech signal to the distant room because it creates unnecessary degradation in the quality. An adaptive filter is used to eliminate the part of signal being transmitted that is generated by passing the distant speech signal through the room. Thus, in signal processing language, the adaptive filter is set up to match the acoustic response of the local room. It has been mentioned in literature [Gil88] that for the sampling frequency of 16 KHz, the room acoustic response can have several thousand coefficients. It has also been mentioned that this response is typically bandlimited. Fig. 4.3 shows the frequency characteristics of the room-acoustic response for the practical setup used in [Gil88]. It can be seen that the response is bandlimited to $7\pi/8$ frequency. Since the adaptive filter has to have a large number of coefficients, for a real time application of this echo cancellation scheme, it becomes important to achieve a reduction in the computational complexity.

4.1.2 Adaptive Equalization of Telephone Channels

Researchers have shown that telephone channels are typically bandlimited [Ung76]. Fig. 4.4 shows the frequency response of a typical telephone channel. It can be seen that for an 8 KHz sampling frequency, the channel is bandlimited to $3\pi/4$ frequency. Adaptive filtering is used to enhance performance of signal processing applications. Speech signal transmitted over the telephone channels is known to be bandlimited. The signal is oversampled for a variety of reasons, the

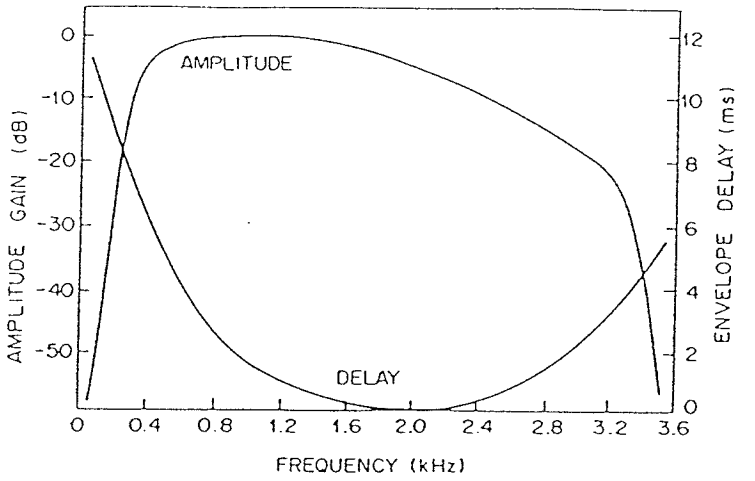


Fig. 4.4 Frequency response of a telephone channel (from [Lee85]).

most important one being to reduce noise sensitivity of the receiver [Ung76]. The various applications include clock-phase recovery for data modems [Ung76], interference cancellation or echo supression [Wid85, Chapter 12].

Researchers have realized the fact that computational savings can be achieved by using the information about bandlimitedness of the unknown system. Use of the multirate building blocks like decimators and interpolators has been proposed [Lee86] to reduce the speed and amount of computations performed. In a recently proposed method, the authors split the input signal in frequency bands of equal widths and perform adaptation only in the sub-bands with substantial energy [Gil88] in them. This method is shown in Fig. 4.5. In this example, the input $x(n)$ is split into four sub-bands. The output $d(n)$ of the unknown channel is also split into four sub-bands and adaptation is performed in each sub-band separately using the error in that sub-band. If the unknown system is bandlimited to $3\pi/4$, then

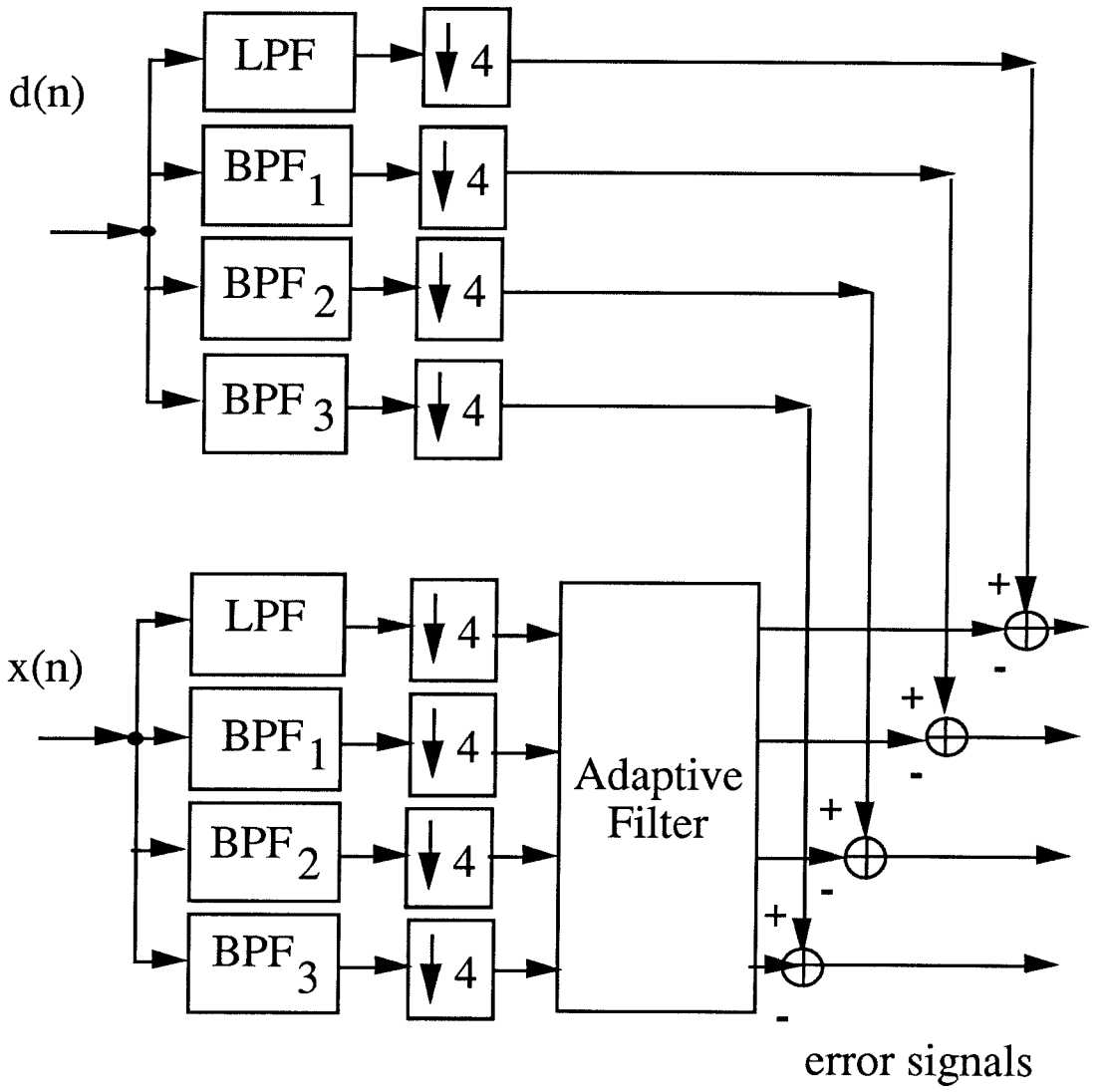


Fig. 4.5 The sub-band adaptation scheme.

the energy in the highest energy sub-band is not going to be significant. If we do not adapt in this sub-band, there will not be a significant change in the total error of the adaptive procedure, but this clearly results in savings in computations. In general, for a $\pi L/M$ signal, adaptation is performed in L of M sub-bands. Later in the chapter, we will compare the computational complexity of this method with the new method.

4.2 The New Method

Consider again the channel identification scheme of Fig. 4.1. Let us assume that the channel can be represented as a linear time invariant system with transfer function $C(z)$. The adaptive filter is represented by coefficients $a_{i,n}, (\ i = 0, \ldots, L_a - 1)$ where L_a is the length of the adaptive filter. The second subscript indicates the time instant of adaptation. In the adaptive identification procedure, the coefficients are updated to minimize an appropriate measure (mean square error) of the error $e(n)$. Qualitatively speaking, if the adaptation procedure converges to some steady state values a_i', then the frequency response of the corresponding filter $A'(e^{j\omega})$ resembles the channel frequency response. Suppose that the channel is bandlimited to frequency σ (i.e., σ-BL). A typical plot of magnitude of the channel frequency response for this case is shown in Fig. 4.6. For this case, the response $A'(e^{j\omega})$ will also be close to being σ-BL. We can use this information to modify the adaptive filter as follows: split the adaptive filter as a cascade of a fixed σ-BL filter $H(z)$ and an adaptive part (Fig. 4.7). One advantage of this is that the adaptive part would now typically have fewer coefficients as it is required to match the passband shape of $|C(e^{j\omega})|$ only.

We can further note that the input $u(n)$ to the adaptive filter is a σ-BL signal. This suggests that there is some redundancy in $u(n)$ due to oversampling. We can thus decimate the signal using a fractional decimator [Vai90] before feeding it to

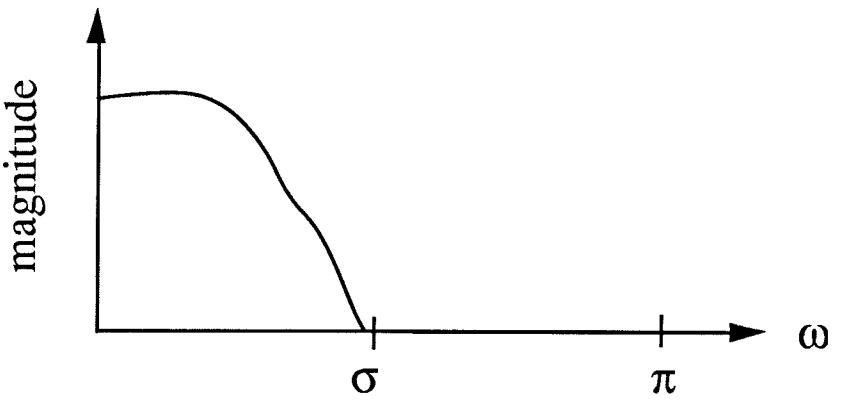


Fig. 4.6 An example of the frequency response of a bandlimited channel.

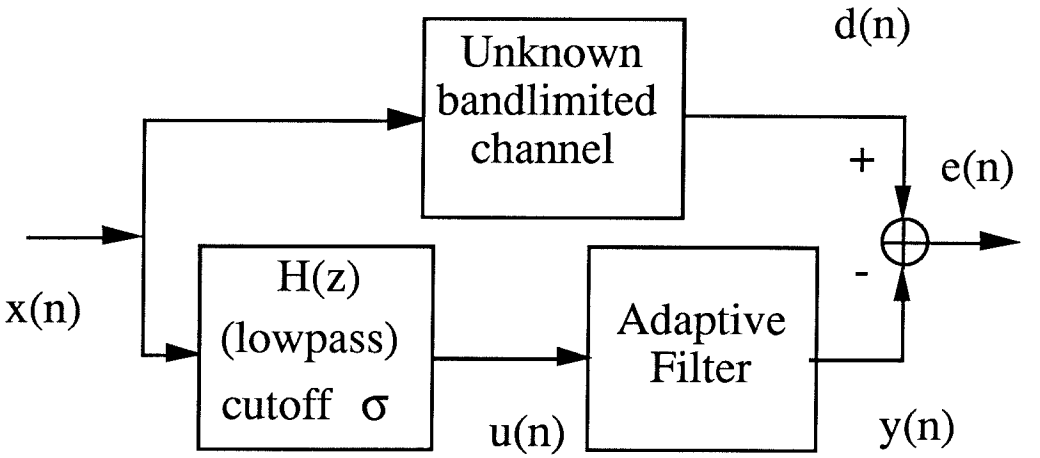


Fig. 4.7 Splitting the adaptive filter into a fixed and an adaptive part.

the adaptive filter. If L and M are two integers such that $\sigma \leq \pi L/M$ (L and M are assumed relatively prime without loss of generality), then we can decimate the signal by the ratio M/L. The modified adaptive filtering structure incorporating the fractional decimator is shown in Fig. 4.8.

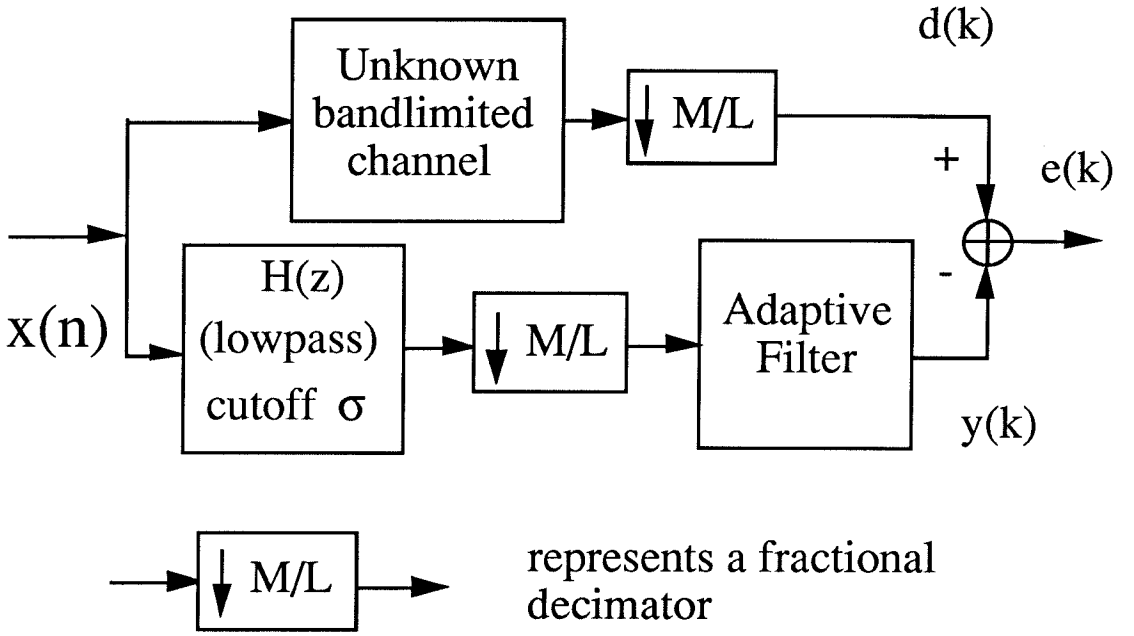


Fig. 4.8 Fractional decimation of bandlimited signals in the adaptation scheme.

Let us study the effect of fractionally decimating the signals. The fractional decimator is drawn in Fig. 4.9. The input $u(n)$ is a $\pi L/M$ bandlimited signal. The signal $t(n)$ has a frequency response which is an L-fold shrunk version of $U(e^{j\omega})$ repeating every $2\pi/L$. The lowpass filter $H'(z)$ is used to clean out the images created due to interpolation (Fig. 4.9(c)). Finally, the decimator stretches out the output frequency response to occupy the entire bandwidth.

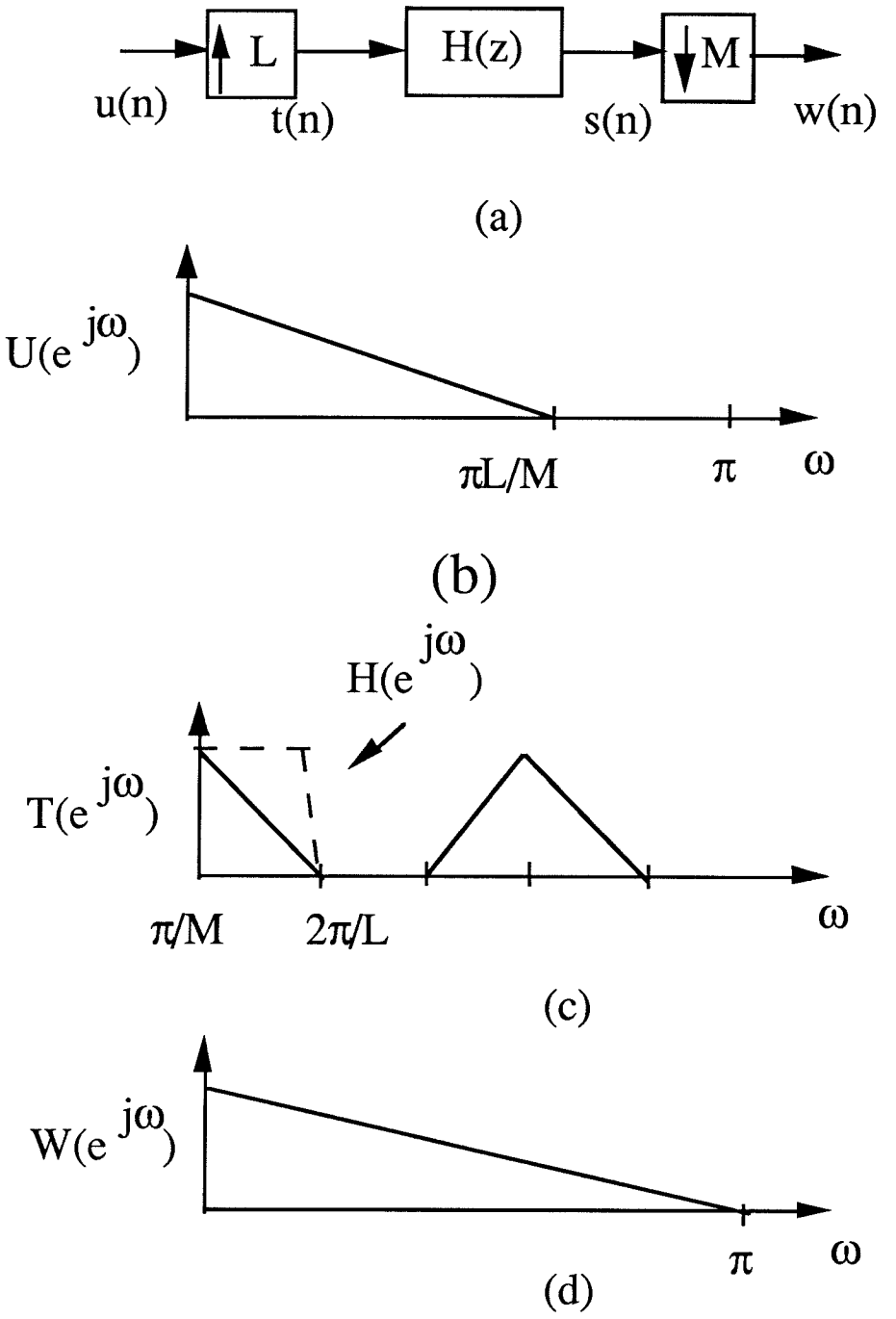


Fig. 4.9 Fractional decimation by the factor M/L

An immediate advantage of using the fractional decimation circuit is that the signal $w(n)$ input to the adaptive filter has a slower rate compared to the input $x(n)$. We can further reduce the complexity by noticing that the filters $H(z)$ and $H'(z)$ are performing the same filtering operation on the input signal $x(n)$. This is explained in Fig. 4.10. Using the multirate identity shown in Fig. 3.9, we can redraw Fig. 4.10(a) as Fig. 4.10(b). As we can see from Fig. 4.10(c), the frequency responses of $H(z^L)$ and $H'(z)$ are identical where the input frequency response is nonzero. Thus, it is enough to have only one filter $H'(z)$. We will denote this combined filter as $H_a(z)$. Qualitatively, we can say that the adaptive filter is now required to match the M/L-fold stretched response $C(e^{j\omega L/M})$ in the region $0 \leq \omega \leq \pi$. This in turn results in computational savings. This idea is explained as follows. Consider the "stretched passband" of the channel (shown in Fig. 4.11). From Fig. 4.11, we can see that this stretched version is smoother. This in turn means that, roughly speaking, the adaptive filter now can have a smaller length.

The final scheme is shown in Fig. 4.12. This method offers the following advantages:

(1) the adaptation takes place at a lower rate, and

(2) the adaptive filter has a smaller length because it needs only match passband of the channel frequency response.

Implementation Aspects

Two types of computations need to be performed for implementation of the new method:

1. implementation of the fixed lowpass filters $H_c(z)$ and $H_a(z)$,
2. implementation of the adaptive algorithm.

For given values of L and M we have to design filters $H_a(z)$ and $H_c(z)$. Both

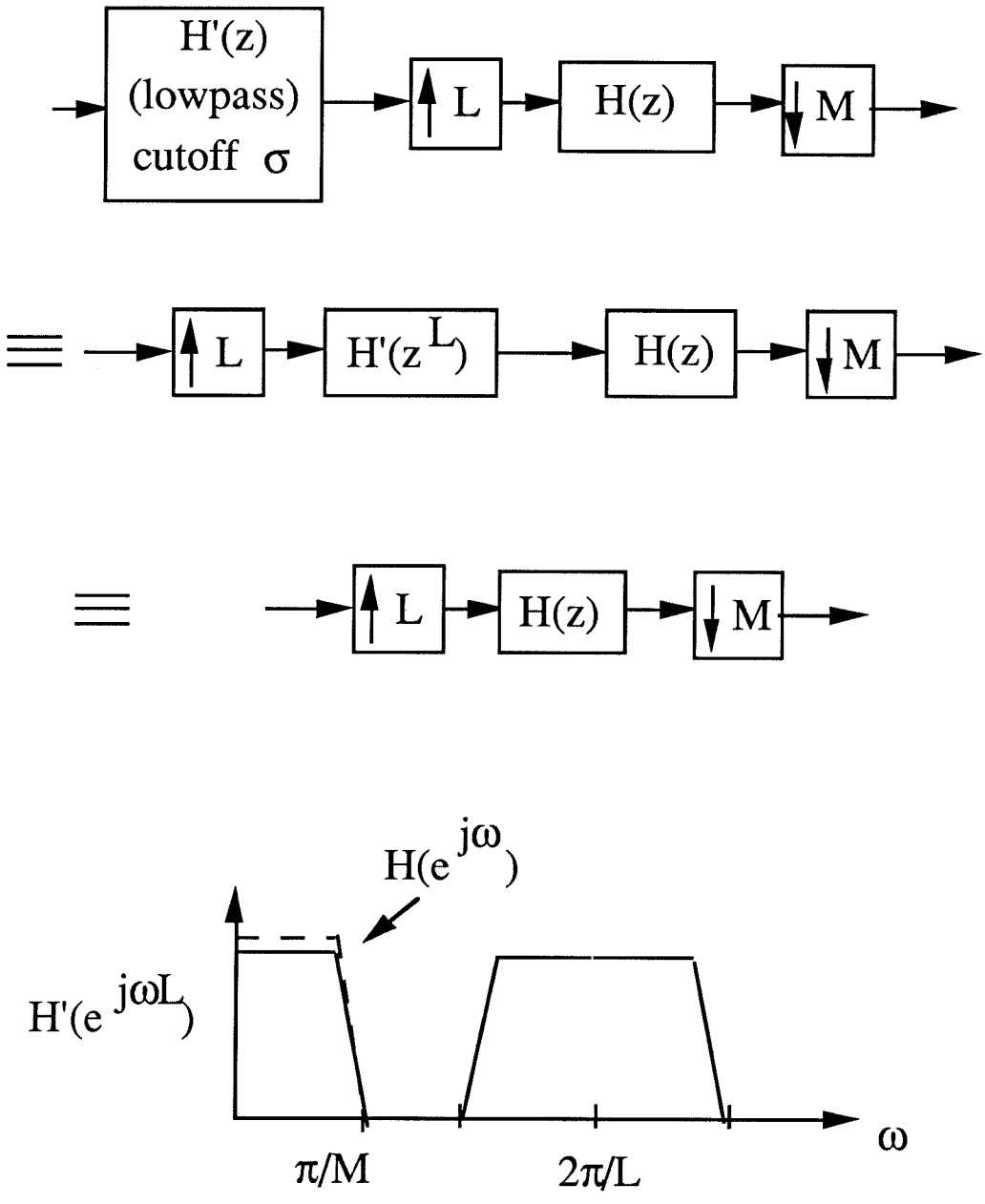


Fig. 4.10 Combining H(z) and H'(z).

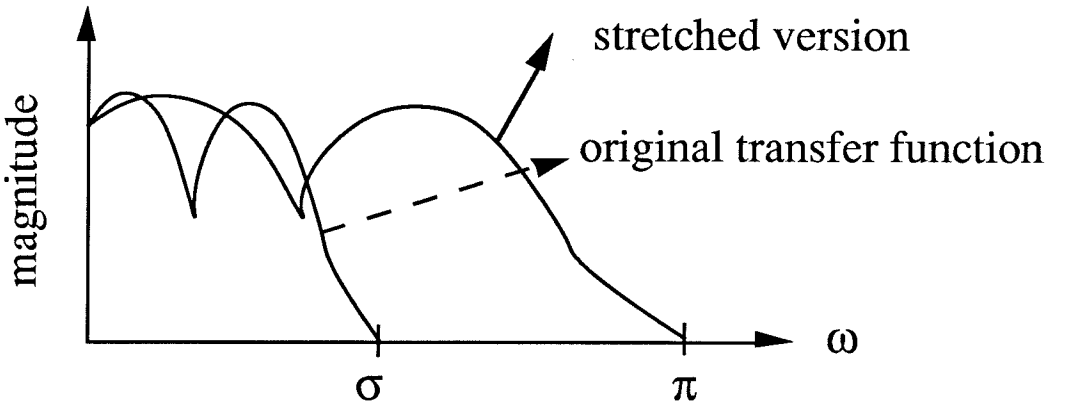


Fig. 4.11 Stretching the passband of a channel transfer function makes it smoother.

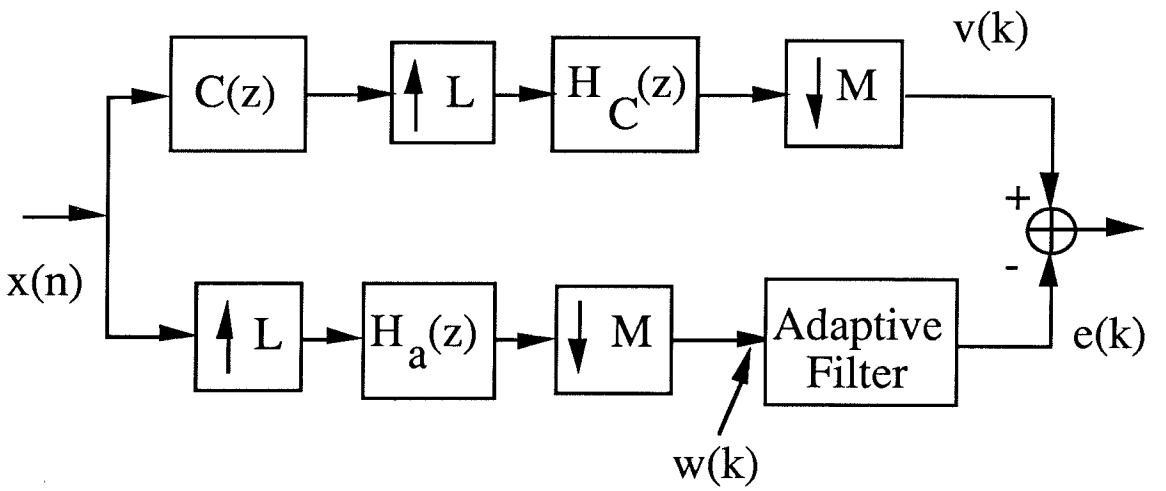


Fig. 4.12 Multirate adaptive filtering scheme for identification of bandlimited channels.

these filters are lowpass with π/M cutoff frequency. However, since the input to $H_c(z)$ is a $\pi L/M$ bandlimited signal, we can design $H_c(z)$ to have a wider transition band ($(\pi/L - \pi/M)$ wide). This reduces the filter length required. Similarly, the passband of $H_a(z)$ need not be very carefully designed. The adaptive filter that

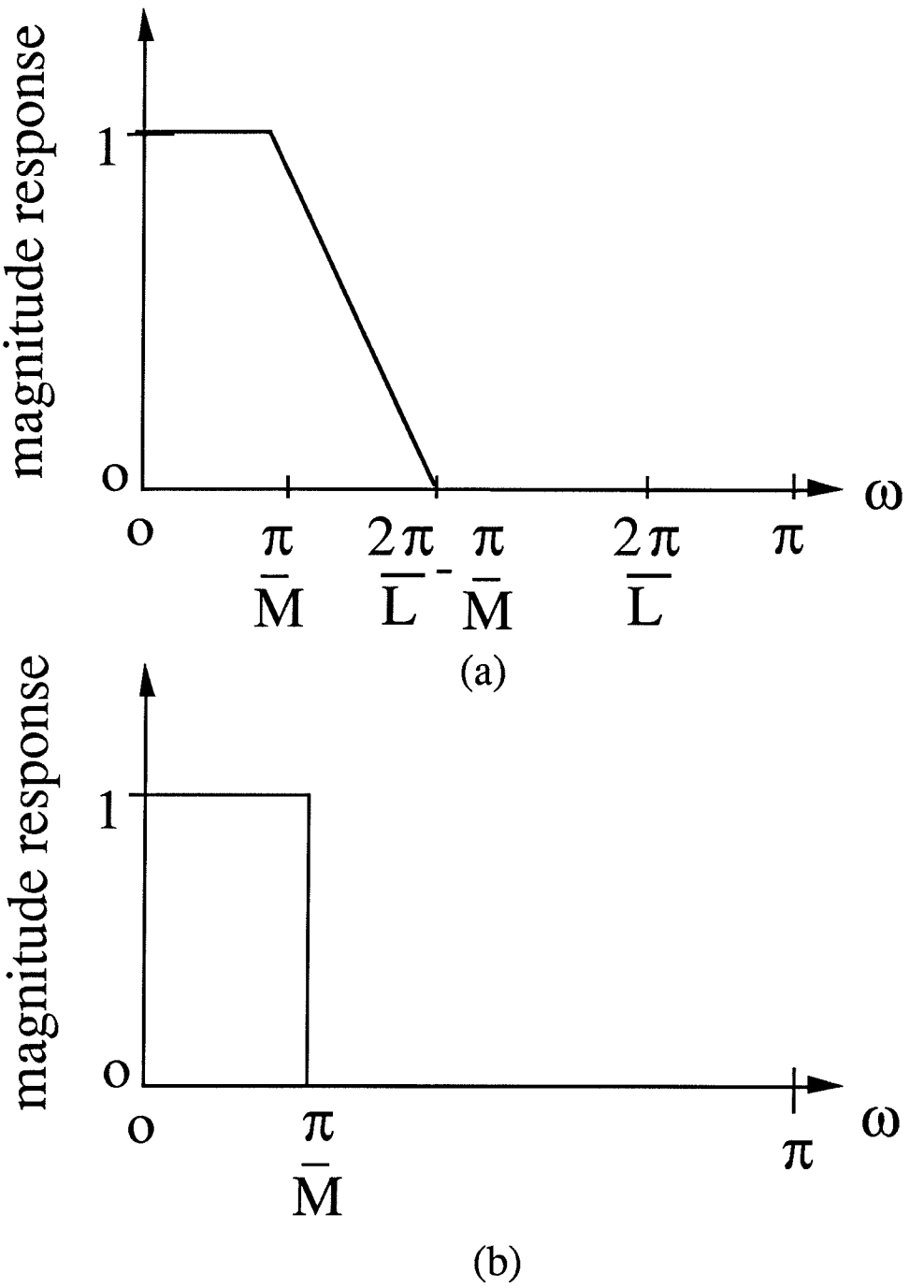


Fig. 4.13 Ideal magnitude response for (a) $H_c(z)$ and (b) $H_a(z)$.

follows $H_a(z)$ performs the function of matching the passband shape of the lower branch and upper branch in Fig. 4.12. This helps reduce the length of $H_a(z)$.

Fig. 4.13 shows the ideal frequency responses for the filters $H_a(z)$ and $H_c(z)$. We assume that these filters are FIR linear phase filters designed by the McClellan-Parks program [McC73]. We can implement these filters at a rate lower than the input data rate by using both Type I and Type II polyphase decompositions. Using this method proposed in [Hsi87], we can implement these filters at the slowest rate in the system. This method eliminates redundancies in filter implementation by using the fact that the input has nonzero elements every L^{th} sample only, and the output has to be calculated every M^{th} sample.

The adaptive filter can be implemented using an appropriate adaptive algorithm. In the subsequent discussion, we assume that we are using the LMS algorithm. The method offers both reduction in the length of the adaptive filter and reduced speed of computation.

Computational Complexity of the New Method

Suppose the unknown channel impulse response is estimated to have length L_c. For running the LMS algorithm on the adaptation scheme shown in Fig. 4.1, we have to perform $2L_c + 1$ multiplications and $2L_c + 1$ additions per input sample (abbreviated MPU and APU).

For the new multirate method, we have to calculate computational complexity for (i) the adaptation procedure, and (ii) implementation of the nonadaptive filters. Let L_f be the length of the interpolation filter $H_c(z)$. Using the efficient way of implementing multirate filters by decomposing in polyphase components mentioned in [Hsi87], we require L_f/M MPU and $(L_f - 1)/M$ APU. Similarly if L_h is the length of $H_a(z)$, we need L_h/M MPU and $(L_h - 1)/M$ APU for implementating $H_a(z)$. One way of choosing the length of the adaptive filter is by writing

$$L_h - 1 + M \times (L_a - 1) = L_f - 1 + L \times (L_c - 1). \tag{4.1}$$

Satisfying this equation ensures that the highest order of z^{-1} in the transfer func-

tions in the upper and lower branches of Fig. 4.12 are equal. However, in our experience, the adaptive filter can have a smaller length without degrading performance significantly. Using (4.1), and using the fact that the adaptive filter is implemented at $(L/M)th$ the rate of the input data rate, we get the computational complexity for the algorithm as

$$MPU = 2(\frac{L}{M})^2 L_c + \frac{2L}{M^2}(L_f - L_h) + \frac{2L}{M}(2 - \frac{L}{M}) \tag{4.2a}$$

$$APU = 2(\frac{L}{M})^2 L_c + \frac{2L}{M^2}(L_f - L_h) + \frac{2L}{M}(2 - \frac{L}{M}) - \frac{2}{M} \tag{4.2b}$$

Compared to the LMS method using transversal adaptive filter, this method thus offers savings in computations by a factor $(M/L)^2$ for sufficiently long length L_c of the channel impulse response. For long channel lengths, the dominant term in the complexity expression is

$$\frac{2L^2}{M^2} L_c. \tag{4.3}$$

For the case of transversal adaptive filter where no assumption about the bandlimitedness of the unknown system is made, the corresponding term is

$$2L_c. \tag{4.4}$$

The corresponding expression for the sub-band method is [Gil88]

$$\frac{10L}{3M^2} L_c. \tag{4.5}$$

Strictly speaking, the complexity expressions (4.3) and (4.5) can not be compared because both expressions are derived assuming ad hoc ways of deciding the length of the adaptive filter. We would like to emphasize here that our experience is that for a comparable computational complexity, the new method offers superior performance in terms of minimizing the mean square error at convergence.

4.3 Wiener Filter Solution

Even though the above scheme has its own advantages, it also brings with it some disadvantages. To explain this, let us analyze the scheme of Fig. 4.12 to get the optimal set of coefficients a_i' under the assumption that the input $x(n)$ is WSS. From Fact 3.11 we know that the signal $w(k)$ input to the adaptive filter is CWSS with period $K = L/gcd(L, M) = L$ (the signal would be WSS if and only if $H_a(z)$ is ideally bandlimited). Due to this, we can not use traditional Wiener filter theory to derive the optimal set of coefficients a_i'. The L-folded blocked version $\mathbf{w}(k)$ is a vector WSS process.

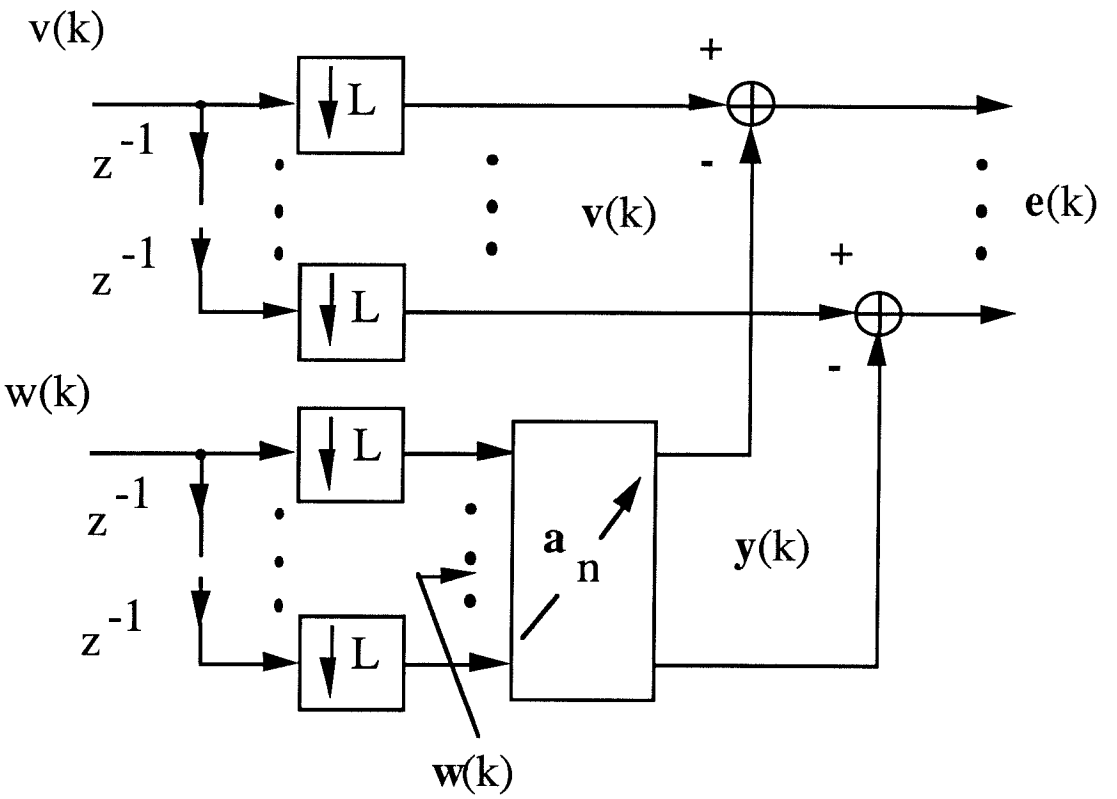


Fig. 4.14 Optimal filtering problem for the scheme in Fig. 4.12.

We can thus pose a matrix-Wiener filtering problem (Fig. 4.14) to solve for

the $L \times L$ optimal coefficient matrix $\mathbf{a}'_n$. The $L \times 1$ output $\mathbf{y}(k)$ of the block filter is compared with the blocked signal $\mathbf{v}(k)$ to obtain error $\mathbf{e}(k)$. The error $\mathbf{e}(k)$ will be WSS and we can obtain the Wiener solution $\mathbf{A}(z)$ which minimizes error energy $E[\mathbf{e}^T(k)\mathbf{e}(k)]$. We summarize the discussion above by proving the following result.

Fact 4.1: Consider the channel identification problem of Fig. 4.12 with L and M relatively prime. The Wiener (optimal) solution is in general an $(\text{LPTV})_L$ system and is scalar LTI if and only if both the filters $H_a(e^{j\omega})$ and $H_c(e^{j\omega})$ are image-free interpolation filters.

Remark: In particular, we are going to consider lowpass filters only. Hence we can say that the Wiener solution is scalar LTI if and only if the filters are ideally bandlimited to $\pi L/M$.

Proof of Fact 4.1: From Theorem 3.1, for nonideal lowpass filters $H_c(e^{j\omega})$ or $H_a(e^{j\omega})$, the input $w(k)$ and the desired signal $v(k)$ will be $(\text{CWSS})_L$. Both the signals are WSS if and only if the filters are ideal lowpass filters. Since the signals are $(\text{CWSS})_L$ in general, the blocked versions $\mathbf{v}(k)$ and $\mathbf{w}(k)$ are vector WSS. From (3.24) we know that the Wiener solution is given by

$$\mathbf{A}(e^{j\omega}) = \mathbf{S}^{\dagger}_{\mathbf{wv}}(e^{j\omega})\mathbf{S}^{-1}_{\mathbf{ww}}(e^{j\omega}). \tag{4.6}$$

The matrices $\mathbf{S}_{\mathbf{wv}}(e^{j\omega})$ and $\mathbf{S}^{-1}_{\mathbf{ww}}(e^{j\omega})$ in general are not pseudocirculant, so $\mathbf{A}(e^{j\omega})$ in general need not be pseudocirculant. From Fact 3.3, it can be seen that the optimal filter can not be thus written as an LTI system. If the optimal solution has a matrix form as in (4.6), then the output at time n is the output of one of L filters (each filter being a row of $\mathbf{A}(e^{j\omega})$) depending on the value (k modulo L). Thus, the optimal filter is an $(\text{LPTV})_L$ system.

However, in the case when both the lowpass filters are ideal, both the signals $w(k)$ and $v(k)$ are WSS. Thus, both the matrices in (4.6) are pseudocirculant. It is known that product of two pseudocirculant matrices is a pseudocirculant matrix

[Vai88]. Thus, the optimal filter will be an LTI system for this case. The transfer function of the optimal filter in this case would be $A(e^{j\omega}) = \sum_{k=0}^{L-1} e^{-j\omega k} A_k(e^{j\omega L})$ where $A_k(e^{j\omega})$ are entries of the 0th row of $\mathbf{A}(e^{j\omega})$.

For a real-time setup, the above result leads us to the following conclusions.

1. To be able to converge to the optimal solution, we should use an $L \times L$ adaptive filter $\mathbf{a}_n$. This would result in a better performance compared to a scalar adaptive filter. However, a block filter is inherently more complicated than a scalar filter and might offset the advantages offered by the multirate approach.
2. As the stopband attenuation of filters $H_a(z)$ and $H_c(z)$ increases, the performance of the scalar filter will approach that of the block filter. As a result, the use of a scalar adaptive filter would result in little loss in performance.

Clearly there is a tradeoff involved in designing $H_a(z)$ and $H_c(z)$. We shall not further discuss the optimal choice in this tradeoff (which appears to require careful study) but proceed to demonstrate the above ideas with an example. However, as mentioned before, the passband ripple specifications for $H_a(z)$ and the transition bandwidth for $H_c(z)$ are not very stringent.

A Special Case: $\sigma = \pi/M$

Now we consider the special case where the channel bandwidth is $\sigma = \pi/M$. For this case since $L = 1$, the filtering scheme does not have an interpolator. This scheme is shown in Fig. 4.15. This means that we now decimate the appropriate signals by the integer factor M. The adaptive filter has approximately M times fewer coefficients, and operates at M times lower rate. So we achieve computational savings by a factor M^2. At convergence, the system is equivalent to Fig. 4.16. The cascade transfer function $H_a(z)A(z^M)$ now approximates the channel $C(z)$ (neglecting the effect of $H_c(z)$). This scheme can therefore be considered to be an extension, to the adaptive regime, of the Interpolated FIR (IFIR) approach for

efficient design and implementation of narrowband FIR filters [Neu84]. Once again, the main purpose of $H_a(z)$ is to provide satisfactory out-of-band attenuation. Its passband ripple size is not crucial.

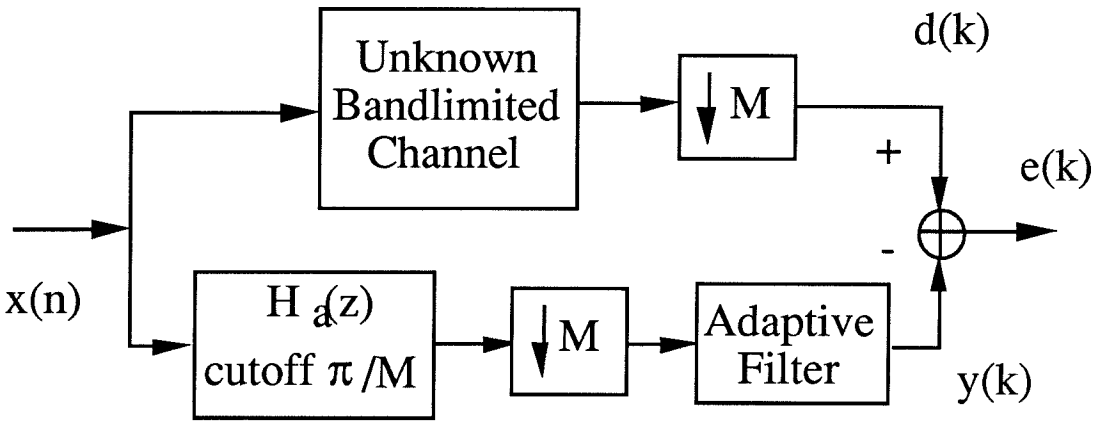


Fig. 4.15 Adaptive filtering scheme for a π/M bandlimited channel.

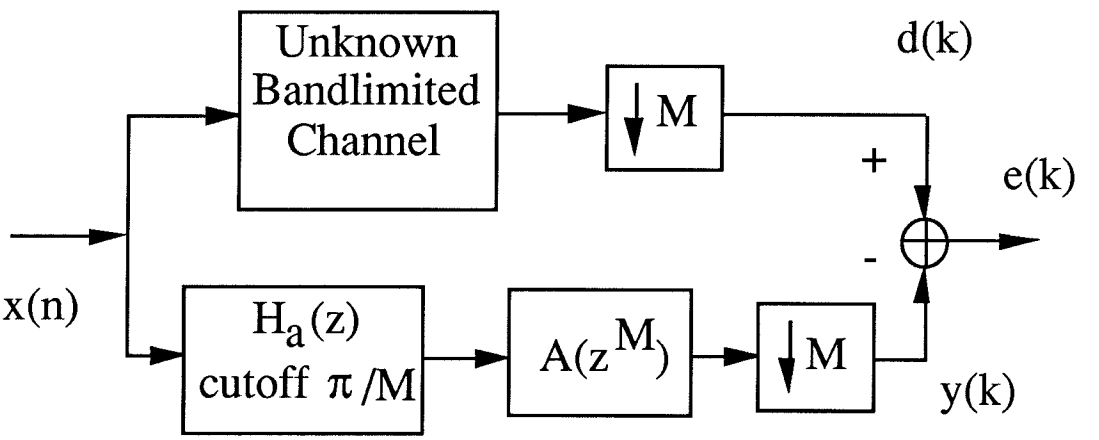


Fig. 4.16 Redrawing the adaptation scheme of Fig. 4.15 after convergence.

4.4 Simulations

Consider a simulation example where the channel $C(z)$ was simulated using an FIR filter of length 77 (not linear phase). The channel was designed to be bandlimited to frequency $3\pi/4$. The channel frequency response is shown in Fig. 4.16. We thus chose $L = 3$, $M = 4$ for the simulation. The filters $H_a(z)$ and $H_c(z)$ were designed to be linear phase FIR filters of length 47 and 31 respectively, designed by the McClellan-Parks program [McC73]. The adaptive filter was chosen to have length 30, since this length gave satisfactory performance (although this length does not satisfy (4.1)). The filter was adapted for 2000 samples of the input signal. The magnitude of the channel frequency response is compared with the magnitude of the adaptive filter (after convergence) in Fig. 4.17. A plot of the difference in the phases is also shown in Fig. 4.18. From these results, it can be seen that the new method gives excellent performance. The learning curve for this adaptation procedure is shown in Fig. 4.19.

Since the optimal solution is a matrix filter, it is of interest to compare the performance of a scalar filter with a corresponding matrix adaptive filter. Two different cases were studied: (1) scalar adaptive filter, and (2) matrix adaptive filter. For the scalar adaptive filtering case also, the adaptive filter was chosen to have length 30. For the matrix adaptive filter, the adaptive filter was a 3×3 matrix with each entry being a length 10 filter. The reason to choose this length is that if the lowpass filters were ideal, blocking a length 30 filter (chosen for the scalar adaptive case) would have resulted in block matrix with entries 10 coefficients long each. The blocked signals $\mathbf{w}(n)$ and $\mathbf{v}(n)$ were used in the adaptive updating. The adaptation procedure was performed for various values of the step size μ. The results presented here are for $\mu = 0.1$ for the scalar and $\mu = 0.3$ for the matrix adaptive filter, because these values gave minimum error energy at convergence

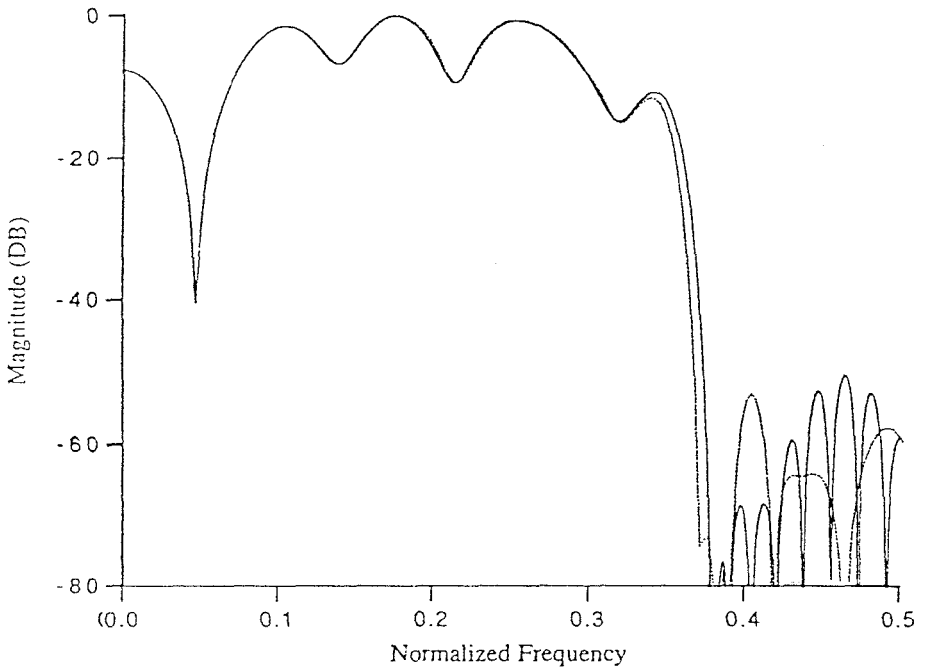


Fig. 4.17 A comparison of the channel frequency response with the frequency response of the adaptive filter at convergence. (..... adaptive filter, ____ channel transfer function).

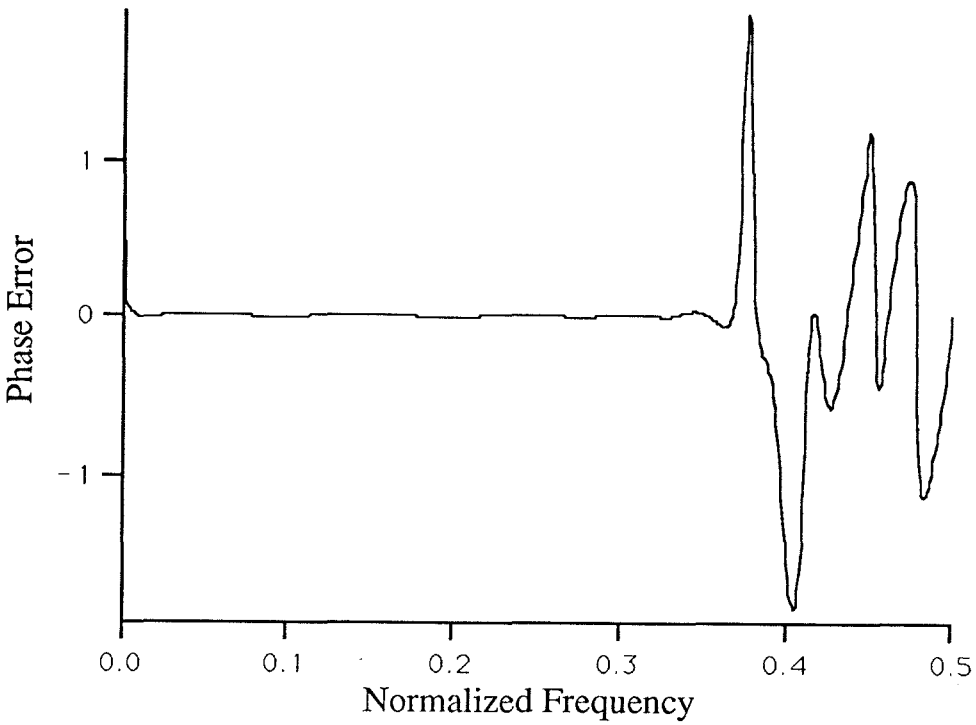


Fig. 4.18. Error in phase response of the adaptive filter after convergence and the channel transfer function.

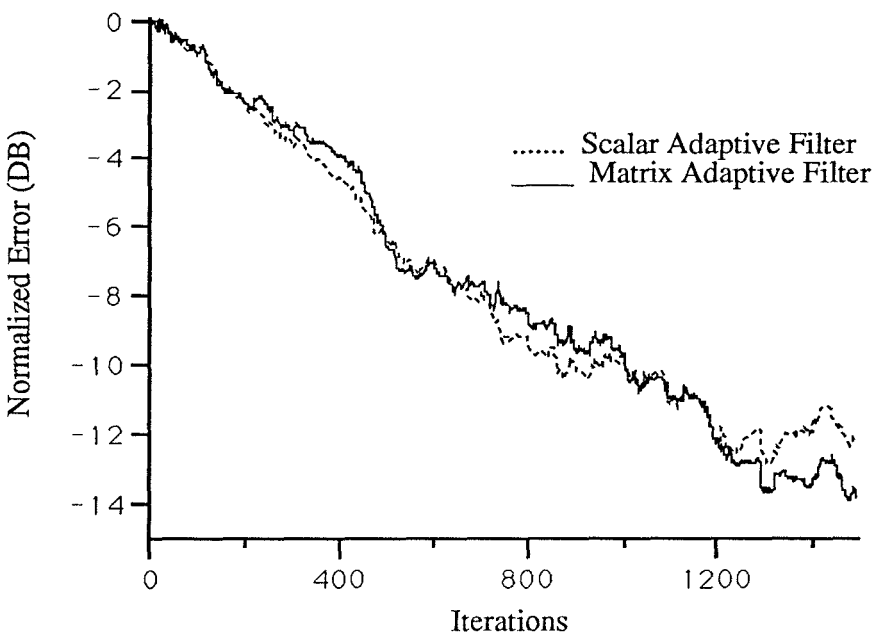


Fig. 4.19. Error energy v/s iteration curves for scalar and matrix adaptive filtering cases.

(over a wide range of values for μ). The error energy at convergence of these two cases was compared for different attenuations of the lowpass filter $H_a(z)$. The filter $H_c(z)$ had 51 DB stopband attenuation and it was not changed throughout the simulations. Table 4.1 gives the results of the simulation. A typical plot of how error energy reduces with iterations for both the methods is shown in Fig. 4.19.

Stopband attenuation of Ha(z) (DB)	Error energy at convergence (db)	
	Scalar	Matrix
41	-15.4	-14.1
29	-13.1	-13.4
21	-9.5	-11.4
18	-7.4	-9.0

Table 4.1 A comparison of the performance of scalar and matrix adaptive filter.

The results show that the matrix adaptive filter performs better than the scalar adaptive filter in terms of minimizing the error energy at convergence. This agrees with Fact 4.1 because the scalar adaptive filter converges to an LTI system which is suboptimal if $H_a(z)$ is not ideally bandlimited. On the other hand, the matrix filter converges to an $(\text{LPTV})_L$ filter. As the stopband attenuation of $H_a(z)$ increases the signal $w(n)$ gets closer and closer to being WSS. So the relative degradation in the performance of the scalar adaptive filter reduces, as seen from Table 4.1. This shows that designing $H_a(z)$ to have "good" attenuation reduces the degradation in the performance.

4.5 Summary

In this chapter, we have presented a computationally efficient method for adaptive identification of bandlimited channels. We have seen some applications where

bandlimited unknown systems occur. We have shown that the bandlimitedness of the unknown system can be used to reduce the length and the speed of operation of the adaptive filter. Using the theory developed in the previous chapter, we have shown that the optimal solution for the filtering scheme developed here is an LPTV system. Simulations however show that a transversal adaptive filter has comparable performance if the lowpass filters in the optimal filtering scheme have good stopband attenuation. Simulations are presented showing that the new adaptive method gives excellent performance compared to the existing methods. For a special case of the cutoff frequency $\sigma = \pi/M$, we have pointed out the similarity between this method and IFIR filter designing technique.

Chapter 5

Floating Point Error Variance Analysis of Signal Processing Algorithms

5.1 Introduction

In practice, when signal processing algorithms are implemented on computers or special purpose hardware, they are associated with *roundoff errors* due to the fact that the computations are performed using a finite bit representation of multiplier values and signals involved in the algorithm. In a finite bit representation, all the quantities involved in the computation can take on only finitely many values belonging to a set of real numbers. The set of these permissible values depends on the number system used for the computations. Thus, we have to represent any real number by an appropriate permissible number. This is called "quantizing" a real number. If x is a given number which is represented by a permissible number x_p, then the quantity

$$e = x - x_p \tag{5.1}$$

is called the **quantization error** associated with the representation. In the implementation of an algorithm, since all the quantities involved are quantized in this way, any intermediate quantities computed will tend to be different from the exact value (that which would be the result of an exact or an infinite bit precision computation). Although the quantization error in (5.1) associated with each number is

very small, depending on the sequence of computations, the final answer can differ from the exact answer considerably. The study of such errors is referred to as the roundoff error analysis. Most signal processing algorithms consist of repeated use of some basic computational steps. It is therefore important to first perform the roundoff error analysis of these basic steps. The results obtained can then be used in the error analysis of more complex algorithms. As an example consider the LMS algorithm mentioned in Section 1.1. The computations performed are:

$$y(n) = \mathbf{a}^T(n)\mathbf{x}$$

$$e(n) = d(n) - y(n)$$

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \mu e(n)\mathbf{x}(n).$$

It can be seen that all these computational steps can be written as a dot product of appropriate vectors. Hence to be able to analyze roundoff errors occuring in the LMS algorithm, understanding of errors incurred in dot product computation is important.

The two most important issues involved in the roundoff error analysis context are:

(1) stability of a given algorithm, and

(2) the correctness of the final solution obtained.

Most of the signal processing algorithms are in some form an iterative technique to arrive at a certain result. In real time applications, these iterations have to be performed as the new data comes in. During the iterative computations, it is possible for the quantization errors to accumulate in such a fashion that the algorithm fails to converge. In the worst case, the algorithm might become unstable. It has been shown that certain algorithms such as the RLS algorithm used in adaptive filtering can become unstable under quantization [Lin84], [Lju85]. This makes the study of the stability of a given implementation (fixed point or floating point) and a corresponding quantization scheme very important. In a not so disastrous effect

of quantization, an algorithm might converge to a solution with some error in it. To get an idea about the correctness of the final solution is also very important from the application viewpoint.

In this chapter, we study a third important aspect of roundoff error analysis:

(3) the typical behavior of an algorithm under quantization.

For the first two types of roundoff error analyses mentioned above, the aim is to derive an upper bound on some appropriate norm of the error incurred at a certain stage in the computation. To do this, one assumes the worst case accumulation of errors. The error is assumed as being the "output" of the algorithm. This idea is shown in Fig. 5.1.

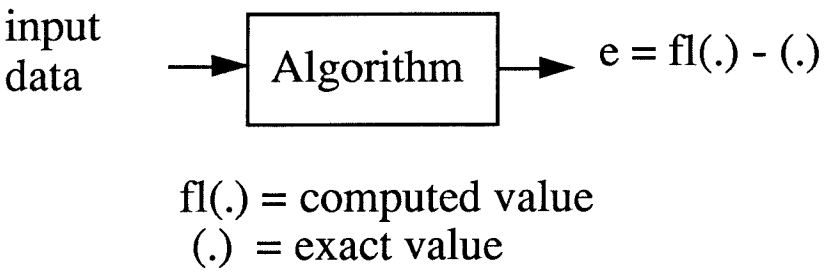


Fig. 5.1 Roundoff error as the output of an algorithm.

This bound gives an idea about the stability and correctness of the computation. This analysis however does not answer probabilistic questions of the type "What would a typical value of the error be?" More specifically, we would like to know the mean and variance of the error for some appropriate choice of the statistics of the input signals. Using these quantities we can write the signal to computational noise ratio of the algorithm defined as

$$\text{SNR} = \frac{\text{Output signal energy}}{\text{Error variance}}. \tag{5.2}$$

This quantity gives us additional insight regarding the behavior of an algorithm under finite bit precision. The SNR figure for an algorithm tells us about the energy

level of the computational noise added to the output signal. We can thus compare error performance of two implementations on the basis of their SNR figures.

In this chapter, we present results for the error variance analysis of some basic signal processing algorithms. As mentioned before, computation of dot product is a key step in signal processing algorithms. We therefore perform the error variance analysis for dot product computation. The SNR or the error variance values of different algorithms can be used to compare their relative performances, or "robustness" to quantization. We highlight this aspect by performing this type of comparison for Givens Rotations (GR) and Householder Transformations (HT) used for QR decomposition of matrices.

In the discussion that follows, we perform error variance analysis for floating point implementation only. In the recent years, floating point hardware for real time signal processing applications has become a reality. Floating point implementation offers some advantages such as increased dynamic range. Due to the increasing popularity of floating point implementation, it is important to perform error variance analysis for this case.

Computation of dot product of a vector $\mathbf{x}$ with another vector $\mathbf{y}$ or with itself needs to be performed in a number of signal processing algorithms. Many researchers consider the dot product computation as a basic arithmetic operation along with the four fundamental arithmetic operations: addition, subtraction, multiplication and division [Kul86]. Well-known adaptive filtering algorithms like the LMS algorithm [Chu87] or the fast RLS algorithm [Cio84] repetitively use these dot product computations. Numerical properties of these algorithms are of considerable interest and have been extensively studied recently [Lju85]. Most of the roundoff error analyses for these algorithms derive expressions for upper bounds for the worst case accumulation of errors. In Section 5.2, we derive expressions for variance of errors for both the cases (i) dot product of two vectors $\mathbf{x}$ and $\mathbf{y}$,

and (ii) dot product of a vector $\mathbf{x}$ with itself. We also obtain expressions for the ratio of signal power (mean square of the dot product value) to roundoff error variance (SNR) for both the cases. We compare these results with the corresponding results from fixed point error analysis. Finally, we present simulation results demonstrating that the theoretical error variance expressions closely agree with actual simulations on a floating point mainframe computer.

A similar analysis is performed for Givens Rotation (GR) and Householder Transforms (HT). Both these techniques are used in the triangularization of matrices (as explained in section 5.3). One key use of this triangularization is finding out the QR decomposition [Boj86] of a matrix. These techniques are also used to find out Cholesky factor of a matrix [Gol89]. The QR decomposition has a variety of signal processing applications. For example, it is used to solve least squares problems [Gol89]. The GR algorithm is more suited to parallel implementation. However as we shall see later, the HT algorithm has a better SNR, which means that *on an average* it adds less computational error to the final result. In sections 5.4 and 5.5, we explain the GR and HT techniques respectively, and their application to upper triangularization of matrices. Error variance analyses of these algorithms are presented in the corresponding sections. For these analyses we use the error variance results obtained for dot product computations. An analysis of how the errors increase as we succesively use GR and HT for triangularization is presented in the same sections respectively. A comparison of the two methods based on their SNR values is performed in section 5.6. Some simulation results are also presented which show a close agreement between the theory and actual roundoff errors obtained on a floating point mainframe computer. We will begin the discussion by first describing the floating point number system convention used in this chapter.

The floating point number system

In this chapter, we follow the notations used in [Gol89, Chapter 2]. Thus, any number belonging to the floating point number system can be represented as

$$f = \pm .d_1 d_2 \ldots d_t \times \beta^e, \quad 0 \leq d_i < \beta, \quad d_1 \neq 0, \quad L \leq e \leq U. \tag{5.3}$$

We shall denote the set of floating point numbers by $\mathbf{F}$. Thus there are four integers (β, t, L, U) associated with the description of a floating point number system. The floating point representation above is said to have a t-bit mantissa.

The quantization convention we use for the error analysis presented here is called "rounding arithmetic." In this arithmetic, a real number x is represented by a corresponding floating point number $fl(x) \in \mathbf{F}$ such that $fl(x)$ is nearest to x. In case of a tie, we round away from zero.

The floating point number system considered here is a binary system ($\beta = 2$), with a t-bit mantissa. The $fl(.)$ operator can be shown to satisfy [Gol89]

$$fl(x) = x(1 + \epsilon), \qquad |\epsilon| \leq \mathbf{u}, \tag{5.4}$$

where the constant $\mathbf{u}$ is called the machine precision and is given by

$$\mathbf{u} = 2^{-t} \tag{5.5}$$

for the rounding arithmetic. It follows that for two numbers a and b, and an arithmetic operation "op",

$$fl(a \text{ op } b) = (a \text{ op } b)(1 + \epsilon). \tag{5.6}$$

It is easy to see that fixed point representation is a special case of floating point representation, where the exponent e has a fixed value ($= 0$). Thus we need to perform scaling of signals to confine the values to the representation (5.3) above [Opp75].

5.2 Dot Product Computation

The accumulation of roundoff errors incurred depends on the order in which computations are performed to arrive at a result. Hence, before we analyze an algorithm to obtain an expression for its SNR, we have to study the exact steps in which the results are computed. For analyzing dot product computations, we first describe the algorithm used to obtain the dot product.

5.2.1 Computation of Dot Product

Let $\mathbf{x} = [x_1\ x_2 \ldots x_n]^T$ and $\mathbf{y} = [y_1\ y_2 \ldots y_n]^T$ be two $n \times 1$ real vectors. The dot product can be represented as $\mathbf{x}^T\mathbf{y}$ where the superscript T denotes transpose of a vector. We assume that the dot product is evaluated by computing the partial sum

$$s_p = fl(\sum_{k=1}^{p} x_k y_k) \tag{5.7}$$

and updating it as

$$s_{p+1} = fl(s_p + fl(x_{p+1}y_{p+1})) \tag{5.8}$$

for $p = 1, \ldots, n-1$. Clearly $s_n = fl(\mathbf{x}^T\mathbf{y})$. Thus, in terms of roundoff errors, we can write (5.8) as

$$s_{p+1} = (s_p + x_{p+1}y_{p+1}(1+\delta_{p+1}))(1+\epsilon_{p+1}) \tag{5.9}$$

where δ_{p+1} and ϵ_{p+1} are roundoff errors associated with multiplication and addition, satisfying $|\delta_{p+1}|, |\epsilon_{p+1}| \leq \mathbf{u}$ for $0 \leq p \leq n-1$.

5.2.2 Derivation of Error Variance and SNR

We first consider the case of computing dot product of two vectors $\mathbf{x}$ and $\mathbf{y}$. We can write the floating point roundoff error as

$$e_1 = fl(\mathbf{x}^T\mathbf{y}) - \mathbf{x}^T\mathbf{y}. \tag{5.10}$$

We wish to derive an expression for $E[e_1^2]$. After some algebra, (5.9) gives

$$s_n = fl(\mathbf{x}^T\mathbf{y}) = \sum_{k=1}^{n} x_k y_k (1+\gamma_k). \tag{5.11}$$

The variable γ_k is defined by

$$1+\gamma_k = (1+\delta_k)\prod_{j=k}^{n}(1+\epsilon_j). \tag{5.12}$$

For the derivation of $E[e_1^2]$, we make the following assumptions:

1. The errors δ_p, ϵ_p are mutually independent random variables, independent of all the entries x_j and y_j of $\mathbf{x}$ and $\mathbf{y}$.
2. The multiplication error δ_i is uniformly distributed in the interval [-**u**,**u**] for all i. For any i, the error ϵ_i associated with addition takes three values -**u**/2, 0, **u**/2 with equal probability (see below for the explanation).
3. All the entries of the vectors $\mathbf{x}$ and $\mathbf{y}$ are zero mean random variables, with $E[\mathbf{x}\mathbf{x}^T] = \mathbf{C}_{xx}$ and $E[\mathbf{y}\mathbf{y}^T] = \mathbf{C}_{yy}$.
4. All the entries x_i are independent of all the entries y_j, so that $E[\mathbf{x}\mathbf{y}^T] = \mathbf{0}$.

To explain the particular probability distribution chosen for the roundoff errors ϵ_i's in assumption (2), we have to look at the origin of these errors. From (5.9), we see that the error ϵ_i is the roundoff error that is incurred when the partial sum s_{i-1} is added to the product $x_i y_i$. These quantities have about the same magnitude. This restricts the possible values of the roundoff error because we assume that the exponents of the two numbers being added are equal.

Using these assumptions, we get $E[e_1] = 0$. Substituting (5.11) in (5.10), squaring and taking expectation of both the sides gives

$$E[e_1^2] = \sum_{i=1}^{n}\sum_{j=1}^{n} E[x_i x_j]E[y_i y_j]E[\gamma_i \gamma_j]. \tag{5.13}$$

We first evaluate the term $E[\gamma_i\gamma_j]$ for $i \geq j$. For this, we multiply the expressions for $(1+\gamma_i)$ and $(1+\gamma_j)$ as given by (5.12) and take expected value of both the

sides. Using the facts that $E[\gamma_i] = E[\gamma_j] = 0$ and all the errors are zero mean, mutually independent random variables, we obtain the following equality

$$E[1 + \gamma_i\gamma_j] = E[1 + \delta_i\delta_j]E[\prod_{m=i}^{n}(1 + \epsilon_m^2)]. \tag{5.14}$$

From the probability distribution assumptions mentioned above, we have

$$E[\epsilon_i^2] = \mathbf{u}^2/6, E[\delta_i^2] = \mathbf{u}^2/3. \tag{5.15}$$

Using these expressions in (5.12) above gives

$$E[\gamma_i\gamma_j] = \frac{(n + 1 - i)\mathbf{u}^2}{6} \quad i > j. \tag{5.16}$$

Similarly we obtain

$$E[\gamma_i^2] = \frac{(n + 3 - i)\mathbf{u}^2}{6}. \tag{5.17}$$

In the above derivations, we have used the assumption that the errors $\epsilon_i's$ and $\delta_i's$ are mutually independent. We can write down (5.13) as a summation of term-by-term multiplication of the entries of three symmetric matrices $\mathbf{C}_{xx}$, $\mathbf{C}_{yy}$ and $\mathbf{\Gamma}$,

$$E[e_1^2] = \sum_{i=1}^{n}\sum_{j=1}^{n}[\mathbf{C}_{xx}]_{i,j}[\mathbf{C}_{yy}]_{i,j}[\mathbf{\Gamma}]_{i,j}. \tag{5.18}$$

The element in the ith row and the jth column ($i > j$) of $\mathbf{\Gamma}$ is given by (5.16) and the ith diagonal element is given by (5.17). The elements above the diagonal can be obtained using the fact that $\mathbf{\Gamma}$ is a symmetric matrix.

In particular, if we assume that $\mathbf{C}_{xx} = \sigma_x^2\mathbf{I}$, and $\mathbf{C}_{yy} = \sigma_y^2\mathbf{I}$, then the expression simplifies to

$$E[e_1^2] = \sigma_x^2\sigma_y^2\frac{\mathbf{u}^2}{6}\frac{(n^2 + 5n)}{2}. \tag{5.19}$$

Expression for SNR: For the assumptions mentioned above, we get $E[(\mathbf{x}^T\mathbf{y})^2] = n\sigma_x^2\sigma_y^2$. Thus

$$\begin{aligned} SNR_1 &= \frac{E[(\mathbf{x}^T\mathbf{y})^2]}{E[e_1^2]} \\ &= \frac{12}{\mathbf{u}^2(n + 5)}. \end{aligned} \tag{5.20}$$

Evaluation of the dot product $\mathbf{x}^T\mathbf{x}$.

Next, we consider the case of finding the dot product $\mathbf{x}^T\mathbf{x}$. The dot product is computed by evaluating the partial sums as mentioned before. For this case, let us write down the roundoff error as

$$e_2 = fl(\mathbf{x}^T\mathbf{x}) - \mathbf{x}^T\mathbf{x}. \tag{5.21}$$

This can be considered as a special case of the dot product $\mathbf{x}^T\mathbf{y}$ with $\mathbf{x} = \mathbf{y}$. Thus, the assumption (4) mentioned before clearly no longer holds. For this case, the errors ϵ_i are assumed to have uniform probability distribution in the interval [-**u**,**u**]. This assumption differs from assumption (2) of the previous case. Unlike the previous case, in the computation of the dot product $\mathbf{x}^T\mathbf{x}$, $E[s_{i-1}] = (i-1)\sigma_x^2$ and $E[x_i^2] = \sigma_x^2$, so that typically two quantities of different orders of magnitude are being added. Since no further assumptions can be made about exponents, we shall have to assume a uniform probability of distribution for the errors.

It is easy to see that $E[e_2] = 0$. Using the modified assumptions about statistics of various quantities, we can write an error variance expression similar to (5.13) as

$$E[e_2^2] = \sum_{i=1}^{n}\sum_{j=1}^{n} E[x_i^2 x_j^2]E[\gamma_i\gamma_j], \tag{5.22}$$

where γ_j, $1 \leq j \leq n$ are given by (5.22). Using the new assumptions for the errors ϵ_j and δ_j, we get

$$E[\gamma_i\gamma_j] = \begin{cases} (n+1-i)\mathbf{u}^2/3, & i > j \\ (n+2-i)\mathbf{u}^2/3, & j = i. \end{cases} \tag{5.23}$$

To further simplify (5.22), we assume that x_i $(i = 1, \ldots, n)$ are Gaussian random variables with variance σ_x^2. We use the result that for zero mean Gaussian random variables, fourth order expectations can be written down in terms of second order expectations as [Urk83]

$$E[z_1 z_2 z_3 z_4] = E[z_1 z_2] + E[z_2 z_3] + E[z_3 z_4] + E[z_1 z_4]. \tag{5.24}$$

Using (5.24), we can write

$$E[x_i^2 x_j^2] = \begin{cases} 3\sigma_x^4, & i = j \\ \sigma_x^4 + 2(E[x_i x_j])^2, & i \neq j. \end{cases} \tag{5.25}$$

If we assume that the entries of $\mathbf{x}$ are mutually independent, then $E[x_i x_j] = 0$ if $i \neq j$, and using (5.25), the error variance expression (5.22) can be simplified as

$$E[e_2^2] = \frac{\sigma_x^4 \mathbf{u}^2}{3}\left(\frac{n^3}{6} + \frac{7n^2}{4} + \frac{49n}{12} + \frac{1}{2}\right). \tag{5.26}$$

Expression for SNR

The output signal power is

$$\begin{aligned} E[(\mathbf{x}^T\mathbf{x})^2] &= \sum_{i=1}^{n}\sum_{j=1}^{n} E[x_i^2 x_j^2] \\ &= \sigma_x^4(n^2 + 2n). \end{aligned} \tag{5.27}$$

Hence

$$\begin{aligned} SNR_2 &= \frac{E[(\mathbf{x}^T\mathbf{x})^2]}{E[e_2^2]} \\ &= \frac{3}{\mathbf{u}^2}\frac{n^2 + 2n}{\left(\frac{n^3}{6} + \frac{7n^2}{4} + \frac{49n}{12} + \frac{1}{2}\right)}. \end{aligned} \tag{5.28}$$

For sufficiently large values of dimension n, the SNR expressions (5.20) and (5.28) for the floating point errors can be approximated as

$$SNR_1 \approx \frac{12}{n\mathbf{u}^2}, \quad (\text{for } \mathbf{x}^T\mathbf{y}) \tag{5.29}$$

$$SNR_2 \approx \frac{18}{n\mathbf{u}^2}, \quad (\text{for } \mathbf{x}^T\mathbf{x}). \tag{5.30}$$

5.2.3 Comparison with Fixed Point SNR

For fixed point computation of dot product, we have to scale the entries of $\mathbf{x}$ and $\mathbf{y}$ to avoid overflow. Let x_{max} and y_{max} be the maximum magnitudes of the entries of $\mathbf{x}$ and $\mathbf{y}$ respectively. The maximum magnitude of the dot product in this case will be

$$(\mathbf{x}^T\mathbf{y})_{max} = nx_{max}y_{max}. \tag{5.31}$$

To avoid overflow, we have to ensure $(\mathbf{x}^T\mathbf{y})_{max} < 1$. One way to achieve this is to scale down entries of $\mathbf{x}$ and $\mathbf{y}$ by $(nx_{max}y_{max})^{1/2}$. Let $\mathbf{u}_f$ be the unit roundoff for the fixed point case ($\mathbf{u}_f = 2^{-(b+1)}$ for a b-bit representation). Then using the error model of [Opp76], we can write down the fixed point error variance as a sum of n mutually independent error sources, each with error variance $\mathbf{u}_f^2/12$. Thus, the variance of the fixed point roundoff error is $\sigma_{1f}^2 = n\mathbf{u}_f^2/3$. The SNR expression for this case will be

$$\begin{aligned} SNR_{f1} &= \frac{E[(\mathbf{x}^T\mathbf{y})^2/(nx_{max}y_{max})^2]}{\sigma_{1f}^2} \\ &= \frac{12\sigma_x^2\sigma_y^2}{x_{max}^2y_{max}^2}(\frac{1}{n^2\mathbf{u}_f^2}). \end{aligned} \tag{5.32}$$

A similar analysis for the case of computing $\mathbf{x}^T\mathbf{x}$ gives

$$SNR_{f2} = \frac{12\sigma_x^4}{x_{max}^4}(\frac{1}{n\mathbf{u}_f^2}). \tag{5.33}$$

In the derivation of (5.32) and (5.33) we assume that all the entries x_i and y_j are mutually independent zero mean Gaussian random variables with variance σ_x^2 and σ_y^2 respectively. Comparing (5.29) with (5.32), we can conclude that the performance of fixed point computation of $\mathbf{x}^T\mathbf{y}$ degrades faster with n than the corresponding floating point evaluation. However (5.30) and (5.33) indicate that in the case of computation of $\mathbf{x}^T\mathbf{x}$, both the SNR expressions have the same functional dependence on dimension n. These results are tabulated in Table 5.1.

Dot Product	Floating point Error Variance	SNR	
		Floating point	Fixed point
$x^T y$	N^2	$1/N$	$1/N^2$
$x^T x$	N^3	$1/N$	$1/N$

Table 5.1. Dependence of roundoff errors on the dimension N for dot product computations.

5.2.4 Simulations

We verify the results developed for the floating point error variance analysis by simulations. The simulations were performed on a MIPS computer using single precision floating point arithmetic. This representation has a 24-bit mantissa and an 8-bit exponent. Double precision results were used as substitutes for the exact (infinite precision) values of the dot product. The elements of the vectors $\mathbf{x}$ and $\mathbf{y}$ were generated by a random number generator, to approximate a zero mean Gaussian probability distribution with unit variance. The errors e_1 and e_2 were computed for 100 different sets of vectors $\mathbf{x}$ and $\mathbf{y}$. The variances were estimated using these 100 values. Fig. 5.2 shows variance of e_1 as a function of n and Fig. 5.3 shows the corresponding SNR plot. Fig. 5.4 and Fig. 5.5 show the variance and SNR plots for the error e_2 as a function of n. We see that there is a close agreement between the results derived in the chapter, and the actual results obtained. It is also interesting to see that the error e_2 is much higher than e_1 for a given n due to high correlation of data but the SNR values are about the same.

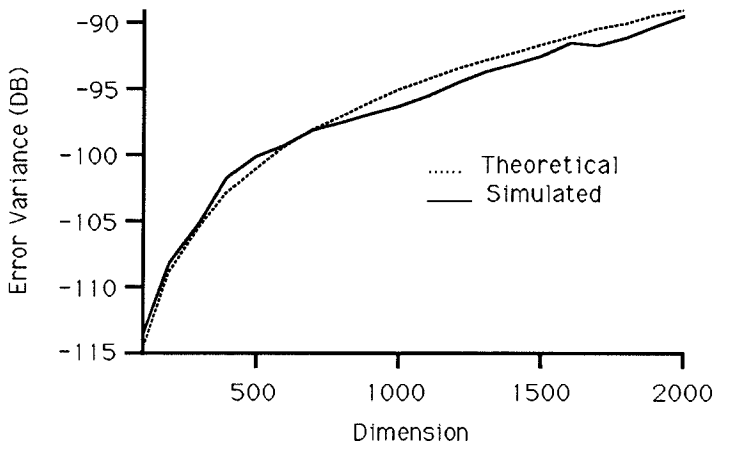


Fig. 5.2 Error variance for the error $e_1 = fl(\mathbf{x}^T\mathbf{y}) - \mathbf{x}^T\mathbf{y}$.

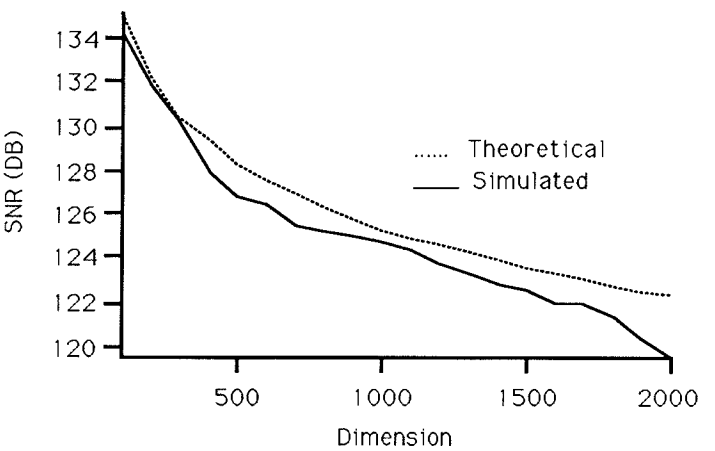


Fig. 5.3 SNR for dot product $\mathbf{x}^T\mathbf{y}$.

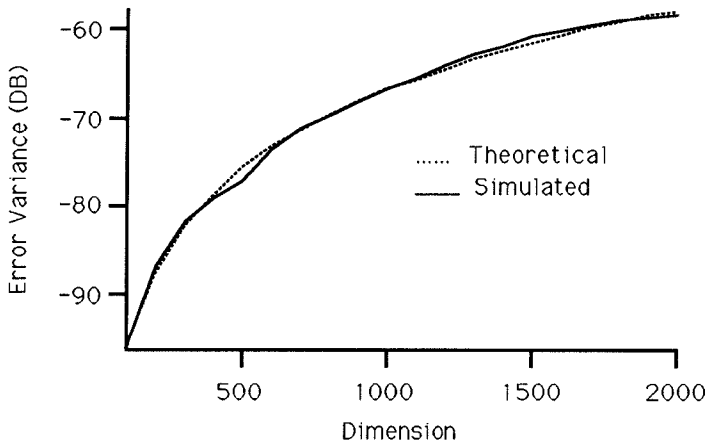


Fig. 5.4 Error variance for the error $e_2 = fl(\mathbf{x}^T\mathbf{x}) - \mathbf{x}^T\mathbf{x}$.

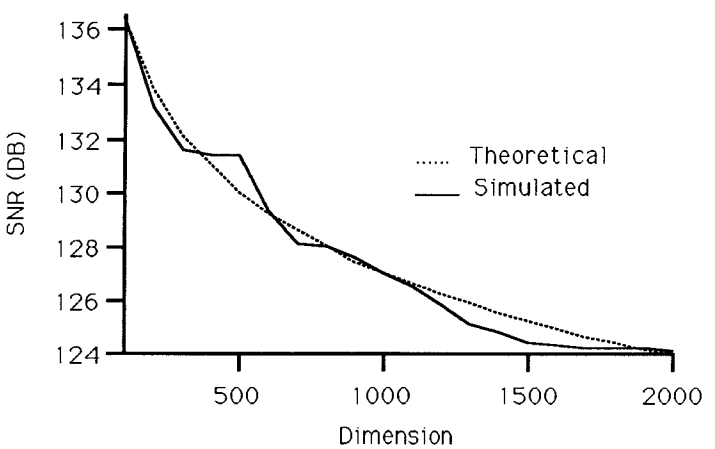


Fig. 5.5 SNR for dot product $\mathbf{x}^T\mathbf{x}$.

Although the SNR values obtained here are very high, one should bear in mind that in a signal processing application, these computations are performed over and over again, causing a considerable degradation in the SNR value of the entire algorithm. Thus, understanding roundoff errors involved in dot product computations is important. Also, as we shall see in the next section, the results derived in this section are useful for derivation of similar results for more complex algorithms.

5.3 QR Decomposition

Consider the problem of factorizing a square matrix $\mathbf{A} \in R^{M\times M}$ as $\mathbf{A} = \mathbf{QR}$, where $\mathbf{Q} \in R^{M\times M}$ is orthogonal and $\mathbf{R} \in R^{M\times M}$ is an upper triangular matrix. As we have already mentioned, this factorization is used in a variety of signal processing algorithms. The matrix $\mathbf{A}$ in general has nonzero entries,

$$\mathbf{A} = \begin{bmatrix} \times & \times & \dots & \times \\ \vdots & \vdots & \ddots & \vdots \\ \times & \times & \dots & \times \end{bmatrix}. \tag{5.34}$$

To calculate the desired factorization, we premultiply $\mathbf{A}$ by a sequence of orthogonal matrices which reduce it to an upper triangular form. Let these orthogonal matrices be denoted as $\mathbf{Q}_0^T$, $\mathbf{Q}_1^T$ $\dots$ $\mathbf{Q}_{N-1}^T$. This process of successive premultiplications can be written as

$$\mathbf{Q}_{N-1}^T \dots \mathbf{Q}_1^T \mathbf{Q}_0^T \mathbf{A} = \mathbf{R}. \tag{5.35}$$

Since the product of orthogonal matrices is also orthogonal, we can write the above equation as

$$\mathbf{Q}^T \mathbf{A} = \mathbf{R} \tag{5.36}$$

where $\mathbf{Q}^T = \mathbf{Q}_{N-1}^T \dots \mathbf{Q}_1^T \mathbf{Q}_0^T$. Looking at (5.36), we can see that we have achieved the desired factorization because $\mathbf{Q}^T = \mathbf{Q}^{-1}$ for an orthogonal matrix. The questions that remains to be answered is how to find the sequence of matrices $\mathbf{Q}_{N-1}^T$,

..., $\mathbf{Q}_1^T$, $\mathbf{Q}_0^T$ and the number N. The orthogonal matrices are themselves calculated using on the entries of $\mathbf{A}$. These matrices are used to modify the columns of $\mathbf{A}$ as follows. First, we introduce zero elements in the entries under the diagonal of the first column only by premultiplying by an appropriate number of orthogonal matrices. The modified matrix now looks as follows

$$\begin{bmatrix} \times & \times & \dots & \times \\ 0 & \times & \dots & \times \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \times & \dots & \times \end{bmatrix}. \tag{5.37}$$

Now we select a second set of orthogonal matrices such that the elements of the second column under its diagonal entry are made zero. The matrix now has the following structure

$$\begin{bmatrix} \times & \times & \times & \dots & \times \\ 0 & \times & \times & \dots & \times \\ \vdots & 0 & \times & \dots & \times \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \times & \dots & \times \end{bmatrix}. \tag{5.38}$$

Successively applying this procedure results in an upper triangular matrix. The key step in this triangularization is to modify a column vector such that all the entries except the first one are zero. This procedure is repeated from the first column to the last. This step

$$\begin{bmatrix} \times \\ \times \\ \vdots \\ \times \end{bmatrix} \rightarrow \begin{bmatrix} \times \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.39}$$

is achieved using either Givens Rotations (GR) or Householder Transformation (HT). In GR, this is achieved by making one entry zero at a time (as explained later). In HT, one finds out the appropriate orthogonal matrix such that premultiplying a vector directly results in the desired conversion (5.39). Thus, for the complete triangularization of an $M \times M$ matrix using GR, we need $N = M(M-1)/2$ matrix premultiplications. This number is $N = M - 1$ for the HT technique. We

will now take a look at the Givens rotations and Householder transformation and see how the orthogonal matrices $\mathbf{Q}_i$ are chosen.

5.4 Givens Rotations

5.4.1 Givens Rotation for upper triangularization

Consider a 2×2 matrix

$$\mathbf{Q}^T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}. \tag{5.40}$$

Let the entries of this matrix satisfy

$$\cos\theta = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \quad \sin\theta = \frac{-x_j}{\sqrt{x_i^2 + x_j^2}} \tag{5.41}$$

where x_i and x_j are two real numbers. Using the above two equations, we can establish that

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} \tag{5.42}$$

where $r = \sqrt{x_i^2 + x_j^2}$. It can be easily verified that the matrix defined in (5.40) is orthogonal. This matrix is called a Givens Rotation (GR) matrix (of dimension 2). We have shown in (5.42) that choosing the elements of $\mathbf{Q}$ using x_i and x_j produces a zero at the place of x_j after premultiplication by $\mathbf{Q}$. This property is used in the triangularization procedure. For a column vector of dimension $M \times 1$, suppose we want to make the jth entry zero using the jth entry x_j and the ith entry x_i, the

corresponding $M \times M$ orthogonal matrix will have the form:

$$\mathbf{Q}_{ij}^T = \begin{array}{c} \\ \\ i \\ \\ j \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & i & & j & \\ \end{array} \\ \begin{pmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \dots & cos\theta & \dots & sin\theta & \dots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \dots & -sin\theta & \dots & cos\theta & \dots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{pmatrix} \end{array} \tag{5.43}$$

such that (5.42) holds. Premultiplication by this matrix produces a zero at the jth entry as follows

$$\mathbf{Q}_{ij}^T \begin{bmatrix} \times \\ \vdots \\ x_i \\ \vdots \\ x_j \\ \vdots \\ \times \end{bmatrix} = \begin{bmatrix} \times \\ \vdots \\ r_i \\ \vdots \\ 0 \\ \vdots \\ \times \end{bmatrix}. \tag{5.44}$$

The element x_i is called the pivot for the Givens Rotation. The element r_i is called the residue of GR. Since one GR premultiplication introduces one zero in the resultant vector, the result in (5.39) is obtained by successively multiplying by GR which are computed using the first entry of the vector as the pivot. The sequence in which a column vector is reduced to the desired form is:

$$\begin{bmatrix} \times \\ \times \\ \times \\ \times \\ \vdots \\ \times \end{bmatrix} \rightarrow \begin{bmatrix} \times \\ 0 \\ \times \\ \times \\ \vdots \\ \times \end{bmatrix} \rightarrow \begin{bmatrix} \times \\ 0 \\ 0 \\ \times \\ \vdots \\ \times \end{bmatrix} \dots \rightarrow \begin{bmatrix} \times \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}. \tag{5.45}$$

This makes it clear that the residue computed in the previous step is used as the

pivot in the next step. This observation is very important in the roundoff error analysis of the GR algorithm.

5.4.2 Error Variance Analysis of Givens Rotation

When the steps described in (5.45) are performed in finite bit precision, the roundoff errors occur at two stages:

1. computation of the entries of the orthogonal matrix, and
2. computation of the residue.

As explained before by equation (5.45), several GR multiplications are required for each column. We will perform error variance analysis of the GR method by first taking a closer look at multiplication by a single GR only and then extending the result to the entire procedure in (5.45).

Multiplication by a Givens Rotation

Consider the matrix-vector multiplication:

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} = \begin{bmatrix} b_i \\ b_j \end{bmatrix} \tag{5.46}$$

satisfying (5.41). We will henceforth use the abbreviations $c = \cos\theta$ and $s = \sin\theta$. If we perform exact calculations, then we get $b_i = \sqrt{x_i^2 + x_j^2}$ and $b_j = 0$. For a finite bit implementation, there will be errors introduced in the computation of these quantities. We want to find out the means and variances of the errors

$$e_i = fl(b_i) - b_i, \quad e_j = fl(b_j) - b_j. \tag{5.47}$$

We have seen that the residue computed by one GR is used as the pivot for the next GR. Hence, the element x_i will in general have roundoff error associated with it. For the sake of simplicity, we are however going to assume that x_i is exact. The only errors introduced are in the computations of c, s, b_i and b_j.

Errors Introduced In the Computation of c and s:

The entries of the orthogonal matrix are computed as follows:

$$
\begin{aligned}
a &= fl(fl(x_i^2) + fl(x_j^2)) \\
b &= fl(a^{1/2}) \\
c &= fl(x_i/b) \\
s &= fl(x_j/b).
\end{aligned} \tag{5.48}
$$

We assume that all floating point roundoff errors are mutually independent random variables. A uniform probability of distribution in the range $[-\mathbf{u}, \mathbf{u}]$ is assumed for each roundoff error unless otherwise mentioned. If we write down the errors in the equations (5.48), we get

$$
\begin{aligned}
a &= [x_i^2 + x_j^2](1 + \epsilon_3) \\
b &= a^{1/2}(1 + \epsilon_4) \\
c &= (x_i/b)(1 + \epsilon_5) \\
s &= (x_j/b)(1 + \epsilon_6).
\end{aligned} \tag{5.49}
$$

To make the derivation easier, we assume that the error ϵ_3 is uniformly distributed in the range $[-2\mathbf{u}, 2\mathbf{u}]$. The error ϵ_3 is the combined effect of the errors introduced due to squaring and addition. Although this assumption does not strictly fit the error model we have been using so far, it makes the derivation easier without changing it significantly. From (5.49), we can derive the following results

$$
fl(c) = \frac{c(1 + \epsilon_5)}{(1 + \epsilon_3)^{1/2}(1 + \epsilon_4)}, \quad fl(s) = \frac{s(1 + \epsilon_6)}{(1 + \epsilon_3)^{1/2}(1 + \epsilon_4)}. \tag{5.50}
$$

Thus, if $fl(c) = c(1 + \epsilon_c)$ and $fl(s) = s(1 + \epsilon_s)$ then from (5.50) we get,

$$
\epsilon_c \approx \epsilon_5 - \frac{\epsilon_3}{2} - \epsilon_4, \ \epsilon_s \approx \epsilon_6 - \frac{\epsilon_3}{2} - \epsilon_4. \tag{5.51}
$$

We have used the following approximations in this derivation.

1. All terms involving product of two errors are neglected because they are small.
2. $(1+\epsilon)^q = 1+q\epsilon$ for any rational number q. This approximation holds because the errors ϵ are much smaller than unity. From (5.51) we can see that errors ϵ_c and ϵ_s are zero mean and

$$\begin{aligned} E[\epsilon_c^2] &= E[\epsilon_5^2] + E[\epsilon_3^2/4] + E[\epsilon_4^2] \\ &= u^2/3 + u^2/3 + u^2/3 + O(u^3) = u^2 + O(u^3). \end{aligned} \tag{5.52}$$

The $O(u^3)$ term is introduced to account for the cross product terms. Similarly, we get $E[\epsilon_s^2] = u^2 + O(u^3)$.

Multiplication by a Givens Rotation

We compute the results b_i and b_j as

$$b_i = fl(fl(cx_i) + fl(sx_j)), \quad b_j = fl(fl(-sx_i) + fl(cx_j)). \tag{5.53}$$

Again, writing out the errors incurred in the computations, we get

$$\begin{aligned} b_i &= \big[cx_i(1+\epsilon_c)(1+\epsilon_7) + sx_j(1+\epsilon_s)(1+\epsilon_8)\big](1+\epsilon_1) \\ b_j &= \big[cx_j(1+\epsilon_c)(1+\epsilon_9) - sx_j(1+\epsilon_s)(1+\epsilon_{10})\big](1+\epsilon_2) \end{aligned} \tag{5.54}$$

Using the above equations, we can write down the expressions for the errors e_i and e_j defined in (5.47) can be approximately written down as

$$\begin{aligned} e_i &\approx cx_i(\epsilon_1 + \epsilon_7) + sx_j(\epsilon_s + \epsilon_1 + \epsilon_8) \\ e_j &\approx cx_j(\epsilon_2 + \epsilon_9) - sx_j(\epsilon_s + \epsilon_2 + \epsilon_{10}). \end{aligned} \tag{5.55}$$

From the above expressions, we can see that both the errors have zero mean. Squaring both sides of the equations in (5.55) and taking expected value, we arrive at the result

$$E[e_i^2 + e_j^2] = \frac{10(E[x_i^2] + E[x_j^2])\mathbf{u}^2}{3} + O(u^3). \tag{5.56}$$

The derivation makes use of the fact that all the errors, x_i and x_j are mutually independent. This expression gives the energy of the error between the actual results of GR multiplication and the computed results.

Multiplication by a Sequence of GR

Now we consider the problem of introducing $M-1$ zeroes in an $M \times 1$ vector as shown in (5.45). The first element is used as the pivot for all the GR. There will be $M-1$ such GR performed. Each GR will in turn change the value of the pivot. Let the starting vector be defined as $\mathbf{x} = (x_1\ x_2\ \ldots x_M)^T$. Let the residue at the end of the kth iteration be r_k. Clearly $r_1 = x_1$. An efficient way of computing the residue is not to compute the values of c and s [Wil65] but to directly evaluate the residue as

$$r_k = fl(fl([fl(r_{k-1})]^2) + fl(x_k^2))^{1/2},\ k = 2, \ldots M-1. \tag{5.57}$$

The computation of c and s is however needed to evaluate the other entries in the partially triangularized data matrix. We also assume that the entry which is supposed to become zero is indeed zero [Wil65]. For each computation (5.57), we can write

$$fl(r_k) = \left[\left(r_{k-1}^2(1+\alpha_{k-1}) + x_k^2(1+\beta_k)\right)(1+\gamma_k)\right]^{1/2}(1+\delta_k),\ k = 2, \ldots, M-1, \tag{5.58}$$

where α_{k-1} and β_{k-1} are the roundoff errors due to squaring, γ_k is the error due to addition and δ_k is the error due to square rooting. For the first iteration, $r_1 = x_1$. Applying this recursively, we get

$$r_M = \left(x_1^2(1+e_1) + \ldots + x_M^2(1+e_M)\right)^{1/2}(1+\delta_M) \tag{5.59}$$

where

$$e_k = \sum_{i=k}^{M-1} \alpha_i + \sum_{i=k}^{M} \gamma_i + 2\sum_{i=k}^{M-1} \delta_i + \beta_k. \tag{5.60}$$

Again, we assume that all the roundoff errors are mutually independent, zero mean random variables uniformly distributed in the range $[-\mathbf{u}, \mathbf{u}]$. Then, the cross-correlation between two errors given by (5.60) above can be written as

$$E[e_k e_l] = 6\frac{\mathbf{u}^2}{3}(M-k), \quad k \geq l. \tag{5.61}$$

The expression (5.59) for the final residue, can be written as

$$fl(r_M) = \left(r_M + \frac{\sum x_k^2 e_k}{2\sqrt{\sum x_k^2}}\right)(1+\delta_M). \tag{5.62}$$

Hence if $e_g = fl(r_M) - r_M$, then

$$E[e_g^2] = \frac{1}{4}E\left[\frac{\sum\sum x_k^2 x_l^2 e_k e_l}{\sum x_i^2}\right] + E[r_M^2]E[\delta_M^2] \tag{5.63}$$

In the derivation of (5.63), we take into account the dominating terms only. In the derivation, all the terms involving higher powers of $\mathbf{u}$ are grouped together as $O(\mathbf{u}^3)$. We assume that all the entries of $\mathbf{x}$ are mutually independent and are independent of all the roundoff errors and have variance σ_x^2 each.

The first term in (5.63) involves expected value computation for division of two random variables. This can be simplified as division of the expected values of those random variables, if the random variable in denominator varies slower than the numerator. Thus, the assumption we are making is that $E[X/Y] = E[X]/E[Y]$. This assumption has been successfully used in similar derivations before [Mat90]. We use this simplification to derive the following result:

$$E[e_g^2] = \frac{M^2\sigma_x^2\mathbf{u}^2}{12} + O(M\mathbf{u}^2) + O(\mathbf{u}^3). \tag{5.64}$$

For sufficiently large dimension M, the last two terms can be neglected.

5.5 Householder Transformation

The task of introducing zeroes in a column vector as shown in (5.39) can be performed using one matrix multiplication only. Householder matrices are used for this computation.

5.5.1 Review of Householder Matrices

Definition: Any matrix which can be represented as

$$\mathbf{H}_v = \mathbf{I} - \frac{\mathbf{v}\mathbf{v}^\dagger}{\mathbf{v}^\dagger\mathbf{v}} \tag{5.65}$$

where $\mathbf{v} \neq \mathbf{0}$ to avoid trivial case (if $\mathbf{v} = \mathbf{0}$, then $\mathbf{H}_v = \mathbf{I}$), is called a Householder matrix. The properties of Householder matrices are discussed in details in [Gol89]. Householder matrices are orthogonal due to the form (5.65). To null out all the entries of a vector $\mathbf{x}$ except the first one, we can perform the matrix-vector multiplication $\mathbf{H}_v\mathbf{x}$ which gives

$$\mathbf{H}_v\mathbf{x} = \begin{bmatrix} r_h \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.66}$$

if

$$\mathbf{v} = \mathbf{x} + s\beta\mathbf{e}_1 \tag{5.67}$$

where s is the sign of the first entry x_1 of $\mathbf{x}$, $\beta = \sqrt{\sum x_i^2}$ and $\mathbf{e}_1 = (1\ 0\ \ldots\ 0)^T$. In the triangularization of an $M \times M$ matrix, one needs to compute $M-1$ Householder matrices to achieve the desired result. This method is known for its superior numerical properties [Wil65], but is more expensive to implement in hardware because it is not pipelineable like the GR method [Hay86].

As in the case of GR, there are two steps involved in the HT procedure. The first is to compute the entries of the Householder matrix, and the second is to compute the residue r_h in equation (5.66). For a similar situation for the GR

case, we made an assumption that the elements which are to become zero after multiplication do indeed become zero. For the case of HT however, the sequence of computations can be arranged such that entries on the right hand side of (5.66) which are supposed to become zero after multiplication do indeed remain exactly zero under quantization. Hence we study the error involved in the computation of the residue r_h only.

5.5.2 Error Variance Analysis of HT:

We have seen that the elements which are supposed to be zero, are indeed zero even under quantization. Using (5.66) and (5.67), it is easy to see that $r_h = \beta = \sqrt{\sum x_i^2}$. Hence computation of the residue involves computation of dot product. We will therefore use the results form section 5.2.

Problem Statement

For the floating point evaluation of (5.66), let the error incurred due to quantization be

$$e_h = fl(r_h) - r_h, \tag{5.68}$$

then we want to find out $E[e_h]$ and $E[e_h^2]$.

The following sequence of computation is used

$$a = fl(\mathbf{x}^T\mathbf{x}) \tag{5.69a}$$

$$\beta = r_h = fl(a^{1/2}). \tag{5.69b}$$

Error Analysis

For computation (5.69a), using equation (5.11) for dot product computation we can write

$$a = \sum_{i=1}^{M} x_i^2(1+\gamma_i) \tag{5.70}$$

with all the quantities as defined before. Hence for the second step, we can write

$$r_h = \left(fl(\sum x_i^2 + \sum x_i^2\gamma_i)\right)^{1/2}$$

$$= (\sum x_i^2 + \sum x_i^2 \gamma_i)(1 + \epsilon_1)$$
$$\approx \sum x_i^2 (1 + \frac{\sum x_i^2 \gamma_i}{2 \sum x_i^2}) \tag{5.71}$$

where the error ϵ_1 incurred in the square root evaluation is neglected because it does not change the final result significantly. Hence we get

$$e_h = \frac{\sum x_i^2 \gamma_i}{2\sqrt{\sum x_i^2}}. \tag{5.72}$$

This gives $E[e_h] = 0$ and

$$E[e_h^2] = \frac{1}{4} E[\frac{\sum_i \sum_j x_i^2 x_j^2 \gamma_i \gamma_j}{\sum_i x_i^2}]. \tag{5.73}$$

Using assumptions similar to the ones used in the derivation of (5.64) from (5.63), we get,

$$E[e_h^2] = \frac{M^2 \sigma_x^2 \mathbf{u}^2}{72} + O(M\mathbf{u}^2) + O(\mathbf{u}^3). \tag{5.74}$$

The last two terms in (5.74) account for the less significant cross product terms which we have neglected throughout the derivation. The results (5.23) are also used in the above derivation.

5.6 Comparison of the Error Variances for GR and HT

Both these techniques are used to perform upper triangularization of matrices. As we have already seen, the elements on the diagonal of the upper triangular matrix **R** are the residues of the GR and HT computations. The errors involved in the computations are important because they directly affect the eigenvalues of the upper triangular matrix **R**.

Under infinite bit precision the residues calculated using either GR or HT will be identical. The results will be different when we perform the computations using finite bit representation. To get an idea about the energy of the error introduced in

the computation of the residue as in (5.39), we have performed the error variance analysis.

The results of the theoretical analysis presented indicate that HT has better error performance "on an average." If we compare the results (5.64) and (5.74), we can see that the error variance expressions differ by a factor of 6, or 8 DB on the logarithmic scale. This means that the SNR for HT is better than the SNR for GR by about 8 DB.

Simulations

The above theoretical results are compared with simulations on a mainframe computer. The GR and HT algorithms are implemented using single precision floating point arithmetic. As mentioned before, the double precision results were used as a substitute for the exact results. The computations shown in (5.39) were simulated for different values of dimension M of the vector $\mathbf{x}$ using GR and HT. For each value of M, the entries of the vectors were generated as zero mean, mutually independent random variables with Gaussian probability distribution having variance $\sigma_x^2 = 0.1$. The error variances for GR and HT were computed from the error values obtained for 100 such vectors. A plot of how the error variances $E[e_h^2]$ and $E[e_g^2]$ vary with the dimension M is shown in Fig. 5.5. Also shown in the figure are plots for the theoretical results given by (5.64) an d (5.74). It can be seen that the simulation results closely agree with the theoretical results. The plot also shows that the difference in the error variance values for GR and HT is constant (about 8 DB). The simulations demonstrate the usefulness of the error variance analysis presented in this chapter.

5.7 Discussion

In this chapter, we have introduced the concept of error variance analysis.

This analysis is performed to answer probabilistic questions of the type, "What is the typical value of the roundoff errors introduced in a computation ?" We derive the expression for the variance of roundoff error incurred in the computation of a result. This idea is elaborated by performing error variance analysis of dot

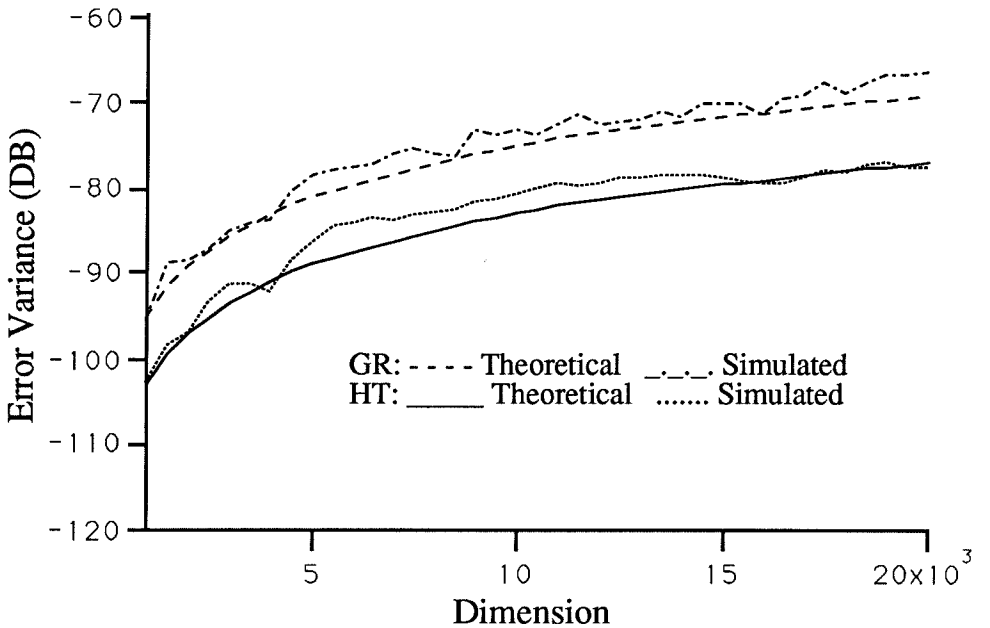


Fig. 5.6 Error variance v/s dimension curves for GR and HT.

product computation. The reason to choose dot product computation for demonstration of this idea is that it is the most basic computational step involved in all

signal processing algorithms. The theoretical analysis and simulations show that the SNR of dot product computation is relatively insensitive to the correlation of the input data. It has been shown that the SNR degrades as $1/N$ where N is the dimension of the vectors involved in the computation.

The SNR values of algorithms can be used to compare their relative error performances. The GR and HT algorithms are compared on the basis of their SNR figures. These algorithms are used to perform upper triangularization of matrices. The diagonal elements of this upper triangular matrix are shown to be the residues of the GR and HT computations. We have derived expressions for the error variances for the computation of the residues using GR and HT. These results have been compared to show that the SNR for HT is 8 DB more than the SNR for GR.

The error variance analysis described in this chapter does not have to be performed in real time. If the input statistics is known, one can precompute the SNR and error variance values for an algorithm. The computational complexity for the SNR computation is therefore not very crucial. In choosing the appropriate algorithm for an application, the performance of that algorithm under quantization is an important consideration. Some other considerations are the computational complexity, pipelineability etc. Choosing an algorithm with a lower SNR ensures that smaller roundoff error is incurred on an average. It is important to remember that this analysis does not give any idea about the numerical stability of the algorithm (it implicitly assumes that the algorithm is numerically stable).

References

[Ada83] J. W. Adams and A. N. Willson, Jr., "A new approach to FIR digital filters with fewer multipliers and reduced sensitivity," IEEE Trans. Circuits and Systems, vol. 30, pp. 277-283, May. 1983.

[And79] B. D. O. Anderson and J. Moore, *Optimal filtering,* Prentice Hall Inc., 1979.

[Ans86] R. Ansari, "Multi-level IIR digital filters," IEEE Trans. Circuits and Systems, vol. 33, pp. 337-341, March 1986.

[Cad79] J. A. Cadzow, "An extrapolation procedure for band-limited signals," IEEE Trans. Acoust., Speech, Signal Processing, vol. 27, pp. 4-12, Feb. 1979.

[Cas87] IEEE Trans. on Circuits and Systems, *special issue on adaptive systems and applications,* vol. 34, no. 7, July 1987.

[Chu87] J. Chun, T. Kailath and H. Lev-Ari," Fast parallel algorithms for QR and triangular factorization," SIAM Journal of Stat. Comput., pp. 899-913, November 1987.

[Cio84] J. Cioffi and T. Kailath," Fast recursive least squares transversal filters for adaptive filtering," IEEE Trans. ASSP, pp. 304-337, April 1984.

[Cio87] J. Cioffi," Limited precision effects in adaptive filtering," IEEE Trans. Circuits and Systems, vol. 34, no. 7, July 1987.

[Cla81] G. Clark, S. Mitra and S. Parker," Block implementation of adaptive digital filters," IEEE Trans. ASSP, vol. 29, no. 3, June 1981.

[Cro83] R. E. Crochiere and L. R. Rabiner, *Multirate digital signal processing,* Prentice Hall, Inc., 1983.

[Dav58] W. B. Davenport, Jr., and W. L. Root, *An Introduction to the Theory of Random Signals and Noise* , McGraw Hill Co., New York, NY, 1958.

[Del85] P. Delsarte, A. J. E. M. Janssen, and L. B. Vries, "Discrete prolate spheriodal wave functions and interpolation," SIAM J. of Appl. Math., vol. 45, No. 4, pp. 641-650, Aug. 1985.

[Gar91] W. A. Gardner, "Exploitation of spectral redundancy in cyclostationary signals," IEEE Signal Processing Magazine, April 1991.

[Gil87] A. Gilloire, "Experiments with sub-band acoustic echo cancellers for teleconferencing," Proc. IEEE Int. Conf. ASSP, pp. 2141-2144, 1987.

[Gil88] A. Gilloire and M. Vetterli, "Adaptive Filtering in Sub-bands," Proc. IEEE International Conf. On ASSP, 1988, pp. 1572-1575.

[Gol89] G. H. Golub and C. F. Van Loan, *Matrix computations,* The Johns Hopkins Univ. Press, 1989.

[Hay83] M. H. Hayes and R. W. Schafer, "On the band-limited extrapolation of discrete signals," Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, pp. 1450-1453, April 1983.

[Hay86] S. Haykin, *Adaptive filter theory,* Prentice Hall, Inc., 1986.

[Hon84] M. Honig and D. Messerschmitt, *Adaptive filters: Structures, Algorithms and Applications,* Kluwer Academic Publishers, 1984.

[How81] S. J. Howard, "Method for continuing Fourier spectra given by the fast Fourier transform," J. Opt. Soc. Am. vol. 71, pp. 95-98, Jan. 1981.

[Hsi87] C. Hsiao, "Polyphase filter matrix for rational sampling rate conversions," Proc. IEEE Int. Conf. ASSP, pp. 2173-2176, April 1987.

[Jai81] A. K. Jain and S. Ranganath, "Extrapolation algorithms for discrete signals with application in spectral estimation," IEEE Trans. Acoust., Speech,

Signal Processing, vol. 29, pp. 830-845, Aug. 1981.

[Jan84] A. J. E. M. Janssen and L. B. Vries, "Interpolation of bandlimited discrete-time signals by minimizing out-of-band energy," Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, pp. 12B.2.1-12B.2.4, 1984.

[Jur91] R. Jurgen and W. Schrieber, "All-digital TV's promise/problems," IEEE Spectrum, April 1991.

[Kal84] N. Kalaouptsidis, G. Carayannis, and D. Manolakis, "A fast sequential type algorithm for sequential least-squares filtering and prediction," IEEE Trans. on Automatic Control, vol. 29, pp. 752-755, Aug. 1984.

[Kel88] W. Kellerman, "Analysis and design of multirate systems for cancellation of acoustic echoes," Proc. Int. Conf. On ASSP, 1988, pp. 2570-2573.

[Kol78] D. P. Kolba and T. W. Parks, "Extrapolation and spectral estimation for bandlimited signals," Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, pp. 372-374, April 1978.

[Kol83] D. P. Kolba and T. W. Parks, "Optimal estimation for bandlimited signals including time domain constraints," IEEE Trans. Acoust., Speech, Signal Processing, vol. 31, pp. 113-122, Feb. 1983.

[Koi89] R. D. Koilpillai, P. P. Vaidyanathan, and S. K. Mitra, "On arbitrary level IIR and FIR filters," submitted.

[Kul86] U. W. Kulisch and L. Miranker," The arithmetic of the digital computer: A new approach," SIAM Review, vol. 28, no. 1, March 1986.

[Kun82] H. T. Kung, "Why systolic architectures?," IEEE Computer, vol. 15, pp. 37-46, 1982.

[Lee86] J. C. Lee and C. K. Un, "Block realization of multirate adaptive digital filters," IEEE Trans. Acoust. Speech Signal Proc., vol. 34, February 1986.

[Liu89] V. C. Liu and P. P. Vaidyanathan, "Finite length band-limited extrapolation of discrete signals," Proc. IEEE Int. Symp. on Circuits and Systems,

pp. 1037-1040, Portland, May 1989.

[Lju85] S. Ljung and L. Ljung, " Error propogation properties of recursive least squares algorithms," Automatica, vol. 21, no. 2, pp. 157-167, 1985.

[Mar76] J. D. Markel and A. H. Gray, Jr., *Linear prediction of speech,* Springer Verlag, 1976.

[Mar83] R. J. Marks, II, "Restoring lost samples from an oversampled band-limited signal," IEEE Trans. Acoust., Speech, Signal Processing, vol. 31, pp. 752-754, June. 1983.

[Mar86] S. A. Marple, *Digital Spectral Analysis with Applications,* Prentice Hall, Englewood Cliffs, NJ, 1986.

[McC73] J. H. McClellan, T. W. Parks, and L. R. Rabiner, "A computer program for designing optimum FIR linear phase digital filters," IEEE Trans. Audio Electroacoustics, vol. 21, pp. 506-526, Dec. 1973.

[Mor87] D. R. Morgan and A. Aridgides, "Interpolation and Extrapolation of an ideal band-limited random process," IEEE Trans. Acoust., Speech, Signal Processing, vol. 35, pp. 43-47, Jan. 1987.

[Neu14] Y. Neuvo, C.-Y. Dong and S. K. Mitra, " Interpolated finite impulse response filters," IEEE Trans. ASSP, pp. 563-570, June 1984.

[Opp76] A. V. Oppenheim and R. W. Schafer,*Digital Signal Processing,* Prentice Hall, Englewood Cliffs, NJ, 1976.

[Pap65] A. Papoulis, *Probability, Random Variables and Stochastic Processes,* McGraw Hill Co., New York, NY, 1965.

[Pap75] A. Papoulis, "A new algorithm in spectral analysis and band-limited extrapolation" IEEE Trans. Circuits and Systems, vol. 22, pp. 735-742, Sept. 1975.

[Pap77] A. Papoulis, *Signal Analysis,* McGraw Hill Book Co., 1977.

[Rus82] C. K. Rushforth, A. E. Crawford, and Y. Zhou, "Least-squares reconstruc-

tion of objects with missing high-frequency components," J. Opt. Soc. Am. vol. 72, pp. 204-211, Feb. 1982.

[Sab78] M. S. Sabri and W. Steenaart, "An approach to band-limited signal extrapolation: the extrapolation matrix," IEEE Trans. Circuits and Systems, vol. 25, pp. 74-78, Feb. 1978.

[San83] J. L. C. Sanz, and T. S. Huang, "Some aspects of band-limited signal extrapolation: Models, discrete approximations, and noise," IEEE Trans. Acoust., Speech, Signal Processing, vol. 31, pp. 1492-1501, Dec. 1983.

[Sat89] V. Sathe and P. P. Vaidyanathan, "Fast techniques for bandlimited extrapolation of finite length sequences," Caltech Internal Report, June 1989.

[Sat90a] V. Sathe and P. P. Vaidyanathan, "Error variance analysis of floating point dot product computations," *Proceedings of the twenty-fourth Asilomar Conference on Signals, Systems and Computers,* Pacific Grove, Calif. November 1990.

[Sat90b] V. Sathe and P. P. Vaidyanathan, "Efficient adaptive identification and equalization of bandlimited channels using multirate/multistage FIR filters," *Proceedings of the twenty-fourth Asilomar Conference on Signals, Systems and Computers,* Pacific Grove, Calif. November 1990.

[Sat91a] V. Sathe and P. P. Vaidyanathan, "Effects of multirate systems on the statistical properties of random inputs," *submitted, IEEE ASSP Transactions.*

[Sat91b] V. Sathe and P. P. Vaidyanathan, "Analysis of the effects of multirate filters on stationary random inputs, with application in adaptive filtring," *to be presented at the IEEE ASSP Conference, May 1991.*

[Sch81] R. W. Schafer, R. M. Mersereau and M. A. Richards, "Constrained iterative restoration algorithms," Proc. of the IEEE, vol. 69, pp. 432-450, April 1981.

[Slo88] D. T. Slock and T. Kailath, "Numerically stable fast recursive least-squares transversal filters," Proc. IEEE International Conf. on ASSP, 1988, pp. 1365-1368.

[Sul84] B. J. Sullivan, and B. Liu, "On the use of singular value decomposition and decimation in discrete-time band-limited signal extrapolation," IEEE Trans. Acoust., Speech, Signal Processing, vol. 32, pp. 1201-1212, Dec. 1984.

[Tho88] P. J. Thompson and M. T. Manry, "An efficient, noise tolerant, linear extrapolator," Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, pp. 1750-1753, April 1988.

[Ung76] G. Ungerboeck, "Fractional tap-spacing equalizer and consequences for clock recovery in data modems," IEEE Trans. on Comm., vol. 24, August 1976.

[Urk83] H. Urkowitz, *Signal Theory and Random Processing,* Artech House, Dedham, MA, 1983.

[Vai88] P. P. Vaidyanathan and S. K. Mitra, "Polyphase networks, block digital filtering, LPTV systems, and alias free QMF banks: A unified approach based on pseudocirculants," IEEE Trans. Acoust. Speech Signal Proc., vol. 36, March 1988.

[Vai90] P. P. Vaidyanathan, "Multirate digital Filters, filter Banks, polyphase networks, and applications: A tutorial," Proc. of IEEE, Vol. 78, No 1, Jan 1990.

[Wid85] B. Widrow and S. Stearns, *Adaptive Signal Processing,* Prentice Hall, Englewood Cliffs, NJ, 1985.

[Wil65] J. H. Wilkinson, *The algebraic eigenvalue problem,* Oxford Science Publications, 1965.

[Xu83] W. Y. Xu and C. Chamzas, "On the extrapolation of band-limited func-

tions with energy constraints," IEEE Trans. Acoust., Speech, Signal Processing, vol. 31, pp. 1222-1234, Oct. 1983.

[You78] D. C. Youla, "Generalized image restoration by the method of alternating orthogonal projections," IEEE Trans. Circuits and Systems, vol. 25, pp. 694-702, Sept. 1978.