Spring 1994

On issues of equalization with the decorrelation algorithm : fast converging structures and finite-precision

Andrew James Bateman
New Jersey Institute of Technology

ABSTRACT

ON ISSUES OF EQUALIZATION WITH THE DECORRELATION ALGORITHM: FAST CONVERGING STRUCTURES AND FINITE-PRECISION

by
Andrew James Bateman

To increase the rate of convergence of the blind, adaptive, decision feedback equalizer based on the decorrelation criterion, structures have been proposed which dramatically increase the complexity of the equalizer. The complexity of an algorithm has a direct bearing on the cost of implementing the algorithm in either hardware or software. In this thesis, more computationally efficient structures, based on the fast transversal filter and lattice algorithms, are proposed for the decorrelation algorithm which maintain the high rate of convergence of the more complex algorithms. Furthermore, the performance of the decorrelation algorithm in a finite-precision environment will be studied and compared to the widely used LMS algorithm.

ON ISSUES OF EQUALIZATION WITH THE DECORRELATION ALGORITHM: FAST CONVERGING STRUCTURES AND FINITE-PRECISION

by
Andrew James Bateman

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

May 1994

APPROVAL PAGE

ON ISSUES OF EQUALIZATION WITH THE DECORRELATION ALGORITHM: FAST CONVERGING STRUCTURES AND FINITE-PRECISION

Andrew James Bateman

Dr. Yeheskel Bar-Ness, Thesis Advisor — Date
Director of the Center for Communications and Signal Processing Research
Distinguished Professor of Electrical and Computer Engineering,
NJIT

Dr. Alexander Haimovich, Committee Member — Date
Associate Professor of Electrical and Computer Engineering,
NJIT

Dr. Zoran Siveski, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering,
NJIT

BIOGRAPHICAL SKETCH

Author: Andrew James Bateman

Degree: Master of Science in Electrical Engineering

Date: May 1994

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1994

- Bachelor of Science in Electrical Engineering,
  Cornell University, Ithaca, NY, 1990

Major: Electrical Engineering

Presentations and Publications:

A.J. Bateman, Y. Bar-Ness, and R.E. Kamel, "Decorrelation Algorithm for Blind Decision Feedback Equalizer with Lattice Structures," to be presented at *Sixth Digital Signal Processing Workshop*, Yosemite Lodge, Yosemite National Park, California, October 1994.

This thesis is dedicated to
my parents, Ken and Jane,
without whose unconditional love and support
this, and all my work, would not be possible

ACKNOWLEDGMENT

The author wishes to express his sincere gratitude to his advisor, Dr. Yeheskel Bar-Ness, for his guidance, help, and consultation throughout this research. I have learned much from his knowledge and expertise.

Extended thanks to Dr. Alexander Haimovich and Dr. Zoran Siveski for serving as members of the thesis committee.

Special thanks to Lisa Fitton for providing timely help and suggestions during the editing of the thesis. Gratitude is also due to Dr. Joseph Strano, Dr. Kenneth Sohn, and Ms. Brenda Walker of the Electrical and Computer Engineering Department at NJIT for providing funding, as well as moral support, during the last few years.

The author deeply appreciates the love and continuous encouragement from his parents, Ken and Jane, his brother, Mark, and his aunt and uncle, Jean and Tom Seery, as well as the rest of his extended family.

Special thanks to the staff and management of Star Semiconductor for the donation, and subsequent upgrade, of the SPROClab Development System, on which much of this work is based. Special thanks to Mr. Frank Sgmatto, Senior Applications Engineer, for his timely and consistent help and advice on the intracacies of working with the SPROC system.

Finally, the author wishes to acknowledge his colleagues at the Center for Communications and Signal Processing Research at NJIT, especially Raafat Edward Kamel, whose friendship, support, and remarkable patience were invaluable throughout this research.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

To transmit data over conventional telephone lines, many systems and devices convert the data into symbols for transmission at the signaling rate. In pulse modulation systems, such as pulse-amplitude modulation (PAM) and quadrature-amplitude modulation (QAM), each symbol corresponds to a different amplitude level. Time dispersion results when the frequency characteristics of the channel deviate from the ideal of constant amplitude and constant delay (linear phase) [38]. In both PAM and QAM systems, transmission over time-dispersive channels causes each pulse to extend beyond the time interval used to represent that symbol. The distortion that results from this overlap is called intersymbol interference (ISI). ISI can also be caused by multipath propagation in radio or undersea channels. Multipath propagation can be viewed as the transmission through a group of channels with differing relative amplitudes and delays [38]. This distortion is one of the major obstacles to reliable high-speed data transmission over low background noise channels of limited bandwidth. Therefore, it is necessary to devise structures which effectively remove the ISI from the incoming signal. Devices which perform such a filtering task are referred to as equalizers.

These PAM- and QAM-based systems must effectively transmit data through communication channels that have different frequency response characteristics and, hence, result in different distortion effects. In such transmission systems, the coefficients of the channel equalization filter cannot be specified *a priori* since the statistical characteristics of the signals to be filtered are either unknown or, in some case, slowly time-variant (nonstationary). The only way the channel equalizer can compensate for the channel distortion is if the channel equalizer has adjustable coefficients that, in many cases, can be optimized to minimize some measure of the distortion. Typically, this is done by performing measurements on the characteristics of the channel. Such an equalizer with adjustable parameters is referred to as an adaptive equalizer. Throughout this work, the words *filter* and *equalizer* will be used interchangeably.

Although both infinite-impulse response (IIR) and finite-impluse response (FIR) filters have been considered for adaptive equalization, the linear FIR filter is the most practical and widely used. The reason for this preference is because the FIR filter has only adjustable zeros and, hence, it is free of the stability problems associated with IIR filters that have both adjustable poles and zeros. The FIR equalizer, which will also be referred to as a tapped-delay-line, nonrecursive, or moving-average equalizer, is comprised of both current and delayed samples of the received signal, which are weighted by the equalizer coefficients (tap weights) and summed to produce the output.

To cancel the ISI in the unknown channel, these linear transversal filters attempt to approximate the inverse of the channel. Equalizers of this type are referred to as zero-forcing (ZF) equalizers [38]. If the length of the ZF equalizer is increased without bound, the resulting infinite-length equalizer would perfectly invert the channel and, therefore, have zero ISI at its output. However, since a finite-length ZF equalizer can only approximate the inverse of the channel, such filters have been shown to enhance noise at frequencies where the channel spectrum has high attenuation [38]. Furthermore, for non-minimum phase channels (channels whose transfer functions consist of zeros outside the unit circle in the z-plane), the inverse filter can be unstable.

To overcome these difficulties with the zero-forcing equalizer, a nonlinear equalizer can be used. Such an equalizer, referred to as a decision feedback equalizer (DFE), feeds back (subtracts) a weighted sum of the decisions on previously detected symbols in order to remove, from the current channel output, the interference contributed from these past symbols. The DFE can compensate for amplitude distortion with much less noise enhancement than the linear equalizer [32].

An important consideration in the use of an adaptive equalizer is the criterion for optimizing the adjustable filter parameters. The criterion must not only provide a meaningful measure of equalizer performance, but it must also result in a practically realizable algorithm. In some cases, a performance index that is a nonlinear function of the filter parameters possesses many relative minima (or maxima), so that there may be doubt as to whether the adaptive equalizer has converged to the optimal solution or to one of the relative minima (or maxima). For these reasons, some desirable performance indices, such as the average probability of error in a digital communication system, must be rejected on the grounds that they are impractical to implement [37].

One criterion that provides a good measure of performance in adaptive filtering applications is the least-squares (LS) criterion, and its counterpart in a statistical formulation of the problem; namely, the mean-square-error (MSE) criterion. The LS and MSE criteria both result in quadratic performance indices as a function of the filter coefficients and, therefore, each possesses a single minimum.

Using the method of steepest descent, the MSE criterion has been used to develop a recursive algorithm to update the tap weights of the adaptive equalizer. The resulting algorithm, known as the least-mean-square (LMS) algorithm uses instantaneous estimates of the gradient vector of the performance surface to update the tap weights of the transversal filter [44]. The widespread use and popularity of the LMS algorithm stems from its inherent simplicity. However, the LMS suffers from a relatively slow rate of convergence (convergence to the correct channel parameters) and is affected by the eigenvalue spread of the channel correlation matrix.

To improve on the rate of convergence of the LMS, Godard, using an LS criterion, cast the equalizer adjustment problem as an estimation of a stationary state vector in Gaussian noise [14]. The resulting recursive least-squares (RLS) algorithm can be considered as a special case of the Kalman algorithm for adaptive transversal filters. Subsequently, the RLS algorithm has been shown to be the fastest known converging adaptive algorithm [38]. Furthermore, the rate of convergence of the RLS algorithm does not depend on the eigenvalue spread of the channel correlation matrix.

Typically, in adaptive equalizers, such as those based on the LMS and RLS algorithms, a training sequence is used to adapt the equalizer to the unknown channel. During the training period, a known signal is transmitted and a synchronized version of this signal is generated in the receiver to acquire information about the channel characteristics. After the training period, the coefficients of the adaptive equalizer may then be continuously adjusted is a decision-directed mode. However, in many applications, such as remote site receivers in a multipoint telephone modem network, the adaptive equalizers are required to bootstrap in a decision-directed mode without the help of a training sequence [38]. By decision-directed, it is meant that the outputs of a decision (threshold) device—not a training sequence—are used by the equalizer to update the tap weights [31]. Equalizers which do not require a training sequence are referred to as blind equalizers. Conventional blind equalizers, like their non-blind counterparts, are typically of the linear, FIR type.

There are several classes of blind equalization algorithms. The first, known as Bussgang algorithms, consists of an iterative process that uses some form of zero-memory nonlinear estimator to make decisions on the output of the transversal filter [18]. These decisions are then subtracted from the output of the transversal filter to form the error signal by which the LMS algorithm, for example, can be updated. Once the algorithm has converged in the mean, the resulting equalized sequence has been shown to assume Bussgang statistics [2]. Special cases of the Bussgang class of algorithms include the Sato algorithm [39] and the Godard algorithm [15]. Both of these algorithms involve the minimization of some nonconvex cost function. A special case of the Godard algorithm, known as the constant modulus algorithm (CMA), is considered the most widely used of the blind equalization algorithms [18]. The Bussgang class of algorithms is characterized by low computational complexity and an initially slow rate of convergence, which increases as equalization progresses.

The second class of blind equalization algorithms is based on the use of higher-order statistics (higher than second-order correlation functions) of the distorted received signal. These higher-order statistics and their Fourier transforms are referred to as cumulants and polyspectra, respectively [18]. The tricepstrum-based algorithm for blind equalization uses the fourth-order cumulant of the received signal to extract phase information about the channel in order to perform the required blind equalization [16]. This class of algorithms is characterized by a high computational complexity and an initially fast rate of convergence that slows as equalization progresses.

Recently, another blind equalization algorithm was introduced, known as the decorrelation algorithm [24]. The algorithm is based on decorrelating the input to the decision or threshold device (slicer) [24]. By doing so, the decision which is fed back to control the tap weights is gradually improved and, hence, bootstrap the performance of the equalizer. Therefore, for a zero-mean, independent identically distributed (i.i.d.) data source, the channel introduces ISI (correlation), thus decorrelating the input to the slicer will reduce the ISI at the equalizers output [24].

To increase the rate of convergence of the decorrelation algorithm, it was shown in [24] that by minimizing the time-average weighted correlations, the decorrelation algorithm (in a decision feedback configuration) could be implemented using a Kalman, or recursive least, algorithm. However, a disadvantage of this decorrelation-based RLS algorithm, as with all RLS algorithms, is its complexity. For both the traditional and decorrelation-based RLS algorithms, an $M \times M$ matrix must be adapted and stored once per iteration, where M is the number of equalizer tap coefficients. Thus, $\mathcal{O}(M^2)$ operations (multiplications and divisions) must be performed per iteration, where $\mathcal{O}(\cdot)$ denotes *order of.* By contrast, both the LMS and decorrelation algorithms have $\mathcal{O}(M)$ complexity in that their computational complexities increase linearly with M.

The computational complexity of a given algorithm has a direct relationship to the cost of constructing microprocessor-based hardware for use in the practical implementation of these algorithms. When this cost is an issue of primary concern, there is motivation to develop what are called *fast* algorithms for solving the recursive least problem. A fast algorithm is one which conforms to following definition [18]:

> *An algorithm is said to be fast if its computational complexity increases linearly with the dimension of the adjustable weight vector.*

A fast algorithm, then, would be one whose computational requirements are similar to the LMS algorithm or the decorrelation DFE, for example.

With this definition in mind, it is both desirable and possible to reduce the number of operations per iteration of the RLS-type algorithms to be proportional to M, while still maintaining its rapid rate of convergence. The mathematical basis of the resulting fast recursive least algorithms is the exploitation of the *shifting property* inherent in most sequential estimation (prediction) problems. In equalization, this property expresses the fact that at each iteration the number of new samples entering and old samples leaving the equalizer is not M, but a much smaller integer p. Falconer and Ljung [10] were the first to recognize this and subsequently developed the Fast Kalman algorithm. The fast Kalman, or fast RLS, algorithm provides a means for the recursive updating of the Kalman gain vector used in the RLS algorithms without explicit computation of the inverse correlation matrix of the channel.

Carayannis, *et al.* [5] and Cioffi and Kailath [7] took the fast Kalman algorithm one step further and derived another member of this class of fast algorithms, referred to as the fast transversal filter (FTF). The FTF structure, like the fast Kalman, was originally derived as a means of reducing the computational complexity of the traditional RLS algorithm. In fact, it has been shown that the FTF algorithm is approximately forty percent more computationally efficient than the fast Kalman algorithm [5, 7]. This class of algorithms differs from the fast Kalman in that it is based, primarily, on an *a posteriori* error formulation. Furthermore, the FTF makes better use of the relationships between the *a priori* and *a posteriori* errors. It is through an exploitation of these relationships that the FTF algorithm is able to further reduce the complexity of the fast Kalman algorithm. However, it should be noted that both algorithms are classified as members of the same group—transversal filter-based fast algorithms [7].

The FTF class of algorithms gets its name from the fact that the algorithms use four separate transversal filters that share a common input. Like the fast Kalman, the FTF algorithm uses a combination of a recursive forward linear predictor, a recursive backward linear predictor, and a recursive computation of the Kalman gain vector to recursively compute the desired tap weight vector.

Using a similar methodology, the decorrelation algorithm can be applied to both the fast Kalman and FTF algorithms in order to reduce the computational complexity of the decorrelation-based RLS algorithm. It will be shown in this work that both the fast Kalman and FTF algorithms for decorrelation offer a comparable reduction in complexity, while maintaining the high rate of convergence of the more computationally complex decorrelation RLS.

A basic property of the fast transversal-based filters is that the algorithms are recursive in time only. The length of the transversal filter is fixed. Any change in the length of the filter results in a new set of filter coefficients that is totally different from the previous set. Furthermore, the disadvantage of the fast Kalman and fast transversal type algorithms is that they suffer from numerical instability in finite precision environments. This is caused by the long-term accumulation of finite precision errors which eventually makes the algorithms unstable and cause them to diverge.

Several approaches to improve the resistance of these recursive least algorithms to such instabilities have been proposed [18]. However, one possible solution is to define an alternative structure for the filter. It is possible to develop an RLS lattice filter by formulating the LS estimation problem in terms of linear prediction [41]. A beneficial result of the development of the fast Kalman structure is the derivation of the forward and backward prediction coefficients and prediction errors, typical of those in a Levinson-Durbin type recursion. There is a direct relationship between the Levinson-Durbin recursions for the linear predictor coefficients and the reflection coefficients in the lattice filter. The lattice structure for adaptive equalization offers several advantages over the fast transversal structures. First, the lattice filter is order recursive. As a consequence, the number of sections that it contains can be easily increased or decreased without affecting the reflection coefficients of the remaining sections. Furthermore, the lattice coefficients have been found to have a low sensitivity to the effects of finite precision [41]. However, a disadvantage of the lattice structure is in its increased complexity. There is approximately a two-fold increase in computational complexity over the fast Kalman algorithm, although the lattice still maintains an $\mathcal{O}(M)$ complexity.

Like the fast transversal family of algorithms, there are many implementational forms of the basic lattice filter structure, each differing in several ways. The two main forms that will be dealt with are the application of the decorrelation algorithm to the RLS lattice-ladder algorithm and the gradient lattice-ladder algorithm of [41] and [40], respectively. Although the structure of the filter is the same in both cases, the computational complexity of the gradient lattice-ladder is less than the RLS lattice-ladder. The reason for this is that the reflection coefficients for the gradient lattice are identical, while the forward and backward reflection coefficients for the RLS lattice have to be updated separately.

The numerical stability problems of the fast transversal-based algorithms reveal a problem that is not always evident in an analog, or infinite precision, environment. The issue of finite-precision and the effect it has on the performance of an algorithm (in terms of both numerical stability and numerical accuracy) is of direct concern if an algorithm is to be eventually implemented in real-time hardware. The performance of an algorithm is not truly known until an attempt is made to implement an algorithm in real-time in a finite-precision environment. An algorithm that performs well in simulation may, in fact, diverge when actually implemented in hardware [43]. In this work, the decorrelation-based DFE will be implemented in real-time using a new microprocessor-based system. Therefore, it will be necessary to better understand the effect of a finite-precision environment on the decorrelation algorithm. In order to do so, models for the quantization error will have to be studied and relevant results applied to the decorrelation algorithm. To facilitate this study, the effects of finite-precision on the decorrelation algorithm will be compared in relation to the LMS algorithm, for which much work on finite-precision modeling has been done [4]. The SPROClab Development System has, at its heart, a 24-bit, fixed-point arithmetic microprocessor. Since fixed-point arithmetic is used in all implementations in the SPROC environment, it is within this context that the LMS and decorrelation algorithms will be considered.

The thesis will be organized in the following manner: In Chapter 2, the system parameters will be introduced. Namely, the DFE, the channel model, and the decorrelation algorithm will be reviewed. In addition, the decorrelation-based DFE will be implemented on the SPROC digital signal processor. In Chapter 3, a study of the effects of finite-precision on the decorrelation algorithm will be made. To facilitate the study, a review of the necessary finite-precision models will be performed and a comparison will be made with the limited-precision LMS. In Chapter 4, a review of the Recursive Least Correlation (RLC) algorithm will be made. Then, two new fast decorrelation algorithms will be proposed, derived, and simulated—the fast Kalman algorithm for decorrelation (FRLC) and the fast transversal filter (FTF) for decorrelation. The performance (rate of convergence) of these new, fast algorithms will be compared to that of the RLC. In Chapter 5, alternative fast structures for the decorrelation algorithm will be discussed. The decorrelation algorithm will be incorporated into existing RLS and gradient lattice structures in order to update the tap weights. Simulation results and comparisons to the RLC algorithm are provided. Finally, in Chapter 6, conclusions on the work presented in this thesis will be discussed, along with areas of possible future research.

CHAPTER 2

THE SYSTEM

Before discussion of the main topics can begin, the problem statement must first be formulated and background theory reviewed. Therefore, in this chapter, the characteristics of the system and channel model under consideration will be proposed. The effect of ISI in high-speed data communication systems and its removal through the use of DFE's will be reviewed. As a means of comparison that will be used throughout this work, background theory on the LMS algorithm will be presented. Finally, a discussion of the decorrelation algorithm and its implementation in a blind, decision feedback configuration using real-time hardware will be performed.

2.1 Channel Model

To better understand ISI from a mathematical perspective, consider the baseband binary PAM system of Figure 2.1 [17], which will be the basic transmission system considered in this work.

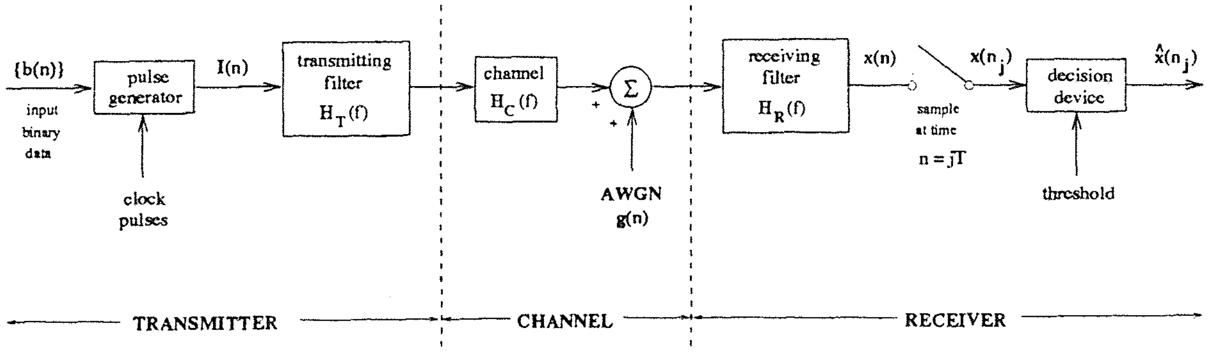

Figure 2.1 Model of Baseband Binary PAM System (without Equalization).

A binary data sequence $\{b(n)\}$, with each bit having a duration of T seconds, is applied to the input of the system. The resulting output pulse waveform of the pulse generator is

$$I(n) = \sum_k a(k)p(n - kT) \tag{2.1}$$

where $p(n)$ is the shaping pulse of the pulse generator. The quantity $a(n)$ is an amplitude that corresponds to the identity of the input bit, $b(n)$. For the PAM system under consideration, polar or nonreturn-to-zero (NRZ) signalling will be assumed. Therefore, $a(n)$ is defined as

$$a(n) = \begin{cases} +1, & \mathrm{b(n)} = 1 \\ -1, & \mathrm{b(n)} = 0 \end{cases} \tag{2.2}$$

The modulated signal $I(n)$ is then passed through a transmitting filter, with transfer function, $H_T(f)$, to be sent over a channel with transfer function, $H_C(f)$. At the input to the receiving filter, random noise, $g(n)$, which can be modelled as zero-mean, additive, white Gaussian noise (AWGN), is added to the output of the channel. The received signal is passed through a receiving filter, with transfer function $H_R(f)$, which is then sampled at multiples of the bit duration, kT. For purposes of this work, the cascade of the transmit filter, channel, and receive filter can be modelled as in Figure 2.2 (see [38] and [17]).

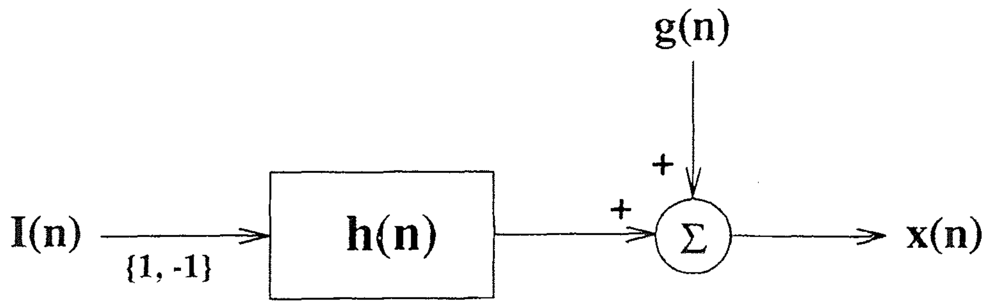

Figure 2.2 Discrete Channel Model.

In Figure 2.2, $h(n)$ is a fading dispersive channel which can itself be modelled as a non-recursive (moving-average) finite impulse response (FIR) filter with impulse response

$$h(n) = \sum_k h(k)\delta(n - kT) \tag{2.3}$$

where $\delta(\cdot)$ is the Kronecker delta. The resulting signal is then passed through a decision device (memoryless detector) to slice (threshold) the signal in order to retrieve the original amplitude levels of $a(n)$. In this work, the slicer will make decisions on the received signal in the following manner:

$$\hat{x}(n) = \begin{cases} 1 & \text{if } x(n) > 0 \\ -1 & \text{if } x(n) < 0 \end{cases} \tag{2.4}$$

Therefore, the output of the receiving filter can be written as

$$x(n) = \sum_k I(k)h(n - kT) + g(n) \ . \tag{2.5}$$

If the output of the channel, $x(n)$, is sampled at instant $n_j{=}jT$, (where j is an integer for symbol-rate equalizers), then

$$\begin{aligned} x(n_j) &= \sum_k I(k)h(jT - kT) + g(n_j) \\ &= I(j) + \sum_{k \neq j} I(k)h((j-k)T) + g(n_j) \ . \end{aligned} \tag{2.6}$$

The first term, $I(j)$, represents the desired signal, since it can be used by the decision device to identify the transmitted amplitude level. The second term represents the residual effect of all other transmitted symbols. This residual effect is the ISI as discussed earlier. The last term is the noise at instant n_j. Note that in the absence of ISI and noise,

$$x(n_j) = I(j) \tag{2.7}$$

and the j^{th} transmitted symbol can be decoded correctly. However, the presence of ISI and noise will introduce errors, in the form of incorrect decisions, at the slicer. The purpose of the equalizer, therefore, is to remove the ISI and noise so that correct decisions on the symbols can be made.

2.2 The Decision Feedback Equalizer and the LMS Algorithm

To remove the ISI from the received signal, several equalization structures have been proposed [38]. One, simple, nonlinear equalizer that is particularly effective for channels with severe amplitude distortion is the DFE [32]. The general structure of the DFE is shown in Figure 2.3. Notice that the DFE consists of both a feedforward and feedback filter.

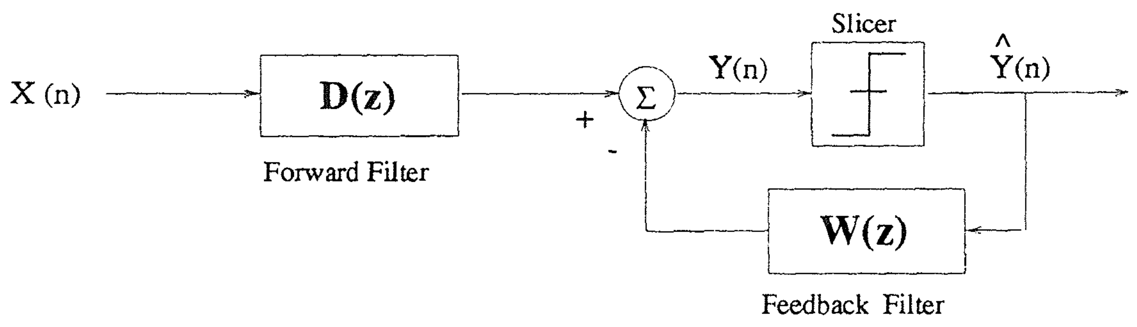

Figure 2.3 Decision Feedback Equalizer.

The basic idea behind the DFE is to use the previous decisions on the incoming data to cancel the interference contributed by symbols which have already been detected. Therefore, if the values of the previously detected symbols are known (where the past decisions are assumed to be correct), then the ISI contributed by these symbols can be canceled exactly by subtracting weighted values of the past symbols from the current channel output [38]. Note that if an incorrect decision is fed back, the output of the DFE will reflect this error during the next several symbols as the incorrect decision traverses the feedback line. As a result, there is a greater likelihood of more incorrect decisions following the first one. This situation is known as error propagation, and can have potentially catastrophic consequences for the equalizer. Fortunately, on most typical channels, errors usually occur in short bursts, which only degrade performance slightly [38].

For the purposes of this work, given the channel discussed in the previous section, it can be shown (see [24] and [32]) that, in the absence of precursors, the forward filter of Figure 2.3 is not needed. The feedback filter, $\mathbf{W}(z)$, will be sufficient to cancel the ISI. Therefore, the DFE under consideration for this thesis is shown in Figure 2.4.

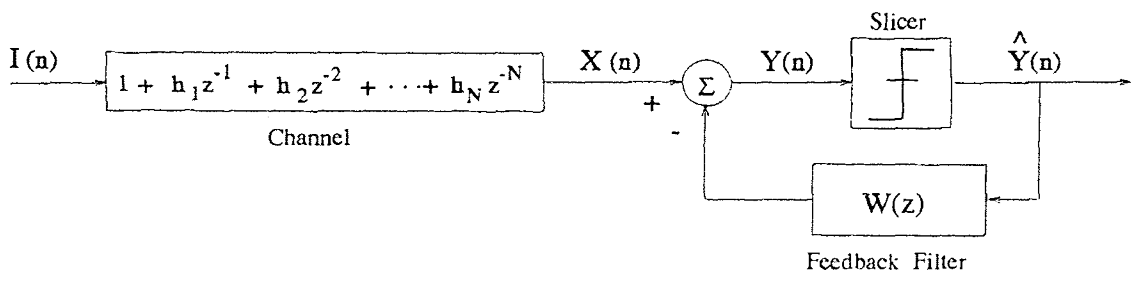

Figure 2.4 Revised Structure of the Decision Feedback Equalizer.

Referring to Figure 2.4, it will be assumed that the input sequence, $I(n)$, is a binary, white sequence with zero-mean, consisting of the elements $\{1, -1\}$, each occurring with equal probability. Normalizing the impulse response of Equation 2.3 relative to the first cursor (h_0), the output of the channel is given by

$$x(n) = I(n) + \sum_{k=1}^{N} h_k(n) I(n-k) \; . \tag{2.8}$$

The channel weights (post-cursors) $\{h_1(n), h_2(n), \ldots, h_M(n)\}$ introduce ISI on the correct data symbol, $I(n)$. For the feedback filter, $\mathbf{W}(z)$, the impulse response is given by

$$w(n) = \sum_{k=1}^{N} w_k(n) \delta(n - kT) \tag{2.9}$$

where N is the number of taps. As can be seen from Figure 2.4, the input to the slicer, $y(n)$, is given by

$$y(n) = x(n) - \sum_{k=1}^{N} w_k(n) \hat{y}(n-k) \; . \tag{2.10}$$

In vector form,

$$y(n) = x(n) - \hat{\mathbf{Y}}'_{\mathrm{N}}(n) \mathbf{W}_{\mathrm{N}}(n) \tag{2.11}$$

where $\hat{\mathbf{Y}}'_{\mathrm{N}}(n) = [\hat{y}_{k-1}, \hat{y}_{k-2}, \ldots, \hat{y}_{k-N}]$ is the vector of the past N decisions, $\mathbf{W}' = [w_1(n), w_2(n), \ldots, w_N(n)]$ is the vector of equalizer tap weights, and the superscript

(′) denotes transposition. The question now concerning the DFE of Figure 2.4 is how to update the tap weights of Equation 2.11 in an adaptive manner to cancel the ISI from the current output of the channel. Perhaps the most popular of the adaptive equalization algorithms is the LMS algorithm [44]. A review of the LMS will be given, since comparisons between the LMS and the decorrelation algorithm, discussed in subsequent sections, will become necessary throughout this work. The following discussion is based on that presented in [44] and [18].

As a review, the LMS algorithm is a member of a larger family of stochastic gradient-based algorithms which are capable of searching a multidimensional performance surface for some desired global minimum. Its widespread use in adaptive filtering stems from its inherent simplicity. As will be shown, the LMS requires neither measurements of relevant correlation functions nor matrix inversions, which are characterized with the steepest-descent algorithm. In the method of steepest descent, changes in the tap weight vector $\mathbf{W}_{\mathrm{N}}(n)$ are made in a direction opposite to that of the gradient vector, which is calculated according to the MSE performance surface. Due to these corrections, the tap weight vector $\mathbf{W}_{\mathrm{N}}(n)$ will move down the error-performance surface and eventually reach a stationary minimum, which is the Wiener solution, $\mathbf{W}_{opt} = \mathbf{R}^{-1}\mathbf{p}$ [18].

Since it is difficult to get an exact measure of the gradient vector at any point on the performance surface, the LMS uses noisy estimates of the gradient vector in order to update the tap weight vector $\mathbf{W}_{\mathrm{N}}(n)$. Therefore, instead of terminating on the Wiener solution, $\mathbf{W}_{opt}$, the estimate of the tap weight vector, $\hat{\mathbf{W}}_{\mathrm{N}}(n)$, exhibits a random motion around the minimum of the error-performance surface. An implementation of an adaptive filter is depicted in Figure 2.5.

Notice that the transversal filter of Figure 2.5 consists of an adaptive process by which the set of tap weights is automatically adjusted. Furthermore, it consists of a filtering process which involves the generation of an estimate of the desired response

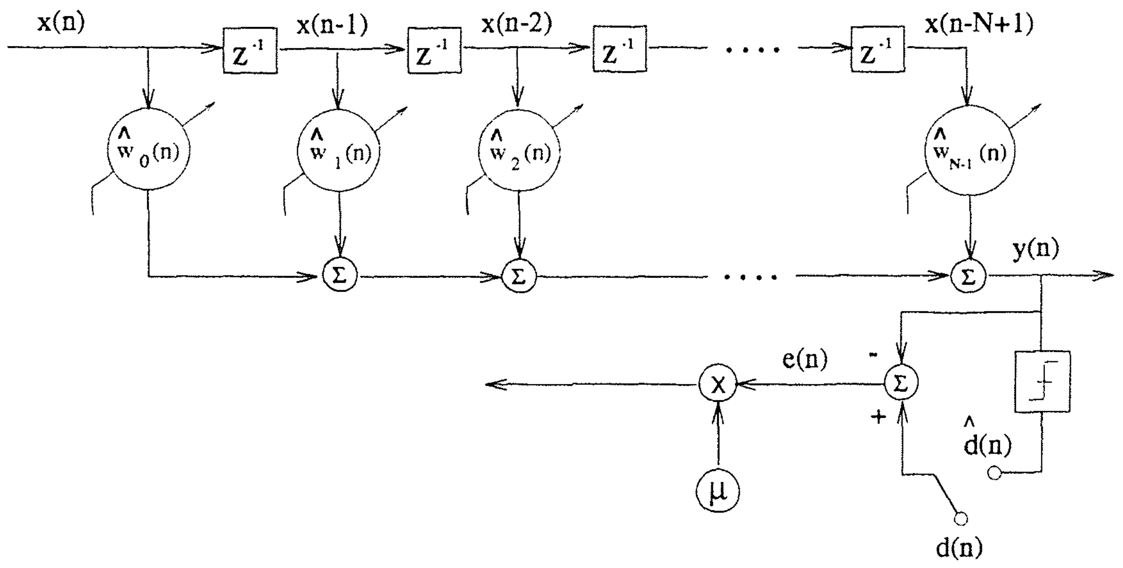

Figure 2.5 Model of an Adaptive Filter.

formed from the inner product of the tap inputs and the corresponding tap weights. An estimate of the error is also calculated by comparing the estimate of the desired response with the actual value of the desired response. This estimate will, in turn, be used in the adaptive process to update the tap weights. In Figure 2.5, the desired response, $d(n)$, is usually a data sequence known to the equalizer that is used to train it. Once trained, the equalizer operates in a decision-directed mode, using the estimates of the desired response, $\hat{d}(n)$, in the calculation of the tap weights.

Let $\mathbf{X}(n) = [x(n), x(n-1), \ldots, x(n-N+1)]'$ denote the vector of length N containing the N most recent samples of a single input sequence. Let $\mathbf{W}(n) = [w_0(n), w_1(n), \ldots, w_{N-1}(n)]'$ denote, at time n, the N-length vector of tap weights. Therefore, the output $y(n)$, at any time n, of the transversal filter is given by

$$y(n) = \sum_{k=0}^{N-1} w_k(n)x(n-k) = \mathbf{W}'(n)\mathbf{X}(n) = \mathbf{X}'(n)\mathbf{W}(n) \ . \tag{2.12}$$

Denoting the desired response at time n as $d(n)$, the estimation error between the desired response and the output of the filter (which is an estimate of the desired response) is

$$e(n) = d(n) - y(n) = d(n) - \mathbf{W}'(n)\mathbf{X}(n) = d(n) - \mathbf{X}'(n)\mathbf{W}(n) \tag{2.13}$$

If the estimation error is squared,

$$\begin{aligned} e^2(n) &= \{d(n) - \mathbf{W}'(n)\mathbf{X}(n)\}^2 \\ &= d^2(n) - d(n)\mathbf{X}'(n)\mathbf{W}(n) - d(n)\mathbf{W}'(n)\mathbf{X}(n) + \mathbf{W}'(n)\mathbf{X}(n)\mathbf{X}'(n)\mathbf{W}(n) \\ &= d^2(n) - 2\mathbf{W}'(n)\mathbf{X}(n)d(n) + \mathbf{W}'(n)\mathbf{X}(n)\mathbf{X}'(n)\mathbf{W}(n) \ . \end{aligned} \tag{2.14}$$

Taking the expectation of both sides of Equation 2.14 yields the MSE,

$$\begin{aligned} \mathcal{J}(\mathbf{W}) &= \mathcal{E}\left[e^2(n)\right] = \mathcal{E}\left[d^2(n)\right] - 2\mathbf{W}'(n)\mathcal{E}\left[\mathbf{X}(n)d(n)\right] \\ &\quad + \mathbf{W}'(n)\mathcal{E}\left[\mathbf{X}(n)\mathbf{X}'(n)\right]\mathbf{W}(n) \end{aligned} \tag{2.15}$$

where $\mathcal{E}(\cdot)$ represents expectation. Define the autocorrelation matrix $\mathbf{R}$ of the input signals as

$$\mathbf{R} = \mathcal{E}\left[\mathbf{X}(n)\mathbf{X}'(n)\right] \tag{2.16}$$

and the cross-correlation vector $\mathbf{p}$ between the input signal vector and the desired response as

$$\mathbf{p} = \mathcal{E}\left[\mathbf{X}(n)d(n)\right] \tag{2.17}$$

Thus, the MSE can be written as follows (where, for convenience, the time-index n has been dropped)

$$\mathcal{J}(\mathbf{W}) = \sigma_d^2 - 2\mathbf{W}'\mathbf{p} + \mathbf{W}'\mathbf{R}\mathbf{W} \tag{2.18}$$

where σ_d^2 is the variance of the desired response, $d(n)$. Notice from Equation 2.18 that the MSE is a quadratic function of the tap weights. Therefore, the dependence of the cost function $\mathcal{J}$ on the tap weights can be visualized as a *bowl-shaped* surface with a unique minimum [44, 18]. This surface is referred to as the error performance surface of the transversal filter of Figure 2.5. As discussed previously, the adaptive process will continuously adjust the tap weights in order to seek out the bottom (minimum) of the bowl. In other words, the adaptive process seeks to minimize the MSE [18].

If the MSE function of Equation 2.18 is differentiated with respect to the tap weight vector, the gradient of the error performance surface can be obtained. Consequently,

$$\nabla\left(\mathcal{J}\left(\mathbf{W}\right)\right) = -2\mathrm{p} + 2\mathbf{RW} \quad . \tag{2.19}$$

Recall that according to the method of steepest descent, updates to the tap weight vector are proportional to the negative of the gradient vector. Accordingly, the tap update equation for the steepest-descent can be written as

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \frac{1}{2}\mu\left[-\nabla\left(\mathcal{J}\left(\mathbf{W}\right)\right)\right] \tag{2.20}$$

where μ is a scalar quantity which controls both the stability and rate of adaptation (convergence) of the equation. Substituting Equation 2.19 into Equation 2.20 gives the algorithm for the method of steepest descent:

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu\left[\mathrm{p} - \mathbf{RW}\right] \tag{2.21}$$

To derive the LMS algorithm, an estimate of the gradient vector will be used. From Equation 2.21, substituting instantaneous estimates of $\mathbf{R}$ and p would result in one form of estimate for the gradient vector [44]. Define,

$$\hat{\mathbf{R}}(n) = \mathbf{X}(n)\mathbf{X}'(n) \tag{2.22}$$

and

$$\hat{\mathrm{p}}(n) = \mathbf{X}(n)d(n) \quad . \tag{2.23}$$

Therefore, the instantaneous estimate of the gradient vector would be

$$\nabla\left(\mathcal{J}\left(\mathbf{W}\right)\right) = -2\mathbf{X}(n)d(n) + 2\mathbf{X}(n)\mathbf{X}'(n)\hat{\mathbf{W}}(n) \tag{2.24}$$

where, again, $\hat{\mathbf{W}}(n)$ is the estimate of the tap weight vector. From Equation 2.21,

$$\hat{\mathbf{W}}(n+1) = \hat{\mathbf{W}}(n) + \mu\mathbf{X}(n)\left[d(n) - \mathbf{X}'(n)\hat{\mathbf{W}}(n)\right] \tag{2.25}$$

Substituting the estimation error, as defined in Equation 2.13, into Equation 2.25, the LMS algorithm can be written as follows:

$$\hat{\mathbf{W}}(n+1) = \hat{\mathbf{W}}(n) + \mu \mathbf{X}(n)e(n) \tag{2.26}$$

where the second term on the right-hand side of Equation 2.26, $\mu\mathbf{X}(n)e(n)$, is the correction or update term applied to the current estimate of the tap weight vector, $\hat{\mathbf{W}}(n)$. The term $\mathbf{X}(n)e(n)$ of the update expression is the noisy estimate of the gradient vector. Because of this noisy estimate, the LMS algorithm will exhibit a random walk motion around the minimum of the error-performance surface [44]. The rate of convergence of the LMS algorithm is highly sensitive to variations in the step size parameter, μ. If the step size is too small, the algorithm is said to be *overdamped* in that the trajectory of the tap weight vector follows a continuous path [18]. However, if the step size is too large, the algorithm is said to be *underdamped* in that the trajectory of the tap-weight vector exhibits an oscillatory behavior [18].

As a final note, recall that the use of the LMS-based adaptive equalizer requires the need for a training sequence to allow the equalizer to *learn* the characteristics of the channel. It is this reason that distinquishes equalizers of this type from their blind counterparts. Blind equalizers, as previously mentioned and for which the decorrelation-based DFE will be discussed, are able to adapt to the given channel without any such training.

2.3 The Decorrelation Algorithm

The algorithm under consideration in this work is a new, blind adaptive equalization algorithm that is based on decorrelating the input to the slicer of Figure 2.4 [24]. In doing so, the decisions which are fed back to control the tap weights are gradually improved and, hence, bootstrap the performance of the equalizer. Therefore, for a zero-mean, i.i.d. data source, since the channel introduces ISI (*i.e.*, correlation), then decorrelating the input to the slicer will reduce the ISI at the equalizer's output [24].

In order to cancel the ISI, at the input of the slicer $y(n)=I(n)$ and the sequence $\{Y(n)\}$ will be uncorrelated. In other words, $\mathcal{E}\left[y(n)y(n-k)\right]=0$ for $k \neq 0$ at the input to the slicer. It was shown in [24] that decorrelation of the slicer's input is a necessary and sufficient condition for the ideal cancellation of ISI. Therefore, the decorrelation of the slicer's input can be used as a criterion for controlling the update algorithm for the tap weight vector $\mathbf{W}$ [24].

Using the method of steepest descent, as discussed in the previous section, and the aforementioned decorrelation criterion, the tap weights of the feedback filter can be updated according to

$$w_k(n+1) = w_k(n) + \mu\mathcal{E}\left[y(n)y(n-k)\right] \tag{2.27}$$

where $k=1,2,\ldots,N$. The step size parameter, μ, controls the stability and rate of adaptation (convergence) of the algorithm, in a manner similar to that discussed with the LMS algorithm. For a digital implementation of Equation 2.27, an instantaneous estimate of the expectation can be used. Therefore, the update equation given in Equation 2.27 can be realized as

$$w_k(n+1) = w_k(n) + \mu y(n)y(n-k) \tag{2.28}$$

where, again, $k=1,2,\ldots,N$. It was shown in [24] that Equation 2.28 will converge in the mean, so that the mean value of w_k will converge to the corresponding channel parameter h_k. Furthermore, an extensive discussion of the transient and steady-state performance, convergence in the mean, sufficiency, and the global admissability of the decorrelation algorithm can also be found in [24]. Bounds on the probability of error of the decorrelation-based DFE were derived in [23]. The derivation and proof of the transient and steady-state performance, as well as the error performance, of the blind decorrelation-based DFE are beyond the scope of this work and the interested reader is referred to the respective references.

Equation 2.28 can be written in vector form as follows:

$$\mathbf{W}_{\mathrm{N}}(n+1) = \mathbf{W}_{\mathrm{N}}(n) + \mu y(n)\mathbf{Y}_{\mathrm{N}}(n) \tag{2.29}$$

where $\mathbf{Y}_{\mathrm{N}}(n) = [y_{n-1}, y_{n-2}, \ldots, y_{n-N}]$ is the vector of past inputs to the slicer, $\mathbf{W}_{\mathrm{N}}(n) = [w_1(n), w_2(n), \ldots, w_N(n)]$ is the vector of tap weights, and $y(n)$ is the current input to the slicer, given by Equation 2.11. A representation of the decorrelation algorithm of Equation 2.29 implemented in a decision feedback configuration is shown in Figure 2.6.

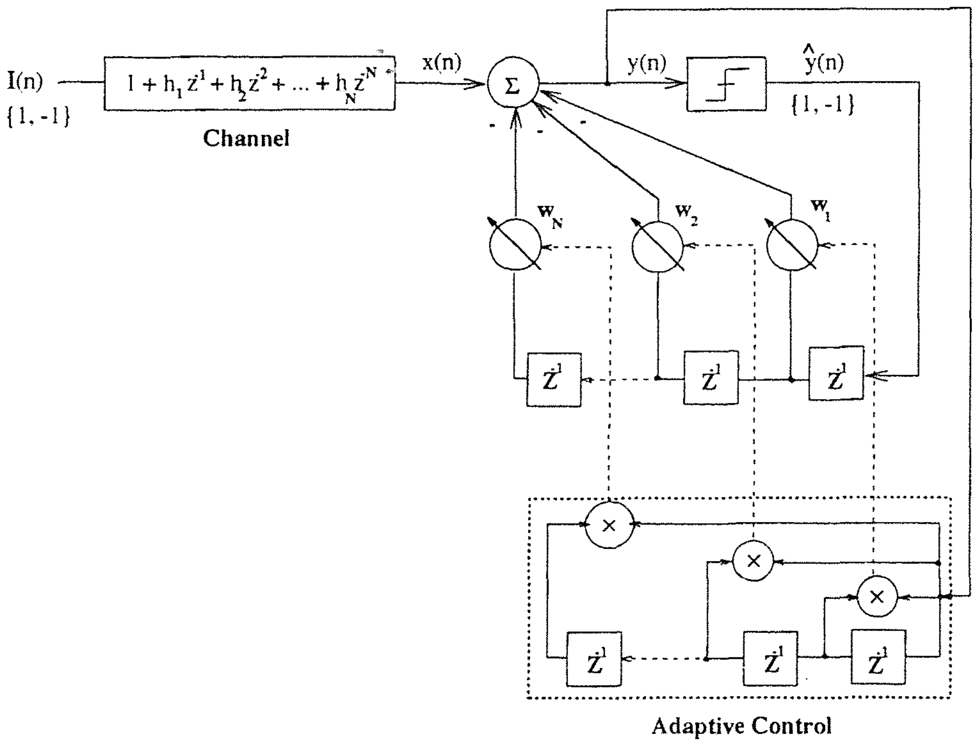

Figure 2.6 Decorrelation-Based DFE.

2.3.1 Real-Time Hardware Implementation of the Decorrelation Algorithm

In [24], numerous simulations with a blind, decorrelation-based DFE were performed and shown to effectively remove the ISI imposed by the channel. Therefore, further simulations in this work will not be performed. However, using a new microprocessor-based system, the decorrelation-based DFE of the previous section will be implemented in real-time hardware.

In traditional approaches to the implementation of signal processing algorithms in hardware using digital signal processing (DSP) chips, lengthy software code must be written. The effect of the sometimes arduous coding effort is the lengthy extension of the product development period. A way to circumvent many of the drawbacks of DSP's is to use a system-level design approach. Star Semiconductor's SPROC (Signal PROCessor) digital signal processor follows this approach. The SPROClab Development System allows the engineer to enter a design at the system level via signal-flow diagrams, using cells from a cell library of hundreds of system building blocks. Once the signal flow diagram has been entered, the SPROC system analyzes the diagram and produces executable code for the SPROC DSP. Since the SPROC DSP is a RAM-based DSP (essentially, an erasable, programmable, read-only-memory (EPROM)), the executable code generated by the development system is used to dynamically program (*burn*) the actual SPROC chip. The advantage to using an EPROM DSP is that modifications to the system design can be made quickly, easily, and in an extremely efficient manner. Multiple hardware configurations can be tested and judged without the need for the creation of separate, dedicated ROM-based DSPs. Consequently, the SPROC chip can reduce production time from many months to, quite literally, a few minutes.

The microprocessor at the heart of the SPROClab Development System is a 24-bit, fixed-point arithmetic machine. Fixed-point numbers, $x(n)$, in the SPROC environment are defined by the limit, $-2.0 \leq x(n) < 2.0$. As will be discussed in Chapter 3, this restriction poses several problems to the successful implementation of the decorrelation-based DFE of Figure 2.6. Overflow and the accumulated effects of the finite-precision environment can affect the convergence of the tap weights catastrophically (*i.e.*, divergence). In any fixed-point, finite-precision machine, the finite register lengths can create severe overflow problems in performing basic mathematical operations such as addition and multiplication if care is not taken to properly scale all relevant data. More on this subject will be discussed in Chapter 3.

To test the real-time performance of the decorrelation algorithm, the decorrelation-based DFE of Figure 2.6 was implemented in SPROC. The SPROC-based decorrelation DFE is shown in Figure 2.7.

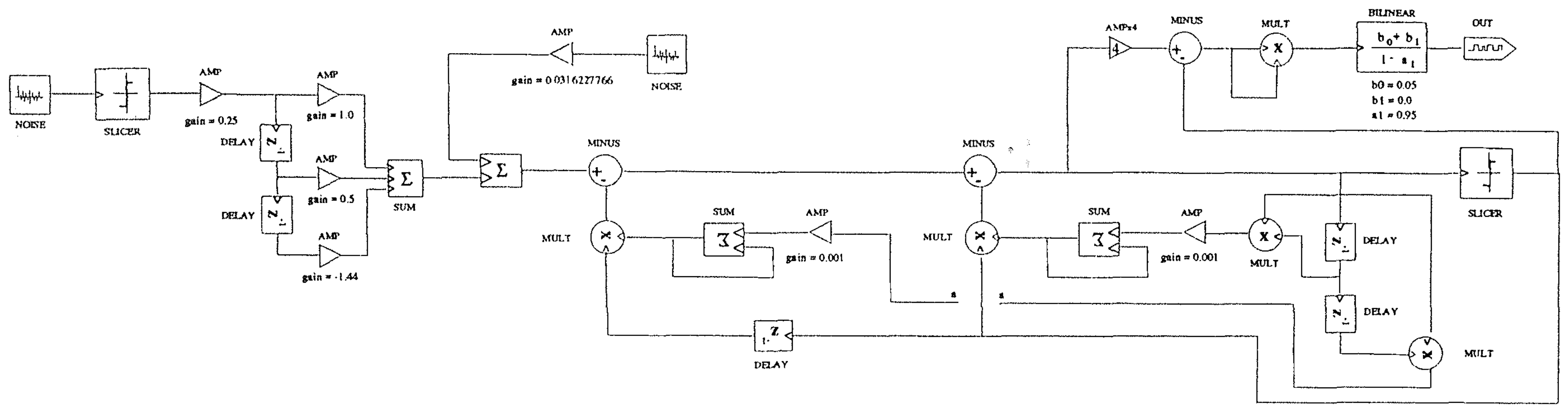

Figure 2.7 SPROC Implementation of Decorrelation-Based DFE.

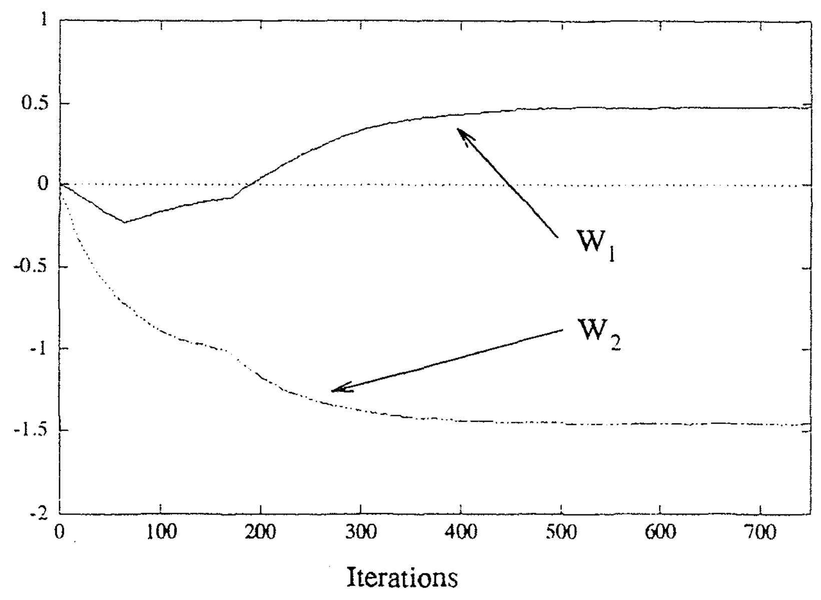

Figure 2.8 Learning Curve of the Tap Weights of SPROC-Based Decorrelation DFE.

In this particular system-level design, the i.i.d. source, channel, and equalizer were all implemented together on chip in order to facilitate experimentation. For the decorrelation DFE under study, the following non-minimum phase channel was used

$$H(z) = 1 + 0.5z^{-1} - 1.44z^{-2} \tag{2.30}$$

where the step size parameter, μ, was set equal to 0.001. Note from Figure 2.7 that each icon represents one piece of assembly-language code. The SPROClab Development System is able to take these individual pieces of code and generate a complete program, which it then *burns* into the actual SPROC chip. Notice from Figure 2.7 that a scaling factor of 0.25 has been included in order to prevent overflow. The selection of the proper scaling factor will be discussed extensively in Chapter 3.

In the first several examples of the SPROC-based decorrelation DFE, the white, Gaussian noise pictured in Figure 2.7 will not be added to the output of the channel. Consequently, the learning curve of the tap weights of the SPROC-based decorrelation algorithm is shown in Figure 2.8. As can be seen in Figure 2.8, both tap weights converge to the appropriate values of w_1=0.5 and w_2=-1.44, respectively. In Figure 2.8, as well as in the other learning curves presented in this chapter, the high rate of speed at which the SPROC-based algorithm converged for μ=0.001 (less than one second) precluded the taking of relevant data. Therefore, it was necessary to decrease the step size to μ=0.0001 in order to generate the necessary learning curves. Lastly, since the SPROC chip is rated at a clock speed of 50 MHz, there is a large discrepancy between the clock speed of the microprocessor and the rate at which the computer receives samples of the data through its serial port (9600 bits per second). Since the computer only receives a fraction of the total number of data samples from the SPROC's output, the actual number of iterations before convergence is much higher. The learning curves of the SPROC-based DFE are meant to show that the decorrelation DFE is able to converge; not how fast nor how long it takes to do so.

For the SPROC implementation, an appropriate measure of the algorithm's performance is given by considering the learning curve of the MSE between the equalized data signal, $y(n)$, and its estimate (decision), $\hat{y}(n)$. The estimate of the residual ISI power is obtained by passing the sequence of instantaneous squared errors $(y(n) - \hat{y}(n))^2$ (see Figure 2.6) through a smoothing filter whose transfer function is given by $0.05/(1 - 0.95z^{-1})$ [20]. According to [20], this will provide a sense of *recent average* performance of the decorrelation-based DFE. Figure 2.9 shows the learning curve of the MSE of the SPROC-based decorrelation algorithm of Figure 2.7.

Notice from Figure 2.9 that the error does indeed appear to converge to zero. In actuality, the residual MSE oscillates around zero at a proximity that varies anywhere between $\pm 10^{-4}$ and $\pm 10^{-}5$. This residual MSE occurs as a result of implementation in a finite-precision environment. In infinite-precision simulations of the decorrelation algorithm, the MSE has been shown to decay to true zero [24]. For purposes of discussion of the real-time implementation of the decorrelation algorithm, references to convergence to zero will mean *effectively* to zero.

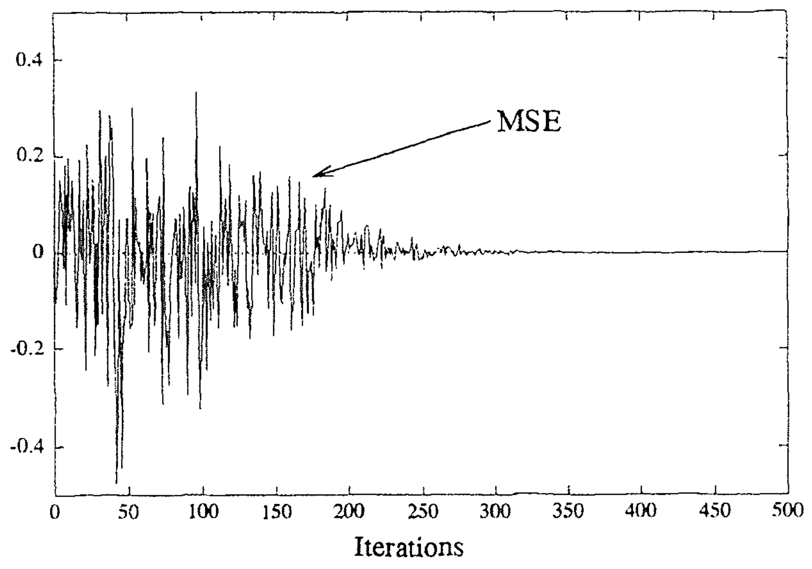

Figure 2.9 Learning Curve of MSE of SPROC-Based Decorrelation DFE.

In performing the previous tests, the tap weights were initialized to zero. The question arises as to whether the algorithm can still converge if the tap weights are initialized to any finite value, not including zero. To study this question, the same non-minimum phase channel was used, but with the first tap weight initialized to -0.5 and the second tap weight to +1.44. This can be done in the SPROClab environment by writing the initial values to the appropriate data locations prior to running the algorithms. The learning curve of the tap weights for non-zero initial conditions is shown in Figure 2.10.

Notice in Figure 2.10 that even with arbitrary initial conditions, the decorrelation algorithm is still able to converge to the correct tap weights, although the convergence time appears longer than that in Figure 2.8. The corresponding learning curve of the MSE for non-zero initial conditions is shown in Figure 2.11.

From Figure 2.11 it is again evident that the error does indeed converge to zero, as before. However, this convergence appears to take longer than in the corresponding MSE of Figure 2.9. In comparing the convergence rate performance of Figures 2.9

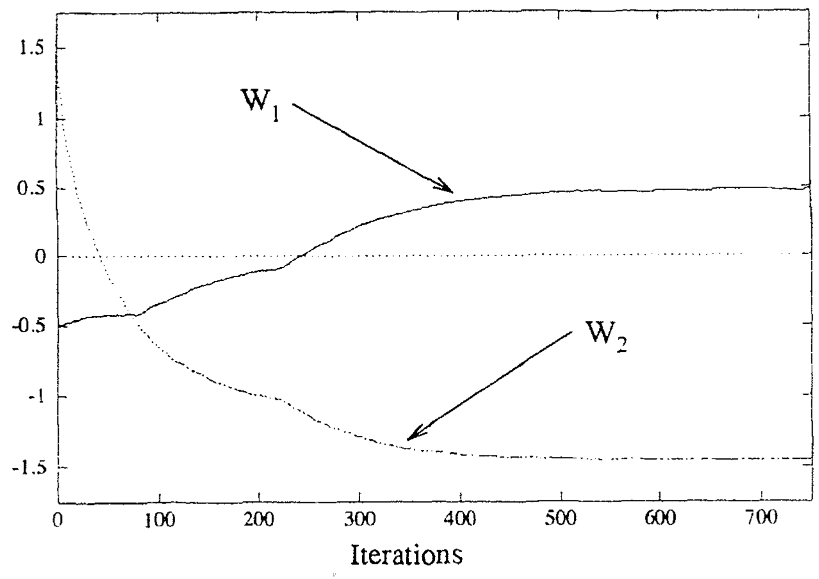

Figure 2.10 Learning Curve of Non-zero Initialized Tap Weights.

and 2.11, the algorithm converges for any given initial state, thus the algorithm is said to be globally convergent [20]. This agrees well with the findings and simulations of [24].

As a final demonstration of the SPROC-based implementation of the decorrelation algorithm, white, zero-mean, Gaussian noise with variance 0.001 (a signal-to-noise ratio (SNR) of 30 dB) will be added to the output of the channel. The learning curve of the tap weights, which are both initialized to zero, is shown in Figure 2.12 and the corresponding MSE is shown in Figure 2.13. In comparison with the previous learning curves, the addition of AWGN does not pose a burden for the decorrelation algorithm. Convergence of the DFE still occurs, although it appears to take longer than in the non-noise case. Furthermore, close inspection of Figure 2.13 reveals that the MSE does exhibit a small oscillation around zero. This is a direct consequence of the addition of the AWGN to the channel. It should be noted that the feedback filter is effectively removing the ISI, since the MSE does converge to zero. However, because no forward filter is being used, the AWGN is not being

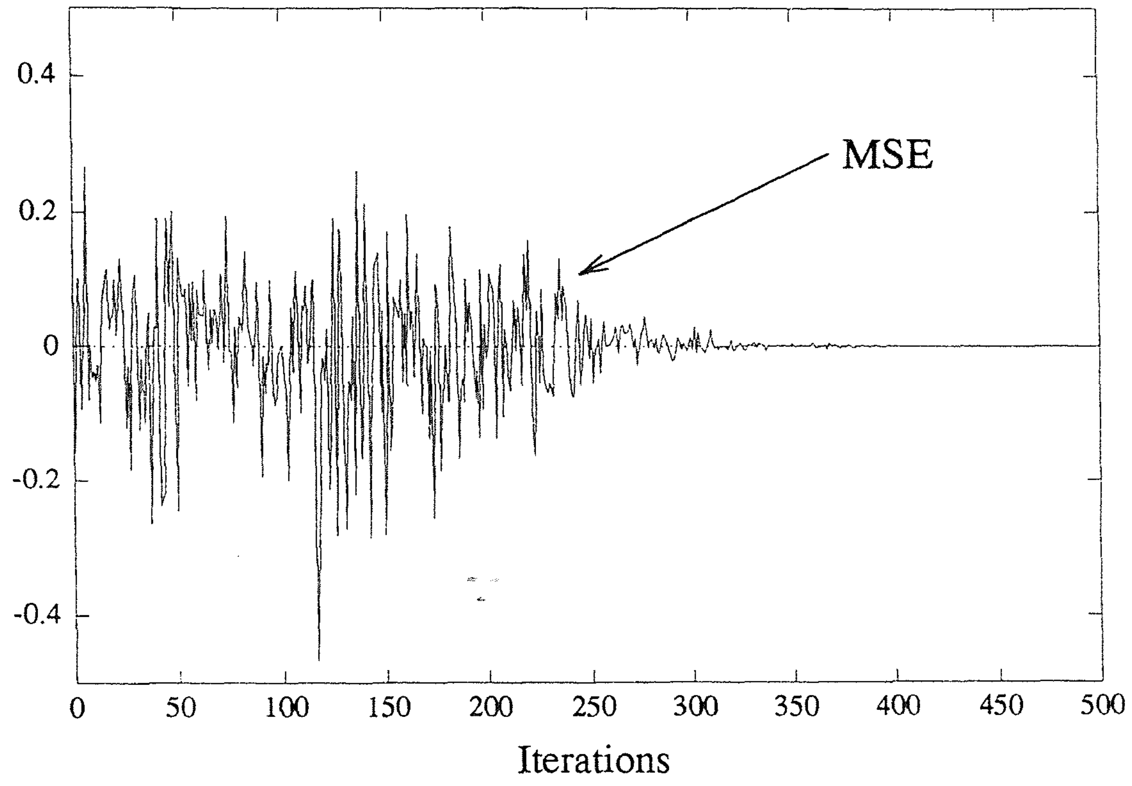

Figure 2.11 Learning Curve of MSE of SPROC-Based Decorrelation DFE.

effectively removed and its residual presence is the cause for the minor fluctuations in the MSE after the MSE has converged.

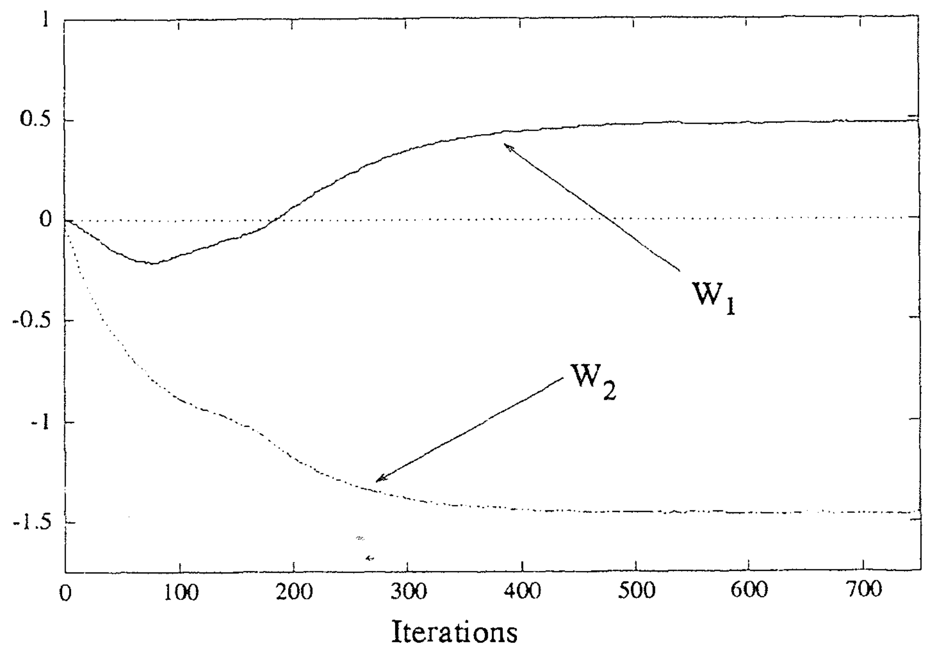

Figure 2.12 Learning Curve of the Tap Weights of SPROC-Based Decorrelation DFE with AWGN.

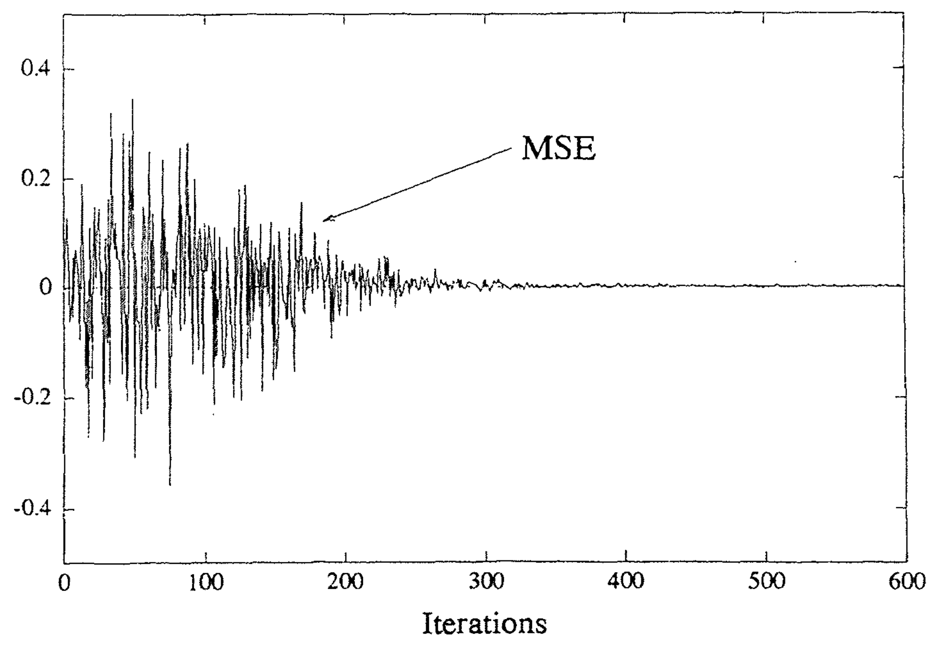

Figure 2.13 Learning Curve of MSE of SPROC-Based Decorrelation DFE with AWGN.

CHAPTER 3

THE DECORRELATION ALGORITHM AND FINITE-PRECISION

Throughout the main body of this thesis, the derivations of all the relevant algorithms are considered in an analog, or infinite precision, environment. However, in any digital realization, whether hardware or software, the effects of the finite word-length of the registers can alter the performance (*i.e.*, convergence) of the algorithm. Therefore, in this chapter, the effect of a finite-precision environment on the decorrelation algorithm will be discussed. Models for the decorrelation quantization error will be studied. In the previous chapter, the decorrelation algorithm was implemented in real-time using the SPROC digital signal processing microprocessor. As discussed, the SPROC chip is a 24-bit, fixed-point arithmetic machine. Since fixed-point arithmetic is used in all implementations in the SPROC environment, it is within this context that the LMS and decorrelation algorithms will be considered. To facilitate this study, the effects of finite-precision on the decorrelation algorithm will be compared in relation to the LMS algorithm, for which much work on finite-precision modelling has been done [4].

3.1 Quantization Effects in Adaptive Algorithms

In implementing an algorithm in a finite-precision environment, there are several areas in which the effects of finite-precision can introduce errors. Suppose that a number can be represented by b bits. If two b-bit numbers are multiplied together, the product will be a number which is $2b$ bits long. Therefore, for fixed-point arithmetic, the product of two fixed-point fractions will remain a fraction, but to maintain a limited register length the least-significant bits of the result must be either rounded or truncated. Furthermore, the addition of two fixed-point fractions will not need rounding or truncation. However, the magnitude of the resulting sum may exceed the maximum allowable fixed-point number. This overflow can be prevented by properly scaling the incoming data. Note that in the SPROC environment, all numbers and results of arithmetic operations are rounded to fit the 24-bit registers. Consequently, only the quantizing errors due to rounding will be studied.

Specifically, considering the implementation of adaptive algorithms in finite-precision environments such as SPROC, the error in the steady-state output of either the LMS or decorrelation algorithms due to the effects of finite word-lengths can be confined to three areas. First, there will be an error due to the quantization of the input data. In the implementation of the decorrelation algorithm considered in this thesis, since the input data is binary (consisting of ± 1, each occurring with equal probability), this quantization error will not present any appreciable problem. Second, an error can occur due to the rounding of the arithmetic (summation) operations in calculating the output of the equalizer. Since only fixed-point numbers are being added, the result will also be a fixed-point number. However, care must be taken to prevent overflow from occurring in these arithmetic operations. In the SPROC environment, for example, this is a critical issue in any design implementation. Overflow can catastrophically affect the performance of any algorithm in that convergence of the tap weights will never occur. A scaling factor for the decorrelation algorithm will now be derived so that errors due to overflow can be neglected in the forthcoming quantization model.

The requirement for scaling in the SPROC environment, and, therefore, any similar fixed-point environment, for an adaptive equalizer has been found to be dependent on the channel. As noted earlier, the cascade of the transmit, channel, and receive filters is modelled as an FIR filter with impulse response

$$h(n) = \sum_{k=0}^{N} h(k)\delta(n-k) \tag{3.1}$$

where $\delta(\cdot)$ is the Kronecker delta and $h(0)=1$ can be taken without loss of generality. The input to the channel, $\mathbf{I}(n)$, is assumed to be a binary white sequence with zero-mean. Therefore, the output of the channel, $x(n)$, is given by

$$x(n) = \sum_{k=0}^{N} h(k)I(n-k) \ . \qquad (3.2)$$

Given the representation of fixed-point numbers in the SPROC environment, overflow will be prevented so long as the channel output lies within the following limits:

$$-2.0 \leq x(n) < 2.0 \ . \qquad (3.3)$$

Therefore, for the magnitude of the channel output

$$0.0 \leq |x(n)| < 2.0 \ . \qquad (3.4)$$

Since

$$|x(n)| = \left|\sum_{k=0}^{N} h(k)I(n-k)\right| \leq \sum_{k=0}^{N} |h(k)|\,|I(n-k)| \qquad (3.5)$$

the channel output is bounded by

$$0 \leq \sum_{k=0}^{N} |h(k)|\,|I(n-k)| < 2.0 \ . \qquad (3.6)$$

As has been done in the literature [13], to a first-order approximation, $|I(n-k)|$ can be replaced by its RMS value, I_{rms}. Since the input sequence is binary (polar), the RMS value of the input data stream is unity. Accordingly,

$$0 \leq \sum_{k=0}^{N} |h(k)| < 2.0 \ . \qquad (3.7)$$

Taking $h(0)=1$,

$$0 \leq \sum_{k=1}^{N} |h(k)| < 1.0 \ . \qquad (3.8)$$

Now, suppose that the sum of the channel weights does not meet the bound given in Equation 3.8. Then a scaling factor, β, must be included in the bound to ensure compliance. Therefore,

$$0 \leq \beta \sum_{k=1}^{N} |h(k)| < 1.0 \ . \qquad (3.9)$$

Consequently, an upper bound on the scaling factor for the SPROC, or any similar finite-precision, implementation of the decorrelation algorithm is

$$\beta < \frac{1.0}{\sum_{k=1}^{N} |h(k)|} . \tag{3.10}$$

This is a necessary and sufficient condition to prevent overflow. It should again be emphasized that although this bound was derived in lieu of the decorrelation algorithm implemented using SPROC, this bound will apply to any adaptive equalizer implemented in a similar fixed-point environment using the aforementioned channel and input. It should also be mentioned that through experimentation it was found that the use of a scaling factor which was about one-half that calculated in Equation 3.10 worked best.

With the channel output assumed to be properly scaled so as to prevent overflow, the third possible area in which quantization errors can result is in the calculation of the tap weight coefficients. The error due to the deviation of the equalizer's coefficients from the values taken when infinite precision is used will directly affect the performance of the adaptive algorithm. It is for this reason that the quantization effects in the calculation of the tap weights will be studied for both the LMS and decorrelation algorithms.

3.2 Model and Statistical Properties of the Quantization Error

As discussed previously, when two fixed-point numbers are multiplied together, the result must be rounded to fit the designated register length. In the following discussion, each data sample and filter coefficient is to be considered as being represented by B bits, including sign (in SPROC, for example, 23 bits plus one for sign). Therefore, the least significant bit (LSB) is 2^{-B}. Note that 2^{-B} is referred to as the width of quantization, Δ, since the fixed-point numbers are quantized in steps of 2^{-B} [34]. Since rounding involves choosing the closest quantization level, the maximum error has a magnitude of $\frac{1}{2} * 2^{-B} = 2^{-B-1}$. In order to study the effects of this rounding error, the quantization error is usually modelled as an additive noise to the unquantized value of the input quantity [34].

Denote $Q[x(n)]$ as the number after quantization and $x(n)$ as the number before quantization. Therefore, the rounding error, $e_r(n)$, is given by

$$e_r(n) = Q[x(n)] - x(n) \tag{3.11}$$

The model of the quantization process is shown in Figure 3.1.

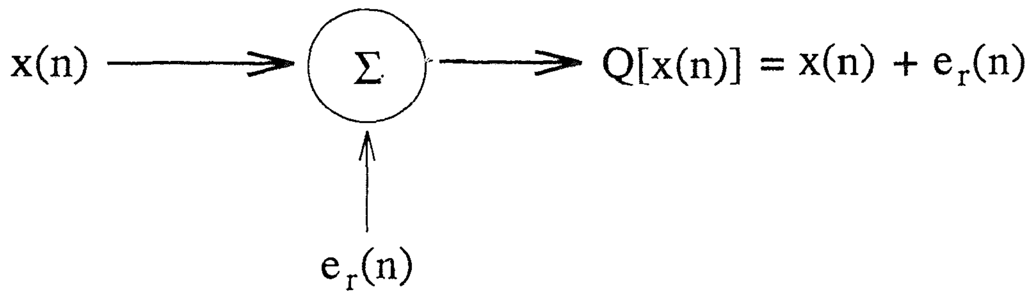

Figure 3.1 Model of the Quantization Process.

Since $e_r(n)$ can be either positive or negative, the rounding error is in the range

$$-\frac{1}{2} * 2^{-B} < e_r(n) < \frac{1}{2} * 2^{-B} \tag{3.12}$$

The quantization process is depicted graphically in Figure 3.2.

It is assumed that if a number lies exactly in the middle of a quantization level, the number is rounded up. It can be readily seen from Figure 3.2 that one of the effects of the quantization error is the introduction of nonlinearities in the system, which, in some cases, can cause the system to become unstable [34]. The quantity $x(n)$ can fall into any of the quantizer levels of Figure 3.2. Therefore, the rounding error is usually modelled as a random variable that is uniformly distributed in the ranges given by Equation 3.12 [34]. With $\Delta = 2^{-B}$, the probability density function of the quantization error is shown in Figure 3.3. With reference to Figure 3.3, the

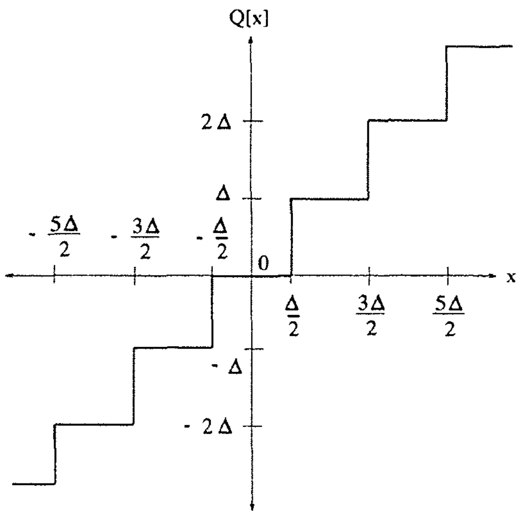

Figure 3.2 Graphical Representation of the Quantization Process.

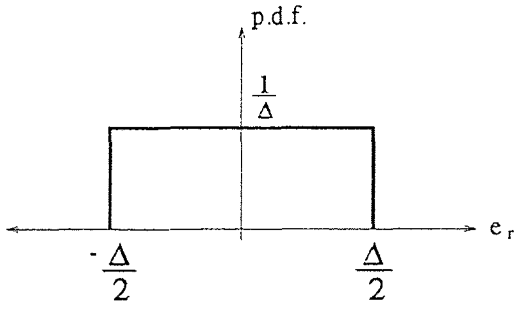

Figure 3.3 Probability Density Function of the Quantization Error.

mean of the error due to rounding is

$$m_e = \mathcal{E}\left[e_r(n)\right] = \int_{-\Delta/2}^{\Delta/2} e p_{e_r(n)}(e)\, de = \frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2} e\, de = \frac{1}{\Delta}\frac{e^2}{2}\Big]_{-\Delta/2}^{\Delta/2} = 0 \ . \qquad (3.13)$$

The variance of the error due to rounding is

$$\begin{aligned} \sigma_e^2 &= \mathcal{E}\left[(e_r(n) - m_e)^2\right] = \mathcal{E}\left[e_r^2(n)\right] = \int_{-\Delta/2}^{\Delta/2} e^2 p_{e_r(n)}(e)\, de = \frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2} e^2\, de \\ &= \frac{1}{\Delta}\frac{e^3}{3}\Big]_{-\Delta/2}^{\Delta/2} = \frac{\Delta^2}{12} = \frac{2^{-2B}}{12} \ . \qquad (3.14) \end{aligned}$$

As a summary of the previous discussion, it should be noted that in the literature (see [34], [37]) several other assumptions on $e_r(n)$ are made in order to simplify the analysis of the effect of the quantization error due to rounding. These assumptions are:

1. The error sequence $e_r(n)$ is a stationary white noise sequence. In other words, the random variables of the error process are uncorrelated.
2. The error sequence $e_r(n)$ is uncorrelated with the signal sequence $x(n)$.

These statements are made in conjunction with the previous assumption that the error sequence $e_r(n)$ is uniformly distributed in the range given by Equation 3.12. Although these assumptions do not hold for many cases, the assumptions do hold if the signal is sufficiently complex and the quantization steps are sufficiently small so that the signal sequence $x(n)$ traverses several quantization levels between successive samples [34].

3.3 Performance of the Decorrelation Algorithm in the Presence of Roundoff Errors: Digital Residual Error

One of the most important aspects of the limited precision implementation of an adaptive filter is the numerical accuracy of the algorithm. The accuracy of a given adaptive algorithm, implemented in a finite-precision environment, is given in terms of the magnitude of the deviation from infinite-precision performance [6]. Therefore, the smaller the deviation, the more accurate the implementation. Numerical accuracy is strongly a function of the number of bits used in the implementation [6]. To compare the numerical accuracy of the LMS and decorrelation algorithms, it will be necessary to introduce the concept of digital residual error.

Recall that the tap weight update equation of the LMS algorithm is given by

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu\mathbf{X}(n)e(n) \ . \tag{3.15}$$

In a finite-precision environment, the tap weights will stop adapting when the correction term, $\mu\mathbf{X}(n)e(n)$, is less than one-half the LSB of the registers. This is referred to as the *stalling phenomenon* of the LMS algorithm [18]. As was stated earlier, it is assumed that numbers falling exactly in the middle of a quantization interval are rounded up. Referring to $n = n_0$ as the time the i^{th} tap stops adapting, the following inequality will hold

$$|\mu e(n_0)x(n_0 - i)| < \frac{1}{2} * 2^{-B} = 2^{-B-1} \tag{3.16}$$

where B is the total number of bits used to represent a number. Therefore,

$$|e(n_0)| < \frac{2^{-B-1}}{\mu\,|x(n_0 - i)|} \equiv e_d(\mu) \ . \tag{3.17}$$

The term $e_d(\mu)$ is called the digital residual error (DRE) [13]. To a first approximation, $|x(n_0 - i)|$ can be represented by its RMS value, X_{rms} [13]. Accordingly,

$$|e(n_0)| < \frac{2^{-B-1}}{\mu X_{rms}} \equiv e_d(\mu) \ . \tag{3.18}$$

The DRE provides an *upper bound* on the magnitude of the deviation from the ideal (infinite-precision). As can be seen from Equation 3.18, the DRE for the LMS algorithm is inversely proportional to the step size parameter, μ. Therefore, if tap weight adaption ceases due to digital effects, the step size should be made as large as possible (while still guaranteeing convergence) in order to minimize the DRE.

An equivalent result can also be derived for the decorrelation algorithm. Given the tap weight update expression,

$$\mathbf{W}(n + 1) = \mathbf{W}(n) + \mu y(n)\mathbf{Y}(n) \tag{3.19}$$

the decorrelation algorithm will stall when the correction term, $\mu y(n)\mathbf{Y}(n)$, is less than one-half the LSB of the register. Referring to $n = n_0$ as the time the i^{th} tap stops adapting, the following inequality for the decorrelation algorithm will hold:

$$|y(n_0)y(n_0 - i)| < \frac{2^{-B-1}}{\mu} \equiv e_d(\mu) \ . \tag{3.20}$$

Note that since X_{rms} is bounded by unity, the expressions for the DRE of the LMS and decorrelation algorithms are equivalent. To make the finite-precision analysis of the LMS and decorrelation algorithms more tractable, the digital residual error will offer a reasonable means of comparison of the finite-precision performance (numerical accuracy) of the LMS and decorrelation algorithms.

3.3.1 Simulation Results for the Digital Residual Error

In order to experimentally confirm the previous findings concerning the performance of the LMS and decorrelation algorithms in a finite-precision environment, a model of the rounding (quantization) process occurring in the respective tap weight update equations was developed. This model was used to simulate the quantization process in a fixed-point environment, such as SPROC, via a computer program that would allow the user to decrease the word-length used by the equalizer. The SPROClab development system was not used to study finite-precision effects for the simple reason that the system is already quantized to 24 bits. Consequently, if the bit-length used in the SPROC-based equalizer was lowered, one would be studying the quantization of quantized numbers, not the quantization of infinite precision numbers. Since the computer program allows the use of double precision arithmetic (effectively infinite precision), all experimental results (except where noted) are gathered from computer-based simulations. Results will be referred back to the SPROC-based implementations when necessary.

As discussed previously, with proper scaling assumed, a finite-precision environment will have the greatest effect on the tap weights. Therefore, any quantization effects that occur as a consequence of the calculation of the tap weight correction term ($\mu\mathbf{X}(n)e(n)$ and $\mu y(n)\mathbf{Y}(n)$, respectively) would be caused by the corresponding multiplications that are rounded to fit the B-bit length of the registers. The process to be studied can be represented mathematically for the LMS and decorrelation algorithms, respectively, as follows:

$$\begin{aligned}\hat{\mathbf{W}}(n+1) &= \hat{\mathbf{W}}(n) + Q\left[\mu \mathbf{X}(n) e(n)\right] \\ \mathbf{W}_{\mathbf{M}}(n+1) &= \mathbf{W}_{\mathbf{M}}(n) + Q\left[\mu y(n) \mathbf{Y}_{\mathbf{M}}(n)\right]\end{aligned}$$

where the process $Q[\cdot]$ is defined by Equation 3.11.

In the simulations, a program was written to implement the LMS and decorrelation algorithms in a decision feedback configuration. The tap weight correction terms were rounded according to the following algorithm:

$$c_q = \frac{INT\left[c * 2^k + 0.5\right]}{2^k} \tag{3.21}$$

where c is the unquantized (infinite precision) tap weight coefficient, c_q is the quantized tap weight coefficient, and k is the desired bit-length of the registers. The $INT[\cdot]$ notation indicates the rounding to the nearest integer. All arithmetic (summation) operations in the program were left unquantized since, as noted previously, proper scaling is assumed. In the following simulations, the bit-length used in computing the correction term will be decreased in order to determine the digital residual error of the LMS and decorrelation equalizers. The equalizer being studied is a simple one-tap implementation using a channel with a post-cursor of h_1=0.67. The experimental results will be compared to the theoretical limits as expressed in Equations 3.18 and 3.20. The step size parameter, μ, will then be changed in order to determine the variation in the magnitude of the DRE and its compliance to the aforementioned limits.

For ease of comparison, the results of the LMS DFE have been summarized in Table 3.1. For the implementation of the LMS DFE, it was found that the tap weight, w_1, was able to converge and settle on a particular value. The experimental DRE is then the magnitude of the deviation of the converged tap weight from the actual value of $w_1 = 0.67$. As can be seen in Table 3.1, the theoretical DRE provides a good upper bound on the finite-precision performance (accuracy) of the LMS DRE.

As the bit-length is decreased, the DRE of the LMS DFE increases. Notice that at very small bit lengths (less than ten bits), the DRE can indeed be made smaller by increasing the step size. This agrees well with theory.

The results of the decorrelation DFE have been summarized in Table 3.2. It should be stated that the tap weight of the decorrelation DFE performs differently than the LMS, due to the nature of the decorrelation algorithm. The decorrelation tap weight, w_1, does not settle on a particular value. Rather, it exhibits a random fluctuation around $w_1 = 0.67$, as discussed in Chapter 2. Therefore, to study the numerical accuracy of the decorrelation algorithm in a finite-precision environment, several provisions had to be made. First, the decorrelation DFE was run for 15,000 iterations. To determine numerical accuracy, given a particular bit-length, all tap weight values that fell within a given interval around 0.67 were summed and their average value determined. For all bit lengths in Table 3.2, this interval was $0.67 \pm (1000/\Delta)$. For 24 bits, the interval was $0.67 \pm (10000/\Delta)$. The desired DRE for the decorrelation algorithm, of Table 3.2, was then computed using a Monte-Carlo averaging of 100 independent trials of the experiment. The DRE was then calculated as the magnitude of the difference between this *average* DRE and the actual tap weight value. Notice in Table 3.2 that even with infinite precision, the decorrelation algorithm still exhibits a deviation from the actual tap weight value of $w_1 = 0.67$.

From Table 3.2, it can be seen that the decorrelation algorithm performs in a manner very similar to the LMS. As the bit-length of the environment is decreased the DRE subsequently increases. Furthermore, the theoretical DRE also provides a good upper bound on the residual error. Notice that at low bit-lengths (less than ten bits), the DRE of the decorrelation algorithm, like that of the LMS, can be decreased by increasing the step size. This agrees well with theory.

By comparing the results of Tables 3.1 and 3.2, it can be seen that the decorrelation algorithm is at least as robust as the LMS algorithm in the presence of a finite-precision environment. The effect of the quantization error can, in both algorithms, be reduced by increasing the step size. This concept seems to contradict the conventional notion that the step size should be decreased in order to improve the performance of the algorithm [6]. However, in a digital implementation, it is necessary to achieve a balance between performance and numerical accuracy that is not usually required in an infinite-precision environment.

Table 3.1 Digital Residual Errors of the LMS DFE.

WORD LENGTH (BITS)	STEP SIZE	DRE (experimental)	DRE (theoretical)
∞	0.1	0.0	0.0
24	0.1	2.8133392×10^{-7}	2.9802322×10^{-7}
22	0.1	1.1157989×10^{-6}	1.1920928×10^{-6}
20	0.1	4.6920776×10^{-6}	4.7683715×10^{-6}
18	0.1	1.7089843×10^{-5}	1.9073486×10^{-5}
16	0.1	6.2866211×10^{-5}	7.6293945×10^{-5}
14	0.1	2.6123046×10^{-4}	3.0517578×10^{-4}
12	0.1	0.0010546875	0.0012207031
10	0.1	0.003984375	0.0048828125
8	0.1	0.01765625	0.01953125
8	0.05	0.0371875	0.0390625
8	0.5	0.00203125	0.00390625
6	0.1	0.07625	0.078125
6	0.05	0.154375	0.15625
6	0.5	0.01375	0.015625

Table 3.2 Digital Residual Errors of the Decorrelation DFE.

WORD LENGTH (BITS)	STEP SIZE	DRE (experimental)	DRE (theoretical)
∞	0.001	1.8897554×10^{-5}	0.0
24	0.001	3.7844416×10^{-5}	2.9802322×10^{-5}
22	0.001	3.2527589×10^{-5}	1.1920928×10^{-4}
20	0.001	5.4807156×10^{-5}	4.7683715×10^{-4}
18	0.001	2.1144686×10^{-4}	1.9073486×10^{-3}
16	0.001	1.4475860×10^{-3}	7.6293945×10^{-3}
14	0.001	7.4792925×10^{-3}	3.0517578×10^{-2}
12	0.001	0.02309694	0.12207035
10	0.001	0.17090000	0.48828125
8	0.001	0.39714725	1.953125
8	0.005	0.0134619	0.390625
8	0.01	0.006340341	0.1953125
6	0.001	0.67	7.8125
6	0.005	0.2429903	1.5625
6	0.01	0.0116528	0.78125

CHAPTER 4

RAPIDLY CONVERGING ALGORITHMS FOR DECORRELATION

In the design of blind, adaptive equalizers, there are many criteria by which the equalizers are judged and compared. The two of most interest to this work are complexity and speed of convergence. In the previous chapter, it was shown that the decorrelation-based DFE converges, in the mean, to the correct tap weights. The DFE of Figure 2.6 offers implementational simplicity at the cost of the rate at which the equalizer converges. The complexity of this equalizer, and others like it, is linear in that the number of operations (multiplications and divisions) that are needed to update the tap weights is proportional to M, where M is the number of tap weights. It is possible to dramatically increase the rate of convergence of the equalizer by increasing the complexity of the implementation. This particular class of Kalman, or RLS, algorithms has a complexity proportional to M^2. However, it is possible to retain the speed of the Kalman algorithm, while maintaining an order of M complexity. Two of the members of a computationally efficient class of algorithms that will be dealt with in this chapter are referred to as the fast Kalman and FTF algorithms. It will be shown in this chapter how the decorrelation algorithm can be formulated and implemented in terms of an RLS equalizer and how this theory forms the basis for the more computationally efficient fast Kalman and FTF type equalizers.

4.1 The Kalman Algorithm for Decorrelation

To improve the convergence speed of the classical LMS equalizer, Godard [14] suggested the use of the Kalman algorithm with equalization. The Kalman algorithm, or RLS algorithm, minimizes the time-average exponentially weighted squared error, where the error was defined as the difference between some desired signal (data) and its estimate (see Figure 2.5). The exponential weighting factor was used to allow the filter coefficients to adapt to the time-varying statistical characteristics of the data. In other words, the most recent data points are given a heavier *weight* than the past samples, which are eventually *forgotten*.

Referring to the blind DFE structure of Figure 4.1, the input to the slicer is given by

$$y(n) = x(n) - \sum_{k=0}^{n} w_k(n)\hat{y}(n-k), \tag{4.1}$$

where $x(n)$ is the input to the equalizer (output of the channel) at time n and the $w_k(n)$'s are the equalizer's weights. Equation 4.1 can be written in matrix form as

$$y(n) = x(n) - \hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{W}_{\mathrm{M}}(n), \tag{4.2}$$

where $\hat{\mathbf{Y}}'_{\mathrm{M}}(n)=[\hat{y}_{n-1}, \hat{y}_{n-2}, \ldots, \hat{y}_{n-M}]$ and $\mathbf{W}'=[w_1(n), w_2(n), \ldots, w_M(n)]$.

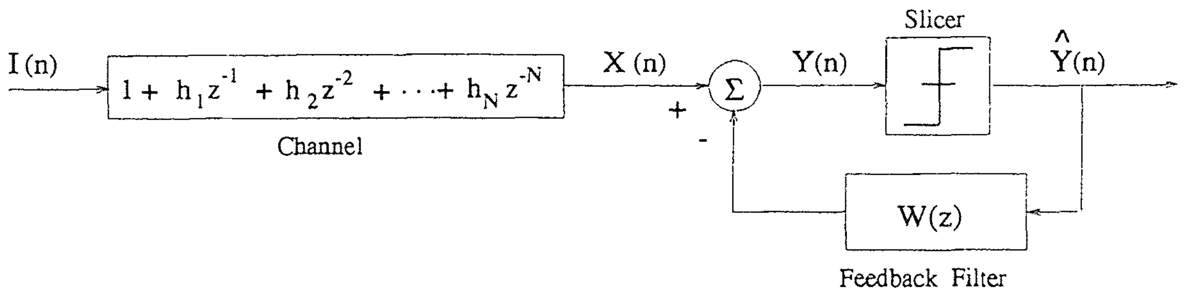

Figure 4.1 Decision Feedback Equalizer.

Using a philosophy similar to that expressed in [14], it was shown in [24] that is possible to use the time-average exponentially weighted correlations as the cost function to be minimized. It should be emphasized that this is possible due to the inherent simplicity of the error function of the decorrelation algorithm. Therefore, instead of decorrelating instantaneous realizations of the input to the slicer of

Figure 4.1, the time-average weighted input is decorrelated, *i.e.*, solve:

$$\sum_{k=0}^{n} \lambda^{n-k} y(n)\mathbf{Y}_{\mathrm{M}}(\mathbf{k}) = 0$$

where $\mathbf{Y}'_{\mathrm{M}}(k) = [y_{k-1}, y_{k-2}, \ldots, y_{k-M}]$ and λ is a positive constant close to, but less than, one. The quantity $(1-\lambda)^{-1}$ can be considered the *memory* of the algorithm, where λ=1 corresponds to infinite memory. An alternative approach to the exponential weighting would be to use a finite-duration sliding window with uniform weighting over the window length [37]. However, this method will not be considered here. Therefore, proceeding as in [24], substituting for $y(n)$ from Equation 4.2 and setting the weighted correlation time average to zero results in

$$\sum_{k=0}^{n} \lambda^{n-k} \mathbf{Y}_{\mathrm{M}}(k) \left(x(k) - \hat{\mathbf{Y}}'_{\mathrm{M}}(k)\mathbf{W}_{\mathrm{M}}(n) \right) = 0 \ .$$

Expanding the above equation and collecting terms yields

$$\sum_{k=0}^{n} \lambda^{n-k} \mathbf{Y}_{\mathrm{M}}(k)\hat{\mathbf{Y}}'_{\mathrm{M}}(k)\mathbf{W}_{\mathrm{M}}(n) = \sum_{k=0}^{n} \lambda^{n-k} x(k)\mathbf{Y}_{\mathrm{M}}(k) \ . \tag{4.3}$$

Equation 4.3 can be written in matrix form as

$$\mathbf{W}_{\mathrm{M}}(n) = \mathbf{R}^{-1}_{\mathrm{M,M}}(n)\mathbf{D}_{\mathrm{M}}(n) \tag{4.4}$$

where the cross-correlation matrix of the vector of inputs and vector of outputs of the slicer is given by

$$\mathbf{R}_{\mathrm{M,M}}(n) \triangleq \sum_{k=0}^{n} \lambda^{n-k} \mathbf{Y}_{\mathrm{M}}(k)\hat{\mathbf{Y}}'_{\mathrm{M}}(k) \tag{4.5}$$

and the cross-correlation matrix of the current output of the channel and the vector of inputs to the slicer is given by

$$\mathbf{D}_{\mathrm{M}}(n) \triangleq \sum_{k=0}^{n} \lambda^{n-k} x(k)\mathbf{Y}_{\mathrm{M}}(k). \tag{4.6}$$

Equation 4.4 involves the inversion of an $M \times M$ matrix, $\mathbf{R}_{\mathrm{M,M}}(n)$. In the following derivation, it will be convenient to develop a recursive relation for the cross-correlation matrix, $\mathbf{R}_{\mathrm{M,M}}(n)$. Isolating the term corresponding to k=n from the rest of the summation on the right-hand side (RHS) of Equation 4.5,

$$\mathbf{R}_{\mathrm{M,M}}(n) = \lambda \left[\sum_{k=0}^{n-1} \lambda^{n-1-k} \mathbf{Y}_{\mathrm{M}}(k) \hat{\mathbf{Y}}'_{\mathrm{M}}(k) \right] + \mathbf{Y}_{\mathrm{M}}(n) \hat{\mathbf{Y}}'_{\mathrm{M}}(n) \quad . \tag{4.7}$$

However, by the definition of Equation 4.5, the expression inside the brackets of the above equation is equal to the *old* cross-correlation matrix, $\mathbf{R}_{\mathrm{M,M}}(n-1)$. Therefore, the recursion for updating the value of the cross-correlation matrix is

$$\mathbf{R}_{\mathrm{M,M}}(n) = \lambda \mathbf{R}_{\mathrm{M,M}}(n-1) + \mathbf{Y}_{\mathrm{M}}(n) \hat{\mathbf{Y}}'_{\mathrm{M}}(n) \quad . \tag{4.8}$$

Similarly, a recursive equation can be derived for the cross-correlation matrix, $\mathbf{D}_{\mathrm{M}}(n)$, of Equation 4.6

$$\mathbf{D}_{\mathrm{M}}(n) = \lambda \mathbf{D}_{\mathrm{M}}(n-1) + x(n) \mathbf{Y}_{\mathrm{M}}(n) \quad . \tag{4.9}$$

It is known that for any nonsingular matrix, $\mathbf{A}$, and vectors $\mathbf{u}$ and $\mathbf{v}$, the following definition of the inverse of a matrix holds (assuming that $\mathbf{A} + \mathbf{u}\mathbf{v}'$ is nonsingular) [21]:

$$(\mathbf{A} + \mathbf{u}\mathbf{v}')^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}'\mathbf{A}^{-1}}{1 + \mathbf{v}'\mathbf{A}^{-1}\mathbf{u}} \quad . \tag{4.10}$$

Therefore, using Equation 4.10 in Equation 4.8, it is possible to derive a recursive formula for $\mathbf{R}^{-1}_{\mathrm{M,M}}(n)$ as follows:

$$\mathbf{R}^{-1}_{\mathrm{M,M}}(n) = \frac{1}{\lambda} \left(\mathbf{R}^{-1}_{\mathrm{M,M}}(n-1) - \frac{\mathbf{R}^{-1}_{\mathrm{M,M}}(n-1) \mathbf{Y}_{\mathrm{M}}(n) \hat{\mathbf{Y}}'_{\mathrm{M}}(n) \mathbf{R}^{-1}_{\mathrm{M,M}}(n-1)}{\lambda + \hat{\mathbf{Y}}'_{\mathrm{M}}(n) \mathbf{R}^{-1}_{\mathrm{M,M}}(n-1) \mathbf{Y}_{\mathrm{M}}(k)} \right) \tag{4.11}$$

For convenience of computation, let

$$\mathbf{P}_{\mathrm{M}}(n) \triangleq \mathbf{R}^{-1}_{\mathrm{M,M}}(n). \tag{4.12}$$

It is also convenient to define the $M \times 1$ vector, $\mathbf{K}_{\mathrm{M}}(n)$, referred to as the Kalman gain vector, as

$$\mathbf{K}_{\mathrm{M}}(n) = \frac{1}{\lambda + \mu_M(n)} \mathbf{P}_{\mathrm{M}}(n-1) \mathbf{Y}_{\mathrm{M}}(n) \tag{4.13}$$

where the scalar $\mu_M(n)$ is given by

$$\mu_M(n) = \hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{R}^{-1}_{\mathrm{M,M}}(n-1)\mathbf{Y}_{\mathrm{M}}(n) \ . \tag{4.14}$$

Using the previous definitions, Equation 4.11 can be written as

$$\mathbf{P}_{\mathrm{M}}(n) = \frac{1}{\lambda}\left(\mathbf{P}_{\mathrm{M}}(n-1) - \mathbf{K}_{\mathrm{M}}(n)\hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{P}_{\mathrm{M}}(n-1)\right) \ . \tag{4.15}$$

In comparison with traditional Kalman filter theory, Equation 4.15 can be considered the Riccati equation for the RLS with the decorrelation algorithm [18]. Using Equations 4.4 and 4.12, it is possible to write

$$\mathbf{W}_{\mathrm{M}}(n) = \mathbf{P}_{\mathrm{M}}(n)\mathbf{D}_{\mathrm{M}}(n) \ . \tag{4.16}$$

It is now necessary to develop an expression that solves Equation 4.16 recursively. Therefore, substituting Equation 4.6 for $\mathbf{D}_{\mathrm{M}}(n)$ in the above equation results in

$$\mathbf{W}_{\mathrm{M}}(n) = \lambda\mathbf{P}_{\mathrm{M}}(n)\mathbf{D}_{\mathrm{M}}(n-1) + x(n)\mathbf{P}_{\mathrm{M}}(n)\mathbf{Y}_{\mathrm{M}}(n) \ .$$

Substitute Equation 4.15 into only the first term on the RHS of the above equation.

$$\begin{aligned}
\mathbf{W}_{\mathrm{M}}(n) &= \lambda\left[\frac{1}{\lambda}\left(\mathbf{P}_{\mathrm{M}}(n-1) - \mathbf{K}_{\mathrm{M}}(n)\hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{P}_{\mathrm{M}}(n-1)\right)\right]\mathbf{D}_{\mathrm{M}}(n-1) \\
&\qquad + x(n)\mathbf{P}_{\mathrm{M}}(n)\mathbf{Y}_{\mathrm{M}}(n) \\
&= \mathbf{P}_{\mathrm{M}}(n-1)\mathbf{D}_{\mathrm{M}}(n-1) - \mathbf{K}_{\mathrm{M}}(n)\hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{P}_{\mathrm{M}}(n-1)\mathbf{D}_{\mathrm{M}}(n-1) \\
&\qquad + x(n)\mathbf{P}_{\mathrm{M}}(n)\mathbf{Y}_{\mathrm{M}}(n) \\
&= \mathbf{W}_{\mathrm{M}}(n-1) - \mathbf{K}_{\mathrm{M}}(n)\hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{W}_{\mathrm{M}}(n-1) + x(n)\mathbf{P}_{\mathrm{M}}(n)\mathbf{Y}_{\mathrm{M}}(n) \ .
\end{aligned}$$

It will be shown in the derivation of the fast Kalman algorithm that the Kalman gain vector, $\mathbf{K}_{\mathrm{M}}(n)$, equals $\mathbf{P}_{\mathrm{M}}(n)\mathbf{Y}_{\mathrm{M}}(n)$. Using this fact in the last equality of the above equation

$$\begin{aligned}
\mathbf{W}_{\mathrm{M}}(n) &= \mathbf{W}_{\mathrm{M}}(n-1) - \mathbf{K}_{\mathrm{M}}(n)\hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{W}_{\mathrm{M}}(n-1) + x(n)\mathbf{K}_{\mathrm{M}}(n) \\
&= \mathbf{W}_{\mathrm{M}}(n-1) + \mathbf{K}_{\mathrm{M}}(n)\left(x(n) - \hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{W}_{\mathrm{M}}(n-1)\right) \ . \qquad (4.17)
\end{aligned}$$

Define the *a priori* estimation error for decorrelation as

$$\eta_M(n) = x(n) - \hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{W}_{\mathbf{M}}(n-1) \tag{4.18}$$

since it has a form similar to that expressed in [18]. Consequently, substituting Equation 4.18 into Equation 4.17 results in the desired recursive relationship for the tap weight vector

$$\mathbf{W}_{\mathbf{M}}(n) = \mathbf{W}_{\mathbf{M}}(n-1) + \eta_M(n)\mathbf{K}_{\mathbf{M}}(n). \tag{4.19}$$

The order that constitutes the time-average exponentially weighted decorrelation algorithm is summarized below:

$$\mathbf{\Gamma}_{\mathbf{M}}(n) = \hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{P}_{\mathbf{M}}(n-1) \tag{4.20}$$

$$\mu_M(n) = \mathbf{\Gamma}_{\mathbf{M}}(n)\mathbf{Y}_{\mathbf{M}}(n) \tag{4.21}$$

$$\mathbf{K}_{\mathbf{M}}(n) = \frac{\mathbf{P}_{\mathbf{M}}(n-1)\mathbf{Y}_{\mathbf{M}}(n)}{\lambda + \mu_M(n)} \tag{4.22}$$

$$\eta_M(n) = x(n) - \hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{W}_{\mathbf{M}}(n-1) \tag{4.23}$$

$$\mathbf{W}_{\mathbf{M}}(n) = \mathbf{W}_{\mathbf{M}}(n-1) + \eta_M(n)\mathbf{K}_{\mathbf{M}}(n) \tag{4.24}$$

$$\tilde{\mathbf{P}}_{\mathbf{M}}(n-1) = \mathbf{K}_{\mathbf{M}}(n)\mathbf{\Gamma}_{\mathbf{M}}(n) \tag{4.25}$$

$$\mathbf{P}_{\mathbf{M}}(n) = \frac{1}{\lambda}\left(\mathbf{P}_{\mathbf{M}}(n-1) - \tilde{\mathbf{P}}_{\mathbf{M}}(n-1)\right) . \tag{4.26}$$

The algorithm can be initialized by setting $\mathbf{W}_{\mathbf{M}}(0)=\mathbf{0}_{\mathbf{M}}$ and $\mathbf{P}_{\mathbf{M}}(0)=\delta\mathbf{I}_{\mathbf{M}}$, where $\delta > 0$. The algorithm of the previous section will subsequently be referred to in this thesis as the Recursive Least Correlation algorithm (RLC), since it is based on the application of the decorrelation algorithm to a recursive-least-type criterion [24].

The increased rate of convergence of the blind, adaptive decorrelation DFE based on the Kalman algorithm compared with the conventional decorrelation blind DFE was shown in [24]. The RLC will be used as a measure by which the rate of convergence of other rapidly converging algorithms may be gauged. Note that the increased rate of convergence of the modified Kalman/Godard algorithm comes at a cost of greater complexity, since the $M \times M$ matrix $\mathbf{P}_{\mathbf{M}}(n)$ must be adapted and stored at each iteration. A measure of a particular algorithm's complexity is usually given in terms of the number of multiplications and divisions per iteration (MADPI) that are needed in order to update the tap weight vector [7]. Therefore, the RLC algorithm of Equations 4.20 through 4.26 requires approximately $4M^2 + 4M + 2$ MADPI. Thus, the RLC algorithm is said to have an order M^2 (*i.e.*, $\mathcal{O}(M^2)$) complexity. As a consequence of this trade-off between complexity and speed, there has been a heavy emphasis on the search for rapidly converging, less complex structures. The results derived in this section will form the basis by which the more computationally efficient fast Kalman algorithm for decorrelation (FRLC) and fast transversal filter (FTF) with decorrelation will be derived.

4.2 The Fast Kalman Algorithm for Decorrelation

In the last section, it was shown that by minimizing the time-average weighted correlations, the decorrelation algorithm (in a decision feedback configuration) could be implemented using a Kalman, or recursive least, algorithm, thereby significantly improving the rate of convergence of the equalizer. However, a disadvantage of the RLC algorithm, as with all recursive least-squares algorithms, is its complexity. For the RLC, $\mathcal{O}(M^2)$ operations (multiplications and divisions) must be performed per iteration.

To reduce the complexity of the RLC algorithm, while still maintaining its high rate of convergence, the decorrelation criterion will be applied to the more computationally efficient family of fast transversal-based algorithms. The first of these algorithms to be studied is the fast Kalman algorithm, originally derived by Falconer and Ljung [10]. The mathematical basis of the resulting fast recursive least algorithms is the exploitation of the *shifting property* inherent in most sequential estimation (prediction) problems. In equalization, this property expresses the fact that at each iteration the number of new samples entering and old samples leaving the equalizer is not M, but a much smaller integer p. For this particular equalizer application, $p = 1$. In referring to Figure 4.1, this shifting property corresponds to the fact that at any time n the most recent output of the slicer, $\hat{y}(n)$, enters the feedback filter while the oldest slicer output, $\hat{y}(n - M + 1)$, leaves. The fast Kalman (fast RLS) algorithm will provide a means for the recursive updating of the Kalman gain vector of Equation 4.13 without explicit computation of the inverse correlation matrix of Equation 4.15.

4.2.1 Reformulation of the RLC Algorithm

In order to show how the decorrelation algorithm can be implemented in a fast Kalman form, it is necessary to reformulate the problem as expressed in the previous section. Referring to the previous section, in order to update the tap-weight coefficients using the RLC algorithm, it is necessary to solve the following equation at each iteration:

$$\mathbf{W}_{\mathbf{M}}(n) = \mathbf{R}^{-1}_{\mathbf{M},\mathbf{M}}(n)\mathbf{D}_{\mathbf{M}}(n) \tag{4.27}$$

In the previous section, it was shown that the inverse autocorrelation matrix, $\mathbf{R}^{-1}_{\mathbf{M},\mathbf{M}}(n)$, can be obtained recursively as

$$\mathbf{R}^{-1}_{\mathbf{M},\mathbf{M}}(n) = \frac{1}{\lambda}\left(\mathbf{R}^{-1}_{\mathbf{M},\mathbf{M}}(n-1) - \mathbf{K}_{\mathbf{M}}(n)\hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{R}^{-1}_{\mathbf{M},\mathbf{M}}(n-1)\right) \tag{4.28}$$

where $\mathbf{K}_{\mathrm{M}}(n)$ is the Kalman gain vector and $\hat{\mathbf{Y}}'_{\mathrm{M}}(n) = [\hat{y}_{n-1}, \hat{y}_{n-2}, \cdots, \hat{y}_{n-M}]$. Postmultiplying Equation 4.28 by $\mathbf{Y}_{\mathrm{M}}(n) = [y_{n-1}, y_{n-2}, \cdots, y_{n-M}]'$,

$$\begin{aligned} &\mathbf{R}^{-1}_{\mathrm{M,M}}(n)\mathbf{Y}_{\mathrm{M}}(n) = \\ &= \frac{1}{\lambda}\left(\mathbf{R}^{-1}_{\mathrm{M,M}}(n-1)\mathbf{Y}_{\mathrm{M}}(n) - \mathbf{K}_{\mathrm{M}}(n)\hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{R}^{-1}_{\mathrm{M,M}}(n-1)\mathbf{Y}_{\mathrm{M}}(n)\right). \end{aligned} \tag{4.29}$$

Recall that the Kalman gain vector was defined as (collecting terms)

$$\mathbf{R}^{-1}_{\mathrm{M,M}}(n-1)\mathbf{Y}_{\mathrm{M}}(n) = (\lambda + \mu_M(n))\,\mathbf{K}_{\mathrm{M}}(n). \tag{4.30}$$

Substituting Equations 4.14 and 4.30 into Equation 4.29 yields

$$\begin{aligned} \mathbf{R}^{-1}_{\mathrm{M,M}}(n)\mathbf{Y}_{\mathrm{M}}(n) &= \frac{1}{\lambda}\left((\lambda + \mu_M(n))\,\mathbf{K}_{\mathrm{M}}(n) - \mathbf{K}_{\mathrm{M}}(n)\mu_M(n)\right) \\ &= \mathbf{K}_{\mathrm{M}}(n). \end{aligned}$$

Therefore, as alluded to in the previous section, the Kalman gain vector can also be defined as

$$\mathbf{K}_{\mathrm{M}}(n) = \mathbf{R}^{-1}_{\mathrm{M,M}}(n)\mathbf{Y}_{\mathrm{M}}(n). \tag{4.31}$$

The following sections will provide a recursive algorithm to calculate the sequence of vectors

$$\mathbf{K}_{\mathrm{M}}(n) = \left[\sum_{k=0}^{n} \lambda^{n-k}\mathbf{Y}_{\mathrm{M}}(k)\hat{\mathbf{Y}}'_{\mathrm{M}}(k)\right]^{-1}\mathbf{Y}_{\mathrm{M}}(n) \tag{4.32}$$

using a number of operations proportional to M. The fast Kalman gain vector of Equation 4.32 can then be used in place of the Kalman gain vector in the algorithm presented in the previous section. The derivation of the algorithm is based on the work done in [29].

4.2.2 Derivation of the Algorithm

Rewriting Equation 4.31 as

$$\mathbf{R}_{\mathrm{M,M}}(n-1)\mathbf{K}_{\mathrm{M}}(n-1) = \mathbf{Y}_{\mathrm{M}}(n-1) \tag{4.33}$$

suppose that $\mathbf{K}_\mathbf{M}(j)$, $j = 0, \ldots, n-1$, have been determined. To find $\mathbf{K}_\mathbf{M}(n)$, define

$$\mathbf{R}_{\mathbf{M},\mathbf{M}}(n)\mathbf{K}_\mathbf{M}(n) = \mathbf{Y}_\mathbf{M}(n). \tag{4.34}$$

To take advantage of the shifting properties of $\hat{\mathbf{Y}}_\mathbf{M}(n)$ and $\mathbf{Y}_\mathbf{M}(n)$, it will be convenient to introduce the augmented vectors

$$\bar{\mathbf{Y}}(n) = \mathbf{Y}_{\mathbf{M+1}}(n) = \begin{bmatrix} y(n-1) \\ \mathbf{Y}_\mathbf{M}(n-1) \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_\mathbf{M}(n) \\ y(n-M-1) \end{bmatrix} \tag{4.35}$$

and

$$\bar{\hat{\mathbf{Y}}}(n) = \hat{\mathbf{Y}}_{\mathbf{M+1}}(n) = \begin{bmatrix} \hat{y}(n-1) \\ \hat{\mathbf{Y}}_\mathbf{M}(n-1) \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{Y}}_\mathbf{M}(n) \\ \hat{y}(n-M-1) \end{bmatrix} . \tag{4.36}$$

Consequently, the augmented cross-correlation matrix can be defined as

$$\bar{\mathbf{R}}(n) = \mathbf{R}_{\mathbf{M+1},\mathbf{M+1}}(n) = \sum_{k=0}^{n} \lambda^{n-k} \bar{\mathbf{Y}}(k)\bar{\hat{\mathbf{Y}}}'(k) . \tag{4.37}$$

It should be noted that the augmented cross-correlation matrix can be represented in matrix form as

$$\bar{\mathbf{R}}(n) = \begin{bmatrix} \pi(n) & \mathbf{S}'_\mathbf{M}(n) \\ \mathbf{Q}_\mathbf{M}(n) & \mathbf{R}_{\mathbf{M},\mathbf{M}}(n-1) \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\mathbf{M},\mathbf{M}}(n) & \check{\mathbf{Q}}_\mathbf{M}(n) \\ \check{\mathbf{S}}'_\mathbf{M}(n) & \check{\pi}(n) \end{bmatrix} \tag{4.38}$$

where

$$\pi(n) = \sum_{k=0}^{n} \lambda^{n-k} y(k-1)\hat{y}(k-1) \tag{4.39}$$

$$\mathbf{S}_\mathbf{M}(n) = \sum_{k=0}^{n} \lambda^{n-k} \hat{\mathbf{Y}}_\mathbf{M}(k-1) y(k-1) \tag{4.40}$$

$$\mathbf{Q}_\mathbf{M}(n) = \sum_{k=0}^{n} \lambda^{n-k} \mathbf{Y}_\mathbf{M}(k-1)\hat{y}(k-1) \tag{4.41}$$

$$\mathbf{R}_{\mathbf{M},\mathbf{M}}(n-1) = \sum_{k=0}^{n} \lambda^{n-k} \mathbf{Y}_\mathbf{M}(k-1)\hat{\mathbf{Y}}'_\mathbf{M}(k-1) \tag{4.42}$$

and

$$\check{\pi}(n) = \sum_{k=0}^{n} \lambda^{n-k} y(k-M-1)\hat{y}(k-M-1) \tag{4.43}$$

$$\check{\mathbf{S}}_\mathbf{M}(n) = \sum_{k=0}^{n} \lambda^{n-k} \hat{\mathbf{Y}}_\mathbf{M}(k) y(k-M-1) \tag{4.44}$$

$$\check{\mathbf{Q}}_{\mathbf{M}}(n) = \sum_{k=0}^{n} \lambda^{n-k} \mathbf{Y}_{\mathbf{M}}(k)\hat{y}(k-M-1) \tag{4.45}$$

and $\mathbf{R}_{\mathbf{M},\mathbf{M}}(n)$ is given by Equation 4.5. Using the previous notation, it is shown in Appendix A that Equations 4.33 and 4.34 are contained in the following expressions:

$$\bar{\mathbf{R}}(n)\begin{bmatrix} 0 \\ \mathbf{K}_{\mathbf{M}}(n-1) \end{bmatrix} = \begin{bmatrix} \rho(n) \\ \mathbf{Y}_{\mathbf{M}}(n-1) \end{bmatrix} = \bar{\mathbf{Y}}(n) + \begin{bmatrix} \rho(n) - y(n-1) \\ \mathbf{0}_{\mathbf{M}} \end{bmatrix} \tag{4.46}$$

and

$$\bar{\mathbf{R}}(n)\begin{bmatrix} \mathbf{K}_{\mathbf{M}}(n) \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{\mathbf{M}}(n) \\ \beta(n) \end{bmatrix} = \bar{\mathbf{Y}}(n) + \begin{bmatrix} \mathbf{0}_{\mathbf{M}} \\ \beta(n) - y(n-M-1) \end{bmatrix} \tag{4.47}$$

where $\mathbf{0}_{\mathbf{M}}$ is the M-dimensional null vector. The scalars $\rho(n)$ and $\beta(n)$ are given by

$$\rho(n) = \mathbf{S}'_{\mathbf{M}}(n)\mathbf{K}_{\mathbf{M}}(n-1) \quad \textit{and} \quad \beta(n) = \tilde{\mathbf{S}}'_{\mathbf{M}}(n)\mathbf{K}_{\mathbf{M}}(n) \ . \tag{4.48}$$

Before proceeding with the derivation, an important remark must be made concerning the augmented correlation matrix of Equation 4.38. In the *traditional* proofs of the fast Kalman algorithm, *i.e.*, those not based on the decorrelation criterion, the derivation of the fast Kalman algorithm is much simpler (see [10], [25], and [35]). The reason for this simplicity stems from the fact that the aforementioned shifting property need only be applied to a single vector, comprised of the input signal, $\mathbf{X}'_{\mathbf{M}}(n) = [x(n), x(n-1), \ldots, x(n-M+1)]$. As a result, in these more traditional proofs, the augmented correlation matrix of Equation 4.38 is symmetric. By using an identity for the inverse of a symmetric matrix which is itself composed of square matrices on its diagonal, the derivation of the inverse of the augmented (auto)correlation matrix becomes straightforward. In fact, most proofs first derive order update recursions for the needed quantities and then fix the length of the filter to get the desired fast Kalman algorithm. However, since the current fast Kalman derivation is based on the decorrelation criterion, the necessary symmetry of the augmented correlation matrix is *not* present in Equation 4.38. Because the proof given in this section cannot rely on this symmetry, it is necessary to derive the fast

Kalman in a manner which avoids it, hence, the need for an inductive proof of all relevant quantities. This lack of symmetry will also play a major role in development of lattice-based structures for the decorrelation algorithm. The use of lattice structures with the decorrelation criterion will be covered in detail in Chapter 5.

4.2.3 Updating the Kalman Gain Vector

In view of the expressions given in the last section, a vector $\bar{\mathbf{K}}(n) = \mathbf{K}_{\mathrm{M+1}}(n)$ with properties

$$\bar{\mathbf{R}}(n)\bar{\mathbf{K}}(n) = \bar{\mathbf{Y}}(n) \tag{4.49}$$

will be calculated as an intermediate step before determining $\mathbf{K}_{\mathrm{M}}(n)$. From Equations 4.46 and 4.47, it can be seen that it is only necessary to modify

$$\begin{bmatrix} 0 \\ \mathbf{K}_{\mathrm{M}}(n-1) \end{bmatrix}$$

so that

$$\begin{bmatrix} \rho(n) - y(n-1) \\ \mathbf{0}_{\mathrm{M}} \end{bmatrix}$$

on the right-hand side of Equation 4.46 is eliminated. To accomplish this, it will be assumed that an $M \times 1$ vector $\mathbf{F}_{\mathrm{M}}(n)$ is known, such that

$$\bar{\mathbf{R}}(n) \begin{bmatrix} 1 \\ \mathbf{F}_{\mathrm{M}}(n) \end{bmatrix} = \begin{bmatrix} \mathcal{F}_M(n) \\ \mathbf{0}_{\mathrm{M}} \end{bmatrix} \tag{4.50}$$

where, it should be noted, $\mathcal{F}_M(n)$ is a scalar. Consequently, using the above expression, it is shown in Appendix B that the following equality holds:

$$\bar{\mathbf{R}}(n) \begin{bmatrix} 1 \\ \mathbf{F}_{\mathrm{M}}(n) \end{bmatrix} \mathcal{F}_M^{-1}(n)\left[\rho(n) - y(n-1)\right] = \begin{bmatrix} \rho(n) - y(n-1) \\ \mathbf{0}_{\mathrm{M}} \end{bmatrix}. \tag{4.51}$$

Therefore, using the following definition

$$\bar{\mathbf{K}}(n) \triangleq \begin{bmatrix} -\mathcal{F}_M^{-1}(n)\left[\rho(n) - y(n-1)\right] \\ \mathbf{K}_{\mathrm{M}}(n-1) - \mathbf{F}_{\mathrm{M}}(n)\mathcal{F}_M^{-1}(n)\left[\rho(n) - y(n-1)\right] \end{bmatrix} \tag{4.52}$$

it is proved in Appendix C that it follows from Equations 4.46 and 4.51 that

$$\bar{\mathbf{R}}(n)\bar{\mathbf{K}}(n) = \bar{\mathbf{Y}}(n) \tag{4.53}$$

Now, partition $\bar{\mathbf{K}}(n)$ so that

$$\bar{\mathbf{K}}(n) = \begin{bmatrix} \mathbf{C}_{\mathrm{M}}(n) \\ c_M(n) \end{bmatrix} \tag{4.54}$$

where, it should be noted, $c_M(n)$ is a scalar. To go from Equation 4.49 to 4.47, it will be necessary to eliminate $c_M(n)$ in Equation 4.54 without affecting the upper part of $\bar{\mathbf{Y}}(n)$. In order to do this, a vector $\bar{\mathbf{D}}(n) = \mathbf{D}_{\mathrm{M+1}}(n)$ is required with the following properties:

$$\bar{\mathbf{R}}(n)\bar{\mathbf{D}}(n) = \begin{bmatrix} 0_{\mathrm{M}} \\ 1 \end{bmatrix} \tag{4.55}$$

where

$$\bar{\mathbf{D}}(n) \triangleq \begin{bmatrix} \mathbf{D}_{\mathrm{M}}(n) \\ \mathcal{B}_M^{-1}(n) \end{bmatrix}$$

and where $\mathcal{B}_M^{-1}(n)$ is a scalar. Then, subtracting $\bar{\mathbf{D}}(n)\mathcal{B}_M(n)c_M(n)$ from $\bar{\mathbf{K}}(n)$ in Equation 4.54 results in

$$\begin{aligned} \left[\bar{\mathbf{K}}(n) - \bar{\mathbf{D}}(n)\mathcal{B}_M(n)c_M(n)\right] &= \left[\begin{bmatrix} \mathbf{C}_{\mathrm{M}}(n) \\ c_M(n) \end{bmatrix} - \begin{bmatrix} \mathbf{D}_{\mathrm{M}}(n)\mathcal{B}_M(n)c_M(n) \\ \mathcal{B}_M^{-1}(n)\mathcal{B}_M(n)c_M(n) \end{bmatrix}\right] \\ &= \begin{bmatrix} \mathbf{C}_{\mathrm{M}}(n) - \mathbf{D}_{\mathrm{M}}(n)\mathcal{B}_M(n)c_M(n) \\ 0 \end{bmatrix}. \end{aligned} \tag{4.56}$$

Postmultiplying $\bar{\mathbf{R}}(n)$ by $\bar{\mathbf{K}}(n)-\bar{\mathbf{D}}(n)\mathcal{B}_M(n)c_M(n)$ and using the relation of Equation 4.55

$$\bar{\mathbf{R}}(n)\left[\bar{\mathbf{K}}(n) - \bar{\mathbf{D}}(n)\mathcal{B}_M(n)c_M(n)\right] = \bar{\mathbf{Y}}(n) - \begin{bmatrix} 0_{\mathrm{M}} \\ \mathcal{B}_M(n)c_M(n) \end{bmatrix} \tag{4.57}$$

It follows, then, from the second expression for $\bar{\mathbf{R}}(n)$ in Equation 4.38, together with the definition in Equation 4.35, that

$$\begin{aligned}
&\bar{\mathbf{R}}(n)\left[\bar{\mathbf{K}}(n) - \bar{\mathbf{D}}(n)\mathcal{B}_M(n)c_M(n)\right] = \\
&= \begin{bmatrix} \mathbf{R}_{\mathbf{M},\mathbf{M}}(n) & \check{\mathbf{Q}}_{\mathbf{M}}(n) \\ \check{\mathbf{S}}'_{\mathbf{M}}(n) & \check{\pi}(n) \end{bmatrix} \begin{bmatrix} \mathbf{C}_{\mathbf{M}}(n) - \mathbf{D}_{\mathbf{M}}(n)\mathcal{B}_M(n)c_M(n) \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{R}_{\mathbf{M},\mathbf{M}}(n)\left[\mathbf{C}_{\mathbf{M}}(n) - \mathbf{D}_{\mathbf{M}}(n)\mathcal{B}_M(n)c_M(n)\right] \\ \check{\mathbf{S}}'_{\mathbf{M}}(n)\left[\mathbf{C}_{\mathbf{M}}(n) - \mathbf{D}_{\mathbf{M}}(n)\mathcal{B}_M(n)c_M(n)\right] \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{Y}_{\mathbf{M}}(n) \\ y(n-M-1) \end{bmatrix} = \bar{\mathbf{Y}}_{\mathbf{M}}(n).
\end{aligned}$$

Inspection of the first row of the vector of $\bar{\mathbf{Y}}_{\mathbf{M}}(n)$ in the last line of the above equation reveals that

$$\mathbf{R}_{\mathbf{M},\mathbf{M}}(n)\left[\mathbf{C}_{\mathbf{M}}(n) - \mathbf{D}_{\mathbf{M}}(n)\mathcal{B}_M(n)c_M(n)\right] = \mathbf{Y}_{\mathbf{M}}(n). \tag{4.58}$$

Furthermore, comparing Equations 4.58 and 4.34, it can be seen that

$$\mathbf{K}_{\mathbf{M}}(n) = \mathbf{C}_{\mathbf{M}}(n) - \mathbf{D}_{\mathbf{M}}(n)\mathcal{B}_M(n)c_M(n) \tag{4.59}$$

which completes the inductive step from $n-1$ to n.

4.2.4 Updating the Auxiliary Variables

The matrices $\mathbf{F}_{\mathbf{M}}(n)$ and $\bar{\mathbf{D}}(n)$ with the properties of Equations 4.50 and 4.55, respectively, will now be determined. This will be done by means of induction. Therefore, assume that the vectors $\mathbf{F}_{\mathbf{M}}(n-1)$ and $\bar{\mathbf{D}}(n-1)$ are given, such that

$$\bar{\mathbf{R}}(n)\begin{bmatrix} 1 \\ \mathbf{F}_{\mathbf{M}}(n-1) \end{bmatrix} = \begin{bmatrix} \mathcal{F}_M(n-1) \\ \mathbf{0}_{\mathbf{M}} \end{bmatrix} \tag{4.60}$$

and

$$\bar{\mathbf{R}}(n-1)\bar{\mathbf{D}}(n-1) = \begin{bmatrix} \mathbf{0}_{\mathbf{M}} \\ 1 \end{bmatrix}. \tag{4.61}$$

With the method used to derive Equation 4.8, $\bar{\mathbf{R}}(n)$, as defined in Equation 4.37, can be written using a recursive expression:

$$\bar{\mathbf{R}}(n) = \lambda\bar{\mathbf{R}}(n-1) + \bar{\mathbf{Y}}(n)\bar{\bar{\mathbf{Y}}}'(n). \tag{4.62}$$

Therefore, together with Equations 4.35 and 4.36

$$\bar{\mathbf{R}}(n)\begin{bmatrix} 1 \\ \mathbf{F_M}(n-1) \end{bmatrix} = \left[\lambda\bar{\mathbf{R}}(n-1) + \bar{\mathbf{Y}}(n)\bar{\hat{\mathbf{Y}}}'(n)\right]\begin{bmatrix} 1 \\ \mathbf{F_M}(n-1) \end{bmatrix}$$

$$\begin{aligned} &= \lambda\bar{\mathbf{R}}(n-1)\begin{bmatrix} 1 \\ \mathbf{F_M}(n-1) \end{bmatrix} \\ &\quad + \bar{\mathbf{Y}}(n)\left[\hat{y}(n)\hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\right]\begin{bmatrix} 1 \\ \mathbf{F_M}(n-1) \end{bmatrix} \\ &= \lambda\begin{bmatrix} \mathcal{F}_M(n-1) \\ \mathbf{0_M} \end{bmatrix} \\ &\quad + \begin{bmatrix} y(n-1) \\ \mathbf{Y_M}(n-1) \end{bmatrix}\left(\hat{y}(n-1) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\mathbf{F_M}(n-1)\right) . \end{aligned} \tag{4.63}$$

Postmultiplying both sides of Equation 4.46 by $\left(\hat{y}(n-1) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\mathbf{F_M}(n-1)\right)$ yields

$$\bar{\mathbf{R}}(n)\begin{bmatrix} 0 \\ \mathbf{K_M}(n-1) \end{bmatrix}\left(\hat{y}(n-1) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\mathbf{F_M}(n-1)\right)$$

$$= \begin{bmatrix} \rho(n) \\ \mathbf{Y}(n) \end{bmatrix}\left(\hat{y}(n-1) + \hat{\mathbf{Y}}'(n)\mathbf{F}(n-1)\right) . \tag{4.64}$$

Consequently, subtracting Equation 4.64 from both sides of Equation 4.63:

$$\begin{aligned} &\bar{\mathbf{R}}(n)\begin{bmatrix} 1 \\ \mathbf{F_M}(n-1) - \mathbf{K_M}(n-1)\left(\hat{y}(n-1) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\mathbf{F_M}(n-1)\right) \end{bmatrix} \\ &= \bar{\mathbf{R}}(n)\left[\begin{bmatrix} 1 \\ \mathbf{F_M}(n-1) \end{bmatrix} - \begin{bmatrix} 0 \\ \mathbf{K_M}(n-1) \end{bmatrix}\left(\hat{y}(n-1) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\mathbf{F_M}(n-1)\right)\right] \\ &= \lambda\begin{bmatrix} \mathcal{F}_M(n-1) \\ \mathbf{0_M} \end{bmatrix} + \begin{bmatrix} y(n-1) \\ \mathbf{Y_M}(n-1) \end{bmatrix}\left(\hat{y}(n-1) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\mathbf{F_M}(n-1)\right) \\ &\quad - \begin{bmatrix} \rho(n) \\ \mathbf{Y_M}(n-1) \end{bmatrix}\left(\hat{y}(n-1) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\mathbf{F_M}(n-1)\right) \\ &= \begin{bmatrix} \lambda\mathcal{F}_M(n-1) + (y(n-1) - \rho(n))\left(\hat{y}(n-1) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\mathbf{F_M}(n-1)\right) \\ \mathbf{0_M} \end{bmatrix} . \end{aligned} \tag{4.65}$$

Therefore, if Equation 4.50 is satisfied for $n-1$, then it will be satisfied for n provided

$$\mathbf{F_M}(n) = \mathbf{F_M}(n-1) - \mathbf{K_M}(n-1)\left(\hat{y}(n-1) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\mathbf{F_M}(n-1)\right) . \tag{4.66}$$

Furthermore, it will also be satisfied for n provided

$$\mathcal{F}_M(n) = \lambda \mathcal{F}_M(n-1) + (y(n-1) - \rho(n)) \left(\hat{y}(n-1) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\mathbf{F}_{\mathbf{M}}(n-1)\right) \tag{4.67}$$

For simplicity, define the following scalar

$$f_M(n) = \hat{y}(n-1) + \mathbf{F}'_{\mathbf{M}}(n-1)\hat{\mathbf{Y}}_{\mathbf{M}}(n-1) \tag{4.68}$$

which will be used in Equations 4.66 and 4.67. Equation 4.68 will be referred to as the *a priori* forward prediction error for decorrelation, since it has a form similar to that expressed in [18]. Furthermore, Equation 4.67 has a form similar to the sum of weighted forward *a posteriori* prediction-error squares, as seen in [18]. Equations 4.66 and 4.67 define the updating process for the matrix $\mathbf{F}_{\mathbf{M}}(\mathbf{n})$ and the scalar $\mathcal{F}_M(n)$.

Now postmultiply Equation 4.62 by $\bar{\mathbf{D}}(n-1)$:

$$\begin{aligned} \bar{\mathbf{R}}(n)\bar{\mathbf{D}}(n-1) &= \lambda\bar{\mathbf{R}}(n-1)\bar{\mathbf{D}}(n-1) + \bar{\mathbf{Y}}(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{D}}(n-1) \\ &= \lambda \begin{bmatrix} 0_{\mathrm{M}} \\ 1 \end{bmatrix} + \bar{\mathbf{Y}}(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{D}}(n-1) \ . \end{aligned} \tag{4.69}$$

To obtain Equation 4.55, it will be necessary to eliminate the last vector on the right-hand side of Equation 4.69 that is proportional to $\bar{\mathbf{Y}}(\mathbf{n})$. Therefore, subtracting $\bar{\mathbf{R}}(n)\bar{\mathbf{K}}(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{D}}(n-1)$ from both sides of Equation 4.69:

$$\begin{aligned} &\bar{\mathbf{R}}(n)\left[\bar{\mathbf{D}}(n-1) - \bar{\mathbf{K}}(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{D}}(n-1)\right] \\ &= \lambda \begin{bmatrix} 0_{\mathrm{M}} \\ 1 \end{bmatrix} + \bar{\mathbf{Y}}(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{D}}(n-1) - \bar{\mathbf{Y}}(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{D}}(n-1) \\ &= \lambda \begin{bmatrix} 0_{\mathrm{M}} \\ 1 \end{bmatrix} . \end{aligned} \tag{4.70}$$

Therefore, if Equation 4.55 is satisfied for $n-1$ then it will be satisfied for n provided

$$\bar{\mathbf{D}}(n) = \frac{1}{\lambda}\left(\bar{\mathbf{D}}(n-1) - \bar{\mathbf{K}}(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{D}}(n-1)\right) . \tag{4.71}$$

Equation 4.71 represents the updating algorithm for $\bar{\mathbf{D}}(n)$.

The elements of $\bar{\mathbf{D}}(n)$ are used only in the combination $\mathbf{B}_{\mathbf{M}}(n) \triangleq \mathbf{D}_{\mathbf{M}}(n)\mathcal{B}_M(n)$. Therefore, it is more convenient to rewrite Equations 4.55, 4.71, and 4.59 in terms of

$$\bar{\mathbf{B}}(n) \triangleq \bar{\mathbf{D}}(n)\mathcal{B}_M(n) \triangleq \begin{bmatrix} \mathbf{B}_{\mathbf{M}}(n) \\ 1 \end{bmatrix} \tag{4.72}$$

which obeys

$$\bar{\mathbf{R}}(n)\bar{\mathbf{B}}(n) = \begin{bmatrix} 0_{\mathbf{M}} \\ \mathcal{B}_M(n) \end{bmatrix} . \tag{4.73}$$

Then, as in Equation 4.70

$$\begin{aligned} \bar{\mathbf{R}}(n)\left[\bar{\mathbf{B}}(n-1) - \bar{\mathbf{K}}(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{B}}(n-1)\right] &= \\ = \bar{\mathbf{R}}(n)&\begin{bmatrix} \mathbf{B}_{\mathbf{M}}(n-1) - \mathbf{C}_{\mathbf{M}}(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{B}}(n-1) \\ 1 - c_M(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{B}}(n-1) \end{bmatrix} \end{aligned} \tag{4.74}$$

where Equations 4.54 and 4.72 were used, along with the relation between $\bar{\mathbf{D}}(n)$ and $\bar{\mathbf{B}}(n)$. Consequently,

$$\bar{\mathbf{R}}(n)\left[\bar{\mathbf{B}}(n-1) - \bar{\mathbf{K}}(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{B}}(n-1)\right] = \lambda \begin{bmatrix} 0_{\mathbf{M}} \\ \mathcal{B}_M(n-1) \end{bmatrix} . \tag{4.75}$$

Therefore, postmultiplying Equation 4.75 with $\left[1 - c_M(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{B}}(n-1)\right]^{-1}$,

$$\begin{aligned} \bar{\mathbf{R}}(n)&\begin{bmatrix} \left[\frac{\mathbf{B}_{\mathbf{M}}(n-1) - \mathbf{C}_{\mathbf{M}}(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{B}}(n-1)}{1 - c_M(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{B}}(n-1)}\right] \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 0_{\mathbf{M}} \\ \lambda\mathcal{B}(n-1)\left[1 - c_M(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{B}}(n-1)\right]^{-1} \end{bmatrix} . \end{aligned} \tag{4.76}$$

Therefore, if Equation 4.72 is satisfied for $n-1$ then it is satisfied for n provided

$$\mathbf{B}_{\mathbf{M}}(n) = \frac{\mathbf{B}_{\mathbf{M}}(n-1) - \mathbf{C}_{\mathbf{M}}(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{B}}(n-1)}{1 - c_M(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{B}}(n-1)} \tag{4.77}$$

and

$$\mathcal{B}_M(n) = \frac{\lambda\mathcal{B}_M(n-1)}{1 - c_M(n)\bar{\hat{\mathbf{Y}}}'(n)\bar{\mathbf{B}}(n-1)} . \tag{4.78}$$

Using the definitions of Equations 4.36 and 4.72, Equation 4.77 can be written as

$$\mathbf{B_M}(n) = \frac{\mathbf{B_M}(n-1) - \mathbf{C_M}(n)\left(\hat{y}(n-M) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{B_M}(n-1)\right)}{1 - c_M(n)\left(\hat{y}(n-M) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{B_M}(n-1)\right)} \tag{4.79}$$

Define the scalar

$$b_M(n) = \hat{y}(n-M) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{B_M}(n-1) \ . \tag{4.80}$$

Equation 4.80 will be referred to as the *a priori* backward prediction error for decorrelation, since it has a form similar to that expressed in [18]. Therefore, Equation 4.77 can be written as

$$\mathbf{B_M}(n) = \frac{\mathbf{B_M}(n-1) - \mathbf{C_M}(n)b_M(n)}{1 - c_M(n)b_M(n)} \tag{4.81}$$

and, following a similar methodology with the expression for $\mathcal{B}_M(n)$, Equation 4.78 can be written as

$$\mathcal{B}_M(n) = \frac{\lambda\mathcal{B}_M(n-1)}{1 - c_M(n)b_M(n)} \ . \tag{4.82}$$

The algorithm is now complete. Given $\mathbf{F_M}(n-1)$, $\mathbf{B_M}(n-1)$, $\mathcal{F}_M(n-1)$, and $\mathbf{K_M}(n-1)$, a computation is performed in the order of Equations 4.66, 4.67, 4.52, 4.77, 4.54, and 4.59 in order to generate $\mathbf{F_M}(n)$, $\mathbf{B_M}(n)$, $\mathcal{F}_M(n)$, and $\mathbf{K_M}(n)$.

As a final note on the derivation of the fast Kalman algorithm for decorrelation, it should be mentioned that the variables $y(n-1)$, $\hat{y}(n-1)$, $\mathbf{Y_M}(n-1)$, $\hat{\mathbf{Y}}_{\mathbf{M}}(n)$ and so forth appearing in these equations are known. However, because of the asymmetry present in the augmented correlation matrix $\bar{\mathbf{R}}(n)$ of Equation 4.38, it will be necessary to store $\mathbf{S}(n)$ and update it at each iteration. With the method used to derive the recursive expression for Equation 4.5, a similar technique yields the following recursive expression for $\mathbf{S_M}(n)$:

$$\mathbf{S_M}(n) = \lambda\mathbf{S_M}(n-1) + \hat{\mathbf{Y}}_{\mathbf{M}}(n-1)y(n-1) \ . \tag{4.83}$$

$\mathbf{S_M}(n)$ will then be used to update $\rho(n)$ in Equation 4.67.

4.2.5 The Algorithm

Let

$$\mathbf{Y_M}(n) = \begin{bmatrix} y(n-1) \\ \vdots \\ y(n-M-1) \end{bmatrix} \text{ and } \hat{\mathbf{Y}}_\mathbf{M}(n) = \begin{bmatrix} \hat{y}(n-1) \\ \vdots \\ \hat{y}(n-M-1) \end{bmatrix}$$

Then the quantity

$$\mathbf{K_M}(n) = \left[\sum_{k=0}^{n} \lambda^{n-k} \mathbf{Y_M}(k) \hat{\mathbf{Y}}'_\mathbf{M}(k)\right]^{-1} \mathbf{Y_M}(n)$$

can be determined recursively as follows:

$$f_M(n) = \hat{y}(n-1) + \mathbf{F}'_\mathbf{M}(n-1)\hat{\mathbf{Y}}_\mathbf{M}(n-1) \tag{4.84}$$

$$\mathbf{F_M}(n) = \mathbf{F_M}(n-1) - \mathbf{K_M}(n-1) f_M(n) \tag{4.85}$$

$$\mathbf{S_M}(n) = \lambda \mathbf{S_M}(n-1) + \hat{\mathbf{Y}}_\mathbf{M}(n-1) y(n-1) \tag{4.86}$$

$$g_M(n) = y(n-1) - \mathbf{S}'_\mathbf{M}(n)\mathbf{K_M}(n-1) \tag{4.87}$$

$$\mathcal{F}_M(n) = \lambda \mathcal{F}_M(n-1) + g_M(n) f_M(n) \tag{4.88}$$

$$\mathbf{K_{M+1}}(n) = \begin{bmatrix} \mathcal{F}_M^{-1}(n) g_M(n) \\ \mathbf{K_M}(n-1) + \mathbf{F_M}(n)\mathcal{F}_M^{-1}(n) g_M(n) \end{bmatrix} . \tag{4.89}$$

Partition $\mathbf{K_{M+1}}(n)$ as

$$\mathbf{K_{M+1}}(n) \triangleq \begin{bmatrix} \mathbf{C_M}(n) \\ c_M(n) \end{bmatrix} . \tag{4.90}$$

Let

$$b_M(n) = \hat{y}(n-M-1) + \mathbf{B}'_\mathbf{M}(n-1)\hat{\mathbf{Y}}_\mathbf{M}(n) \tag{4.91}$$

$$\mathbf{B_M}(n) = \frac{\mathbf{B_M}(n-1) - \mathbf{C_M}(n) b_M(n)}{1 - c_M(n) b_M(n)} \tag{4.92}$$

$$\mathbf{K_M}(n) = \mathbf{C_M}(n) - \mathbf{B_M}(n) c_M(n) \ . \tag{4.93}$$

The tap weight vector, $\mathbf{W_M}(n)$, used in the RLC algorithm can then be updated using the fast Kalman gain vector of Equation 4.93 according to

$$\eta_M(n) = x(n) - \hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{W}_{\mathbf{M}}(n-1)$$

$$\mathbf{W}_{\mathbf{M}}(n) = \mathbf{W}_{\mathbf{M}}(n-1) + \eta_M(n)\mathbf{K}_{\mathbf{M}}(n)$$

The initial conditions can be taken as $\mathbf{K}_{\mathbf{M}}(0)=\mathbf{0}_{\mathbf{M}}$, $\mathbf{F}_{\mathbf{M}}(0)=\mathbf{0}_{\mathbf{M}}$, $\mathbf{S}_{\mathbf{M}}(0)=\mathbf{0}_{\mathbf{M}}$, $\mathbf{B}_{\mathbf{M}}(0)=\mathbf{0}_{\mathbf{M}}$, $\mathbf{W}_{\mathbf{M}}(0)=\mathbf{0}_{\mathbf{M}}$, and $\mathcal{F}_M(0)=\delta > 0$.

4.2.6 Proof

The algorithm of Equations 4.84 through 4.93 follows the derivations given in this section. Given $\mathbf{F}_{\mathbf{M}}(n-1)$, $\mathbf{B}_{\mathbf{M}}(n-1)$, $\mathcal{F}_M(n-1)$, and $\mathbf{K}_{\mathbf{M}}(n-1)$, perform Equation 4.66 = 4.84 + 4.85 to determine $\mathbf{F}_{\mathbf{M}}(n)$. Then Equation 4.67 restores $\mathcal{F}_M(n)$, which, by the use of Equations 4.86, 4.87, and 4.84 can be written as Equation 4.88. With $\mathbf{F}_{\mathbf{M}}(n)$ and $\mathcal{F}_M(n)$, $\mathbf{K}_{\mathbf{M+1}}(n)$ can be determined from Equation 4.52 = 4.91 + 4.92. Finally, updating $\mathbf{K}_{\mathbf{M}}(n)$ by Equation 4.59 is carried out in Equation 4.93, where it is again noted that $\mathbf{B}_{\mathbf{M}}(n) \triangleq \mathbf{D}_{\mathbf{M}}(n)\mathcal{B}_M(n)$.

4.2.7 Alternative Declaration of the Algorithm

Given the algorithm of the previous section, it is possible to rewrite the algorithm into a form which can be condsidered as a type of Levinson-Durbin recursion. In fact, Equation 4.85 can be considered as the declaration of the forward prediction coefficients of a Levinson-Durbin recursion. Consequently, Equation 4.84 can be considered as the forward prediction error and Equation 4.91 as the backward prediction error of the same Levinson-Durbin-type recursion. All that is necessary is to generate an equation, of the same form as Equation 4.85, that will describe the backward prediction coefficients.

To accomplish this, recall from Equation 4.50 that the forward prediction coefficients can be written in matrix form as

$$\begin{bmatrix} \pi(n) & \mathbf{P}'_{\mathrm{M}}(n) \\ \mathbf{Q}_{\mathrm{M}}(n) & \mathbf{R}_{\mathrm{M,M}}(n-1) \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{F}_{\mathrm{M}}(n) \end{bmatrix} = \begin{bmatrix} \mathcal{F}_M(n) \\ 0_{\mathrm{M}} \end{bmatrix} \tag{4.94}$$

In a completely analogous development (as will be shown in subsequent sections), the backward prediction coefficients can be written in a similar form to that seen in Equation 4.94. Using the second equality for the augmented autocorrelation matrix in Equation 4.38, the backward prediction coefficients can be written in matrix form as follows:

$$\begin{bmatrix} \mathbf{R}_{\mathrm{M,M}}(n) & \check{\mathbf{Q}}_{\mathrm{M}}(n) \\ \check{\mathbf{S}}'_{\mathrm{M}}(n) & \check{\pi}(n) \end{bmatrix} \begin{bmatrix} \mathbf{B}_{\mathrm{M}}(n) \\ 1 \end{bmatrix} = \begin{bmatrix} 0_{\mathrm{M}} \\ \mathcal{B}_M(n) \end{bmatrix} \tag{4.95}$$

From Equation 4.95, the backward prediction coefficients can be defined as

$$\mathbf{B}_{\mathrm{M}}(n) = -\mathbf{R}^{-1}_{\mathrm{M,M}}(n)\check{\mathbf{Q}}_{\mathrm{M}}(n). \tag{4.96}$$

A recursive expression for the $Mx1$-dimensional vector $\check{\mathbf{Q}}_{\mathrm{M}}(n)$ of Equation 4.45 can be derived in a manner similar to that employed in the derivation of Equation 4.8. Therefore,

$$\check{\mathbf{Q}}_{\mathrm{M}}(n) = \lambda\check{\mathbf{Q}}_{\mathrm{M}}(n-1) + \hat{y}(n-M-1)\mathbf{Y}_{\mathrm{M}}(n) \ . \tag{4.97}$$

Substituting Equations 4.28 and 4.97 into Equation 4.96,

$$\begin{aligned} \mathbf{B}_{\mathrm{M}}(n) &= -\frac{1}{\lambda}\left(\mathbf{R}^{-1}_{\mathrm{M,M}}(n-1) - \mathbf{K}_{\mathrm{M}}(n)\hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{R}^{-1}_{\mathrm{M,M}}(n-1)\right)\check{\mathbf{Q}}_{\mathrm{M}}(n) \\ &= \mathbf{B}_{\mathrm{M}}(n-1) - \frac{1}{\lambda}(\lambda + \mu_M(n))\,\hat{y}(n-M-1)\mathbf{K}_{\mathrm{M}}(n) \\ &\quad - \mathbf{K}_{\mathrm{M}}(n)\hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{B}_{\mathrm{M}}(n-1) + \frac{1}{\lambda}\mu_M(n)\hat{y}(n-M-1)\mathbf{K}_{\mathrm{M}}(n) \\ &= \mathbf{B}_{\mathrm{M}}(n-1) - \mathbf{K}_{\mathrm{M}}(n)\left[\hat{y}(n-M-1) + \hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{B}_{\mathrm{M}}(n-1)\right] \\ &= \mathbf{B}_{\mathrm{M}}(n-1) - \mathbf{K}_{\mathrm{M}}(n)\left[\hat{y}(n-M-1) + \mathbf{B}'_{\mathrm{M}}(n-1)\hat{\mathbf{Y}}_{\mathrm{M}}(n)\right] \end{aligned}$$

where Equations 4.14 and 4.30 were used in the second equality. Using Equation 4.91, $\mathbf{B}(n)$ can be written as

$$\mathbf{B}_{\mathrm{M}}(n) = \mathbf{B}_{\mathrm{M}}(n-1) - \mathbf{K}_{\mathrm{M}}(n)b_M(n) \ . \tag{4.98}$$

This is the desired form for the equation for the backward prediction coefficients. With Equation 4.98 now established, it is possible to restate the algorithm given in the previous section. The revised algorithm is summarized below:

$$f_M(n) = \hat{y}(n-1) + \mathbf{F}'_\mathbf{M}(n-1)\hat{\mathbf{Y}}_\mathbf{M}(n-1) \tag{4.99}$$

$$b_M(n) = \hat{y}(n-M-1) + \mathbf{B}'_\mathbf{M}(n-1)\hat{\mathbf{Y}}_\mathbf{M}(n) \tag{4.100}$$

$$\mathbf{F}_\mathbf{M}(n) = \mathbf{F}_\mathbf{M}(n-1) - \mathbf{K}_\mathbf{M}(n-1)f_M(n) \tag{4.101}$$

$$\mathbf{S}_\mathbf{M}(n) = \lambda\mathbf{S}_\mathbf{M}(n-1) + \hat{\mathbf{Y}}_\mathbf{M}(n-1)y(n-1) \tag{4.102}$$

$$g_M(n) = y(n-1) - \mathbf{S}'_\mathbf{M}(n)\mathbf{K}_\mathbf{M}(n-1) \tag{4.103}$$

$$\mathcal{F}_M(n) = \lambda\mathcal{F}_M(n-1) + g_M(n)f_M(n) \tag{4.104}$$

$$\begin{bmatrix} \mathbf{C}_\mathbf{M}(n) \\ c_M(n) \end{bmatrix} \equiv \mathbf{K}_{\mathbf{M+1}}(n) = \begin{bmatrix} 0 \\ \mathbf{K}_\mathbf{M}(n-1) \end{bmatrix} + \frac{g_M(n)}{\mathcal{F}_M(n)} \begin{bmatrix} 1 \\ \mathbf{F}_\mathbf{M}(n) \end{bmatrix} \tag{4.105}$$

$$\mathbf{K}_\mathbf{M}(n) = \frac{\mathbf{C}_\mathbf{M}(n) - c_M(n)\mathbf{B}_\mathbf{M}(n-1)}{1 - c_M(n)b_M(n)} \tag{4.106}$$

$$\mathbf{B}_\mathbf{M}(n) = \mathbf{B}_\mathbf{M}(n-1) - \mathbf{K}_\mathbf{M}(n)b_M(n) \tag{4.107}$$

where, in the derivation of Equation 4.106, Equation 4.93 was used together with Equation 4.98. Thus, as can be seen by the equations of the previous two sections, the fast Kalman algorithm uses forward and backward predictors to update the Kalman gain vector as a new input decision $\hat{y}(n-1)$ enters the equalizer and the oldest decision $\hat{y}(n-M-1)$ is discarded. A count on the number of MADPI of the fast Kalman algorithm necessary to update the tap weight vector reveals that the fast Kalman for decorrelation requires approximately $10M + 6$ MADPI. This is a substantial savings over the $\mathcal{O}(M^2)$ complexity of the RLC algorithm. In the subsequent discussions, the fast Kalman algorithm for decorrelation will also be referred to as the FRLC (Fast Recursive Least Correlation) algorithm. The terms *fast Kalman* and *FRLC* will be used interchangeably to refer to the same algorithm.

4.3 The Fast Transversal Filter for Decorrelation

In the previous section, it was shown that it is possible to decrease the complexity of the RLC algorithm while still maintaining its high rate of convergence. A simple inspection of the algorithm definition for the fast Kalman shows that the algorithm is mainly based on the *a priori* error formulation. Carayannis, *et al.* [5] and Cioffi and Kailath [7] took the fast Kalman algorithm one step further and derived another member of this class of transversal filter-based fast algorithms, referred to as the fast transversal filter (FTF). This particular fast structure differs from the fast Kalman in that it is based, primarily, on an *a posteriori* error formulation. Furthermore, the FTF makes better use of the relationships between the *a priori* and *a posteriori* errors. It is through exploitation of these relationships that the FTF algorithm is able to further reduce the complexity of the fast Kalman algorithm. The inherent similarity between the fast Kalman and FTF algorithms will become apparent in the derivation that follows.

In this section, as a natural extension of the derivation of the fast Kalman algorithm with decorrelation, the decorrelation criterion will be applied to the fast transveral structure. It will be shown that the FTF for decorrelation offers a comparable reduction in complexity as compared with the FRLC. In one of the original derivations of the FTF algorithm, Cioffi and Kailath [7] used a geometrical approach to derive the FTF. However, for the following derivation of the FTF with decorrelation, an algebraic approach, similar to that presented in [5], will be used.

4.3.1 Derivation of the Algorithm

The key to the development of the FTF with decorrelation is the use of the so-called *alternative Kalman gain vector.* Multiplying the numerator and denominator of Equation 4.30 by $\frac{1}{\lambda}$, the Kalman gain vector defined in Equation 4.30 can be written as

$$\mathbf{K}_{\mathbf{M}}(n) = \frac{\frac{1}{\lambda}\mathbf{R}^{-1}_{\mathbf{M,M}}(n-1)\mathbf{Y}_{\mathbf{M}}(n)}{1 + \frac{1}{\lambda}\mu_M(n)} \tag{4.108}$$

Therefore, considering the above expression, let

$$\alpha_M(n) = 1 + \frac{1}{\lambda}\mu_M(n) = 1 + \frac{1}{\lambda}\hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{R}^{-1}_{\mathbf{M,M}}(n-1)\mathbf{Y}_{\mathbf{M}}(n) \ . \tag{4.109}$$

Given the above definition and the definition of the Kalman gain vector of Equation 4.108, the alternative Kalman gain vector for decorrelation, $\tilde{\mathbf{K}}_{\mathbf{M}}(n)$, is given as

$$\tilde{\mathbf{K}}_{\mathbf{M}}(n) = \alpha_M(n)\mathbf{K}_{\mathbf{M}}(n) = \frac{1}{\lambda}\mathbf{R}^{-1}_{\mathbf{M,M}}(n-1)\mathbf{Y}_{\mathbf{M}}(n) \ . \tag{4.110}$$

Equation 4.110 is similar in form to the alternative Kalman gain vector used by Proakis and Manolakis [37] and is similar to the form of an exponentially weighted version of the alternative Kalman gain vector used by Carayannis, *et al.* [5]. However, as should be noted, the difference in the forms lies in the fact that the alternative Kalman gain vector of Equation 4.110 is based on the decorrelation criterion, where that used in [37] and [5] is not.

It will be useful to define the following scalar quantity,

$$\gamma_M(n) = \frac{1}{\alpha_M(n)} = 1 - \hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{K}_{\mathbf{M}}(n) = \frac{\lambda}{\lambda + \mu_M(n)} \ . \tag{4.111}$$

It should be noted that if the derivation of the FTF for decorrelation followed the approach given by Haykin in [18], the alternative Kalman gain vector defined using $\gamma_M(n)$ in Equation 4.111 would be used instead of $\alpha_M(n)$ in Equation 4.109. Although this derivation of the FTF with decorrelation will follow the approach used in [5], to some extent, the need for the quantity $\gamma_M(n)$ will become apparent in the following proof.

The first step in the proof of the FTF will be to redefine the extended Kalman gain vector of Equation 4.105 in terms of the alternative Kalman gain vector and the *a priori* estimate of the forward prediction coefficients, $\mathbf{F}_{\mathbf{M}}(n)$. Therefore, let

$$\mathbf{a}_{\mathbf{M}}(n) = \begin{bmatrix} 1 \\ \mathbf{F}_{\mathbf{M}}(n) \end{bmatrix} \ . \tag{4.112}$$

Rewriting Equation 4.101 accordingly, gives

$$\mathbf{a}_{\mathrm{M}}(n) = \mathbf{a}_{\mathrm{M}}(n-1) - \begin{bmatrix} 0 \\ \mathbf{K}_{\mathrm{M}}(n-1) \end{bmatrix} f_M(n) \tag{4.113}$$

Substituting Equation 4.113 into Equation 4.105 and collecting terms:

$$\begin{aligned} \mathbf{K}_{\mathrm{M+1}}(n) &= \begin{bmatrix} 0 \\ \mathbf{K}_{\mathrm{M}}(n-1) \end{bmatrix} + \frac{g_M(n)}{\mathcal{F}_M(n)}\mathbf{a}_{\mathrm{M}}(n) \\ &= \begin{bmatrix} 0 \\ \mathbf{K}_{\mathrm{M}}(n-1) \end{bmatrix} + \frac{g_M(n)}{\mathcal{F}_M(n)}\mathbf{a}_{\mathrm{M}}(n-1) - \frac{g_M(n)f_M(n)}{\mathcal{F}_M(n)}\begin{bmatrix} 0 \\ \mathbf{K}_{\mathrm{M}}(n-1) \end{bmatrix} \\ &= \left[1 - \frac{g_M(n)f_M(n)}{\mathcal{F}_M(n)}\right]\begin{bmatrix} 0 \\ \mathbf{K}_{\mathrm{M}}(n-1) \end{bmatrix} + \frac{g_M(n)}{\mathcal{F}_M(n)}\mathbf{a}_{\mathrm{M}}(n-1) \ . \end{aligned} \tag{4.114}$$

Consequently, in order to write Equation 4.114 in terms of the alternative Kalman gain vector, $\tilde{\mathbf{K}}_{\mathrm{M}}(n)$, it will be necessary to show that

$$\left[1 - \frac{g_M(n)f_M(n)}{\mathcal{F}_M(n)}\right] \equiv \frac{\lambda\mathcal{F}_M(n-1)}{\mathcal{F}_M(n)} \equiv \frac{\alpha_M(n-1)}{\alpha_{M+1}(n)} \tag{4.115}$$

Thus, using Equation 4.105 and the first expression for $\hat{\mathbf{Y}}_{\mathrm{M+1}}(n)$ in Equation 4.36 as follows

$$\begin{aligned} \gamma_{M+1}(n) &= 1 - \hat{\mathbf{Y}}'_{\mathrm{M+1}}(n)\mathbf{K}_{\mathrm{M+1}}(n) \\ &= 1 - \hat{\mathbf{Y}}'_{\mathrm{M+1}}(n)\left\{\begin{bmatrix} 0 \\ \mathbf{K}_{\mathrm{M}}(n-1) \end{bmatrix} + \frac{g_M(n)}{\mathcal{F}_M(n)}\begin{bmatrix} 1 \\ \mathbf{F}_{\mathrm{M}}(n) \end{bmatrix}\right\} \\ &= 1 - \hat{\mathbf{Y}}'_{\mathrm{M}}(n-1)\mathbf{K}_{\mathrm{M}}(n-1) && (4.116) \\ &\quad - \frac{g_M(n)}{\mathcal{F}_M(n)}\left(\hat{y}(n-1) + \hat{\mathbf{Y}}'_{\mathrm{M}}(n-1)\mathbf{F}_{\mathrm{M}}(n)\right) && (4.117) \end{aligned}$$

Define the *a posteriori* forward prediction error for decorrelation as

$$f_m(n,n) = \hat{y}(n-1) + \hat{\mathbf{Y}}'_{\mathrm{M}}(n-1)\mathbf{F}_{\mathrm{M}}(n) \tag{4.118}$$

where, it should be noted, $f_m(n,n)$ is a scalar. Comparing Equation 4.118 with Equation 4.99, to determine the basic relationship between $f_m(n,n)$ and $f_M(n)$, substitute Equation 4.101 for $\mathbf{F_M}(n)$ in Equation 4.118:

$$\begin{aligned} f_m(n,n) &= \hat{y}(n-1) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\left[\mathbf{F_M}(n-1) - \mathbf{K_M}(n-1)f_M(n)\right] \\ &= \hat{y}(n-1) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\mathbf{F_M}(n-1) - f_M(n)\hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\mathbf{K_M}(n-1) \\ &= f_M(n)\left[1 - \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\mathbf{K_M}(n-1)\right] \\ &= \gamma_M(n-1)f_M(n) \;=\; \frac{f_M(n)}{\alpha_M(n-1)} \;. \end{aligned} \tag{4.119}$$

Using Equations 4.111, 4.118, and 4.119 in Equation 4.117 yields

$$\begin{aligned} \gamma_{M+1}(n) &= \gamma_M(n-1) - \frac{g_M(n)f_M(n,n)}{\mathcal{F}_M(n)} \\ &= \gamma_M(n-1)\left[1 - \frac{g_M(n)f_M(n)}{\mathcal{F}_M(n)}\right] \end{aligned} \tag{4.120}$$

Thus, according to Equations 4.111 and 4.120

$$1 - \frac{g_M(n)f_M(n)}{\mathcal{F}_M(n)} = \frac{\gamma_{M+1}(n)}{\gamma_M(n-1)} = \frac{\alpha_M(n-1)}{\alpha_{M+1}(n)} \tag{4.121}$$

Furthermore, substituting Equation 4.104 into the left-hand side (LHS) of Equation 4.121,

$$\begin{aligned} 1 - \frac{g_M(n)f_M(n)}{\mathcal{F}_M(n)} &= \frac{\mathcal{F}_M(n) - g_M(n)f_M(n)}{\mathcal{F}_M(n)} \\ &= \frac{\lambda\mathcal{F}_M(n-1) + g_M(n)f_M(n) - g_M(n)f_M(n)}{\mathcal{F}_M(n)} \\ &= \frac{\lambda\mathcal{F}_M(n-1)}{\mathcal{F}_M(n)} \;. \end{aligned} \tag{4.122}$$

Thus, Equation 4.115 is proved. Next, substitute the results of Equation 4.121 into Equation 4.114 and then premultiply both sides of Equation 4.114 by $\alpha_{M+1}(n)$ to get

$$\alpha_{M+1}(n)\mathbf{K_{M+1}}(n) = \begin{bmatrix} 0 \\ \alpha_M(n-1)\mathbf{K_M}(n-1) \end{bmatrix} + \frac{g_M(n)\alpha_{M+1}(n)}{\mathcal{F}_M(n)}\mathbf{a_M}(n-1) \;.$$

Noting from Equation 4.115 that

$$\frac{\alpha_{M+1}(n)}{\mathcal{F}_M(n)} = \frac{\alpha_M(n-1)}{\lambda\mathcal{F}_M(n-1)} \tag{4.123}$$

and using the definition of Equation 4.110, the extended fast Kalman gain vector of Equation 4.105 can be rewritten in terms of the FTF alternative Kalman gain vector as

$$\tilde{\mathbf{K}}_{\mathrm{M+1}}(n) = \begin{bmatrix} 0 \\ \tilde{\mathbf{K}}_{\mathrm{M}}(n-1) \end{bmatrix} + \frac{g_M(n)\alpha_M(n-1)}{\lambda \mathcal{F}_M(n-1)} \begin{bmatrix} 1 \\ \mathbf{F}_{\mathrm{M}}(n-1) \end{bmatrix} \tag{4.124}$$

where $\tilde{\mathbf{K}}_{\mathrm{M+1}}(n)$ can be partitioned as

$$\tilde{\mathbf{K}}_{\mathrm{M+1}}(n) = \begin{bmatrix} \tilde{\mathbf{C}}_{\mathrm{M}}(n) \\ \tilde{c}_M(n) \end{bmatrix} \tag{4.125}$$

At this point, to proceed with the proof it will be necessary to derive the extended Kalman gain vector of Equation 4.105 in terms of the backward prediction coefficients of Equation 4.107. To do this, a process similar to that followed for the derivation of the fast Kalman gain vector, $\mathbf{K}_{\mathrm{M}}(n)$, will be used, but now reformulated in terms of $\mathbf{B}_{\mathrm{M}}(n)$ and not $\mathbf{F}_{\mathrm{M}}(n)$. Therefore, in lieu of Equations 4.47, 4.48, and 4.49, it can be seen that it will be necessary to modify

$$\begin{bmatrix} \mathbf{K}_{\mathrm{M}}(n) \\ 0 \end{bmatrix}$$

so that

$$\begin{bmatrix} 0_{\mathrm{M}} \\ \beta(n) - y(n-M-1) \end{bmatrix}$$

on the RHS of Equation 4.47 is eliminated. To accomplish this, it will be assumed that an $M \times 1$ vector $\mathbf{B}_{\mathrm{M}}(n)$ is known, such that

$$\bar{\mathbf{R}}(n) \begin{bmatrix} \mathbf{B}_{\mathrm{M}}(n) \\ 1 \end{bmatrix} = \begin{bmatrix} 0_{\mathrm{M}} \\ \mathcal{B}_M(n) \end{bmatrix} \tag{4.126}$$

where, it should be noted, $\mathcal{B}_M(n)$ is a scalar. Consequently, using the above expression and a method similar to that shown in Appendix B,

$$\bar{\mathbf{R}}(n) \begin{bmatrix} \mathbf{B}_{\mathrm{M}}(n) \\ 1 \end{bmatrix} \mathcal{B}_M^{-1}(n)\left[\beta(n) - y(n-M-1)\right] = \begin{bmatrix} 0_{\mathrm{M}} \\ \beta(n) - y(n-M-1) \end{bmatrix} \tag{4.127}$$

Therefore, with a method similar to that expressed in Appendix C, using the following definition,

$$\bar{\mathbf{K}}(n) = \mathbf{K}_{\mathrm{M+1}}(n) \triangleq \begin{bmatrix} \mathbf{K}_{\mathrm{M}}(n) - \mathbf{B}_{\mathrm{M}}(n)\mathcal{B}_M^{-1}(n)\left[\beta(n) - y(n-M-1)\right] \\ -\mathcal{B}_M^{-1}(n)\left[\beta(n) - y(n-M-1)\right] \end{bmatrix} \quad (4.128)$$

with Equations 4.47 and 4.127 and the second expression for $\bar{\mathbf{R}}(n)$ in Equation 4.38, it can be shown that

$$\bar{\mathbf{R}}(n)\bar{\mathbf{K}}(n) = \bar{\mathbf{Y}}(n) \quad (4.129)$$

where $\bar{\mathbf{K}}(n)$ can be partitioned as in Equation 4.54.

The determination of the time update recursion for $\mathcal{B}_M(n)$ will be done by means of induction. The time-update equation for $\mathbf{B}_{\mathrm{M}}(n)$ was previously given in Equation 4.107. Therefore, assume that the vector $\mathbf{B}_{\mathrm{M}}(n-1)$ is given such that

$$\bar{\mathbf{R}}(n)\begin{bmatrix} \mathbf{B}_{\mathrm{M}}(n-1) \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{\mathrm{M}} \\ \mathcal{B}_M(n-1) \end{bmatrix} . \quad (4.130)$$

Therefore, together with Equations 4.35, 4.36, and 4.62,

$$\begin{aligned} \bar{\mathbf{R}}(n)\begin{bmatrix} \mathbf{B}_{\mathrm{M}}(n-1) \\ 1 \end{bmatrix} &= \left[\lambda\bar{\mathbf{R}}(n-1) + \bar{\mathbf{Y}}(n)\bar{\hat{\mathbf{Y}}}'(n)\right]\begin{bmatrix} \mathbf{B}_{\mathrm{M}}(n-1) \\ 1 \end{bmatrix} \\ &= \lambda\begin{bmatrix} \mathbf{0}_{\mathrm{M}} \\ \mathcal{B}_M(n-1) \end{bmatrix} \\ &\quad + \begin{bmatrix} \mathbf{Y}_{\mathrm{M}}(n) \\ y(n-M-1) \end{bmatrix}\left(\hat{y}(n-M-1) + \hat{\mathbf{Y}}_{\mathrm{M}}'(n)\mathbf{B}_{\mathrm{M}}(n-1)\right) \quad (4.131) \end{aligned}$$

where, from Equation 4.100, $b_M(n) = \hat{y}(n-M-1) + \mathbf{B}_{\mathrm{M}}'(n-1)\hat{\mathbf{Y}}_{\mathrm{M}}(n)$. Postmultiplying both sides of Equation 4.47 by $b_M(n)$ yields

$$\bar{\mathbf{R}}(n)\begin{bmatrix} \mathbf{K}_{\mathrm{M}}(n) \\ 0 \end{bmatrix} b_M(n) = \begin{bmatrix} \mathbf{Y}_{\mathrm{M}}(n) \\ \beta(n) \end{bmatrix} b_M(n) \quad (4.132)$$

Consequently, subtracting Equation 4.132 from both sides of Equation 4.131

$$\begin{aligned}
&\bar{\mathbf{R}}(n)\begin{bmatrix} \mathbf{B_M}(n-1)-\mathbf{K_M}(n)b_M(n) \\ 1 \end{bmatrix} \\
&= \bar{\mathbf{R}}(n)\left[\begin{bmatrix} \mathbf{B_M}(n-1) \\ 1 \end{bmatrix} - \begin{bmatrix} \mathbf{K_M}(n) \\ 0 \end{bmatrix} b_M(n)\right] \\
&= \lambda\begin{bmatrix} \mathbf{0_M} \\ \mathcal{B}_M(n-1) \end{bmatrix} + \begin{bmatrix} \mathbf{Y_M}(n) \\ y(n-M-1) \end{bmatrix} b_M(n) \\
&\quad - \begin{bmatrix} \mathbf{Y_M}(n) \\ \beta(n) \end{bmatrix} b_M(n) \\
&= \begin{bmatrix} \mathbf{0_M} \\ \lambda\mathcal{B}_M(n-1)+(y(n-M-1)-\beta(n))\,b_M(n) \end{bmatrix} .
\end{aligned} \tag{4.133}$$

Comparing Equation 4.133 with Equation 4.130, the time update expression for $\mathcal{B}_M(n)$ is

$$\mathcal{B}_M(n) = \lambda\mathcal{B}_M(n-1) + (y(n-M-1)-\beta(n))\,b_M(n) \tag{4.134}$$

where $\beta(n)$ is given by Equation 4.48. For simplicity, let

$$h_M(n) = y(n-M-1) - \beta(n) \ . \tag{4.135}$$

Therefore, with the previously defined quantities in mind, the extended Kalman gain vector defined by Equation 4.128 can be written as

$$\bar{\mathbf{K}}(n) = \mathbf{K_{M+1}}(n) \triangleq \begin{bmatrix} \mathbf{K_M}(n) \\ 0 \end{bmatrix} + \frac{h_M(n)}{\mathcal{B}_M(n)}\begin{bmatrix} \mathbf{B_M}(n) \\ 1 \end{bmatrix} . \tag{4.136}$$

Now that the extended Kalman gain vector has been defined in terms of the backward prediction coefficients, it will be necessary to formulate the extended alternative Kalman gain vector using Equation 4.136. To do this, let

$$\mathbf{b_M}(n) = \begin{bmatrix} \mathbf{B_M}(n) \\ 1 \end{bmatrix} . \tag{4.137}$$

Rewriting Equation 4.107 accordingly, gives

$$\mathbf{b_M}(n) = \mathbf{b_M}(n-1) - \begin{bmatrix} \mathbf{K_M}(n) \\ 0 \end{bmatrix} b_M(n) \ . \tag{4.138}$$

Substituting Equation 4.138 into Equation 4.136 and collecting terms:

$$\begin{aligned}\mathbf{K}_{\mathbf{M+1}}(n) &= \begin{bmatrix}\mathbf{K}_{\mathbf{M}}(n-1)\\0\end{bmatrix}+\frac{h_M(n)}{\mathcal{B}_M(n)}\mathbf{b}_{\mathbf{M}}(n-1)-\frac{h_M(n)b_M(n)}{\mathcal{B}_M(n)}\begin{bmatrix}\mathbf{K}_{\mathbf{M}}(n)\\0\end{bmatrix}\\ &= \left[1-\frac{h_M(n)b_M(n)}{\mathcal{B}_M(n)}\right]\begin{bmatrix}\mathbf{K}_{\mathbf{M}}(n)\\0\end{bmatrix}+\frac{h_M(n)}{\mathcal{B}_M(n)}\mathbf{b}_{\mathbf{M}}(n-1)\ . \qquad (4.139)\end{aligned}$$

Therefore, in order to write Equation 4.139 in terms of the alternative gain vector, $\tilde{\mathbf{K}}_{\mathbf{M}}(n)$, it will be necessary to show that

$$\left[1-\frac{h_M(n)b_M(n)}{\mathcal{B}_M(n)}\right]\equiv\frac{\lambda\mathcal{B}_M(n-1)}{\mathcal{B}_M(n)}\equiv\frac{\alpha_M(n)}{\alpha_{M+1}(n)}\ . \qquad (4.140)$$

Thus, using Equation 4.136 and the second expression for $\hat{\mathbf{Y}}_{\mathbf{M+1}}(n)$ in Equation 4.36 as follows

$$\begin{aligned}\gamma_{M+1}(n) &= 1-\hat{\mathbf{Y}}'_{\mathbf{M+1}}(n)\mathbf{K}_{\mathbf{M+1}}(n)\\ &= 1-\hat{\mathbf{Y}}'_{\mathbf{M+1}}(n)\left\{\begin{bmatrix}\mathbf{K}_{\mathbf{M}}(n)\\0\end{bmatrix}+\frac{h_M(n)}{\mathcal{B}_M(n)}\begin{bmatrix}\mathbf{B}_{\mathbf{M}}(n)\\1\end{bmatrix}\right\}\\ &= 1-\hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{K}_{\mathbf{M}}(n) \qquad (4.141)\\ &\quad -\frac{h_M(n)}{\mathcal{B}_M(n)}\left(\hat{y}(n-M-1)+\hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{B}_{\mathbf{M}}(n)\right)\ . \qquad (4.142)\end{aligned}$$

Define the *a posteriori* backward prediction error for decorrelation as

$$b_m(n,n)=\hat{y}(n-M-1)+\hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{B}_{\mathbf{M}}(n) \qquad (4.143)$$

where, it should be noted, $b_m(n,n)$ is a scalar. Comparing Equation 4.143 with Equation 4.100, to determine the basic relationship between $b_m(n,n)$ and $b_M(n)$ substitute Equation 4.107 for $\mathbf{B}_{\mathbf{M}}(n)$ in Equation 4.143:

$$\begin{aligned}b_m(n,n) &= \hat{y}(n-M-1)+\hat{\mathbf{Y}}'_{\mathbf{M}}(n)\left[\mathbf{B}_{\mathbf{M}}(n-1)-\mathbf{K}_{\mathbf{M}}(n)b_M(n)\right]\\ &= \hat{y}(n-M-1)+\hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{B}_{\mathbf{M}}(n-1)-b_M(n)\hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{K}_{\mathbf{M}}(n)\\ &= b_M(n)\left[1-\hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{K}_{\mathbf{M}}(n)\right]\\ &= \gamma_M(n)b_M(n)\ =\ \frac{b_M(n)}{\alpha_M(n)}\ . \qquad (4.144)\end{aligned}$$

Using Equations 4.111, 4.143, and 4.144 in Equation 4.142 yields

$$\begin{aligned}\gamma_{M+1}(n) &= \gamma_M(n) - \frac{h_M(n)b_M(n,n)}{\mathcal{B}_M(n)} \\ &= \gamma_M(n)\left[1 - \frac{h_M(n)b_M(n)}{\mathcal{B}_M(n)}\right] \quad . \end{aligned} \tag{4.145}$$

Thus, according to Equations 4.111 and 4.145

$$1 - \frac{h_M(n)b_M(n)}{\mathcal{B}_M(n)} = \frac{\gamma_{M+1}(n)}{\gamma_M(n)} = \frac{\alpha_M(n)}{\alpha_{M+1}(n)} \quad . \tag{4.146}$$

Furthermore, substituting Equations 4.134 and 4.135 into the LHS of Equation 4.146,

$$\begin{aligned}1 - \frac{h_M(n)b_M(n)}{\mathcal{B}_M(n)} &= \frac{\mathcal{B}_M(n) - h_M(n)b_M(n)}{\mathcal{B}_M(n)} \\ &= \frac{\lambda\mathcal{B}_M(n-1) + h_M(n)b_M(n) - h_M(n)b_M(n)}{\mathcal{B}_M(n)} \\ &= \frac{\lambda\mathcal{B}_M(n-1)}{\mathcal{B}_M(n)}\end{aligned}$$

and Equation 4.140 is proved. Next, substitute the results of Equation 4.146 into Equation 4.139 and then premultiply both sides of Equation 4.139 by $\alpha_{M+1}(n)$ to get

$$\alpha_{M+1}(n)\mathbf{K}_{\mathrm{M+1}}(n) = \begin{bmatrix} \alpha_M(n)\mathbf{K}_{\mathrm{M}}(n) \\ 0 \end{bmatrix} + \frac{h_M(n)\alpha_{M+1}(n)}{\mathcal{B}_M(n)}\mathbf{b}_{\mathrm{M}}(n-1) \ .$$

Noting from Equation 4.115 that

$$\frac{\alpha_{M+1}(n)}{\mathcal{B}_M(n)} = \frac{\alpha_M(n)}{\lambda\mathcal{B}_M(n-1)}$$

and using the definition of Equation 4.110, the extended fast Kalman gain vector of Equation 4.136 can be rewritten in terms of the FTF alternative Kalman gain vector as

$$\tilde{\mathbf{K}}_{\mathrm{M+1}}(n) = \begin{bmatrix} \tilde{\mathbf{K}}_{\mathrm{M}}(n) \\ 0 \end{bmatrix} + \frac{h_M(n)\alpha_M(n)}{\lambda\mathcal{B}_M(n-1)} \begin{bmatrix} \mathbf{B}_{\mathrm{M}}(n-1) \\ 1 \end{bmatrix} \tag{4.147}$$

where $\tilde{\mathbf{K}}_{\mathrm{M+1}}(n)$ can be partitioned as in Equation 4.125.

All that remains in the proof of the FTF with decorrelation is to write Equations 4.19, 4.101, 4.106, and 4.107 in terms of $\tilde{\mathbf{K}}_{\mathrm{M}}(n)$ and derive a time-update recursion for the scalar $\alpha_M(n)$. First, substitution of Equation 4.119 into Equation 4.101 and using the definition of Equation 4.110, the forward prediction coefficients can be written in terms of the alternative Kalman gain vector as

$$\mathbf{F}_{\mathrm{M}}(n) = \mathbf{F}_{\mathrm{M}}(n-1) - \tilde{\mathbf{K}}_{\mathrm{M}}(n-1) f_M(n,n) \ . \tag{4.148}$$

Next, substitution of Equation 4.144 into Equation 4.107 and using the definition of Equation 4.110, the backward prediction coefficients can be written as

$$\mathbf{B}_{\mathrm{M}}(n) = \mathbf{B}_{\mathrm{M}}(n-1) - \tilde{\mathbf{K}}_{\mathrm{M}}(n) b_M(n,n) \ . \tag{4.149}$$

To write the tap weight update recursion of Equation 4.19, it will first be necessary to define the *a posteriori* estimation error for decorrelation as

$$\eta_M(n,n) = x(n) - \hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{W}_{\mathrm{M}}(n) \tag{4.150}$$

where, again, $\eta_M(n,n)$ is a scalar. Comparing Equation 4.150 with Equation 4.18, to determine the basic relationship between $\eta_M(n,n)$ and $\eta_M(n)$ substitute Equation 4.19 for $\mathbf{W}_{\mathrm{M}}(n)$ in Equation 4.150:

$$\begin{aligned}
\eta_M(n,n) &= x(n) - \hat{\mathbf{Y}}'_{\mathrm{M}}(n)\left[\mathbf{W}_{\mathrm{M}}(n-1) + \eta_M(n)\mathbf{K}_{\mathrm{M}}(n)\right] \\
&= x(n) - \hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{W}_{\mathrm{M}}(n-1) - \eta_M(n)\hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{K}_{\mathrm{M}}(n) \\
&= \eta_M(n)\left[1 - \hat{\mathbf{Y}}'_{\mathrm{M}}(n)\mathbf{K}_{\mathrm{M}}(n)\right] \\
&= \gamma_M(n)\eta_M(n) \quad = \quad \frac{\eta_M(n)}{\alpha_M(n)}
\end{aligned} \tag{4.151}$$

Consequently, substituting Equation 4.151 into Equation 4.19 and using the definition of Equation 4.110, the tap weight recursion can be written as

$$\mathbf{W}_{\mathrm{M}}(n) = \mathbf{W}_{\mathrm{M}}(n-1) + \eta_M(n,n)\tilde{\mathbf{K}}_{\mathrm{M}}(n) \ . \tag{4.152}$$

A comparison of Equations 4.124 and 4.147 with the partitioning of Equation 4.125 reveals the following useful identity:

$$\tilde{c}_M(n) = \frac{h_M(n)\alpha_M(n)}{\lambda \mathcal{B}_M(n-1)} \tag{4.153}$$

where $\tilde{c}_M(n)$ is a scalar. Equation 4.153 will be used to formulate the necessary time-update recursion for $\tilde{\mathbf{K}}_{\mathbf{M}}(n)$. An inspection of Equation 4.147 with the partitioning of Equation 4.125 reveals that $\tilde{\mathbf{C}}_{\mathbf{M}}(n)$ can be written as

$$\tilde{\mathbf{C}}_{\mathbf{M}}(n) = \tilde{\mathbf{K}}_{\mathbf{M}}(n) + \frac{h_M(n)\alpha_M(n)}{\lambda \mathcal{B}_M(n-1)} \mathbf{B}_{\mathbf{M}}(n-1) \ .$$

Substitution of the identity of Equation 4.153 and isolating $\tilde{\mathbf{K}}_{\mathbf{M}}(n)$ on the RHS of the above equation yields the necessary recursion:

$$\tilde{\mathbf{K}}_{\mathbf{M}}(n) = \tilde{\mathbf{C}}_{\mathbf{M}}(n) - \tilde{c}_M(n)\mathbf{B}_{\mathbf{M}}(n-1) \ . \tag{4.154}$$

The final step in the derivation of the FTF with decorrelation is to determine the necessary time-update recursion for $\alpha_M(n)$. To do so, note that using the definition of Equation 4.110 in the definition of Equation 4.109, $\alpha_{M+1}(n)$ can be written as

$$\alpha_{M+1}(n) = 1 + \hat{\mathbf{Y}}'_{\mathbf{M+1}}(n)\tilde{\mathbf{K}}_{\mathbf{M+1}}(n) \ . \tag{4.155}$$

Therefore, using the first expression for $\hat{\mathbf{Y}}_{\mathbf{M+1}}(n)$ in Equation 4.36 and using Equation 4.124 for $\tilde{\mathbf{K}}_{\mathbf{M+1}}(n)$, Equation 4.155 can be written as

$$\begin{aligned}
\alpha_{M+1}(n) &= \\
&= 1 + \left[\hat{y}(n)\hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\right]\left\{\begin{bmatrix} 0 \\ \tilde{\mathbf{K}}_{\mathbf{M}}(n-1) \end{bmatrix} + \frac{g_M(n)\alpha_M(n-1)}{\lambda \mathcal{F}_M(n-1)}\begin{bmatrix} 1 \\ \mathbf{F}_{\mathbf{M}}(n-1) \end{bmatrix}\right\} \\
&= 1 + \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\tilde{\mathbf{K}}_{\mathbf{M}}(n-1) \\
&\qquad + \frac{g_M(n)\alpha_M(n-1)}{\lambda \mathcal{F}_M(n-1)}\left(\hat{y}(n) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n-1)\mathbf{F}_{\mathbf{M}}(n-1)\right) \\
&= \alpha_M(n-1) + \frac{g_M(n)\alpha_M(n-1)}{\lambda \mathcal{F}_M(n-1)} f_M(n)
\end{aligned} \tag{4.156}$$

where Equation 4.99 was used in the last line. To determine the current realization of $\alpha_M(n)$, using the second expression for $\hat{\mathbf{Y}}_{\mathbf{M+1}}(n)$ in Equation 4.36, and Equation 4.147 for $\tilde{\mathbf{K}}_{\mathbf{M+1}}(n)$, Equation 4.155 can be written as

$$\begin{aligned}
\alpha_{M+1}(n) &= 1 + \left[\hat{\mathbf{Y}}'_{\mathbf{M}}(n)\hat{y}(n-M)\right]\left\{\begin{bmatrix}\tilde{\mathbf{K}}_{\mathbf{M}}(n)\\ 0\end{bmatrix} + \frac{h_M(n)\alpha_M(n)}{\lambda\mathcal{B}_M(n-1)}\begin{bmatrix}\mathbf{B}_{\mathbf{M}}(n-1)\\ 1\end{bmatrix}\right\}\\
&= 1 + \hat{\mathbf{Y}}'_{\mathbf{M}}(n)\tilde{\mathbf{K}}_{\mathbf{M}}(n) + \frac{h_M(n)\alpha_M(n)}{\lambda\mathcal{B}_M(n-1)}\left(\hat{y}(n-M) + \hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{B}_{\mathbf{M}}(n-1)\right)\\
&= \alpha_M(n) + \frac{h_M(n)\alpha_M(n)}{\lambda\mathcal{B}_M(n-1)}b_M(n) \qquad (4.157)
\end{aligned}$$

where Equation 4.100 was used in the last line. Substituting the identity of Equation 4.153 and isolating $\alpha_M(n)$ on the RHS of Equation 4.157, the desired time-update recursion for $\alpha_M(n)$ is

$$\alpha_M(n) = \alpha_{M+1}(n) - \hat{c}_M(n)b_M(n)\ . \qquad (4.158)$$

Therefore, $\alpha_{M+1}(n)$ must first be calculated according to Equation 4.156 and then $\alpha_M(n)$ can be updated using Equation 4.158.

4.3.2 The Algorithm

The algorithm which defines the fast transversal filter with decorrelation is now complete. The order which constitutes the fast transversal filter with decorrelation is given below:

$$f_M(n) = \hat{y}(n-1) + \mathbf{F}'_{\mathbf{M}}(n-1)\hat{\mathbf{Y}}_{\mathbf{M}}(n-1) \qquad (4.159)$$

$$f_M(n,n) = \frac{f_M(n)}{\alpha_M(n-1)} \qquad (4.160)$$

$$\mathbf{F}_{\mathbf{M}}(n) = \mathbf{F}_{\mathbf{M}}(n-1) - \tilde{\mathbf{K}}_{\mathbf{M}}(n-1)f_M(n,n). \qquad (4.161)$$

$$\mathbf{S}_{\mathbf{M}}(n) = \lambda\mathbf{S}_{\mathbf{M}}(n-1) + \hat{\mathbf{Y}}_{\mathbf{M}}(n-1)y(n-1) \qquad (4.162)$$

$$\tilde{\mathbf{S}}_{\mathbf{M}}(n) = \frac{\mathbf{S}_{\mathbf{M}}(n)}{\alpha_M(n-1)} \qquad (4.163)$$

$$g_M(n) = y(n-1) - \tilde{\mathbf{S}}'_{\mathbf{M}}(n)\tilde{\mathbf{K}}_{\mathbf{M}}(n-1) \qquad (4.164)$$

$$\mathcal{F}_M(n) = \lambda \mathcal{F}_M(n-1) + g_M(n) f_M(n) \tag{4.165}$$

$$\tilde{\mathbf{K}}_{\mathbf{M+1}}(n) = \left[\begin{array}{c} 0 \\ \tilde{\mathbf{K}}_{\mathbf{M}}(n-1) \end{array} \right] + \frac{g_M(n)\alpha_M(n-1)}{\lambda \mathcal{F}_M(n-1)} \left[\begin{array}{c} 1 \\ \mathbf{F}_{\mathbf{M}}(n-1) \end{array} \right] \quad . \tag{4.166}$$

Partition $\tilde{\mathbf{K}}_{\mathbf{M+1}}(n)$ according to

$$\tilde{\mathbf{K}}_{\mathbf{M+1}}(n) \equiv \left[\begin{array}{c} \tilde{\mathbf{C}}_{\mathbf{M}}(n) \\ \tilde{c}_M(n) \end{array} \right] \quad . \tag{4.167}$$

Let

$$\tilde{\mathbf{K}}_{\mathbf{M}}(n) = \tilde{\mathbf{C}}_{\mathbf{M}}(n) - \tilde{c}_M(n)\mathbf{B}_{\mathbf{M}}(n-1) \tag{4.168}$$

$$\alpha_{M+1}(n) = \alpha_M(n-1) + \frac{g_M(n)\alpha_M(n-1)}{\lambda \mathcal{F}_M(n-1)} f_M(n) \tag{4.169}$$

$$b_M(n) = \hat{y}(n-M-1) + \mathbf{B}'_{\mathbf{M}}(n-1)\hat{\mathbf{Y}}_{\mathbf{M}}(n) \tag{4.170}$$

$$\alpha_M(n) = \alpha_{M+1}(n) - \tilde{c}_M(n) b_M(n) \tag{4.171}$$

$$b_M(n,n) = \frac{b_M(n)}{\alpha_M(n)} \tag{4.172}$$

$$\mathbf{B}_{\mathbf{M}}(n) = \mathbf{B}_{\mathbf{M}}(n-1) - \tilde{\mathbf{K}}_{\mathbf{M}}(n) b_M(n,n) \tag{4.173}$$

$$\eta_M(n) = x(n) - \hat{\mathbf{Y}}'_{\mathbf{M}}(n)\mathbf{W}_{\mathbf{M}}(n-1) \tag{4.174}$$

$$\eta_M(n,n) = \frac{\eta_M(n)}{\alpha_M(n)} \tag{4.175}$$

$$\mathbf{W}_{\mathbf{M}}(n) = \mathbf{W}_{\mathbf{M}}(n-1) + \eta_M(n,n)\tilde{\mathbf{K}}_{\mathbf{M}}(n) \quad . \tag{4.176}$$

To initialize the FTF for decorrelation algorithm, set all vectors and quantities to zero, except let $\mathcal{F}_M(-1)=\delta > 0$ and $\alpha_M(-1)=1$ [37]. A count on the number of MADPI of the FTF algorithm necessary to update the tap weight vector reveals that the FTF requires approximately $10M + 16$ MADPI. The modest increase in MADPI over the fast Kalman algorithm is a direct result of the asymmetry present in the augmented correlation matrix of Equation 4.38. However, this is still a substantial savings over the $O(M^2)$ complexity of the RLC algorithm.

Several remarks need to be made regarding the algorithm of Equations 4.159 through 4.176. First, a simple inspection of Equation 4.166 reveals that there is no longer any coupling between the computation of the extended Kalman gain vector and the calculation of the forward prediction coefficients. This comes as a result of the fact that $\mathbf{F_M}(n-1)$ and not $\mathbf{F_M}(n)$ appears in Equation 4.166.

Finally, an interesting result becomes apparent when the expressions defining the relationships between the *a priori* and *a posteriori* errors are compared. A remarkable consequence of Equations 4.160, 4.172, and 4.175 (comparing them to Equations 4.118, 4.143, and 4.150, respectively) is that the *a posteriori* errors at time n can be computed before the filter parameters producing them, *i.e.*, before the computation of $\mathbf{F_M}(n)$, $\mathbf{B_M}(n)$, and $\mathbf{W_M}(n)$, respectively [5]. It is for this reason that $\alpha_M(n)$ is referred to in the literature as a conversion factor [18]. In either its regular or delayed form, $\alpha_M(n)$ converts the *a priori* forward, backward, and estimation errors into the corresponding *a posteriori* errors.

4.4 Simulation Results for the Fast Algorithms

In this section, the rate of convergence of the fast Kalman (FRLC) algorithm, given by Equations 4.99 through 4.107, and the FTF, given by Equations 4.159 through 4.176, will be compared to each other and to the Kalman (RLC) algorithm given in the first section. Both the blind, decorrelation fast Kalman and FTF in a decision feedback configuration were simulated for two channels representing pure, heavy amplitude distortion. The model of the DFE used in these simulations is shown in Figure 4.2.

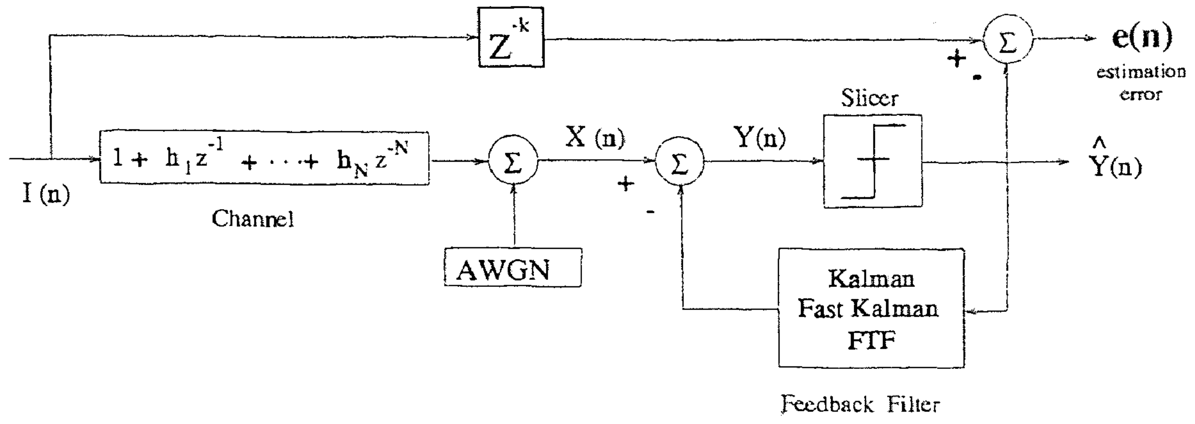

Figure 4.2 Decision Feedback Equalizer Used for Simulations.

For each simulation of each of the respective structures, a nine-tap equalizer with a weighting factor of λ=0.999 was used. The particular channel sampled impulse response used in all of the simulations was the raised-cosine pulse, defined by

$$h(k) = \begin{cases} \frac{1}{2\sqrt{\mathrm{h}'\mathrm{h}}}\left[1+\cos\left(\frac{2\pi}{W}(k-3)\right)\right], & \mathrm{k} = 1,2,3,4,5 \\ 0 & \text{otherwise} \end{cases} \tag{4.177}$$

and $\mathrm{h} = [h_1, h_2, h_3, h_4, h_5]'$, where W in Equation 4.177 was set equal to either 3.1 or 3.6 to provide for an eigenvalue ratio of 11 or 49, respectively. Note that the first cursor, h_0, is chosen to be 1 so that the channel satisfies the criterion for intersymbol interference. The channel's impulse response is normalized such that $h_1^2+h_2^2+h_3^2+h_4^2+h_5^2 = 1$ for all values of the bandwidth parameter W. Additive white Gaussian noise of zero-mean and variance 0.001 was added to the output of the channel to form the received waveform $x(n)$. The channel model is similar to the one used by Satorius and Alexander in [40], where the normalization was added by Axford in [1]. To provide a bound on which to judge the convergence rate performance of these rapidly converging algorithms, the decorrelation DFE presented in Chapter 2 was also implemented. The performance of the fast Kalman and decorrelation equalizers for W=3.1 is shown in the learning curve of Figure 4.3.

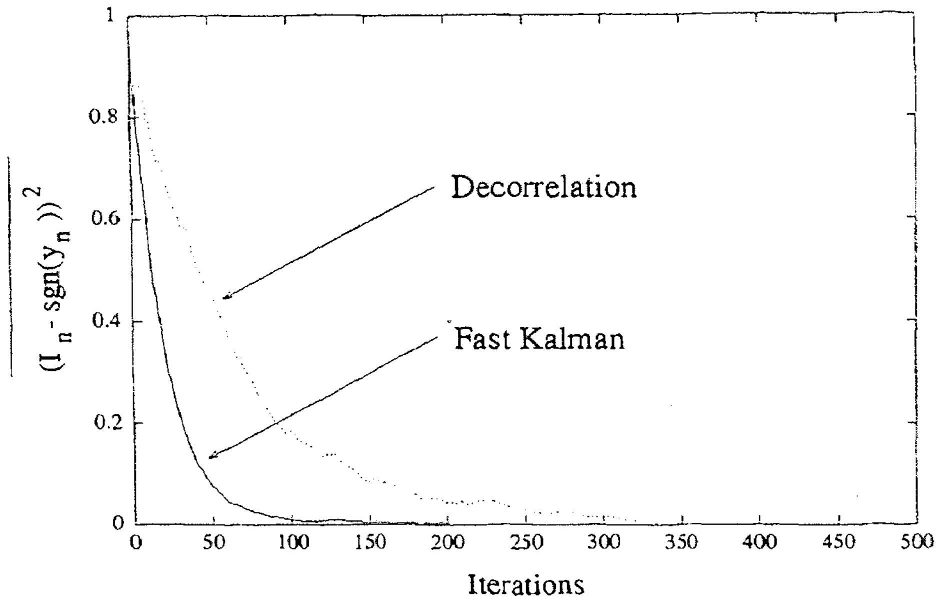

Figure 4.3 MSE of the Fast Kalman DFE with W=3.1.

The performance of the fast transversal filter-based and decorrelation equalizers for W=3.1 is shown in the learning curve of Figure 4.4. In these figures, the estimate of the residual ISI power is obtained by passing the sequence of the squared error $(I(n) - \hat{y}(n-1))^2$ through a smoothing filter whose transfer function is given by $0.05/(1 - 0.95z^{-1})$ [20]. This particular definition of the MSE is the same as used by Satorius and Pack [41], Satorius and Alexander [40], and Haykin [18]. Each curve was obtained by the Monte-Carlo averaging of the MSE over 100 independent trials. In these simulations, the length of the delay of the element, z^{-k}, of Figure 4.2 was chosen to produce the smallest MSE for the algorithms being considered [40]. Unless specified, this delay was chosen to be z^{-1}.

As mentioned previously, the speed of convergence of the fast Kalman and FTF algorithms will be compared with the Kalman-based (RLC) algorithm given in the first section of this chapter. The reason for this comparison is that the Kalman algorithm, as shown by Godard [14] and modified in [24], has been recognized to be the fastest known equalizer adaption algorithm [38]. The Kalman algorithm is an

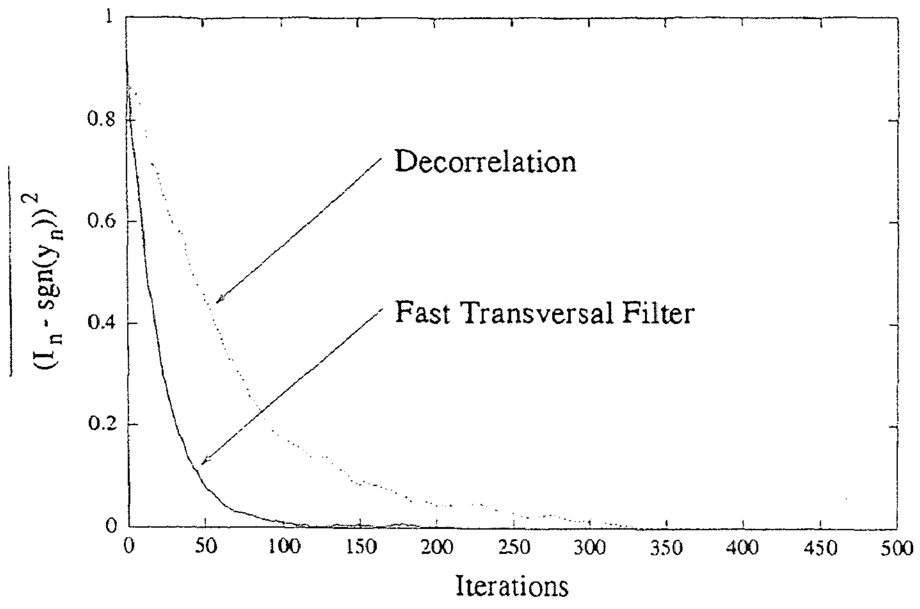

Figure 4.4 MSE of the FTF DFE with W=3.1.

ideal self-orthogonalizing algorithm in that the received equalizer input signals are used to build up the inverse of the input correlation matrix which is applied to the coefficient adjustment process [12]. Consequently, the algorithm of Equations 4.20 through 4.26 was also implemented using the channel given by Equation 4.177. The learning curve of the mean squared error for the Kalman (RLC) algorithm (compared with the decorrelation DFE) is shown in Figure 4.5.

By comparing Figures 4.3, 4.4, and 4.5, it can be seen that the fast Kalman and FTF algorithms offers virtually identical performance to the Kalman algorithm—all algorithms converge in approximately 100 iterations. This is a substantial increase over the convergence rate of the corresponding decorrelation DFE. The intimate relationship between the convergence rates of the three rapidly converging algorithms is more readily apparent when the three separate curves are overlayed on one another, as done in Figure 4.6. It should be emphasized that the fast Kalman for decorrelation and FTF for decorrelation algorithms are mathematically equivalent to the RLC algorithm, resulting in their comparable performance.

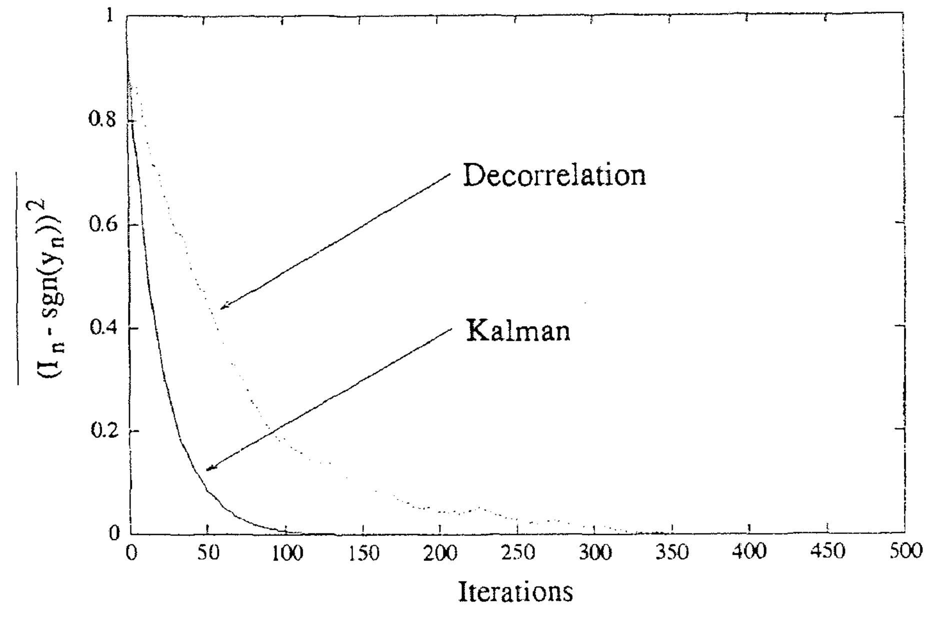

Figure 4.5 MSE of the RLC Algorithm with W=3.1.

The learning curves of the fast Kalman- and FTF-based equalizers (using the convergence rate of the decorrelation DFE as a reference) with W=3.6 in Equation 4.177 are shown in Figures 4.7 and 4.8, respectively. Again, to compare the performance of the fast Kalman and FTF algorithms, the Kalman (RLC) algorithm is implemented with W=3.6, as shown in Figure 4.9. By comparing Figures 4.7 and 4.8 with Figure 4.9, it can be seen that the fast Kalman and FTF algorithms again offer virtually identical performance to the RLC algorithm, offering a substantial increase in performance as compared with the decorrelation DFE. As had been done previously, this conclusion becomes more readily apparent when the three separate curves are overlayed on one another, as done in Figure 4.10. It should be noted that the comparable performance characteristics of the FRLC and FTF algorithms to the RLC algorithm are a direct result of the mathematical equivalence of the three algorithms.

One final remark concerning the learning curves of Figures 4.6 and 4.10 is necessary. A comparison of the two curves reveals that the convergence rate of the

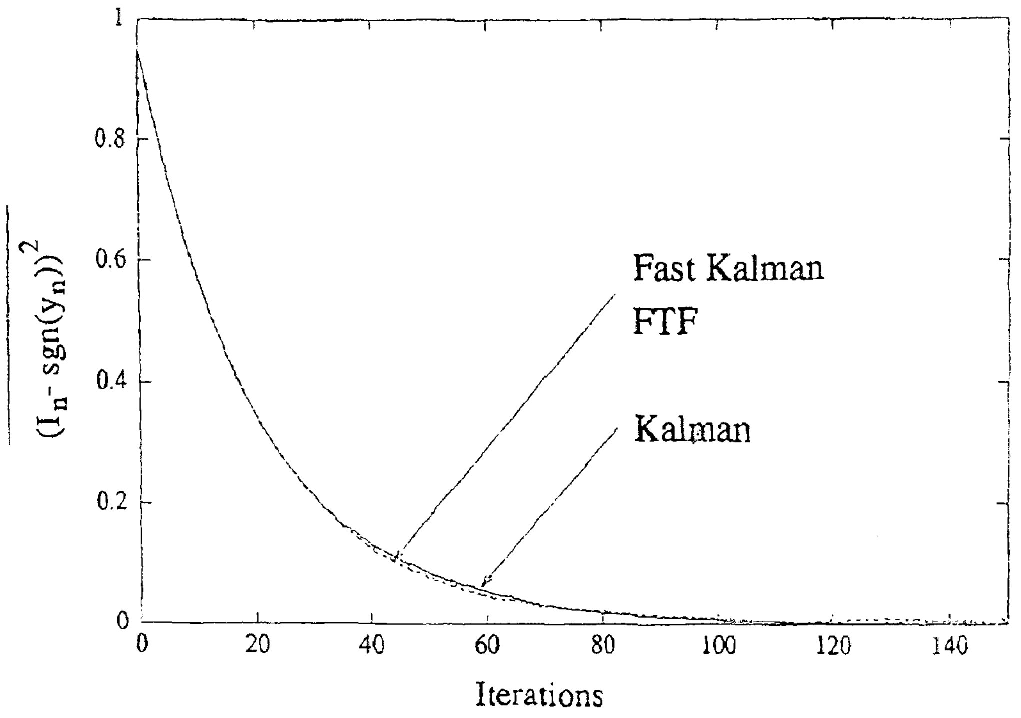

Figure 4.6 Performance Comparison of the FRLC, FTF, and RLC Algorithms with W=3.1.

fast Kalman (FRLC), FTF, and Kalman (RLC) algorithms are reasonably insensitive to the eigenvalue spread of the channel. This agrees well with theory. Since all three algorithms are types of self-orthogonalizing equalizers, the convergence rates of the respective equalizers do not depend on the eigenvalue spread of the channel [18].

4.5 Numerical Properties of the Fast Kalman and FTF Algorithms

As mentioned throughout this chapter, the fast Kalman and FTF algorithms offer the advantage of rapid convergence with a substantial decrease in the overall complexity, compared with the RLC algorithm. However, a disadvantage to the fast Kalman and FTF structures is that they have a tendency to become unstable in finite-precision environments and have exhibited numerical instability [25, 8]. To overcome these problems, several remedies have been proposed which are applicable to the

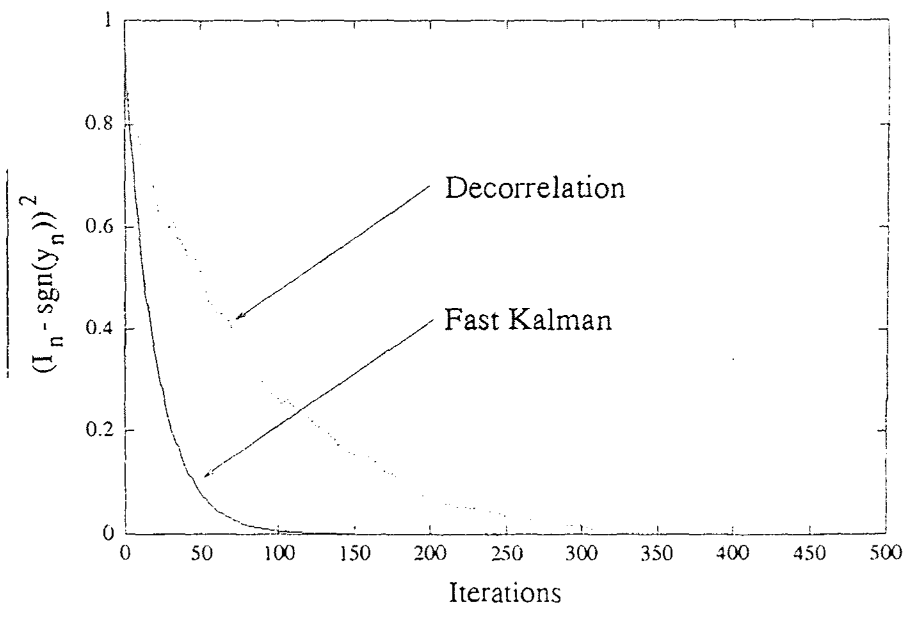

Figure 4.7 MSE of the Fast Kalman DFE with W=3.6.

decorrelation-based implementations of the algorithms presented in this chapter. Two of the most widely used solutions will be discussed.

The simplest of the procedures is a periodic reinitialization of the respective algorithms [25, 9]. Through experimentation, it has been shown [25] that a certain quantity derived for the fast Kalman and FTF algorithms goes negative just before the algorithms diverge. Let

$$\zeta_M(n) = \frac{\alpha_m(n)}{\alpha_{M+1}(n)} \tag{4.178}$$

which is merely a redeclaration of Equation 4.140. It has been shown that $\zeta_M(n)$ is the ratio of two non-negative quantities, and, therefore, is itself a non-negative quantity [25]. For the ideal case of infinite precision, $0 \leq \zeta_M(n) \leq 1$. A violation on this bound of the value of $\zeta_M(n)$ is a direct result of finite-precision effects. Due to the accumulation of finite-precision errors, this quantity becomes negative just before divergence occurs. Therefore, $\zeta_M(n)$ has been termed the *rescue variable* [25].

Using Equation 4.140, Equation 4.178 can be written as

$$\zeta_M(n) = \frac{\lambda \mathcal{B}_M(n-1)}{\mathcal{B}_M(n)} \quad . \tag{4.179}$$

This will be considered the rescue variable pertaining to the fast Kalman algorithm for decorrelation. For the FTF, substitution of the identity of Equation 4.153 into Equation 4.140 yields

$$\zeta_M(N) = 1 - \frac{\tilde{c}_M(n) b_M(n)}{\alpha_{M+1}(n)} \quad . \tag{4.180}$$

Equation 4.180 will be the corresponding rescue variable for the FTF algorithm for decorrelation.

When either of the rescue variables becomes negative, the algorithm must be restarted. However, this now poses a problem. The original fast Kalman and FTF algorithms were derived for what is termed the prewindowed data case. In other words, the relevant input data sequences were considered zero for $n < 0$. If and when these algorithms are restarted, this condition is no longer true. Consequently, the algorithms must be modified for the unwindowed or covariance data case. In Appendix D, the covariance fast Kalman algorithm for decorrelation (CFRLC) has been derived. The algorithm consists of a slight modification to the fast Kalman algorithm of Equations 4.99 through 4.107 and is shown to reduce to the actual fast Kalman algorithm as $n \to \infty$. A corresponding covariance FTF algorithm is not necessary, since an elaborate initialization scheme, derived in [18] and [7], can be used in the case of unwindowed data. The initialization scheme of [18] can be directly applied to the FTF for decorrelation derived in this chapter. The interested reader is referred to either [18] or [7] for details on the initialization. However, it should be noted that a derivation of a corresponding covariance FTF for decorrelation can be made by following the procedure as expressed in [22]. The proof in [22] is similar to that presented in this work, and the appropriate extensions can be readily made.

Therefore, if, at time $n = n_0$, the quantity $\zeta_M(n)$, of Equations 4.179 and 4.180, respectively, is observed going negative, to restart the two fast algorithms of this chapter the following procedure must be performed. First, save the current estimate of the tap weight vector, $\mathbf{W}_{\mathbf{M}}(n_0)$, as its initial condition. Then, allow another equalization algorithm to perform the adaptive process while reinitialization is occurring. Following the conventional approach, the decorrelation DFE presented in Chapter 2 could be used to update the tap weights as reinitialization is occurring. Note that since the reinitialization process lasts for only a short time (approximately M to $1.5M$ iterations), virtually no degradation in the performance of the fast algorithms has been found [7]. Reinitialize all other relevant quantities in the respective algorithms. For the fast Kalman algorithm, the covariance structure will be used after the reinitialization process is complete. According to Appendix D, this will require the use of unwindowed data, so the appropriate non-zero input vectors, $\mathbf{Y}_{\mathbf{M}}(n_0)$ and $\hat{\mathbf{Y}}_{\mathbf{M}}(n_0)$, must also be stored. For the FTF, the initialization procedure as outlined in [18] is to be followed. After completion of the procedure, transfer adaptive control back to the CFRLC or FTF algorithms and proceed with normal operation.

As remarked earlier, this is the simplest of the procedures to deal with accumulated finite-precision errors. Another approach is to use the so-called normalized or stabilized versions of the respective algorithms (see [3], [8], and [43]). In summary, these normalized/stabilized algorithms incorporate square-roots, error feedback, and inherent redundancies in order to limit any effects of a finite word-length environment. However, the disadvantage to these algorithms is in the dramatic increase in complexity that they incur. In some cases, this can result in a two- to three-fold increase in MADPI over the original fast Kalman and FTF algorithms [43].

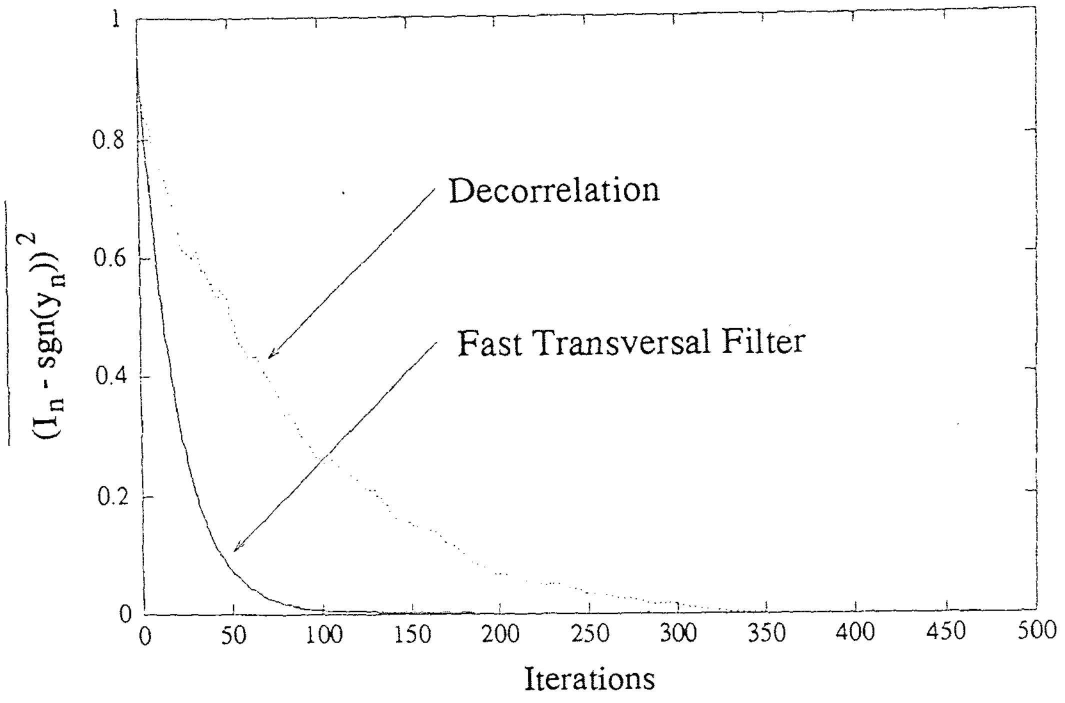

Figure 4.8 MSE of the FTF DFE with W=3.6.

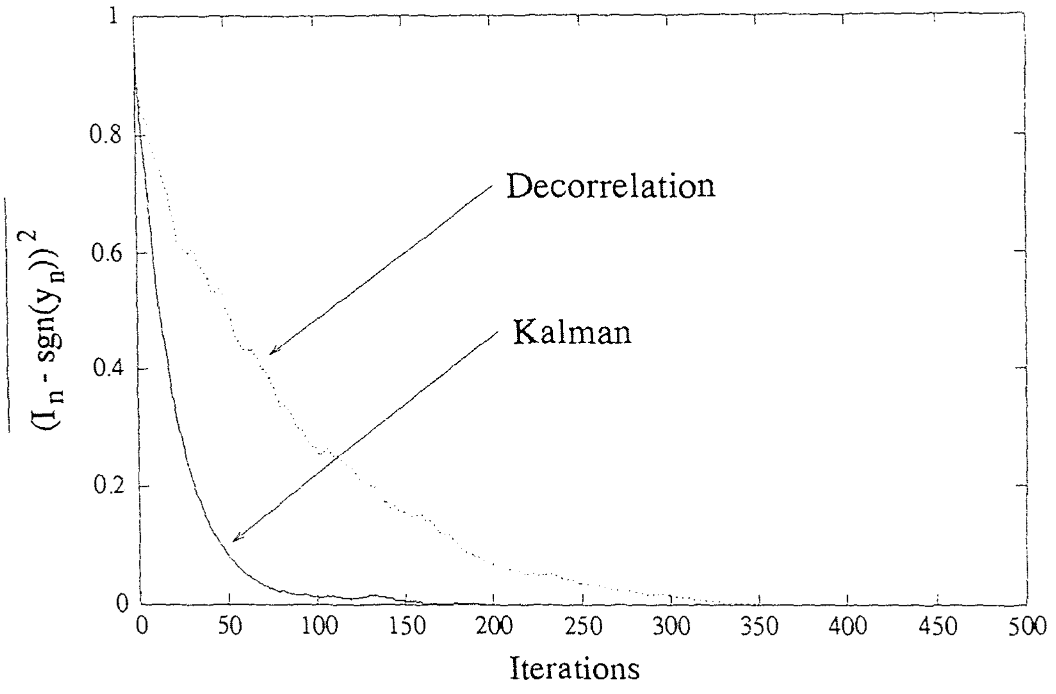

Figure 4.9 MSE of the RLC Algorithm with W=3.6.

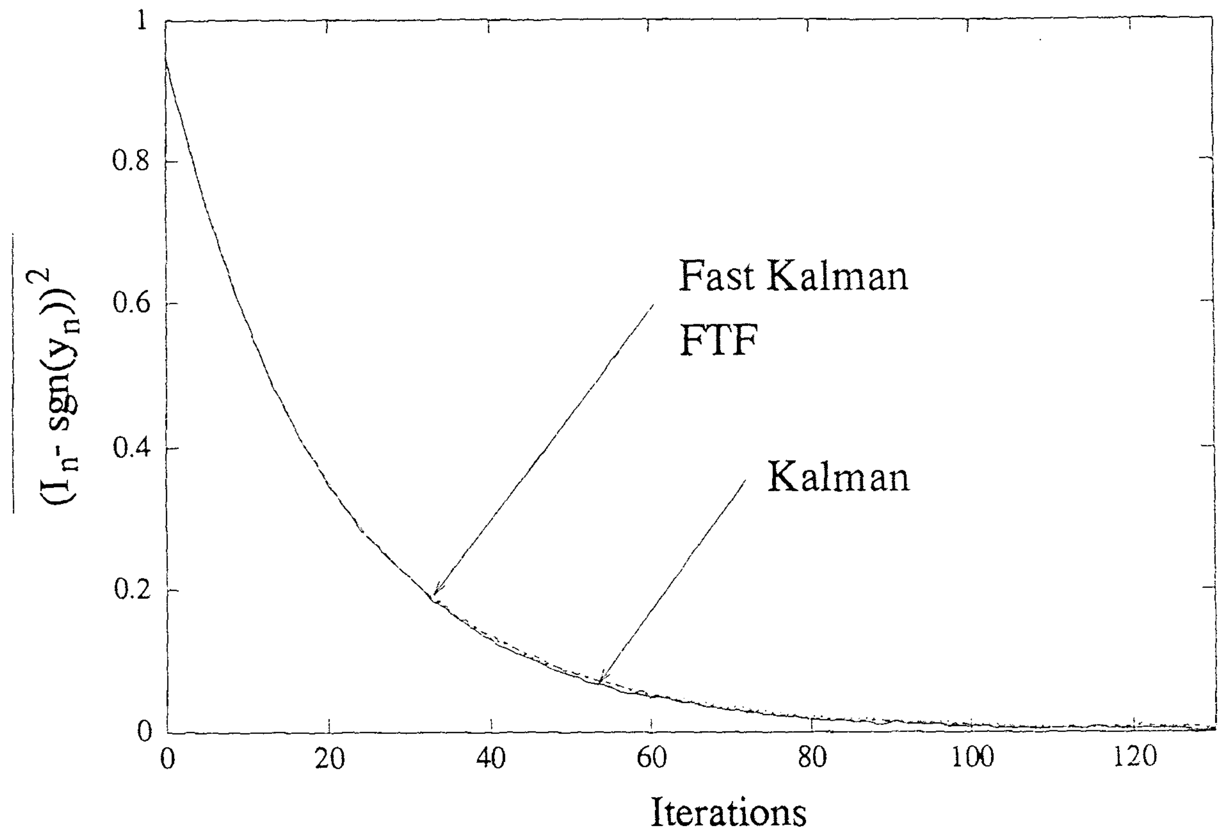

Figure 4.10 Performance Comparison of the FRLC, FTF, and RLC Algorithms with W=3.6.

CHAPTER 5

LATTICE STRUCTURES WITH DECORRELATION

The fast, transversal-based equalizers of Chapter 4 were shown to exhibit several advantageous properties, including low computational complexity and high rate of convergence. However, as was discussed, these algorithms can behave catastrophically in environments governed by a finite word length. Several remedies to overcome these difficulties were proposed. Another approach to solving the finite-precision performance problem is to use an alternative structure for the decorrelation algorithm. Lattice-based structures offer many preferable properties, among them fast convergence rates and the modularity inherent in their implementation. However, the property that is of the most interest is that of their resistance to the accumulated effects of quantization errors in a finite-precision environment. Consequently, in this chapter, the decorrelation algorithm will be applied to several different formulations of the lattice structure.

There are many different implementational forms of the basic lattice filter structure, including both normalized and non-normalized, *a priori* and *a posteriori*, and error-feedback realizations (see [11], [26], and [37], for example). In this chapter, the conventional recursive least-squares lattice, originally developed by Satorius and Pack [41], and the gradient lattice algorithm, originally developed by Satorius and Alexander [40], will be modified to incorporate the decorrelation algorithm. Both structures will be implemented in a decision feedback configuration. As will be discussed, the main structural difference between the RLS and gradient lattices is in the type of computation required of the respective reflection coefficients.

5.1 Recursive Least-Squares Lattice-Ladder with Decorrelation

It has been shown extensively in the literature that it is possible to derive an alternative solution to the direct-form RLS algorithm by incorporating a lattice filter structure [41, 37]. The derivation of the lattice structure is still based on the minimization of the exponentially weighted sum of the output squared error, which results in a form that is mathematically equivalent to the direct-form RLS [37]. It is this relation to the direct-form RLS from which the RLS lattice inherits its fast convergence rate [18]. Although the RLS lattice does maintain this mathematical equivalence, the structure itself is no longer based on a transversal filter.

Unlike the RLC, FRLC, and FTF for decorrelation, derivation of the least-squares lattice based on the decorrelation criterion is not possible, at least not in any conventional sense. As mentioned in Chapter 4, the asymmetry of the decorrelation-based augmented cross-correlation matrix, $\bar{\mathbf{R}}(n)$, of Equation 4.38 prevents the application of the expression for the inverse of a square matrix, which is itself comprised of square matrices on its diagonal. The invertibility of this matrix is a necessary component to the derivation of the order-update recursions which define the RLS lattice [37, 18]. Consequently, because the RLS lattice formulated in this section will not be derived based on the decorrelation algorithm, the mathematical equivalence between the RLC and the RLS lattice for decorrelation will be lost. A direct result of this will be the decrease in the rate of convergence of the decorrelation lattice, compared with the conventional RLS lattice and its relation to the direct-form RLS.

The RLS lattice-ladder equalizer has many desirable properties which make it ideal for adaptive equalization. As discussed, these include its convergence rate, order-recursive nature, and its modest computational complexity (compared with the direct-form RLS). Therefore, it is desirable to formulate the RLS lattice in terms of the decorrelation algorithm. Following the methodology expressed in [1], it is possible to retain the existing RLS lattice structure and incorporate the decorrelation criterion to update the tap weights. The modified algorithm follows the form of the RLS lattice as presented by Satorius and Pack in [41], with the explicit formulation based on that presented in [33]. Since the RLS lattice will not be derived based on the decorrelation criterion, the lengthy proof which comprises the derivation of the RLS lattice structure is unnecessary, and will not be presented in this work. The interested reader is referred to [37], [18], or [33] for the complete derivation.

The RLS lattice with decorrelation will be implemented in a decision feedback configuration. Traditional approaches to the lattice DFE use a multi-channel configuration, with the forward filter comprised of a single channel lattice and the feedback filter comprised of a two-channel lattice [28, 27, 42, 36]. The signal which is fed back is actually the estimation error at each stage of the feedback (two-channel) portion of the DFE lattice [28]. Since this particular formulation is not applicable to the decorrelation algorithm, an alternative scheme for the lattice DFE will be used. The structure of the adaptive least-squares lattice/joint-process (ladder) estimator using decorrelation is shown in Figures (5.1) and (5.2), where Figure (5.2) represents one stage of the lattice equalizer.

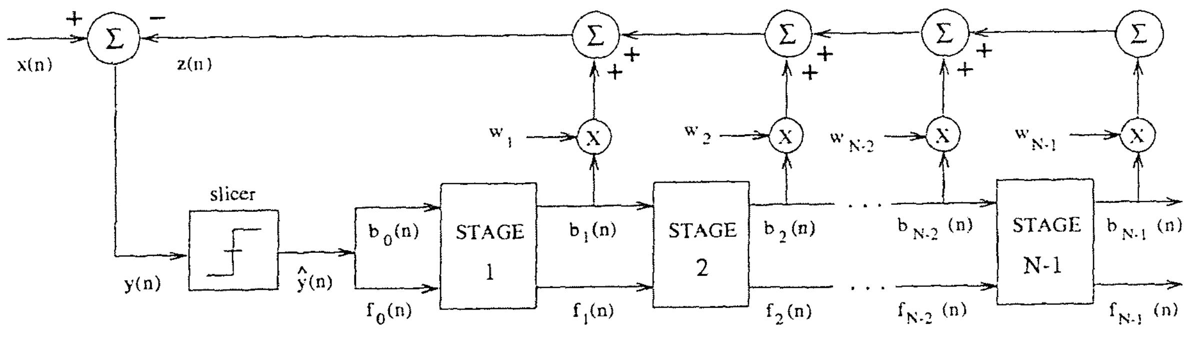

Figure 5.1 Structure of Lattice-Based Decorrelation DFE.

As noted in Chapter 2, since the channels under consideration do not require the use of a forward filter, a feedforward lattice is not used in Figure 5.1. Unlike the lattice DFE of [28], the lattice structure of Figure 5.1 uses the output of the

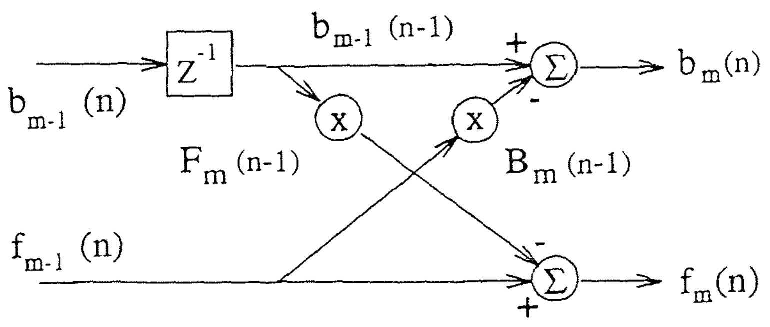

Figure 5.2 Stage m of the RLS Lattice.

slicer as the quantity to be fed back. Note from the previous figures that in distinguishing the RLS lattice-ladder from the gradient lattice-ladder equalizer, the forward and backward reflection coefficients are not equal. Each is independently updated to minimize the weighted sum of squared forward and backward prediction errors, respectively.

The algorithmic formulation of the RLS lattice with decorrelation is as follows. At time n, the inputs to the first lattice stage are set to the newly received output of the slicer (see Figure 5.1):

$$f_0(n) = b_0(n) = \hat{y}(n) \ .$$

The order-update recursions for the estimated sum of the squared forward and backward prediction errors ($E^f_m(n)$ and $E^b_m(n)$, respectively, at stage m) are initialized as follows:

$$E^f_0(n) = E^b_0(n) = \lambda E^f_0(n-1) + \hat{y}(n)\hat{y}(n) \ . \qquad (5.1)$$

For stages $m = 1, 2, \ldots, M-1$ the order updates for the RLS lattice-ladder recursions are performed in the following manner. Referring to Figure 5.2, the forward prediction errors are updated according to

$$f_m(n) = f_{m-1}(n) - F_m(n-1)b_{m-1}(n-1) \qquad (5.2)$$

and the backward prediction errors according to

$$b_m(n) = b_{m-1}(n-1) - B_m(n-1)f_{m-1}(n). \tag{5.3}$$

Calculate the scalar $K_m(n)$ according to the time-update recursion

$$K_m(n) = \lambda K_m(n-1) + t_m(n-1)f_{m-1}(n) \tag{5.4}$$

where the scalar $t_m(n)$ is referred to as the adaptive step-size parameter [18]. The quantity, $K_m(n)$, is used in both of the update equations for the forward and backward reflection coefficients. The forward reflection coefficients of Figure 5.2 are updated in the following manner:

$$F_m(n) = \frac{K_m(n)}{E^b_{m-1}(n-1)} \tag{5.5}$$

while the backward reflection coefficients are updated according to

$$B_m(n) = \frac{K_m(n)}{E^f_{m-1}(n)} \quad . \tag{5.6}$$

The estimated sum of the squared forward and backward prediction errors are updated according to the following order recursions:

$$E^f_m(n) = E^f_{m-1}(n) - F_m(n)K_m(n) \tag{5.7}$$

$$E^b_m(n) = E^b_{m-1}(n) - B_m(n)K_m(n) \tag{5.8}$$

respectively. The adaptive step size parameter, $t_m(n)$, used in the update equation for $K_m(n)$ is itself updated according to

$$t_m(n) = [1 - \gamma_{m-1}(n)]\, b_{m-1}(n) \quad . \tag{5.9}$$

The estimation error or conversion factor, $\gamma_{m-1}(n)$, is updated according to

$$\gamma_m(n) = \gamma_{m-1}(n) + \frac{|t_m(n)|^2}{E^b_{m-1}(n)} \quad . \tag{5.10}$$

It should be noted that this parameter enables the RLS lattice algorithm to adapt rapidly to sudden changes in the input data [18]. The output of the RLS lattice-ladder is formed by

$$z(n) = \sum_{k=1}^{N-1} w_k(n) b_k(n) \ . \tag{5.11}$$

Finally, update the tap weight coefficients in the following manner using the decorrelation criterion:

$$w_m(n+1) = w_m(n) + \mu y(n) y(n-m) \tag{5.12}$$

where μ is the step size which controls the speed of adaption of the tap weight algorithm. The components can be initialized as follows

$$\begin{aligned}
f_m(-1) &= b_m(-1) = 0 \\
F_m(-1) &= B_m(-1) = 0 \\
K_m(-1) &= t_m(-1) = \gamma_m(-1) = 0 \\
E^f_m(-1) &= E^b_m(-1) = \delta > 0 \\
w_m(0) &= 0 \\
z(0) &= 0 \ .
\end{aligned}$$

The lattice DFE based on the decorrelation criterion is shown in Figure 5.3.

One important property of the lattice structure in the context of channel equalization is its ability to transform $\{\hat{y}(m), \hat{y}(m-1), \ldots, \hat{y}(m-M+1)\}$, the correlated input sequence, into $\{b_0(m), b_1(m), \ldots, b_M(m)\}$, the uncorrelated sequence of backward prediction errors. This process may be viewed as a deterministic form of the Gram-Schmidt orthogonalization procedure [18]. The increased rate of convergence of the decorrelation algorithm using a lattice structure is related to the self-orthogonalizing nature of the RLS-lattice [13, 18]. This assertion will become more apparent in the simulations to follow.

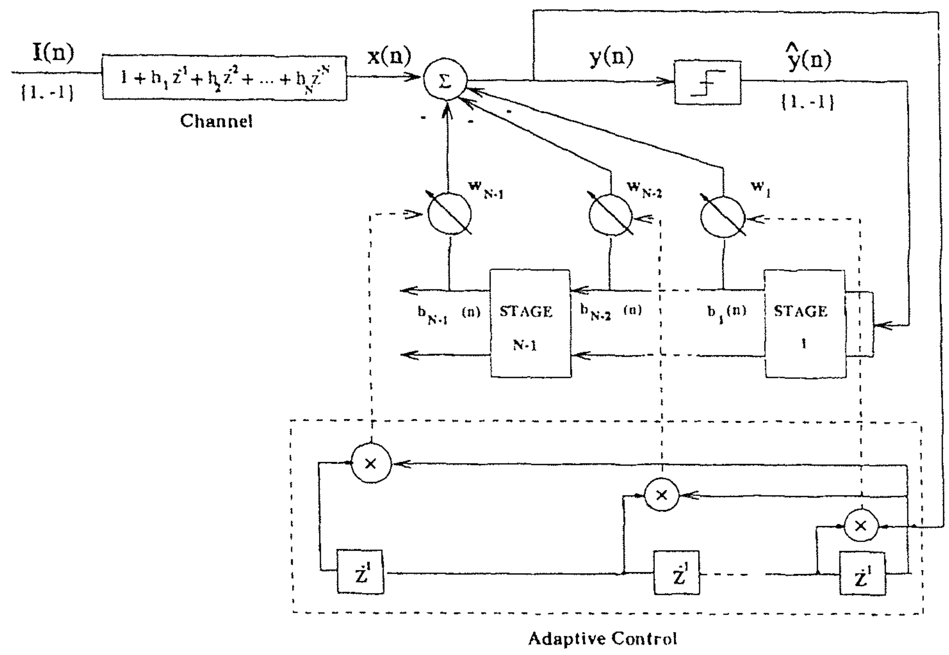

Figure 5.3 Lattice DFE Incorporating the Decorrelation Algorithm.

5.2 Gradient Lattice-Ladder with Decorrelation

In an attempt to simplify the computational aspects of this particular class of algorithms, while still retaining many of their optimal qualities, it is possible to introduce an alternative lattice-ladder structure in which the number of filter parameters is significantly reduced. In keeping with the methodology as expressed in the previous section and in [1], the existing gradient lattice structure will be retained, with the decorrelation criterion used to update the tap weights. The gradient lattice-ladder structure under consideration is shown in Figures 5.4 and 5.5.

Each stage of the lattice is characterized by the following input-output relations:

$$f_m(n) = f_{m-1}(n) - k_m(n-1)b_{m-1}(n-1) \tag{5.13}$$

and

$$b_m(n) = b_{m-1}(n-1) - k_m(n-1)f_{m-1}(n) \tag{5.14}$$

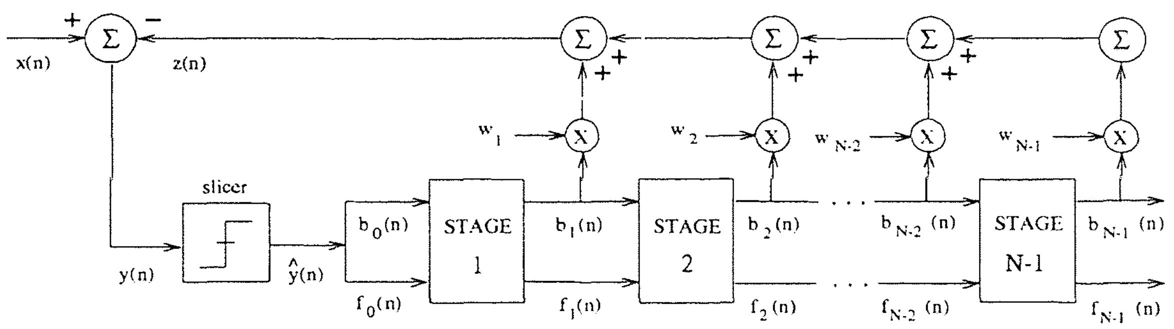

Figure 5.4 Lattice Structure for the Decorrelation Algorithm.

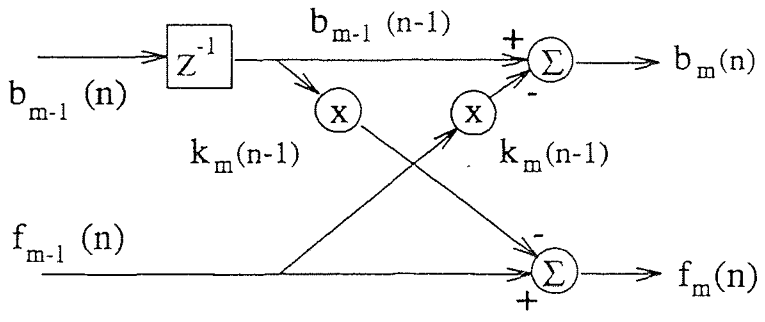

Figure 5.5 Stage m of the Gradient Lattice.

where $k_m(n)$ is the reflection coefficient and $f_m(n)$ and $b_m(n)$ are the forward and backward prediction errors, respectively, of the mth stage of the lattice. It should be noted that this form of the lattice filter is identical to that obtained from the Levinson-Durbin algorithm, except that now $k_m(n)$ is allowed to vary with time so that the lattice filter can adapt the the time variations in the signal statistics [37]. In comparison with the RLS lattice filter, the lattice described by Equations 5.13 and 5.14 are more restrictive in that the forward and backward predictors have identical reflection coefficients.

The reflection coefficients, $k_m(n)$, may be optimized according to either an MSE criterion or by employing the method of least squares. In an adaptive filtering application, since the statistical properties of the signal are unknown, the least-squares criterion will be adopted for determining $k_m(n)$. The performance index to be minimized will consist of a weighted sum of the squares of the forward and backward prediction errors. The derivation to follow is similar to that presented in [37]. Therefore,

$$\begin{aligned} \mathcal{E}_m^{LS} &= \sum_{l=0}^{n} \lambda^{n-l} \left[|f_m(n)|^2 + |b_m(n)|^2 \right] \\ &= \sum_{l=0}^{n} \lambda^{n-l} \left[|f_{m-1}(n) - k_m(n-1) b_{m-1}(n-1)|^2 + \right. \\ &\qquad \left. + \ |b_{m-1}(n-1) - k_m(n-1) f_{m-1}(n)|^2 \right] \ . \end{aligned} \tag{5.15}$$

Minimization of $\mathcal{E}_m^{LS}$ with respect to $k_m(n)$ yields the solution

$$k_m(n) = \frac{-2\sum_{l=0}^{n} \lambda^{n-l} f_{m-1}(n) b_{m-1}(n-1)}{\sum_{l=0}^{n} \lambda^{n-l} \left[|f_m(n)|^2 + |b_m(n)|^2 \right]} \ . \tag{5.16}$$

Equation 5.16 can be computed recursively, where the numerator and denominator may be updated as follows:

$$u_m(n) = \lambda u_m(n-1) + 2 f_{m-1}(n) b_{m-1}(n-1) \tag{5.17}$$

$$v_m(n) = \lambda v_m(n-1) + |f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2 \ . \tag{5.18}$$

Then

$$k_m(n) = \frac{-u_m(n)}{v_m(n)} \ . \tag{5.19}$$

Accordingly, $k_m(n)$ may be updated recursively in time according to the relation

$$k_m(n) = k_m(n-1) + \frac{f_{m-1}(n) b_m(n) + b_{m-1}(n-1) f_m(n)}{v_m(n)} \ . \tag{5.20}$$

It should be noted that this particular lattice structure is referred to as a *gradient lattice-ladder*, because the traditional implementation of the lattice uses a gradient algorithm to update the tap weight coefficients. Although this is no longer the case for the decorrelation criterion, in order to distinguish this structure from the RLS lattice, this notational description will be continued.

The algorithm is now complete. At time n, the backward and forward prediction errors are updated according to

$$f_0(n) = b_0(n) = \hat{y}(n)$$

and for stages $m = 1, 2, \ldots, M-1$ the order-update recursions for the gradient lattice-ladder are performed as follows. Update the forward and backward prediction errors according to

$$f_m(n) = f_{m-1}(n) - k_m(n-1)b_{m-1}(n-1) \tag{5.21}$$

$$b_m(n) = b_{m-1}(n-1) - k_m(n-1)f_{m-1}(n) \quad . \tag{5.22}$$

Update the adaptive step size as

$$v_m(n) = \lambda v_m(n-1) + |f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2 \quad . \tag{5.23}$$

The reflection coefficients are updated according to

$$k_m(n) = k_m(n-1) + \frac{f_{m-1}(n)b_m(n) + b_{m-1}(n-1)f_m(n)}{v_m(n)} \quad . \tag{5.24}$$

Form the output of the gradient lattice-ladder as

$$z(n) = \sum_{k=1}^{N-1} w_k(n)b_k(n) \quad . \tag{5.25}$$

Finally, update the tap weight coefficients in the following manner using the decorrelation criterion:

$$w_m(n+1) = w_m(n) + \mu y(n)y(n-m) \quad . \tag{5.26}$$

The components can be initialized as follows:

$$\begin{aligned} f_m(-1) &= b_m(-1) = 0 \\ v_m(-1) &= k_m(-1) = 0 \\ w_m(0) &= 0 \\ z(0) &= 0 \ . \end{aligned}$$

5.3 Simulation Results for the Lattice-Ladder Algorithms

For the purpose of examining the convergence of the RLS lattice-ladder and gradient lattice-ladder algorithms with decorrelation, the blind, decorrelation lattice in the configuration of Figure (5.3) was simulated for a channel which introduces severe intersymbol interference. In the simulation, four-stage lattices were used. The step size for the tap-weight updates was chosen to be μ=0.05 and the weighting factor was chosen as λ=0.99. The particular channel sampled impulse response used in all of the simulations is the same as used by Proakis in [35] and is given by

$$H(z) = 0.407 + 0.815z^{-1} + 0.407z^{-2} \ . \tag{5.27}$$

The correlation matrix, $\mathbf{R}$, of the channel of Equation 5.27 has an eigenvalue spread of

$$\chi(\mathbf{R}) = \frac{\lambda_{max}}{\lambda_{min}} = 436.6379 \ . \tag{5.28}$$

This can be effectively considered an infinite eigenvalue spread. The channel's impulse response is normalized such that $h_0^2+h_1^2+h_2^2 = 1$. Zero-mean, white, Gaussian noise with variance 0.001 was added to the output of the channel. As a means to compare the relative speeds of the two lattice algorithms, the RLC algorithm given in Chapter 4 (with M=4) and a four-tap version of the conventional decorrelation DFE of Chapter 2 were also implemented using the channel given by Equation 5.27. For the decorrelation DFE, the step size was chosen to be μ=0.05 in order to offer the best possible comparison to the lattice algorithms. These two structures will provide upper and lower bounds, respectively, on the rate of convergence of the lattice-based decorrelation DFEs.

The model of the DFE used in the simulation is similar to that presented in Figure 4.2. In the simulations to follow, the length of the delay of the element, z^{-k}, of Figure 4.2 was chosen to produce the smallest MSE for the algorithms being considered. Unless specified, this delay was chosen to be z^{-1}. Figure (5.6) shows the comparison of the rates of convergence for the various algorithms using the channel of Equation 5.27.

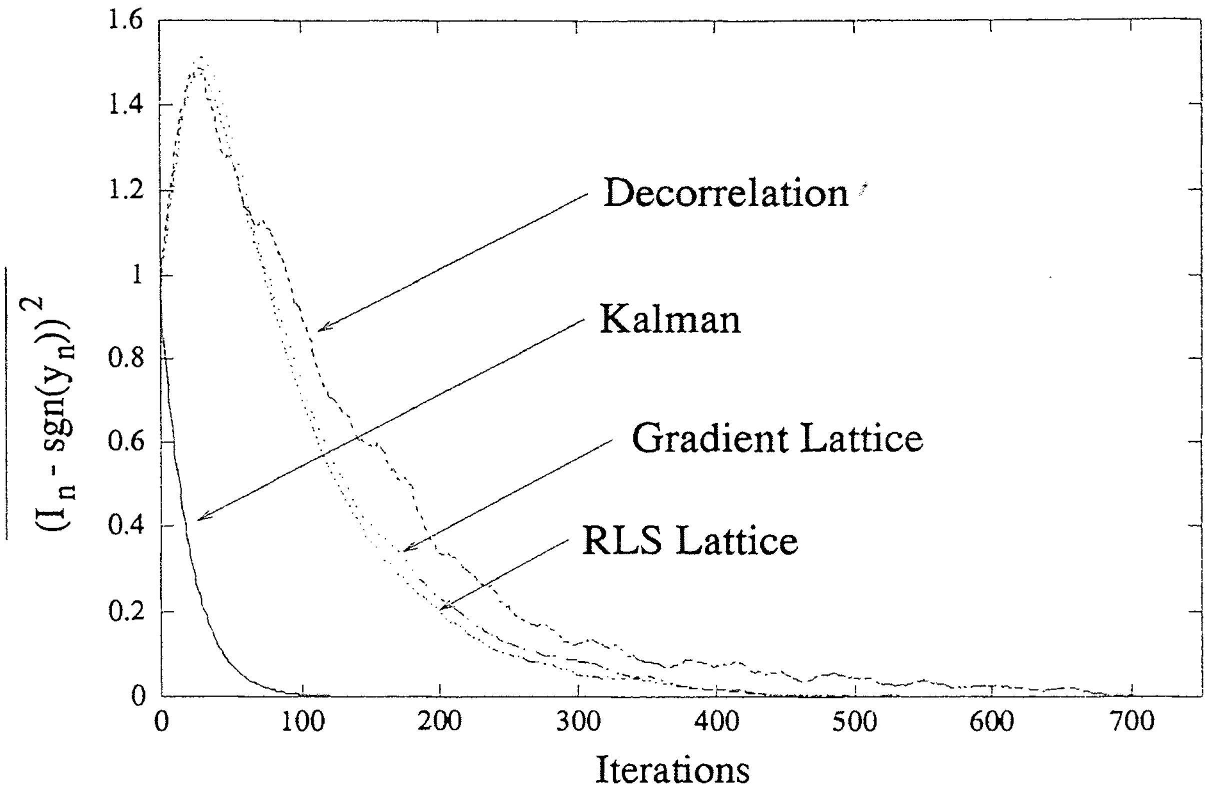

Figure 5.6 Comparison of Gradient, RLS, Decorrelation, and Kalman Equalizers.

In this figure, the estimate of the residual ISI power is obtained by passing the sequence of the squared error $(I(n)-\hat{y}(n))^2$ through a smoothing filter whose transfer function is given by $0.05/(1-0.95z^{-1})$. As mentioned in Chapter 4, this particular definition of the MSE is the same as used by Satorius and Pack [41], Satorius and Alexander [40], and Haykin [18]. Each curve was obtained by Monte-Carlo averaging the MSE over 100 independent trials.

From Figure 5.6, it can be seen that the RLS and gradient lattice algorithms converge in approximately 450 iterations, while the conventional decorrelation DFE requires almost 700 iterations. Notice, however, there is a decrease in the rate of convergence of the lattice-based structures compared with the RLC algorithm. This is a direct result of the fact that the lattice structures have not been derived based on the decorrelation criterion. Rather, only the tap weights are updated according to the decorrelation algorithm. The increase in the rate of convergence of the lattice algorithms over the decorrelation DFE is due to the self-orthogonalizing nature of the lattice algorithms.

As was mentioned earlier, the RLS lattice algorithm offers a slightly faster rate of convergence than the gradient lattice. Notice from Figure 5.6 that there is only a negligible increase in convergence of the RLS over the gradient lattice. From Figure 5.6, it can be seen that the rate of convergence of the two lattice algorithms is virtually identical. In fact, in numerous simulations with channels of varying complexity, it was found that the differences between the convergence rates of the two algorithms was almost negligible. For simple channels, the speed of convergence of the two adaptive lattice equalizers was found, in most cases, to be identical.

Again, it should be emphasized that although there is a decrease in the speed of convergence, the RLS and gradient lattice-ladder equalizers offer a greater savings in computational complexity (compared with the RLC algorithm) and exhibit better numerical stability and robustness to finite-precision errors (compared with the fast Kalman and FTF algorithms). Furthermore, since the RLS and gradient lattice structures are types of self-orthogonalizing equalizers, the convergence rate of the equalizers do not depend on the eigenvalue spread of the channel [18].

5.4 Numerical Properties of the Decorrelation-Based Equalizers

As a final note on the adaptive lattice equalizers presented in this paper, a comparison of the computational complexity of the various algorithms should be made. Table 5.1 shows a comparison of the number of operations needed to update the tap weight coefficients (in terms of the number of multiplications and divisions per iteration), based on the length of the filter.

Table 5.1 Comparison of the Computational Complexities of the Decorrelation-Based Algorithms.

ALGORITHM	MADPI
RLC	$4M^2 + 4M + 2$
RLS-Lattice	$15M - 11$
Gradient-Lattice	$13M - 8$
Fast Transversal Filter	$11M + 16$
Fast RLC	$11M + 6$
Decorrelation DFE	$2M + 1$

To facilitate comparison, the results of Table 5.1 are plotted in Figure 5.7 to offer a better comparison of the relative complexities of the various algorithms.

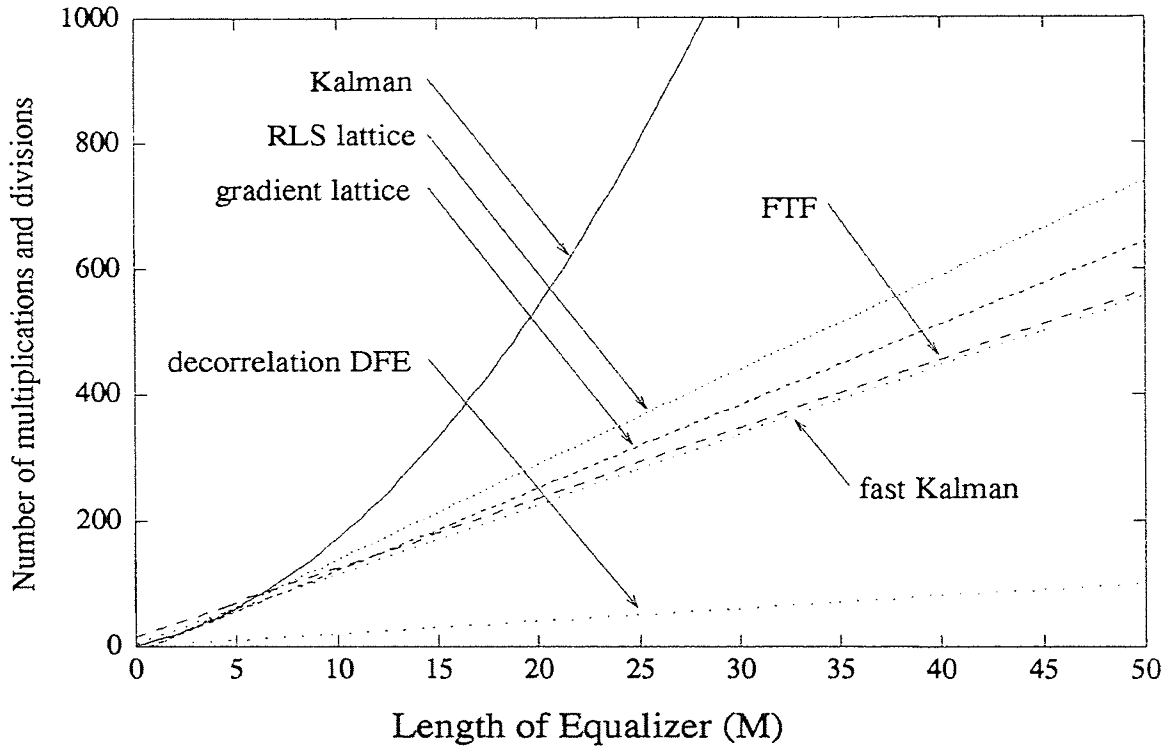

Figure 5.7 Computational Complexity of Decorrelation-Based Equalizers.

The results of Table 5.1 and the individual curves in Figure (5.7) are based on estimates of the count of the number of multiplications and divisions per iteration for the various algorithms as described in Chapter 4, [33], [41], and Chapter 2. The parabolic nature of the curve for the RLC algorithm is due to its order M^2 computational complexity. All other algorithms in the figure have order M complexity. The decorrelation algorithm presented in Chapter 2 was included as a reference on the lower bound of complexity. The decorrelation algorithm, like the LMS algorithm, can be considered as one of the simplest of the adaptive equalization algorithms in terms of implementation. Consequently, as can be clearly seen in the figure, the fast Kalman is the most efficient of the recursive least algorithms discussed in this work. However, its computational complexity is only slightly less than the fast transversal filter. Closely following is the gradient lattice algorithm, then the RLS lattice, and, finally, the direct-form Kalman algorithm. Note that for small values of M (equalizers of very short length), there is little difference in the complexity among the rapidly convergent algorithms.

CHAPTER 6

CONCLUSIONS AND AREAS FOR CONTINUED WORK

In this work, it was shown that the rate of convergence of the decorrelation-based decision feedback equalizer can be increased dramatically by a subsequent increase in the complexity of the algorithm. However, this complexity has a direct bearing on the cost of implementing the algorithm in either hardware or software. It is this cost that provides the motivation for the development of more computationally efficient algorithms which maintain the high rate of convergence, but at a complexity that increases linearly with the length of the tap weight vector. Several such algorithms have been proposed and developed in this thesis which alleviate the $\mathcal{O}(M^2)$ complexity of the RLC algorithm of [24].

The fast Kalman algorithm for decorrelation and the fast transversal filter for decorrelation both have $\mathcal{O}(M)$ complexity. Furthermore, since the fast Kalman and FTF algorithms based on decorrelation are mathematically equivalent to the RLC algorithm, these two fast algorithms should have a rate of convergence which is comparable to that of the RLC. This was confirmed through simulation and comparison of the performance of the various algorithms in differing channels. However, a disadvantage of these fast algorithms is that they have a tendency to become unstable in finite-precision environments. Remedies to this problem were proposed, among which was the derivation of the covariance fast Kalman algorithm for decorrelation. The CFRLC is the unwindowed case of the fast Kalman algorithm, used when the fast Kalman algorithm is restarted to overcome the accumulated effects of the finite-precision environment.

Two additional fast structures, which exhibit better numerical stability in finite-precision environments, were also proposed to deal with the complexity of the RLC. The RLS and gradient lattices have $\mathcal{O}(M)$ complexity and have been shown to be more inherently stable in environments governed by a finite word-length. To incorporate the decorrelation algorithm into these algorithms, the existing lattice structures were used, with the tap weights now updated according to the decorrelation algorithm. In addition, since these lattice structures were implemented in a decision feedback configuration, the decision on the most recent output of the equalizer is used as the input to the various lattice structures. This differs from the conventional approach taken in DFE lattice implementations, which use a two-channel lattice and an error feedback structure. Again, through simulation with various channels, the rate of convergence of both the RLS and gradient lattices based on decorrelation was shown to offer a substantial increase in speed over the decorrelation DFE. A comparison of the relative computational complexities of the various algorithms (given in terms of the number of multiplications and divisions per iteration) was also performed. In terms of linear complexity of the rapidly converging structures presented in this work, the fast Kalman and FTF algorithms are the least computationally complex, followed by the gradient and RLS lattices.

As was discussed, the performance of the algorithms proposed in this thesis depends on the type of environment in which they are implemented. In other words, the convergence properties of a given algorithm may be dramatically different when that algorithm is implemented in a finite-precision environment. It is for this reason that a discussion on the finite-precision performance of the decorrelation algorithm has been made. Models for the finite word-length environment were proposed and an expression for the quantization error of the correction term of the decorrelation algorithm, referred to as the digital residual error, was calculated and shown to be similar to the corresponding LMS. These results were compared with those of the LMS algorithm, on which much work has been done in the area of limited-precision environments. It was shown that the decorrelation algorithm offers comparable finite-precision performance to the LMS.

The work proposed in this thesis may progress into many areas. The most obvious extension of this work would be studying the performance of the fast algorithms for decorrelation in finite-precision environments. Methods similar to that proposed in [18] and [30] could be used. Furthermore, to truly understand the performance of the decorrelation algorithm in a finite-precision environment, more complex models of the decorrelation quantization error will need to be derived. In the context of finite-precision, the fast algorithms could also be modified into their corresponding normalized or stabilizing forms, as proposed in [8], [3], and [43]. Recall that this would entail a relative increase in the complexities of the fast algorithms due to the inclusion of square-root computations. Moreover, the decorrelation algorithm, itself, could be modified so as to offer better performance in finite-precision environments. The application of the technique known as *leakage* to the decorrelation algorithm would help to stabilize the digital implementation of the algorithm in that occurence of overflow would be prevented [6]. Consequently, the scaling factor derived in Chapter 2 would no longer be needed.

Next, with the channel model used for this thesis, there was subsequently no need for a forward filter in the decision feedback equalizer. Consequently, only single channel forms of the fast algorithms were derived. Therefore, another area into which this work may progress is the creation of multichannel representations of the FRLC and FTF algorithms. This extension is quite straightforward and could easily follow the derivations given in this work, since all that is required is a new declaration of the augmented vectors of the input and output of the slicer.

The decorrelation algorithm can also be extended to many other fast structures. There are literally dozens of competing fast algorithms that have been proposed in the literature, in both fast Kalman/FTF and lattice manifestations (see [38] for an extensive bibliographical listing of fast algorithms). Since most of these other fast algorithms are based on much of the same theory that has been proposed in this thesis, the further application of the decorrelation algorithm to other computational efficient structures is very possible.

Finally, if better numerical stability and rapid initial convergence of the decorrelation algorithm is desired without regard to complexity, a decorrelation-based structure using square-root Kalman filtering can be developed. This class of algorithms, like the RLS, has an $\mathcal{O}(M^2)$ complexity, but has been shown to be the most numerically stable of all rapidly converging algorithms [19].

APPENDIX A

PROOF OF EQUATIONS 4.46 AND 4.47

It will now be shown that Equations 4.33 and 4.34 are contained in the expressions of Equations 4.46 and 4.47. To prove Equation 4.46, substitute the first matrix representation of Equation 4.38 for $\bar{\mathbf{R}}(n)$ in Equation 4.33 and multiply out all terms, as shown:

$$\begin{aligned}
&\begin{bmatrix} \pi(n) & \mathbf{S}'_{\mathrm{M}}(n) \\ \mathbf{Q}_{\mathrm{M}}(n) & \mathbf{R}_{\mathrm{M,M}}(n-1) \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{K}_{\mathrm{M}}(n-1) \end{bmatrix} = \\
&= \begin{bmatrix} \mathbf{S}_{\mathrm{M}}(n)\mathbf{K}_{\mathrm{M}}(n-1) \\ \mathbf{R}_{\mathrm{M.M}}(n-1)\mathbf{K}_{\mathrm{M}}(n-1) \end{bmatrix} \\
&= \begin{bmatrix} \rho(n) \\ \mathbf{Y}_{\mathrm{M}}(n-1) \end{bmatrix} \\
&= \begin{bmatrix} y(n-1) \\ \mathbf{Y}_{\mathrm{M}}(n-1) \end{bmatrix} + \begin{bmatrix} \mathbf{S}_{\mathrm{M}}(n)\mathbf{K}_{\mathrm{M}}(n-1) - y(n-1) \\ \mathbf{0}_{\mathrm{M}} \end{bmatrix} \\
&= \bar{\mathbf{Y}}_{\mathrm{M}}(n) + \begin{bmatrix} \rho(n) - y(n-1) \\ \mathbf{0}_{\mathrm{M}} \end{bmatrix}.
\end{aligned}$$

To prove Equation 4.47, substitute the second matrix representation of Equation 4.38 for $\bar{\mathbf{R}}(n)$ in Equation 4.34 and multiply out all terms, as shown:

$$\begin{aligned}
&\begin{bmatrix} \mathbf{R}_{\mathrm{M,M}}(n) & \check{\mathbf{Q}}_{\mathrm{M}}(n) \\ \check{\mathbf{S}}'_{\mathrm{M}}(n) & \check{\pi}(n) \end{bmatrix} \begin{bmatrix} \mathbf{K}_{\mathrm{M}}(n) \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\mathrm{M,M}}(n)\mathbf{K}_{\mathrm{M}}(n) \\ \check{\mathbf{S}}'_{\mathrm{M}}(n)\mathbf{K}_{\mathrm{M}}(n) \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{Y}_{\mathrm{M}}(n) \\ \beta(n) \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{Y}_{\mathrm{M}}(n) \\ y(n-M-1) \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{\mathrm{M}} \\ \check{\mathbf{S}}'_{\mathrm{M}}(n)\mathbf{K}_{\mathrm{M}}(n) - y(n-M-1) \end{bmatrix} \\
&= \bar{\mathbf{Y}}_{\mathrm{M}}(n) + \begin{bmatrix} \mathbf{0}_{\mathrm{M}} \\ \beta(n) - y(n-M-1) \end{bmatrix}.
\end{aligned}$$

APPENDIX B

PROOF OF EQUATION 4.51

In order to prove Equation 4.51, it will first be necessary to substitute the first matrix representation of Equation 4.38 for $\bar{\mathbf{R}}(n)$ in Equation 4.50. Thus,

$$\begin{aligned}
\bar{\mathbf{R}}(n)\begin{bmatrix} 1 \\ \mathbf{F}_{\mathrm{M}}(n) \end{bmatrix} &= \begin{bmatrix} \pi(n) & \mathbf{S}'_{\mathrm{M}}(n) \\ \mathbf{Q}_{\mathrm{M}}(n) & \mathbf{R}_{\mathrm{M,M}}(n-1) \end{bmatrix}\begin{bmatrix} 1 \\ \mathbf{F}_{\mathrm{M}}(n) \end{bmatrix} \\
&= \begin{bmatrix} \pi(n) + \mathbf{S}'_{\mathrm{M}}(n)\mathbf{F}_{\mathrm{M}}(n) \\ \mathbf{Q}_{\mathrm{M}}(n) + \mathbf{R}_{\mathrm{M,M}}(n-1)\mathbf{F}_{\mathrm{M}}(n) \end{bmatrix} \\
&= \begin{bmatrix} \mathcal{F}_M(n) \\ \mathbf{0}_{\mathrm{M}} \end{bmatrix} .
\end{aligned}$$

Consequently, according to the above equations, the following two equalities hold:

$$\mathcal{F}_M^{-1}(n) = \frac{1}{\pi(n) + \mathbf{S}'_{\mathrm{M}}(n)\mathbf{F}_{\mathrm{M}}(n)} \tag{B.1}$$

and

$$\mathbf{Q}_{\mathrm{M}}(n) + \mathbf{R}_{\mathrm{M,M}}(n-1)\mathbf{F}_{\mathrm{M}}(n) = \mathbf{0}_{\mathrm{M}} \ . \tag{B.2}$$

Therefore, taking these two equalities into account, postmultiply both sides of Equation 4.50 by $\mathcal{F}_M^{-1}(n)\,(\rho(n) - y(n-1))$. Expanding terms yields:

$$\begin{aligned}
&\begin{bmatrix} \pi(n) + \mathbf{S}'_{\mathrm{M}}(n)\mathbf{F}_{\mathrm{M}}(n) \\ \mathbf{Q}_{\mathrm{M}}(n) + \mathbf{R}_{\mathrm{M,M}}(n-1)\mathbf{F}_{\mathrm{M}}(n) \end{bmatrix} \mathcal{F}_M^{-1}(n)\,(\rho(n) - y(n-1)) = \\
&= \begin{bmatrix} \{\pi(n) + \mathbf{S}'_{\mathrm{M}}(n)\mathbf{F}_{\mathrm{M}}(n)\}\left\{\mathcal{F}_M^{-1}(n)\,(\rho(n) - y(n-1))\right\} \\ \{\mathbf{Q}_{\mathrm{M}}(n) + \mathbf{R}_{\mathrm{M,M}}(n-1)\mathbf{F}_{\mathrm{M}}(n)\}\left\{\mathcal{F}_M^{-1}(n)\,(\rho(n) - y(n-1))\right\} \end{bmatrix} \\
&= \begin{bmatrix} \rho(n) - y(n-1) \\ \mathbf{0}_{\mathrm{M}} \end{bmatrix} .
\end{aligned}$$

APPENDIX C

PROOF OF EQUATION 4.53

In order to prove Equation 4.53, it will first be necessary to substitute the first matrix representation of Equation 4.38 for $\bar{\mathbf{R}}(n)$ in Equation 4.53 and use the definition of $\bar{\mathbf{K}}(n)$ given in Equation 4.52. Thus,

$$\begin{aligned}
&\bar{\mathbf{R}}(n)\bar{\mathbf{K}}(n) = \\
&= \begin{bmatrix} \pi(n) & \mathbf{S}'_{\mathrm{M}}(n) \\ \mathbf{Q}_{\mathrm{M}}(n) & \mathbf{R}_{\mathrm{M,M}}(n-1) \end{bmatrix} \begin{bmatrix} -\mathcal{F}_M^{-1}(n)\,[\rho(n) - y(n-1)] \\ \mathbf{K}_{\mathrm{M}}(n-1) - \mathbf{F}_{\mathrm{M}}(n)\mathcal{F}_M^{-1}(n)\,[\rho(n) - y(n-1)] \end{bmatrix} \\
&= \begin{bmatrix} -\left\{\{\pi(n) + \mathbf{S}'_{\mathrm{M}}(n)\mathbf{F}_{\mathrm{M}}(n)\}\,\mathcal{F}_M^{-1}(n)\,(\rho(n) - y(n-1))\right\} + \mathbf{S}_{\mathrm{M}}(n)\mathbf{K}_{\mathrm{M}}(n-1) \\ -\left\{\{\mathbf{Q}(n) + \mathbf{R}(n-1)\mathbf{F}(n)\}\,\mathcal{F}_M^{-1}(n)\,(\rho(n) - y(n-1))\right\} + \mathbf{R}(n-1)\mathbf{K}(n-1) \end{bmatrix} .
\end{aligned}$$

Using Equations 4.33 and 4.35, the identities derived in Appendix B for $\mathcal{F}_M^{-1}(n)$ and $\mathbf{Q}_{\mathrm{M}}(n) + \mathbf{R}_{\mathrm{M.M}}(n-1)\mathbf{F}_{\mathrm{M}}(n)$ and the definition of $\rho(n)$ in Equation 4.48, the above equation can be simplified as follows:

$$\begin{aligned}
\bar{\mathbf{R}}(n)\bar{\mathbf{K}}(n) &= \begin{bmatrix} -(\rho(n) - y(n-1)) + \rho(n) \\ \mathbf{0}_{\mathrm{M}} + \mathbf{R}_{\mathrm{M.M}}(n-1)\mathbf{K}_{\mathrm{M}}(n-1) \end{bmatrix} \\
&= \begin{bmatrix} y(n-1) \\ \mathbf{R}_{\mathrm{M.M}}(n-1)\mathbf{K}_{\mathrm{M}}(n-1) \end{bmatrix} = \begin{bmatrix} y(n-1) \\ \mathbf{Y}_{\mathrm{M}}(n-1) \end{bmatrix} = \bar{\mathbf{Y}}_{\mathrm{M}}(n) \ .
\end{aligned}$$

APPENDIX D

THE COVARIANCE FAST KALMAN ALGORITHM FOR DECORRELATION

In this appendix, a modification to the fast Kalman algorithm for decorrelation will be made so that the FRLC can accomodate the case of unwindowed data, as discussed in Chapter 4. The covariance fast Kalman algorithm for decorrelation (CFRLC) assumes that the input data vectors $\mathbf{Y}_{\mathrm{M}}(n)$ and $\hat{\mathbf{Y}}_{\mathrm{M}}(n)$ are not zero for $n < 0$. The proof of the algorithm closely parallels that presented in Chapter 4 for the prewindowed fast Kalman. Therefore, similar steps in the proof will be omitted for sake of brevity and only relevant and necessary results presented. The derivation is based on the work done in [25].

For the covariance fast Kalman, like the fast Kalman, it will be necessary to derive a set of equations to compute the covariance Kalman gain vector, $\mathbf{\Omega}_{\mathrm{M}}(n) = \mathbf{R}^{-1}_{\mathrm{M,M}}(n)\mathbf{Y}_{\mathrm{M}}(n)$ recursively in time. The modified cross-correlation matrix, $\mathbf{R}_{\mathrm{M,M}}(n)$ is given by

$$\mathbf{R}_{\mathrm{M,M}}(n) = \sum_{k=0}^{n} \lambda^{n-k}\mathbf{Y}_{\mathrm{M}}(k)\hat{\mathbf{Y}}'_{\mathrm{M}}(k) + \delta \mathcal{W}_{\mathrm{M}}(n) \tag{D.1}$$

where

$$\mathcal{W}_{\mathrm{M}}(n) = \mathrm{diag}\left[\lambda^{n}, \lambda^{n-1}, \ldots, \lambda^{n-M+1}\right] \tag{D.2}$$

and δ is a small, positive constant. The constant λ is chosen close to, but less than, one. This inclusion of the weighting factor in Equation D.1 is to insure the initial nonsingularity of the cross-correlation matrix, $\mathbf{R}_{\mathrm{M,M}}(n)$ [18]. It has been shown in the literature [25] that the modification will not affect the time-update recursions for the fast Kalman algorithm, only the starting value. Furthermore, the additional weighting term will decay to zero as $n \to \infty$. To go from $\mathbf{\Omega}_{\mathrm{M}}(n-1)$ to $\mathbf{\Omega}_{\mathrm{M}}(n)$, it is possible to write

$$\mathbf{R}_{\mathrm{M,M}}(n-1)\boldsymbol{\Omega}_{\mathrm{M}}(n-1) = \mathbf{Y}_{\mathrm{M}}(n-1) \tag{D.3}$$

and

$$\mathbf{R}_{\mathrm{M,M}}(n)\boldsymbol{\Omega}_{\mathrm{M}}(n) = \mathbf{Y}_{\mathrm{M}}(n) \ . \tag{D.4}$$

The matrices $\mathbf{R}_{\mathrm{M,M}}(n-1)$ and $\mathbf{R}_{\mathrm{M,M}}(n)$ can be related through the augmented correlation matrix (see [25])

$$\bar{\mathbf{R}}(n) = \mathbf{R}_{\mathrm{M+1,M+1}}(n) = \sum_{k=1}^{n} \lambda^{n-k}\bar{\mathbf{Y}}(k)\bar{\hat{\mathbf{Y}}}'(k) + \delta\overline{\mathcal{W}}_{\mathrm{M}}(n) \tag{D.5}$$

where $\bar{\mathbf{Y}}(n)$ and $\bar{\hat{\mathbf{Y}}}'(n)$ are given by Equations 4.35 and 4.36, respectively, and

$$\delta\overline{\mathcal{W}}_{\mathrm{M}}(n) = \mathrm{diag}\left[\lambda^{n}, \lambda^{n-1}, \ldots, \lambda^{n-M}\right] \tag{D.6}$$

as follows

$$\begin{aligned} \bar{\mathbf{R}}(n) &= \begin{bmatrix} \breve{\pi}(n) & \breve{\mathbf{S}}'_{\mathrm{M}}(n) \\ \breve{\mathbf{Q}}_{\mathrm{M}}(n) & \mathbf{R}_{\mathrm{M,M}}(n-1) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_{\mathrm{M,M}}(n) - \lambda^{n}\mathbf{Y}_{\mathrm{M}}(0)\hat{\mathbf{Y}}'_{\mathrm{M}}(0) & \tilde{\mathbf{Q}}_{\mathrm{M}}(n) \\ \tilde{\mathbf{S}}'_{\mathrm{M}}(n) & \tilde{\pi}(n) \end{bmatrix} \ . \end{aligned} \tag{D.7}$$

The quantities which comprise Equation D.7 are defined according to

$$\begin{aligned} \breve{\pi}(n) &= \sum_{k=1}^{n} \lambda^{n-k} y(k-1)\hat{y}(k-1) + \lambda^{n}\delta \\ \breve{\mathbf{S}}_{\mathrm{M}}(n) &= \sum_{k=1}^{n} \lambda^{n-k}\hat{\mathbf{Y}}_{\mathrm{M}}(k-1)y(k-1) \\ \breve{\mathbf{Q}}_{\mathrm{M}}(n) &= \sum_{k=1}^{n} \lambda^{n-k}\mathbf{Y}_{\mathrm{M}}(k-1)\hat{y}(k-1) \\ \mathbf{R}_{\mathrm{M,M}}(n-1) &= \sum_{k=0}^{n} \lambda^{n-k}\mathbf{Y}_{\mathrm{M}}(k-1)\hat{\mathbf{Y}}'_{\mathrm{M}}(k-1) + \delta\mathcal{W}_{\mathrm{M}}(n-1) \end{aligned}$$

and

$$\tilde{\pi}(n) = \sum_{k=1}^{n} \lambda^{n-k} y(k-M)\hat{y}(k-M) + \lambda^{n-M}\delta \tag{D.8}$$

$$\tilde{\mathbf{S}}_{\mathrm{M}}(n) = \sum_{k=1}^{n} \lambda^{n-k}\hat{\mathbf{Y}}_{\mathrm{M}}(k)y(k-M) \tag{D.9}$$

$$\tilde{\mathbf{Q}}_{\mathrm{M}}(n) = \sum_{k=1}^{n} \lambda^{n-k} \mathbf{Y}_{\mathrm{M}}(k)\hat{y}(k-M) \tag{D.10}$$

and $\mathbf{R}_{\mathrm{M,M}}(n)$ is given by Equation D.1. Referring to Equation 4.54, the augmented Kalman gain vector, $\bar{\mathbf{K}}(n)$, can be partitioned as

$$\bar{\mathbf{K}}(n) = \begin{bmatrix} \mathbf{C}_{\mathrm{M}}(n) \\ c_M(n) \end{bmatrix} . \tag{D.11}$$

Substituting Equation D.11, the second expression for $\bar{\mathbf{R}}(n)$ in Equation D.7, and the second expression for $\bar{\mathbf{Y}}(n)$ in Equation 4.35 into Equation 4.49,

$$\begin{bmatrix} \mathbf{R}_{\mathrm{M,M}}(n) - \lambda^n \mathbf{Y}_{\mathrm{M}}(0)\hat{\mathbf{Y}}'_{\mathrm{M}}(0) & \tilde{\mathbf{Q}}_{\mathrm{M}}(n) \\ \tilde{\mathbf{S}}'_{\mathrm{M}}(n) & \tilde{\pi}(n) \end{bmatrix} \bar{\mathbf{K}}(n) = \begin{bmatrix} \mathbf{C}_{\mathrm{M}}(n) \\ c_M(n) \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{\mathrm{M}}(n) \\ y(n-M-1) \end{bmatrix}$$

it can be seen from the first line of the above equation that

$$\left[\mathbf{R}_{\mathrm{M,M}}(n) - \lambda^n \mathbf{Y}_{\mathrm{M}}(0)\hat{\mathbf{Y}}'_{\mathrm{M}}(0)\right] \mathbf{C}_{\mathrm{M}}(n) + \tilde{\mathbf{Q}}_{\mathrm{M}}(n) c_M(n) = \mathbf{Y}_{\mathrm{M}}(n) \ . \tag{D.12}$$

Recall from Chapter 4 that following expression for the backward prediction coefficients holds

$$\bar{\mathbf{R}}(n) \begin{bmatrix} \mathbf{B}_{\mathrm{M}}(n) \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{\mathrm{M}} \\ \mathcal{B}_M(n) \end{bmatrix} . \tag{D.13}$$

Substitution of the second expression for the augmented cross-correlation matrix of Equation D.7 into the above equation results in

$$\begin{bmatrix} \mathbf{R}_{\mathrm{M,M}}(n) - \lambda^n \mathbf{Y}_{\mathrm{M}}(0)\hat{\mathbf{Y}}'_{\mathrm{M}}(0) & \tilde{\mathbf{Q}}_{\mathrm{M}}(n) \\ \tilde{\mathbf{S}}'_{\mathrm{M}}(n) & \tilde{\pi}(n) \end{bmatrix} \begin{bmatrix} \mathbf{B}_{\mathrm{M}}(n) \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{\mathrm{M}} \\ \mathcal{B}_M(n) \end{bmatrix} . \tag{D.14}$$

Therefore, expanding Equation D.14 and collecting terms, it can be seen that

$$\tilde{\mathbf{Q}}_{\mathrm{M}}(n) = -\left[\mathbf{R}_{\mathrm{M,M}}(n) - \lambda^n \mathbf{Y}_{\mathrm{M}}(0)\hat{\mathbf{Y}}'_{\mathrm{M}}(0)\right] \mathbf{B}_{\mathrm{M}}(n) \ . \tag{D.15}$$

With a workable definition for $\tilde{\mathbf{Q}}_{\mathrm{M}}(n)$ now established, Equation D.12 can be rewritten according to

$$\begin{aligned} & \left[\mathbf{R}_{\mathrm{M,M}}(n) - \lambda^n \mathbf{Y}_{\mathrm{M}}(0)\hat{\mathbf{Y}}'_{\mathrm{M}}(0)\right] \left[\mathbf{C}_{\mathrm{M}}(n) - \mathbf{B}_{\mathrm{M}}(n) c_M(n)\right] = \\ = \ & \mathbf{R}_{\mathrm{M,M}}(n) \left[\mathbf{I}_{\mathrm{M}} - \lambda^n \mathbf{R}^{-1}_{\mathrm{M,M}}(n) \mathbf{Y}_{\mathrm{M}}(0)\hat{\mathbf{Y}}'_{\mathrm{M}}(0)\right] \left[\mathbf{C}_{\mathrm{M}}(n) - \mathbf{B}_{\mathrm{M}}(n) c_M(n)\right] \\ = \ & \mathbf{Y}_{\mathrm{M}}(n) \end{aligned} \tag{D.16}$$

where $\mathbf{I}_\mathrm{M}$ is the identity matrix. Let

$$\mathbf{d}_\mathrm{M}(n) = \lambda^n \mathbf{R}^{-1}_{\mathrm{M,M}}(n)\mathbf{Y}_\mathrm{M}(0) \ . \tag{D.17}$$

Note that, since $\lambda < 1$, as $n \to \infty$, $\mathbf{d}_\mathrm{M}(n) \to 0$. Therefore, comparing Equation D.16 with Equation D.4, it can readily be seen that the covariance Kalman gain vector can be defined as

$$\mathbf{\Omega}_\mathrm{M}(n) = \left[\mathbf{I}_\mathrm{M} - \mathbf{d}_\mathrm{M}(n)\hat{\mathbf{Y}}'_\mathrm{M}(0)\right]\left[\mathbf{C}_\mathrm{M}(n) - \mathbf{B}_\mathrm{M}(n)c_M(n)\right] \ . \tag{D.18}$$

Inspection of Equation D.18 reveals that it will be necessary to derive a time-update recursion for $\mathbf{d}_\mathrm{M}(n)$. Therefore, rewrite Equation D.17 as

$$\lambda^n \mathbf{Y}_\mathrm{M}(0) = \mathbf{R}_{\mathrm{M,M}}(n)\mathbf{d}_\mathrm{M}(n) \ . \tag{D.19}$$

Equation D.19, at time $n-1$, correponds to

$$\lambda^{n-1}\mathbf{Y}_\mathrm{M}(0) = \mathbf{R}_{\mathrm{M,M}}(n-1)\mathbf{d}_\mathrm{M}(n-1) \ .$$

Equivalently,

$$\lambda^n \mathbf{Y}_\mathrm{M}(0) = \lambda\mathbf{R}_{\mathrm{M,M}}(n-1)\mathbf{d}_\mathrm{M}(n-1) \ . \tag{D.20}$$

Substitution of the time-update recursion for $\mathbf{R}_{\mathrm{M,M}}(n)$ of Equation 4.8 into the last line of the above equation yields

$$\mathbf{R}_{\mathrm{M,M}}(n)\mathbf{d}_\mathrm{M}(n) = \left[\mathbf{R}_{\mathrm{M,M}}(n) - \mathbf{Y}_\mathrm{M}(n)\hat{\mathbf{Y}}'_\mathrm{M}(n)\right]\mathbf{d}_\mathrm{M}(n-1) \ . \tag{D.21}$$

Rearranging terms in Equation D.21 and using the definition of $\mathbf{\Omega}_\mathrm{M}(n)$ in Equation D.4, the time-update recursion for $\mathbf{d}_\mathrm{M}(n)$ is

$$\begin{aligned}\mathbf{d}_\mathrm{M}(n) &= \left[\mathbf{I}_\mathrm{M} - \mathbf{R}^{-1}_{\mathrm{M,M}}(n)\mathbf{Y}_\mathrm{M}(n)\hat{\mathbf{Y}}'_\mathrm{M}(n)\right]\mathbf{d}_\mathrm{M}(n-1)\\ &= \left[\mathbf{I}_\mathrm{M} - \mathbf{\Omega}_\mathrm{M}(n)\hat{\mathbf{Y}}'_\mathrm{M}(n)\right]\mathbf{d}_\mathrm{M}(n-1) \ . \end{aligned} \tag{D.22}$$

Notice in Equation D.22 that the current estimate of $\mathbf{d}_\mathrm{M}(n)$ requires the current estimate of $\mathbf{\Omega}_\mathrm{M}(n)$. But from Equation D.18, $\mathbf{\Omega}_\mathrm{M}(n)$ depends on $\mathbf{d}_\mathrm{M}(n)$. To remedy this problem, use the definition of the fast Kalman gain vector of Equation 4.93, $\mathbf{K_M}(n)$, and substitute Equation D.18 into the last line of Equation D.22:

$$\begin{aligned}\mathbf{d_M}(n) &= \left[\mathbf{I_M} - \left[\mathbf{I_M} - \mathbf{d_M}(n)\hat{\mathbf{Y}}'_\mathbf{M}(0)\right]\mathbf{K_M}(n)\hat{\mathbf{Y}}'_\mathbf{M}(n)\right]\mathbf{d_M}(n-1)\\ &= \left[\mathbf{I_M} - \mathbf{K_M}(n)\hat{\mathbf{Y}}'_\mathbf{M}(n) + \mathbf{d_M}(n)\hat{\mathbf{Y}}'_\mathbf{M}(0)\mathbf{K_M}(n)\hat{\mathbf{Y}}'_\mathbf{M}(n)\right]\mathbf{d_M}(n-1)\ .\end{aligned}$$

Collecting terms in the last line of the above equation, the time-update recursion for $\mathbf{d_M}(n)$ can be expressed as

$$\mathbf{d_M}(n) = \frac{\left[\mathbf{I_M} - \mathbf{K_M}(n)\hat{\mathbf{Y}}'_\mathbf{M}(n)\right]\mathbf{d_M}(n-1)}{1 - \hat{\mathbf{Y}}'_\mathbf{M}(0)\mathbf{K_M}(n)\hat{\mathbf{Y}}'_\mathbf{M}(n)\mathbf{d_M}(n-1)}\ . \tag{D.23}$$

The algorithm is now complete. To perform the covariance fast Kalman algorithm for decorrelation, follow Equations 4.84 through 4.92. Then, let

$$\mathbf{K_M}(n) = \mathbf{C_M}(n) - \mathbf{B_M}(n)c_M(n) \tag{D.24}$$

$$\mathbf{d_M}(n) = \frac{1}{1 - \hat{\mathbf{Y}}'_\mathbf{M}(0)\mathbf{K_M}(n)\hat{\mathbf{Y}}'_\mathbf{M}(n)\mathbf{d_M}(n-1)}\left[\mathbf{I_M} - \mathbf{K_M}(n)\hat{\mathbf{Y}}'_\mathbf{M}(n)\right]\mathbf{d_M}(n-1) \tag{D.25}$$

and

$$\mathbf{\Omega_M}(n) = \left[\mathbf{I_M} - \mathbf{d_M}(n)\hat{\mathbf{Y}}'_\mathbf{M}(0)\right]\mathbf{K_M}(n) \tag{D.26}$$

where $\mathbf{\Omega_M}(n)$ is the desired covariance Kalman gain vector. To initialize the algorithm, set $\mathbf{F_M}(0)=\mathbf{0_M}$, $\mathbf{B_M}(0)=\mathbf{0_M}$, $\mathcal{F}_M(0)=\lambda\delta$, and

$$\begin{aligned}\mathbf{K_M}(0) &= \mathbf{d_M}(0) = \mathbf{R}^{-1}_\mathbf{M,M}(0)\mathbf{Y_M}(0)\\ &= \frac{\mathbf{Y_M}(0)}{\mathbf{Y_M}(0)\hat{\mathbf{Y}}'_\mathbf{M}(0) + \delta\mathcal{W}_\mathbf{M}(0)}\\ &= \frac{\mathcal{W}^{-1}_\mathbf{M}(0)\mathbf{Y_M}(0)}{\delta + \hat{\mathbf{Y}}'_\mathbf{M}(0)\mathcal{W}^{-1}_\mathbf{M}(0)\mathbf{Y_M}(0)}\ .\end{aligned} \tag{D.27}$$

It should be noted that for the unwindowed, or covariance, data case, the vectors $\mathbf{Y_M}(n)$ and $\hat{\mathbf{Y}}_\mathbf{M}(n)$ at time $n = 0$ are now comprised of the previous M data samples. In other words,

$$\mathbf{Y}'_\mathbf{M}(0) = [y(-1), y(-2), \ldots, y(-M)] \tag{D.28}$$

$$\hat{\mathbf{Y}}'_\mathbf{M}(0) = [\hat{y}(-1), \hat{y}(-2), \ldots, \hat{y}(-M)]\ . \tag{D.29}$$

REFERENCES

1. R. A. Axford, Jr., "Blind equalization with the lattice constant modulus algorithm," in *Proc. IEEE Military Communications Conf.*, Boston, Massachusetts, pp. 268–272, October 1993.

2. S. Bellini, "Bussgang techniques for blind equalization," in *Proc. IEEE Global Telecommunications Conf.*, Houston, Texas, pp. 1634–1640, 1986.

3. J.-L. Botto and G. V. Moustakides, "Stabilizing the fast Kalman algorithms," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 37, no. 9, pp. 1342–1348, September 1989.

4. C. Caraiscos and B. Liu, "A roundoff error analysis of the LMS adaptive algorithm," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-32, no. 1, pp. 34–41, February 1984.

5. G. Carayannis, D. G. Manolakis, and N. Kalouptsidis, "A fast sequential algorithm for least-squares filtering and prediction," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-31, no. 6, pp. 1394–1402, December 1983.

6. J. M. Cioffi, "Limited-precision effects in adaptive filtering," *IEEE Trans. Circuits and Systems*, vol. CAS-34, no. 7, pp. 821–833, July 1987.

7. J. M. Cioffi and T. Kailath, "Fast, recursive least-squares transversal filters for adaptive filtering," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-32, no. 2, pp. 304–337, April 1984.

8. J. M. Cioffi and T. Kailath, "Windowed fast transversal filters adaptive algorithms with normalization," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-33, no. 3, pp. 607–625, June 1985.

9. E. Eleftheriou and D. D. Falconer, "Restart methods for stabilizing FRLS adaptive equalizers in digital HF transmission," in *Proc. IEEE Global Telecommunications Conf.*, Atlanta, GA, pp. 1558–1562, November 1984.

10. D. D. Falconer and L. Ljung, "Application of fast Kalman estimation to adaptive equalization," *IEEE Trans. Communications*, vol. COM-26, no. 10, pp. 1439–1446, October 1978.

11. B. Friedlander, "Lattice filters for adaptive processing," *Proc. IEEE*, vol. 70, no. 8, pp. 829–867, August 1982.

12. R. D. Gitlin and F. R. Magee, Jr., "Self-orthogonalizing adaptive equalization algorithms," *IEEE Trans. Communications*, vol. COM-25, no. 7, pp. 666–672, July 1977.

13. R. D. Gitlin, J. E. Mazo, and M. G. Taylor, "On the design of gradient algorithms for digitally implemented adaptive filters," *IEEE Trans. Communications*, vol. CT-20, no. 2, pp. 125–136, March 1973.

14. D. N. Godard, "Channel equalization using a Kalman filter for fast data transmission," *IBM J. Research and Development*, pp. 267–273, May 1974.

15. D. N. Godard, "Self-recovering equalization and carrier tracking in two-dimensional data communication systems," *IEEE Trans. Communications*, vol. COM-28, no. 11, pp. 1867–1875, November 1980.

16. D. Hatzinakos and C. L. Nikias, "Blind equalization using a tricepstrum-based algorithm," *IEEE Trans. Communications*, vol. COM-39, no. 5, pp. 669–682, May 1991.

17. S. Haykin, *Communication Systems*, John Wiley and Sons, New York, second ed., 1983.

18. S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, New Jersey, second ed., 1991.

19. F. M. Hsu, "Square-root Kalman filtering for high-speed data received over fading dispersive HF channels," *IEEE Trans. Information Theory*, vol. IT-28, no. 5, pp. 753–763, September 1982.

20. C. R. Johnson, Jr., "Admissibility in blind adaptive channel equalization," *IEEE Control Systems Mag.*, vol. 11, pp. 3–15, 1991.

21. T. Kailath, *Linear Systems*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1980.

22. N. Kalouptsidis, G. Carayannis, and D. G. Manolakis, "A fast covariance type algorithm for sequential least-squares filtering and prediction," *IEEE Trans. Automatic Control*, vol. AC-29, no. 8, pp. 752–755, August 1984.

23. R. E. Kamel and Y. Bar-Ness, "Error analysis of the blind decision feedback equalizer using the decorrelation criterion," submitted to *MILCOM 1994*.

24. R. E. Kamel, *Blind Detection in Channels with Intersymbol Interference*, PhD thesis, New Jersey Institute of Technology, Newark, New Jersey, 1994.

25. D. W. Lin, "On digital implementation of the fast Kalman algorithms," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-32, no. 5, pp. 998–1005, October 1984.

26. F. Ling, D. G. Manolakis, and J. G. Proakis, "Numerically robust least-squares lattice-ladder algorithms with direct updating of the reflection coefficients," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-34, no. 4, pp. 837–845, August 1986.

27. F. Ling and J. G. Proakis, "A generalized multichannel least squares lattice algorithm based on sequential processing stages," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-32, no. 2, pp. 381–389, April 1984.

28. F. Ling and J. G. Proakis, "Adaptive lattice decision-feedback equalizers–their performance and application to time-variant multipath channels," *IEEE Trans. Communications*, vol. COM-33, no. 4, pp. 348–356, April 1985.

29. L. Ljung, M. Morf, and D. D. Falconer, "Fast calculation of gain matrices for recursive estimation schemes," *International J. Control*, vol. 27, no. 1, pp. 1–19, January 1978.

30. S. Ljung and L. Ljung, "Error propagation properties of recursive least-squares adaption algorithms," *Automatica*, vol. 21, no. 2, pp. 157–167, March 1985.

31. R. W. Lucky, "Automatic equalization for digital communications," *The Bell System Tech. J.*, vol. 44, pp. 547–588, April 1965.

32. P. Monsen, "Feedback equalization for fading dispersive channels," *IEEE Trans. Information Theory*, vol. IT-17, pp. 56–64, January 1971.

33. M. S. Mueller, "Least-squares algorithms for adaptive equalizers," *The Bell System Tech. J.*, vol. 60, no. 8, pp. 1905–1925, October 1981.

34. A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

35. J. G. Proakis, *Digital Communications*, McGraw-Hill, New York, 1983.

36. J. G. Proakis, "Adaptive equalization for TDMA digital mobile radio," *IEEE Trans. Vehicular Technology*, vol. 40, no. 2, pp. 333–341, May 1991.

37. J. G. Proakis and D. G. Manolakis, *Introduction to Digital Signal Processing*, Macmillan Publishing Company, New York, 1988.

38. S. U. H. Qureshi, "Adaptive equalization," *Proc. IEEE*, vol. 73, no. 9, pp. 1345–1387, September 1985.

39. Y. Sato, "A method of self-recovering equalization for multilevel amplitude-modulation systems," *IEEE Trans. Communications*, vol. COM-23, pp. 679–682, June 1975.

40. E. H. Satorius and S. T. Alexander, "Channel equalization using adaptive lattice algorithms," *IEEE Trans. Communications*, vol. COM-27, no. 6, pp. 899–905, June 1979.

41. E. H. Satorius and J. D. Pack, "Application of least squares lattice algorithms to adaptive equalization," *IEEE Trans. Communications*, vol. COM-29, no. 2, pp. 136–141, February 1981.

42. M. J. Shensa, "A least-squares lattice decision feedback equalizer," in *Proc. IEEE Int. Conf. Communications*, pp. 57.6.1–57.6.5, June 1980.

43. D. T. M. Slock and T. Kailath, "Numerically stable fast transversal filters for recursive least squares adaptive filtering," *IEEE Trans. Signal Processing*, vol. 39, no. 1, pp. 92–113, January 1991.

44. B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson, Jr., "Stationary and nonstationary learning characteristics of the LMS adaptive filter," *Proc. IEEE*, vol. 64, no. 8, pp. 1151–1162, August 1976.