Multistage Adaptive Filtering in a Multirate Digital Signal Processing System

by

Jen Mei Chen

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science
and
Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1993

Signature redacted

Author ....
Department of Electrical Engineering and Computer Science
May 20, 1993

Signature redacted

Certified by ....
Professor George Verghese
Department of Electrical Engineering
MIT Thesis Supervisor

Signature redacted

Certified by ....
Dr. Ganesh Rajan
Tektronix Laboratories
VI-A Cooperating Company Thesis Supervisor

Signature redacted

Accepted by ....
Professor Campbell L. Searle
Chairman, Departmental Committee on Graduate Students

Multistage Adaptive Filtering in a Multirate Digital Signal Processing System

by

Jen Mei Chen

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 1993, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science
and
Master of Science

Abstract

Recent years have seen the increasing application of multirate digital signal processing in the adaptive filtering context, leading to improvement both in computational complexity and in convergence speed. This thesis proposes a modification to the current multirate, adaptive filter structure, whereby we decompose the adaptive filter into smaller ones and insert each filter in between successive decimation stages. We explore this new multistage, multirate adaptive filter in the context of adaptive equalization of a downsampled, narrow-band signal.

Two types of filter-design techniques are studied: 1) the classic least-squares method, and 2) the frequency-domain block least mean square (FBLMS) algorithm. The first algorithm is in the time domain and is run non-adaptively while the second one is adaptive and in the frequency domain. With the aid of software simulations, we compare the multistage, multirate filter with the single-stage, multirate structure based on two performance criteria: 1) the residual error, and 2) the computational complexities of the filter design and operation. Our study shows that for our particular model of the system, the multistage, multirate scheme shows no clear improvement over the single-stage scheme for the non-adaptive least-squares method. For the adaptive FBLMS filter, the multistage structure demonstrates significant computational savings over the single-stage filter, but at the cost of an increased residual error at convergence.

Thesis Supervisor: Professor George Verghese
Title: Professor of Electrical Engineering

Thesis Supervisor: Dr. Ganesh Rajan
Title: Manager of DSP Group, Tektronix Laboratories

Acknowledgments

While waiting for my thesis to print out, I could easily recall many individuals I met over the years who have enriched my life in some ways. I would like to acknowledge and thank them here.

First, I would like to express my sincere gratitude to my thesis advisor, Professor George Verghese, for his careful reading of my thesis and for his many helpful comments which have improved the writing. Furthermore, his friendliness and accessibility is very much appreciated.

The work of this thesis was carried out in the Digital Signal Processing Group of Tektronix Laboratories in Beaverton, OR. I was fortunate to work with many knowledgeable and energetic people who had made my stay there very enjoyable. I would like to thank my Tektronix thesis advisor, Dr. Ganesh Rajan, for his enthusiastic support throughout my work on the thesis, even up to the very last minute. I would like to acknowledge Ben Ward for getting my feet wet on the thesis. I first learned about adaptive filtering and multirate DSP during my work with him on the calibration project for the Test & Measurement Division. In addition, Ben's help with computer-related questions are appreciated, too. I also wish to thank Ganesh and Ben for reading the electronic drafts of my thesis and Gail for express mailing the corrected versions back to me. (Thank God for emails and express mails!) Many special thanks also go to Dr. V. S. Somayazulu for helping me demystify many concepts related to my thesis and for his general guidance throughout my work. The discussions with Dr. Ajay Luthra were also valuable and appreciated. My interactions with the other individuals in the group–Craig, Dean, Norm, Mike, and Regis– were also rewarding. Finally, I cannot express enough heart-felt thanks to Gail Floyd who in many little ways had made my days at the Lab more pleasant.

Long distance communication with other individuals also contributed to my thesis. My email correspondences with Dr. Vinay Sathe and Professor John Shynk at the University of California, Santa Barbara have helped clarify some of my questions. I especially appreciate Professor Shynk's generous sharing of his knowledge, even though he has not met me before, by replying in great detail to my questions about the frequency-domain adaption.

I wish to acknowledge Mr. Vic Hansen of the Test & Measurement Division of Tektronix, Inc. for his financial support of my work and for presenting me with the calibration problem in the summer that had led to the development of this thesis. I also like to thank my advisors from my two previous VI-A assignments, Mr. Mike Cronk & the late Mr. Todd Mount and Dr. Kevin Smith & Dr. Tom Hawkins, for working with me and teaching me.

The three summers and one fall that I spent in Oregon would not have been as enjoyable without the friendship of some of the people I met there. I was fortunate enough to be "adopted" by two families– Mr. & Mrs. Ky Bui and Mr. Tran Thong & Mrs. Thuy Nguyen– who went out of their ways to welcome me to the life in Oregon. Their willingness to lend a hand, their friendliness, and their delicious cookings had made my stay more pleasant. Mrs. Marty Peiffer, the former VI-A coordinator of Tektronix, had also eased my transition to the new living environment. And then there are those summer interns who made living "all the way out there" much more fun. Thanks to the past and present VI-A interns who had made it possible: Kathy, Johnson, Garrison, Chris, Cecilia, Phil, Craig, Andy & Andy, Dan & Dan, Frances, Claude, David, and Peter.

I also wish to thank the many good friends who have stood (or stuck) by me during my last five years at MIT. To the "gang"–Tony, Ona, Mona, and Kiet–you guys are still great since the days we all first met at our freshman R/O picnic. I'll never forget all those birthday outings! And thanks for all those "Keep it up!" zwrites and emails throughout the writing of my thesis. Then there is one special friend who you can always count on. Thanks to Sujata–"Mama Su"–for always being there to listen and for making those long nights of problem sets so much fun. (Let's go out in the rain at 3 a.m. to search for ice cream!) A special thank to Jeff for his friendship, encouragement, and help with Latex for the thesis. And long live the VSA! What a a bunch of fun pranksters you all are! Thanks for your friendship and for all those great Thanksgiving and Senior dinners. (And I still couldn't believe what I did on stage!) And to the many other friends who in small ways make my days at the 'tute more bearable and who remind me that after all of this, times shared with friends are the memories that will last the longest and matter the most.

Much appreciation also goes to David for his care and support, especially during the last few weeks of the writing of my thesis. I would like to thank him for proof reading my thesis and helping me with Latex, and also for cooking and bringing me food while I typed away at the computer cluster.

I also appreciate my brother, Luong, for staying up until 4 a.m. with me on the night before the due date to help me with the graphs. I know my other siblings– Hao, An, Mei-Yi, and Alex–would have done likewise if they were around here. Thanks you all for making growing up in a big family so much fun. And yes, thanks to Grandma too for all the many wonderful tales of the past that have helped me to understand my roots. But my deepest gratitude and biggest hug go to mom and dad for their unfailing love, nurture, encouragement, and support throughout my life. Thanks for praying for me every single night and for always being there to listen and care. This thesis is dedicated to you both!

Above all, my humble gratitude to the Almighty and loving God who has guided and watched over me through each moment of my life.

I would like to acknowledge and thank the American Association of University Women (AAUW) Foundation for partially funding my Master's studies through the Selected Profession Fellowship (1992-1993).

Contents

1 Introduction **8**

1.1 The Big Picture 8

1.2 Motivation and Scope 9

2 Background **14**

2.1 Adaptive Filtering 14

2.1.1 Introduction 14

2.1.2 The Least Squares Filter 19

2.1.3 Computational Complexity 23

2.1.4 The Minimum Mean Square Error Method 24

2.1.5 Least Mean Square Algorithm 27

2.1.6 Frequency-Domain Adaptation 31

2.1.7 Frequency-Domain Block Least Mean Squares (FBLMS) . . . 34

2.1.8 Summary 41

2.2 Multirate Digital Signal Processing 42

2.2.1 Introduction 42

2.2.2 Review of Decimation 42

2.2.3 Multistage Decimation 44

3 The Multistage, Multirate Adaptive Filter **52**

3.1 Introduction 52

3.2 The Multistage, Multirate Adaptive Filter 54

3.3 Basis for Comparison 57

3.4 Motivation for a Simulation Study 59
3.5 A Model System for Simulation Studies 60
3.5.1 Introduction 60
3.5.2 The Distortion Channel 62
3.5.3 The Desired Response 65
3.5.4 The Delay Element 65
3.5.5 The Decimation Filters 67
3.6 The Least-Squares Filter 71
3.6.1 Computational Complexity of the Least-Squares Filter 72
3.6.2 Results of Time-Domain Least Squares 74
3.7 The FBLMS Filter 82
3.7.1 The Setup 82
3.7.2 Computational Complexity of the FBLMS Filter 83
3.7.3 Results of FBLMS with Gradient Constraint 86
3.8 The Modified FBLMS Filter 93

4 Conclusion **97**
4.1 Summary 97
4.2 Discussion and Suggestions for Future Research 99

List of Figures

1-1 Adaptive filtering in subbands. 11

1-2 Block diagram of a channel equalization in a multirate environment. . 11

2-1 Structure of a non-adaptive FIR filter. 15

2-2 Structure of an adaptive FIR filter. 15

2-3 Channel equalization with an adaptive filtering. 16

2-4 System identification with adaptive filtering. 16

2-5 Structure of the adaptive FIR filter. 18

2-6 Least squares as an orthogonal projection. 22

2-7 Structure of Overlap-Save Frequency-Domain Block LMS Filter. . . . 36

2-8 Block diagram for decimation by an integer factor M. 42

2-9 Single stage decimation and multistage decimation. 50

2-10 Structure of (a) single stage and (2) multistage decimation for factor of 10. 50

2-11 Example of filters for (a) single stage and (b), (c) multistage decimation. 51

3-1 Block diagram of of the single-stage decimation, single-stage adaptive filter 54

3-2 Block diagram equivalence for the decimator. 54

3-3 Block diagram of the two-stage decimation, two-stage adaptive filter. 55

3-4 Three different system configurations for study. 61

3-5 Characteristics of the distortion channel 64

3-6 Characteristics of the desired response. 66

3-7 Characteristics of the decimation-by-10 filter. 68

3-8 Characteristics of the decimation-by-5 filter. 69
3-9 Characteristics of the decimation-by-2 filter. 70
3-10 Comparison of three systems using the least-squares filter. 75
3-11 Study of decomposition of the two-stage equalizer using the least-squares method. 76
3-12 EFL for the different decompositions of the two-stage equalizer. . . . 76
3-13 Magnitude responses of the two-stage equalizer with 11 taps in the first stage. 78
3-14 Magnitude responses of the two-stage equalizer with 91 taps in the first stage. 80
3-15 LSE versus taps in the second-stage equalizer. 81
3-16 Three setups for simulation study. 87
3-17 Summary of the simulation results for the FBLMS algorithm. 88
3-18 Result of single-stage adaptive FBLMS filter with 128 taps. 90
3-19 Result of stage 1 in two-stage adaptive FBLMS filters with 64 taps each. 91
3-20 Result of stage 2 in two-stage adaptive FBLMS filters with 64 taps each. 92
3-21 Block diagram of adaptive filtering in subbands. 96

Chapter 1

Introduction

1.1 The Big Picture

Two of the prevalent problems confronting telecommunications today are (1) the imperfection and time-varying nature of the channel in which it takes place, and (2) the limitations of the hardware speed and storage. In recent years, many techniques in digital signal processing (DSP) have been developed to solve these problems. This thesis will study two of these techniques, digital adaptive filtering and multirate DSP.

Before describing the thesis problem, we will first place the subject matter in a broader context. The first problem mentioned above appears where the properties of the communication channel are unknown, where it is corrupted with noise or nonlinear distortion, or is time-varying due to some physical phenomena. In this situation, any signal sent through the channel will be distorted unfavorably and the received signal is no longer the message intended by the transmitter.

An example of this problem comes from our common experience of watching broadcast television programs where annoying "ghost" images usually appear. This occurs because the broadcast signal does not reach our television receptor directly but may be reflected from interfering buildings or trees in its path. This reflection from multipath is a delayed and attenuated version of the original signal which shows up as a ghost image on the television set. Thus, it is desirable to cancel the ghost in order to improve the picture quality, and it is easy to do so in the digital domain as will be possible with the advent of digital televisions and High Definition Television (HDTV). Other examples of this multipath problem appears in mobile and cellular communications, where the transmitted signal is speech. The problem is even more difficult here since the receiver is usually not fixed, but moving. To correct these problems, we usually use digital adaptive filters with coefficients that can be adjusted as more information is acquired about the unknown statistics of the channel and as the channel varies in time. This is necessary to perform the echo or noise cancellation, for example.

Multirate digital signal processing is as prevalent as adaptive filtering. It appears wherever there is a need for varying the sampling rate of a signal, by decimation or interpolation, from point to point in a system or between systems. Such alteration of the sampling rate often makes the processing of the signal more efficient since the the sampling rates at different points can be made small as possible. An example of the use of multirate DSP is in subband coding of speech and image signals. Since the human auditory and visual systems are more sensitive to certain frequencies than others, coding the different frequency bands non-uniformly can greatly reduce the number of bits for transmission or storage. Furthermore, the subbands can be downsampled to reduce the rate of the system before the data are transmitted or stored and subsequently reconstructed by interpolation.

In summary, changing signal environments call for adaptive filtering while physical or hardware limitations require multirate digital signal processing. These problems are becoming more pronounced as telecommunications become increasingly sophisticated. Although we have only mentioned applications of adaptive filtering and multirate DSP in the context of communications, these two fields are of numerous use in other interesting areas as well, ranging from biomedical applications to seismology.

1.2 Motivation and Scope

In the previous section, we show some common examples where adaptive filtering and multirate digital signal processing techniques have been utilized. In this thesis we would like to combine these two fields in the problem of adaptive equalization of a narrow band signal, and study a new adaptive filter structure that we propose. In particular, we would like to explore the feasibility of performing adaptive equalization in multiple stages on a narrow band signal or a signal that has been processed by a multirate system. We call this filter structre the *multistage, multirate adaptive filter*. Before we describe the problem, let's look at two practical applications where we find adaptive filtering and multirate DSP appearing together.

Research in adaptive filtering and multirate digital signal processing spans several decades, but only within the last five years or so have these two fields merged in many useful applications. A prime example of this union is adaptive filtering in subbands, as illustrated in Figure 1-1 [37]. In this scenario, a bank of parallel bandpass filters, called the *analysis filter bank*, splits the distorted signal into adjacent frequency bands. Each subband is then downsampled and corrected individually by an adaptive filter. To reconstruct the corrected signal, we upsample the subbands by the same factor used in downsampling, filter them with a *synthesis filter bank* consisting of parallel bandpass filters, and recombine the results.

A practical application of this scheme, which has been under continual research in recent years, is acoustic echo cancellation in teleconferencing, where the goal of the adaptive filter is to identify the unknown acoustics of the conference room or track its time-varying properties that result from the movement of the people, for example. [16], [17], [18], [24], [42], [37]. In practice, it is most desirable to perform the echo cancellation in real time, but correcting the full spectrum typically would require a single adaptive filter with a very high order, perhaps in the thousands, which requires a tremendous amount of computation and a long time to converge. By doing echo cancellation in subbands, we allow multiple smaller adaptive filters to do the job of one single filter. Thus, unlike the full-band case, subband filtering can greatly reduce computational complexity because the signal is downsampled, and it usually improves the convergence speed of the adaptive filter as well, because each subband is filtered separately [18].

Another example of the merging of those two areas is channel equalization for a

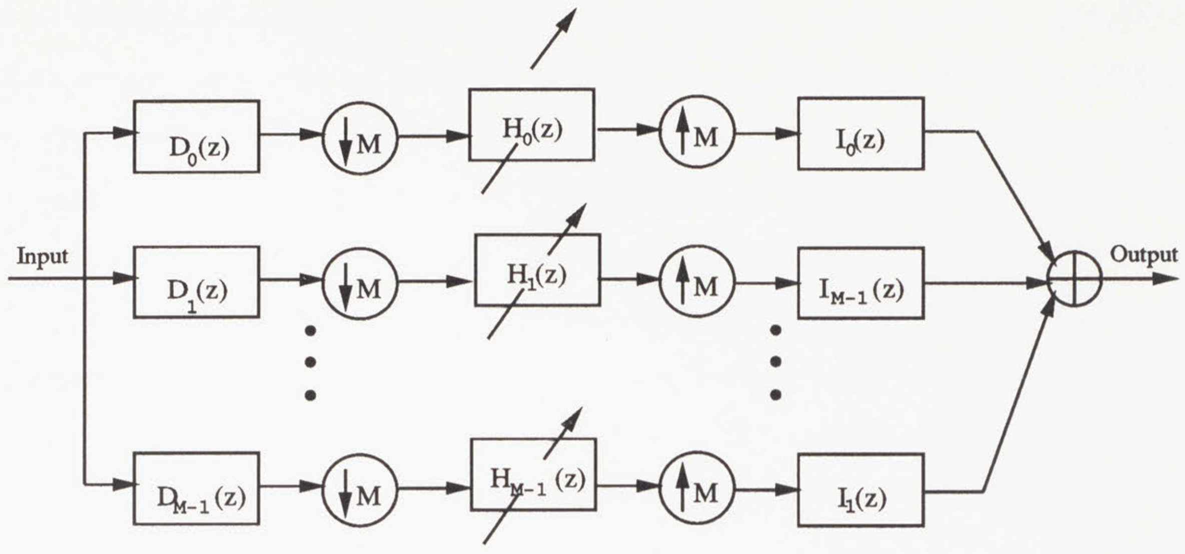


Figure 1-1: Adaptive filtering in subbands.

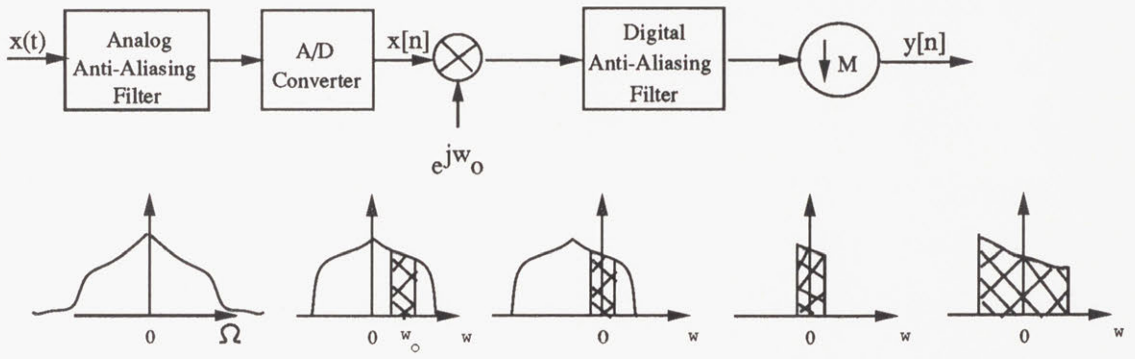


Figure 1-2: Block diagram of a channel equalization in a multirate environment.

narrow band signal as shown in Figure 1-2. The channel in this example may be an analog anti-aliasing filter that is placed before an A/D converter. This analog filter has ripple in the passband that is bigger than desired, and has nonlinear phase, both of which will produce distortion. We want to correct this distortion but only over a selected narrow band. Since the selected distortion is confined to a narrow band of the spectrum, we can allow it to run at a lower sampling rate by filtering and decimating it. If the selected narrow band's center frequency, ω_o, is not at zero, we must demodulate it to DC first before decimating it, as shown in the figure.

Equalization will be done adaptively because the analog filter is not known initially or it may be time-varying. Furthermore, the subband we want to correct will also differ according to selection. Not only does the analog anti-aliasing filter deviate from specifications from time to time, but the center frequency of the band to be filtered also varies from case to case. The decimation factor also changes according to the width of the selected band. Occurence of any of these factors will require the equalization filter to update its coefficients again.

We use this channel equalization system to study the multistage, multirate filter. To simplify the problem, we will only deal with the correction of the narrow band distortion centered at DC, which can be thought of as a subset of the problem of adaptive filtering in subbands described earlier. The narrow band corresponds to the subband centered at zero frequency, and we will deal only with the analysis and adaptive filtering part, and not the synthesis or reconstruction part.

Based on the fact that multistage decimation leads to significant computational savings, as we shall see in Section 2.2, here we will also downsample the signal in multiple stages and equalize it with multiple adaptive filters, each inserted in between successive decimation stages. The length of the single-stage equalizer must be long in order to justify its decomposition in stages, which is the case when we have to correct very bad channel distortions or similarly to finely correct moderate ones. By performing the correction in multiple stages, each of the equalizer can be smaller and less stringent, whereby the filter at each successive stage corrects whatever distortion that did not get corrected in the previous stages.

As in the multistage decimation scenario, multistage equalization perhaps may perhaps simplify the design of the adaptive filter, lower its total length, and reduce the number of multiplications per second for a given equalization error. In practice, however, multistage decimation and multistage equalization will not go hand-in-hand in all instances. In the extreme case, for example, if the decimation factor is very high, meaning that we are looking at a very narrow band, the magnitude response of the distorted signal will approximate a straight line. The equalizer then needs to be only one tap long to shift this distortion to the right level of correction. Thus, while the multistage structure may be helpful for decimation, it is not necessary for equalization if the required equalizers are short. In this study, we consider a situation, which exists in some applications, where a signal after being decimated with a reasonably high factor still has a fairly large distortion. This setting allows for a sensible study of multistage equalization in a multistage, multirate environment.

To address the above questions, we formulate models of the multistage, multirate filter and the single-stage structure and study them for two filter-design techniques: 1) the classic least-squares method and 2) the frequency-domain block least mean square (FBLMS) algorithm. While the first method is in the time-domain and is run nonadaptively, the second algorithm is adaptive and in the frequency domain. The background for these algorithms is discussed in Section 2.1. Due to the some inherent difficulties in rigorous analysis of the multistage, multirate filter as discussed in Section 3.4, we will approach the study of the proposed structure by using solftware simulations with a representative example. The models that we develop for the multistage and single-stage filters are discussed in Section 3.5, and the results of the simulations for the least squares method and the FBLMS algorithm are included in Sections 3.6 and 3.7, respectively. What we present in this thesis is not a comprehensive study of the proposed filter structure, but rather a possible approach to explore the multistage, multirate adaptive filter.

Chapter 2

Background

2.1 Adaptive Filtering

2.1.1 Introduction

The basis of adaptive filters arises from the non-adaptive data-fitting problem as shown in Figure 2-1. The goal here is to design a filter $h[n]$ that processes an input $x[n]$ and produces some output $y[n]$ that must match as closely as possible to a given desired output $d[n]$. The difference between the filter output and the desired response is the error, $e[n] = d[n] - y[n]$. The most commonly-used error criterion involves a quadratic function of the error. *deterministic* or *stochastic*, and the optimum filter for that error criterion is the deterministic *least-squares* or the probabilistic *minimum-square error* filter, respectively. To obtain the "optimum" solution for this nonadaptive filter, we must have full knowledge of the deterministic input or the statistical properties of the stochastic input. The solution involves solving a system of equation by some matrix inversion.

Figure 2-2 depicts the adaptive filtering problem. The difference between this adaptive structure and the nonadaptive one is that the error here is fed back into an algorithm that minimizes some function of the error to obtain the optimum filter coefficients. There are generally three situations where finding the filter that best performs the data fitting must be done adaptively. The first is when, despite a known input signal or one with specificied statistics, we do not want to perform the matrix inversion because it may lead to a computational overload and numerical instability. Therefore, we resort to an iterative or adaptive procedure to solve for the optimum solution. The second reason for finding the filter adaptively arises when the statistical properties of the stochastic input are unknown. We must then estimate them adaptively, based on the incoming input samples, in order to find the optimum solution. In both cases, the filter's coefficients will vary, but only during the time of adaptation, and they remain fixed after the adaptation is over. In the third situation, the statistics of the input vary with time so that even after the optimum solution is found, the filter must adapt again whenever the signal environment changes.

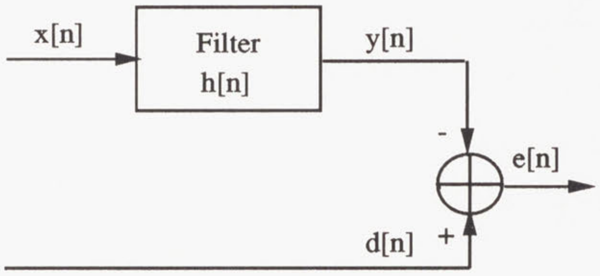


Figure 2-1: Structure of a non-adaptive FIR filter.

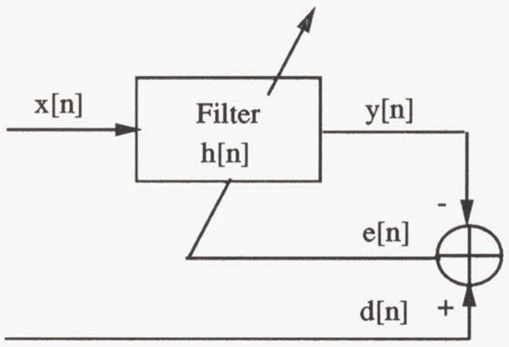


Figure 2-2: Structure of an adaptive FIR filter.

Two common system configurations that include adaptive filters: (1) channel equalization, and (2) system identification. In channel equalization, we have a channel that causes some undesirable distortion of the signal that passes through it. Examples of such channels are cables or the atmosphere, where a signal that travels through it is being degraded. The objective here is to design an adaptive filter that estimates the unknown statistics of the channel and undoes–or equalizes–the distortion caused by the channel. Such a filter is called an equalizer. The situation is illustrated in Figure 2-3. In system identification, as depicted in Figure 2-4, the task of the adaptive filter is to identify the system function of some unknown system. An example of this is in echo cancellation in teleconferencing, where the adaptive filter tries to find the unknown acoustics of the room and/or track its changing acoustic properties over time.

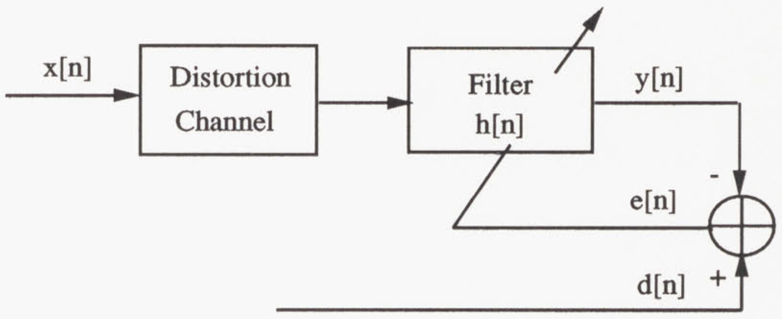


Figure 2-3: Channel equalization with an adaptive filtering.

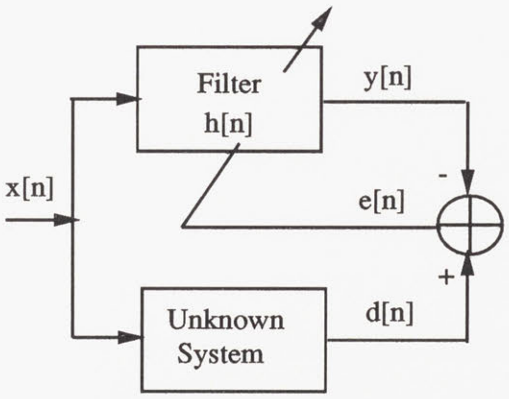


Figure 2-4: System identification with adaptive filtering.

Like non-adaptive filters, adaptive filters also consist of two types: IIR (infinite impulse response) and FIR (finite impulse response). We choose to study the FIR filter in this thesis because it has several attractive properties that the IIR filters lack, and it is more widely used in practice. The reasons will be made clear once we have an understanding of adaptive filters.

The FIR adaptive filter consists of tapped delays and a set of adjustable coefficients, $\{h_n\}$, that are updated by a controlling algorithm at each iteration or as the filter input, $x[n]$, or its statistics vary over time. See Figure 2-5. The output of the adaptive filter, $y[n]$, is compared to a given desired output, $d[n]$. The difference between these two signals is the error $e[n] = d[n] - y[n]$. The goal is to minimize some function $f(e)$ of the error:

$$\min\{f(e)\} \tag{2.1}$$

to obtain the optimum set of filter coefficients, $h^*[n]$.

The following sections will discuss different ways to obtain the solution to the adaptive filter. The optimum solution for the general stochastic signal environment was first developed by Norbert Wiener in the 1940's, and it is commonly known as the Wiener filter. However, its origins extend back several centuries to the well-known deterministic data-fitting by least-squares, first explored by Gauss and Legendre. Therefore, in order to have a proper understanding of the subject matter, we will start from the foundation by reviewing the least-squares filter from both the calculus and linear algebra points of view. Following this, we will extend the deterministic solution to the probabilistic one to obtain the Wiener filter. In both the deterministic and stochastic cases, the optimum solution can be obtained by solving a system of equations. We next delve into the problem of finding the Wiener solution by an adaptive or recursive means. This will lead us to focus on the least-mean squares (LMS) algorithm, which is one of the simplest and most widely used variations of the gradient descent method for solving the Wiener solution recursively. Although the LMS algorithm is not used explicitly in this thesis, it is nevertheles the basis of many frequency-domain adaptive algorithms. The frequency-domain block LMS filter is the specific frequency-domain algorithm used in this thesis and will be discussed in the last section on adaptive filtering.

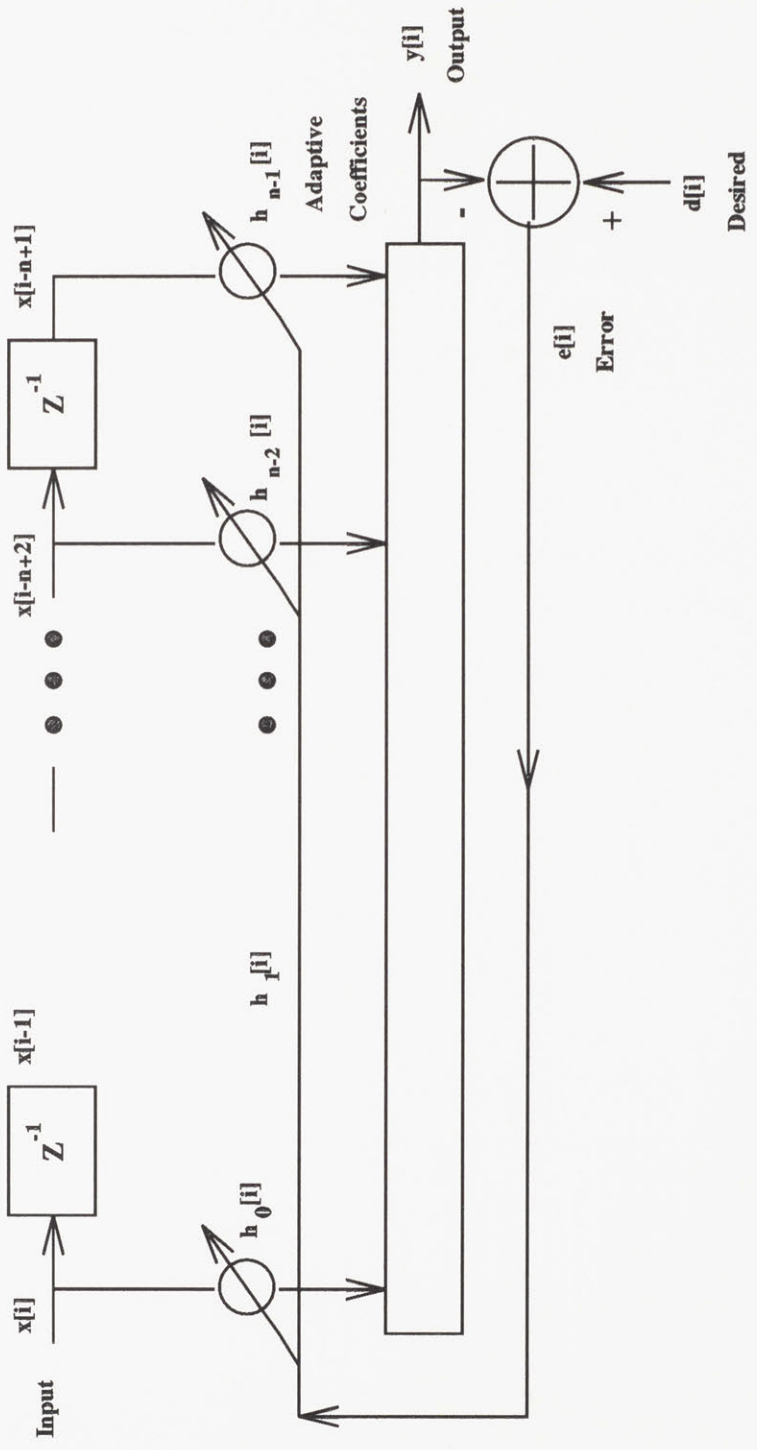


Figure 2-5: Structure of the adaptive FIR filter.

2.1.2 The Least Squares Filter

Given the filter structure shown in Figure 2-5, we want to minimize some function of the error $e[n] = d[n] - y[n]$ to obtain an optimum set of filter weights $h[n]$. Let's examine the simplest case where the input to the filter is *deterministic*, in which case the goal is to find a filter $h[n]$ such that its output, $y[n] = x[n] * h[n]$, best matches the desired response $d[n]$. The classic *method of least squares* is commonly employed to solve this problem, where we want to minimize the sum of error squares:

$$\mathcal{E} = \sum_{n=i_1}^{i_2} e^2[n] \tag{2.2}$$

within some time window limited by i_1 and i_2.

First, we want to write a system of equations that describes just the linear combiner part of the filter structure. We can see that for an N-tap filter, the output $y[n]$ at time n is the convolution of the N filter coefficients $h[n]$ with the last N input samples of $x[n]$:

$$y[n] = x[n] * h[n] \tag{2.3}$$

or

$$y[n] = \sum_{i=0}^{N-1} h[i]x[n-i]. \tag{2.4}$$

Expanding the convolution equation out for L outputs (where $L > N$) and assuming that $x[n] = 0$ for $n < 0$, we obtain a system of equations in (2.5). Note that the time index n for each time variable is subscripted here for compactness in notation.

$$\begin{array}{lllllllll}
y_0 & = & x_0 h_0 & + & 0 & + & \dots & + & 0 \\
y_1 & = & x_1 h_0 & + & x_0 h_1 & + & \dots & + & 0 \\
y_2 & = & x_2 h_0 & + & x_1 h_1 & + & \dots & + & 0 \\
\vdots & & & & \vdots & & & & \\
y_{N-1} & = & x_{N-1} h_0 & + & x_{N-2} h_1 & + & \dots & + & x_0 h_{N-1} \\
\vdots & & & & \vdots & & & & \\
y_{L-1} & = & x_{L-1} h_0 & + & x_{L-2} h_1 & + & \dots & + & x_{L-N} h_{N-1} \\
\vdots & & & & \vdots & & & &
\end{array} \tag{2.5}$$

Let's assume that our input $x[n]$ is L long and our time window of interest in (2.2) has index limits $i_1 = 0$ to $i_2 = L - 1$. We next cast the system of equations in (2.5) into vector notation and obtain:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{N-1} \\ \vdots \\ y_{L-1} \end{bmatrix} = \begin{bmatrix} x_0 & 0 & 0 & \dots & 0 \\ x_1 & x_0 & 0 & \dots & 0 \\ x_2 & x_1 & x_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N-1} & x_{N-2} & x_{N-3} & \dots & x_0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{L-1} & x_{L-2} & x_{L-3} & \dots & x_{L-N} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-1} \end{bmatrix} \tag{2.6}$$

or

$$\mathbf{y} = X\mathbf{h} \tag{2.7}$$

where $\mathbf{y}$ and $\mathbf{h}$ are vectors and X a matrix as shown in (2.6) above.

We want to find a filter $\mathbf{h}$ that would match the filter output $\mathbf{y} = X\mathbf{h}$ with a desired response vector $\mathbf{d}$ of the same length:

$$X\mathbf{h} = \mathbf{d}\ . \tag{2.8}$$

However, from (2.6) we can see that there are more equations than unknowns

$(L > N)$ so that the system of equations (2.8) is typically inconsistent and has no exact solutions. Instead, we want to find an $\mathbf{h}$ that would minimize a measure of the error vector, $\mathbf{e} = \mathbf{d} - X\mathbf{h}$. A convenient way to define this measure is the *sum of squares*:

$$\mathcal{E} = \sum_{n=0}^{L-1} e^2[n], \tag{2.9}$$

which is the error criterion of the least-squares method.

Substituting $d[n] - y[n]$ for $e[n]$ in (2.9), we obtain:

$$\mathcal{E} = \sum_{n=0}^{L-1} (d[n] - y[n])^2, \tag{2.10}$$

which becomes

$$\mathcal{E} = \sum_{n=0}^{L-1} (d[n] - \sum_{i=0}^{N-1} h[i]x[n-i])^2, \tag{2.11}$$

by replacing $y[n]$ with the convolution sum in (2.4).

Since the error function is quadratic and always positive, it is guaranteed to have a unique global minimum. The set of filter coeffients, $\{h_0, h_1, ..., h_{N-1}\}$ that would give this unique minimum can be found by solving the following set of equations:

$$\frac{\partial \mathcal{E}}{\partial h_i} = 0, \quad i = 0, 1, ..., N-1\ . \tag{2.12}$$

We can also look at the least-squares filter from the linear algebra point of view [36]. Since $X\mathbf{h} = \mathbf{d}$ is inconsistent, the desired vector $\mathbf{d}$ cannot be expressed as a linear combination of the columns of the matrix X; it lies outside the column space, or the range, of X. The error here is the distance of $\mathbf{d}$ from $\mathbf{y}$, which lies in the range of X. Minimizing the error corresponds to finding a solution $\mathbf{h}$ in the range (or the column space) of X such that $\mathbf{y} = X\mathbf{h}$ has the closest distance to $\mathbf{d}$. This is the same as finding the projection of the desired vector $\mathbf{d}$ onto the column space of the matrix X, and the perpendicular line from this projection corresponds to the minimum error. Figure 2-6 illustrates this with a simple example where the input length is $L = 3$ and the filter length is $N = 2$.

Thus, in the least-squares method, minimizing the sum of error squares can be

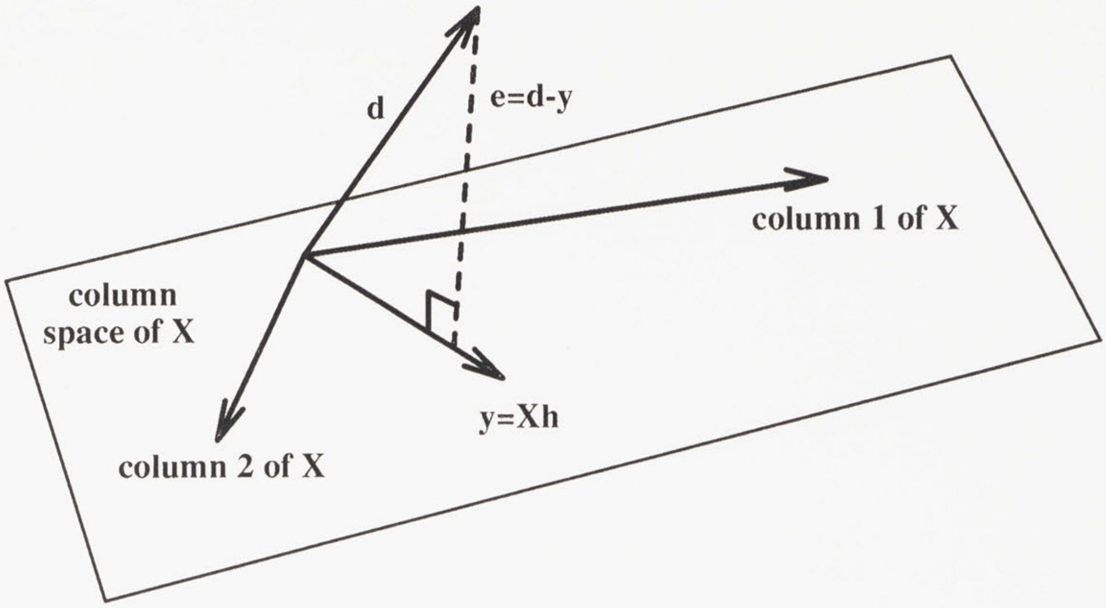


Figure 2-6: Least squares as an orthogonal projection.

equivalently viewed as minimizing the *length* of the error vector $\mathbf{e}$:

$$\min(\mathcal{E}) = \min(\sum_{n=0}^{L-1} e^2[n]) = \min(\mathbf{e}^T\mathbf{e}) \ . \tag{2.13}$$

Using linear algebra, we can solve for the system of equations $X\mathbf{h} = \mathbf{d} - \mathbf{e}$ by multiplying both sides of this equation by the transpose of X, denoted by X^T, and using the orthogonality to recognize that $X^T\mathbf{e} = 0$, so that

$$X^T X\mathbf{h} = X^T\mathbf{d} \ . \tag{2.14}$$

The optimum filter coefficients $\mathbf{h}^*$ can now be found through a matrix inversion

$$\mathbf{h} = (X^T X)^{-1} X^T\mathbf{d} \ . \tag{2.15}$$

If the columns of X are linearly independent, then the resulting matrix $X^T X$ is invertible. Equations in (2.15) are sometimes called the *deterministic normal equations* since the principle of orthogonality is exploited here.

2.1.3 Computational Complexity

The least squares filter requires solving the following matrix equations for the filter weights vector $\mathbf{h}$:

$$\mathbf{h} = (X^T X)^{-1}(X^T \mathbf{d}) \tag{2.16}$$

or equivalently

$$\mathbf{h} = A^{-1}\mathbf{b} \tag{2.17}$$

where $A = X^T X$ and $\mathbf{b} = X^T \mathbf{d}$.

Let the following denotes the lengths of the vectors of interest here:

- N_h : length of filter vector $\mathbf{h}$
- N_x : length of input vector $\mathbf{x}$
- N_d : length of desired response vector $\mathbf{d}$

The matrix X is of size $N_h \times N_x$ which makes A of size $N_h \times N_h$. Each entry in A takes N_x multiplications so forming the matrix A takes a total of

$$N_x N_h^2 \tag{2.18}$$

multiplications.

To invert the matrix A, assuming using Gaussian elimination, would take approximately

$$N_h^3 \tag{2.19}$$

multiplications.

The vector $\mathbf{b}$ is $N_h \times 1$ long and each entry takes N_x multiplications which gives a total number of multiplications for $\mathbf{b}$ to be

$$N_x N_h \ . \tag{2.20}$$

Performing the products $A^{-1}\mathbf{b}$ takes

$$N_h^2 \tag{2.21}$$

mutiplications.

Thus, the total number of multiplications required in the time-domain least squares solution is the sum of Equations above is

$$N_h^3 + N_h^2(N_x + 1) + N_h N_x \tag{2.22}$$

From (3.25) above, we can see the length of the filter figures highly in the computational complexity equation.

2.1.4 The Minimum Mean Square Error Method

The least squares approach to finding the filter coefficients, as discussed previously, is effective only when the filter input is deterministic. If the input to the filter is random, we must find another solution that will take into account the statistical properties of the filter input. This method, also based on the deterministic method of least squares, is known as the *minimum mean squared error*, or *MMSE*, method and the resulting solution is often called the *Wiener filter*[1],[21], [44].

In the MMSE scheme, we try to minimize the *ensemble average*, or the *expected value*, of the squared error:

$$min\{E(e^2[n])\} \tag{2.23}$$

Let us formulate the equations for the Wiener solution. First, assume that the filter input $x[n]$ and a desired response $d[n]$ are real, discrete-time stochastic processes that are *wide-sense stationary.* The N-element vector of filter weights we want to find is

$$\mathbf{h} = [h_0 \; h_1 \; ... \; h_{N-1}]^T \; . \tag{2.24}$$

At each time n, let us also form a vector $\mathbf{x}_n$ of length n as follows:

$$\mathbf{x}_n = [x_n \; x_{n-1} \; \dots \; x_{n-N+1}]^T, \tag{2.25}$$

where each sample $x[n]$ is subscripted here for compactness in notation.

Furthermore, the statistics of the random input and the desired response are assumed to be known. In particular, we are interested in the *autocorrelation matrix* R of the input $\mathbf{x}_n$, defined as

$$R = E(\mathbf{x}_n \mathbf{x}_n^T)^1 \tag{2.26}$$

and the *cross correlation vector* of the desired signal $d[n]$ and input vector $\mathbf{x}_n$:

$$\mathbf{p} = E(d[n]\mathbf{x}_n) \tag{2.27}$$

The elements of the autocorrelation matrix R and the cross-correlation vector $\mathbf{p}$ above are defined by these functions:

$$\text{Autocorrelation}: \quad r_{xx}(k) = E(x[n]x[n+k]) \tag{2.28}$$

$$\text{Cross-correlation}: \quad r_{dx}(k) = E(d[n]x[n+k]) \; . \tag{2.29}$$

Note that these correlation functions apply only to wide-sense stationary processes, where the correlation functions depend only on the *difference*, or *lag*, between the sample times n and $n+k$, which is k.

For $\mathbf{x}$ a wide-sense stationary process, R becomes

$$R = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \dots & r_{xx}(N-1) \\ r_{xx}(1) & r_{xx}(0) & \dots & r_{xx}(N-2) \\ \vdots & \vdots & \vdots & \vdots \\ r_{xx}(N-1) & r_{xx}(N-2) & \dots & r_{xx}(0) \end{bmatrix}, \tag{2.30}$$

where each element, $r_{xx}(k)$, is as given in (2.28). The special properties of the matrix

[1]Note: For $\mathbf{x}$ complex, the Hermitian transpose $\mathbf{x}^H$ replaces the real transpose in (2.26).

R are that it is square, symmetric, and Toeplitz (the terms on each diagonal are equal), which makes it easy to invert if the matrix is not singular.

For $\mathbf{x}$ and $\mathbf{d}$ stationary, the autocorrelation vector $\mathbf{p}$ becomes

$$\mathbf{p} = \begin{bmatrix} r_{dx}(0) \\ r_{dx}(1) \\ \vdots \\ r_{dx}(N-1) \end{bmatrix} \tag{2.31}$$

where $r_{dx}(k)$ is as given in (2.29).

Similar to the least-squares filter, here we also try to match the filter output $y[n]$ with the desired response $d[n]$, but these signals now have stochastic models. Therefore, we want to minimize, not the sum of the squared errors as in the deterministic least-squares filter, but the mean-square error (MSE) given by

$$\mathrm{MSE} = E(e^2[n]) = E((d[n] - y[n])^2) \ . \tag{2.32}$$

The expanded form of the MSE is

$$\mathrm{MSE} = E(d^2[n] - 2\mathbf{h}^T d[n]\mathbf{x}_n + \mathbf{h}^T \mathbf{x}_n \mathbf{x}_n^T \mathbf{h}) \tag{2.33}$$

which becomes

$$\mathrm{MSE} = E(d^2[n]) - 2\mathbf{h}^T E(d[n]\mathbf{x}_n) + \mathbf{h}^T E(\mathbf{x}_n \mathbf{x}_n^T)\mathbf{h} \ . \tag{2.34}$$

Substituting R and $\mathbf{p}$ from (2.26) and (2.27) above, we obtain the following form for the MSE:

$$\mathrm{MSE} = \sigma_d^2 + 2\mathbf{h}^T \mathbf{p} + \mathbf{h}^T R \mathbf{h} \tag{2.35}$$

where $\sigma_d^2 = E(d^2[n])$.

Analogous to the least-squares method, the goal here is to find a vector of filter weights, $\mathbf{h}^*$, that is optimum but in a probabilistic mean-square sense. This is achieved by minimizing the MSE in (2.35) with respect to $\mathbf{h}$ by solving

$$\frac{\partial(MSE)}{\partial\mathbf{h}} = 2R\mathbf{h} - 2\mathbf{p} = 0 \tag{2.36}$$

for $\mathbf{h}^*$.

The optimum filter weights vector, $\mathbf{h}^*$, that minimizes the error function in the mean square sense simply involves the inverted autocorrelation matrix of the input, R^{-1}, and the cross correlation vector of the input and the desired output, $\mathbf{p}$:

$$\mathbf{h}^* = R^{-1}\mathbf{p} \ . \tag{2.37}$$

This is known as the *Wiener solution.*

Compared to Equation (2.15) from the deterministic least squares approach, the minimum mean squared solution also has a similar structure. The autocorrelation matrix, R, is the probabilistic counterpart of the deterministic matrix X^TX; likewise for $\mathbf{p}$ and $X^T\mathbf{d}$.

The minimum mean squared error function has a geometrical interpretation, which is called the *error performance surface.* For an N-weight FIR filter operating in a stationary environment, the error performance surface is an upward concave paraboloid in an $N+1$ dimensional space with a unique global minimum. The expression in (2.36) is the *gradient* ∇ of the error surface. The goal of the minimum mean-square error method is to find the bottom of this "bowl", which corresponds to a set of filter weights that is optimum in the mean square sense.

2.1.5 Least Mean Square Algorithm

The least mean square, or LMS, algorithm belongs to a class of algorithms that computes the Wiener filter iteratively using the *method of steepest descent* [1],[21], [44]. The method of steepest descent searches for the optimum filter weights by iteratively calculating the gradient of the error performance surface at regular time intervals and incrementally sliding down toward the bottom of the bowl. The filter coefficients are first initialized, usually to zero, and they are updated at each time increment as follows:

$$\mathbf{h}_{n+1} = \mathbf{h}_n - \mu\nabla \tag{2.38}$$

where ∇ is the gradient of the error performance surface calculated at the previous time n and μ is the *step size* of the adaptation. The subscripts in $\mathbf{h}$ denote the filter weights at update times n and $n+1$. Note that the algorithm is moving in the negative direction of the gradient in order to get to the minimum of the upward concave bowl.

To calculate the gradient ∇ in the steepest descent method, we differentiate the mean squared error MSE of the Wiener filter in (2.35) with respect to the filter weight $\mathbf{h}$ to obtain

$$\nabla = 2R\mathbf{h}_n - 2\mathbf{p} \tag{2.39}$$

Substituting this in (2.38) we get

$$\mathbf{h}_{n+1} = \mathbf{h}_n + 2\mu(\mathbf{p} - R\mathbf{h}_n)\ . \tag{2.40}$$

In practice, however, the true gradient in (2.39) is hard to find since we must have full knowledge of the second-order statistics of the input $x[n]$ and the desired output $d[n]$ in order to calculate the autocorrelation matrix R and the cross-correlation vector $\mathbf{p}$. A way to avoid this difficulty is to simply estimate R and $\mathbf{p}$ from the instantaneous sample values of the input vector $\mathbf{x}_n$ and the desired response $d[n]$ at time n. The estimated autocorrelation matrix, $\hat{R}_n$, is

$$\hat{R}_n = \mathbf{x}_n\mathbf{x}_n^T \tag{2.41}$$

and the estimated cross-correlation vector, $\hat{\mathbf{p}_n}$, is

$$\hat{\mathbf{p}}_n = d[n]\mathbf{x}_n\ . \tag{2.42}$$

Substituting $\hat{R}_n$ and $\hat{\mathbf{p}}_n$ into (2.40), we obtain

$$\mathbf{h}_{n+1} = \mathbf{h}_n + 2\mu(\hat{\mathbf{p}}_n - \hat{R}_n\mathbf{h}_n) \tag{2.43}$$

which simplifies to

$$\mathbf{h}_{n+1} = \mathbf{h}_n + 2\mu\mathbf{x}_n\hat{e}[n] \tag{2.44}$$

where $\hat{e}[n] = d[n] - \mathbf{x}_n^T\mathbf{h}_n$. Equation (2.44) is known as the *Widrow-Hoff least mean square*, or *LMS*, filter update equation.

Note that while the gradient descent algorithm is deterministic, the LMS algorithm is stochastic since the direction it takes at each iteration is based on an instantaneous estimate which is random.

The convergence of the LMS algorithm is governed by the eigenvalue spread of the autocorrelation matrix R of the input. Convergence is obtained for any step size μ satisfying

$$0 < \mu < \frac{1}{\sum_{i=1}^{N}\lambda_i} \tag{2.45}$$

where the λ_i's are the eigenvalues of the autocorrelation matrix R. Equation (2.45) can also be equivalently expressed as

$$0 < \mu < \frac{1}{\textit{total input power}} \tag{2.46}$$

since the total power of the input signal equals the sum of the eigenvalues of its autocorrelation matrix. [2] Choosing μ outside the specified ranges in (2.45) would make the algorithm diverge. Furthermore, the step size μ, also determines the convergence

[2]Recall that the trace of a square matrix A is the sum of its diagonal elements. Thus, the trace, denoted as tr[$\cdot$], of the $N \times N$ autocorrelation matrix R in (2.30) is

$$\begin{aligned} \mathrm{tr}[R] &= \sum_{i=1}^{N} r_{xx}(0) && (2.47)\\ &= N r_{xx}(0) && (2.48) \end{aligned}$$

where N is the number of filter taps, and $r_{xx}(0)$ is the autocorrelation function of the tap inputs with zero lag. Furthermore, the total power of a stationary input equals $Nr_{xx}(0)$. It is also true that the trace of a autocorrelation matrix is the sum of its eigenvalues. Hence, the trace of R can rate of the adaptation as well as the minimum mean-square error achievable by the algorithm. A large μ would lead to fast convergence but to a large mean-square error. This is because we are taking big steps down the error performance surface and are not likely to get close to the actual minimum point. At the other extreme, if μ is really small then we would be able to "inch" closer to the actual minimum point, but it would take a long time for the algorithm to converge. Which value of μ to choose depends on the application at hand.

In practice, using an iterative procedure to search for the Wiener solution does not usually yield the exact optimum solution. One reason is that since the step size μ used in the iteration is finite, there is always some residual error even when the step size is very small. Another reason is that there are always round-off errors in representing the numbers in finite bit precision. A measure of how close the iterative search for the solution is to the actual minimum square error is given by the *misadjustment error* defined as

$$\mathcal{M} = \frac{\sum_{i=1}^{N} \lambda_i}{2 - \mu \sum_{i=1}^{N} \lambda_i}. \tag{2.51}$$

where $\mathcal{M}$ expresses the ratio of the average excess mean-square error obtained by the steepest descent algorithm to the actual minimum mean-square error of the Wiener filter. A smaller $\mathcal{M}$ implies the adaptive solution is closer to the optimum solution.

In summary, the least mean square (LMS) algorithm is one of the most widely used adaptive algorithms because its update equation is simple to implement and compute. Furthermore, it has been proven to perform well in both stationary and non-stationary signal environments. Refer to [1],[21], or [44] for more extensive coverage of the LMS algorithm and its properties.

---

be equivalently expressed as

$$\mathrm{tr}[R] = \sum_{i=1}^{N} \lambda_i \tag{2.49}$$

where λ_i's are the eigenvalues of R. Therefore, (2.49) and (2.48) are identical. Consequently,

$$\sum_{i=1}^{N} \lambda_i = N r_{xx}(0) = \textit{total input power}\ . \tag{2.50}$$

2.1.6 Frequency-Domain Adaptation

Adaptive filters can be implemented not only in the time domain but also in the frequency domain. There are generally two broad classes of frequency-domain adaptation. The first class uses FFT's to perform the block convolution and correlation needed in the time-domain adaptive algorithm. The second class involves splitting the input signal up into subbands using a filter bank and adapting on each subband individually. We will use an algorithm from the first class to study the multistage, multirate adaptive filter.

This thesis studies the frequency-domain block least mean square (FBLMS) algorithm [10],[15],[32], which is an efficient implementation of its time-domain counterpart, the time-domain block LMS (TBLMS) [9]. The difference between the block LMS and the conventional, nonblock LMS is that instead of updating the adaptive filter coefficients at every pair of input vector and desired response, we update them once per data block. The estimated gradient is computed from a block of data, and the same gradient is used to update all the filter coefficients once per block of data until the filter converges. The gradient computed this way is actually a better estimate of the true ensemble gradient of the optimum Wiener filter because it takes an average of a block of data instead of just one sampling instant as in the nonblock LMS case. Consequently, block adaptation allows for a smoother convergence since the estimated gradient is less "noisy".

Another appealing attribute of the block LMS filter is that it needs fewer updates because the filter coefficients are adjusted only once every block of data, which, consequently, reduces the total number of computations in the adaptation. Finally, block implementation of the filter can take advantage of the computational efficiency of parallel procesors or serial processors plus the fast Fourier transform (FFT) [9]. The latter form is used in this thesis and will be discussed in the next section.

Algorithm for Time-Domain Block LMS

Before we start, let us establish some notations. In the time-domain block LMS method, an input sample is accumulated at each discrete time n to form a block of length N. The adaptive filter $\mathbf{h}$, also of length N, is updated at every nN input samples, which we shall call the *kth block iteration* where $k = nN$.

Let $\mathbf{h}_k$ be a vector of N filter coefficients at the kth block iteration. It is represented as

$$\mathbf{h}_k = [h_0\ h_1\ \ldots\ h_{N-1}]^T\ . \tag{2.52}$$

where h_i are the coefficients of the filter.

Step 1: First, initialize the filter coefficients to zero:

$$\mathbf{h_0} = [0\ \ldots\ 0]^T \tag{2.53}$$

Step 2: At each discrete time n, form an input vector $\mathbf{x}_n$ of length N:

$$\mathbf{x}_n = [x_n\ x_{n-1}\ \ldots\ x_{n-N+1}]^T \tag{2.54}$$

Step 3: At each kth block iteration, form an $N \times N$ matrix $\mathcal{X}_k$ from N input vectors $\mathbf{x}_n$ in Step 2 above:

$$\mathcal{X}_k = \begin{bmatrix} \mathbf{x}_{(k-1)N+1} \\ \mathbf{x}_{(k-1)N+2} \\ \vdots \\ \mathbf{x}_{(k-1)N} \end{bmatrix} \tag{2.55}$$

This is a block of length N from the total input matrix X in (2.6) in Section 2.1.2.

Step 4: The output block of the filter, $\mathbf{y}_k$ at the kth block iteration is the convolution, or vector multiplication, of the input matrix $\mathcal{X}_k$ in (2.55) with the current filter vector $\mathbf{h}_n$ as follows:

$$\mathbf{y}_k \ = \ \mathcal{X}_k\mathbf{h}_k \tag{2.56}$$

Step 5: The error vector at the kth block iteration is

$$\mathbf{e}_k = [e_{(k-1)N+1}\ e_{(k-1)N+2}\ \ldots\ e_{kN}]^T \tag{2.57}$$

which represents the difference between the kth filter output block $\mathbf{y}_k$ and the kth desired output block:

$$\mathbf{e}_k = \mathbf{y}_k - \mathbf{d}_k \; . \tag{2.58}$$

Step 6: The estimate block gradient at iteration k is

$$\hat{\nabla}_k = -\frac{2}{N}\mathcal{X}_k^T \mathbf{e}_k \; . \tag{2.59}$$

Step 7: The update equation at the $(k+1)$st block iteration is

$$\begin{aligned} \mathbf{h}_{k+1} &= \mathbf{h}_k - \mu_B \hat{\nabla}_k & (2.60) \\ &= \mathbf{h}_k + \frac{2\mu_B}{N}\mathcal{X}_k^T \mathbf{e}_k & (2.61) \end{aligned}$$

where μ_B is the step size of the block LMS. Steps 2 to 7 are repeated until the algorithm converges to give a filter $\mathbf{h}$ that approximates the Wiener filter. Athough we have shown the case where the data block has the same length as the adaptive filter, N, this does not have to be so in general. However, it has been shown that the optimum block length is equal to the filter length [9], [10].

Block Mean Square Error

For stationary inputs, both the block and nonblock LMS try to minimize essentially the same mean-square error (MSE), but the MSE estimated by the block LMS algorithm is closer to the true MSE of the Wiener filter because it is averaged over a block of data. The *block mean square error* (BMSE) is defined as

$$\text{BMSE} = \frac{1}{N}E(\mathbf{e}_k^T \mathbf{e}_k) = E(\frac{1}{N}\sum_{i=(k-1)N+1}^{kN} e_i^2) \tag{2.62}$$

Note that for block length $N = 1$, the BLMS algorithm reduces to the LMS algorithm.

Convergence Properties of the BLMS Algorithm

It has been proven that the BLMS algorithm converges, which involves showing that as the number of blocks in the adaptation approaches infinity, the expected value of the filter vector, $E(\mathbf{h}_k)$, approaches the optimum or Wiener solution [9]. The bounds on the step size, μ_B, for the BLMS are the same as the bounds on μ for the LMS algorithm in (2.45) [9]:

$$0 < \mu_B < \frac{1}{\sum_{i=1}^{N} \lambda_i} \; . \tag{2.63}$$

This similarity exists because the eigenvalue spread of the autocorrelation matrix R does not change when data averaging is used to estimate the gradient.

2.1.7 Frequency-Domain Block Least Mean Squares (FBLMS)

A further improvement in the time-domain block LMS can be achieved by implementing it in the frequency domain. Here we can take advantage of the computational efficiency of the fast Fourier transform (FFT) to perform the linear convolution and linear correlation needed in the time-domain filter. Of the two well-known data sectioning techniques (overlap-save and overlap-add) for block convolution [28], we will use the overlap-save method because it has been shown to take fewer computations than the overlap-add [10].

The linear convolution required in the time-domain block LMS filter is that between an N coefficients filter, $h[n]$, and a block of N samples from a relatively long filter input $x[n]$, to yield the filter output $y[n]$ as given by

$$y[n] = \sum_{i=0}^{N-1} x[i]h[n-i] \; . \tag{2.64}$$

The linear correlation is between a block of N samples from $x[n]$ and N samples of the error, $e[n]$, to produce the estimate gradient $\hat{\nabla}$:

$$\hat{\nabla} = \sum_{i=0}^{N-1} x[i]e[n+i]. \tag{2.65}$$

The convolution and the correlation operations are essentially the same except that for correlation neither sequence is reversed in the sum of products.

Using FFT's and the overlap-save block convolution method, we can efficiently implement the time-domain convolution and correlation above, whereby the convolution/correlation are now complex multiplications of the two respective transformed signals. To obtain an N point linear convolution/correlation, a $2N$-points FFT must be used since half of the data points will be aliased because multiplying the FFT's (or DFT's in general) of two signals correspond to their circular convolution in the time domain. Using the overlap-save method, the linear convolution part corresponds to the *last* N points of their circular convolution, whereas their linear correlation corresponds to the *first* N points.

The FBLMS Algorithm

Implementing the FBLMS using the overlap-save sectioning method and FFT's can easily be understood from Figure 2-7. The objective here is to adaptively find an N-point filter $\mathbf{H}$ in the frequency-domain:

$$\mathbf{H} = [H_0 H_1 \ \ldots \ H_{N-1}]^T \tag{2.66}$$

Since the overlap-save method calls for an overlap of a block of old data with the new one for each block convolution, we will choose the filter length N as the length of the overlap block. Therefore, we must use $2N$ long FFT's or IFFT's (inverse FFT) in Figure 2-7 to ensure that only N of the points will be aliased while the other half of the points will give a perfect convolution or correlation. The length of the adaptive filter $\mathbf{H}$ in (2.66) must be $2N$ long as well:

$$\mathbf{H} = [H_0 \ H_1 \ \ldots \ H_N \ \ldots \ H_{2N-1}]^T \tag{2.67}$$

At each iteration k, a new block of N input samples

$$\mathbf{x}_{k,new} = [x_0 \ x_1 \ \ldots \ x_{N-1}]^T \tag{2.68}$$

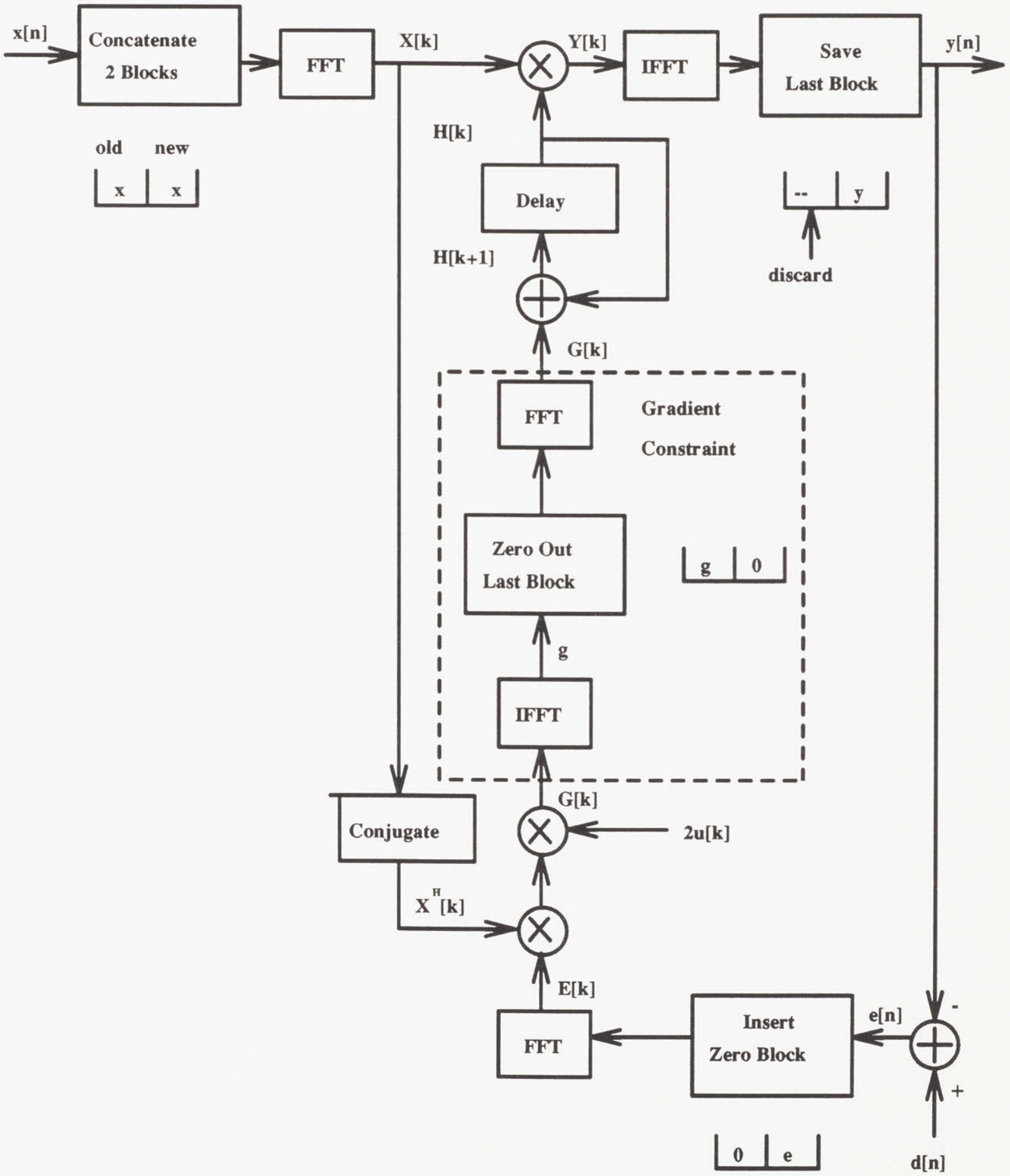


Figure 2-7: Structure of Overlap-Save Frequency-Domain Block LMS Filter.

is concatenated to an old block of N input samples

$$\mathbf{x}_{k,old} = [x_N \; x_{N+1} \; ... \; x_{2N-1}]^T \tag{2.69}$$

and the result is transformed by a $2N$-point FFT and represented by a diagonal matrix X_k as follows

$$X_k = \mathrm{diag}(X_0 \; X_1 \; ... \; X_N \; ... \; X_{2N-1}) \tag{2.70}$$

where diag$(\cdot)$ is an operator that forms a diagonal matrix. Note that each element X_i in (2.70) corresponds to a frequency bin while each element x_i in (2.68) and (2.69) is a time sample.

The transformed filter output,

$$\mathbf{Y} = [Y_0 \; Y_1 \; \ldots \; Y_N \; Y_{2N-1}]^T \tag{2.71}$$

is the product of the input matrix X with the transformed filter vector $\mathbf{H}$,

$$\mathbf{Y} = X\mathbf{H} \; . \tag{2.72}$$

The time-domain output vector, $\mathbf{y}$ is the inverse transform of $\mathbf{Y}$:

$$\mathbf{y} = [y_0 \; y_1 \; \ldots \; y_N \; y_{2N-1}]^T = \mathrm{IFFT}(\mathbf{Y}). \tag{2.73}$$

However, only the last N-points of $\mathbf{y}$ constitute the result of the linear convolution, and the first block must be discarded because it is the aliased portion of the circular convolution, according to the overlap-save method.

The error at each iteration is first computed in the time domain by subtracting the valid latter half of the current output block in (2.73) from the current block of the desired response,

$$\mathbf{e}_k = \mathbf{d}_k - \mathbf{y}_k. \tag{2.74}$$

This N-point error vector must be first augmented with N zeros,

$$\mathbf{e} = [0\ 0\ \ldots\ 0\ e_N\ e_{N+1}\ \ldots\ e_{2N-1}]^T \tag{2.75}$$

before taking its $2N$-point FFT. The augmented zeros must be in the first block because the first block is where no aliasing occurs in the correlation calculation to obtain the gradient estimate. The transformed error vector $\mathbf{E}_k$ at iteration k is

$$\mathbf{E}_k = [E_0\ E_1\ \ldots\ E_N\ \ldots\ E_{2N-1}]^T \tag{2.76}$$

The estimate gradient vector, $\mathbf{G}_k$, at the kth iteration is

$$\mathbf{G}_k = 2\mu_k X^H \mathbf{E} \tag{2.77}$$

where μ_k is a diagonal matrix of step sizes for the frequency bins, which will be discussed later.

The product of the FFT's in (2.77) above yields a circular correlation but only the first N points of the gradient vector in the time domain is valid. Therefore, we must eliminate the latter aliased half by constraining the time-domain gradient, $\mathbf{g}_k$, to be all zeros. The gradient constraint is done in the time domain so $\mathbf{G}_k$ must be inverse transformed to the time domain first:

$$\mathbf{g}_k = \mathrm{IFFT}(\mathbf{G}_k) \tag{2.78}$$

before constraining the last N elements of $\mathbf{g}_k$ as follows

$$\mathbf{g}_k = \mathrm{diag}(g_0\ g_1\ \ldots\ g_{N-1}\ 0\ 0 \ldots\ 0]^T)\ . \tag{2.79}$$

The time-domain constraint gradient above is then transformed back to $\mathbf{G}_k$ and is applied in the filter update equation:

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \mathbf{G}_k\ . \tag{2.80}$$

Step Sizes for the FBLMS Algorithm

Besides the computational savings, implementing the adaptive filter in the frequency domain has another advantage in that the DFT tends to decorrelate any input signal that is correlated (as usually the case in speech or image signals). Once uncorrelated, a different step size can be used for each frequency bin to achieve a more uniform convergence. It is well known that the converenge behavior of the adaptive algorithm is governed by the eigenvalue spread of the input autocorrelation matrix. In general, the eigenvalue spread of a signal is bounded by the maximum and minimum values of its power spectrum:

$$\frac{\lambda_{max}}{\lambda_{min}} \leq \frac{P_{max}}{P_{min}} \tag{2.81}$$

where the λ's are the eigenvalues and P's are values of the power spectrum [21].

In practice, one would not have available all the input signal to calculate the power spectrum but must estimate it instead. There are several different ways to estimate the power spectrum, but in this thesis we will use the following *exponential power averaging* function:

$$\hat{\mathbf{P}}_{\mathbf{k+1}} = \alpha\hat{\mathbf{P}}_{\mathbf{k}} + (1-\alpha)|\mathbf{X}_k|^2 \tag{2.82}$$

where α is the *memory factor* that controls how much of the current power estimation is based on the past estimation [34].

Using the power estimate $\hat{\mathbf{P}}_k$, we can normalize the eigenvalue spread by having a different step size for each frequency bin. The step size μ_i for the ith frequency bin is

$$\mu_i = \frac{\mu}{\hat{P}_i} \tag{2.83}$$

where μ is some convergence constant and P_i the estimate power of the ith bin. A diagonal matrix μ_k formed from the μ_i's above can be used in the estimate gradient $\mathbf{G}$ in (2.77).

The error function that this frequency-domain algorithm tries to minimize is the same as the block mean square error (BMSE) in (2.62) of the time-domain block LMS filter. Alternatively, we can also think that the FBLMS filter is designed to minimize the frequency-domain block mean-square error directly, since by Parseval's theorem, the error energy is equivalent both in the time and frequency domain. The frequency-domain block error is

$$\mathrm{FBSE} = \sum_{i=0}^{2N-1} |E_i|^2 = \mathbf{E}_k^H \mathbf{E}_k \tag{2.84}$$

where E_i's are the individual elements of the frequency-domain error vector $\mathbf{E}_k$ in (2.76).

Variations on the FBLMS Algorithms

There are two variations to the FBLMS algorithms. One is removing the gradient constraint in the dotted box in Figure 2-7. By doing so we can save two transform operations, an FFT and an IFFT, but the gradient vector will contain some aliasing since we no longer have the exact linear block correlation. Hence, this *unconstrained gradient* FBLMS, or UFBLMS, method does not exactly implement the time-domain block LMS (TBLMS) as the one with gradient constraint. It has been shown to converge to the Wiener solution under certain signal conditions, and to converge fast, especially for highly correlated input signals, but to a worse mean-squared error than the FBLMS with gradient constraint. [27].

The other variation of the FBLMS is to eliminate not only the gradient constraint but also the error constraint. In other words, the error is computed directly in the frequency domain by subtracting the transformed output $\mathbf{Y}$ from a current block of transformed desired vector $\mathbf{D}$:

$$\mathbf{E} = \mathbf{Y} - \mathbf{D} \ . \tag{2.85}$$

This is one of the earliest frequency-domain algorithms, first proposed by Dentino et al. [13].

By removing both the gradient and error constraints, this algorithm no longer implements the linear convolution or correlation, but the circular convolution/correlation instead, which gives this method the name *circular convolution algorithm.* Since it implements circular convolution/correlation, it is no longer necessary to use $2N$-point

FFT's or IFFT's as in the FBLMS filter. Recall that we need $2N$-point transforms so that N of those points would correspond to the linear convolution/correlation while the other N points are discarded because of the aliasing. Furthermore, $\mathbf{E}$ in (2.85) above provides N approximately orthogonal error outputs, each corresponding to a frequency bin. This is in contrast to a single global error that the FBLMS algorithms try to minimize.

By eliminating the error constraint, we further get rid of one FFT operation in addition to the two from using the unconstrained gradient. The total number of transforms needed here is only three, instead of five as in the original FBLMS algorithm, and each FFT and IFFT has only N points instead of $2N$ points. The tremendous gain in computational savings, however, come at a loss in filter performance. The circular convolution introduces large aliasing where only the first point of the convolution is valid. In other words, the circular convolution algorithm no longer implements the time-domain block LMS, and thus the converged solution is not exactly the Wiener filter. The paper [32] discusses all three frequency-domain algorithms (FBLMS, UFBLMS, and circular convolution algorithms) covered in this section.

2.1.8 Summary

In the first half of this background section, we have covered an array of different methods to solve the data-fitting problem discussed at the beginning. Although we will use only the least-squares and the frequency-domain block LMS (FBLMS) filters in this thesis, discussing the other algorithms (the Wiener filter, the LMS algorithm, and the block LMS algorithm) helps us to place the ones we use in proper perspective.

2.2 Multirate Digital Signal Processing

2.2.1 Introduction

In many applications, it may be more convenient and efficient to have a DSP system perform different processing algorithms at different sampling rates. Such a system, whose basic components are decimators or interpolators, is called a *multirate system* since its sampling rate varies across the system. Multirate techniques can often reduce the operation rate of the digital filters and lower the computational complexity of the system in terms of number of multiplications per second. More extensive treatments of multirate DSP may be found in [7],[11], [20], [26],[40].

2.2.2 Review of Decimation

Since we are using the decimator in this thesis, let's review it briefly. For more detailed coverage of decimation and interpolation, consult [28], [11].

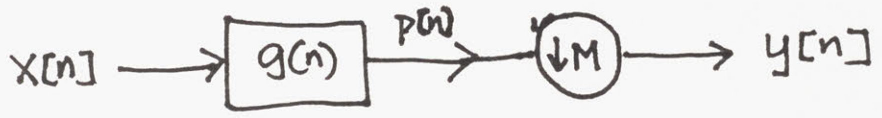


Figure 2-8: Block diagram for decimation by an integer factor M.

Figure 2-8 shows the block diagram of performing decimation by an integer factor M. Our objective in decimation is to reduce the sampling rate of a discrete-time signal

$x[n]$ by some factor M. Equation (2.86) shows the effect of decimation by M on the sampling period T and the sampling frequency F of $x[n]$, where the prime on them denote the downsampled output

$$\frac{T'}{T} = \frac{M}{1}$$

or

$$F' = \frac{1}{T'} = \frac{1}{MT} = \frac{F}{M} \,. \tag{2.86}$$

We can see from the results above that downsampling reduces the number of samples of $x[n]$ in the time domain wgle expanding its spectrum in the frequency domain.

Since $x[n]$ is a sampled version of an analog signal, its frequency spectrum is replicated at every 2π in the frequency domain. From Nyquist's sampling theorem, if the spectrum of $x[n]$ is not bandlimited to $\frac{\pi}{M}$, replications of the spectrum will overlap, or *alias*, with each other when the signal is downsampled by M. To ensure no aliasing, it is necessary to filter $x[n]$ with a digital lowpass filter $G(e^{j\omega})$ that approximates the ideal characteristic:

$$\tilde{G}(e^{j\omega}) = \begin{cases} 1, & |w| \leq \frac{2\pi F'T}{2} = \frac{\pi}{M} \\ 0, & \text{otherwise} \end{cases}$$

The output $p[n]$ in Figure 2-8 is simply the convolution

$$p[n] = x[n]g[n]$$

or

$$p[n] = \sum_{i=-\infty}^{\infty} g[i]x[n-i]$$

where $g[n]$ is the impulse response of the *digital anti-aliasing filter* $G(e^{jw})$.

The circle with the down arrow is a often called the *the compressor* because it compresses the sequence $x[n]$ by keeping only every Mth sample of the filtered output $p[n]$:

$$y[n] = p[Mn]$$

or

$$y[n] = \sum_{i=-\infty}^{\infty} g[i]x[Mn-i]$$

where $y[n]$ is the final result of decimation by M. The output $y[n]$ in the frequency domain is

$$Y(e^{j\omega}) = \frac{1}{M}\sum_{l=0}^{M-1} G(e^{-j(w-2\pi l)/M})X(e^{-j(w-2\pi l)/M})) .$$

If G is very close to the ideal response $\tilde{G}$, then

$$Y(e^{j\omega}) \approx \frac{1}{M}X(e^{j\omega/M}) \quad , |w| \leq \pi . \tag{2.87}$$

2.2.3 Multistage Decimation

Past research shows that for large decimation or interpolation factors, sampling rate conversion can be best done in cascaded stages [3], [11], [31]. Because the greatest complexity and cost in decimation or interpolation lie in the filtering, multistage decimation or interpolation will:

1. Simplify the design of the anti-aliasing (or anti-imaging) digital filters
2. Significantly reduce computational complexity
3. Reduce storage of filter coefficients
4. Reduce quantization error in the implementation of the filters.

Let's examine the multistage decimation process, which is used in this thesis. If the decimation factor is large and composite, then it may be factored into a product of I positive integers.

$$M = \prod_{i=1}^{I} M_i \tag{2.88}$$

The digital anti-aliasing filter before the decimator can in turn be decomposed into I smaller filters. A sub-decimator with a factor of M_i and a sub-filter together form a decimation stage. The cascade of all I decimation stages will yield the equivalence of a single-stage decimator with the downsampling factor of M. See Figure 2-9

Let's briefly examine where the savings in multistage decimation come from. For a symmetric, equiripple FIR filter, the number of taps, N_i, in stage i is approximately:

$$N_i \approx \frac{D(R_p^i, R_s^i)}{\triangle f_i / f_{si}} \tag{2.89}$$

where

$D(R_p^i, R_s^i)$ is a function of the maximum passband ripple R_p^i and the maximum stopband ripple R_s^i in stage i,

$\triangle f_i$ is the transition width, and

f_{si} is the sampling frequency.

Since decimation is performed in multistages, the filter in each stage has fewer constraints than in the single-stage case. Although the filters in the early stages must operate at higher sampling rates, they can have wider transition bands because aliasing is allowed. The aliasing poses no concern for it will be "cleaned up" in later stages by filters with narrower transition bands. Thus, higher sampling rates in the early stages are counterbalanced by the wider transition bands of the filters, which together give shorter overall filter lengths according to Equation (2.89). For the later stages, the transition widths of the filters must be narrower but the sampling rates are also lower, so this combination likewise yields smaller filter lengths. In other words, the computation in each stage is kept as low as possible so that the multistage structure can have fewer total number of filter taps than the single-stage one [11].

Similarly, the multiplication rate in the multistage structure is also reduced. The number of multiplication per second, or MPS, of stage i in the multirate system is approximately:

$$MPS \approx \frac{N_i f_{si}}{2M_i} \tag{2.90}$$

Since the total filter length is lower, the total number of multiplication is also proportionally reduced as shown in Equation (2.90). The high sampling frequencies in the early stages will give a larger MPS, but this effect can be offset by placing decimators with high downsampling factors in the beginning stages.

In summary, for a large decimation (or interpolation) factor, the multistage scheme is a better choice than the single-stage one in terms of computational and filter complexities. Multistage structures, however, also have their drawbacks. They require more control than the single-stage ones, and there are also the issues of choosing the appropriate number of decimation stages I, and the best factor M_i, for each stage [11].

More Exact Equations For Filter Length

The more exact equation for the filter length N for stage i is ([11, 20]):

$$N = 1 + \frac{D(R_p, R_s)}{\triangle F} - g(R_p, R_s)\triangle F \tag{2.91}$$

where

$\triangle F = \triangle f_i / f_{si}$ is the normalized transition width, where $\triangle f_i$ is the transition width of the ith stage and f_{si} its sampling frequency,

$D(R_p, R_s) = [a(\log R_p)^2 + b \log R_p - c] \log R_s - [d(\log R_p)^2 + e \log R_p + f]$,

$g(R_p, R_s) = g \log(R_p / R_s) + h$

and

$a = 0.005309, b = 0.07114, c = 0.4761, d = 0.00266,$

$e = 0.5941, f = 0.4278, g = 0.51244, h = 11.01$

All logs here have base 10.

A simpler equation of the filter length is given by Kaiser [20]:

$$N = \frac{-10log(R_p, R_s) - 13}{14.6\triangle F} + 1 \tag{2.92}$$

where again

$\triangle F = \triangle f_i / f_s$ is the normalized transition width.

Example of a Two-Stage Decimation

The following is an illustration of why multistage decimation can reduce the overall length of the digital anti-aliasing filter and its computational complexity. Let's look at decimation by 10, where we divide the downsampler into two stages with factors of 5 and 2 as shown in Figure 2-10.

First of all, let's state the assumptions and filter parameters used throughout this example. In order to avoid any confusion, we will do the the analysis in the unnormalized frequency domain. The sampling rate of the input x to the decimator is set at $2000Hz$, as shown in Figure 2-11. The passband ripple of the filter is R_p and the stopband ripple is R_s. Finally, to avoid aliasing, the cutoff frequency of the decimation filter should be:

$$f_{cutoff} = \frac{f_s}{2M} \tag{2.93}$$

where

$f_s/2$ is the Nyquist frequency or half of the sampling frequency, and

M is the decimation factor.[3]

One-Stage Decimation

For downsampling by 10, the cutoff frequency of the single-stage decimation filter without aliasing is

$$f_{cutoff} = 1000/10 = 100Hz \tag{2.94}$$

The passband is chosen to be from 0 to $90Hz$, which leaves a $10Hz$ transition width. Using these parameters along with the chosen passband ripple and stopband attenuation, Equations (2.91) and (2.92) give filter lengths of 557 and 517, respectively. The actual design takes 527 taps. The number of multiplications per second

[3]The exact parameters and designed filters are detailed in Chapter 3 on the Simulation.

(MPS) is given by $MPS = \frac{1}{2}Nf_s/M$ as discussed earlier. Since the sampling rate is reduced after decimation, only every Mth output point needs multiplications. Using the actual filter length, MPS for this single-stage decimation is 52,700.

Two-Stage Decimation

Now let us look at the filter requirements for two-stage decimation and see where the savings in filter lengths and number of multiplications come from. First of all, keep in mind that the (logarithmic) passband ripple and the stopband attenuation of each of the stages in cascade add up. Therefore, the specifications of these ripples in each of the two stages should be less than that of the single-stage, and their sum in cascade should be equal to that of the single-stage.

Since 10 is the ultimate downsampling factor we want, each of the two decimation filters in cascade can have the same passband, 0 to 90 Hz, as the single stage filter above. The maximum possible stopband, or cutoff frequency, starts at:

$$f_{ci} = f_{si}/M_i - f_{cutoff} \tag{2.95}$$

where

f_{ci} is the cutoff frequency of the ith stage,
f_{si} is the sampling frequency of the ith stage,
M_i is the decimation factor of the ith stage,
and f_{cutoff} is the ultimate desired cutoff frequency as in equation above

Because of the narrower passband, the transition width of the first decimation filter can stretch out further than usual without aliasing. According to the filter length equation, for a given sampling rate the filter length will become shorter as the the transition width gets wider. In this example, the filter for the decimation by 5 of stage 1 can have its cutoff frequency at

$$f_{c1} = 2000/5 - 100 = 300Hz$$

and a transition width of

$$\triangle f_1 = 300 - 90 = 270Hz$$

as followed from Equation (2.95). With these parameters, Equations (2.91) and (2.92) predict filter lengths of 26 and 25, respectively. The actual design needs 27 taps and 527 MPS.

After the first stage, the sampling rate of the signal has been reduced by 5 and now becomes

$$f_{s2} = 2000/5 = 400Hz$$

Again, following from Equation (2.95), its cutoff frequency should be

$$f_{c2} = 400/2 - 100 = 100Hz$$

which gives a transition width of $10Hz$. Although the transition width of this second stage is the same as that of the single-stage decimation-by-10 filter, its sampling rate is reduced and hence still yields a shorter filter length. From Equations (2.91) and (2.92) we obtain lengths of 149 and 135, respectively, but the actual design gives 137 with 2700 MPS.

Summary of Example

In summary, for decimation by 10, single-stage filtering requires 527 taps and 52,700 multiplications per second, whereas two-stage filtering requires a total of only 164 taps and 8,100 MPS. The total filter length of the two-stage structure is reduced by about a third while its total MPS is more than six times lower than that of the single stage. The reduction is indeed significant even at a low decimation factor as shown in this example, and will become more substantial as the downsampling factor gets larger.

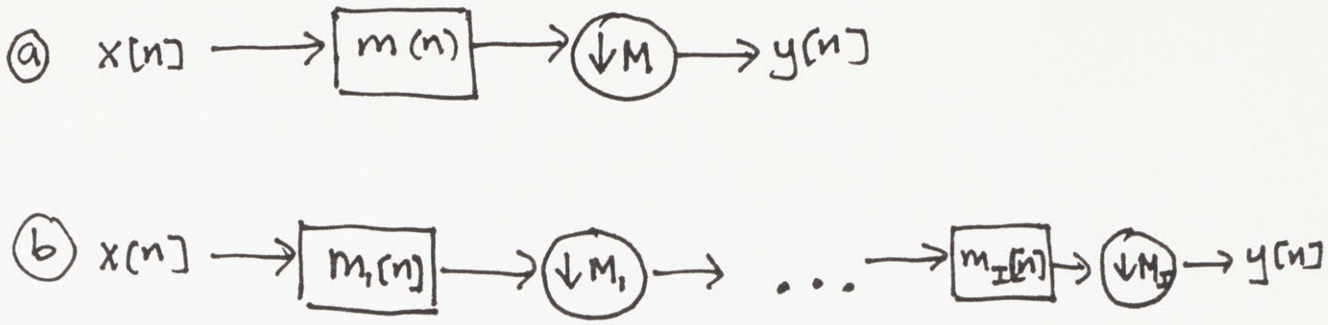
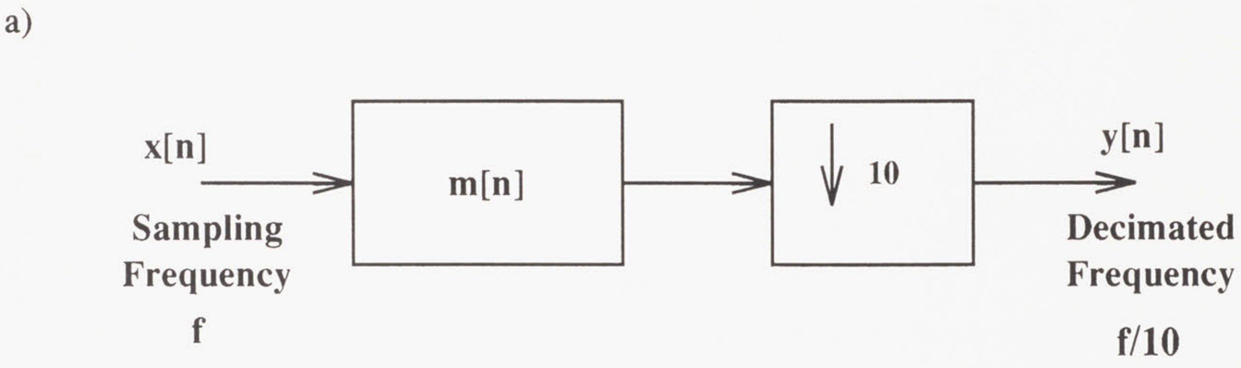


Figure 2-9: Single stage decimation and multistage decimation.



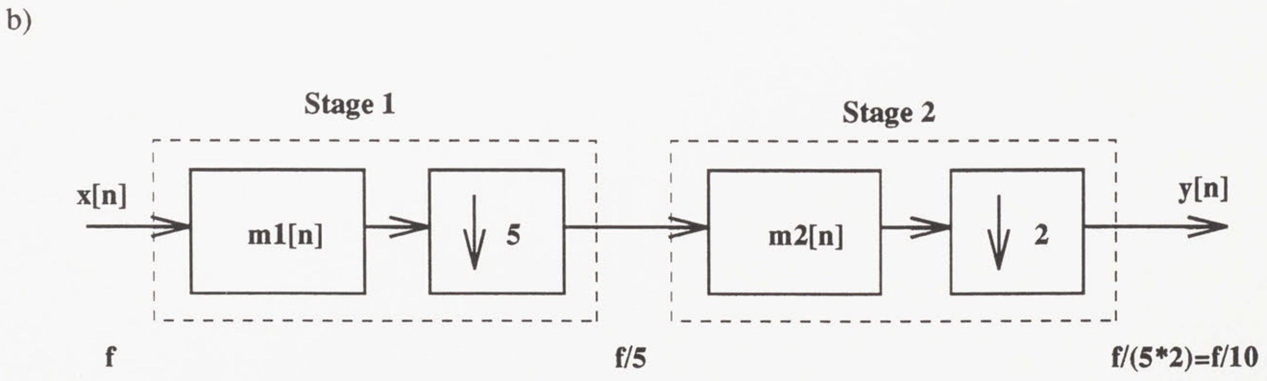


Figure 2-10: Structure of (a) single stage and (2) multistage decimation for factor of 10.

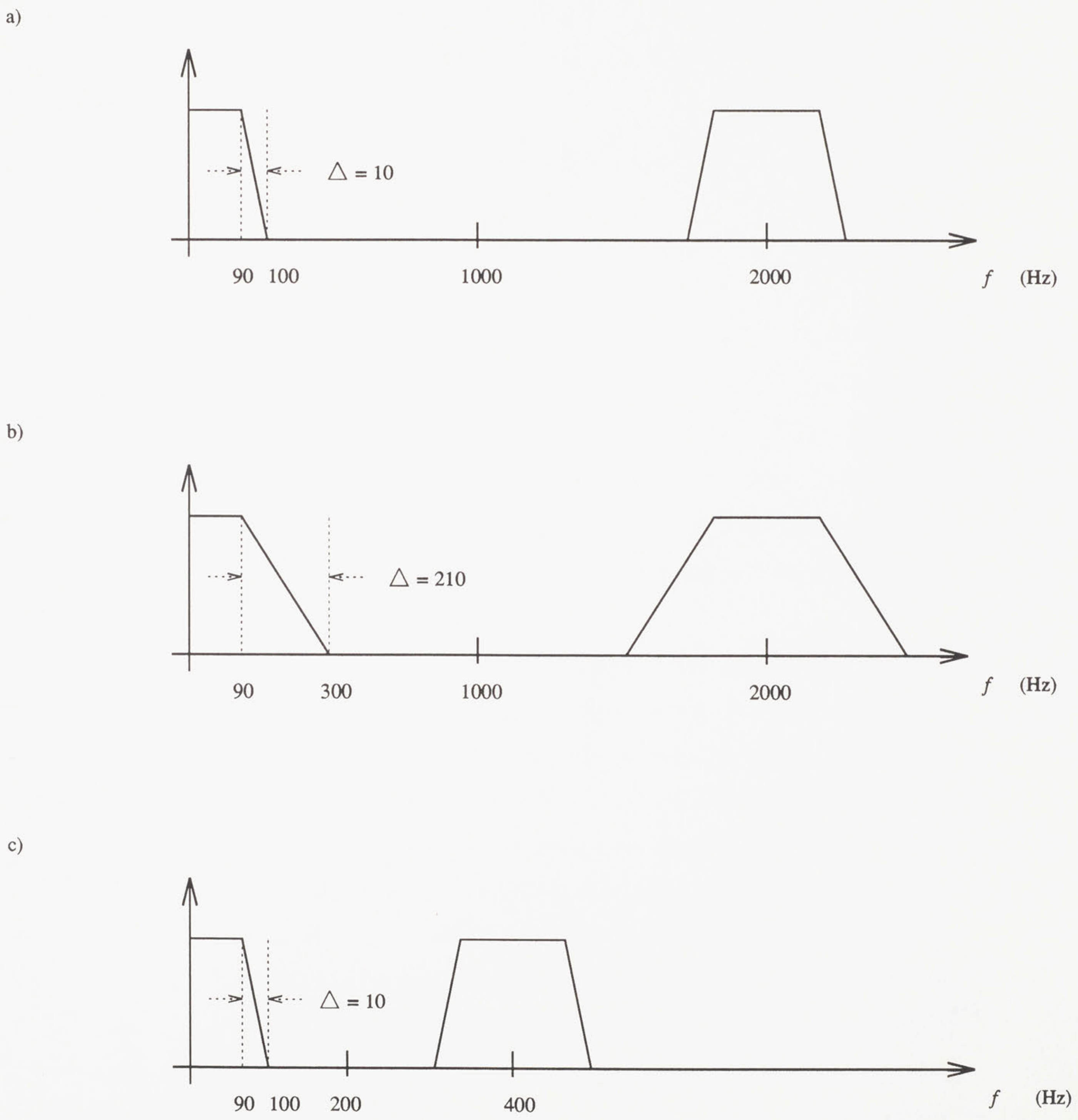


Figure 2-11: Example of filters for (a) single stage and (b), (c) multistage decimation.

Chapter 3

The Multistage, Multirate Adaptive Filter

3.1 Introduction

Research in adaptive filtering and multirate digital signal processing spans several decades, but only within the last five years or so have these two fields started to merge in some useful applications. A prime example of this union is adaptive filtering in subbands as introduced in Section 1.2.

The problem of this thesis may be viewed as adaptive filtering for only one subband from the filter bank structure, in particular, the subband centered around zero frequency. Instead of using a filter bank, it is only necessary to design a filter that picks out just the subband of interest. We will then use this subband to study the multistage adaptive filter structure as will be described shortly.

There are several reasons for not investigating our proposed multistage, multirate adaptive filter on all the subbands from the filter bank. Firstly, isolating the problem to one subband simplifies the study a great deal. Secondly, we want to use a multistage decimator to downsample the signal, which has not been done before in the context of adaptive filtering in subbands. Finally, the most crucial reason of all is that adaptive filtering in subbands poses many problematic issues that are still under investigation. For example, aliasing between adjacent bands occurs when critically sub-sampled filter banks are used, where the signal is downsampled at the exact cut-off frequency of the digital anti-aliasing filter with finite transition widths. This aliasing problem can be avoided by using non-overlapping bands filter banks, but the reconstructed signals will have spectral gaps which may not be acceptable in some applications. A scheme has been proposed to avoid both of those issues by adding "cross-terms" to the adaptive filter in adjacent bands, but this modification is complicated and has its own problems as well [16, 17, 18].

Therefore, avoiding the filter bank structure in our study of the multistage, multirate adaptive filter is a sensible approach. Furthermore, the thesis problem can be generalized outside of the context of adaptive filtering in subbands. It can be viewed as adaptive filtering of any narrowband signal.

The multistage, multirate adaptive filter will be introduced and compared to the single-stage, multirate adaptive filter in Section 3.2. There we will present, using Fourier analysis, the effects of the single-stage and multistage equalizer on the input signal, which will clarify the similarities and differences between the two structures. In Section 3.4 we will discuss some of the limitations of performing analysis on the multirate, multistage filter, which will lead us to study the proposed structure using software simulations as discussed in Section 3.5. In that section, we will first discuss the three system configurations used in the simulation, how we model the components in the systems, and finally present two algorithms that we study, the least-squares filter and the adaptive frequency-domain block LMS filter. The reason we want to study the non-adaptive least-squares filter first is that we are interested in the multistage equalization filter structure itself, even without subjecting it to an adaptive environment. The problems that we discovered in our time-domain least-squares algorithm then motivate us to use an adaptive algorithm in the frequency domain. In Sections 3.6 and 3.7, we will present the computational complexities of the multirate, multistage filter using the least-squares method and the FBLMS algorithm. Finally, some representative simulations will illustrate the properties of this multistage, multirate filter.

3.2 The Multistage, Multirate Adaptive Filter

The problem that motivates our work is adaptive filtering of a narrow band of a signal which is depicted by the block diagram in Figure 3-1. We want to equalize only a narrow band of the output $x_0[n]$ of the distortion channel. The decimator picks out this narrow band signal, $x[n]$, by filtering $x_0[n]$ with a digital anti-aliasing filter $g[n]$, and it also downsamples the signal by a factor M. The box with the down arrow box denotes the decimator, which includes both the digital anti-aliasing filter and the time-compressor as illustrated in Figure 3-2. This top path is called the *distortion path.* The *desired path*, parallel to it, contains a desired channel and a delay element to match the delay of the desired path. The equalizer $h[n]$ corrects the narrowband signal, $x[n]$, by minimizing a function of the error $e[n]$, which is the difference between the filter output $y[n]$ and the desired response $d[n]$.

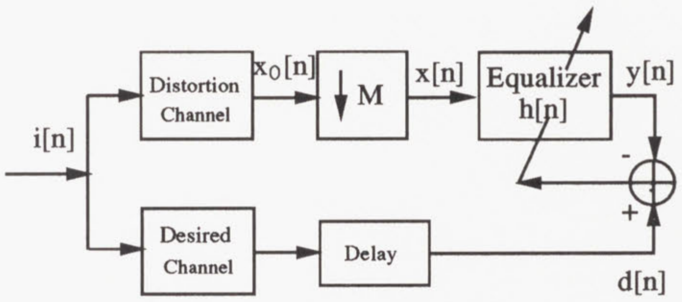


Figure 3-1: Block diagram of of the single-stage decimation, single-stage adaptive filter .

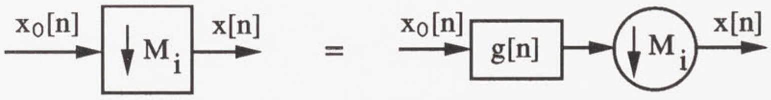


Figure 3-2: Block diagram equivalence for the decimator.

In this thesis, instead of doing the equalization in one single stage, we are interested in doing it in multiple stages, as Figure 3-3 illustrates for a two-stage structure. The decimator is also in multiple stages since that has shown to have some advantages in practice, as discussed in Section 2.2. We decompose the equalizer into two stages and insert the first one in between the two decimators.

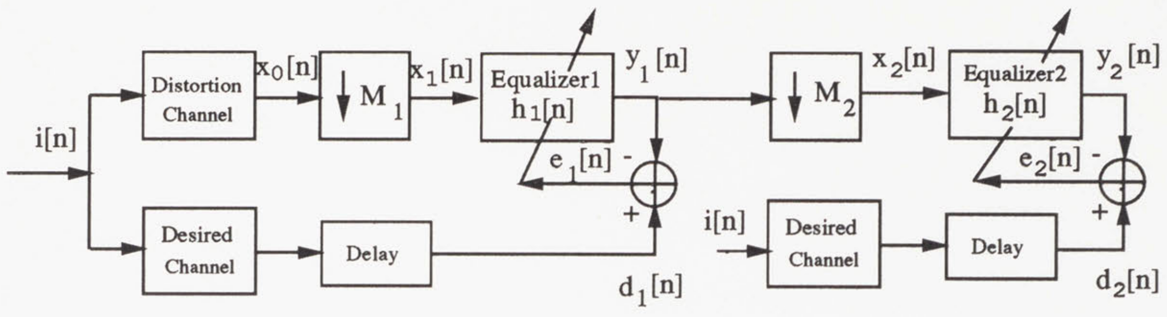


Figure 3-3: Block diagram of the two-stage decimation, two-stage adaptive filter.

Before formulating the problem statement of the multistage, multirate adaptive filter, let us first examine the single-stage decimation, single-stage adaptive filter as a basis of comparison with the proposed filter structure. We will analyze, in the Fourier domain, the effect of the input signal $x_0[n]$, which is the output of the distortion channel, as it passes through the distortion path of the system. First, assume that the digital anti-aliasing filter $g[n]$ for the decimator with factor M is ideal and has the frequency response,

$$G(e^{j\omega}) = \begin{cases} 1, & |\omega| \le \frac{\pi}{M} \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

For $G(e^{j\omega})$ ideal, the output of the decimator, $X(e^{j\omega})$, can be represented as

$$X(e^{j\omega}) = \frac{1}{M} X_0(e^{j\omega/M}) \quad , |w| \le \pi \tag{3.2}$$

as shown in Section 2.2. Because we assume the digital anti-aliasing filter is ideal here, it is not necessary to compare here the proposed multistage adaptive filter with the multistage decimation, single-stage filter. Recall from Section 2.2 that the advantage of the multistage decimation lies in the actual design of the non-ideal digital anti-aliasing filter. However, in our simulations and in actual implementations, the filters are not ideal and the output will be affected by passband ripples and aliasing.

The output $Y(e^{j\omega})$ of the adaptive filter $H(e^{j\omega})$ is

$$\begin{aligned} Y(e^{j\omega}) &= H(e^{j\omega})X(e^{j\omega}) && (3.3) \\ &= \frac{1}{M}H(e^{j\omega})X_0(e^{j\omega/M}) && (3.4) \end{aligned}$$

for $|w| \leq \pi$.

Now we examine what happens to the same $x_0[n]$ as it is processed by the multi-stage, multirate adaptive filter. Again, we assume that the digital anti-aliasing filters $G_1(e^{j\omega})$ and $G_2(e^{j\omega})$ are ideal.

The output $Y_1(e^{j\omega})$ of the first stage adaptive filter has the identical form to (3.4) of the single-stage filter. It is

$$\begin{aligned} Y_1(e^{j\omega}) &= X_1(e^{j\omega})H_1(e^{j\omega}) && (3.5) \\ &= \frac{1}{M_1}X_1(e^{j\omega/M_1})H_1(e^{j\omega}) && (3.6) \end{aligned}$$

for $|w| \leq \pi$.

The critical difference between the two filter structures occurs in the second stage, where the signal is decimated again. After decimating the output of the first equalizer by the second decimator with factor M_2, we get the output, $x_2[n]$, whose Fourier transform is

$$\begin{aligned} X_2(e^{j\omega}) &= \frac{1}{M_2}Y_1(e^{j\omega}) && (3.7) \\ &= \frac{1}{M_1 M_2}X_0(e^{j\omega/(M_1 M_2)})H_1(e^{j\omega/M_2}) && (3.8) \end{aligned}$$

for $|w| \leq \pi$.

Note that not only the original input signal $x_0[n]$ is decimated, but the first adaptive filter $h_1[n]$ as well.

Finally, the output of the second adaptive filter is

$$Y_2(e^{j\omega}) = X_2(e^{j\omega})H_2(e^{j\omega}) \qquad (3.9)$$

$$= \frac{1}{M_1 M_2} X_0(e^{j\omega/(M_1 M_2)}) H_1(e^{j\omega/M_2}) H_2(e^{j\omega}) \tag{3.10}$$

for $|w| \leq \pi$.

Since the digital anti-aliasing filters of both systems are ideal, the decimated input signals, $X_0(e^{j\omega/(M_1 M_2)})$, are the same, but not the adaptive filters. The frequency-response of the single-stage, single-rate filter is

$$H(e^{j\omega}) = \sum_{n=0}^{N-1} h[n] e^{-j\omega n} \ . \tag{3.11}$$

For the multistage, multirate structure, the first filter has N_1 taps and the second N_2 taps, and the combined filter has the frequency response

$$\begin{aligned} H'(e^{j\omega}) &= H_1(e^{j\omega/M_2}) H_2(e^{j\omega}) & (3.12) \\ &= (\sum_{n=0}^{N_1-1} h_1[n] e^{-j\omega n/M_2})(\sum_{n=0}^{N_2-1} h_2[n] e^{-j\omega n}) & (3.13) \end{aligned}$$

The objective of this thesis may be stated as follows:
Given the single-stage filter (3.11) and the multi-stage, multirate filter (3.13), how do the performances of the two structures compare for an equivalent measure of filter lengths. In other words, for $N \approx f(N_1, N_2, M_2)$ where f is some function, how will the two structures compare in terms of residual error from equalization and computational complexity. These performance criteria will be discussed next.

3.3 Basis for Comparison

The comparison between the multistage adaptive filter with single stage adaptive filter will be based on two criteria: (1) residual error and (2) computational complexity. The residual error indicates how well the filter equalizes the distortion channel, and the calculation of the error depends on the algorithm that we use. It is called the least-squares error (LSE) for the least-squares filter, and minimum-mean square error (MSE) for the FBLMS filter.

There are two measures of computational complexity. The first is the number of computations it takes to obtain the equalization filter, which also reflects complexity of the algorithm that we use. For the least-squares method, finding the equalizer involves solving a known system of equations. For the FBLMS algorithm, it involves adapting the filter to the signal environment over time. The parameters that will figure in this complexity are the lengths of the equalizer, its input, and the desired response. For the FBLMS filter, the convergence time also matters because it is adaptive. In addition, the computational efficiency of an algorithm usually involves many issues, such as storage requirements and number of machine cycles for CPU, but these are mainly implementation issues which depend on the specific hardware architecture. Therefore, we will only examine the number of real multiplications as a measure of the computational complexity of the algorithm. The calculations of the filter design complexities will be provided in details in Sections 3.6.1 and 3.7.2 when we discuss the simulation studies of the least-squares and the FBLMS filters.

The second measure of computational complexity involves using the filter once it is designed, such as equalizing a distorted signal, for instance. This convolution between the input signal and the designed filter can be implemented in either the time or frequency domain, each of which yields different amounts of multiplications. To avoid this peripheral complication, we will measure this filtering complexity by the *effective filter length* (EFL), which will be defined for each filter structure.

For the single-stage structure, let's normalize the sampling rate of the filter input to 1 since the input is completely downsampled to the desired factor M. For the two-stage equalizer, the input to the second-stage equalizer also has the same sampling rate as that of the single-stage equalizer. However, the input to the first-stage equalizer has approximately M_2 times the sample points of the input to the second-stage equalizer since it has only been downsampled by the first decimator (M_2 is the decimation factor of the second decimator). Therefore, the *effective filter length* (EFL) of the single-stage structure is simply the filter length N_h:

$$EFL = N_h \ , \qquad (3.14)$$

but for the two-stage adaptive filter structure, it is

$$EFL = M_2 N_{h1} + N_{h2} \ . \tag{3.15}$$

3.4 Motivation for a Simulation Study

In this thesis, we study the multistage, multirate filter based on its computational complexity and the residual error. We are able to assess the computational complexities of this new structure for two algorithms, the least-squares and the frequency-domain block LMS methods. However, we have not been successful with the derivation of some analytical equations that govern the convergence behavior and/or the residual error of the overall structure. The difficulties here are two-fold: 1) the stages are in cascade so the convergence behavior and/or the residue error of each filter are dependent on the previous stages, and 2) each stage in the structure is operating at a different sampling rate.

One way to simplify these problems is to view each stage individually so the residual error and/or the convergence behavior would be the same as that of the single-stage filter. On this local scale, each filter has its own distortion, and it operates at a single sampling rate. The cascaded effect of the filters, however, cannot be entirely ignored.

Because of these issues, we will approach the study of the multistage, multirate adaptive filter using simulations with a representative example. We will first develop some models of the components in the system and then simulate them. What we present here is a possible approach to answer this problem, and the conclusions from this study may not be true for all cases until further investigations confirm so.

3.5 A Model System for Simulation Studies

3.5.1 Introduction

This chapter presents some simulation results of multistage equalization as compared to single-stage equalization, both operating in a multirate environment. The goal of these simulations is to study the multistage, multirate equalizer using the system models we develop. In particular, we want to compare its performance with the single-stage equalizer based on the residual error and computational complexity. We are also interested in how we should decompose the equalizer for the multistage structure. The conclusions that we obtain from these experiments pertain only to the models that we choose, and they may not apply to other system models. However, what we illustrate here is a possible approach that one can take to study this problem. Further investigation is needed before the properties of the multistage, multirate filter can be fully generalized.

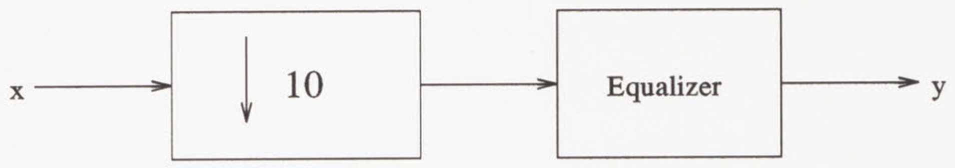
We use software for the simulations and thus do not address some issues that hardware implementation might raise, such as memory storage, delay, hardware architectures, etc. Except for the time-domain least squares algorithm which is written in C, all components of the simulations are written in Matlab. Furthermore, all the calculations are done in double precision, which is 16 bits on the workstations that run the simulations, so quantization error is rather low here.

To simplify the study, only two equalization and two decimation stages are considered here. The decimation factor is 10, and it is divided into two stages of 5 and 2. Figure 3-4 shows three possible system configurations of the adaptive filters in a multirate environment. The box with a down arrow is the decimator which includes both the digital anti-aliasing filter and the time compressor, as illustrated before in Figure 3-2. The equalizer plus the decimator preceding it is called "a stage" from now on. We will sometimes refer to these three system configurations by the following names:

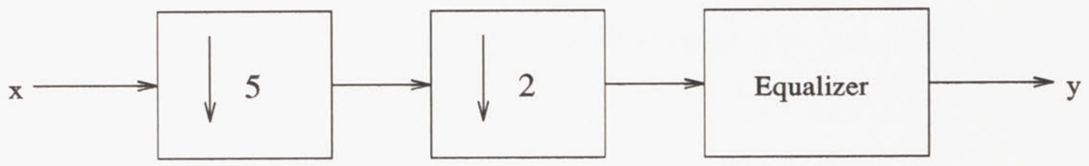
- **System 1**: Single-stage decimation, single-stage equalization filter.

- **System 2**: Two-stage decimation, single-stage equalization filter.
- **System 3**: Two-stage decimation, two-stage equalization filter.

1. A one-stage decimator followed by a one-stage equalizer.



2. A two-stage decimator followed by a one-stage equalizer.



3. A two-stage decimator and a two-stage equalizer.

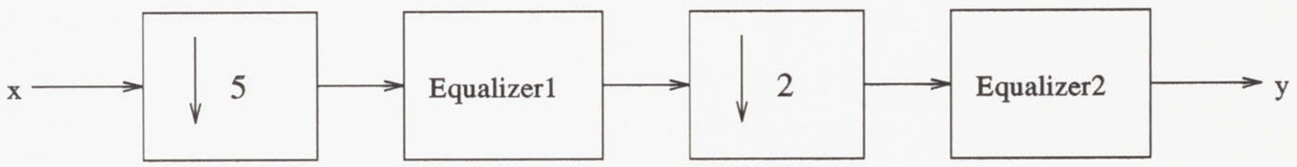


Figure 3-4: Three different system configurations for study.

Only the simulation of the time-domain least squares approach studies all three structures shown above. The rest of the simulations implementing other algorithms compare only configurations 2 and 3.

We use the following adaptive algorithms in the simulations:

- Time-Domain Least Squares
- Frequency-Domain Block Least Mean Squares (FBLMS)

The sections following this introduction will present the simulation results of each of the algorithms above. But before delving into the outcomes, we will first look at how we model the components in the system for the simulations: the distortion, the desired response, the delay, and the decimation filters. Since it is not practical to do an exhaustive study of every possible system component (i.e. different kinds of channel distortion and different decimation factors), we will concentrate only on a few representative models and report on the trends that we observe and provide explanations for them. The procedure we use here may be applied to study other system configurations.

3.5.2 The Distortion Channel

We model the distortion channel $q[n]$ or $Q(z)$ as a cascade of an FIR filter with an all-pole IIR one. That is,

$$Q(z) = A(z)\frac{1}{B(z)} = \frac{A(z)}{B(z)} \tag{3.16}$$

where $A(z)$ and $1/B(z)$ [1] are the system functions of the FIR and the IIR filters, respectively. While the $A(z)$ contributes to the zeros of the final distortion, $B(z)$ provides the poles. This representation of the distortion channel is also known in the literature as *autoregressive, moving-average* (ARMA) modelling. ([21], [20]).

The distortion to be equalized by the adaptive filter must have significant magnitude ripple in the passband as well as nonlinear phase, even after decimation by 10. This is because we are interested in the scenario where the adaptive filter is long in order to justify multistage equalization. This occurs when the desired response has very small ripple and/or the distortion is really severe.

This can be obtained by using the model in (3.16) above, where the function of the FIR filter is to fashion the magnitude ripple of the distortion channel, and the IIR filter is to add nonlinear phase. Of the different digital all-pole filters we tried, none yielded the desired level of magnitude ripple without becoming unstable itself,

[1]In general, an IIR filter may have its own zeros as well so its system function would be $\frac{C(z}{B(z)}$.

because the required filter order is rather large. However, using an FIR filter, we can easily create the desired ripple without demanding a long IIR filter.

The passband ripple of the distortion channel is about 30 dB peak-to-peak and its phase is 8 degrees nonlinear as shown in Figure 3-5. A 50th order, symmetric FIR filter, designed by the Parks-McClellan algorithm, provides the magnitude ripple, and a 9th order Chebyshev Type I filter furnishes the non-linear phase. Note that the digital IIR filter here is the bilinear transformation of an analog filter with frequency warping. (See [28] for a review of digital filters.) These two particular filters fit the ARMA model described in (3.16) rather well. While the FIR filter contains all the zeros, the Chebyshev Type I filter contains all the poles since it has no zeros. Furthermore, because the FIR filter is symmetric, it does not contribute to the nonlinearity of the phase at all. 0

Below is a summary of the characteristics of the FIR and IIR filters and the distortion channel.

Characteristics of the FIR Filter

Filter length:	51
Passband:	0.9
Stopband:	0.92
Transition width:	0.02
Passband ripple in dB (peak-to peak):	11.4
Stopband attenuation in dB:	-67

Characteristics of the IIR Filter

Filter order:	10
Passband:	0.9
Stopband:	0.92
Transition width:	0.02
Passband ripple in dB (peak-to peak):	0.52
Stopband attenuation in dB:	-200
Phase nonlinearity in degree:	8

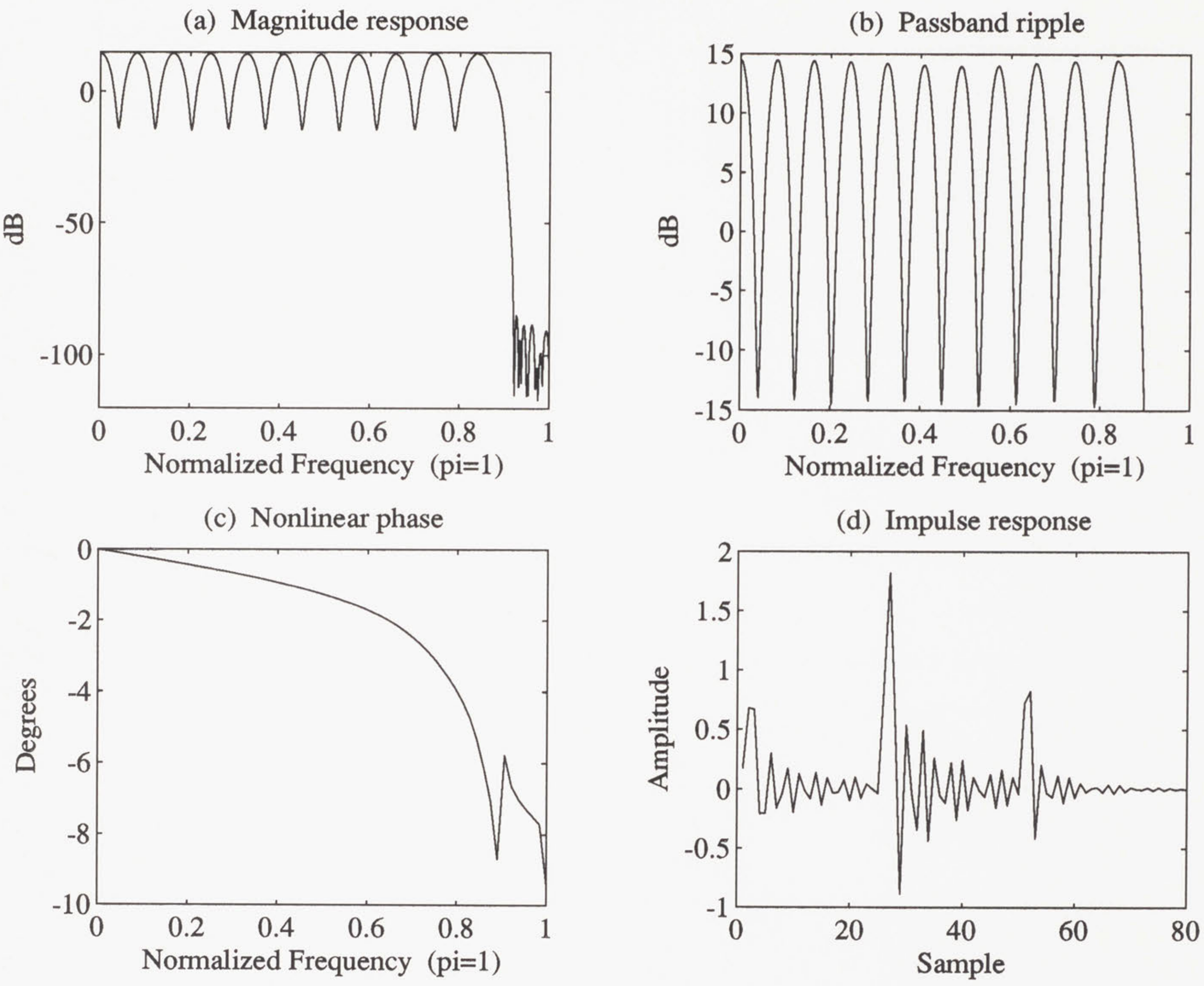


Figure 3-5: Characteristics of the distortion channel

Characteristics of the Distortion

Filter length:	1150
Passband:	0.9
Stopband:	0.92
Transition width:	0.02
Passband ripple in dB (peak-to peak):	30
Stopband attenuation in dB:	-80
Phase nonlinearity in degree:	8

3.5.3 The Desired Response

The desired response is an FIR filter designed using the Parks-McClellan algorithm, and the same desired response is used in each of the system configurations in Figure 3-4. Furthermore, we do not need to decimate the desired signal in the two-stage equalization scheme because in the digital domain the frequency is normalized so the different sampling rate conversions are immaterial. In addition, the desired response has equiripple in the passband so decimating it will give about the same response except with fewer ripples. Thus, by not decimating we can reduce the complexity of the desired path. The desired response's characteristics are given in Figure 3-6 and in the specifications below.

Filter length:	113
Passband:	0.96
Stopband:	0.9
Transition width:	0.06
Passband ripple in dB (peak-to peak):	0.005
Stopband attenuation in dB:	-60

3.5.4 The Delay Element

Besides the delay that makes the system causal, oftentimes another time delay must be added to the desired path or even to the distortion path to compensate for the time difference in the two outputs. Exceptional to simulation studies such as this one is that we know what the distortion signal is since we design it ourselves. With this foreknowledge and the given desired response and adaptive filter length–which are already known beforehand in real situations–we can determine the relative delay between the two paths in the adaptive filter structure.

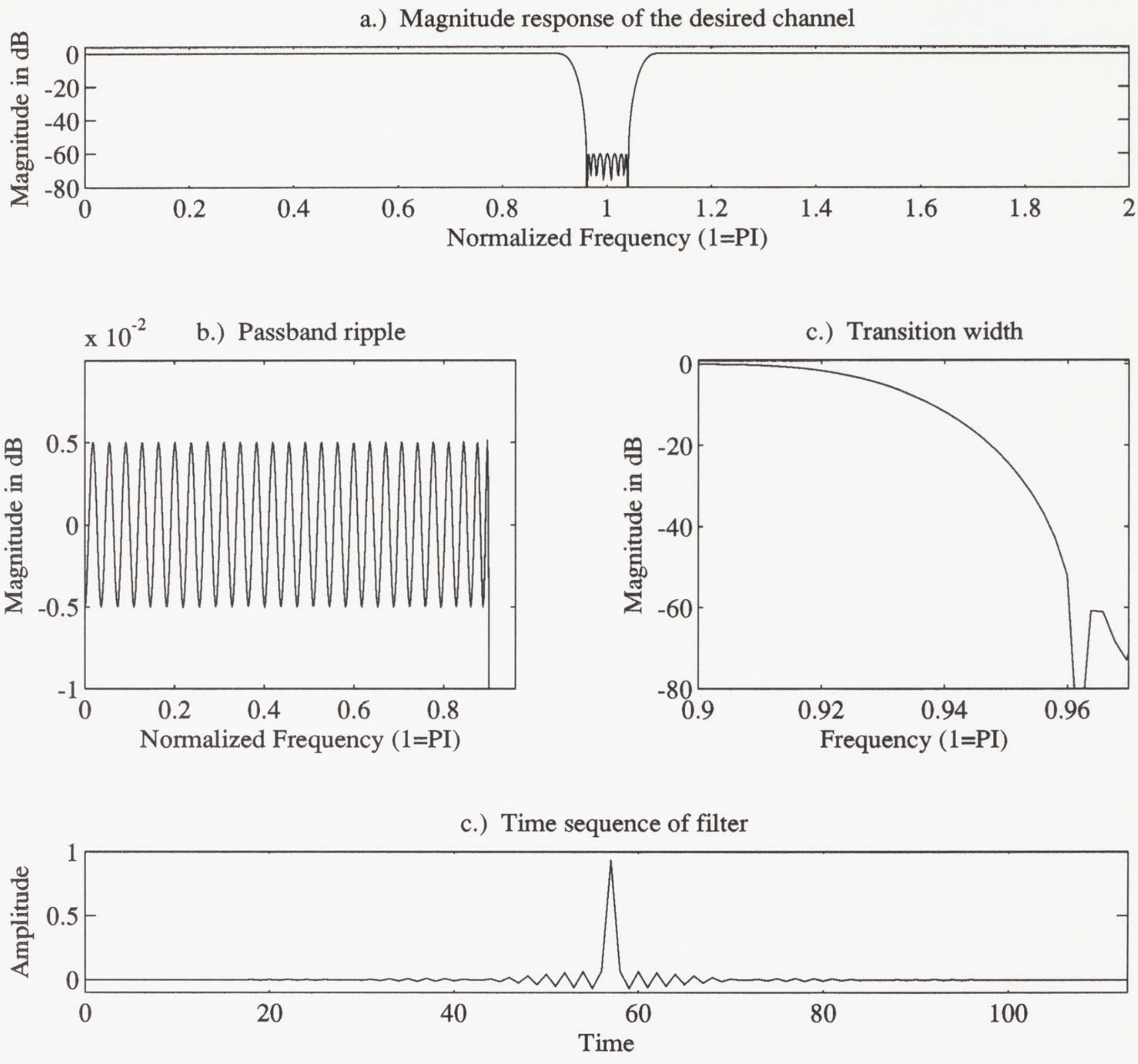


Figure 3-6: Characteristics of the desired response.

The delay in this simulation is estimated from the given filter length, N, and the peaks of the main lobes of the time-domain distortion and desired signals, *xpeak* and *dpeak* respectively, as follows:

$$delay = (xpeak + N/2) - dpeak \ . \tag{3.17}$$

If *delay* in the above equation is *positive* then that much delay must be added to the desired path, else the the distortion path needs that extra delay. The latter case can occur when the adaptive filter is short, for example.

In practice, however, the peak of the distortion might be impossible to pinpoint since it may be time-varying; thus the delay must be estimated by some other methods. Our chosen method, though unrealistic, is necessary to avoid choosing the delay arbitrarily, which may "unfairly" affect the outcome of each case study.

3.5.5 The Decimation Filters

The digital prefilter in each decimator is designed by the Parks-McClellan algorithms. We summarize their characteristics below while Figures ??,??,?? display their properties graphically. (*Note*: All frequencies are normalized where $1 = \pi$.)

Decimation-by-10 Filter

Filter length:	527
Passband:	0.09
Stopband:	0.1
Transition width:	0.01
Passband ripple, R_p:	$5.68 * 10^{-3}$
Stopband ripple, R_s:	$1.51 * 10^{-3}$
Passband ripple in dB (peak-to peak):	$9.83 * 10{-}2$
Stopband attenuation in dB:	-56.4

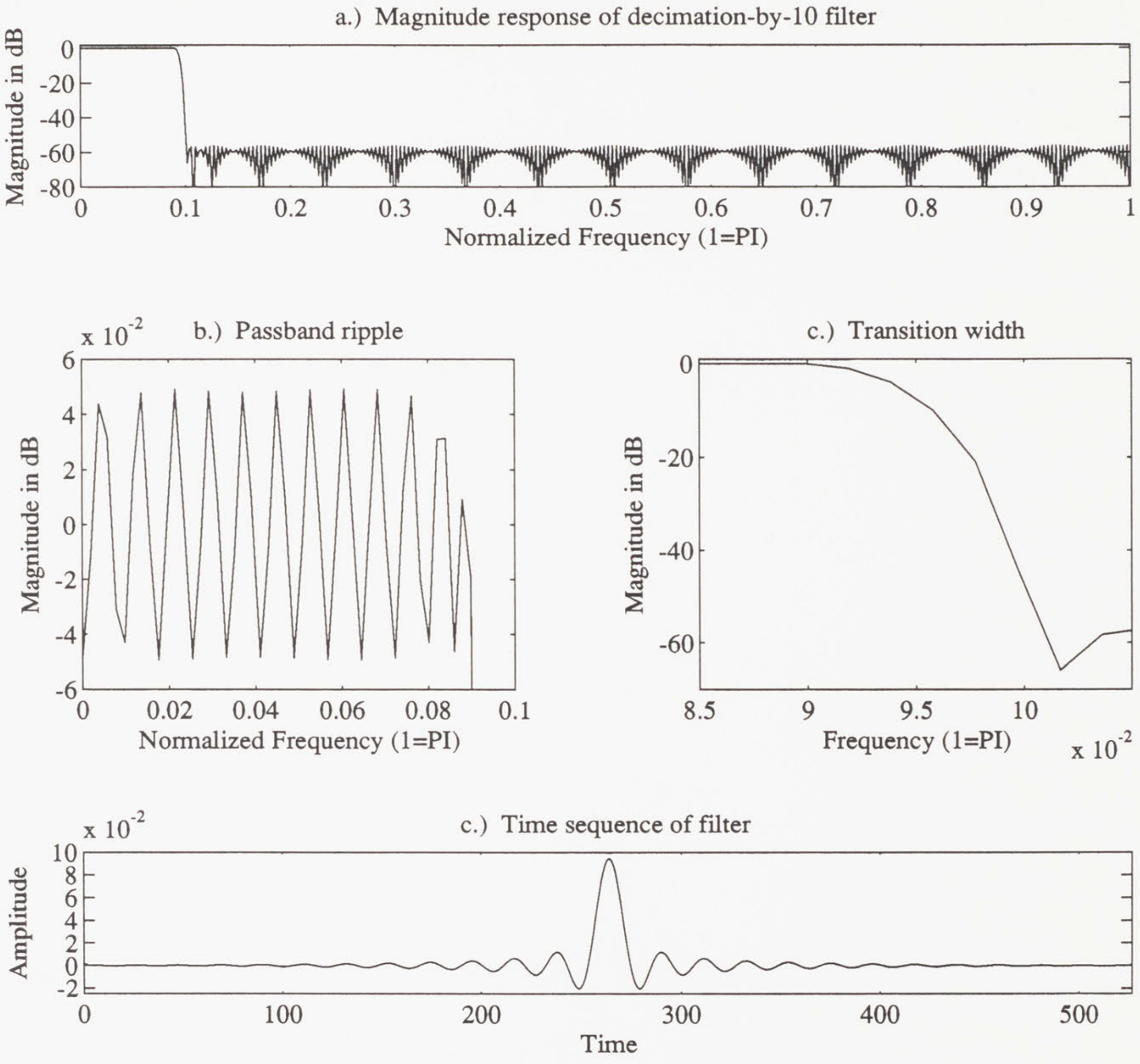


Figure 3-7: Characteristics of the decimation-by-10 filter.

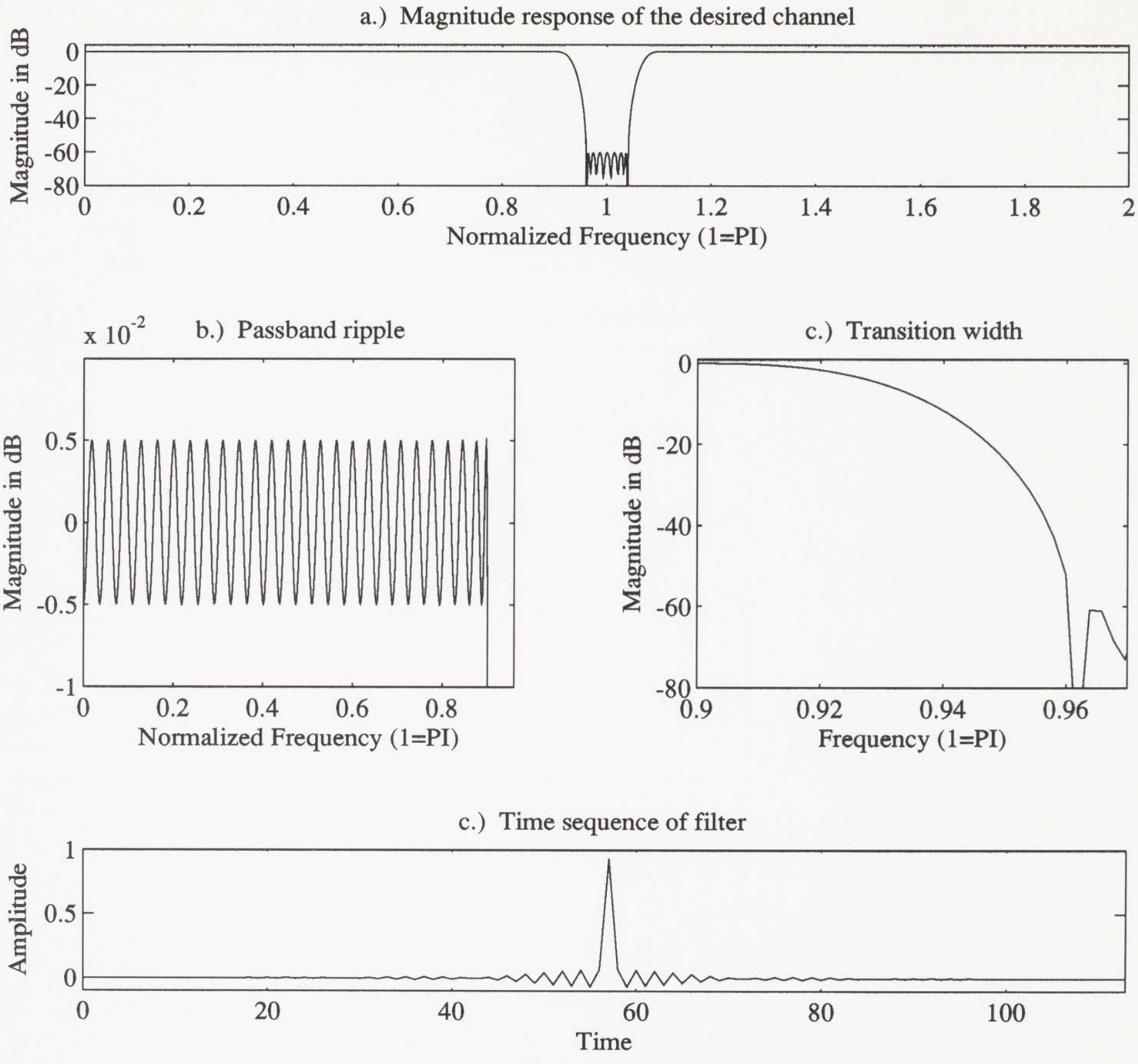


Figure 3-8: Characteristics of the decimation-by-5 filter.

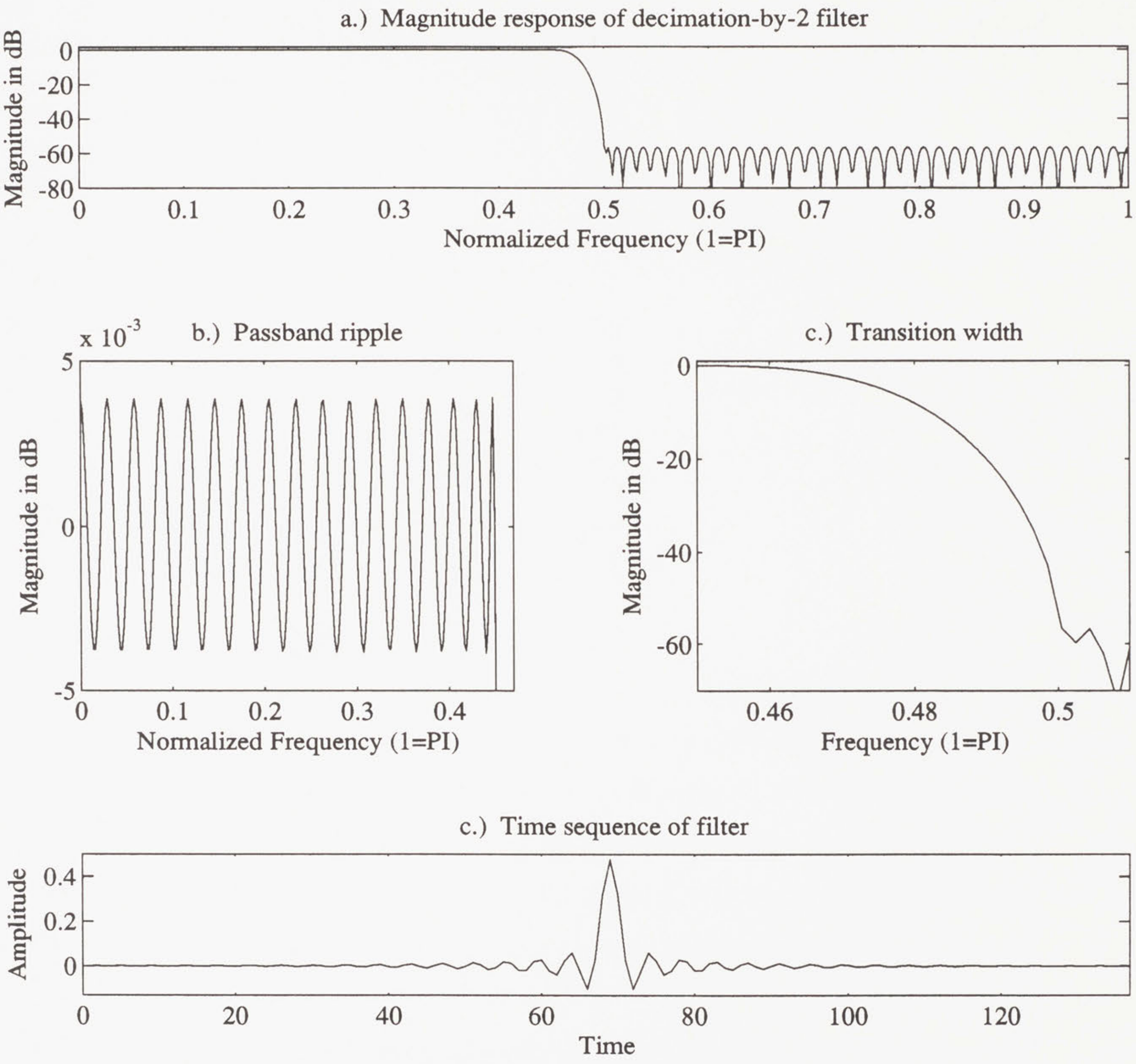


Figure 3-9: Characteristics of the decimation-by-2 filter.

Decimation-by-5 Filter

Filter length:	27
Passband:	0.09
Stopband:	0.3
Transition width:	0.21
Passband ripple, R_p:	$5.23 * 10^{-3}$
Stopband ripple, R_s:	$2.17 * 10^{-3}$
Passband ripple in dB (peak-to peak):	$9.06 * 10^{-2}$
Stopband attenuation in dB:	-53.3

Decimation-by-2 Filter

Filter length:	137
Passband:	0.45
Stopband:	0.5
Transition width:	0.05
Passband ripple, R_p :	$4.42 * 10^{-4}$
Stopband ripple, R_s :	$1.47 * 10^{-3}$
Passband ripple in dB (peak-to peak):	$7.7 * 10^{-3}$
Stopband attenuation in dB:	-56.6

3.6 The Least-Squares Filter

The least-squares algorithm first is a deterministic, non-adaptive approach to design filters that are optimum in the least squares sense, as discussed in Section 2.1.2. The input to be equalized is the impulse response $q[n]$ of the distortion channel.

where $\delta[n]$ is the impulse, $q[n]$ the distortion channel, and $x[n]$ the input to the equalizer. The desired response is similarly the impulse response of the desired channel.

The least squared error (LSE) is obtained by summing the squares of the difference between the filter output $y[n]$ and the desired output $d[n]$:

$$\text{LSE} = \sum_{i=n}^{n+N-1} (d[i] - y[i])^2 . \tag{3.18}$$

3.6.1 Computational Complexity of the Least-Squares Filter

Single-Stage Filter

As reviewed in Section 2.1.2, the least squares filter requires solving the following matrix equations for the filter weight vector **h**:

$$\mathbf{h} = (X^T X)^{-1}(X^T \mathbf{d}) \tag{3.19}$$

or equivalently

$$\mathbf{h} = A^{-1}\mathbf{b} \tag{3.20}$$

where $A = X^T X$ and $\mathbf{b} = X^T \mathbf{d}$. Let us compute the complexity of designing the filter **h**, which is measured by the number of multiplications to solve (3.19) above.

Let the following denote the lengths of the vectors of interest here:

- N_h : length of filter vector **h**
- N_x : length of input vector **x**
- N_d : length of desired response vector **d**

The matrix X is of size $N_h \times N_x$ which makes A of size $N_h \times N_h$. Each entry in A takes N_x multiplications, and since A is symmetric, forming the matrix A takes a total of

$$\frac{N_x N_h^2}{2} \tag{3.21}$$

multiplications.

To invert the matrix A, we assume we use the standard Gaussian elimination method, which would take approximately on the order of

$$N_h^3 \tag{3.22}$$

multiplications [36]. In practice, however, we would exploit some nice properties of the matrix A resulting from the way we window the input data as discussed in Section 2.1.2 The matrix A is always symmetric and positive-semidefinite whatever the windowing, so its inversion can be accomplished by a Cholesky decomposition, which is more efficient than Gaussian elimination. On the other hand, if the windowing is done by the correlation method, A becomes symmetric and Toeplitz, and the faster Levison recursion can be used for the matrix inversion. However, the autocorrelation windowing method gives only an approximation to the least-squares solution [20].

The vector $\mathbf{b}$ is $N_h \times 1$ long and each entry takes N_x multiplications to compute, which gives a total number of multiplications for $\mathbf{b}$ to be

$$N_x N_h \; . \tag{3.23}$$

Performing the products $A^{-1}\mathbf{b}$ takes

$$N_h^2 \tag{3.24}$$

mutiplications.

Thus, the total number of multiplications required in the time-domain least squares solution is the sum of Equations (??) above, which is

$$N_h^3 + N_h^2(N_x + 1) + N_h N_x \tag{3.25}$$

Multistage Filter

Extending the computational complexity result of the single-stage filter given in (3.25), the number of multiplications in designing the filter with two-stages in Figure 3-3 is

$$N_{h1}^3 + N_{h1}^2(N_{x1} + 1) + N_{h1}N_{x1} + \tag{3.26}$$

$$N_{h2}^3 + N_{h2}^2(N_{x2} + 1) + N_{h2}N_{x2} \; , \tag{3.27}$$

where the subscripts 1 and 2 above indicate the first and second stage, respectively. Note that the input data lengths, N_{x1} and N_{x2} for the first and second stage are different because of the decimation operation in between them.

A generalized equation for computational complexity of designing a filter with I stages using the least-squares method is

$$\sum_{i=1}^{I} N_{hi}^3 + N_{hi}^2(N_{xi} + 1) + N_{hi}N_{xi} \ . \tag{3.28}$$

We can see from the complexity equations that a large filter length N_h will figure dominantly in the number of multiplications of the filter since it is cubed. Thus, if we could have smaller filters in the stages of the multistage structure, then its combined complexity will be less than that of the single-stage structure. Whether this decomposition will yield comparable least-squared error will be shown by the simulation results in Section 3.6.

For the single-stage equalizer, let's normalize the sampling rate of its input to 1 since the input has been downsampled by both decimators. For the two-stage equalizer, the input to the second stage equalizer also has the same sampling rate as that of the single-stage equalizer. However, the input to the first stage equalizer has approximately M_2 times the sample points of the second-stage equalizer since it has only been downsampled by the first decimator (M_2 is the decimation factor of the second decimator).

3.6.2 Results of Time-Domain Least Squares

In the first simulation example using the least-squares method, we fix the effective filter lengths (EFL), as defined in (3.14) and (3.15), for all three system configurations in Figure 3-4 and compare the resulting least squared errors and computational complexities of designing the equalization filters. The selected EFL is 129, and the results of this experiment are summarized in the table in Figure 3-10. For two-stage equalization, the optimum number of taps for the first stage is 11 taps, and 107 taps for the second stage. From (3.14) and (3.15), this is equivalent to the single-stage EFL of 129. The LSE is approximately the same for the three system configurations; they are in the order of 10^{-7} and 10^{-8}. The computational complexity of obtaining the filter is slightly lower for the multistage equalizer than for the other two single-stage systems, about 30% less. This computational saving will no longer be evident if the there are more than 11 taps in the first stage, as we will discuss subsequently. The table also shows the advantage of doing decimation in multiple stages, as evident by the reduction in the length of the digital anti-aliasing filters for the multistage scheme.

	System 1	**System 2**	**System 3**
Effective Filter Length (EFL)	**129**	**129**	**11*2+107=129**
LSE	**1.58*10^(-7)**	**6.41*10^(-8)**	**1.06*10^(-7)**
	1	**0.41**	**0.67**
Computational Complexity of Obtaining Filter	**5.0*10^6**	**5.30*10^6**	**3.49*10^6**
	1	**1.06**	**0.69**
Effective Decimator Length	**527**	**27*2+137=191**	**27*2+137=191**
	1	**0.36**	**0.36**

Figure 3-10: Comparison of three systems using the least-squares filter.

In another experiment, we attempt to answer the question of how best to assign the number of taps to each stage of the two-stage equalizer. Fixing the least-square error to 1×10^{-7}, we vary the number of taps in the first-stage equalizer from 9 to 91 and observe the effect of this on the number of taps in the second stage. Initially, we would expect the length of the second-stage equalizer to decrease as we increase the number of taps in the first stage. However, the simulation results indicate otherwise, as shown in Figure 3-11. The x-axis gives the number of taps assigned to the first stage while the y-axis plots the number of taps required to reach the preset least-squared error. For this particular case, the optimum number of taps appears to be

11, while having more or fewer taps in the first stage does not help the equalization of the second stage, since they all still need around 110 taps to correct to the chosen LSE level. Figure 3-12 shows the effective filter lengths (EFL's) for the different decompositions of the multistage equalizer in Figure 3-11.

Figure 3-11: Study of decomposition of the two-stage equalizer using the least-squares method.

Figure 3-12: EFL for the different decompositions of the two-stage equalizer.

Let's examine this phenomenon in more detail by comparing two extreme examples, 11 taps versus 91 taps in the first stage, as illustrated by Figures 3-13 and

3-14, respectively. After equalization with 11 taps in the first stage, the distortion is reduced from 30 dB to about 20 dB, but the number of alternation points, or ripples, is small. It takes an additional 107 taps in the second stage to correct the distortion to the preset LSE level of 1×10^{-7} . On the other hand, using 91 taps in the first stage reduces the magnitude of the distortion tremendously, down to less than 1 dB, but it also adds many ripples to the distortion signal. Correcting the remaining distortion to specification still requires a large number of taps, 113. Increasing the taps in the first stage certainly reduces the magnitude of the passband ripple of the distortion for the second stage but it also adds more ripples, which require more taps to correct.

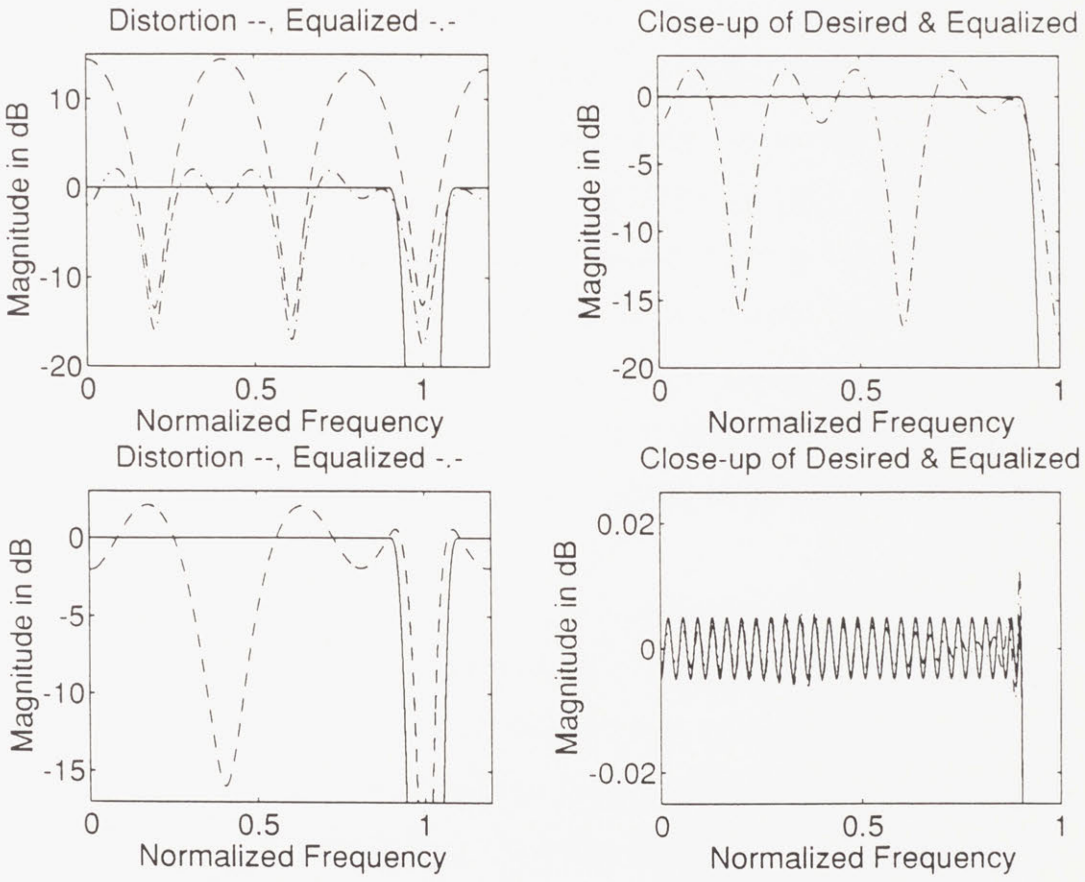


Figure 3-13: Magnitude responses of the two-stage equalizer with 11 taps in the first stage.

To ascertain that our observation above is not peculiar to the small LSE level of 10^{-7} that we chose, we perform another experiment where we assign 51 taps to the first equalizer and note the final LSE as we vary the taps in the second equalizer. Figure 3-15 plots the LSE versus the number of taps in the second equalizer. The LSE decreases as we increase the number of taps, as expected, but it does not reach 10^{-7} until we get to 110 taps. Thus, quantization error is not an issue here as demonstrated by the experiment and by the fact that we use 16 bits for double-precision arithmetic.

From the observations, we can conclude for this case that there is a good choice of number of taps for the first equalization stage, and that this number is usually small. Its function is mainly to perform rough equalization of the distortion while the second stage equalizer performs fine correction. A higher filter length would correct the magnitude of the distortion in the first stage better but it also adds additional zeros to the output which would require a longer equalizer in the second stage. Furthermore, fine equalization in the first stage may be wasteful since it does not help reduce the number of equalizer taps in the second stage. This is, in part, due to the filtering by the second decimator, which discards a part of the frequency spectrum of the first equalizer output. In our case, half of the frequency spectrum of the first equalizer output is eliminated by the second stage decimator with factor $M_2 = 2$.

From our studies of the least-squares method using the models we develop, we conclude that the multistage, multirate equalizer offers no major advantages over the single-stage equalizer. This is due mainly to the fact that the the first equalizer is unnecessarily spending many of its taps to correct over the frequency-band which will be discarded by the second-stage decimator. However, one possible advantage of the multistage equalization scheme is that solving the matrix equation (3.19) would require less computation if the lengths of the equalizers are shorter. As the experiments show, finding a decomposition of the equalizer that would yield the least computations for a given least-squared error is not an easy task. Our observations tend to indicate that a shorter filter in the first stage that performs a "rough" correction followed by a longer filter in the second stage that performs fine equalization would

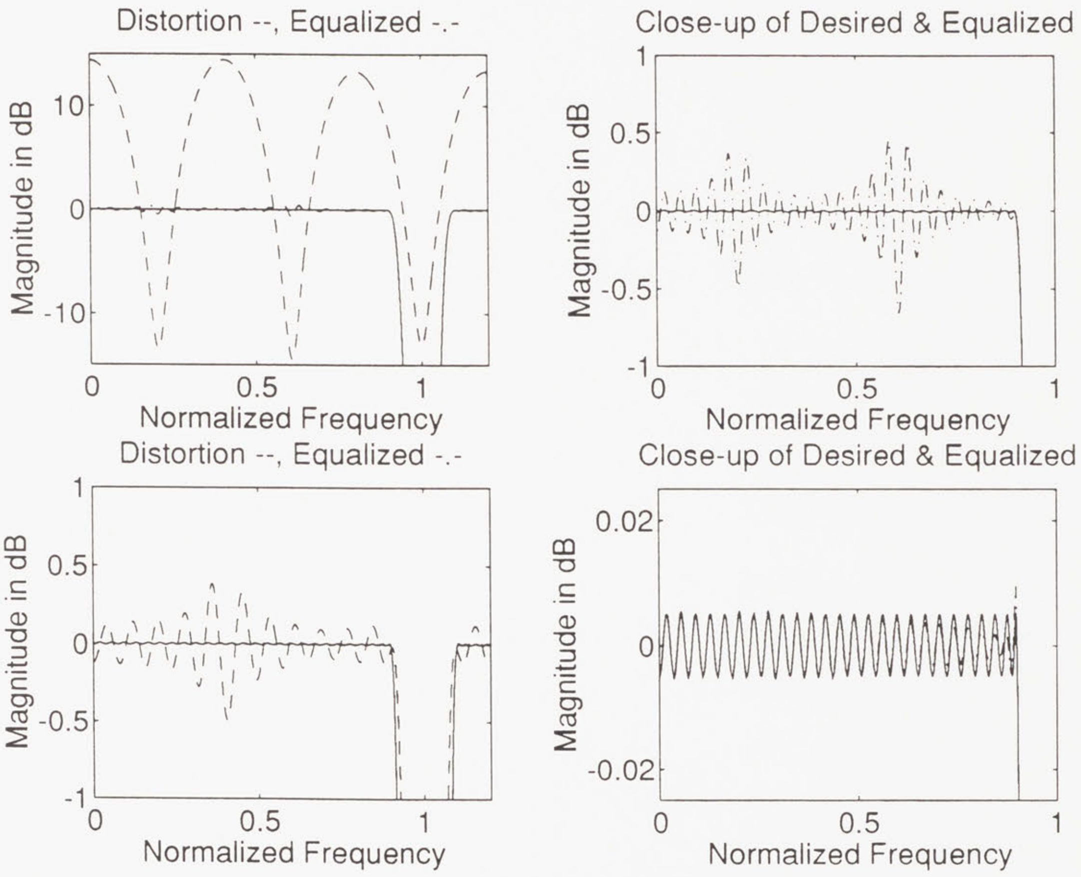


Figure 3-14: Magnitude responses of the two-stage equalizer with 91 taps in the first stage.

Figure 3-15: LSE versus taps in the second-stage equalizer.

yield the best result in terms of least-squared error and computational complexity.

3.7 The FBLMS Filter

The second algorithm we use to study the multistage, multirate equalizer is the frequency-domain block LMS filter discussed in Section 2.1.6. Our initial motivation for using a frequency-domain algorithm is that we can exploit the approximate uncorrelatedness that comes with the transformation of the input signal to selectively adapt only over the frequency band of interest. The time-domain algorithm, on the other hand, corrects indiscriminately over the whole spectrum. This implies that for the multistage equalizer, the first equalizer invests as much effort in correcting the unnecessary frequency band (which will be filtered out by the second stage decimator) as the ultimate band of interest.

Therefore, by performing the adaptation in the frequency domain, we can incorporate a frequency-weighting factor in the transformed error. The weight is a vector of 1's over the frequency-band of interest and 0's over the don't-care band, which essentially is windowing the frequency-domain data over the band of interest. This weighted frequency error is then used in the estimate gradient to update the filter coefficients. However, this idea does not work, as we shall discover, with the FBLMS algorithm. Hence, the work for the no error weighting is the main focus here. Why error weighting does not work for the FBLMS filter will be discussed later. In Section 4.2, we will suggest and discuss another frequency-domain adaptive algorithm where the frequency-error weighting idea might work.

3.7.1 The Setup

A Gaussian white noise with zero mean and unit variance is used to drive the input of the system, so it is represented by the convolution of a white noise $w[n]$ with the distortion channel $q[n]$:

$$x[n] = w[n] * q[n] \ . \tag{3.29}$$

The output of this convolution, $x[n]$, is a discrete-time stochastic process which is used as the input to the adaptive filter. The desired output $d[n]$ is similarly generated by driving the desired channel with the same white noise $w[n]$. An attractive property of white noise is that its spectral density is flat at all frequencies so that all frequencies of the distortion channel can be excited.

To obtain the residual error, we let the adaptation run sufficiently long beyond convergence so that we can estimate the mean square error (MSE) by averaging the error in the last several hundred blocks after convergence. This is based on the assumption that the stochastic process is ergodic whereby a time average of the error $e[n]$ can approximate its ensemble average:

$$E(e^2[n]) = \lim_{N\to\inf} \frac{1}{N} \sum_{n=1}^{N} e^2[n] \tag{3.30}$$

For two-stage adaptive equalization, we let the first stage completely converge first before we adapt the second stage. In practice, it may be more efficient if we can use pipelining, whereby the first stage adapts on a block of data and passes it to the second stage to correct while the first stage goes on to correct the next block of data.

3.7.2 Computational Complexity of the FBLMS Filter

Like the least-squares fiter, there are also two types of computational complexity involved with the FBLMS filter. The first is the total complexity of computing the filter in the adaptation stage, and the second is the complexity of using the filter per unit time, once it has converged and adaptation has been turned off.

Let us determine the computational complexity of adapting the FBLMS filter with gradient constraint. We measure the complexity of the algorithm in terms of the number of multiplications it requires. Refer to Figure 2-7 in Section 2.1.6.

Complexity Per Iteration of Adaptation

First, the number of real multiplications for an M-point FFT (IFFT) is approximately

$$Mlog_2M \tag{3.31}$$

using the radix-2 FFT algorithm.

Each linear convolution or correlation in the FBLMS filter takes $M = 2N_h$ points where N_h is the length of the adaptive filter, as well as the block size. There are a total of 5 FFT's and IFFT's in the overlap-save FBLMS with gradient constraint, so the number of real multiplications to perform these transforms is

$$10N_h log_2(2N_h) \tag{3.32}$$

The computation of each filter output, $Y = XH$, and the gradient, $G = X^H E$, requires $4M$ real multiplications since the DFT's are complex. For $M = 2N$, the total number of real multiplications is $16N$. The total number of multiplications for *each* block of iteration of the FBLMS algorithm is thus

$$10N_h log_2(2N_h) + 16N_h \ . \tag{3.33}$$

This measures the complexity of the FBLMS algorithm.

Total Number of Multiplications in Adaptation Stage

Another comparison of interest is the total number of multiplications required during the adaptation stage. This is the product of the number of multiplications in each iteration and the number of blocks it takes the algorithm to converge.

For a single-stage filter that takes L blocks to converge, the total number of multiplications, abbreviated as TM, during the adaptation stage is:

$$\mathrm{TM} = L[10N_h log_2(2N_h) + 16N_h] \ . \tag{3.34}$$

The total number of multiplications it takes to adapt both equalizers in the multistage structure is

$$\text{TM'} = L_1[10N_{h1}log_2(2N_{h1}) + 16N_{h1}] \tag{3.35}$$

$$+L_2[10N_{h2}log_2(2N_{h2}) + 16N_{h2}] \tag{3.36}$$

where L_1 and L_2 are the numbers of blocks the first and second equalizers take to converge.

Time For Convergence

In practice, the number of input samples, at the original, undecimated rate it takes for the entire filter structure to converge is usually of interest. This gives a measure of the *convergence time* of the entire filter structure, which we shall abbreviated as CT here.

For the single-stage equalizer preceded by a two-stage decimator with factors M_1 and M_2, the convergence time is

$$\text{CT} = M_1M_2LN_h = MLN_h \tag{3.37}$$

for $M = M_1M_2$ and L is the number of blocks the filter takes to converge. We must include the sampling rate here since we downsample the original input signal $i[n]$ of the system. Although each FFT and IFFT is $2N_h$ long, the number of new input data, after decimation, that comes into the equalizer at each iteration is N_h, which is also the length of the equalizer.

For the multistage equalizer, the convergence time for the filter in the first stage, CT$_1$, with respect to the original sampling rate is

$$CT_1 = M_1L_1N_{h1} \tag{3.38}$$

and that of the second stage is

$$CT_2 = M_1M_2L_2N_{h2}\ . \tag{3.39}$$

The total convergence time, CT', of the two-stage adaptive filter is the sum of (3.38) and (3.39)

$$CT' = M_1 L_1 N_{h1} + M_1 M_2 L_2 N_{h2} \tag{3.40}$$

Effective Filter Length

The effective filter length (EFL) for the FBLMS filter is the same as that defined in (3.14) and (3.15). It meaures the complexity of using the filter after adaptation is turned off. Again, the EFL of the single-stage structure is N_h while that of the two-stage structure is

$$\mathrm{EFL} = M_2 N_{h1} + N_{h2}\ . \tag{3.41}$$

3.7.3 Results of FBLMS with Gradient Constraint

To compare the performance of the multistage adaptive filter with that of the single-stage adaptive filter, let us examine the results of three sets of experiments. Here, we fix the total number of equalizer length for both the single-stage and the multistage adaptive filters, and compare their peformances based on:

- Mean-squared error (MSE)
- Total number of multiplications during adaptation(TM)
- Number of multiplications per unit time (MPT)
- Convergence Time (CT)
- Effective Filter Length (EFL)

The three lengths we chose for the single-stage filter in the three sets of experiments are 64, 128, and 192. For filter lengths lower than 32, the algorithm shows poor convergence for the data we have, and for filter lengths that are too high we would be equalizing beyond the desired specification. The table in Figure 3-16 indicates how we decompose the filter of the multistage structure for the chosen filter lengths.

	Setup 1	Setup 2	Setup 3
Singlestage	64	128	192
Multistage	32:32	32:128	64:128
		64:64	

Figure 3-16: Three setups for simulation study.

The complexity of the algorithm is calculated based on the assumption that the filter length N_h is a power of 2 so we can use fast radix-2 algorithm for the FFT. Therefore, for filter lengths in the table above that are not powers of 2, such as 96 and 192 in the table, we have to pad the filters with zeros to obtain the next highest power of 2 before the FFT is perforemd.

The table in 3-17 summarizes the results of the three setups. Note that we also express the results in terms of ratio as follows:

$$\text{Performance Ratio} = \frac{\text{Multistage Performance}}{\text{Single-stage Performance}} \tag{3.42}$$

to ease the comparison between the two filter structures.

First, let us look at the number of blocks it takes for each individual filter to coverge. The simulations show that each of the filters in the multistage filter takes fewer updates to converge than the single-stage equalizer alone. This is due to the fact that each of them have shorter lengths.

Let us look at the convergence plots of two filters in Setup 2: the single-stage filter with 128 taps and the two-stage filter with 64 taps in both stages. Figure 3-18 (a) displays the convergence plot of the single-stage, and Figures 3-19 3-20 a) pertain to the two equalizers of the two-stage structure. The x-axis gives the number of blocks, or iterations, in the adaptation of the FBLMS filter. The y-axis shows the MSE in dB. We define the convergence point as the block number where the convergence curve has steadied out to approximately a flat line. The convergence curve shown here is only one realization of the random process and thus it is noisy. The noise is due to the fact that the gradient has to be estimated at each iteration as well as

	Setup 1		Setup 2			Setup 3	
	Single	Multistage	Single	Multistage 1	Mulitstage 2	Single	Mulitstage
Filter Length	64	32:32	128	32:96	64:64	192	64:128
# Blocks to Converge	100	80:40	300	80:50	50:50	350	50:200
MSE	3.7679e-06	6.7024e-05	2.4349e-07	5.8954e-05	8.5281e-07	9.3124e-09	1.0815e-08
MSE Ratio	1	17.788158	1	242.12083	3.5024436	1	1.1613548
Total Mult (TM)	550400	291840	3686400	808960	550400	9497600	2732800
TM Ratio	1	0.53023256	1	0.21944444	0.14930556	1	0.28773585
Converge Time (CT)	64000	25600	384000	76800	48000	896000	272000
CT Ratio	1	0.75	1	0.2	0.125	1	0.3035
EFL	64	192	128	165	192	192	256

Figure 3-17: Summary of the simulation results for the FBLMS algorithm.

to numerical rounding. Thus, the solution tends to bounce back and forth in the performance surface, never reaching the minimum. Averaging the covergence plots for many different runs would give a smoother convergence curve. We can see that the single-stage filter takes around 300 blocks to converge, whereas the first and second stage of the multistage filter each takes approximately 50 blocks.

Although each of the filters in the multistage structure converges faster than the single-stage filter, each one converges to a higher MSE than the single-stage one as shown in the convergence plot. Specifically, the 64:64-filter-pair converges to an overall MSE of 8.5×10^{-7} but the 128-filter has an MSE of 2.4×10^{-7}, which is about one third less than that of the multistage filter. For other setups that we studied, the single-stage filter consistently converges to the lowest MSE, as shown in the table. It is interesting to observe that the MSE of the multistage structure is rather large when we use 32 taps in the first stage. For instance, the 32:96-filter in Setup 2 has an MSE that is about 240 times larger than that of the 128-filter.

The total number of multiplications (TM) for obtaining a converged filter as defined in (3.34) is consistently lower for all the multistage structures studied here, as can be seen in the table. They range from 15% to 50% of the TM of the single-stage filters. Since the total number of multiplications is a function of the equalizer length as well as the number of blocks for convergence, the multistage structure wins in both of those categories, which explains their lower TM.

The convergence time (CT) is measured by the number of original, undecimated input samples that it takes for the filter to converge, and it is given by (3.37) and (3.40). Again, the multistage structure has consistently low CT's for all the Setups, which again can be explained by their short filter lengths and low number of blocks for convergence.

Finally, we are also interested in the complexity of using the filter once it has converged. This is measured by the effective filter length (EFL) defined in (3.14). Recall that the EFL is a function of the filter lengths plus the sampling rate that it operates at. In this category, the multistage filters does not fare well mainly because the first equalizer has to run at M_2 or twice the sampling rate of the single-stage

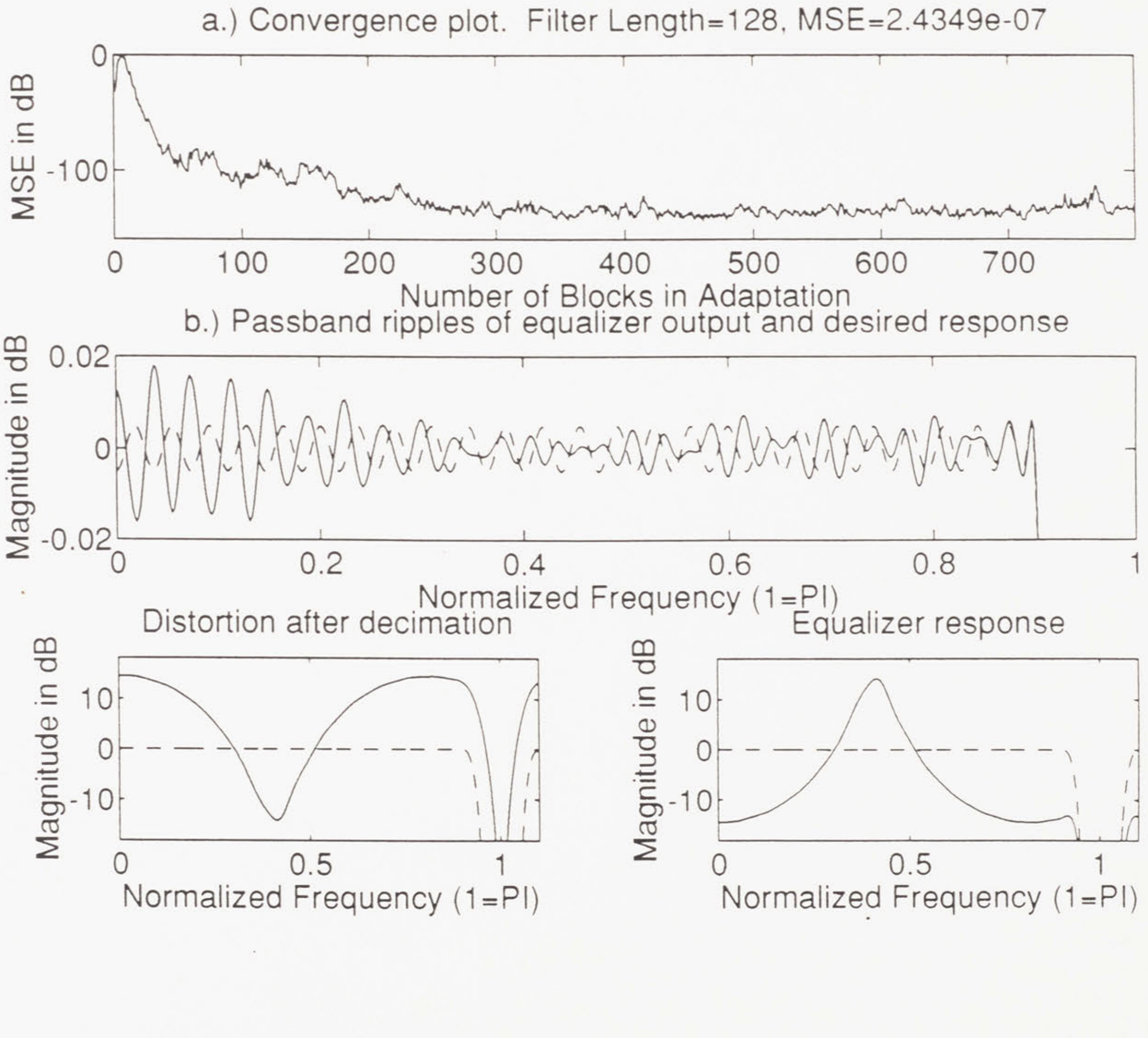


Figure 3-18: Result of single-stage adaptive FBLMS filter with 128 taps.

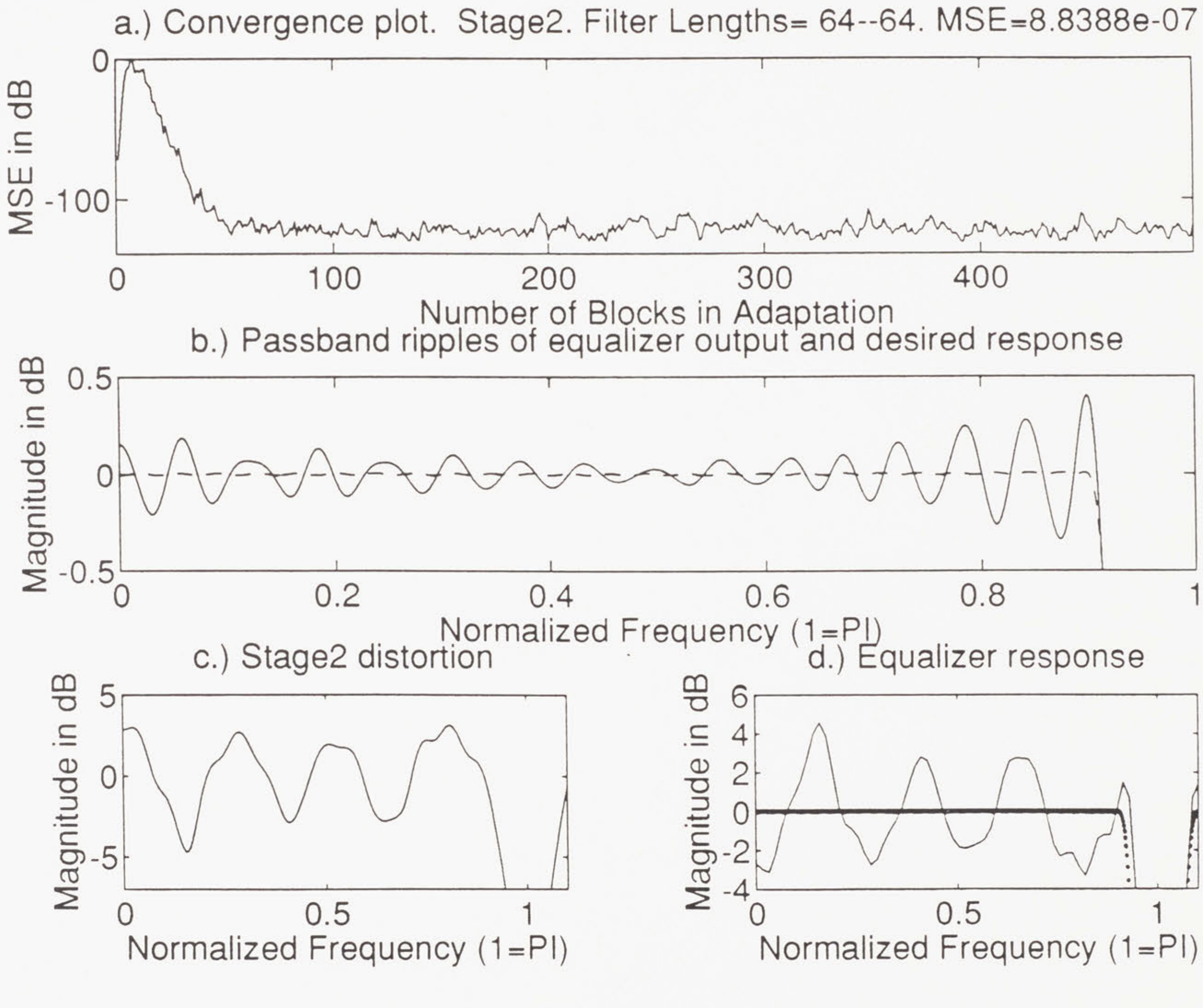


Figure 3-19: Result of stage 1 in two-stage adaptive FBLMS filters with 64 taps each.

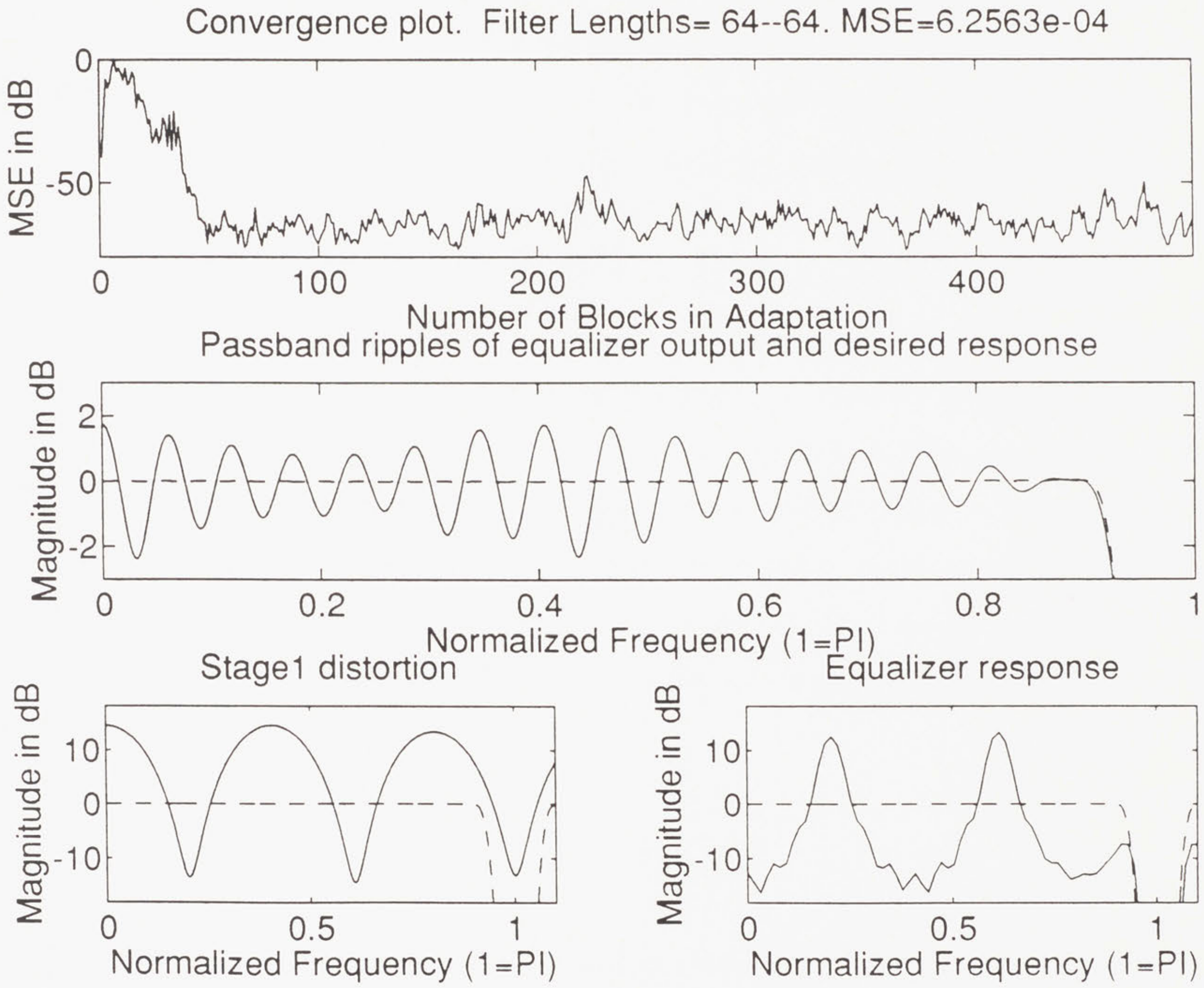


Figure 3-20: Result of stage 2 in two-stage adaptive FBLMS filters with 64 taps each.

equalizer.

In summary, we have studied the multistage and single-stage adaptive filters, where adaptation is done using the FBLMS algorithm. We have three setups with total filter lengths of 64, 128, and 192. We compare the two filter structures by several benchmarks as summarized by the table in Figure 3-17. From our particular model of the systems and parameters, we see that the multistage adaptive filter consistently fares well in terms of the time and computational complexity during adaptation. This is mainly because each of the filter is smaller than the single-stage filter and thus this allows for faster convergence. However, the faster convergence of the multistage structure also leads to larger mean square errors (MSE) than that of the MSE. In our experiments, we fix the step size μ to 1 for all filters. Although the eigenvalue spread of the autocorrelation matrix of the input signal is normalized by the estimated power of the input blocks, this normalization might not be perfect. The μ controls the rate of convergence of the algorithm. In the multistage adaptive filter, we can have multiple μ's for the stages and thus more freedom in choosing the convergence rates of the adaptive filters than that of the single-stage structure. However, in practice, we might not have the luxury of knowing the convergence behaviors of the intermediate stages.

3.8 The Modified FBLMS Filter

As mentioned in the introduction to this section, our initial motive for exploring adaptation in the frequency-domain is that we hope to exploit the approximate uncorrelatedness of the transformed signal in our multistage, multirate adaptive filter structure. In particular, we want to selectively adapt only over the frequency band of interest and not over the entire spectrum of the signal. This is especially useful in the first-stage equalizer since it is wasteful for it to correct the frequency band outside of π/M, where M is the ultimate decimation factor, which will be discarded by the second decimator anyway. Thus, we can incorporate a weighting function in the frequency-domain error which would specify which frequency bins we want adaptation and which ones we do not. We explore this idea using the FBLMS filter, but it does not work as we have hoped and we shall see why. We will also suggest a class of algorithms where the error-weighting idea might perform correctly.

Refer once again to Figure 2-7 to review the block diagram of the FBLMS filter using overlap-save data sectioning with gradient constraint. We want to weight the transformed error vector,

$$\mathbf{E} = [E_0 E_1 ... E_{2N-1}]^T \tag{3.43}$$

with a weight matrix $\mathbf{W}$ of 1's over the frequency band of interest and 0's over the don't-care band. A weight matrix is defined as

$$W = diag[11...00...11] \tag{3.44}$$

where diag[$\cdot$] is an operator that forms a diagonal matrix. The components on the diagonal of W must be symmetric to be consistent with $\mathbf{E}$, which is conjugate symmetric because it is the DFT of a real time-domain error. The weighting is done by a matrix multiplication, $W\mathbf{E}$. This weighted error vector is then used in the calculation of the gradient vector $\mathbf{G}$ as in the original FBLMS algorithm.

The simulation of this modified FBLMS algorithm yields inconsistent results which indicate that the modification has altered the filter in an incorrect way. We have an explanation for why this error weighting does not work. In general, weighting the error $\mathbf{E}$ with the weight matrix W is like truncating the the frequency domain error with a perfect rectangular window whose passband is 1 in the band of interest and 0 in the don't-care band. In the time domain, this operation is equivalent to a circular convolution of the time domain error $\mathbf{e}$ with a sinc function:

$$W\mathbf{E} \quad \longleftrightarrow \quad \frac{sin\omega_c n}{\pi n} \circledast \mathbf{e}, \tag{3.45}$$

where ω_c is the cutoff frequency of the window W. Because of this circular convolution, the weighted error is a "smeared" version of the original error, and such modification might not be desirable since the algorithm now tries to minimize a function of a different error. Thus, the steady state solution is entirely different from that of the unweighted error.

However, the fundamental problem underlying the error weighting idea resides in the error property of the FBLMS filter itself. Each frequency bin in the FBLMS filter does not have its own individual error term, since the error is derived in the time domain. The frequency transformation of the error in the algorithm is only necessary because we want to use the Fast Fourier Transform (FFT) to realize the linear correlation needed in the time-domain algorithm. Hence, the mean-squared error (MSE) that the FBLMS tries to minimize is global, so weighting the frequency-domain error will not make the algorithm adapt selectively over the band of interest. However, it is still true, by Parseval's theorem, that minimizing the error energy in the time domain would be equivalent to minimizing the error energy in the frequency domain, which is achieved by the FBLMS filter, as pointed out in Section 2.1.6. Thus, the FBLMS filter should be viewed more as an efficient implementation of the time-domain block LMS algorithm, rather than an algorithm with a frequency-domain error. These subtle attributes of the FBLMS algorithm were initially not clear to us until we explored the error weighting idea.

We believe that the frequency error weighting idea might work better for the class of adaptive filters that compute and use the frequency-domain error directly in the adaptation. The circular convolution algorithm, mentioned in Section 2.1.7, is such an algorithm since the error is found directly from the difference of the transformed filter output and the transformed desired respone, $\mathbf{E} = \mathbf{Y} - \mathbf{D}$. Here, each frequency bin has its own local MSE to minimize instead of the transform of a global MSE as in the FBLMS filter. However, because the convolution is circular in this algorithm, the performance is degraded and it might not perform well for certain types of signal. In this thesis, we did try to use the circular convolution algorithm but the degradation in performance is not tolerable here.

Another place where error weighting might perform well is adaptive filtering in subbands, as illustrated in Figure 3-21. As introduced in Section 1.2, in this scheme an analysis filter bank splits the input signal $x[n]$ into smaller frequency bands called subbands, and each subband can be downsampled by some factor M. The subbands of the desired response $d[n]$ are similarly obtained. The adaptive filter then tries to adapt over each subband and produces, at the kth iteration, a vector of subband outputs, $\mathbf{Y}$. Thus, each subband has its own error term, and the error vector for all the subbands is $\mathbf{E} = \mathbf{Y} - \mathbf{D}$. Now, we can sensibly incorporate the weighting factor W in 3.44 into the frequency error vector, which is $W\mathbf{E}$.

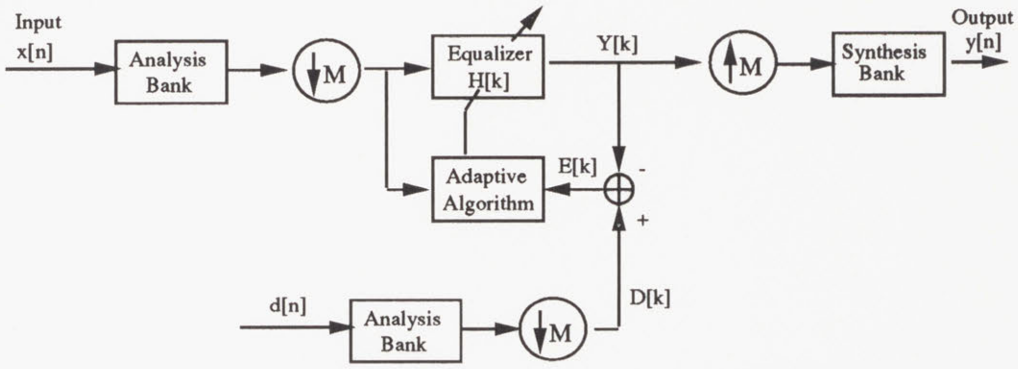


Figure 3-21: Block diagram of adaptive filtering in subbands.

Figure 3-21 shows the general structure of a single-stage adaptive filter in subbands, where the output of the adaptive filter must be reconstructed from the subbands by upsampling and filtering it with a synthesis bank. There are many issues involved in the design of the filter banks and the adaptive filter in the subband schemes as mentioned in Section 3.1. These issues are dealt with in great detail in [32], [37], [16, 17, 18], [40].

What we have presented here is only a rough sketch of a possible improvement in the performance of the multistage, multirate adaptive filter by incorporating an error weighting factor in the frequency-domain adaptation. From our study, we have learned and understood why this idea does not work for the FBLMS algorithm. We then propose that the error weighting idea may be successfully applied to a class of algorithms that minimizes the frequency mean-squared error directly. The circular convolution adaptive filter and the subband adaptive filter belong to such a class.

Chapter 4

Conclusion

4.1 Summary

In this thesis we have proposed a modification to the current structure of the adaptive filter that operates in a multirate system. Specifically, we decompose the single-stage adaptive filter into smaller ones and insert them in between successive decimation stages. We explore this multistage, multirate adaptive filter in the context of adaptive equalization of a downsampled, narrow-band signal. We have investigated a simple decomposition, that of a two-stage equalizer operating at two different sampling rates. We construct a hypothetical model for such equalization system, and study its performance for two filter-design techniques: 1) the classic least-squares method and 2) the frequency-domain block least mean square (FBLMS) algorithm. With the aid of software simulations, we compare the multistage, multirate filter with the single-stage structure based on two performance criteria: 1) the residual error and 2) the computational complexities of the filter design and operation.

In Chapter 2, we cover some fundamental backgrounds on adaptive filtering and multirate digital signal processing. We first review the least-squares method, which is a classic deterministic method for data-fitting. Its exploitation of the orthogonality principle to minimize the error function is a powerful and fundamental concept, and is also the basis of the probabilistic solution, the Wiener filter. Although the Wiener filter is optimum in the mean-square sense, it assumes perfect knowledge of the signal statistics, which is not available in practice. Efforts to relax this assumption led to the development of adaptive filters. We then discuss one of the simplest and most widely used adaptive algorithms, the least mean square (LMS) method. The LMS is an iterative approach to compute the Wiener solution by using instantaneous estimates of the gradient of the error performance surface to descend toward the optimum solution. A variation of the LMS algorithm is to estimate the gradient from a block of input and error data which allows for a smoother convergence of the algorithm. This method is called the time-domain block LMS (TBLMS). These preludes then prepare us for the discussion of the frequency-domain block LMS (FBLMS) algorithm, which is used in this thesis. This adaptive filter is an efficient implementation of the time-domain block LMS by using fast Fourier transforms (FFT's) to compute the linear convolution and correlation needed in the TBLMS filter. Another of its nice attributes is that the Fourier transform tends to decorrelate the signals so that a different step size can be used for each adaptive weight to give a more uniform convergence rate

In Chapter 3, we describe the proposed multistage, multirate adaptive filter and discuss some of the difficulties inherent in the analysis of this system, which lead us to study the problem using simulations. We model the system as an equalization problem of a narrow band signal. For the single-stage structure, the system has two decimators followed by a single-stage equalizer. The multistage structure, on the other hand, has two equalizers, one of which is inserted in between the two stages. We compare the performances of these two structures for the least squares method and the frequency-domain block LMS algorithm that we used in the simulations. The performance criteria are: 1) residual error and 2) the computational complexities of the design of the filter and of using the filter. Our study shows that for our particular model of the system, the multistage, multirate scheme shows no clear improvement over the single-stage filter for the non-adaptive least-squares method. For the adaptive FBLMS filter, the multistage structure demonstrates significant computational savings over the single-stage filter but at a cost of an increased residual error at convergence. In both cases, the proposed scheme also adds complexity to the design of the filter since it is not clear what is the best allocation of the number of taps to each stage of the filters to optimize the convergence rate and the residual error.

4.2 Discussion and Suggestions for Future Research

In this section, we will discuss various points that have not been dealt with in the thesis and suggest some future works.

1. In our study of the FBLMS filter, we simplify the problem by letting the first adaptive filter to converge before starting to adapt the second filter. In real situations, it might be more practical to let both filters adapt simultaneously by pipelining the blocks of input data. However, it is not clear whether doing so would guarantee convergence of the two adaptive equalizers.

2. In our current sheme, we may be unnecessarily over-optimizing the first-stage equalizer by matching its output with the same desired response used for the second stage. Since the ultimate desired response has small ripples, the first equalizer will have to work harder with the relatively few number of taps that it has. We can relax the work load of the first stage equalizer by having a different desired signal that has bigger ripples than the final desire response. However, in practice we might not know what the result of the intermediate corrections will be, which makes it difficult to determine the intermediate desired response.

3. The first-stage equalizer is correcting the distortion not only over the ultimate passband, but also over the don't-care band which will be discarded by the anti-aliasing filter of the decimator. We have attempted to improve on this situation by incorporating a weighting factor to the frequency-domain error vector, which specifies the frequency bands where adaptation should occur. However, we have discovered and understood why this error weighting does not work with the FBLMS algorithm as discussed in Section 3.8. We also suggest applying the error weighting to another class of algorithms where each frequency bin actually has its own error term. Examples are the circular convolution algorithm in Section 2.1.7 and adaptive filtering in subbands in Section 3.8.

4. The FBLMS filter comes from the LMS algorithm which uses a stochastic gradient approach to estimate the Wiener solution. The minimum mean-squared error criterion of the Wiener filter guarantees the designed filter to be optimum when the the distortion channel is fixed over time. However, when the channel is time-varying, the position of the minimum point on the error performance surface corresponding to the optimum filter weights, $\mathbf{h}^*$, is no longer fixed. The LMS algorithm has been shown to have the capability to track the location of this changing minimum point in a nonstationary environment [21]. In such case, the filter must continuously update its coefficients. We did not deal with the issue of nonstationary in this thesis, but it is a realistic and crucial problem confronting many applications in communications.

In summary, what we have presented in this thesis is a possible approach to study the multistage, multirate adaptive filter that we propose. In the course of understanding this new filter structure, we have raised more questions than answers, and further study and analysis are necessary to confirm the properties and merits of this multistage, multirage filter.

Bibliography

[1] S. T. Alexander, *Adaptive Signal Processing*, Springer-Verlag, 1986.

[2] H. Baher, *Analog & Digital Signal Processing.* John Wiley & Sons, 1990.

[3] M. G. Bellanger et al., "Interpolation, Extrapolation, and Reduction of Computation Speed in Digital Filters," *IEEE Trans. ASSP.*, vol. ASSP-22, no. 4, pp. 231-235, Aug. 1974.

[4] M. G. Bellanger et al., "Digital Filtering by Polyphase Network: Application to Sample-Rate Alteration and Filter Banks," *IEEE Trans. ASSP.*, vol. ASSP-24, no. 2, pp. 109-114, Apr. 1976.

[5] M. G. Bellanger, "Computation Rate and Storage Estimation in Multirate Digital Filtering with Half-Band Filters," *IEEE Trans. ASSP.*, vol. ASSP-25, no. 4, pp. 344-346, Aug. 1977.

[6] M. G. Bellanger, "New Applications of Digital Signal Processing in Communications," *IEEE ASSP Magazine*, pp. 6-11, Jul. 1986.

[7] M. G. Bellanger, *Digital Processing of Signal: Theory and Practice.* John Wiley & Sons, 1984.

[8] N. Bershad and P.L, Feintuch, "A Normalized Frequency Domain LMS Adaptive Algorithm ", *IEEE Trans. ASSP.*, vol. ASSP-34, No.3, pp. 452-461, June 1986.

[9] G. Clark, S. Mitra and S. Parker, "Block Implementation of Adaptive Digital Filters" *IEEE TRans. CAS*, vol. CAS-28, No. 6, pp. 584-592, June, 1981.

[10] G. Clark, S. Parker and S. Mitra, "A Unified Approach to Time- and Frequency-Domain Realization of FIR Adaptive Digital Filters", *IEEE TRans. ASSP*, vol. ASSP-31, No. 5, pp.1073-1083, Oct., 1983.

[11] R.E. Crochiere and L.R. Rabiner, *Multirate Digital Signal Processing.* Prentice-Hall, 1983.

[12] R.E. Crochiere and L.R. Rabiner, "A Program for Multistage Decimation, Interpolation, and Narrow-Band Filtering," *Programs for Digital Signal Processing.* IEEE Press, 1979.

[13] M. Dentino, J. McCool and B. Widrow, " Adaptive Filtering in the Frequency Domain", *Proc. of the IEEE*, vol. 66, No. 12, pp. 1658-59, Dec. 1978.

[14] P.M. Embree and B. Kimble, *C Language Algorithm for Digital Signal Processing.* Prentice-Hall, 1991.

[15] , E.E. Ferrara, "Fast Implementation of LMS Adaptive Filters", *IEEE Trans. ASSP.* vol. 28, no.4, pp. 474-475, Aug. 1980.

[16] A. Gilloire, "Experiments with Subband Acoustic Echo Cancellers for Teleconferencing," in *Proc. IEEE ICASSP'87* (Dallas, TX), pp. 2141-2144.

[17] A. Gilloire and M. Vetterli, "Adaptive Filtering in Subbands," in *Proc. IEEE ICASSP'88* (New York, NY), pp. 1572-1575.

[18] A. Gilloire and M. Vetterli, "Adaptive Filtering in Subbands with Critical Sampling: Analysis, Experiments, and Application to Acoustic Echo Cancellation," *IEEE Trans. on Signal Process.*, vol. 40, no. 8, pp. 1862-1875, Aug. 1992.

[19] R. D. Gitlin and S. B. Weinstein, "Fractionally-Spaced Equalization: An Improved Digital Transversal Equalizer," *The Bell System Technical Journal*, vol. 60, no. 2, pp. 275-296, Feb. 1961.

[20] R. A. Haddad and T. W. Parsons, *Digital Signal Processing. Theory, Applications, and Hardware.* W.H. Freeman and Company, 1991.

[21] S. Haykin, *Adaptive Filter Theory.* Prentice-Hall, 1986.

[22] M. L. Honig and D. G. Messerschmitt, *Adaptive Filter: Structures, Algorithms, and Applications.* Kluwer Academic Publishers, 1984.

[23] N.K. Jablon, "On the Complexity of Frequency-Domain Adaptive Filtering", *IEEE Trans. ASSP*, vol.39, no.10, pp. 2331-34.

[24] W. Kellermann, "Analysis and Design of Multirate Systems for Cancellation of Acoustical Echoes," *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing.* New York, NY, pp. 2570-2573, Apr. 1988.

[25] J. C. Lee and C. K. Un, "Block Realization of Multirate Adaptive Digital Filters," *IEEE Trans. ASSP*, vol. ASSP-34, no. 1, pp. 105-117, Feb. 1986.

[26] J. S. Lim and A. V. Oppenheim ed., *Advanced Topics in Signal Processing.* Prentice-Hall, 1988.

[27] D. Mansour and A. Gray, "Unconstrained Frequency-Domain Adaptive Filter", *IEEE Trans. ASSP.*, vol. ASSP-30, No.5, pp. 726-734, Oct. 1982.

[28] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing.* Prentice-Hall, 1989.

[29] F. Reed and P. Feintuch, "A Comparison of LMS Adaptive Cancellers Implemented in the Frequency Domain and the Time Domain", *IEEE Trans. CAS*, vol. 28, no.6, pp. 610-615, June 1981.

[30] T. Saramaki et al., "Design of Computationally Efficient Interpolated FIR Filters," *IEEE Trans. on Circuits and Systems*, vol. 35, no. 1, pp. 70-87, Jan. 1988.

[31] R. R. Shively, "On Multistage FIR Filters with Decimation," *IEEE Trans. ASSP*, vol. ASSP-23, no. 4, pp. 353-357, Aug. 1975.

[32] J. Shynk, "Frequency-Domain and Multirate Adaptive Filtering," *IEEE Signal Processing Magazine*, pp. 14-37, Jan. 1992.

[33] W. M. Siebert, *Circuits, Signals, and Systems.* The MIT Press, 1986.

[34] P. Sommen, P. van Gerwen, H. Kotmans and A. Janssen, "Convergence Analysis of a Frequency-Domain Adaptive Filter with Exponential Power Averaging and Generalized Window Function", *IEEE Trans. CAS*, vol. CAS-34, No. 7, pp. 788-798, July 1987.

[35] S. D. Stearns and Don R. Hush, *Digital Signal Analysis.* Prentice Hall, 1990.

[36] G. Strang, *Linear Algebra and its Application*, Third Edition, 1988.

[37] V. S. Somayazulu, "Adaptive Filtering in Subbands," *Ph.D. Thesis at U. C., Santa Barbara.* October, 1990.

[38] V. Sathe, "Multirate Adaptive Filtering Algorithms: Analysis and Applications," *Ph.D. Thesis at California Institute of Technology.* May, 1991.

[39] V. Sathe and P. P. Vaidyanathan, "Efficient Adaptive Identification and Equalization of Bandlimited Channels Using Multirate/Multistage FIR Filters," *Conference Record of the Twenty-Fourth Asilomar Conference on Signals, Systems and Computers.* Pacific Grove, CA. November, 1990.

[40] P. P. Vaidyanathan, "Multirate Digital Filters, Filter Banks, Polyphase Networks, and Applications: A Tutorial,“ *Proc. of the IEEE*, vol. 78, no. 1, pp.56-93.

[41] G. Ungerboeck, "Fractional Tap-Spacing Equalizer and Consequences for Clock Recovery in Data Modems," *IEEE Trans. on Communications*, vol. COM-24, no. 8, pp. 856-864, Aug. 1976.

[42] R. B. Wallace and R. A. Goubran, "Noise Cancellation Using Parallel Adaptive Filters," *IEEE Trans. on Circuits and Systems*, vol. 39, no. 4, pp.239-243.

[43] T. Walzman and M. Schwartz, "Automatic Equalization Using the Discrete Frequency Domain", *IEEE Trans. IT*, vol. IT-19, No. 1, pp. 59-68, Jan. 1973.

[44] B. Widrow and S.D. Stearns, *Adaptive Signal Processing.* Prentice-Hall, 1985.