12-1998

Adaptive Channel Equalization Techniques

Saied Razavilar

ADAPTIVE CHANNEL EQUALIZATION TECHNIQUES

by

Saied Razavilar

A Thesis
Submitted to the
Faculty of the Graduate College
in partial fulfillment of the
Requirements for the
Degree of Master of Science in Engineering (Electrical)
Department of Electrical engineering

Western Michigan University
Kalamazoo, Michigan
December 1998

ACKNOWLEDGEMENTS

A special note of gratitude is extended to my major thesis advisor, Dr. Raghvendra Gejji, for his continuous guidance and support throughout this thesis. I also wish to thank my committee members Dr. S. Hossein Mousavinezhad and Dr. Dionysios Kountanis for their comments and recommendations and taking the time to review my work.

Saied Razavilar

ADAPTIVE CHANNEL EQUALIZATION TECHNIQUES

Saied Razavilar, M.S.E.

Western Michigan University, 1998

Inter Symbol Interference (ISI) is a characteristic of band-limited communication channels that can severely degrade the digital communication system performance if not treated properly. Channel equalization techniques may be used to mitigate the effects of ISI in the system, thereby improving the system capacity and performance. In this thesis, the problem of Inter Symbol Interference caused by the nonlinear characteristics of band limited communication channels and adaptive equalization techniques to combat the effects of ISI are discussed. First, the analysis of ISI in a band limited communication channel using a concrete theoretical framework is presented. Next, various adaptive equalization techniques of band limited channels are presented. Computer simulations are conducted for various adaptive equalization algorithms such as least mean squares (LMS), normalized least mean squares (NLMS), recursive least squares (RLS), and recursive least squares lattice (RLSL) algorithms. Based on the simulation results, conclusions and comments are made on pros and cons of various adaptive equalization techniques studied in this thesis.

TABLE OF CONTENTS

Table of Contents-Continued

CHAPTER

Table of Contents-Continued

CHAPTER

LIST OF TABLES

LIST OF FIGURES

List of Figures-Continued

CHAPTER I

INTRODUCTION

1.1 General Perspective

During the last three decades, a considerable effort has been devoted to the study of data communication systems that efficiently utilize the available channel bandwidth. Here we will consider the problem of an efficient receiver for digital information transmitted through a communication channel that is band-limited to W Hz. The optimum design of such receiver requires an exact knowledge of the communication channel statistics. In practice, channel characteristics are usually unavailable and often time varying.

Theoretically, a communication channel can be modeled as a linear filter having an equivalent low pass impulse response $c(t)$, with the corresponding frequency response $c(f)$ being zero for $|f| > W$, W being the channel bandwidth. Such band-limited channels cause symbols arriving at the receiver to spread over into adjacent symbol intervals, producing a distortion of the received symbols known as *inter-symbol interference* (ISI)[3]. In high-speed digital communication systems, this kind of distortion plays a very important role in designing an efficient receiver. In fact, in such systems, an efficient use of the available channel bandwidth is almost exclusively limited by the presence of the ISI.

In practical digital communication systems, designed to transmit data over the band-limited channels, the frequency response characteristics C(f) of the channel is not known with sufficient precision for the design of a fixed (time invariant) demodulator for mitigating the ISI. Consequently, a part of this demodulator is made adaptive. The filter or signal processing algorithm for handling the ISI problem at the receiver contains a number of parameters, which are adaptively adjusted on the basis of observable measurement of the channel characteristics. This technique is well known as adaptive channel equalization.

Generally, a fast start up, a high tracking capability of the adaptive equalizer algorithm, and the equalizer computation efficiency, together with the independence of the equalizer on the eigenvalue spread of the correlation matrix of the tap input to the equalizer, are the most desirable performance characteristics in the high speed digital communication applications. Hence various algorithms for adaptive channel equalization have been developed and applied to compensate for the non-ideal characteristics of the communication channel. Adaptive algorithms can be divided into two classes, according to the type of the error function being minimized [3,4,6].

The first class consists of equalizers based on stochastic gradient approach, in which the statistical expectation of the squared error is approximated by its instantaneous value, and is then minimized at each iteration. The main drawbacks of these algorithms are the slow convergence and dependence of convergence on the eigenvalue spread of the correlation matrix. In addition, the algorithm has poor adaptivity for non-stationary channels, which may have rapid time variations.

However, simplicity and low computational complexity of LMS algorithms have made them so widely used thus far.

The second class belongs adaptive filtering algorithms based on the method of least squares. According to this method, a cost function or index of performance that is defined as the sum of weighted error squares is minimized, where error or residual is defined as the difference between some desired response and actual filter output. These algorithms are generally considered as fast converging algorithms, and their rate of convergence is almost independent of the eigenvalue spread of the correlation matrix and therefore quite suitable for channel equalization problem. The recursive least squares family of linear adaptive algorithms fall into three distinct categories as follows [2,4,6]: (1) Standard RLS algorithm; (2) Square-root RLS algorithms; and (3) Fast RLS algorithms: (a) Order recursive adaptive filters (lattice structure) and (b) Fast transversal filters.

The standard RLS and square root RLS algorithms have a computational complexity of $O(M^2)$, where M is the filter order. By contrast fast RLS algorithms have computational complexity of $O(M)$. Low computational complexity, fast rate of convergence, and numerical stability of order recursive adaptive algorithms make them quite attractive for communication channel equalization.

It is very important to point out that the adaptive equalization techniques considered in this thesis have been implemented for communication channels that introduce linear distortion, such as telephone channels. On the fading multi-path channels, which introduce nonlinear distortion, the decision feedback equalizers

(DFE) are appropriate [13]. However, these techniques are currently being used as channel equalizers on the fading multi-path channels, mainly because of their simplicity of implementation.

1.2 Thesis Scope

The objective of this thesis is to study and implement adaptive channel equalization techniques for the case of band limited channels and compare their performances.

The thesis outline is as follows. Chapter II presents the background and problem formulation, Chapter III presents adaptive channel equalization techniques and discusses various aspects of them. Chapter IV presents the simulation environment and the results of the experiments on equalizing a band-limited channel using the linear adaptive algorithms discussed in Chapter III of this thesis. Chapter V presents conclusions and comments based on the results obtained in Chapter IV.

CHAPTER II

BACKGROUND AND PROBLEM FORMULATION

2.1 Problem Description

In transmission over time dispersive channel, each symbol extends beyond the time interval allocated to it. The distortion caused by the resulting overlap of the received symbols is known as inter Symbol Interference (ISI). The efficient use of the available channel bandwidth is essentially limited by the ISI. Other disturbances like additive channel noise usually have fewer effects on the transmission rate. A basic block diagram of a digital communication system consisting of a data source, channel model, an additive noise source, receiving filter and an adaptive channel equalizer in the receiver front end is shown in Figure 1.

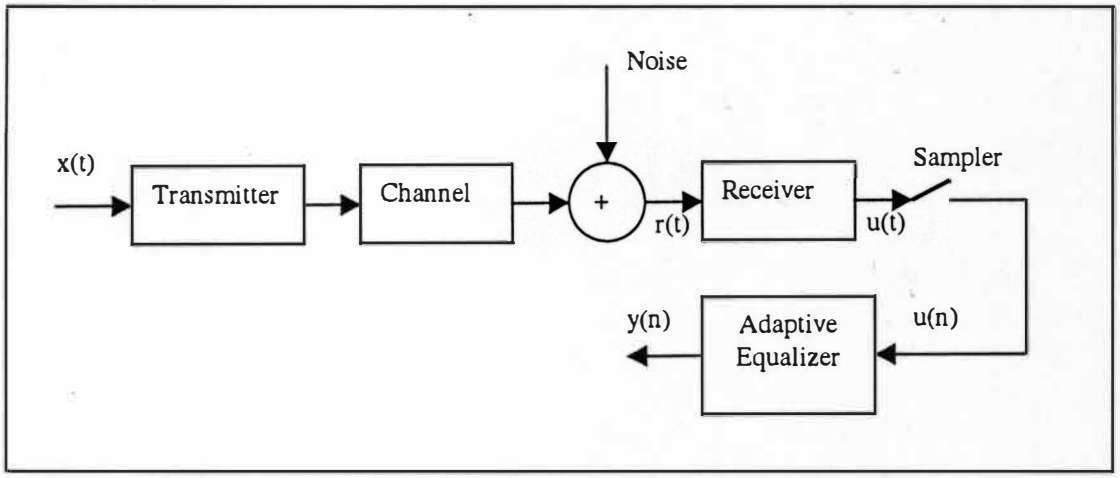

Figure 1. A Basic Block Diagram of a Digital Communication System.

The transmitted symbol sequence {x(n)} is passed through the channel. Prior to being inputted to the adaptive equalizer, the ISI distorted signal is further corrupted by an additive noise, n(t). The channel characteristics are not known a priori. The receiver objective is to accurately estimate the channel parameters and to compensate for the introduced ISI. The task of the adaptive equalizer at the receiver front is to provide estimates of the channel parameters and to reconstruct the transmitted symbol x(n), based on the channel output sequence, using some pre-specified adaptation criterion. Also, the adaptive equalizer must be able to track the time variations of the channel parameter.

2.2 Band-limited Channel Characteristics

In general, a band-limited channel, such as the telephone channel is characterized by a linear filter having an equivalent low pass frequency response $C(f)$:

$$C(f) = A(f)e^{j\theta(f)} \tag{2.1}$$

where $A(f)$ is called the amplitude response and $\theta(f)$ is called the phase response. Another characteristic that is sometimes used in place of the phase response is the envelope delay or group delay, which is defined as

$$\tau(f) = -\frac{1}{2}\frac{d\theta(f)}{df} \tag{2.2}$$

The channel is said to be non-distorting or ideal if, within the bandwidth W occupied by the transmitted signal, $A(f)$=constant and $\theta(f)$ is a linear function of frequency or envelope delay $\tau(f)$=const. On the other hand, if $A(f)$ and $\tau(f)$ are not constant within the bandwidth occupied by the transmitted signal, the channel distorts the signal. If $A(f)$ is not constant, the distortion is called amplitude distortion and if $\tau(f)$ is not constant, the distortion on the transmitted signal is called delay distortion.

As the result of the amplitude and delay distortion caused by the nonideal channel frequency response characteristic $C(f)$, a succession of pulses transmitted through the channel at rates comparable to the bandwidth W, are smeared to the point that they are no longer distinguishable as well defined pulses at the receiving terminal. Instead, they overlap and, thus, we have Inter Symbol Interference (ISI). As an example of the effect of delay distortion on a transmitted pulse, Figure 2(a) illustrates a band-limited pulse having zeros periodically spaced in time at points labeled $\pm T$, $\pm 2T$, *etc.* If the pulse amplitude, as in pulse amplitude modulation (PAM) conveys information, for example, then one can transmit a sequence of pulses, each of which has a peak at the periodic zeros of the other pulses. Transmission of the pulse through a channel modeled as having a linear envelope delay characteristic $\tau(f)$ [quadratic phase $\theta(f)$], however, results in the received pulse shown in Figure 2(b) having zero crossings that are no longer periodically spaced. Consequently a sequence of successive pulses would be smeared into one another, and the peaks of the pulses would no longer be distinguishable. Thus, the channel delay distortion results in inter

Symbol Interference.

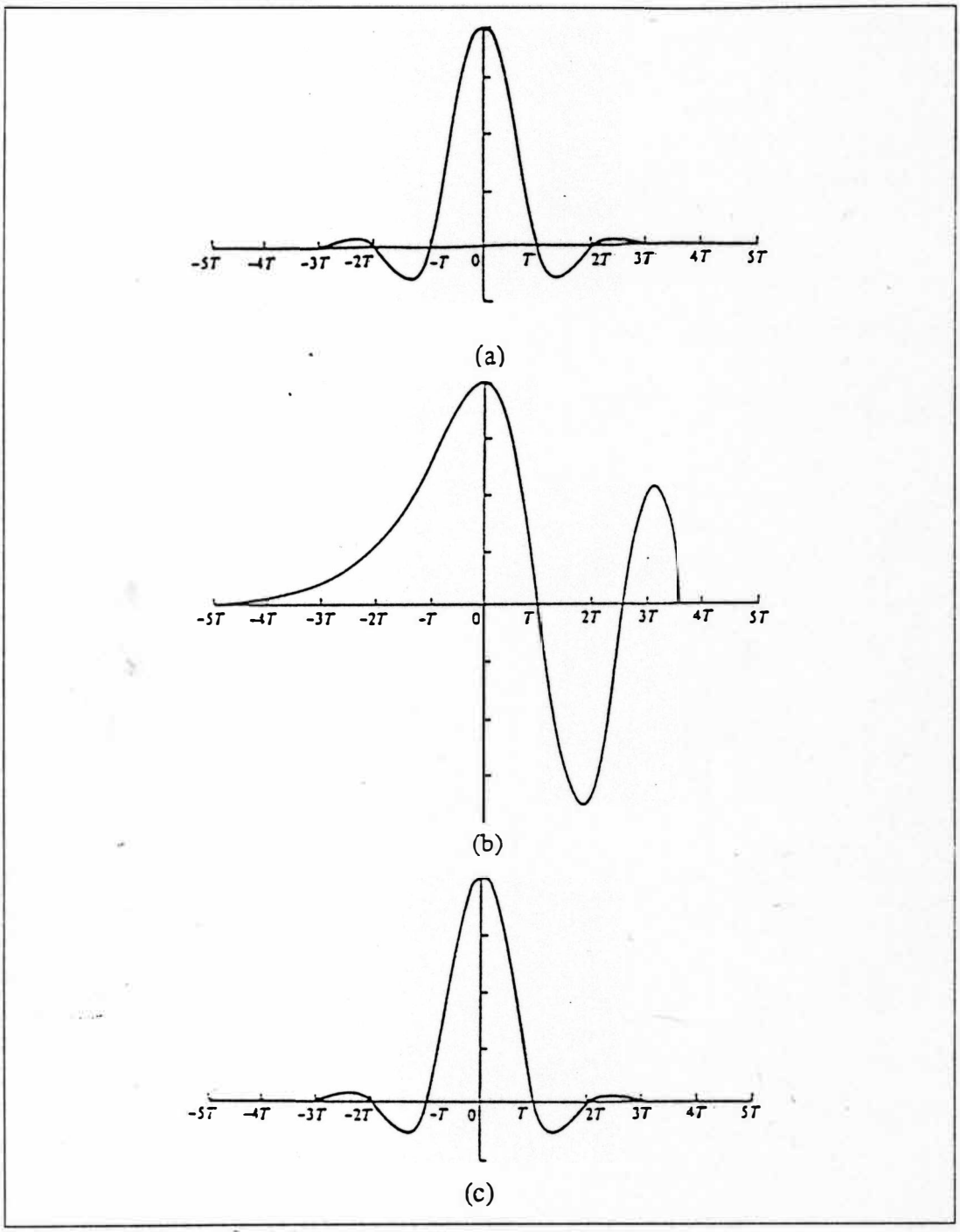

Figure 2. Effect of Channel Distortion.

As will be discussed in later chapters, it is possible to compensate for the nonideal frequency response characteristics of the channel by use of an equalizer at the demodulator. Figure 2(c) illustrates the output of s linear equalizer that compensates for the linear distortion in the channel [1,3].

In addition to linear distortion, signals transmitted through communication channels are subject to nonlinear distortion, frequency offset, phase jitters, and thermal noise. A channel model that encompasses all these impairments becomes too complicated to analyze. The channel models that are adopted in this thesis are linear filters with additive Gaussian noise [2,3].

2.3 Inter Symbol Interference (ISI)

In this section, we shall present a model that characterizes the ISI. The digital modulation methods to which this treatment applies are, pulse amplitude modulation(PAM), phase shift keying (PSK) and quadrature amplitude modulation(QAM). The transmitted signal for these three types of modulation may be expressed as [1,3]

$$\begin{aligned} s(t) &= v_c(t)\cos 2\pi f_c t - v_s(t)\sin 2\pi f_c t \\ &= \mathrm{Re}[v(t)e^{j2\pi f_c t}] \end{aligned} \tag{2.3}$$

where $v(t)=v_c(t)+jv_s(t)$ is called the equivalent low pass signal, f_c is the carrier frequency, and Re[] denotes the real part of the quantity in brackets.

In general, the equivalent low-pass signal is expressed as

$$v(t) = \sum_{n=0}^{\infty} I_n g_T(t - nT) \tag{2.4}$$

where $g_T(t)$ is the basic pulse shape that is selected to control the spectral characteristics of the transmitted signal, $\{I_n\}$ the sequence of transmitted information symbols selected from a signal constellation consisting of M points, and T the signal interval ($1/T$ is the symbol rate). For PAM, PSK, and QAM, the values of I_n are points from M-ary signal constellations. The signal $v(t)$ is transmitted over a bandpass channel that may be characterized by an equivalent low-pass frequency $C(f)$. Consequently, the equivalent received signal can be represented as

$$r(t) = \sum_{n=0}^{\infty} I_n h(t - nT) + n(t) \tag{2.5}$$

in the channel. To characterize the ISI, suppose that the received signal is passed through a receiving filter and then sampled at the rate $1/T$ samples/s. Let $h_R(t)$ denote the impulse response of the receive filter and where $h(t) = g_T(t) * c(t)$ and $c(t)$ is the impulse response of the equivalent low-pass local loop channel, where '$*$' denotes convolution and $n(t)$ represents the additive noise $x(t) = g_T(t) * c(t) * h_R(t)$. Then the output of the received filter can be expressed as

$$y(t) = \sum_{n=0}^{\infty} I_n x(t - nT) + v(t) \tag{2.6}$$

where $v(t) = n(t) * h_R(t)$. Now if $y(t)$ is sampled at times $t=KT$, $k=0,1,2\cdots$ we have

$$\begin{aligned} y(kT) \equiv y_K &= \sum_{n=0}^{\infty} I_n x(kT - nT) + v(kT) \\ &= \sum I_n x_{k-n} + v_k, \qquad k = 0,1,2\cdots \end{aligned} \tag{2.7}$$

The sample values $\{y_k\}$ can be expressed as

$$y_k = x_0(I_k + \frac{1}{x_0} \sum_{n=0, n\neq k}^{\infty} I_n x_{k-n}) \quad k = 0, 1, 2, \cdots \tag{2.8}$$

by setting the arbitrary factor x_0 *to unity we get*

$$y_k = I_k + \sum_{n=0, n\neq k}^{\infty} I_n x_{k-n} + v_k \tag{2.9}$$

the term I_k *represents the desired information symbol at the* k^{th}, sampling instant, the term

$$\sum_{n=0, n\neq k}^{\infty} I_n x_{k-n} \tag{2.10}$$

represents the ISI, and v_k is the additive noise variable at the k^{th} sampling instant [1,3].

2.4 Communication Channel Models

2.4.1 Time Invariant Channels

Generally, the channel can be modeled with a time varying impulse response. However, if the time variations are much slower than the duration of the signaling interval, the channel can be considered as being time invariant over a large number of signaling intervals. This assumption is realistic not only for telephone channels, but also for some radio multipath channels such as tropospheric scatter channels. A common model of a communication channel is the transversal filter structure; i.e. tapped delay line (TDL). The transfer function of such channels may be written as

$$H(z) = \sum_{i=0}^{L} c_i z^{-i} \tag{2.11}$$

where L is the number of signaling intervals spanned by the Inter Symbol Interference and c_i, i=0,...L are the TDL weights. The received signal is given by

$$y(n) = \sum_{i=0}^{L} c_i \, x(n-i) + v(n) \tag{2.12}$$

where $x(n)$ is the symbol sequence at the channel input and $v(n)$ is a sequence of zero

mean white Gaussian noise samples with variance σ^2.

2.4.2 Time Varying Channels

The radio multipath fading channels are examples of time varying channels. The discrete-time transfer function of such channels may be written as

$$H(z) = \sum_{i=0}^{L} c_i(n)\, z^{-i} \tag{2.13}$$

The variable channel coefficients $\{c_i(n)\}$ are modeled by passing white Gaussian noise (WGN) through a low pass filter[4,6]. The received signal is given by

$$y(n) = \sum_{i=0}^{L} c_i(n)\, x(n-1) + v(n) \tag{2.14}$$

2.5 Adaptive Filters

The design of Wiener filter requires a priori information about the statistics of the data to be processed. The filter is optimum only when the statistical characteristics of the input data match the a priori information on which the design of the filter is based. When this information is not known completely, however, it may not be possible to design the Wiener filter or else the design may no longer be optimal. A straightforward approach that may be used in such situations is the "estimates" and "plug" procedure. This is a two-stage process whereby the filter first "estimates" the statistical parameters of the relevant signals and then "plugs" the

results so obtained into a non-recursive formula for computing the filter parameters. For real-time operation, this procedure has the disadvantage of requiring excessively elaborate and costly hardware. A more efficient method is to use an adaptive filter. Such a device is self-designing in that the adaptive filter relies its operation on a recursive algorithm, which makes it possible for the filter to perform satisfactorily in an environment where complete knowledge of the relevant signal characteristics is not available. The algorithm starts from some predetermined set of initial conditions, representing complete ignorance about the environment. Yet, in a stationary environment, it is found that after successive iterations of the algorithm it converges to the optimum Wiener solution in some statistical sense. In a non-stationary environment, the algorithm offers a tracking capability, whereby it can track time variations in the statistics of the input data, provided that the variations are sufficiently slow. A wide variety of recursive algorithms have been developed thus far for the operation of adaptive filters. In the final analysis, the choice of one algorithm over another is determined by various factors [2].

1. Rate of convergence: This is defined as the number of iterations required for the algorithm, in response to stationary inputs, to converge "close enough" to the optimum Wiener solution in the mean square sense. A fast rate of convergence allows the algorithm to adapt rapidly to a stationary environment of unknown statistics.

2. Misadjustment: For an algorithm of interest, this parameter provides a quantitative measure of the amount by which the final value of the mean-squared error, averaged over an ensemble of adaptive filters, deviates from the minimum

mean-squared error that is produced by the Wiener filter.

3. Tracking: When an adaptive filtering algorithm operates in a nonstationary environment, the algorithm is required to track statistical variations in the environment. Two contradictory features, however, influence the tracking performance of the algorithm, (a) rate of convergence, and (b) steady –state fluctuations due to algorithm noise.

4. Robustness: In one context, robustness refers to the ability of the algorithm to operate satisfactorily with ill-conditioned input data. A data set is ill conditioned when condition number of the underlying correlation matrix is large.

5. Computational complexity**:** Here the issue of concern is the number of multiplication's, divisions, and additions/subtractions required to make one complete iteration of the algorithm.

6. Structure: This refers to the structure of information flow in the algorithm, determining the manner in which it is implemented in hardware. For example, an algorithm whose structure exhibits high modularity, parallelism, or concurrency is well suited for implementation using very large scale integration. (VLSI)[2,4,6].

2.6 Linear Filter Structures

The operation of a linear adaptive filtering algorithm involves two basic processes:(1) a *filtering* process designed to produce an output in response to a sequence of input data, and (2) an *adaptive* process, the purpose of which is to provide a mechanism for the *adaptive control* of an adjustable set of parameters used

in the filtering process. These two processes work interactively with each other. Naturally, the choice of a structure for the filtering process has a profound effect on the operation of the algorithm as a whole. There are two types of filter structure that are mostly used in adaptive filtering. These structures are as follows [2,4]:

2.6.1 Transversal Filters

The transversal filter, also referred to as tapped delay line (TDL) filter, consists of three basic elements, as depicted in Figure 3: (1) unit delay element, (2) multiplier, and (3) adder. The number of delay elements used in the filter determines the finite duration of its impulse response and is commonly refereed to as the filter order. The filter output is given by

$$y(n) = \sum_{k=0}^{m-1} w_k^* \, u(n-k) \tag{2.15}$$

This equation is called a finite convolution sum in the sense that it convolves the finite duration impulse response of the filter, w_k^* with the filter input $u(n)$ to produce the filter output $y(n)$[2].

2.6.2 Lattice Predictors

A lattice predictor is modular in structure in that it consists of individual stages, each of which has the appearance of a lattice, hence the name "lattice" as a structural depictor. Figure 4 depicts a lattice predictor consisting of M-1 stages; the

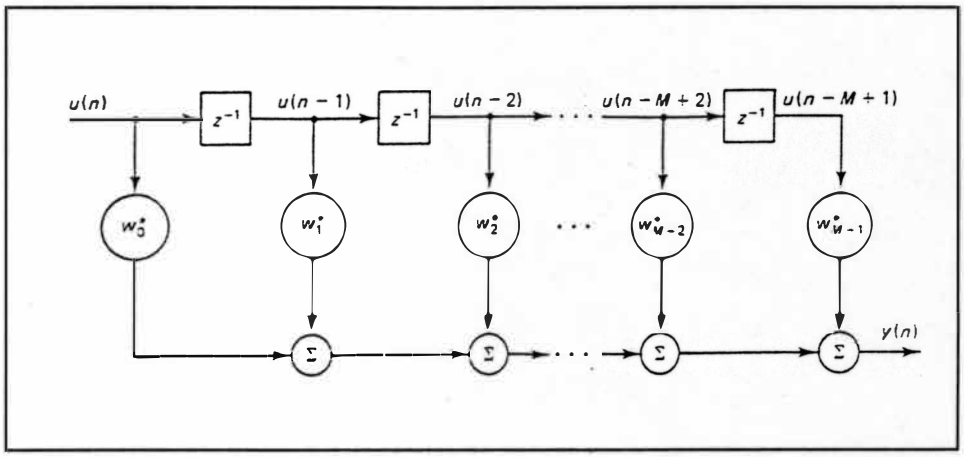

Figure 3. Transversal Filter.

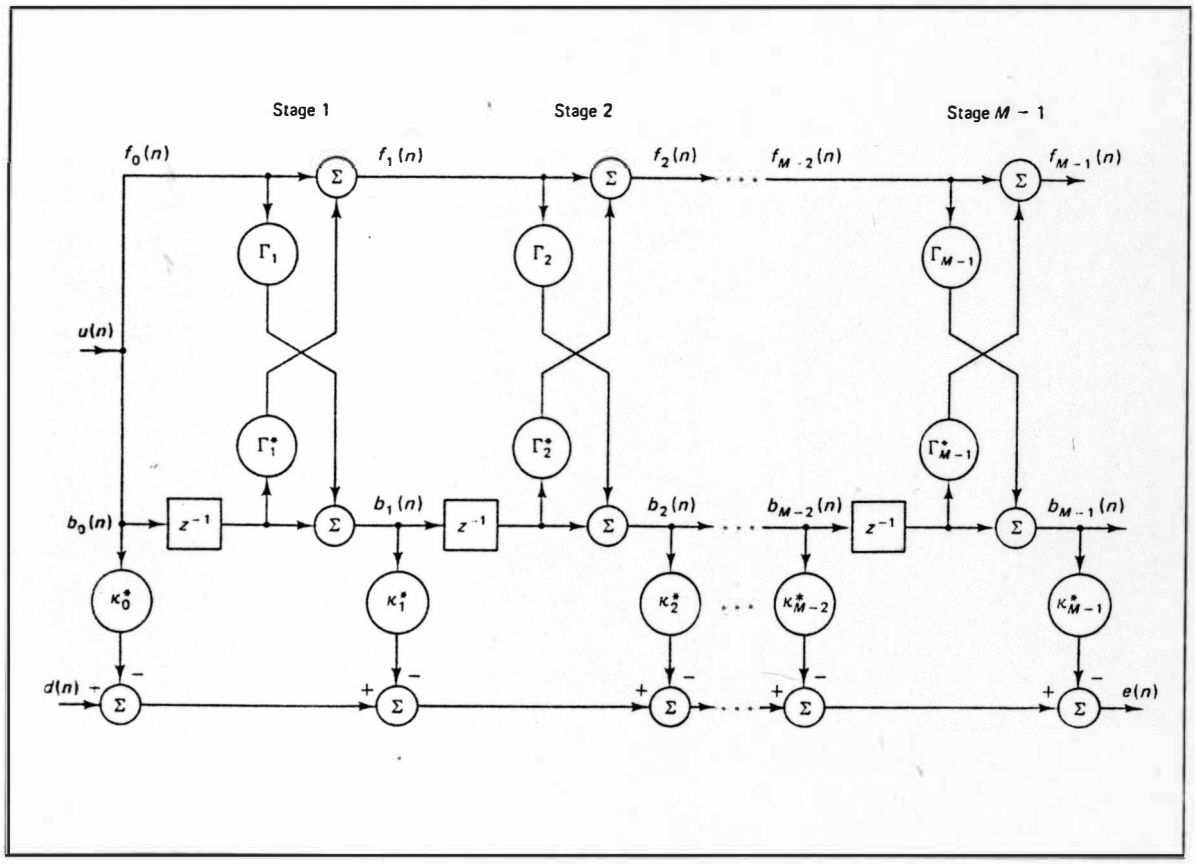

Figure 4. Multistage Lattice Predictor.

number M-1 is referred to as the predictor order. The mth stage of the lattice predictor in Figure 4 is described by the pair of input-output relations (assuming the use of complex-valued, wide-sense stationary input data):

$$\begin{aligned} f_m(n) &= f_{m-1}(n) + \Gamma_m^* b_{m-1}(n-1) \\ b_m(n) &= b_{m-1}(n-1) + \Gamma_m f_{m-1}(n) \end{aligned} \tag{2.16}$$

where m=1,2,..., M-1, and M-1 is the final predictor order. The variable $f_m(n)$ is the mth *forward prediction error* , and $b_m(n)$ is the mth *backward prediction error.* The coefficient Γ_m is called the mth *reflection coefficient.* The forward prediction error $f_m(n)$ is defined as the difference between the inputs *u(n-1),...u(n-m)* and its one step predicted value. Correspondingly, the backward prediction error $b_m(n)$ is defined as the difference between the input *u(n-m)* and its "backward" prediction based on the set of m "future" inputs *u(n),...,u(n-m+1).* Considering the conditions at the input of stage 1 in Figure 4, we have

$$f_0(n) = b_0(n) = u(n) \tag{2.17}$$

where $u(n)$ is the lattice predictor input at time n. Thus, starting with the initial conditions of Equation (2.17) and given the set of reflection coefficients $k_1, k_2, \cdots k_M$ we may determine the final pair of outputs $f_{M-1}(n)$ and $b_{M-1}(n)$ by moving through the lattice predictor, stage by stage.

For a correlated input sequence $u(n)$, $u(n),\ldots,u(n-M+1)$ drawn from a

stationary process, the backward prediction errors b_0, $b_1(n),\ldots,b_{M-1}(n)$ form a sequence of uncorrected random variables. More over, there is one to one correspondence between these two sequences of random variables in the sense that if we are given one of them, we may uniquely determine the other, and vice versa. Accordingly, a linear combination of the backward prediction errors may be used to provide an estimate of some desired response between $d(n)$, as depicted in the lower half of Figure 4. The arithmetic difference between $d(n)$ and the estimate so produced represents the estimation error(n). The process described herein is referred to as a joint-process estimation. Naturally, we may use the original input sequence to produce an estimate of the desired response $d(n)$ directly. The indirect method depicted in Figure 4, however has the advantage of simplifying the computation of the tap weights $k_0, k_1, \cdots k_m$, by exploiting the uncorrected nature of the corresponding backward prediction errors used in the estimation [2].

2.7 Adaptive Algorithms

The challenge facing the user of adaptive algorithms is, first, to understand the capabilities and limitations of various adaptive algorithms and, second, to use this understanding in the selection of the appropriate algorithm for the application at hand.

Basically, we may identify two distinct approaches for deriving recursive algorithms for the operation of linear adaptive filters as follows:

2.7.1 Stochastic Gradient Approach

Here we may use a tapped delay line or transversal filter as the structural basis for implementing the linear adaptive filter. For the case of stationary inputs, the cost function, also referred to as the index of performance, is defined as the *mean squared error* (i.e., the mean squared value of the difference between the desired response and the transversal filter output). There are two stages for updating the tap weights of the adaptive transversal filter recursively. First the system of *Wiener-Hopf* equations (i.e., the matrix equation defining the optimum Wiener solution)is modified through the use of the *method of steepest descent,* a well-known technique in optimization theory. This modification requires the use of a gradient vector, the value of which depends on two parameters: the correlation matrix of the tap inputs in the transversal filter and the cross correlation vector between the desired response and the same tap inputs. Next, the instantaneous values for these correlations are used so as to derive an estimate for the gradient vector, making it assume a stochastic character in general. The resulting algorithm is widely known as the *least mean square (LMS)* algorithm [1,2,3,4].

The LMS algorithm is simple and yet capable of achieving satisfactory performance under the right conditions. Its major limitations are a relatively slow rate of convergence and a sensitivity to variations in the condition number of the correlation matrix of the tap inputs to the equalizer. Nevertheless, the LMS algorithm is highly popular and is widely used in variety of applications [4].

The stochastic gradient approach may also be pursued in the context of a lattice structure. The resulting adaptive filtering algorithm is called *gradient adaptive lattice (GAL) algorithm* [10]. In their own individual ways, the LMS and GAL algorithms are just two members of the stochastic gradient family of linear adaptive filters, although it must be said that the LMS algorithm is by far the most popular member of this family.

2.7.2 Least Square Estimation

The second approach to the development of linear adaptive filtering algorithms is based on the method of least squares. According to this method, a cost function or index of performance that is defined as the sum of *weighted error squares*, is initialized, where the error or residual is itself defined as the difference between some desired response and the actual filter output. *Recursive least squares (RLS)* estimation may be viewed as a special case of Kalman filtering and there is one to one correspondences between the kalman variables and RLS variables. We may classify the recursive least squares family of linear adaptive filtering algorithms into three distinct categories, depending on the approach taken [2,9]:

1. Standard RLS algorithm, which assumes the use of a transversal filter as the structural basis of the linear adaptive filter. This algorithm has the virtues and suffers from the same limitations as the standard Kalman filtering algorithm. The limitations include lack of numerical robustness and excessive computational complexity.

2. Square root RLS algorithms, which are based on QR decomposition of the incoming data matrix. These linear adaptive filters are referred to as *square-root adaptive filters*, because in a matrix sense they represent the square-foot forms of the standard RLS algorithm.

3. Fast RLS algorithm. The standard RLS algorithm and square-root RLS algorithms have a computational complexity that increases as the square of M, where M is the number of adjustable weights in the algorithm. Such algorithms are often referred to as $O(M^2)$ algorithms. By contrast, the LMS algorithm is $O(M)$ algorithm. When M is large, the computational complexity of $O(M^2)$ algorithms may become objectionable from a hardware implementation point of view. There is therefore a strong motivation to modify the formulation of RLS algorithm in such a way that the computational complexity assumes an $O(M)$ form. This objective is achievable, in the case of temporal processing, first by virtue of the inherent redundancy in the Toeplitz structure of the input data matrix and, second, by exploiting this redundancy through the use of linear least-square prediction in both the forward and backward directions. The resulting algorithms are known collectively as fast RLS algorithms; they combine the desirable characteristics of recursive linear least squares estimation with an $O(M)$ computational complexity. Two types of fast RLS algorithms may be identified, depending on the filtering structure employed [2,4]: (a) order-recursive adaptive filters, which are based on a lattice like structure for making linear forward and backward predictions; and (b) fast transversal filters, in which the linear forward and backward predictions are performed using separate transversal, filters.

Certain (but not all) realizations of order-recursive adaptive algorithms are known to be numerically stable, whereas fast transversal filters suffer from a numerical stability problem and therefore require some form of stabilization for them to be of some practical use.

2.8 Equalizer Modes of Operation

The block diagram of a digital communication system in which an adaptive equalizer is used is depicted in Figure 5. There are two modes of operation of the adaptive channel equalizer, the training mode and the steady state mode. During the training mode, the switch (S) is in position 1, where a generated replica of the known transmitted sequence is used to train the equalizer by adjusting its coefficients so as to match those of the unknown channel. After the training period (acquisition time), the switch S is switched to position 2, and the transmitter starts to transmit the information sequence. To track the possible time variations in the channel parameters, the equalizer coefficients must continue to adjust in an adaptive manner while receiving data. This is accomplished, as illustrated in Figure 5, by using the estimates, x(n), of the desired sequence, in place of the reference x(n), to generate the error sequence. In the following chapter we describe the typical adaptive algorithms for recursively adjusting the equalizer coefficients.

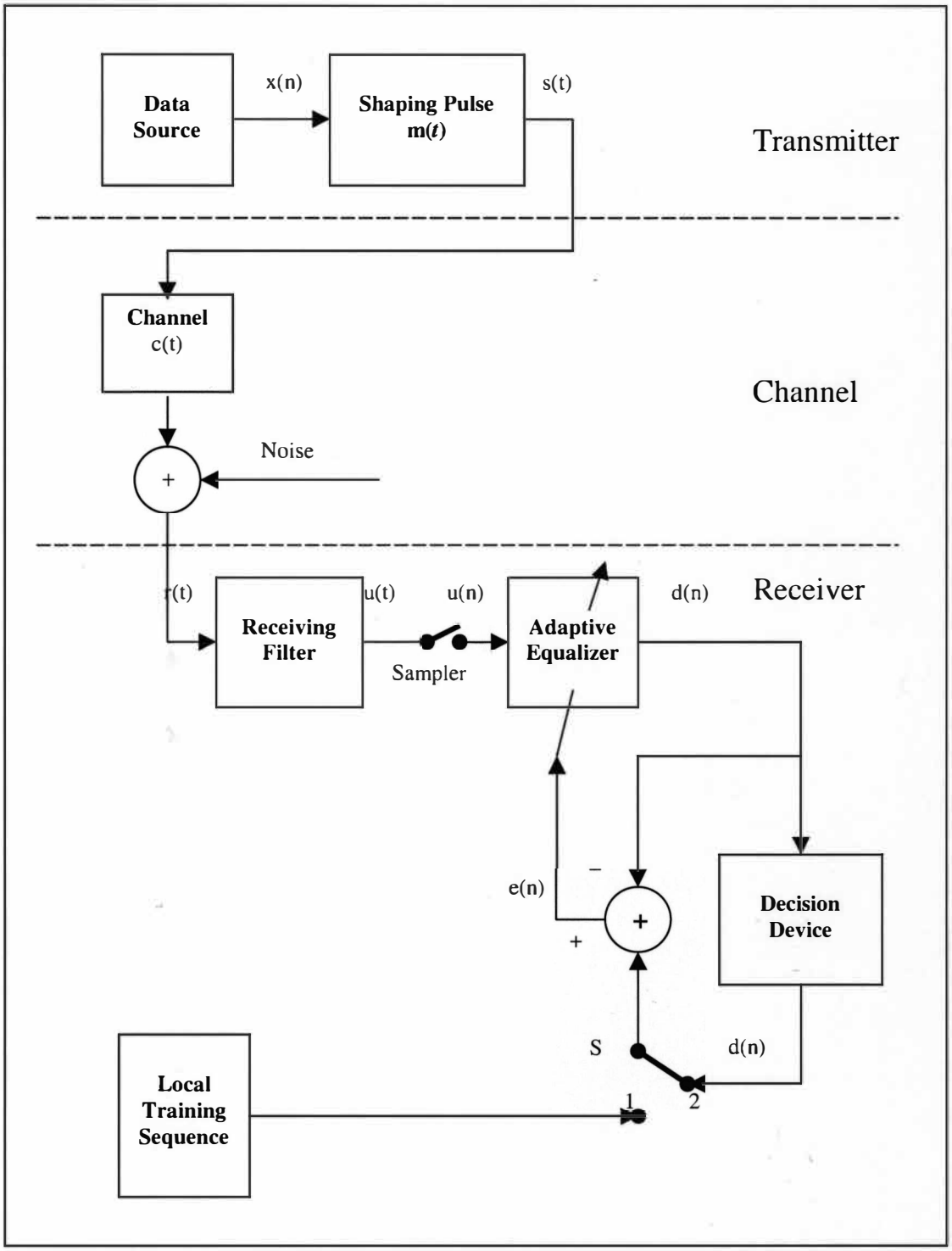

Figure 5. Channel Equalizer Modes of Operation.

CHAPTER III

ADAPTIVE CHANNEL EQUALIZATION TEHNIQUES

To compensate for Inter Symbol Interference (ISI) in digital communication systems, channel equalizers are placed at the receiver front end. They operate on baseband signals so that in band-pass systems, they appear after the demodulators. Most of the equalizer operations are performed using adaptive filtering algorithms, as mentioned in the previous chapter. Adaptive algorithms are used to continuously update the filter weights (coefficients) so as to track the possible moderately fast time variations of the channel. The equalization techniques can be accomplished by a variety of adaptive filtering algorithms.

In this chapter we start with a rewiev of the equalizer coefficients optimization criteria. Then we discuss the steepest decent algorithm and the stochastic gradient least mean square (LMS) adaptive algorithms for the transversal filter structure realizations (TDL). The least squares (LS) minimization procedures will be discussed for both the TDL and the lattice filter structure realization, as they are the ones often used in adaptive equalization applications. Although both finite impulse response (FIR) and infinite impulse response (IIR) have been considered for adaptive filtering, the FIR filter are by far the most practical and widely used [2,4,6].

3.1 Equalizer Coefficient Optimization

The mean square error (MSE) criterion is the most frequently used optimization criterion in filtering applications. In the MSE optimization criterion, the filter coefficient vector w(n) is adjusted to minimize the mean square value of the following error function:

$$e(n) = d(n) - \hat{d}(n) \tag{3.1}$$

where $e(n)$ is the estimation error sequence, $\hat{d}(n)$ represents the estimate of the desired response and $d(n)$ represents the desired response.

The M by 1 equalizer input data vector $\mathbf{u}(n)$ at the time instant n, where M is the order of equalizer, may be written as

$$\mathbf{u}(n) = [\mathbf{u}_1(n) \;\; \mathbf{u}_2(n) \cdots \mathbf{u}_M(n)]^T \tag{3.2}$$

The estimate of the desired sequence at time instant n is the inner product

$$\hat{d}(n) = \mathbf{w}^T(n)\mathbf{u}(n) \tag{3.3}$$

where $\mathbf{w}(n)$ is the M^{th} order equalizer tap weight vector.

$$\mathbf{w}(n) = [\mathbf{w}_1(n) \;\; \mathbf{w}_2(n) \cdots \mathbf{w}_M(n)]^T \tag{3.4}$$

Now mean square value of the error function $e(n)$, denoted by $\xi(n)$, may be defined as

$$\xi(n) = E[e^2(n)] = E\{[d(n) - \hat{d}(n)]^2\} \tag{3.5}$$

$$= E\{[d(n) - \mathbf{w}^T(n)\mathbf{u}(n)]^2\}$$

$$= E[d(n)]^2 - 2\mathbf{w}^T(n)E[\mathbf{u}(n)\mathbf{u}(n)] + \mathbf{w}^T(n)E[\mathbf{u}(n)\mathbf{u}^T(n)]\mathbf{w}(n) \tag{3.6}$$

Defining the M by M correlation matrix of the equalizer inputs as

$$\mathbf{R}(n) = E[\mathbf{u}(n)\mathbf{u}^T(n)] \tag{3.7}$$

and the M by 1 cross-correlation vector between the unequalized input data and the desired sequence as

$$\mathbf{q}(n) = E[\mathbf{d}(n)\mathbf{u}(n)] \tag{3.8}$$

then Equation (3.6) can be written in terms of **R**(n) and **q**(n) as

$$\xi(n) = E[d^2(n)] - 2\mathbf{w}^T(n)\mathbf{q}(n) + \mathbf{w}^T(n)\mathbf{R}(n)\mathbf{w}(n) \tag{3.9}$$

The MSE is the energy of the difference between the desired and the estimated information symbol.

There are two reasons in choosing this function as a performance index: (1) the uniqueness of the obtained solution, since the MSE is a quadratic function of the coefficients, w_i's, and therefore has only one global minimum; and (2) the simplicity of the solution, since the coefficient vector **w**(n) is directly determined by a set of linear equations.

Invoking the orthogonality principle, the minimization of the quadratic function (3.9) yields the normal equations. The coefficient vector **w**(n) shall be so selected that it renders the error e(n) orthogonal to the input data **u**(n), thus

$$E[\mathbf{e}(n)\mathbf{u}^T(n)] = 0 \tag{3.10}$$

Substituting for e(n) from equation (3.1) with d(n) being defined by equation (3.3) into (3.10), yields

$$E[d(n) - \mathbf{w}^T(n)\mathbf{u}(n)\mathbf{u}^T(n)] = 0$$

or, equivalently,

$$E[\mathbf{u}(n)\mathbf{u}^T(n)]\mathbf{w}(n) = E[\mathbf{u}(n)d(n)] \tag{3.11}$$

Using the definition (3.7) and (3.11), we obtain the discrete form of Wiener-Hopf normal equation.

$$\mathbf{R}(n)\mathbf{w}(n) = \mathbf{q}(n) \tag{3.12}$$

The direct solution of Eq. (3.12) yields the optimum (in the mean square sense) tap weights vector $\mathbf{w}_{opt}(n)$

$$\mathcal{W}_{opt}(n) = \mathbf{R}^{-1}(n)\mathbf{q}(n)$$

$$= \{E[d(n)\mathbf{u}^T(n)]\}\{E[\mathbf{u}(n)\mathbf{u}^T(n)]\}^{-1} \tag{3.13}$$

substituting the value of $w_{opt}(n)$ into equation (3.9), we obtain the minimum output MSE as

$$\xi_{\min}(n) = E[d^2(n)] - \mathbf{q}^T(n)\mathbf{R}^{-1}(n)\mathbf{q}(n)]$$

$$= E[d^2(n)] - \mathbf{q}^T(n)\mathbf{w}_{opt}(n)] \qquad (3.14)$$

Since $\mathbf{R}(n)$ and $\mathbf{q}(n)$ are, generally, not available, their estimates must be found first. The procedures for finding their estimates and thus, solving the normal Equation (3.12) are discussed in the following sections.

3.2 The Gradient Least Mean Squares Methods

3.2.1 The Steepest Descent Method

One way to solve equation (3.12) is to use the steepest descent iterative method. Assuming that the autocorrelation matrix, $\mathbf{R}(n)$, and the crosscorrelation vector, $\mathbf{q}(n)$, are known a priori, one may compute the gradient vector $\nabla(n)$ which is defined as

$$\nabla(n) = \partial\xi(n)/\partial\mathbf{w}(n)$$

$$= -2\mathbf{q}(n) + 2\mathbf{R}(n)\mathbf{w}(n) \qquad (3.15)$$

Equation (3.15) can be rewritten as

$$\nabla(n) = -2E[d(n)\mathbf{u}(n)] + 2E[\mathbf{u}(n)\mathbf{u}^T(n)]\mathbf{w}(n) \qquad (3.16)$$

where the correlation matrix **R**(n) and the cross correlation vector **q**(n) are defined by (3.7) and (3.8), which are repeated here for convenience

$$\mathbf{R}(n) = E[\mathbf{u}(n)\mathbf{u}^T(n)] \tag{3.17}$$

$$\mathbf{q}(n) = E[\mathbf{d}(n)\mathbf{u}(n)] \tag{3.18}$$

The steepest descent method can be summarized in the following steps:

1. Initialize the tap weight vector, normally with the zero vector value i.e. **w**(0)=0.

2. Use this value of **w**(n) to compute $\nabla(n)$ according to equation (3.15).

3. Compute the next time update of the weight vector w(n) according to:

$$w(n+1) = w(n) - \mu\nabla(n) \tag{3.19}$$

where μ is a positive number known as the step size or adaptation parameter.

4. Increment n by 1, go back to step 2 and repeat process.

It is intuitively reasonable that successive corrections to the tap weight vector in the direction of the negative of the gradient vector should eventually lead to the minimum value of the mean square error. However, the significant drawback of this algorithm is the lack of the knowledge of the exact values of the **R**(n) and **q**(n).

3.2.2 The Stochastic LMS Algorithm

The channel characteristics are not known beforehand. Therefore, the gradient vector can not be computed exactly, but must be estimated from the available

information. In other words, gradient and tap-weight vectors are iteratively updated with every incoming data sample. One method for performing this task is the least mean square(LMS) algorithm originally introduced by Widrow and Hoff [13]. This LMS algorithm does not require measurements of the pertinent autocorrelation function nor does it require matrix inversion. The algorithm obtains an estimate of the gradient vector ∇(n) by dropping the expectation operation operator E from equation (3.16), to yield the instantaneous estimate:

$$\nabla(n) = -2\mathbf{u}(n)d(n) + 2\mathbf{u}(n)\mathbf{u}^T(n)\mathbf{w}(n) \tag{3.20}$$

Substituting $\nabla(n)$ into equation (3.19), we obtain the tap weight updating recursive relation:

$$\begin{aligned} w(n+1) &= w(n) + \mu\mathbf{u}(n)d(n) - \mu\mathbf{u}(n)\mathbf{u}^T(n)\mathbf{w}(n) \\ &= w(n) + \mu\mathbf{u}(n)[d(n) - \mathbf{u}^T(n)\mathbf{w}(n)] \\ &= w(n) + \mu\mathbf{u}(n)e(n) \end{aligned} \tag{3.21}$$

where e(n) is the posteriori estimate error, defined as

$$e(n) = d(n) - \mathbf{u}^T(n)\mathbf{w}(n) \tag{3.22}$$

The adaptive algorithm described by equation (3.21) and (3.22) is known as the LMS or the stochastic gradient algorithm. Summary of the LMS algorithm is given in Figure 6.

For small values of the parameter μ the algorithm provides a low initial convergence rate. Large values of μ, on the other hand, lead to a noisy adaptation and may even cause divergence. Therefore choice of the step size parameter, μ, determines the trade-off between the algorithms adaptation speed and the minimum attainable output MSE, since μ is the only controllable parameter.

M: number of taps
μ : step size parameter

Initialization:
$\mathbf{w}(0)=\mathbf{0}$

Given $\mathbf{u}(n)$= M-by-1 tap input vector at time n
d (n)=desired response at time n

Compute

$$e(n) = d(n) - \mathbf{w}^{H}(n)\mathbf{u}(n)$$
$$w(n+1) = w(n) + \mu\mathbf{u}(n)e^{*}(n)$$

Figure 6. Summary of LMS Algorithm.

Several variations of the basic LMS algorithm have been proposed to improve the algorithm performance, some use a fixed step size as in equation (3.21), others use variable step size. Normalized LMS algorithm is one variation of basic LMS algorithm where the fixed step size in equation (3.18) is replaced by:

$$\mu(n) = \frac{\beta}{\|\mathbf{u}(n)\|^2} \tag{3.23}$$

We may view the normalized LMS algorithm as LMS algorithm with a time varying step size parameter. The normalized LMS is convergent in the mean square if the adaptation constant β satisfies $0 < \beta < 2$.

When the tap input vector u(n) is small, numerical difficulties may arise because then we have to divide by a small value for the squared norm $\|\mathbf{u}(n)\|^2$. To overcome this problem, we slightly modify (3.23) as:

$$\mu(n) = \frac{\beta}{\alpha + \|\mathbf{u}(n)\|^2} \tag{3.24}$$

where α>0, as before $0 < \beta < 2$.

The summary of Normalized LMS algorithm is given in Figure 7.

M: number of taps
0 <β> 2
α>0
Initialization:
w (0)=**0**

Given **u**(n)= M-by-1 tap input vector at time n
d (n)=desired response at time n

Compute

$$e(n) = d(n) - \mathbf{w}^H(n)\mathbf{u}(n)$$

$$w(n+1) = w(n) + \frac{\beta}{\alpha + \|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n)$$

Figure 7. Summary of Normalized LMS Algorithm.

3.3 Least Square Adaptation Techniques

The major advantage of the LMS gradient algorithms lies in its low computational complexity, $O(M)$, where M is the order of the equalizer filter. However the price to pay is its slow convergence rate. Since the step size μ, which is the only adjustable parameter for controlling the algorithm adaptivity, has to be bounded for the stability purpose, the slow convergence is mainly due to this fundamental limitation. Consequently, in order to obtain faster convergence, it is necessary to devise more complex algorithms, which involve additional parameters. To do this, we adopt the deterministic least squares criterion instead of the statistical approach used in the LMS and the related gradient algorithms. In recursive implementations of the least square method, we start the computation with some known initial conditions and use the information contained in the new data samples to update the algorithm variables.

3.3.1 The Recursive Least Square Algorithm (RLS)

The RLS algorithm is based on the least squares estimate of the tap weight vector. With the estimate of the filter at time n-1, we can calculate the tap weight vector at time n upon the arrival of new data. An important feature of the RLS algorithm is that it uses the information in the input data from the instant of time the algorithm is initiated. This makes the rate of convergence faster by an order of magnitude in comparison with the LMS algorithm. The disadvantage is the

computational complexity of the algorithm and the sensitivity to round off errors that accumulate due to the recursive nature of the algorithm resulting in the numerical instability. The RLS algorithm makes use of the relation in matrix algebra known as the *Matrix Inversion Lemma*, which can be stated as follows: If A and B are two $N\times N$ matrices and C is a $N\times M$ matrix , D is $M\times M$ positive definite matrix, related by

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^{H} \tag{3.25}$$

then, the inverse of A is given as

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^{H}\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^{H}\mathbf{B} \tag{3.26}$$

The above theorem can be proved by multiplying the equations (3.25) and (3.26), evaluating the product and by recognising that the product of a square matrix and its inverse is an identity matrix.

The M-by-M exponentially weighted input correlation matrix is defined as

$$\Phi(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)\mathbf{u}^{H}(i) \tag{3.27}$$

$$= \lambda\,[\sum_{i=1}^{n-1} \lambda^{n-1-i}\mathbf{u}(i)\mathbf{u}^{H}(i)] + \mathbf{u}(n)\mathbf{u}^{H}(n)$$

$$= \lambda\ \Phi(n-1) + \mathbf{u}(n)\mathbf{u}^{H}(n) \tag{3.28}$$

where λ is a positive constant close to, but less than 1. When λ equals 1, we have the ordinary method of least squares. The inverse of 1-λ is, roughly speaking, a measure

of the memory of the algorithm. The case of λ=1 corresponds to infinite memory. Assuming the input correlation matrix to be positive definite and non-singular, we shall apply the matrix inversion lemma to the above equation. From equation (3.26), and identifying

$$\mathbf{A} = \Phi(n)$$

$$\mathbf{B}^{-1} = \lambda\ \Phi(n-1)$$

$$\mathbf{C} = \mathbf{u}(n)$$

$$\mathbf{D} = 1$$

we get the recursive expression for the input correlation matrix as:

$$\Phi^{-1}(n) = \lambda^{-1}\Phi^{-1}(n-1) - \frac{\lambda^{-2}\Phi^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^{H}(n)\Phi^{-1}(n-1)}{1+\lambda^{-1}\mathbf{u}(n)\mathbf{u}^{H}(n)\Phi^{-1}(n-1)\mathbf{u}(n)} \tag{3.29}$$

substituting in the above Equation (3.27),

$$\mathbf{P}(n) = \Phi^{-1}(n) \tag{3.30}$$

and

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^{H}(n)\mathbf{P}(n-1)\mathbf{u}(n)} \tag{3.31}$$

we have,

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}k(n)\mathbf{u}^{H}(n)\mathbf{P}(n-1) \tag{3.32}$$

$$\mathbf{k}(n) = \lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n) - \lambda^{-1}k(n)\mathbf{u}^{H}(n)\mathbf{P}(n-1)\mathbf{u}(n)$$

$$= \mathbf{P}(n)\mathbf{u}(n) = \Phi^{-1}(n)\mathbf{u}(n) \tag{3.33}$$

The M-by-1 cross correlation $\theta(n)$ between the tap inputs of the the transversal filter and the desired response is defined by

$$\theta(n) = \sum \lambda^{n-1}\mathbf{u}(i)d^{*}(i) \tag{3.34}$$

$$\theta(n) = \lambda\theta(n-1) + \mathbf{u}(n)d^{*}(n) \tag{3.35}$$

The recursive equation for the tap weight vector is developed as follows:

$$\begin{aligned} \mathbf{w}(n) &= \Phi^{-1}(n)\theta(n) \\ &= \mathbf{P}(n)\theta(n) \\ &= \lambda\mathbf{P}(n)\theta(n-1) + \mathbf{P}(n)\mathbf{u}(n)d^{*}(n) \end{aligned} \tag{3.36}$$

substituting Equation 3.31 for p(n) in Equation 3.36, we get

$$\begin{aligned} \mathbf{w}(n) &= \mathbf{P}(n-1)\theta(n-1) - \mathbf{k}(n)\mathbf{u}^{H}(n)\mathbf{P}(n-1)\theta(n-1) + \mathbf{P}(n)\mathbf{u}(n)d^{*}(n) \\ &= \Phi^{-1}(n-1)\theta(n-1) - \mathbf{k}(n)\mathbf{u}^{H}(n)\Phi^{-1}(n-1)\theta(n-1) \\ &\quad + \mathbf{P}(n)\mathbf{u}(n)d^{*} \\ &= \mathbf{w}(n-1) - \mathbf{k}(n)\mathbf{u}^{H}(n)\mathbf{w}(n-1) + \mathbf{P}(n)\mathbf{u}(n)d^{*}(n) \end{aligned} \tag{3.37}$$

Finally, using Equation (3.33) that **P**(n)**u**(n) equals the gain vector **k**(n); we get the desired recursive equation for updating the tap weight vector:

$$\begin{aligned}\mathbf{w}(n) &= \mathbf{w}(n-1) + \mathbf{k}(n)[d^*(n) - \mathbf{u}^H(n)\mathbf{w}(n-1)] \\ &= \mathbf{w}(n-1) + \mathbf{k}(n)e^*(n)\end{aligned} \tag{3.38}$$

where e(n) is known as the a priori estimation error defined by

$$e(n) = d(n) - \mathbf{u}^T(n)\mathbf{w}^*(n-1) \tag{3.39}$$

The Equations (3.31), (3.39), (3.38) and (3.32), collectively and in that order, constitute the RLS algorithm. We note that, in particular, Equation (3.39) describes the filtering operation of the algorithm, whereby the transversal filter is excited to compute the a priori estimation error $e(n)$. Equation (3.38) describes the adaptive operation of the algorithm, whereby the tap-weight vector is updated by incrementing its old value by an amount equal to the complex conjugate of the a priori estimation error $e(n)$ times the time-varying gain vector $\mathbf{k}(n)$, hence the name gain vector. Equations (3.31) and (3.32) enable us to update the value of the gain vector itself. An important feature of the RLS algorithm described by these equations is that the inversion of the correlation matrix Φ(n) is replaced at each step by a simple scalar division. The applicability of RLS algorithm requires that we initialize the recursion of Equation (3.31) by choosing a starting value P(0) that assures the non-singularity of the correlation matrix Φ(n). We may do this by evaluating the following inverse.

$$[\sum_{i=-n_0}^{0}\lambda^{-i}\mathbf{u}(i)\mathbf{u}^{H}(i)]^{-1}$$

A simpler procedure, however, is to modify the expression slightly for the correlation matrix Φ(n) by writing

$$\Phi(n) = \sum_{i=1}^{n}\lambda^{n-i}\mathbf{u}(i)\mathbf{u}^{H}(i) + \delta\lambda^{n}\mathbf{I} \qquad (3.40)$$

where **I** is the *M-by-M* identity matrix, and δ is a small positive constant. Thus putting *n*=0 in Equation (3.39), we have

$$\Phi(n) = \delta\ \mathbf{I} \qquad (3.41)$$

correspondingly, for the initial value of P(*n*) equal to the inverse of the correlation matrix Φ(n), we set

$$\mathbf{P}(0) = \delta^{-1}\mathbf{I} \qquad (3.42)$$

It only remains for us to choose an initial value for the tap-weight vector. It is customary to set

$$\mathbf{w}(0) = \mathbf{0} \qquad (3.43)$$

where **0** is the *M*- by-1 null vector. The summary of RLS algorithm is given in Figure 8.

Initialization:

$$\mathbf{P}(0) = \delta^{-1}\mathbf{I},$$
$$\mathbf{w}(0) = \mathbf{0}$$

For each instant of time, n=1, 2, . . . , compute

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^{H}(n)\mathbf{P}(n-1)\mathbf{u}(n)}$$
$$e(n) = d(n) - \mathbf{w}^{H}(n-1)\mathbf{u}(n)$$
$$\mathbf{w}(n) = \mathbf{w}(n-1)\mathbf{k}(n)e^{*}(n)$$
$$\mathbf{P}(n) = \lambda^{-1}(\mathbf{p}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^{H}(n)\mathbf{P}(n-1)$$

Figure 8. Summary of RLS Algorithm.

3.3.2 The Recursive Least Square Lattice Algorithm (RLSL)

In this section we will describe another class of least square (LS) algorithms. This class is based on the lattice filter structure described in Chapter II. These algorithms are mixed time and order (MTO) recursive rather than being time recursive as in the case of RLS, described in the previous section. These algorithms are rooted in recursive least-square estimation theory and therefore retain two unique attributes of the RLS algorithm: (1) fast rate of convergence; and (2) Insensitivity to variations in the eigenvalue spread of the underlying correlation matrix of the input data. However, unlike the RLS algorithm, the computational complexity of the algorithms considered in this section increase linearly with the number of adjustable

filter parameters. This highly desirable property is a direct result of order recursiveness, which gives the adaptive algorithm a computationally efficient, modular, lattice like structure. In particular, as the filter order is increased from m to $m+1$, say, the lattice filter permits us to carry over certain information gathered from the previous computations pertaining to the filter order. A recursive least-squares lattice (LSL) algorithm is a joint estimator in the sense that it provides for the estimation of two sets of filtering coefficients jointly. Forward and backward reflection coefficients that characterize a multistage lattice predictor optimized in the least-squares sense. The number of stages in the predictor equals the prediction order. Regression coefficients characterize a linear least-squares estimator of some desired response. A recursive LSL algorithm may be structured in basically three different forms, depending on the type of prediction and estimation errors used as variables, and the manner in which the reflection coefficients are computed. In version I, summarized in Figure 9, the variables are a posteriori forms of prediction and estimation errors, and the reflection coefficients are computed indirectly. In version II, summarized in Figure 10, the variables are a priori forms of prediction and estimation errors, and the reflection coefficients are again computed indirectly. In version III, summarized in Figure 11, the variables are (as in version II) a priori forms of prediction and estimation errors, but the reflection and regression coefficients are all computed directly. As a result of this direct computation, error feedback is introduced into the operation of the algorithm.

Predictions:

For $n = 1, 2, 3, \ldots$ compute the various order updates in the following sequence: $m = 1, 2, \ldots, M$, where M is the final order of the least-squares lattice predictor:

$$\Delta_{m-1}(n) = \lambda \Delta_{m-1}(n-1) + \frac{b_{m-1}(n-1) f^*_{m-1}(n)}{\gamma_{m-1}(n-1)}$$

$$\Gamma_{f,m}(n) = -\frac{\Delta_{m-1}(n)}{\mathcal{B}_{m-1}(n-1)}$$

$$\Gamma_{b,m}(n) = -\frac{\Delta^*_{m-1}(n)}{\mathcal{F}_{m-1}(n)}$$

$$f_m(n) = f_{m-1}(n) + \Gamma^*_{f,m}(n) b_{m-1}(n-1)$$

$$b_m(n) = b_{m-1}(n-1) + \Gamma^*_{b,m}(n) f_{m-1}(n)$$

$$\mathcal{F}_m(n) = \mathcal{F}_{m-1}(n) - \frac{|\Delta_{m-1}(n)|^2}{\mathcal{B}_{m-1}(n-1)}$$

$$\mathcal{B}_m(n) = \mathcal{B}_{m-1}(n-1) - \frac{|\Delta_{m-1}(n)|^2}{\mathcal{F}_{m-1}(n)}$$

$$\gamma_m(n-1) = \gamma_{m-1}(n-1) = -\frac{|b_{m-1}(n-1)|^2}{\mathcal{B}_{m-1}(n-1)}$$

Filtering:

For $n = 1, 2, 3, \ldots$ compute the various order updates in the following sequence: $m = 0, 1, \ldots, M$

$$\rho_m(n) = \lambda \rho_m(n-1) + \frac{b_m(n)}{\gamma_m(n)} e^*_m(n)$$

$$\kappa_m(n) = \frac{\rho_m(n)}{\mathcal{B}_m(n)}$$

$$e_{m+1}(n) = e_m(n) - \kappa^*_m(n) b_m(n)$$

Initialization

1. To initialize the algorithm, at time $n = 0$ set
   $\Delta_{m-1}(0) = 0$
   $\mathcal{F}_{m-1}(0) = \delta$, δ = small positive constant
   $\mathcal{B}_{m-1}(0) = \delta$
   $\gamma_0(0) = 1$
2. At each instant $n \geq 1$, generate the various zeroth-order variables as follows:
   $f_0(n) = b_0(n) = u(n)$
   $\mathcal{F}_0(n) = \mathcal{B}_0(n) = \lambda \mathcal{F}_0(n-1) + |u(n)|^2$
   $\gamma_0(n-1) = 1$
3. For joint-process estimation, initialize the algorithm by setting at time $n = 0$
   $\rho_m(0) = 0$
   At each instant $n \geq 1$, generate the zeroth-order variable
   $e_0(n) = d(n)$

Note: For prewindowed data, the input $u(n)$ and desired response $d(n)$ are both zero for $n \leq 0$.

Figure 9. Summary of the Recursive LSL Algorithm Using Aposteriori Estimation Error.

Predictions:

Starting with $n = 1$, compute the various order updates in the following sequence $m = 1, 2, \ldots, M$, where M is the final order of the least-squares predictor:

$$\eta_m(n) = \eta_{m-1}(n) + \Gamma^*_{f,m}(n-1)\psi_{m-1}(n-1)$$

$$\psi_m(n) = \psi_{m-1}(n-1) + \Gamma^*_{b,m}(n-1)\eta_{m-1}(n)$$

$$\Delta_{m-1}(n) = \lambda\Delta_{m-1}(n-1) + \gamma_{m-1}(n-1)\psi_{m-1}(n-1)\eta^*_{m-1}(n)$$

$$\mathcal{F}_{m-1}(n) = \lambda\mathcal{F}_{m-1}(n-1) + \gamma_{m-1}(n-1)|\eta_{m-1}(n)|^2$$

$$\mathcal{B}_{m-1}(n) = \lambda\mathcal{B}_{m-1}(n-1) + \gamma_{m-1}(n)|\psi_{m-1}(n)|^2$$

$$\Gamma_{f,m}(n) = -\frac{\Delta_{m-1}(n)}{\mathcal{B}_{m-1}(n-1)}$$

$$\Gamma_{b,m}(n) = -\frac{\Delta^*_{m-1}(n)}{\mathcal{F}_{m-1}(n)}$$

$$\gamma_m(n) = \gamma_{m-1}(n) - \frac{\gamma^2_{m-1}(n)|\psi_{m-1}(n)|^2}{\mathcal{B}_{m-1}(n)}$$

Filtering:

For $n = 1, 2, 3, \ldots$ compute the various order updates in the following sequence $m = 0, 1, \ldots, M$:

$$\rho_m(n) = \lambda\rho_m(n-1) + \gamma_m(n)\psi_m(n)\alpha^*_m(n)$$

$$\alpha_{m+1}(n) = \alpha_m(n) - \kappa^*_m(n-1)\psi_m(n)$$

$$\mathcal{B}_m(n) = \lambda\mathcal{B}_m(n) + \gamma_m(n)|\psi_m(n)|^2$$

$$\kappa_m(n) = \frac{\rho_m(n)}{\mathcal{B}_m(n)}$$

Initialization

1. To initialize the algorithm, at time $n = 0$ set
   $\Delta_{m-1}(0) = 0$
   $\mathcal{F}_{m-1}(0) = \delta$, δ = small positive constant
   $\mathcal{B}_{m-1}(0) = \delta$
   $\Gamma_{f,m}(0) = \Gamma_{b,m}(0) = 0$
   $\gamma_0(0) = 1$
2. At each instant $n \geq 1$, generate the zeroth-order variables:
   $\eta_0(n) = \psi_0(n) = u(n)$
   $\mathcal{F}_0(n) = \mathcal{B}_0(n) = \lambda\mathcal{F}_0(n-1) + |u(n)|^2$
   $\gamma_0(n) = 1$
3. For joint-process estimation, initialize the algorithm by setting at time $n = 0$
   $\rho_m(0) = 0$
   At each instant $n \geq 1$, generate the zeroth-order variable
   $\alpha_0(n) = d(n)$

Note: For prewindowed data, the input $u(n)$ and desired response $d(n)$ are both zero for $n \leq 0$

Figure 10. Summary of the Recursive LSL Algorithm Using Apriori Estimation Error.

Predictions:

For $n = 1, 2, 3, \ldots,$ compute the various order updates in the following sequence $m = 1, 2, \ldots, M$, where M is the final order of the least-squares predictor:

$$\eta_m(n) = \eta_{m-1}(n) + \Gamma^*_{f,m}(n-1)\psi_{m-1}(n-1)$$

$$\psi_m(n) = \psi_{m-1}(n-1) + \Gamma^*_{b,m}(n-1)\eta_{m-1}(n)$$

$$\mathcal{F}_{m-1}(n) = \lambda \mathcal{F}_{m-1}(n-1) + \gamma_{m-1}(n-1)|\eta_{m-1}(n)|^2$$

$$\mathcal{B}_{m-1}(n) = \lambda \mathcal{B}_{m-1}(n-1) + \gamma_{m-1}(n-1)|\psi_{m-1}(n)|^2$$

$$\Gamma_{f,m}(n) = \Gamma_{f,m}(n-1) - \frac{\gamma_{m-1}(n-1)\psi_{m-1}(n-1)\eta^*_m(n)}{\mathcal{B}_{m-1}(n-1)}$$

$$\Gamma_{b,m}(n) = \Gamma_{b,m}(n-1) - \frac{\gamma_{m-1}(n-1)\eta_{m-1}(n)\psi^*_m(n)}{\mathcal{F}_{m-1}(n)}$$

$$\gamma_m(n) = \gamma_{m-1}(n) - \frac{\gamma^2_{m-1}(n)|\psi_{m-1}(n)|^2}{\mathcal{B}_{m-1}(n)}$$

Filtering:

For $n = 1, 2, 3, \ldots,$ compute the various order updates in the following sequence $m = 0, 1, \ldots, M$:

$$\alpha_{m-1}(n) = \alpha_m(n) - \kappa^*_m(n-1)\psi_m(n)$$

$$\mathcal{B}_m(n) = \lambda \mathcal{B}_m(n-1) + \gamma_m(n)|\psi_m(n)|^2$$

$$\kappa_m(n) = \kappa_m(n-1) + \frac{\gamma_m(n)\psi_m(n)\alpha^*_{m+1}(n)}{\mathcal{B}_m(n)}$$

Initialization

1. To initialize the algorithm, at time $n = 0$ set
   $\mathcal{F}_{m-1}(0) = \delta$, δ = small positive constant
   $\mathcal{B}_{m-1}(0) = \delta$
   $\Gamma_{f,m}(0) = \Gamma_{b,m}(0) = 0$
   $\gamma_0(0) = 1$
2. At each instant $n \geq 1$, generate the zeroth-order variables:
   $\eta_0(n) = \psi_0(n) = u(n)$
   $\mathcal{F}_0(n) = \mathcal{B}_0(n) = \lambda \mathcal{F}_0(n-1) + |u(n)|^2$
   $\gamma_0(n) = 1$
3. For joint process estimation, at time $n = 0$ set
   $\kappa_m(0) = 0$
   At each instant $n \geq 1$, generate the zeroth-order variable
   $\alpha_0(n) = d(n)$

Note: For prewindowed data, the input $u(n)$ and desired response $d(n)$ are both zero for $n \leq 0$.

Figure 11. Summary of the Recursive LSL Algorithm Using Apriori Estimation Error With Error Feedback.

In theory, assuming infinite precision, all three versions of the algorithms are mathematically equivalent. However, in practical situation involving the use of finite-precision arithmetic, the three versions behave differently. In particular, version I and II suffer from a numerical instability problem. On the other hand, version III offers robust numerical properties due to the stabilizing influence of the error feedback built into the computation of the forward and backward reflection coefficients. All three versions of LSL algorithm have linear computational complexities just the same as LMS algorithm. An important property of all recursive LSL algorithms is their modular structure. The implication of this property is that the algorithm structure is linearly scalable. In particular, the prediction order can be readily increased without the need to recalculate all previous values. This property is particularly useful when there is no priori knowledge as to what the final value of the prediction order should be. Another implication of the modular structure of recursive LSL algorithm is that they lend themselves to the use of very large scale integration (VLSI) for their hardware implementation. The use of this sophisticated technology can only be justified if the application of interest calls for the use of the VLSI chip in large numbers.

CHAPTER IV

SIMULATION ENVIROMENT AND RESULTS

4.1 Simulation Setup

This chapter presents the simulation set up of adaptive channel equalization and the results of computer simulation of various channel equalization experiments based on the adaptive algorithms discussed in Chapter III of this thesis. Figure 12 shows the block diagram of the setup used to carry out the computer simulations.

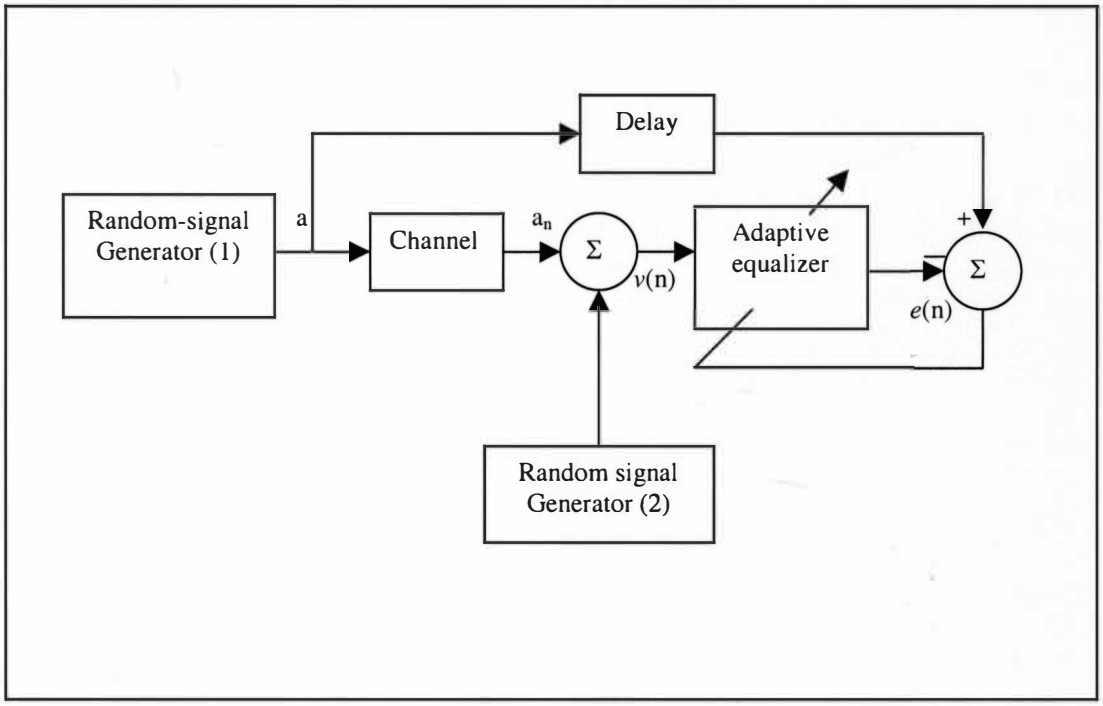

Figure 12. Block Diagram of the Adaptive Equalizer Simulation Setup.

4.2 Channel Impulse Response

The impulse response of a typical band-limited channel is described by the raised cosine:

$$h_n = \begin{cases} \frac{1}{2}[1+\cos(\frac{2\pi}{W}(n-2)], & n = 1,2,3 \\ 0, & otherwise \end{cases}$$

where the parameter W controls the amount of amplitude distortion produced by the channel, with the distortion increasing with W. equivalently, parameter W controls the condition number of the correlation matrix of the tap inputs in the equalizer, with the condition number or equivalently eigenvalue spread $\chi(R)$, increasing with W.

4.3 Signal Generation

Random number generator 1 provides the test signal, $\{a_n\}$, used for probing the channel. The random sequence $\{a(n)\}$ applied to the channel input is in polar form, with $a(n)= \pm 1$, so the sequence has zero mean. Random generator 2 serves as the source of additive white noise $\{v(n)\}$ that corrupts the channel output. These two random number generators are independent of each other. Random number generator 1, after a suitable delay, also supplies the desired response applied to the adaptive equalizer.

4.4 Equalizer Type

The FIR adaptive equalizer filter has been chosen to have M =11 taps. Since the channel has an impulse response $\{h_n\}$ that is symmetric about time n = 2, it follows that the optimum tap wegiths $\{w\}$ of the equalizer are likewise symetric about time n = 5. Accordingly, the channel input $\{a(n)\}$ is delayed by 2+5 =7 samples to provide the desired response for the equalizer. For LMS, Normalized LMS, and RLS the adaptive filter shown in Fig. 4.1 is the Transversal type and for RLSL it is a Lattice type filter.

4.5 Correlation Matrix of the Equalizer Input

The first tap input to the equalizer at time n is given by

$$u(n) = \sum_{1}^{3} h_k a(n-k) + v(n)$$

All the parameters in the above equation are real valued. Hence, the correlation matrix **R**, of the 11 tap inputs of the equalizer, $u(n), U(n-1)\cdots u(n-10)$, is a symmetric 11-by-11 matrix. Also, since h_n, has nonzero values only for n=1, 2, 3, and noise process $v(n)$ is white noise with zero mean and variance σ_v^2, the correlation matrix is quintdiagonal. That is, the only nonzero elements of **R** are on the main diagonal and the four diagonals directly above and below it.

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) & r(2) & 0 & \dots & 0 \\ r(1) & r(0) & r(1) & r(2) & \dots & 0 \\ r(2) & r(1) & r(0) & r(1) & \dots & 0 \\ 0 & r(2) & r(1) & r(0) & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & \dots & r(0) \end{bmatrix}$$

$$where\ r(0) = h_1^2 + h_2^2 + h_3^2 + \sigma_v^2$$

$$r(1) = h_1 h_2 + h_2 h_3$$

$$r(2) = h_1 h_3$$

Tables 1 and 2 show the eigenvalue spread, $\chi(R)$ of the correlation matrix **R**, for W=2.9, W=3.1, W=3.3 and W=3.5 and signal to noise ratios of 30 dB (σ_v^2=0.001) and 20 dB (σ_v^2=0.001) respectively, where σ_v is the vaiance of the input noise to the channel. The eigenvalue spread or the condition number of a matrix is the largest eigenvalue divided by the smallest eigenvalue of the matrix. As shown in Tables 1 and 2, as W increase the eigenvalue spread of the correlation matrix of the input to the equalizer also increases. Higher eigenvalue spreads of correlation matrix, indicate that the input to the equalizer is ill conditioned. It is also noted that the lower signal to noise ratios, reduce the eigenvalue spread of the correlation matrix, thus improving its condition.

Table 1

Summary of the Correlation Matrix
Parameters for S/N=30 dB (σ_v^2=0.001)

W	2.9	3.1	3.3	3.5
r(0)	1.0973	1.1576	1.2274	1.3022
r(1)	0.4388	0.5596	0.6729	0.7775
r(2)	0.0481	0.0783	0.1132	0.1511
λmin	0.3339	0.2136	0.1256	0.0656
λmax	2.0382	2.3761	2.7263	3.0707
$\chi(R)$	6.0782	11.1238	21.7132	46.8216

Table 2

Summary of the Correlation Matrix
Parameters for S/N=30 dB (σ_v^2=0.001)

W	2.9	3.1	3.3	3.5
r(0)	1.1063	1.1666	1.2364	1.3122
r(1)	0.4388	0.5596	0.6729	0.7775
r(2)	0.0481	0.0783	0.1132	0.1511
λmin	0.3429	0.2226	0.1346	0.0746
λmax	2.0385	2.3851	2.7353	3.0797
$\chi(R)$	5.7145	10.7145	20.3278	41.2923

4.6 Equalization Experiments

The adaptive algorithms presented in chapter III are coded using MATLAB version 5.2, to simulate the adaptive channel equalizer shown in Figure 12. The random signals are generated by the two built in random number generators of Matlab. In the following sections of this chapter we shall present the simulation results of LMS, Normalized LMS, RLS and RLSL algorithms summarized in Figures 6, 7, 8, and 11 of chapter III.

4.6.1 Least Mean Square (LMS) Equalizer

Experiment 1: Effect of Eigenvalue Spread. For each eigenvalue spread, an approximation to the ensemble-averaged learning curve of the adaptive transversal equalizer is obtained by averaging the instantaneous squared error versus n over 200 independent trials. The results of this computation are shown in Figure 13. In this Figure, the ensemble averaged squared error is plotted versus number of iteration. These curves are also known as learning curves of the adaptive process. The step size parameter μ=0.075 is assumed for this experiment. The input noise to the channel is assumed to be white noise with zero mean and variance of σ_v^2=0.001 or equivalently the signal to noise ratio equal to 30 dB. We see from Figure 13 that increasing W or the condition number of the correlation matrix of the tap input to the equalizer has the effect of slowing down the rate of convergence of the adaptive equalizer and also, increases the steady-state value of the averaged squared error. For example, when W=2.9, it approximately takes 135 iterations for the adaptive equalizer

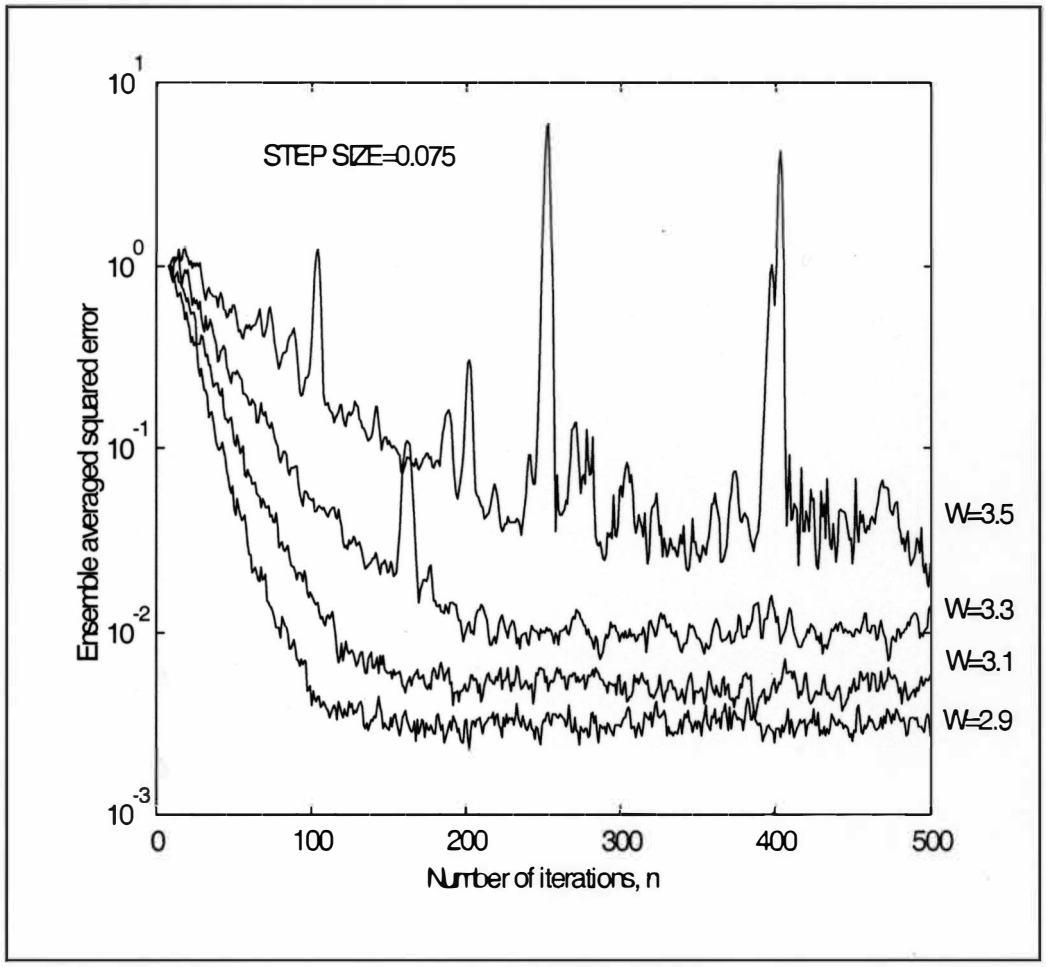

Figure 13. Learning Curve of LMS Equalizer, Experiment 1.

to converge in the mean square, and the averaged squared error after 500 iterations approximately equals 0.003. On the other hand when W=3.3 (i.e., the equalizer input is ill conditioned), the equalizer requires nearly 220 iterations to converge in the mean square, and resulting averaged squared error (after 500 iterations) approximately equals 0.028.

Experiment 2: Effect of Signal to Noise Ratio. In experiment 1, the input noise to the channel was assumed to be white noise with zero mean and variance of σ_v^2=0.001 or equivalently, signal to noise ratio of 30 dB. Here we try σ_v^2=0.01 or equivalently signal to noise ratio of 20 dB. As we see in Figure 14, the higher input noise to the channel has the effect of decreasing the rate of convergence of the adaptive LMS equalizer, as compared with the results of experiment 1. However, the steady-state value of the averaged squared error increases sharply. This would be well expected in the presence of low signal to noise ratio.

Experiment 3: Effect of Step Size Parameter. For this part of the experiment, the learning curves of the channel equalizer are plotted with three different step size parameters for W=2.9, W=3.1 and W=3.3. The results of computations are shown in Figures 15, 16, and 17. As we see in these figures, the rate of convergence of LMS algorithm is greatly affected by the step size parameter. For a very small step size μ=0.0075, the adaptation does not converge even after 1500 iterations. For μ=0.075, the algorithm converges considerably faster than for μ=0.025, while the final steady state value of averaged squared error of the algorithm is lower for μ=0.025 than for μ=0.075. We see in Figure 16, for a relatively ill conditioned correlation matrix of the tap input to the equalizer, the choice over the step size parameter of LMS algorithm is critical and requires careful adjustment. The overall performance of LMS equalizer is seen to be best for the step size parameter, μ=0.075, for the given conditions.

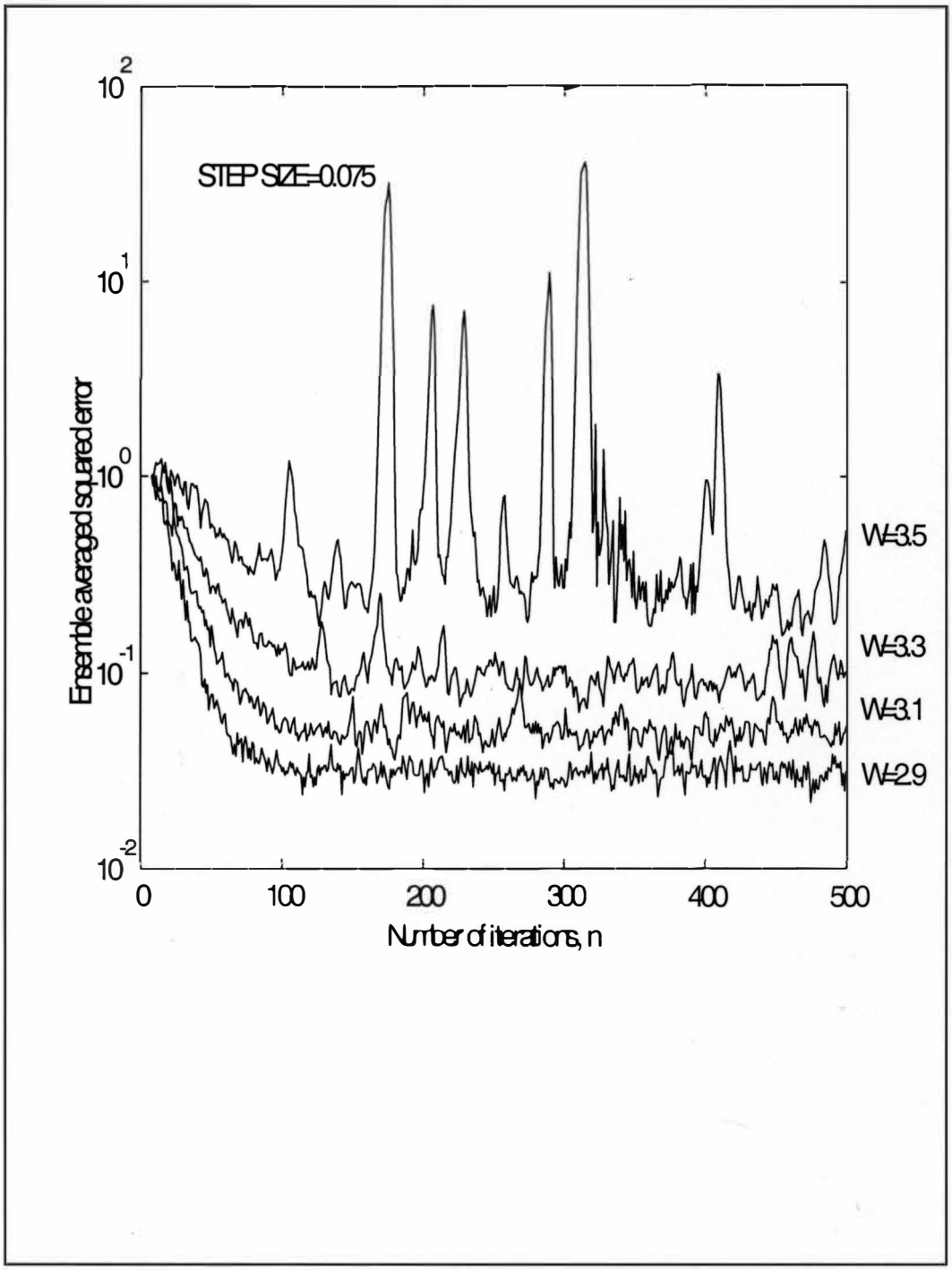

Figure 14. Learning Curve of LMS Eequalizer, Experiment 2.

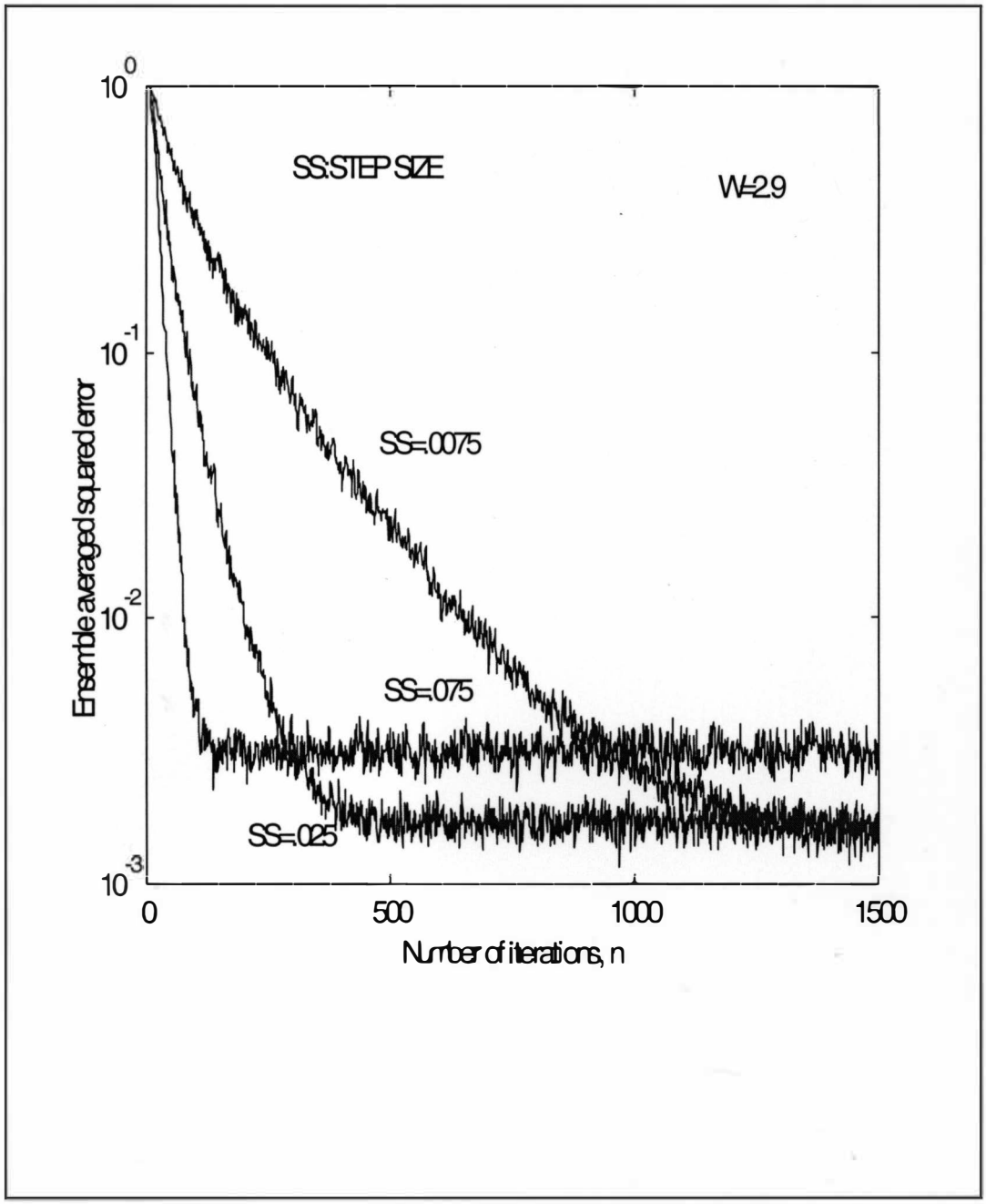

Figure 15. Learning Curve of LMS Equalizer, Experiment 3,W=2.9.

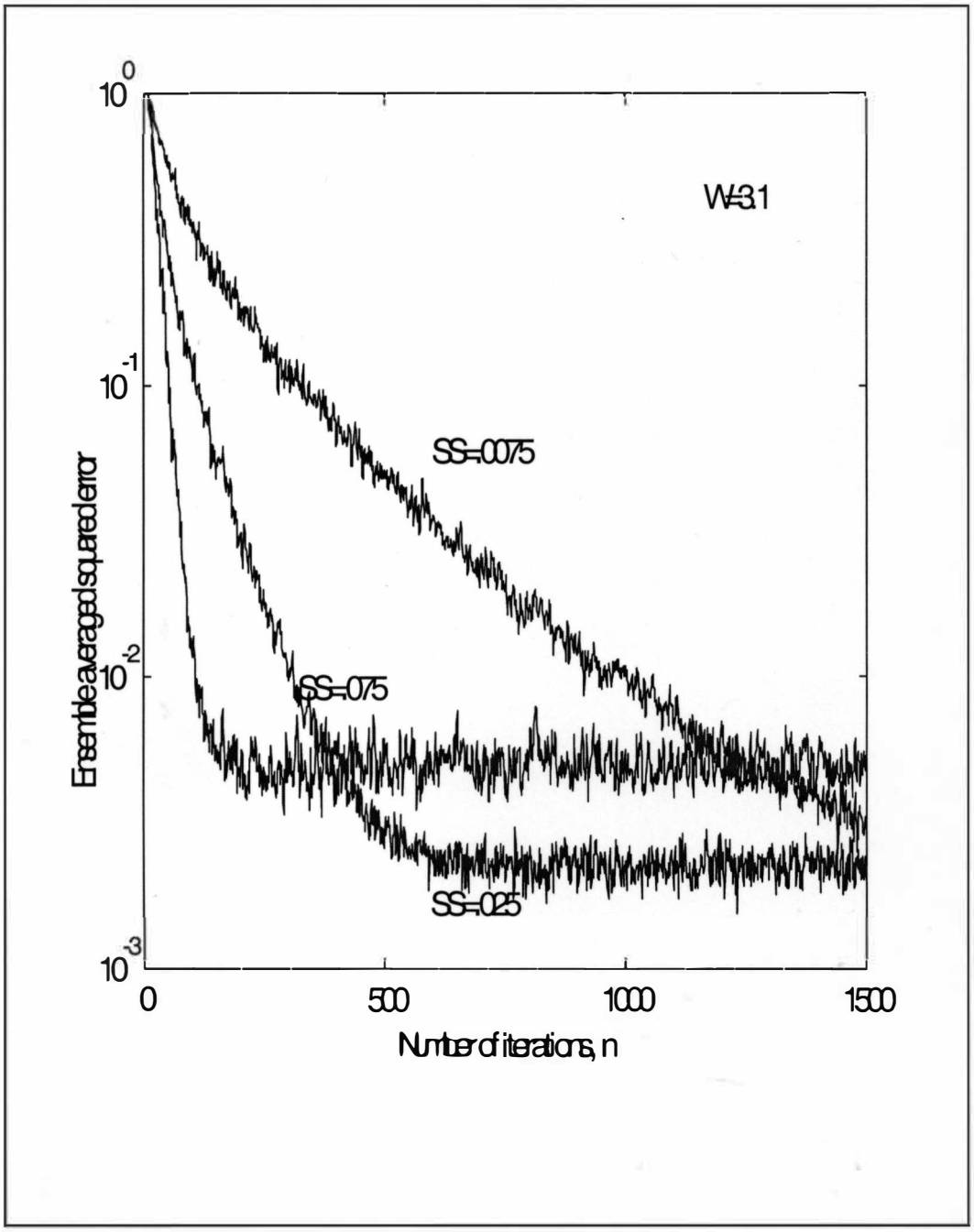

Figure 16. Learning Curve of LMS Equalizer, Experiment 3, W=3.1.

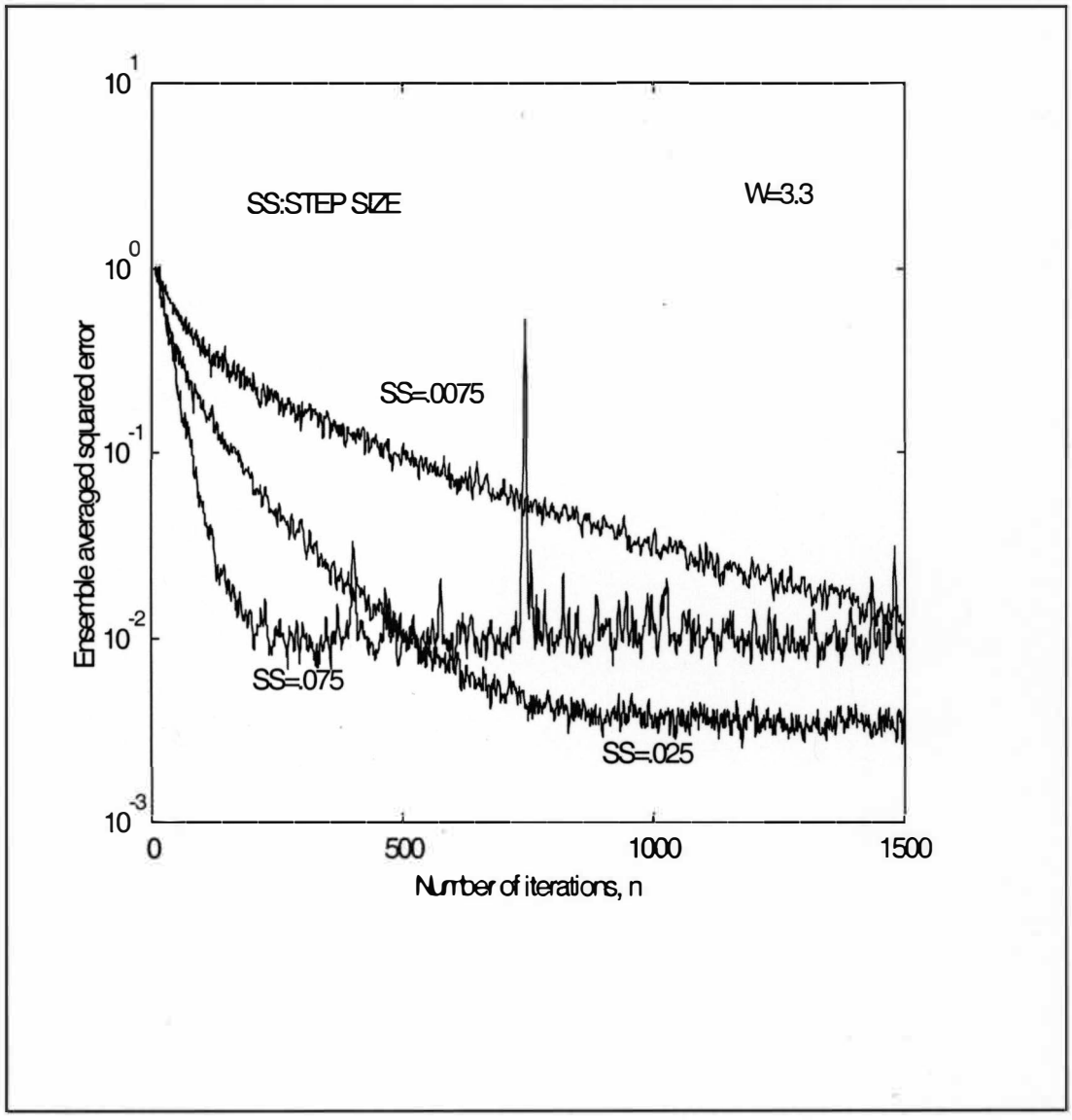

Figure 17. Learning Curve of LMS Equalizer, Experiment 3, W=3.3.

4.6.2 Normalized LMS Equalizer (NLMS)

Experiment 4: Effect of time varying step size parameter. In the NLMS the fixed step size parameter for updating the filter coefficients has been replaced by a time varying parameter, as shown in Table 2. In Figure 18 the learning curves of

ensemble averaged squared errors of this algorithm has been plotted versus number of iteration, n, for W=2.9, W=3.1, W=3.3 and W=3.5. Comparing Figure 13 with Figure 16, we see that the rate of convergence of this equalizer for a relatively ill conditioned input to the equalizer, has been improved by the choice of a time varying step size parameter. In general the adaptation noise, final steady state averaged squared error

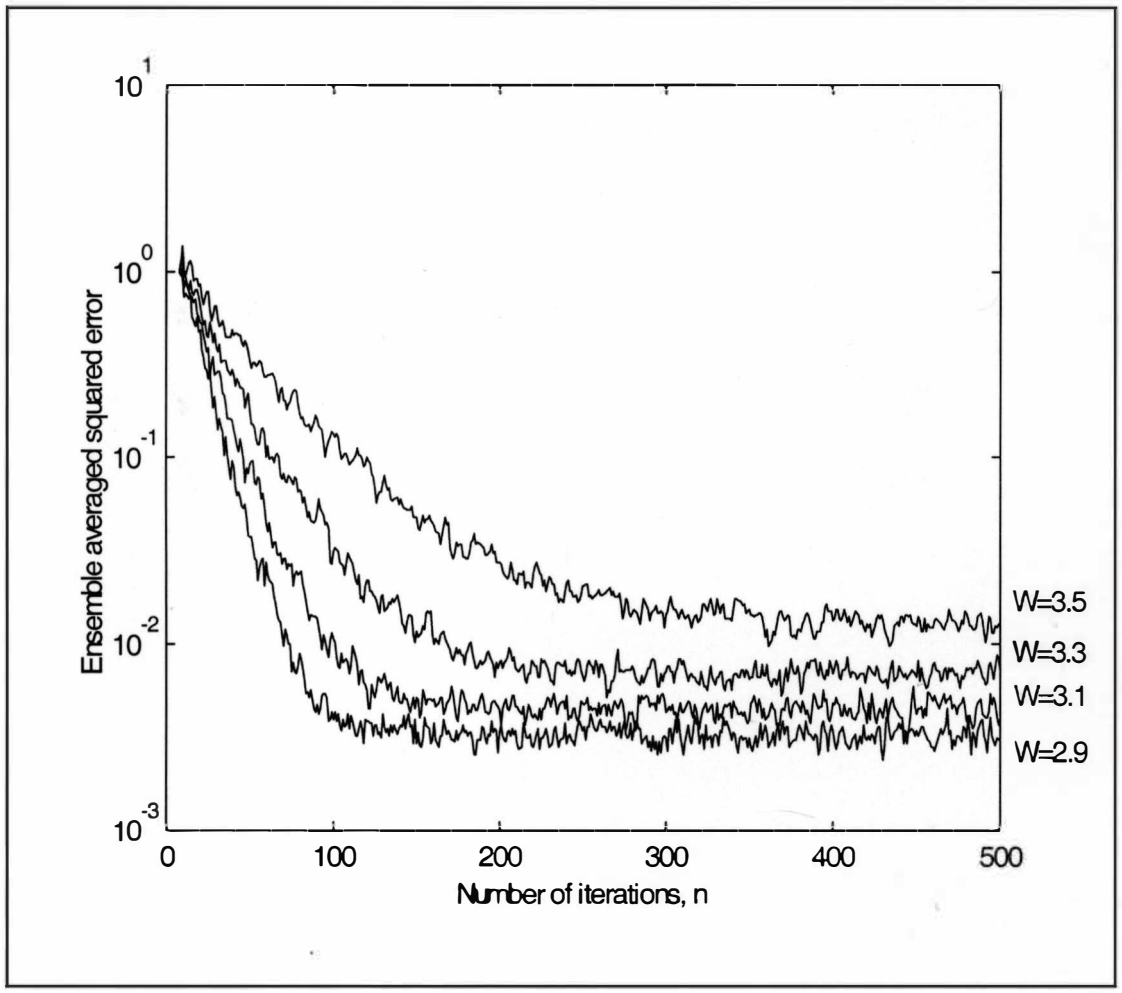

Figure 18. Learning Curve of Normalized LMS Equalizer, Experiment 4.

and the rate of convergence of the NLMS equalizers improve with introducing a time varying step size parameter in the equalizer algorithm.

Experiment 5: Effect of Signal to Noise Ratio. Experiment 4 was carried out with a white noise input to the channel having σ_v^2=0.001. As we see in Figure 19, by decreasing the signal to noise ratio, the convergence of the algorithm speeds up, while, the final steady state averaged squared error of the adaptation process increases.

4.6.3 Recursive Least Square (RLS) Equalizer

Experiment 6: Effect of Eigenvalue Spread. For each eigenvalue spread, an approximation to the ensemble-averaged learning curve of the adaptive transversal equalizer is obtained by averaging the instantaneous squared error versus n over 200 independent trials. In this experiment the signal to noise ratio of 30 dB(σ_v^2=0.001) is assumed. The forgetting factor, λ, which should take values less than or equal to unity is assumed to be equal to 1 in this experiment, meaning that the equalizer has an infinite memory. The result of this experiment is shown in Figure 20. As we see in Figure 20, the rate of convergence of RLS equalizer is almost unaffected by the eigenvalue spread of the correlation matrix of the input to the equalizer and is at least three to five times faster than LMS and NLMS equalizers. The adaptation process of RLS equalizer is less noisy compared with LMS equalizer and its final steady state averaged squared error is considerably lower than LMS and NLMS equalizers for severely corrupted input to the equalizer.

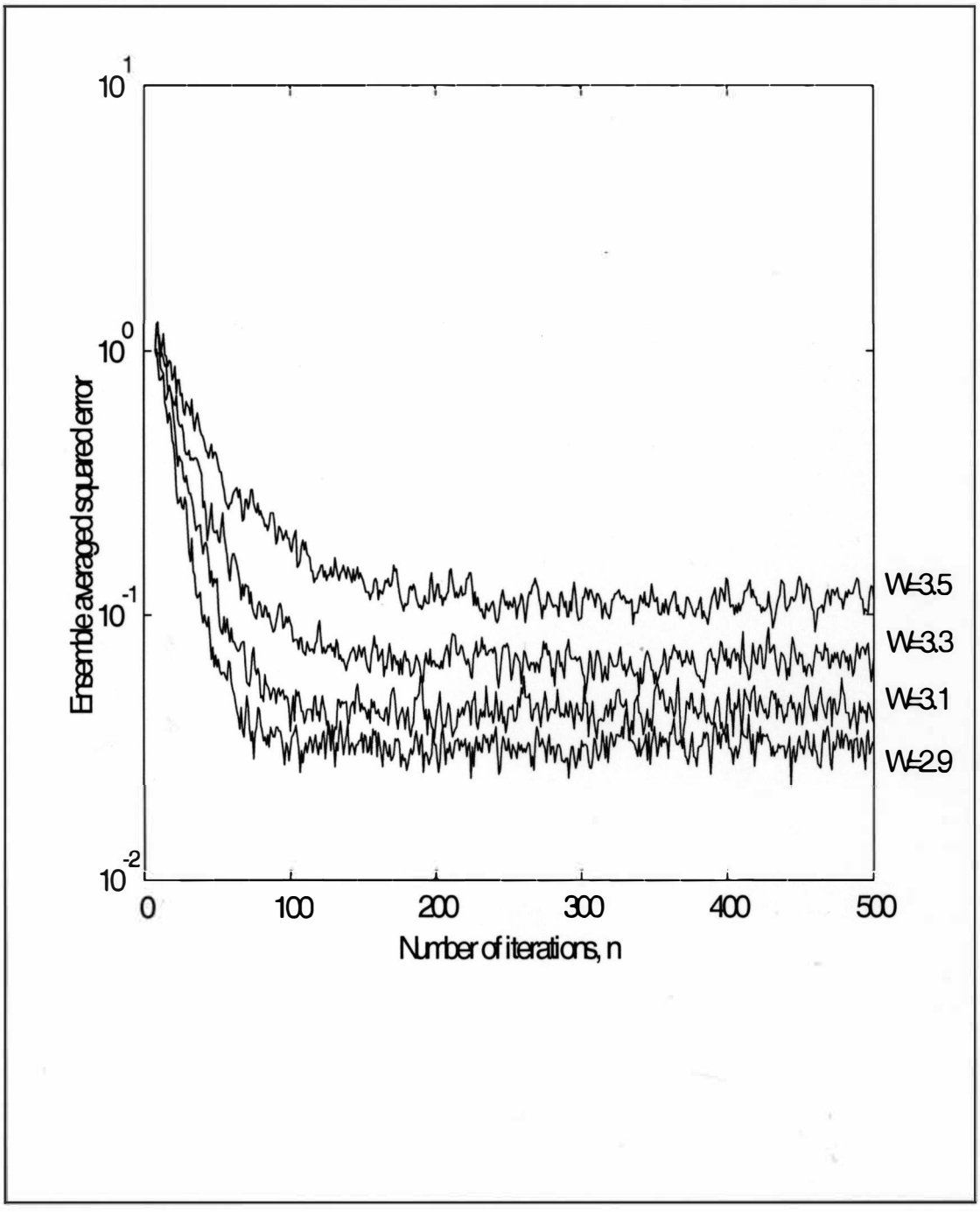

Figure 19. Learning Curve of Normalized LMS Equalizer, Experiment 5.

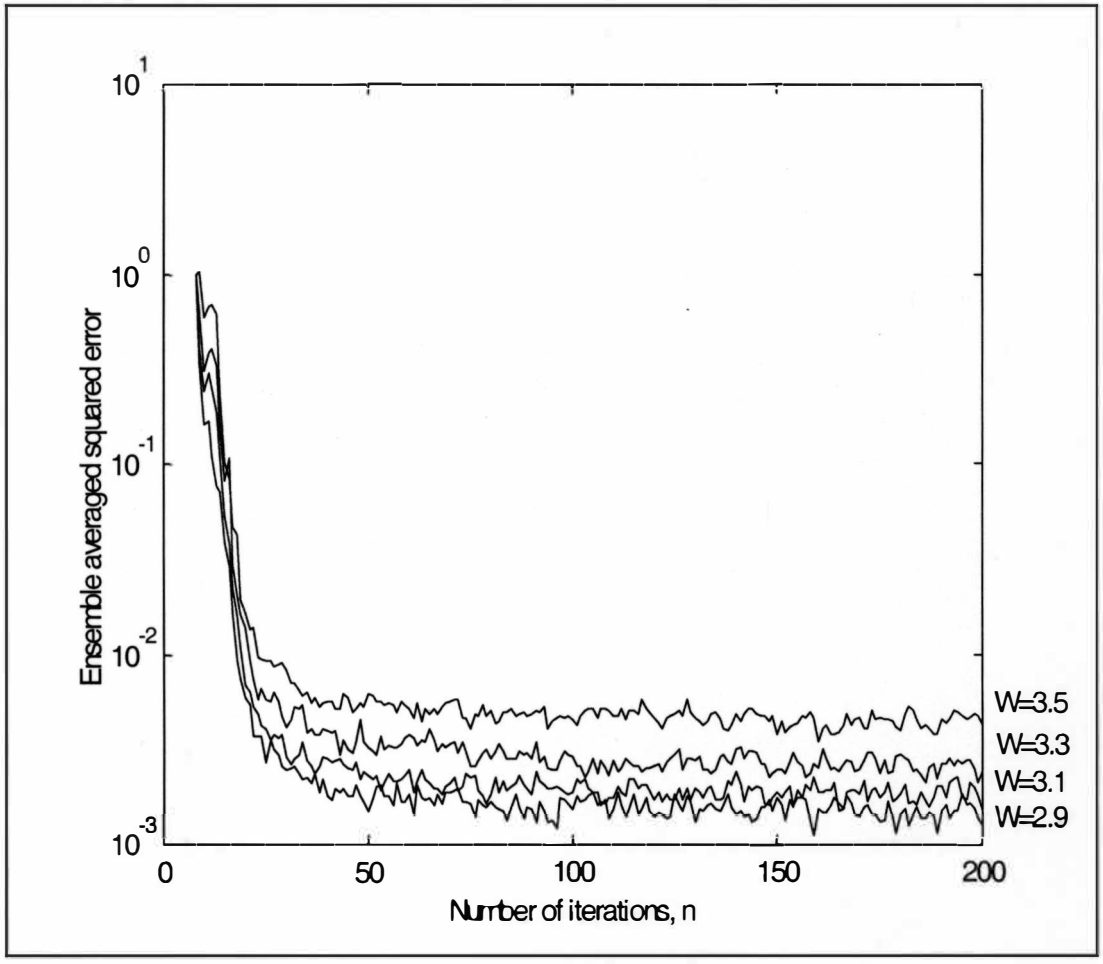

Figure 20. Learning Curve of RLS Equalizer, Experiment 6.

Experiment 7: Effect of Signal to Noise Ratio. In this experiment all the parameters are the same as the experiment 6 except that the signal to noise ratio is reduced to 20 dB (σ_v^2=0.01). The results of this experiment are shown in Figure 21. As we see in this Figure, The reduced signal to noise ratio dose not affect the rate of convergence of the algorithm. This is mainly because the RLS equalizer in experiment 6 was seen to be insensitive to the eigenvalue spread of the correlation

matrix. However, the steady state averaged squared error increases due to the reduced signal to noise ratio.

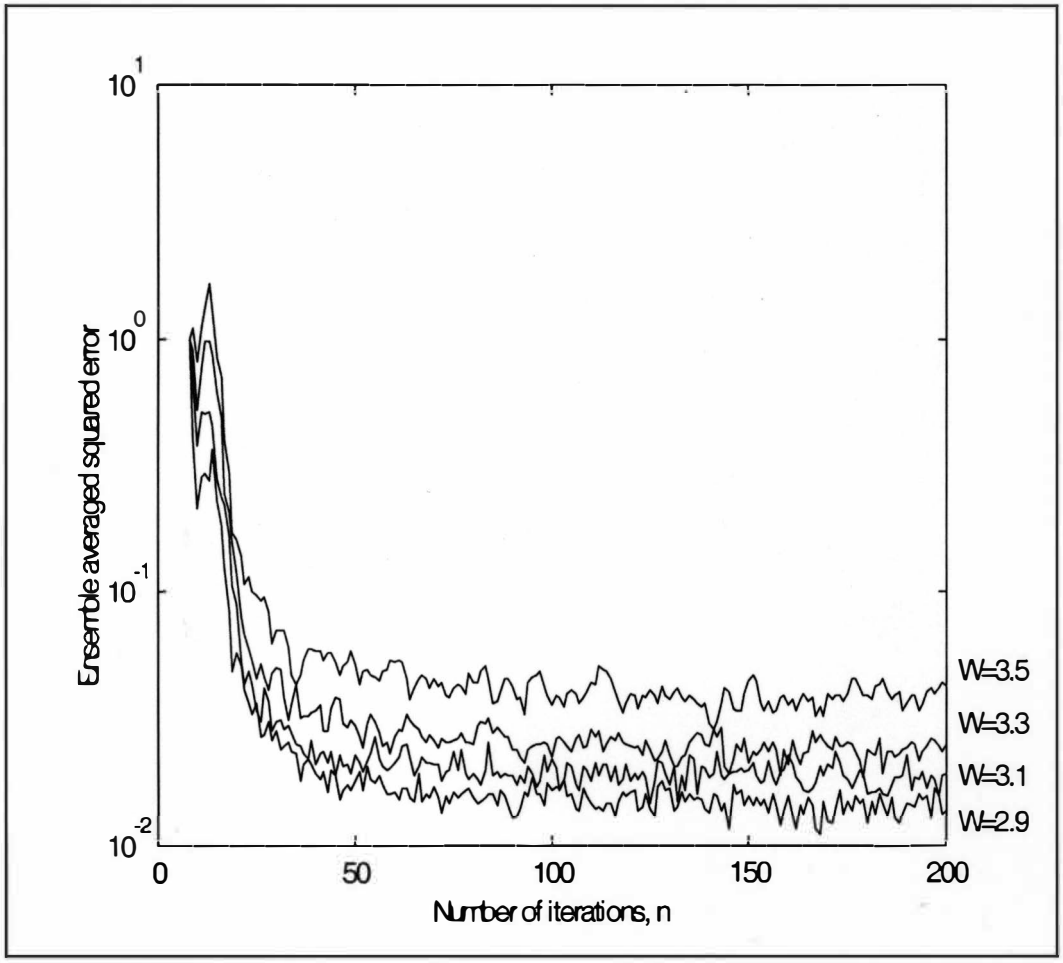

Figure 21. Learning Curve of RLS Equalizer, Experiment 7.

Experiment 8: Effect of the Forgetting Factor, λ. It was pointed out in Chapter III that; λ represents the memory of the algorithm and it takes values less than or equal to unity. In this experiment, λ is set equal to 0.8, while all other

parameters remain the same as in experiment 6. We see in Figure 22, rate of convergence of the algorithm decreases to about half the rate of convergence for $\lambda=1$, while the steady state averaged squared error increases by a factor of two to three . We also see that the adaptation noise of the algorithm increases by reducing the λ. The forgetting factor λ, plays an important role in the tracking performance of least square equalizers, just in the same way as the step size parameter in the least mean square equalizers. Depending on a specific case, a trade off may be made between the rate of convergence and the steady state averaged squared error of the algorithm.

4.6.4 Recursive LSL Equalizer (RLSL)

Experiment 9: Effect of eigenvalue spread. For each eigenvalue spread, an approximation to the ensemble-averaged learning curve of the adaptive transversal equalizer is obtained by averaging the instantaneous squared error versus *n* over 200 independent trials. In this experiment the signal to noise ratio of 30 dB($\sigma_v^2=0.001$) is assumed. The forgetting factor, λ, which should take values less than or equal to unity is assumed to be equal to 1 in this experiment, meaning that the equalizer has an infinite memory. The result of this experiment is shown in Figure 23. As we see in Figure 23, the rate of convergence of RLSL equalizers is insensitive to the eigenvalue spread of the correlation matrix of the input to the equalizer. In many ways RLSL equalizers perform in much the same way as the RLS equalizers. However, the computational complexity of RLSL equalizers is a linear function of *M,* the order of the equalizer, whereas for RLS equalizer it is a quadratic function of *M*. The effect of

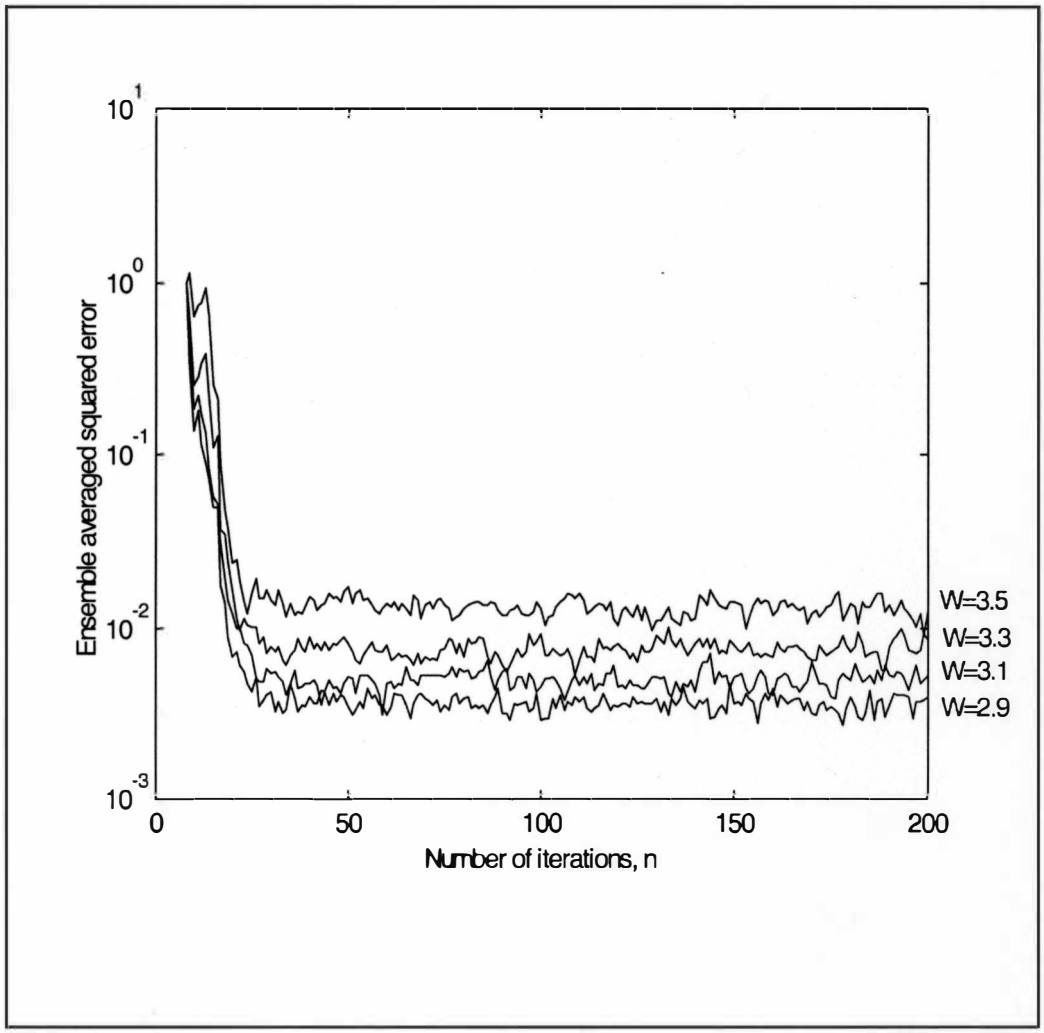

Figure 22. Learning Curve of RLS Equalizer, Experiment 8.

signal to noise ratio and the forgetting factor λ, in RLSL equalizers is the same as that in the RLS equalizer. The results of setting σ_v^2=0.01 and λ=0.8 in RLSL equalizer are shown in Figures 24 and 25 respectively. As we notice, the lower forgetting factor λ, has the effect of faster convergence in the RLSL equalizer. However, the final steady state averaged squared error increases considerably.

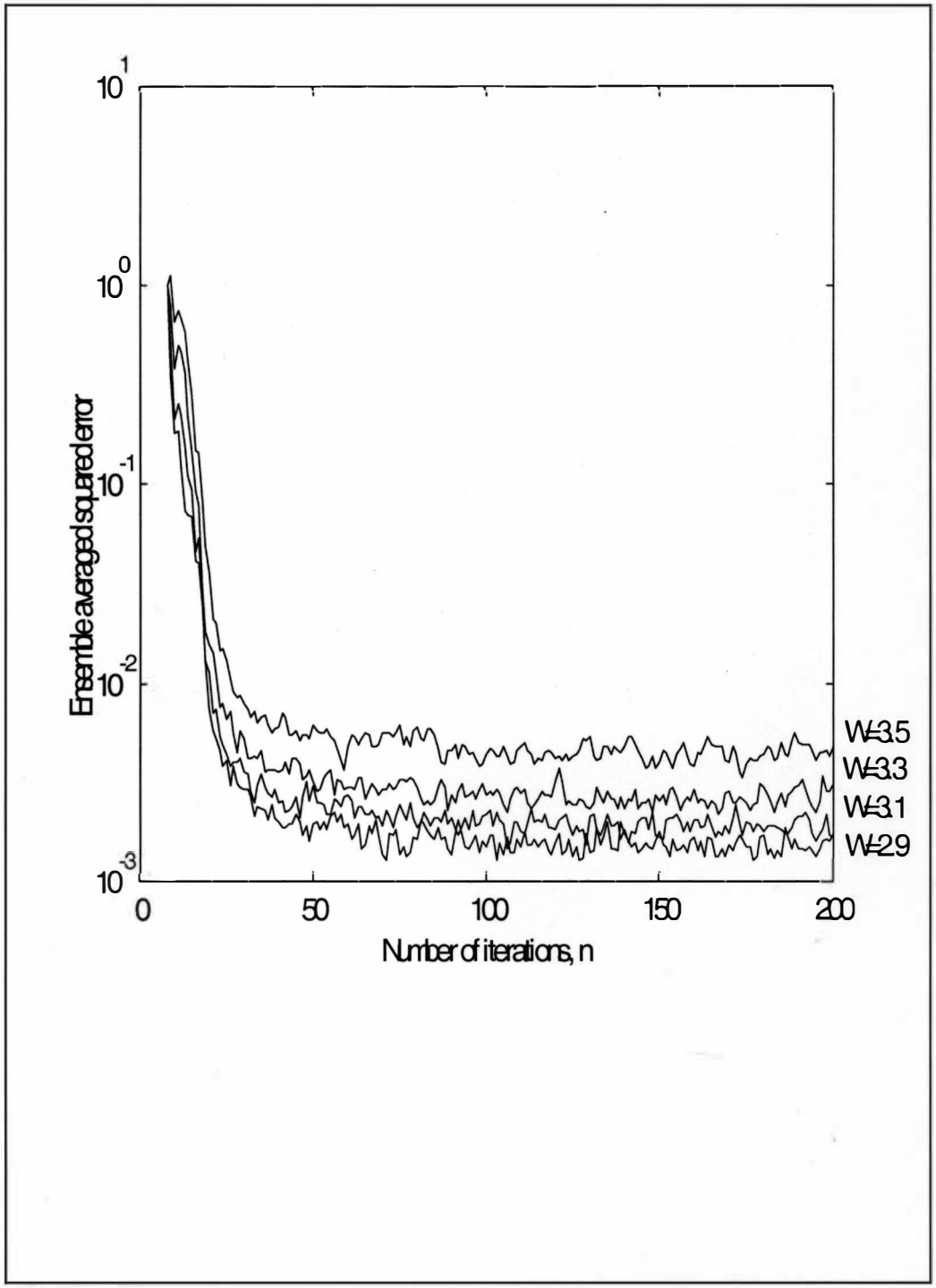

Figure 23. Learning Curve of RLSL Equalizer, Experiment 9, λ=1, σ_v^2 =.001 .

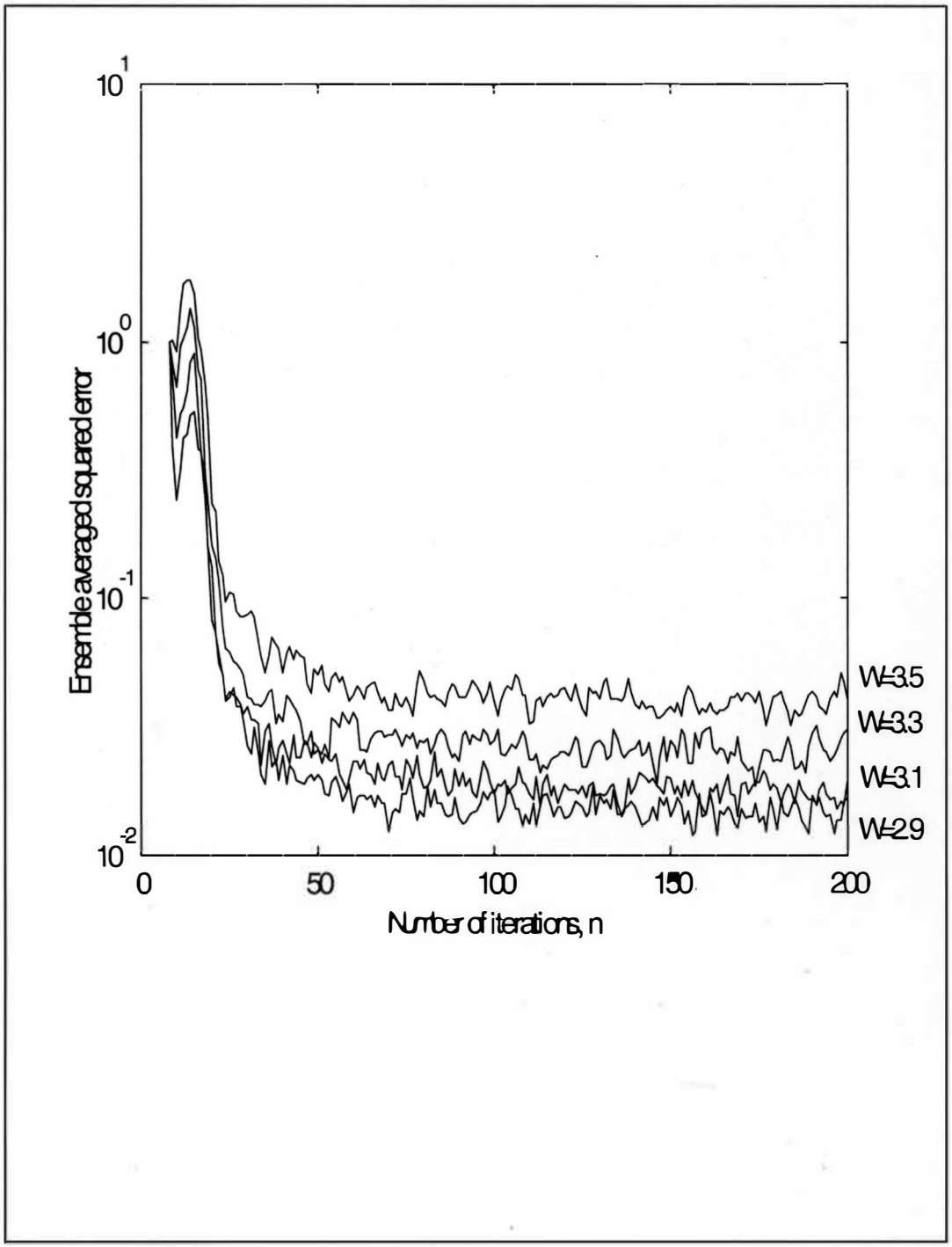

Figure 24. Learning Curve of RLSL Equalizer, Experiment 9, λ=1, σ_v^2 =.01.

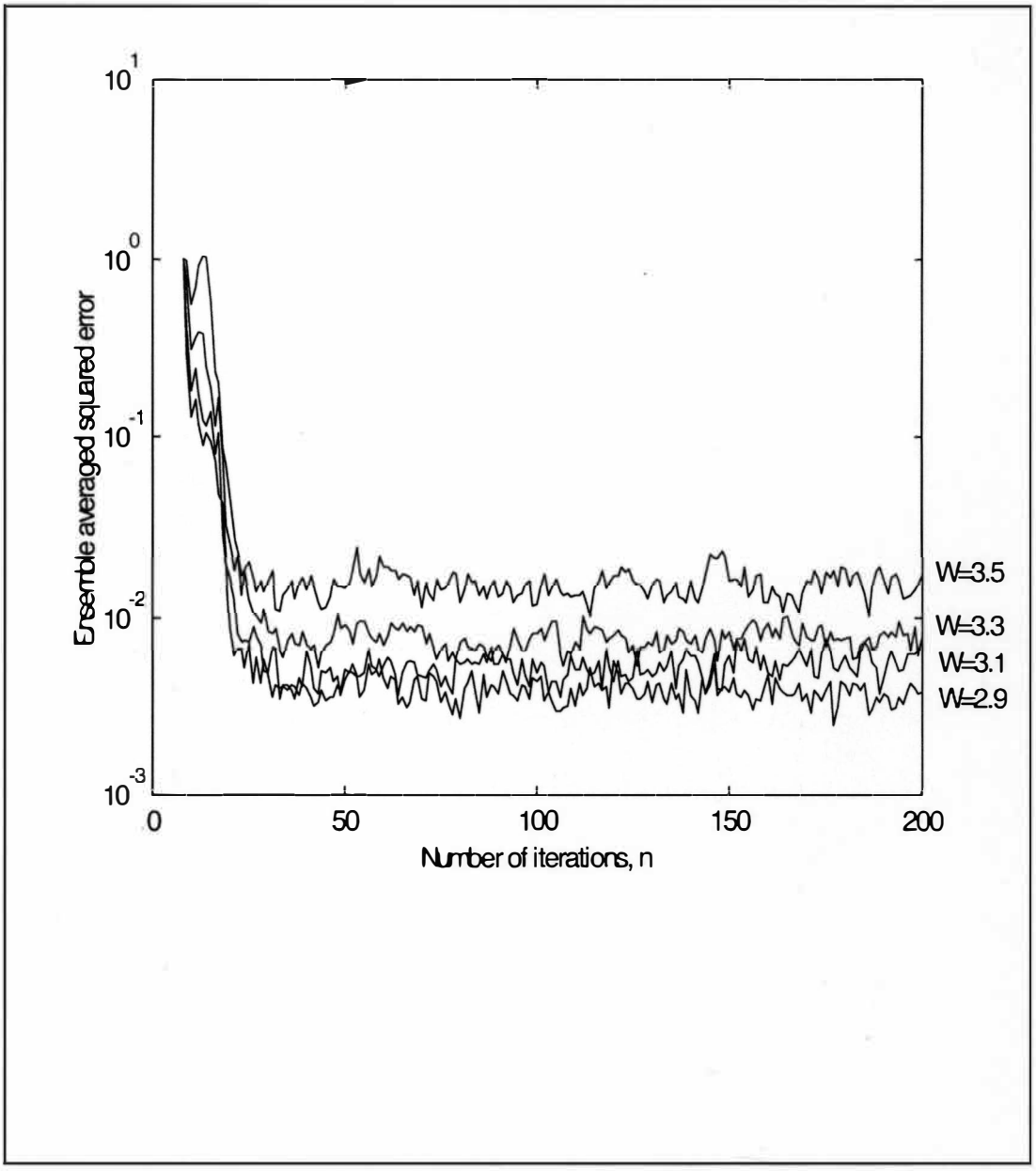

Figure 25. Learning Curve of RLSL Equalizer, Experiment 9, λ=0.8, σ_v^2 =.001 .

CHAPTER V

CONCLUSIONS COMMENTS

This chapter presents conclusions and comments on the simulation results of the linear adaptive channel equalizers, in accordance with the performance factors of adaptive algorithms presented in section 5 of Chapter II.

5.1 Rate of Convergence

As we see in Figures 26, 27, 28, and 29, the RLS and RLSL equalizers converge at least three to five times faster than LMS and Normalized LMS algorithms. RLS and RLSL algorithms are seen to be insensitive to the eigenvalue spread $\chi(\mathbf{R})$, of the correlation matrix of the equalizer input. The LMS equalizers, on the other hand, are quite sensitive to $\chi(\mathbf{R})$ and they hardly converge when the correlation matrix is ill conditioned, $W \geq 3.3$. The convergence rate of the LMS equalizer is mainly determined by the value of the fixed step size parameter in updating the equalizer coefficients. By introducing a time varying step size in the LMS equalizer, the resulting Normalized LMS equalizer converges somewhat faster than LMS equalizer and becomes less sensitive to the condition number of the correlation matrix. However, a choice over different channel equalizers is not solely governed by their rate of convergence, but other crucial factors such as computational

complexity, misadjustment and tracking capabilities of the equalizers must also be considered.

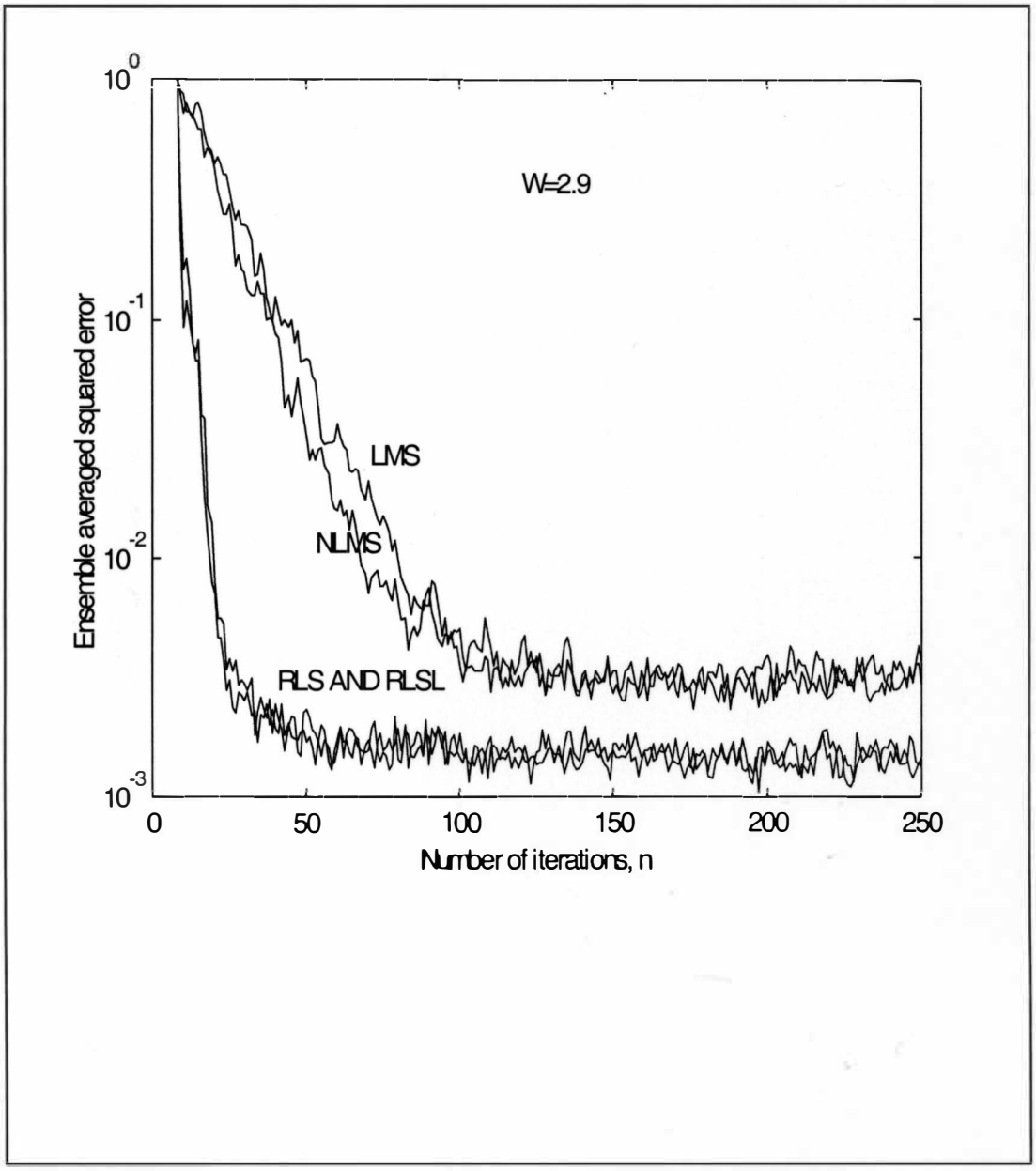

Figure 26. Learning Curves of LMS, NLMS, RLS, and RLSL Equalizers for W=2.9.

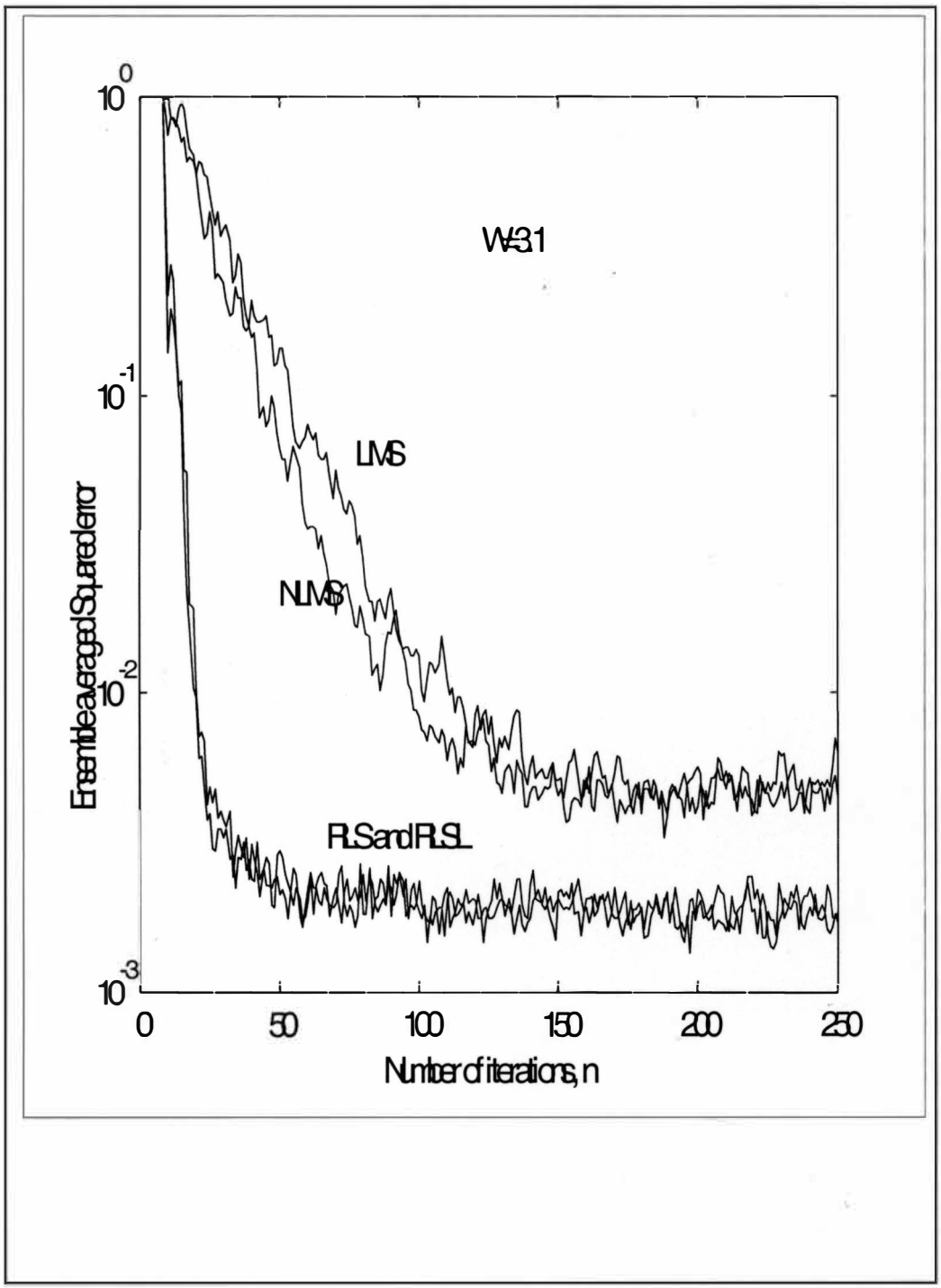

Figure 27. Learning Curves of LMS, NLMS, RLS, and RLSL Equalizers for W=3.1.

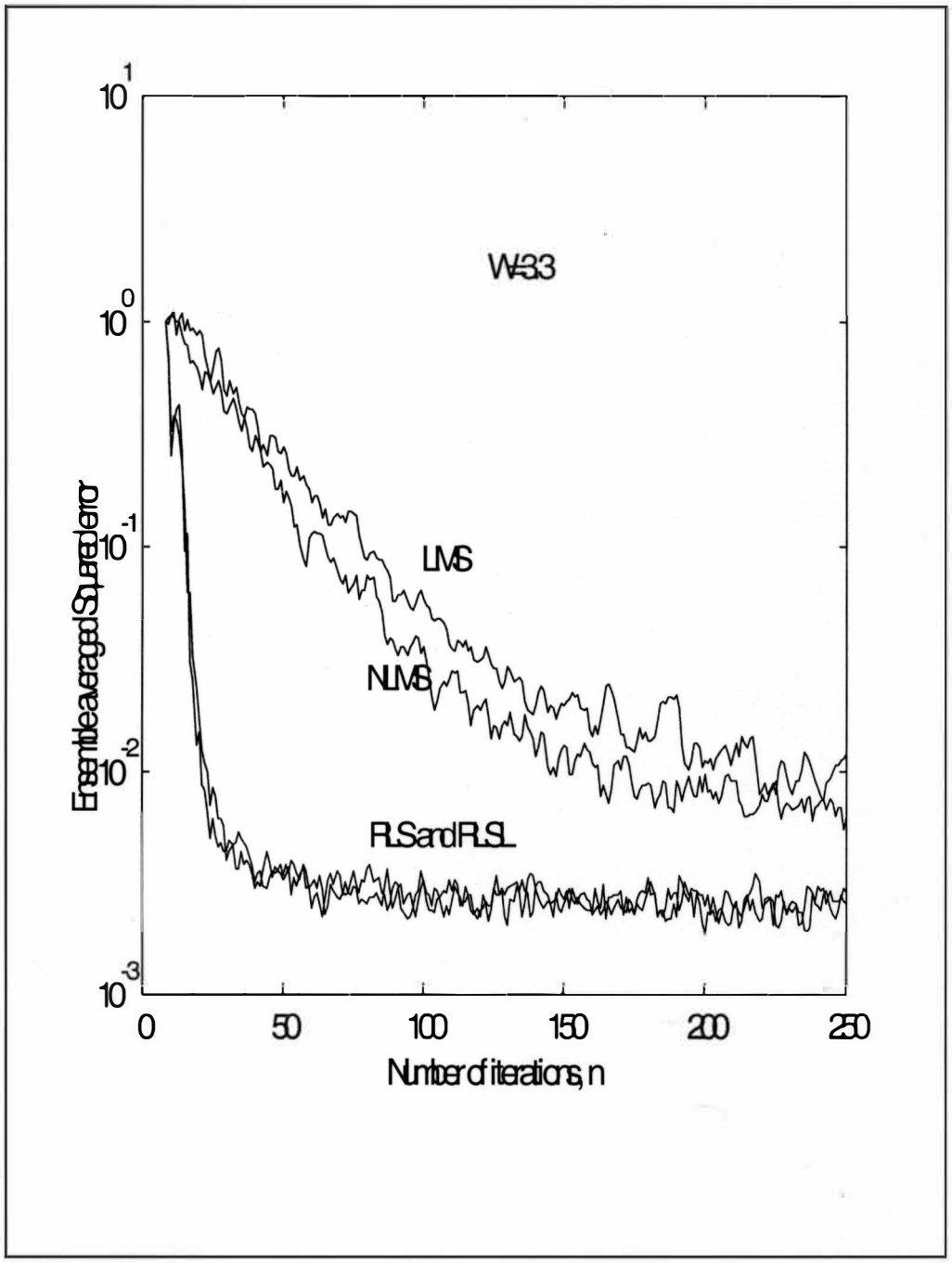

Figure 28. Learning Curves of LMS, NLMS, RLS, and RLSL Equalizers for W=3.3.

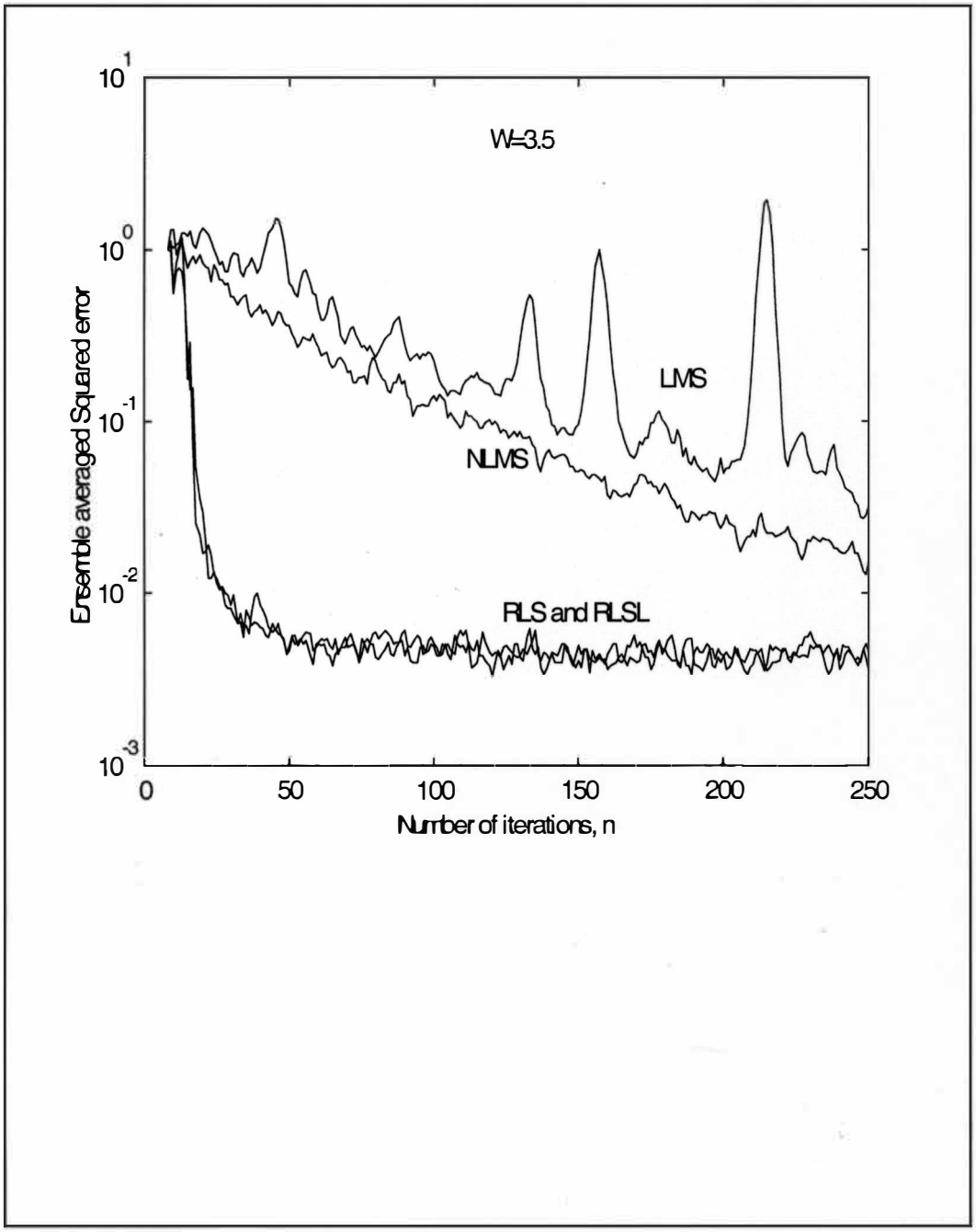

Figure 29. Learning Curves of LMS, NLMS, RLS, and RLSL Equalizers for W=3.5.

5.2 Misadjustment

For an algorithm of interest, this parameter provides a quantitative measure of the amount by which the steady state value of the averaged square error of adaptive equalizer, deviates from the minimum squared error that is produced by the Wiener filter. In Table 3, the steady state averaged squared errors of LMS, NLMS, RLS, and RLSL adaptive equalizers are compared with that obtained by the Wiener filter. The steady state averaged squared error of LMS equalizer is affected by the fixed step size parameter, μ, and that of RLS and RLSL equalizers depend to some degree on the forgetting factor λ. Here μ is assumed to be 0.075, and λ equal to unity. The final steady state averaged squared errors are measured after 250 iterations of the adaptive equalization process. The LMS and NLMS equalizers do not converge for up to 250 iteration, when W =3.5, therefore their misadjusment is not included in Table 3.

Table 3

Misadjustment of Adaptive Equalizers

$\chi(\mathbf{R})$	Wiener filter	LMS	NLMS	RLS	RLSL
6.0782	0.0013755	135%	157%	14%	14%
11.1238	0.0017524	211%	194%	17%	17%
21.7132	0.0024739	312%	92%	4%	3%
46.8216	0.0041559	-	-	11%	9.5%

5.3 Tracking

At first glance, one might conclude that since the RLS and RLSL adaptive equalizers have a faster rate of convergence than the LMS and NLMS adaptive equalizers in the case of a stationary environment; therefore, it will track a nonstationary environment better than the LMS and NLMS equalizers. However, such a conclusion is not justified because the tracking performance of an adaptive filtering algorithm is influenced not only by the rate of convergence but also by fluctuations in the steady state performance of the algorithm due to algorithm noise. With both algorithms tuned to minimize the misadjustment at the equalizer output by a proper optimization of their forgetting factor λ for RLS and RLSL equalizers and the step size parameter μ for the LMS equalizer, the LMS algorithm is found to have a superior tracking performance compared to RLS and RLSL equalizers [2,7,8,14].

5.4 Robustness

Robustness refers to the ability of the adaptive equalizer to operate satisfactorily with ill-conditioned input data. It has been shown throughout the adaptive equalization experiments and we also see in Figures 26 through 29, that the RLS and RLSL equalizers are insensitive to the eigenvalue spread of the correlation matrix of the equalizer input, therefore operate quite satisfactorily almost regardless of the channel input condition. On the contrary, the LMS equalizer is sensitive to the channel input condition and hardly converges when eigenvalue spread of the underlying correlation matrix is large. The Normalized LMS equalizer, however, is

relatively robust for large eigenvalue spread of correlation matrix, though, it converges much slower than RLS and RLSL equalizers.

5.5 Computational Complexity

At each iteration the number of multiplications, divisions, and additions/subtractions required to make one complete iteration of LMS, RLS and RLSL algorithms are found to be function of the order of the equalizer filter:

LMS	$O(\mathbf{M})$	Linear function of M
NLMS	$O(\mathbf{M})$	Linear functiom of M
RLS	$O(M^2)$	Quadratic function of M
RLSL	$O(\mathbf{M})$	Linear function of M

5.6 Comments

RLSL equalizer has a low convergence rate, low computational complexity, and low misadjusment percentage compared with the other equalizers and therefore is very suitable for applications where these three factors combined are of utmost importance. However, this type of equalizer has a relatively complicated structure and is more expensive in terms of its implementation as compared to the LMS and RLS equalizers. The RLS equalizer has a high computational complexity and despite its fast convergence, and low misadjustment, can hardly be justified for channel equalization where the order of the the equalizer filter has to be relatively high.

However in wireless communications where, the number of smeared symbols are about 10, the equalizer will be of an order of 10, therefore the high computational complexity of the RLS algorithm may be compensated with its fast convergence rate. Hence, making RLS algorithm quite suitable as a choice of equalizer. The LMS and Normalized LMS equalizers have low computational complexities and have simple structures, that have made them quite popular thus far, despite their sensitivity to the channel input condition and their relatively high misadjutment.

With increasing availability of low cost, and highly efficient DSP chips in the market, Recursive Least Square Lattice (RLSL) equalizer may be more frequently used in channel equalization applications because of its superiority over other adaptive equalizers. The modular structure of this type of equalizer in particular, enhances the flexibility of the equalizer in terms of future modifications in order to meet the new system requirements.

REFERENCES

[1] J. G. Proakis, "Digital Communications," McGraw-Hill Inc. Third Edition, 1995.

[2] S. Haykin, "Adaptive Filter Theory," Prentice Hall, Third Edition , 1995.

[3] Jerry D. Gibson "The Communications Handbook," CRC PRESS, 1996.

[4] S. Qureshi, "Adaptive Equalization," Proceedings of the IEEE, Vol. 73, pp. 1349-1357, Sept. 1985.

[5] E.H. Satorius and S.T. Alexander, "Channel Equalization Using Adaptive Lattice Algoritms," IEEE Trans. on Comm., Vol. COM-27, No.6, June 1979.

[6] S. Qureshi, "Adaptive Equalization," IEEE Communication Magazine, Vol. 20, pp. 9-16, Mar. 1982.

[7] M.S Muller, "Least Squares algorithms for Adaptive Equalizer", Bell System Tech. J., Vol. 60, pp. 1905-1925, 1981.

[8] E. H. Satorius and S. T. Alexander, "Channel Equalization Using Adaptive Lattice Algorithms," IEEE transaction on Communication Vol. COM-27 pp, 899-905, June 1979.

[9] E. H. Satorius and J. Pack, "Application of Least Squares Lattice Algorithm to adaptive Equalization," IEEE Transaction on Communication, Vol. COM-29, pp.136-142, Feb. 1981.

[10] P. M. Grant, and M. J. Rutter, "Application of Gradient Adaptive Lattice Filter to Channel Equalization," IEEE Proceeding, Vol. 131, pp. 437-479, Aug., 1984.

[11] S. K. Mitra, and J. F. Kaiser, "Handbook for Digital Signal Processing," John Wiley & Sons, 1993.

[12] Theodore S. Rappaport, "Wireless Communications", IEEE Press, 1996

[13] D. Falconer and E. Eleftherion, "Tracking properties and steady state performance of RLS adaptive filter algorithms," IEEE Trans. Acoust. Speech,

and Signal Processing, vol. ASSP-34, no.5, Oct. 1988.

[14] J.M. Ciffi and T. Kailath, "Tracking properties and steady state performance of RLS adaptive filter algorithms," IEEE Trans. Acoust. Speech, and Signal Processing, vol. ASSP-32, no.5, Oct. 1984.