NEW MEXICO STATE UNIVERSITY

THE KLIPSCH SCHOOL OF

ELECTRICAL AND COMPUTER ENGINEERING

TECHNICAL REPORT SERIES

EQUALIZATION AND DETECTION FOR DIGITAL COMMUNICATION OVER NONLINEAR BANDLIMITED SATELLITE COMMUNICATION CHANNELS

Alberto Gutierrez, Jr.
Dr. William Ryan

NMSU-ECE-95-008 December 1995

NAG5-1491

EQUALIZATION AND DETECTION FOR DIGITAL COMMUNICATION OVER NONLINEAR BANDLIMITED SATELLITE COMMUNICATION CHANNELS

BY

ALBERTO GUTIERREZ, JR.

A Dissertation submitted to the Graduate School

in partial fulfillment of the requirements

for the Degree

Doctor of Philosophy, Engineering

Specialization in: Electrical Engineering

New Mexico State University

Las Cruces, New Mexico

December 1995

"Equalization and Detection for Digital Communication over Nonlinear Bandlimited Satellite Communication Channels," a dissertation prepared by Alberto Gutierrez, Jr. in partial fulfillment of the requirements for the degree, Doctor of Philosophy, Engineering, has been approved and accepted by the following:

Timothy J. Pettibone
Dean of Graduate School

William E. Ryan
Chair of the Examining Committee

Date

Committee in charge:

Dr. William E. Ryan, Chair

Dr. Mai Gehrke

Dr. Sheila B. Horan

Dr. Jay B. Jordan

Dr. William P. Osborne

DEDICATION

To Liz and Jenny for their love and patience.

ACKNOWLEDGMENTS

I am grateful to my advisor William E. Ryan whose commitment to excellence has greatly enriched my experience at NMSU. I am especially grateful to Dr. William P. Osborne for sharing his experience and love for the field of telecommunications. I am also grateful to Dr. Mai Gehrke, Dr. Sheila Horan, and Dr. Jay Jordan for their guidance as members of my dissertation committee.

I extend my appreciation to the NASA Goddard GSRP program and staff for their dedication to the higher education of research scientist and engineers. The financial support and honor of the GSRP fellowship is gratefully acknowledged. I acknowledge with thanks Warner Miller my NASA technical advisor for his patience and support.

I am forever indebted to my parents for their love and encouragement. Finally, I thank my wife Liz and daughter Jenny without whose love and encouragement this work would not have been possible.

VITA

Dec. 1994	Bachelor of Science Electrical Engineering University of Texas at El Paso
Dec. 1986	Master of Science Electrical Engineering Purdue University
Jan. 1987 - July 1992	Member of Technical Staff Hewlett Packard, Co. Fort Collins, Colorado
Aug. 1992 - Sep. 1995	Graduate Research Fellow New Mexico State University Las Cruces, New Mexico

Publications

A. Gutierrez, "PAM a Personal Arrhythmia Monitor," M.S.E.E. thesis, Dept. of Electrical and Computer Engineering, Purdue University, 1986

A. Gutierrez and M. Hassoun, "32-Bit Pipelined Floating Point Multiplier," *Hewlett Packard Design Technology Conference*, 1992.

W. P. Osborne and A. Gutierrez, "Sub-Optimum Receiver Filters," *International Telemetry Conference*, 1994.

A. Gutierrez and W. E. Ryan, "Performance of Adaptive Volterra Equalizers on Nonlinear Satellite Channels," *International Conference on Communications*, 1995.

W. E. Ryan and A. Gutierrez, "Performance of Adaptive Volterra Equalizers on Nonlinear Recording Channels," *IEEE Transactions on Magnetics*, September 1995.

ABSTRACT

EQUALIZATION AND DETECTION FOR DIGITAL COMMUNICATION OVER NONLINEAR BANDLIMITED SATELLITE COMMUNICATION CHANNELS

BY

ALBERTO GUTIERREZ, JR.

Doctor of Philosophy, Engineering

Specialization in Electrical Engineering

New Mexico State University

Las Cruces, New Mexico, 1995

Dr. William E. Ryan, Chair

This dissertation evaluates receiver-based methods for mitigating the effects due to nonlinear bandlimited signal distortion present in high data rate satellite channels. The effects of the nonlinear bandlimited distortion is illustrated for digitally modulated signals. A lucid development of the low-pass Volterra discrete-time model for a nonlinear communication channel is presented. In addition, finite-state machine models are explicitly developed for a nonlinear bandlimited satellite channel.

A nonlinear fixed equalizer based on Volterra series has previously been studied for compensation of noiseless signal distortion due to a nonlinear satellite chan-

nel. This dissertation studies adaptive Volterra equalizers on a downlink-limited nonlinear bandlimited satellite channel. We employ as figure of merits performance in the mean-square error and probability of error senses. In addition, a receiver consisting of a fractionally-spaced equalizer (FSE) followed by a Volterra equalizer (FSE-Volterra) is found to give improvement beyond that gained by the Volterra equalizer. Significant probability of error performance improvement is found for multilevel modulation schemes. Also, it is found that probability of error improvement is more significant for modulation schemes, constant amplitude and multilevel, which require higher signal to noise ratios (i.e., higher modulation orders) for reliable operation.

The maximum likelihood sequence detection (MLSD) receiver for a nonlinear satellite channel, a bank of matched filters followed by a Viterbi detector, serves as a probability of error lower bound for the Volterra and FSE-Volterra equalizers. However, this receiver has not been evaluated for a specific satellite channel. In this work, an MLSD receiver is evaluated for a specific downlink-limited satellite channel. Because of the bank of matched filters, the MLSD receiver may be high in complexity. Consequently, the probability of error performance of a more practical suboptimal MLSD receiver, requiring only a single receive filter, is evaluated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AM	Amplitude modulation
AWGN	Additive white Gaussian noise
CB	Citizen's band
CPFSK	Continuous-phase frequency shift keying
DFE	Decision feedback equalizer
EOS	Earth observing system
FDMA	Frequency division multiple access
FSE	Fractionally-spaced equalizer
FSM	Finite-state machine
HPA	High powered amplifier
IDCSP	Initial defense satellite communication program
ICI	Inter-channel interference
ISI	Intersymbol interference
LEO	Low earth orbit
LNA	Low-noise amplifier
LMS	Least mean square
MFB	Matched-filter bank
MLSD	Maximum-likelihood sequence detector
PM	Phase modulation
PSK	Phase shift keying
QAM	Quadrature amplitude modulation
RF	Radio frequency
SF	Single filter
SNR	Signal-to-noise ratio
TWT	Traveling wave tube

Chapter 1

INTRODUCTION

As the 20th century comes to a close, the field of telecommunications is coming of age with voice, video, and data communication systems operating over copper, cable, fiber, and wireless media. Today it is common for a person to turn on a television, tune to a cable channel, and get a weather report at any time of the day based on video obtained from weather satellites. University students, industry personnel, and computer enthusiasts needing information on practically any subject matter can connect to the Internet and download images, programs, and data for their research or enjoyment. The medical community commonly monitors patients and receives vital information remotely via medical telemetry. Law enforcement officers often have CB radios, pagers, computers connected to a central data base, and cellular telephones operating simultaneously from their mobile squad cars. Because of this recent "information explosion," communication systems are being pushed to their capacity. In order to meet these demands, it is essential to design communication systems which make the most efficient use of the precious bandwidth resource. Meeting these demands coupled with the mature theory of modern communication systems makes it an exciting time to be working on almost any aspect of communication systems.

In particular, satellites provide unique capabilities not available from other forms of communication systems. First, is the capability of global coverage for commercial communications use such as in the INTELSAT satellites [1], remote

sensing as in EOS (earth observing system) satellites [2], and surveillance as in IDCSP (Initial Defense Satellite Communication Program) satellites [1]. Second, satellites are capable of providing bandwidth second only to fiber. Currently, satellite systems are being designed to support data rates in the tens of gigabits per second. In order to provide sufficient link margin, satellite channels employ a high power amplifier (HPA) often in the form of a traveling wave tube (TWT).[1] However, the increasing demand for bandwidth and the desire to minimize satellite power consumption often means the TWT is driven at or near saturation. The end result is the introduction of nonlinear bandlimited signal distortion yielding nonlinear ISI (intersymbol interference).

This dissertation evaluates receiver-based methods for mitigating the effects due to nonlinear bandlimited signal distortion. Specifically, Volterra equalizers, FSE-Volterra equalizers, maximum likelihood sequence detection (MLSD), and suboptimal MLSD receivers are evaluated. The results of this dissertation will serve as a baseline for the evaluation of more complex structures based on these. In addition, these results will help gauge the performance of hardware implementations of these structures.

The following section presents a simplified communication subsystem for a typical communications satellite. The communication subsystem is then reduced

[1]Recently solid state amplifiers have become available which will likely replace the TWT as the HPA in new satellites. Since many existing satellites use TWT amplifiers, in this work TWT amplifiers will be considered. However, the equalization and detection methods discussed will be equally valid for solid state amplifiers.

to a model suitable for evaluating the performance of a satellite communication channel. Next, the effects of the nonlinearity and satellite bandlimiting filters on digitally modulated signals is illustrated. A literature review of existing compensation techniques for mitigating the effects of nonlinear bandlimited signal distortion in various types of communication systems is then presented. The last section gives an overview of the chapters in this dissertation.

1.1 Satellite Communication Background

The idea of satellite communication systems was introduced by Arthur C. Clarke in his famous paper, published in 1945, entitled "Extra Terrestrial Relays" [3]. Although this paper was generally regarded as science fiction [4], within 25 years Clarke's ideas materialized into a mature technology. Satellites were first placed in orbit in the late 1950's a few hundred kilometers above the earth. These satellites were known as LEO (low earth orbit) satellites and have continued to be used for remote sensing applications. However, GEO (geostationary equatorial orbit) satellites have been preferred for commercial communications. Placing a GEO satellite in orbit (approximately 22,000 miles above the earth) has been preferred to the expensive tracking and control systems required for LEO satellites. However, by the 1990's, the availability of powerful computing and signal processing devices has made LEO satellites attractive for providing communication services [4]. In this work we will not consider the special issues of tracking

and synchronization presented by LEO satellites, however the methods developed herein may very well be useful for the LEO scenario.

1.1.1 A Satellite's Communication Subsystem

A simplified block diagram of a communication and antenna subsystem for a typical "bent pipe" (transparent repeater) FDMA (frequency division multiple access) satellite is depicted in Fig. 1.1, [4, 5]. Each antenna may operate in both a transmit and receive mode. A diplexer separates the received antenna energy from the transmit energy. The received energy is routed to the satellite input filter which limits the uplink noise into the satellite. The output of the input filter then enters the receiver which consists of an LNA (low noise amplifier), a downconverter (e.g., converts from 6 to 4 gigahertz), and a receiver output filter which removes unwanted frequency components due to the downconversion operation. The signal then enters input multiplexer filters each of which selects the frequencies entering a particular channel. The signal is then amplified by a TWT amplifier.

After amplification, a transmit beam switch then selects the channels which will form the transmit beam of a particular antenna. The output multiplexer filters restrict the TWT output for each channel to the pre-assigned frequencies. The resulting signal is then sent to the output filter, diplexer, and antenna. The output filter assures that the aggregate signal (including all the satellite channels for a particular antenna) lies within the preassigned satellite bandwidth.

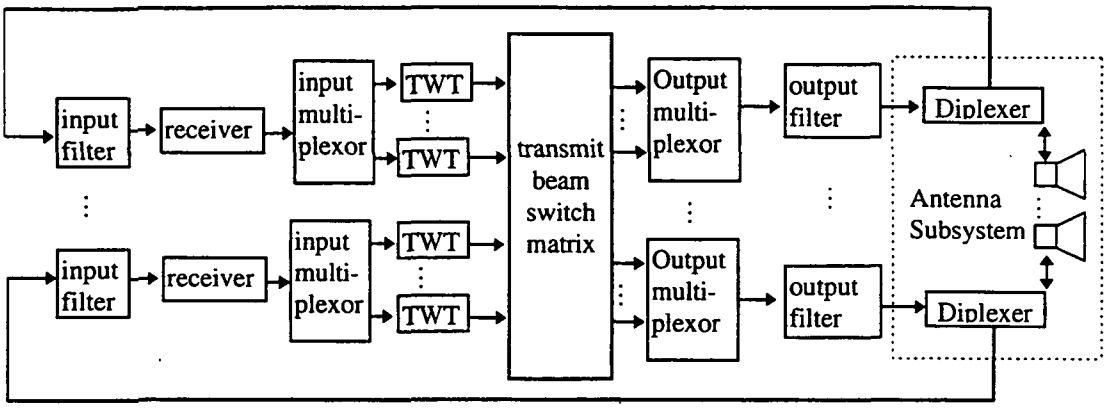

Figure 1.1. Simplified communication subsystem for typical communications satellite.

The bandwidth is typically divided into channels of 36 to 40 MHz, where each channel is handled by a different transponder. A transponder consists of the subset of the communications subsystem responsible for receiving and transmitting a single satellite channel for a bent pipe satellite. For example, a typical 6/4 satellite transponder is depicted in Fig. 1.2. The transponder receives the signal centered at 6 GHz from the receive antenna and subsequently downconverts it to 3.775 GHz (approximately 4 GHz). The RF bandpass filters preceding and following the TWT amplifier represent the input and output multiplexer filters, respectively.

1.1.2 A Satellite Communication System Model

In order to determine the effectiveness of the various equalization and detection methods, it is necessary to reduce the transponder model of Fig. 1.2 to one suitable for evaluation and analysis. In this work the focus is on the communication system performance for an individual user so an individual transponder is

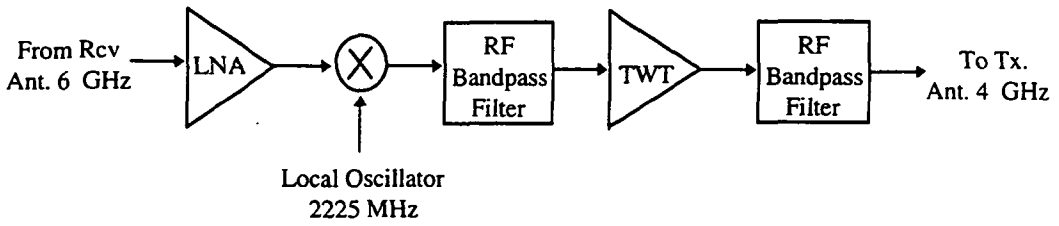

Figure 1.2. Simplified 6/4 transponder.

considered rather than the entire communication subsystem. Only the non-ideal effects due to the nonlinear bandlimiting are considered. Other affects such as ICI (inter-channel interference) are not considered.

Fig. 1.3 represents a low-pass equivalent block diagram of a single-hop transponder communications link which accounts for the dominant performance-limiting components. A single-hop satellite link consists of an uplink from a transmitting earth station to the satellite and a downlink from the satellite to a receiving earth station. The figure contains transmitter, satellite channel, and receiver. Since the modulators, downconverters, diplexers, and demodulators normally present at the transmitter, satellite, and receiver are assumed ideal, it is not necessary to account for them in the model.

The data-bearing waveform $\sum_n d_n \delta(t - nT)$, where T is the symbol interval, $\delta(t)$ is the Dirac delta function [6], and d_n is complex data, is filtered by the transmit filter $h_T(t)$. The satellite model consists of a pre-filter, $h_{pre}(t)$, a TWT high powered amplifier, and post-filter, $h_{pst}(t)$. As mentioned, the pre- and post-filters represent the input multiplexing and output multiplexing filters. Although in general the signal entering the satellite is subject to thermal noise, only the

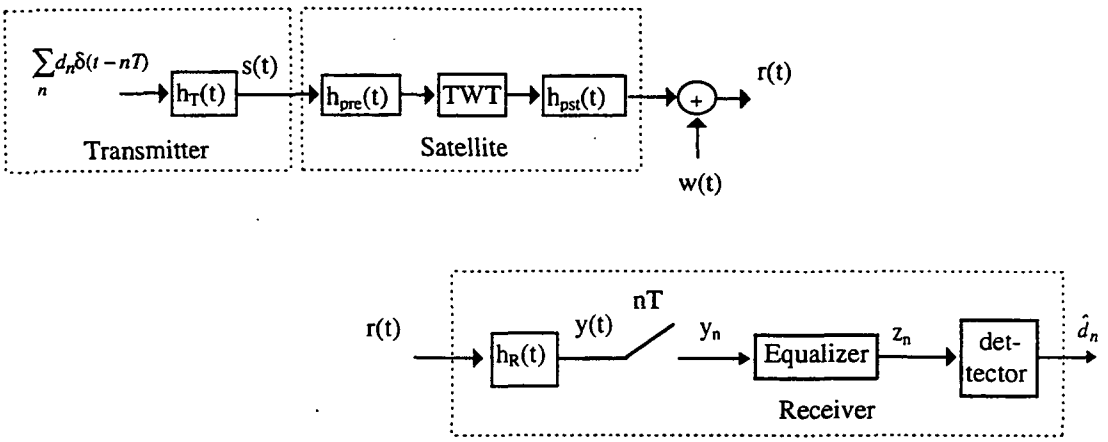

Figure 1.3. Low-pass equivalent communication system model.

noise at the receiver is assumed to be significant. Because the noise at the receiver is dominant, the system is referred to as a downlink-limited satellite system. The output of the receive filter is then sampled, equalized, and detected. Finally, the detector outputs estimates $\hat{d}_n$ of the transmitted symbols d_n.

1.2 Nonlinear Distortion in Satellite Channels

Before discussing the effects of the nonlinear bandlimited satellite channel on digitally modulated signals, a brief discussion of digital modulation formats for satellite communication systems is presented. The combined effect of the TWT nonlinearity and filtering is then illustrated for 8-PSK and 16-QAM systems. Next, a literature review of existing nonlinear distortion compensation methods is presented. The literature review distinguishes between ISI compensation methods based at the transmitter and receiver.

1.2.1 Digital Modulation Formats

A typical modulation format for satellite communications is M-PSK. Because of power limitations and the nonlinear TWT amplifier, bandwidth efficient modulation formats, such as M-QAM, are not commonly employed for satellites. However, 16-QAM has recently been considered for satellite communication [7]. Also, variations to the rectangular 16-QAM signal constellation have recently been considered for satellite communication . In particular (4,12) with an inner circle of 4 signal points and an outer circle of 12 signal points has been considered. In contrast to 16-QAM the (4,12) signal points lie on concentric circles rather than on a square grid [8]. Other more sophisticated modulation methods, such as continuous phase frequency shift keying (CPFSK), will not be considered in this work. However, the equalization and detection techniques studied in this dissertation may also be useful for these modulation methods.

1.2.2 Effects of Nonlinear Distortion

The bandlimited nonlinearity in the transponder results from the pre-filter, TWT, post-filter combination. In this work the TWT is modeled, following Saleh [9], as a frequency-independent memoryless bandpass function. It is completely characterized by its AM/AM and AM/PM conversions given by

$$A(r) = \frac{\alpha_a r}{1 + \beta_a r^2}, \text{ (AM/AM)} \tag{1.1}$$

$$\phi(r) = \frac{\alpha_\phi r^2}{1 + \beta_\phi r^2}, \text{ (AM/PM)} \tag{1.2}$$

where r is the amplitude of the input waveform, and the parameters α_a, β_a, α_ϕ, and β_ϕ are obtained by a minimum mean-square error curve fitting procedure to experimental TWT data . If $r(t)$ and $\theta(t)$ are the instantaneous input modulus and phase, respectively, of the TWT, then $A[r(t)]$ and $\phi[r(t)]+\theta(t)$ represent the instantaneous amplitude and phase, respectively, of the TWT output. For large r, the AM/AM term becomes proportional to $1/r$ by the proportionality constant α_a/β_a. Also, for large r the AM/PM term becomes the constant α_ϕ/β_ϕ.

An amplitude (magnitude, volts) and phase (radians) plot of the functions (1.1) and (1.2) with the parameters $\alpha_a = 1.9638$, $\beta_a = 0.9945$, $\alpha_\phi = 2.5293$, $\beta_\varphi = 2.8168$, is shown in Fig. 1.4. As is evident from the figure, the output amplitude given by (1.1) is normalized such that it is saturated at an input amplitude of unity. For small values of r (input magnitude, volts) the output magnitude (volts) and phase appear to be linear functions. However, as r approaches 1 the output voltage and phase begin to saturate. For $r > 1$ the output voltage begins to decrease and behaves as $1/r$. Because the TWT is between two linear filters, the overall channel is a nonlinear system with memory.

The effects of the nonlinear distortion have been studied extensively for digital radio links [10]. Fig. 1.5 illustrates an 8-PSK scatter plot of noiseless detector samples for the satellite channel of Fig. 1.3. In this case, the pre- and post-filters are 6th order butterworth with 3-dB bandwidths of $0.75R_s$. The transmit filter is rectangular, $h_T(t) = 1$, $0 \leq t < T$, and the receive filter is matched to the transmitter. The TWT is driven at 0-dB input backoff. Here, input backoff is

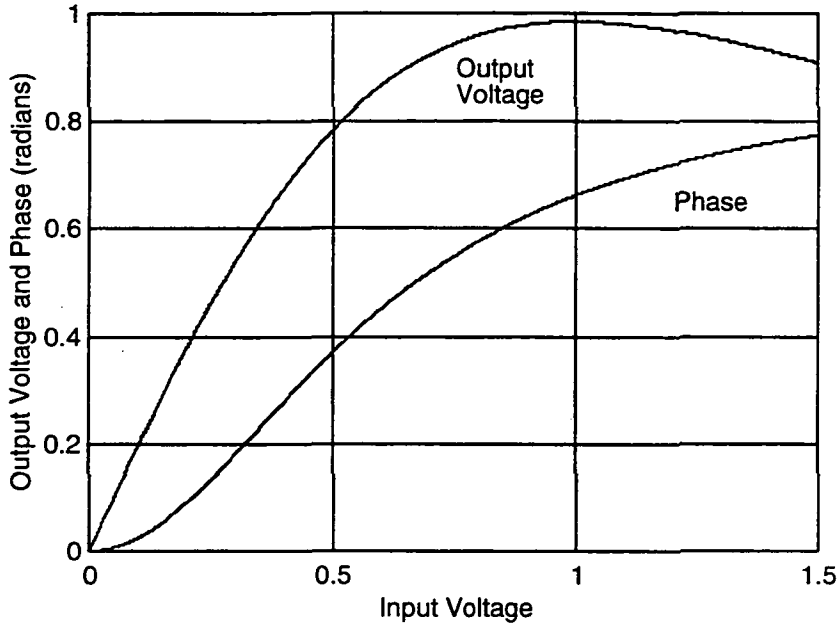

Figure 1.4. TWT amplitude and phase plot.

defined as in [11]. Referring to Fig. 1.4, an input backoff of X indicates that the average input signal power is decreased by X-dB with respect to the input signal power that causes saturation at the output. The scatter plot resembles an 8-PSK constellation with noise, however the clustering about the ideal signal point is due to linear and nonlinear ISI, not thermal noise. Also, the scatter plot resembles the effects of a bandlimited linear channel. However, as will be demonstrated in Chapter 2 the distortion is due to both linear and nonlinear components.

Fig. 1.6 illustrates a 16-QAM scatter plot of noiseless detector samples for the channel model of Fig. 1.3. Other than the 16-QAM modulation and the fact that the TWT is driven at 6-dB input backoff, the channel is identical to that for Fig. 1.5. It is evident that the inner constellation points are subject to different

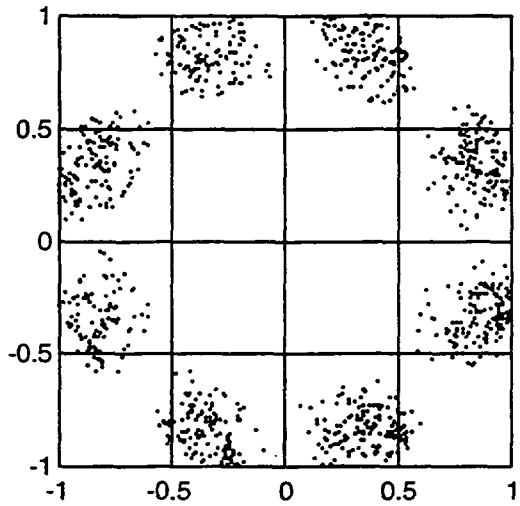

Figure 1.5. 8-PSK Clustering.

amounts of phase shift by the TWT than the outer points. Also the outer corner points receive less amplification by the TWT than the other outer signal points so that the outer constellation points appear to be on a circle. These effects are known as warping [10]. Thus, in addition to clustering, the 16-QAM constellation is subject to warping.

1.2.3 Compensation Methods for Nonlinear ISI

The predominant method of compensation for ISI in satellite channels is linear adaptive equalization in the form of a tapped delay line filter. Several new methods of compensating for nonlinear distortion have been developed for digital radio, magnetic recording, telephony, as well as satellites. These methods can be separated into those which operate at the transmitter and at the receiver.

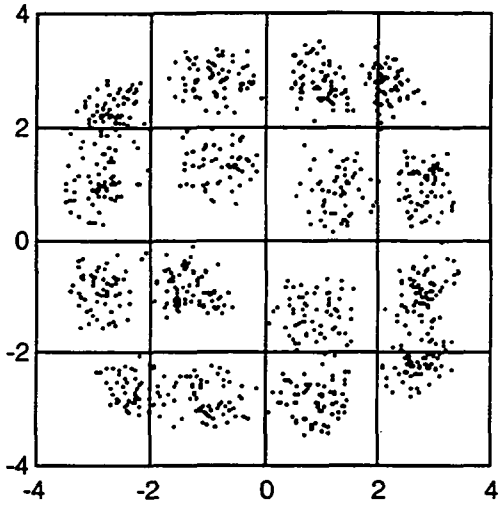

Figure 1.6. 16-QAM clustering and warping

1.2.3.1 Transmitter-Based Methods

Transmitter-based methods for nonlinear digital radio systems include analog signal pre-distortion [10], data pre-distortion [12], and data pre-distortion with memory [13]. When these methods are made adaptive they require a feedback path from the output of the nonlinearity (at the transmitter) to the pre-distortion circuit. An adaptive data pre-distortion algorithm was first developed by Saleh and Salz [12], and later utilized by Karam and Sari for data pre-distortion with memory [13]. In this algorithm, the radial and phase error after the nonlinearity is measured and the pre-distorter is adjusted so as to decrease these errors. A pre-distorter (memoryless) can be implemented with a look-up table where the information symbol serves as the address of the pre-distorted value. A pre-distorter with memory is implemented in much the same way except a concatenation of

past and present information symbols serve as the address to a memory which holds pre-distorted values. For systems with large modulation orders and many symbols of memory, the size of the predistortion memory device may become prohibitively large. However in [13], Karam and Sari have suggested methods for reducing the pre-distorter memory size.

Another issue with pre-distortion with memory is that adaption may be slow since for each memory location several cycles of adaption may be necessary. This is an issue because of the large memory size and adaption for each memory location depends on the frequency of occurrence of each symbol sequence. Despite the practical issues, pre-distortion with memory was found to give the best overall performance improvement for a nonlinear digital radio system when compared to other methods based both at the transmitter and receiver. Unfortunately, this method is not directly applicable to satellite systems since the adaption method requires a feedback path at the output of the nonlinearity.

1.2.3.2 Receiver-Based Methods

There are an abundance of compensation methods which are based at the receiver. These include the well known adaptive tapped delay line equalizer, decision feedback equalizer (DFE), fractionally-spaced equalizer (FSE), Volterra nonlinear equalizer, ISI cancellation, and MLSD. The performance of all of these methods for digital radio systems, with the exception of the FSE and MLSD, were studied by Karam and Sari [11]. A brief description of each of these methods

will be discussed below. In addition to these methods, neural network equalizers have generated some recent interest as nonlinear adaptive equalizers for magnetic recording [14], and satellite communication systems [15]. Because a neural network is very complex and is susceptible to convergence at local minima, the practicality of neural networks as adaptive equalizers is uncertain.

Tapped Delay Line Equalizers - The symbol-spaced (synchronous) tapped delay line equalizer [16] is well known and is discussed in detail in Chapter 3. This device consists of a tapped delay line, with one tap per symbol, and the output is a linear combination of the taps. The tapped delay line equalizer is very effective in reducing the performance degradation due to linear ISI. However, it is not capable of eliminating nonlinear distortion even in the absence of noise. Also, the output spectrum of the symbol-spaced tapped delay line may be aliased due to symbol rate sampling. The FSE is similar to the symbol-spaced equalizer, however it has multiple taps per symbol. The FSE inputs are sampled values of the channel output, where the channel is sampled at greater than twice the highest frequency component (after demodulation). Thus, the FSE does not suffer from aliasing. Also, an adaptive FSE can compensate for sample timing offset and minimizes the mean-square error at the output of the equalizer by matching to the channel and reducing ISI.

Decision Feedback Equalizers - A DFE is an extension of the tapped delay line. In addition to the tapped delay line equalizer preceding the decision device, a DFE equalizer also includes a tapped delay line following the decision device. The

intention is to subtract ISI from the current symbol due to previously detected symbols. In the case of severe amplitude distortion, the DFE is very effective in removing ISI from previously detected symbols without enhancing the noise and offers a performance improvement (in the probability of error sense) compared to the tapped delay line equalizer [17, ch. 6]. This is because the previously detected symbols are no longer noisy. However, the DFE suffers from error propagation due to incorrectly detected symbols.

ISI Cancellers - An ISI canceller is an extension to the DFE equalizer. The intention is to estimate and subtract from the current symbol ISI due to precursor symbols in addition to postcursor ISI. This is accomplished in two stages. First, preliminary decisions are made from which an estimate of precursor ISI is made. The precursor ISI estimate is then subtracted from an input to the final decision device. Second, as in the case of a DFE, postcursor ISI is estimated from final decisions and also subtracted from the input to the final decision device. This device is significantly more complicated than a DFE in that it requires an additional decision device and several tapped delay lines. Wesolowski [18] has found that the cancellers do not always achieve a significant improvement over DFE's of similar complexity.

Volterra Equalizers - All of the equalizer structures described thus far are based on a linear combination of taps from a tapped delay line. These structures can be generalized to devices based on nonlinear combinations of taps from a tapped delay line. These nonlinear devices are founded in the Volterra series

structure for a nonlinear communications channel. The modeling of nonlinear satellite links based on Volterra series was performed by Benedetto, Biglieri, and Daffara [19]. Benedetto and Biglieri [20] studied the performance of a Volterra series based nonlinear equalizer for a satellite channel. The equalizer was not adaptive and the performance was measured in improvement of signal to distortion ratio and did not account for noise.

- *MLSD Receivers* - MLSD structures for nonlinear satellite channels were studied by Mesiya, McLane, and Campbell [21] for binary sequences over a nonlinear satellite channel. Also, an MLSD receiver structure for nonlinear satellite channels of higher order modulation formats (i.e., $M > 2$) was proposed by Benedetto, Biglieri, and Castellani [22]. However, the performance was not analyzed for a specific satellite channel.

1.3 Overview of Chapters

A satellite communication system is a bandpass system. However, for simulation and analysis it is efficient to model such a system as a low-pass discrete-time equivalent. The low-pass discrete-time equivalent model for a linear system with ISI is easily derived [17, 23]. The generalization of this model for a nonlinear bandpass system with memory is given by the low-pass discrete-time equivalent Volterra series characterization [19]. It has also been suggested by several authors that a nonlinear satellite channel may be described as a finite-state machine [22, 24]. Chapter 2 presents a lucid explanation of the low-pass discrete-time equiva-

lent model for a nonlinear bandlimited satellite channel. In addition, an explicit development of the finite-state machine (FSM) model is given including two special cases. First, a receiver with a single receive filter and detector is considered. Second, the receiver consists of a bank of matched filters and detector. For each of these special cases the FSM model yields a state table which may be used to analyze the performance of the nonlinear channel.

As previously discussed, a fixed Volterra equalizer, following the receive filter, for a noiseless satellite communication channel was introduced by Benedetto and Biglieri [20]. In addition, it has been suggested [22] that this structure may be adapted with the LMS (least mean-square) algorithm. However, the performance of this structure was not studied for a specific satellite channel. In Chapter 3, the probability of error and mean-square error performance of this structure is studied for various PSK and QAM modulation formats for a downlink limited nonlinear bandlimited satellite channel. When the receive filter is matched to the transmitter (as is typical in satellite systems) and the transmission bandwidth approaches the satellite bandwidth, then this configuration is no longer optimal in the sense of optimizing the signal to noise ratio. For this case, an adaptive FSE is useful in compromising between optimizing the signal to noise ratio and minimizing the ISI [25]. Chapter 3 demonstrates that an FSE followed by a Volterra equalizer gives improved performance beyond that obtained from a Volterra equalizer. Also, it is found that a receive filter matched to the received pulse shape, ignoring the TWT, followed by a symbol spaced equalizer may replace the FSE with a small

loss in performance. In addition to evaluating the performance of Volterra and FSE-Volterra equalizers, Chapter 3 reviews the necessary background on symbol-spaced and fractionally spaced adaptive linear equalizers.

Forney [23] has shown that the optimum receiver for a linear channel with ISI is a whitened matched filter followed by a nonlinear processor known as the Viterbi algorithm [26]. Benedetto *et al.* [22] has shown that the MLSD receiver for a nonlinear bandlimited satellite channel is a bank of matched filters followed by a Viterbi detector, however, the performance of this structure was not evaluated for a specific satellite channel. This MLSD receiver is optimum in the probability of error sense and serves as a lower bound to the Volterra and FSE-Volterra equalizers. In Chapter 4, the probability of error performance of the MLSD receiver is studied for a specific down link limited satellite channel. Also, the relationship between the matched filter bank outputs and the Viterbi algorithm path metrics is clearly delineated. Because of the matched filter bank, the MLSD receiver may be high in complexity. Consequently the performance of a suboptimal, single receive filter receiver, is also studied. The finite-state machine models of a nonlinear communication channel are utilized for evaluating the performance of both MLSD receivers. In addition, background and justification for the MLSD receivers is presented.

A summary of the dissertation results is presented in Chapter 5. Also, the relative performance of the Volterra, FSE-Volterra, MLSD, and suboptimal MLSD,

structures is discussed. Many variations to these receiver structures merit further study, these will be suggested in Chapter 5.

This dissertation has been focused on the evaluation of receiver structures which effectively compensate for the nonlinear distortion caused by nonlinear bandlimited satellite channels. This endeavor has required a large effort in development of software tools for computer simulation of the various nonlinear effects and compensation methods. Many of the programs are software implementations of fundamental digital communications concepts. The more advanced programs, however, are listed in Appendix A.

Chapter 2

COMMUNICATION SYSTEM MODELS

A satellite communication system is a bandpass system. However, for simulation and analysis, it is efficient to model such a system as a low-pass discrete-time equivalent. The low-pass discrete-time equivalent model for a linear system with ISI is easily derived [17, 23]. The generalization of this model for a nonlinear bandpass system with memory is given by the low-pass discrete-time equivalent Volterra series characterization [19]. It has also been suggested by several authors that a nonlinear satellite channel may be described as a finite-state machine [22, 24].

This chapter first reviews the low-pass discrete-time equivalent model for a linear communication system. Then, a lucid explanation of the low-pass equivalent model for a nonlinear bandlimited satellite channel is given. Finally, the finite-state machine (FSM) model for a nonlinear bandlimited communication system is explicitly developed, including two special cases: one with a single receive filter and one whose receiver contains a bank of matched filters. For each of these cases, the FSM model yields a state table which may be used to analyze the performance of the nonlinear channel.

2.1 Low-Pass Discrete-Time Equivalent Model For a Linear System

A low-pass equivalent communication system is illustrated in Fig. 2.1. The transmit sequence d_n is complex. The transmit signal $s(t)$ has a pulse shape

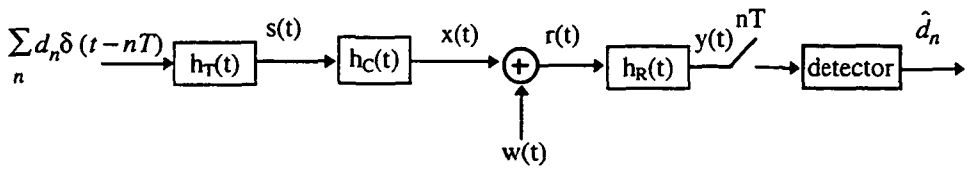

Figure 2.1. Low-pass equivalent communication system.

defined by $h_T(t)$ and is subsequently filtered by the channel $h_c(t)$. The input to the receiver $r(t)$ is the sum of the channel output $x(t)$ and an AWGN noise process $w(t)$. The receive filter outputs the signal $y(t)$ which is sampled at the symbol rate $1/T$. The output of the detector is the estimate (decision) $\hat{d}_n$ of the complex information symbol d_n.

As derived in [23], the discrete-time output of the receive filter is a weighted sum of past and present input symbols plus noise. The analog filters may be replaced with discrete-time filters which yield an identical set of inputs $y_n = y(nT)$ to the detector and thus identical detector outputs. Also, the noise process may be moved to the detector input by an appropriate transformation.

The input to the sampler may be written as

$$y(t) = \sum_k d_k h(t - kT) + \eta(t), \tag{2.1}$$

where $h(t)$ is the combined signaling waveform

$$h(t) = h_T(t) * h_C(t) * h_R(t), \tag{2.2}$$

and $*$ indicates convolution. The signal $\eta(t)$ represents the noise at the output of the receive filter given by

$$\eta(t) = w(t) * h_R(t). \tag{2.3}$$

The sampled input to the detector is given by

$$y_n = y(nT) = \sum_k d_k h_{n-k} + \eta_n, \tag{2.4}$$

where the noise samples, η_n, into the detector are given by $\eta(nT)$, and $h_{n-k} = h[(n-k)T]$. Equation (2.4) indicates that the input to the detector is a linear combination of past, present, and future discrete-time channel inputs with the addition of noise samples.

The noise samples at the output of the receive filter η_n are in general colored. That is, in general the expectation $E[\eta_n \eta_{n'}]$ is nonzero for $n \neq n'$. For the case of colored noise samples it is difficult to evaluate the performance of the given communication system. Consequently, it is desirable to whiten the noise samples via a noise whitening filter [23]. In this dissertation, the receive filter is either square-root raised cosine, or rectangular (i.e., $h(t) = 1$, $0 \leq t < T$). Consequently, the noise samples are uncorrelated, so that a noise whitening filter is not required.

The discrete-time equivalent channel model is described by equation (2.4). The summation over k is in general infinite, however for practical channels it is finite, so that $h_k \approx 0$ for $|k| > L$, where L is a positive integer. The equivalent discrete time model for the system of Fig. 2.1 is shown in Fig. 2.2, with the exception that the detector is not included in the figure. The input to the communication system is the finite sequence of information symbols $\{d_{n-k}\}$ for $|k| \leq L$. Each information symbol d_{n-k} is then multiplied by weight h_k and input to a summing device. The output of the channel model at discrete-time n is the sum of the weighted input symbols and noise samples η_n.

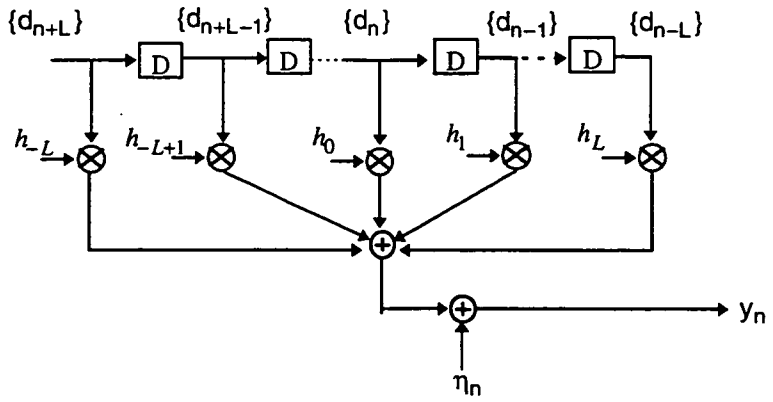

Figure 2.2. Discrete-time channel model for a linear system.

2.2 Low-Pass Discrete-Time Equivalent Model For a Nonlinear System

First, following [22], a continuous-time low-pass equivalent representation of a nonlinear downlink-limited transponder communications link is derived using a low-pass equivalent Volterra series. Next, as in the case of a linear system, the continuous-time low-pass equivalent representation is sampled to form the low-pass discrete-time model.

2.2.1 Low-Pass Equivalent Volterra Series For a Bandlimited Nonlinear Channel

The low-pass equivalent Volterra expansion for a general nonlinear channel with memory is given by [19, 22]

$$y(t) = \sum_{m=0}^{\infty} \int_{-\infty}^{\infty} d\tau_1 \ldots \int_{-\infty}^{\infty} d\tau_{2m+1} k_B(\tau_1, \ldots, \tau_{2m+1}) \cdot \prod_{i=1}^{m+1} x(t-\tau_i) \prod_{l=m+2}^{2m+1} x^*(t-\tau_l), \quad (2.5)$$

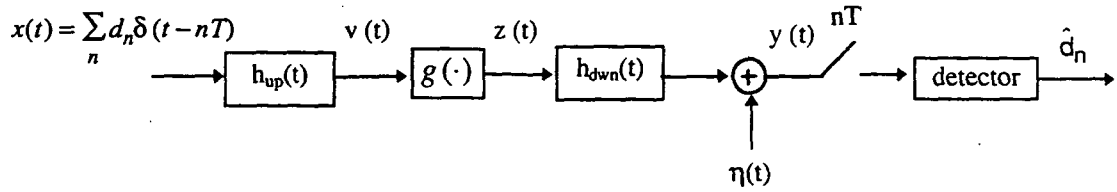

Figure 2.3. Low-pass equivalent nonlinear transponder communication system.

where the input $x(t)$ and output $y(t)$ are low-pass equivalent signals, * denotes complex conjugate, and $k_B(\tau_1, ..., \tau_{2m+1})$ is the baseband equivalent Volterra kernel, as defined in [22, ch. 2]. The low-pass equivalent Volterra expansion will now be developed for the specific case of the low-pass equivalent nonlinear transponder model of Fig. 2.3.

As indicated in the figure, the transmitted signal $x(t)$ is defined in terms of the complex data symbols d_n. The baseband equivalent linear filter $h_{up}(t)$ represents the cascade of linear filters preceding the nonlinearity $g(\cdot)$ (i.e., uplink filters), including the transmit filter and input multiplexing filter of the transponder. The nonlinear function $g(\cdot)$ represents the memoryless nonlinearity of the TWT amplifier, and will be defined in more detail below. The cascade of linear filters following the nonlinearity is represented by the linear filter $h_{dwn}(t)$ (i.e., downlink filters), including the transponder output multiplexing filter and receive filter. As in the case of the linear channel, in general, including the receive filter in the model causes the noise process $\eta(t)$ to be colored. However, as indicated previously, the receive filters used in this dissertation are such that the noise samples at the output of the receive filter are uncorrelated. The baseband equivalent

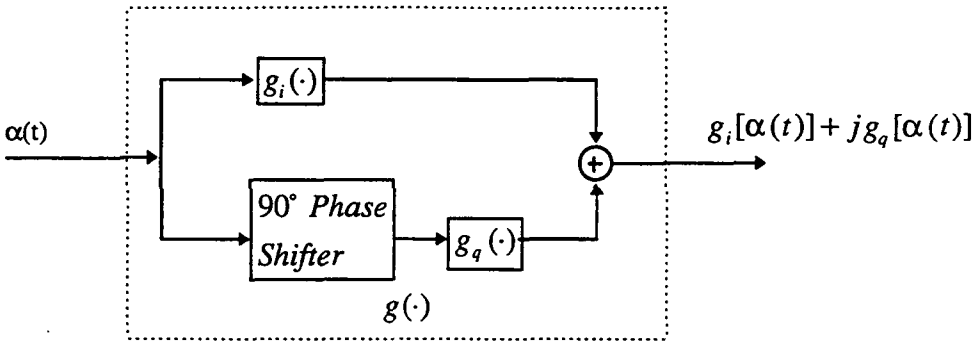

Figure 2.4. Memoryless quadrature nonlinearity.

noise source preceding the detector, $n(t)$ as in Fig. 2.3, is obtained by convolving the noise source preceding the receive filter with the impulse response of the receive filter. The detector, which may include an equalizer, provides a decision on complex data symbol d_n at discrete-time n.

The AM/AM and AM/PM conversions of the TWT amplifier is represented by the function $g(\cdot)$, which consists of the in-phase and quadrature functions $g_i(\cdot)$ and $g_q(\cdot)$, given by [9]

$$g_i = P(r)\cos[\phi(r)], \tag{2.6}$$

$$g_q = A(r)\sin[\phi(r)], \tag{2.7}$$

as shown in Fig. 2.4, [19, 9], where $A(r)$ and $\phi(r)$ are given by (1.1) and (1.2), respectively. The parameters γ_{2m+1} are obtained from a Taylor series expansion of the nonlinearity g(·):

$$g(\beta) = \sum_{m=0}^{\infty} \gamma_{2m+1}\beta^{2m+1}. \tag{2.8}$$

The parameters γ_{2m+1} consist of the in-phase and quadrature terms:

$$\gamma_{2m+1} = \gamma_{i,2m+1} + j\gamma_{q,2m+1}, \tag{2.9}$$

where $\gamma_{i,2m+1}$, and, $\gamma_{q,2m+1}$, are the coefficients obtained from a Taylor series expansion of $g_i(\cdot)$ and $g_q(\cdot)$, respectively. As is shown in [22], the presence of only odd orders in (2.8) is due to the bandpass nature of the communication system. Furthermore, for the low-pass equivalent communication system, application of (2.8) results in the following relationship between the input $v(t)$ and output $z(t)$ of the nonlinearity

$$z(t) = \sum_{m=0}^{\infty} \gamma_{2m+1} [v(t)]^{m+1} [v^*(t)]^m . \tag{2.10}$$

The input to the nonlinearity $v(t)$ and output of the channel $y(t)$ (including the effect of the receive filter) are obtained by applying straightforward linear system theory concepts:

$$v(t) = \int_{-\infty}^{\infty} h_{up}(\tau) x(t-\tau) d\tau, \tag{2.11}$$

and

$$y(t) = \int_{-\infty}^{\infty} h_{dwn}(\tau) z(t-\tau) d\tau. \tag{2.12}$$

Substituting (2.11) into (2.10) and then the result into (2.12) yields after simplification

$$\begin{aligned} y(t) \;=\; & \sum_{m=0}^{\infty} \gamma_{2m+1} \int_{-\infty}^{\infty} d\tau_1 \ldots \int_{-\infty}^{\infty} d\tau_{2m+1} \int_{-\infty}^{\infty} d\tau \cdot h_{dwn}(\tau) \\ & \cdot \prod_{r=1}^{m+1} h_{up}(\tau_r - \tau) \prod_{r=m+2}^{2m+1} h_{up}^*(\tau_s - \tau) \prod_{i=1}^{m+1} x(t-\tau_i) \\ & \cdot \prod_{l=m+2}^{2m+1} x^*(t-\tau_i) \end{aligned} \tag{2.13}$$

The low-pass equivalent Volterra kernels are obtained by comparing (2.13) to (2.5):

$$k_B(\tau_1, \ldots, \tau_{2m+1}) = \gamma_{2m+1} \int_{-\infty}^{\infty} d\tau \cdot h_{dwn}(\tau) \cdot \prod_{r=1}^{m+1} h_{up}(\tau_r - \tau) \prod_{s=m+2}^{2m+1} h_{up}^*(\tau_s - \tau) \tag{2.14}$$

Furthermore, accounting for the form of the input $x(t) = \sum_n d_n \delta(t - nT)$, the Volterra series expansion of (2.5) simplifies to

$$\begin{aligned} y(t) &= \sum_{m=0}^{\infty} \sum_{n_1=-\infty}^{\infty} \cdots \sum_{n_{2m+1}=-\infty}^{\infty} k_B\left(t - n_1 T, ..., t - n_{2m+1} T\right) \\ &\quad \cdot d_{n_1} d_{n_2} ... d_{n_{m+1}} d^*_{n_{m+2}} ... d^*_{n_{2m+1}}, \end{aligned} \tag{2.15}$$

where d_{n_i} are the data inputs at discrete time n_i.

Expressions (2.14) and (2.15) define the low-pass continuous-time Volterra series for the transponder model of Fig. 2.3. Equation (2.15) expresses the time-domain transponder output as a nonlinear combination of past, present, and future information symbols d_{n_i}. Each nonlinear combination of input symbols is scaled by the respective Volterra kernel, $k_{B,2m+1}\left(t - n_1 T, ..., t - n_{2m+1} T\right)$.

2.2.2 Discrete-Time Equivalent Model For a Bandlimited Nonlinear Transponder

The low-pass equivalent discrete-time model for a nonlinear system is obtained by sampling (2.15) at time nT. Thus, the baseband equivalent discrete-time nonlinear channel model is described by

$$\begin{aligned} y_n &= \sum_{m=0}^{\infty} \sum_{n_1=-\infty}^{\infty} \cdots \sum_{n_{2m+1}=-\infty}^{\infty} K_B\left(n_1, ..., n_{2m+1}\right) \\ &\quad \cdot d_{n_1} d_{n_2} ... d_{n_{m+1}} d^*_{n_{m+2}} ... d^*_{n_{2m+1}}, \end{aligned} \tag{2.16}$$

where $K_B\left(n_1, ..., n_{2m+1}\right)$ is the low-pass equivalent discrete-time Volterra kernel given by

$$K_B\left(n_1, ..., n_{2m+1}\right) = k_B\left(nT - n_1 T, ..., nT - n_{2m+1} T\right). \tag{2.17}$$

Equation (2.16) represents the entire nonlinear channel including receive filter. In practice all channels have finite memory and nonlinearity of finite degree so that the summations in (2.16) are finite.

2.3 Finite-State Machine Model

A downlink-limited nonlinear communications channel with finite memory may be modeled with a finite-state machine (FSM) [22, ch. 10], [24]. In contrast to the low-pass discrete time Volterra characterization of a communications channel, the FSM model of a communications channel is easily obtained. However, it may require a large amount of storage. The model is useful in deriving the channel statistics as in [24] or for deriving a state table which may be employed directly by a Viterbi detector. The FSM model consists of a nonlinear transmitter in the form of a FSM and a receiver. The following section explicitly develops the nonlinear FSM transmitter. Then, two special cases of this model will be considered. The state tables obtained from both of these cases are employed in Chapter 4 for studying the performance of nonlinear bandlimited satellite channels.

First, the discrete-time detector inputs will be derived for a receiver consisting of a single receive filter. A state table description of the channel is then obtained from the FSM model, where the state table contains a listing of each channel input, state, and discrete-time detector input. Although for a nonlinear bandlimited satellite channel the single receive filter is suboptimal, it is the typical case. Thus, this case is useful for analyzing the performance of a typical receiver.

Second, a receiver with a bank of matched filters is considered. As in the previous case, a state table description of the channel is obtained. However, in this case the state table listing contains the channel input, state, and oversampled outputs from the channel. If, as in the previous case, the state table were to contain the receive filter outputs for each input-state combination, the state table would become prohibitively large. Therefore, in this case the state table listing contains the oversampled outputs of the channel. The oversampled output of the channel is the output of the channel, over the time interval $[nT, (n+1)T)$, sampled at a rate greater than twice the highest frequency. In this case, the task of calculating the discrete-time filter outputs is left to the receiver. The method of obtaining the discrete time outputs for each filter in the filter bank is then discussed. A receiver consisting of a matched filter bank followed by a Viterbi detector is the optimal detector for this channel. Therefore, this case is useful for analyzing the performance of the optimum receiver. The relationship between the filter bank and the Viterbi algorithm path metrics will be discussed in Chapter 4.

2.3.1 Nonlinear FSM Transmitter

A FSM model useful for modeling a downlink-limited satellite channel is shown in Fig. 2.5. In this model, the transmitter and satellite channel (i.e., as in Fig. 2.3) are modeled as a nonlinear transmitter in the form of a FSM. The input to the FSM is the discrete-time data sequence $\{a_n\}$, $a_n \in \{0, 1, \ldots, M-1\}$, and the output are the chips $h(t - nT, a_n, \sigma_n)$, where each chip is zero outside the in-

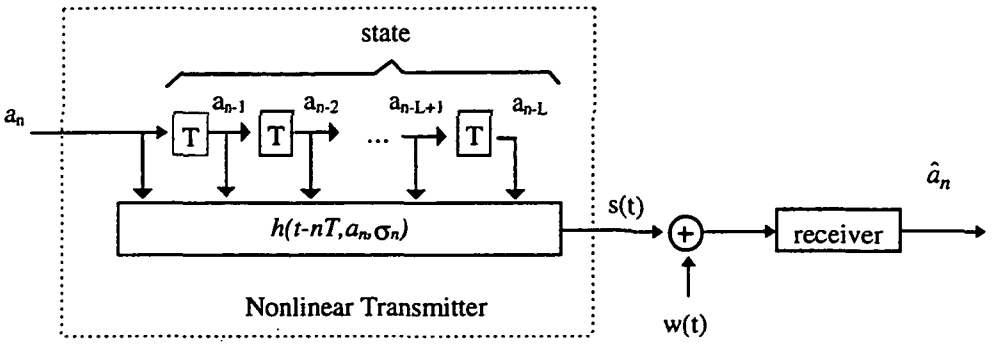

Figure 2.5. FSM model of a communication system.

terval $[nT, (n+1)T)$, σ_n is the state, and M is the symbol set size. Therefore, the output signal $s(t)$ of the nonlinear transmitter is the sum of nonoverlapping chips:

$$s(t) = \sum_n h(t - nT, a_n, \sigma_n). \tag{2.18}$$

The input to the receiver is the signal $s(t)$ plus noise $w(t)$. The receiver filters, samples, and detects the signal to produce an estimate $\hat{a}_n$ of the information symbol.

Assuming that the communications system has a memory of L symbols, the state σ_n of the FSM is the set of L previous channel inputs $\{a_{n-1}, a_{n-2}, \ldots, a_{n-L}\}$. Therefore, the nonlinear transmitter has M^L unique states and M unique inputs. Consequently, the FSM can generate up to M^{L+1} distinct chips.

The output signal of the nonlinear transmitter $s(t)$ of Fig. 2.5 corresponds to the signal $s(t)$ of Fig. 2.6. Assume the system of Fig. 2.6 has a memory of L_p past and L_f future symbols, thus a total memory of $L = L_p + L_f$. Then, the nth symbol interval at the output of the post-filter occurs over the time interval $[nT + L_p,\ (n+1)T + L_p)$. Therefore, the output $s(t)$ of the FSM transmitter is

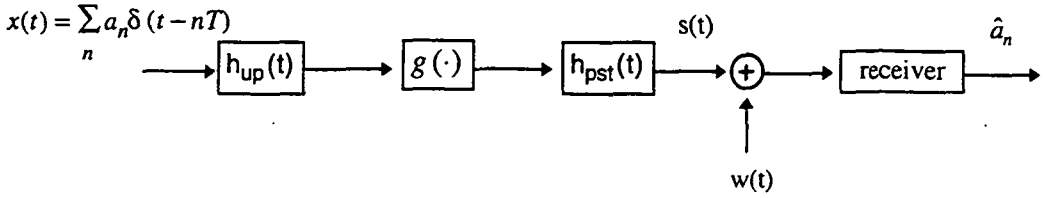

Figure 2.6. Low-pass nonlinear system for development of FSM models.

written with respect to the signals from Fig. 2.6 as

$$h(t - nT, a_n, \sigma_n) = \int_{-\infty}^{\infty} g\left(\sum_k a_k h_{up}(\tau - kT)\right) h_{pst}(t - \tau) d\tau, \qquad (2.19)$$
$$nT + L_p \leq t < (n+1)T + L_p$$

2.3.2. Case I: Single Receive Filter

The FSM model with a single receive filter is shown in Fig. 2.7. The output of the receive filter $\gamma(t)$ is sampled and detected to provide the estimated data symbol $\hat{a}_n$. The discrete time data inputs to the detector are expressed as

$$\gamma_n = \gamma(nT) = s_n + \eta_n, \qquad (2.20)$$

The noiseless detector inputs are given by

$$s_n = \int_{-\infty}^{\infty} h(\tau - nT, a_n, \sigma_n) h_R(nT - \tau) d\tau, \qquad (2.21)$$

and the discrete-time noise samples by

$$\eta_n = \int_{-\infty}^{\infty} w(\tau) h_R(nT - \tau) d\tau. \qquad (2.22)$$

The noiseless discrete-time data inputs together with the input and state of the FSM may be listed in a state table. For example, suppose Fig. 2.7 models a QPSK satellite system with memory $L = L_f + L_p$, where the memory consists

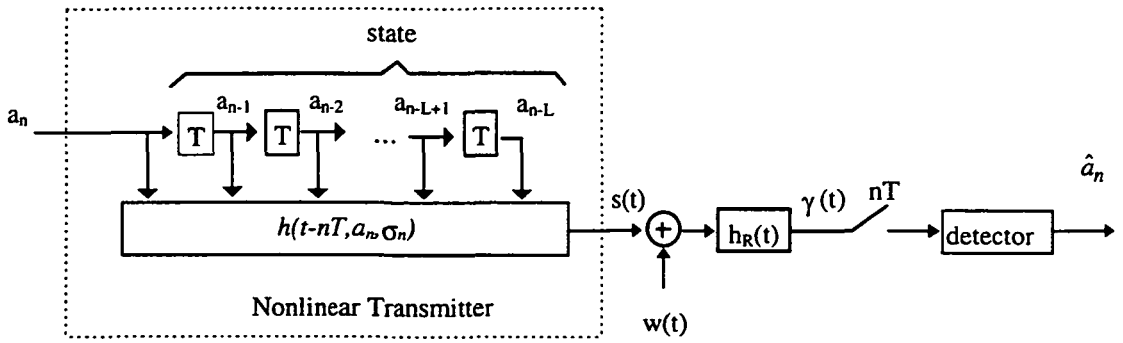

Figure 2.7. FSM model with a single receive filter.

of one past symbol, $L_f = 1$, and one future symbol, $L_p = 1$. A state table for this case contains $M^{L+1} = 64$ input-state combinations, and is listed in table 2.1. The first column lists the input a_n and the next two columns list the state σ_n (i.e., the two previous inputs a_n and a_{n-1}). The last column lists the noiseless complex discrete-time receive filter outputs s_n (i.e., the noiseless detector inputs). The symbol a_{n-1} is defined as the "punctual symbol," since it determines in which quadrant the output resides. In the absence of distortion, the mapping from the punctual symbol to constellation point is defined in Table 2.2.

Since the FSM model transmitter consists of a shift register, then the state transition statistics can be modeled by a Markov process. Also, the present-to-next state transitions are implicit. For example, with present state $\sigma_n = \{a_{n-1}, a_{n-2}\}$ and input a_n the next state is given by $\sigma_{n+1} = \{a_n, a_{n-1}\}$.

This state table may be employed to derive the noiseless discrete time outputs of a nonlinear transmitter. In such a case, the inputs a_n are obtained randomly from the set {0, 1, ..., M-1}. Noise samples η_n may then be added to s_n to form the input to the detector. The state table may then be used to derive the path

Table 2.1. State table for single receive filter receiver

a_n	σ_n		s_n
0	0	0	$-0.7059 - 0.7059i$
1	0	0	$-0.7577 - 0.7059i$
2	0	0	$-.7059 - 0.7577i$
3	0	0	$-0.7577 - 0.7577i$
0	1	0	$0.5178 - 0.7577i$
1	1	0	$0.4660 - 0.7079i$
2	1	0	$0.5178 - 0.7577i$
3	1	0	$0.4660 - 0.7577i$
			$\vdots$
0	3	3	$0.7146 + 0.7146i$
1	3	3	$0.6628 + 0.7146i$
2	3	3	$0.7146 + 0.6628i$
2	3	3	$0.6628 + 0.6628i$

Table 2.2. Punctual symbol to constellation point mapping

$\mathbf{a}_{n-1}$	$\mathbf{s}_n$
0	$-\frac{1}{\sqrt{2}} - \frac{i}{\sqrt{2}}$
1	$\frac{1}{\sqrt{2}} - \frac{i}{\sqrt{2}}$
2	$-\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}$
3	$\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}$

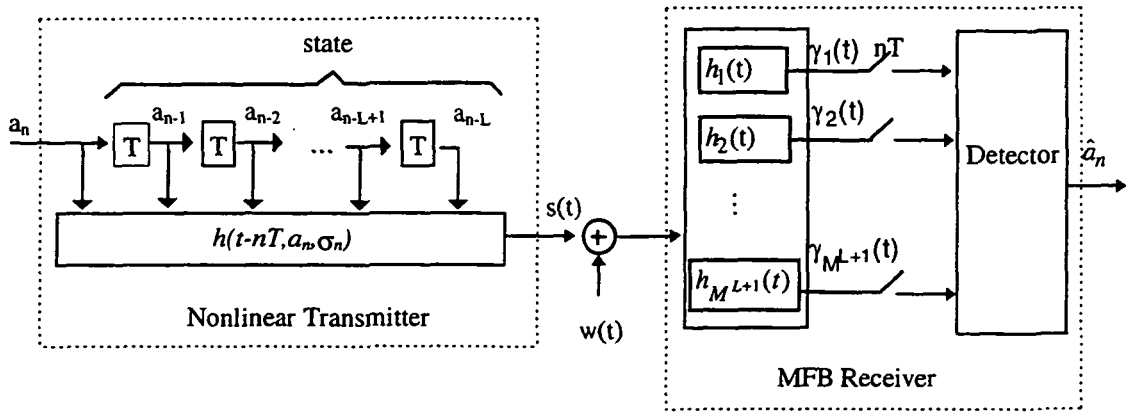

Figure 2.8. FSM model with a matched-filter bank receiver.

metrics for detection of the sequence of input symbols $\{a_n\}$. Alternatively, other information such as channel statistics or minimum distance may be derived from the state table. These topics will be discussed further in chapter 4.

2.3.3 Case II: Filter Bank Receiver

Fig 2.8 shows a communication system with a finite-state machine transmitter where the receiver consists of a bank of matched filters. The bank of matched filters consists of a filter h_i matched to each of the M^{L+1} waveforms $h(t, a_n, \sigma_n)$ generated by the transmitter. For example, the filter h_i corresponding to $h(t, a_n, \sigma_n)$ has the response $h^*(T - t, a_n, \sigma_n)$.

As in the previous case, suppose the FSM model of Fig. 2.8 represents a QPSK satellite system with a memory of one past and one future symbol. As before there are $M^{L+1} = 64$ chips $h(t, a_n, \sigma_n)$. Consequently, the filter bank has 64 matched filters. If, as in the previous case, the state table were to contain the receive filter outputs for each input-state combination, the state table would

Table 2.3. State table for matched filter bank receiver

a_n	σ_n		h^1	h^2	h^3	h^4
0	0	0	$-0.39-0.36i$	$-0.37-0.38i$	$-0.35-0.39i$	$-0.37-0.38i$
1	0	0	$-0.09-0.47i$	$-0.37-0.38i$	$-0.38-0.35i$	$-0.37-0.38i$
2	0	0	$-0.49+0.04i$	$-0.37-0.38i$	$-0.31-0.41i$	$-0.37-0.38i$
3	0	0	$-0.02-0.0i$	$-0.37-0.38i$	$-0.32-0.39i$	$-0.37-0.38i$
				$\vdots$		

become prohibitively large (64 matched filters for the present example). Therefore, in this case the state table listing contains the oversampled outputs of the channel. For each input-state combination (a_n, σ_n), the oversampled output of the channel is the chip $h(t, a_n, \sigma_n)$ sampled at a rate greater than twice its highest frequency component. Table 2.3 contains a partial state table listing for this case, where the sampling rate is four times the symbol rate. The punctual symbol is a_{n-1}, and the punctual symbol to constellation point mapping is the same as for case I. Thus, the table lists only input state combinations corresponding to the third quadrant. The first column contains the input and the next two columns contain the state. The next four columns contain the four samples of the chip $h(t, a_n, \sigma_n)$.

As mentioned previously, a receiver consisting of a matched filter bank followed by a Viterbi detector is the optimal detector for this channel. In this case, however, it is necessary for the receiver to compute the discrete-time filter outputs during each signaling interval. Given that the chip $h(t, a_n, \sigma_n)$ was transmitted over the nth time interval, the output of the ith filter is given by

$$\gamma_i(a_n, \sigma_n) = s_i(a_n, \sigma_n) + \eta_{i,n}, \tag{2.23}$$

where $s_i(a_n, \sigma_n)$ is the noiseless output for the ith filter, given by

$$s_i(a_n, \sigma_n) = \int_0^T h(\tau, a_n, \sigma_n) h_i(T - \tau) d\tau. \tag{2.24}$$

The discrete-time noise samples for the ith filter are given by

$$\eta_{i,n} = \int_{nT}^{(n+1)T} w(\tau - nT) h_i(T - \tau) d\tau, \tag{2.25}$$

where $w(t)$ is the noise process at the input to the filter bank, as in Fig. 2.6. For the case of $w(t)$ an AWGN process, then $\eta_{i,n}$ is a Gaussian random variable with variance

$$\nu_i^2 = \frac{N_0}{2} \int_0^T |h_i(t)|^2 \, dt. \tag{2.26}$$

Since the state table contains the chip samples sampled at a rate greater than twice the highest frequency, the state table (e.g., table 2.3) contains the necessary information to optimally detect the input signal. Given the chip $h(\tau, a_n, \sigma_n)$ was transmitted, then the sampled input to the filter bank is given by the sequence $\{r^j(a_n, \sigma_n)\}$, where j=1, 2 ... N_{ss}, and N_{ss} is an integer representing the number of samples per signal. Because of practicality considerations, here we restrict N_{ss} to an integer. The received samples are given by $r^j(a_n, \sigma_n) = h^j(a_n, \sigma_n) + w_n^j$, where $h^j(a_n, \sigma_n)$ is the jth sample of the chip $h(t, a_n, \sigma_n)$, and w_n^j is the jth sample of the noise process $w(t)$ over the nth time interval. Thus, given that the chip $h(t, a_n, \sigma_n)$ was transmitted, the output of the ith filter is given by

$$\gamma_i(a_n, \sigma_n) = \sum_{j=1}^{N_{ss}} \left(h^j(a_n, \sigma_n) + w_n^j \right) h_i^{N_{ss}-j}, \tag{2.27}$$

where h_i^j is the jth sample of the filter $h_i(t)$. Note that the filter samples may be obtained from the chip samples contained in the state table. This equation may also be decomposed into a form corresponding to the signal and noise as in

(2.24) and (2.25), respectively. Therefore, given the sampled received sequence, the discrete-time output for each filter may be computed from the chip samples in the state table.

2.4 Chapter Summary

This chapter has reviewed the low-pass discrete-time equivalent model for a linear passband communication system. Next, beginning with the low-pass equivalent Volterra series, a lucid development of the low-pass Volterra discrete-time model for a nonlinear satellite communications channel was presented. The resulting model is a polynomial expression relating an output symbol at discrete time n to the past, present, and future, input symbols. Deriving this model for a particular channel is computationally intensive.

The last section of this chapter presented a FSM model, where two special cases were considered. The discrete-time detector inputs were derived for the single receive filter receiver. Although this receiver is suboptimal, this case is useful for evaluating the performance of the typical receiver. In contrast to the low-pass discrete-time Volterra model, the FSM model is easily derived, however it may require a large amount of memory for storing the state machine. Despite the possibly large memory requirement, the model is more efficient, computationally, than the Volterra model. In this case, the FSM model is simply a lookup table containing the discrete-time detector inputs whereas the Volterra model requires evaluation of a polynomial expression.

Finally, a state table listing appropriate for a receiver with a matched filter bank was described. This state table listing contains samples of the chips $h(t, a_n, \sigma_n)$ sampled at a rate greater than twice the highest frequency component. Also, a method of computing the matched filter bank outputs from the state table entries was described. This state table may be used for evaluating the performance of the optimal receiver for the nonlinear bandlimited satellite channel consisting of a matched filter bank followed by a Viterbi detector. As in the previous case, the state table is easily derived and the model is still more efficient than the Volterra model. Although in this case the filter bank outputs must be computed for each symbol interval, this additional computational requirement also exists for the Volterra model.

Chapter 3

ADAPTIVE VOLTERRA EQUALIZERS FOR NONLINEAR SATELLITE CHANNELS

A fixed Volterra equalizer for compensation of ISI due to a noiseless satellite communication channel has previously been studied [20]. In addition, it has been suggested [22] that this structure may be adapted with the LMS (least mean-square) algorithm. However, the performance of this structure has not been studied for a specific satellite channel, as is done here. In addition, here a multiple-step size algorithm is used to improve the convergence characteristics of the equalizer.

Since many of the concepts of linear adaptive filters apply to adaptive Volterra equalizers, first this chapter briefly reviews the related concepts from adaptive linear filter theory. Next, the adaptive Volterra equalizer is introduced, and the conditions for convergence are analyzed. Then, the mean-square error and probability of error performance is studied for various PSK and QAM modulation formats.

Because of the aliased frequency spectrum into the detector, due to symbol rate sampling, a symbol spaced adaptive equalizer can only act to modify this aliased (i.e., folded) spectrum, and is thus limited in its ability to compensate for ISI. In this case, an adaptive FSE followed by a Volterra equalizer is shown to give improved performance beyond that obtained from a T-spaced Volterra equalizer following the receive filter. Also, the ability of an FSE to adaptively realize the optimum linear receive filter is reviewed.

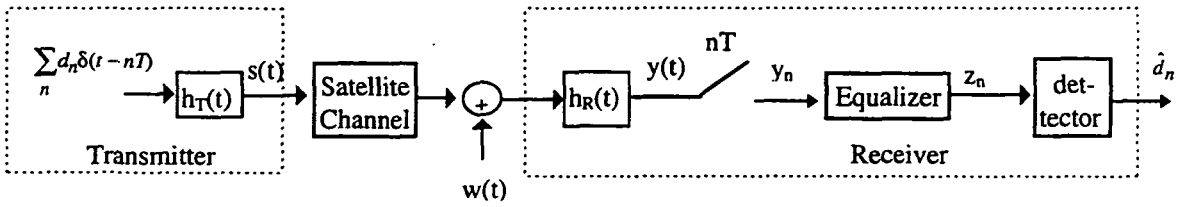

Figure 3.1. Low-pass equivalent satellite communications channel.

3.1 Linear Equalizer Background

The linear equalizer is developed according to the linear communications system model shown in Fig. 3.1. For this discussion, the channel is represented by the linear filter $h_c(t)$. The input to the equalizer y_n is the sampled output of the receive filter $h_R(t)$, and the equalizer output is z_n. In the absence of a nonlinearity, the combined signaling pulse $h(t)$ is obtained by a convolution $h(t) = h_T(t) * h_C(t) * h_R(t)$. The discrete-time equalizer inputs y_n may be expressed as

$$y_n = \sum_k d_{n-k} h_k + \eta_n, \tag{3.1}$$

where $h_k = h(kT)$, and $\eta_n = \eta(nT)$ is the discrete-time noise process at the output of the receive filter. The noise process $\eta(t)$ is given by the convolution $w(t) * h_R(t)$.

It has been shown that for every reasonable optimization criterion that the optimum receive filter is a matched filter followed by a tapped delay line [27]. The transversal equalizer (i.e., tapped delay line) is shown in Fig. 3.2. Its output z_n is a linear combination of the input sequence $\{y_n\}$ given by

$$z_n = \sum_{k=-K}^{K} y_{n-k} c_k, \tag{3.2}$$

where the c_k are complex weights.

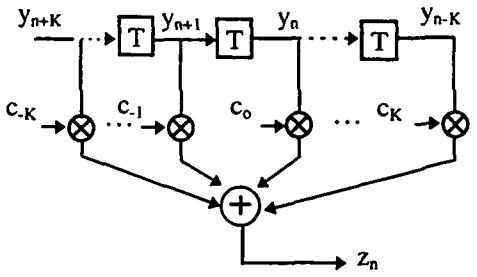

Figure 3.2. Transversal filter.

The optimum, in the MSE sense, tap settings for the infinite length transversal equalizer are found according to the inverse z-transform of [17, ch. 6]

$$C(z) = \frac{H^*(z^{-1})}{H(z)H^*(z^{-1}) + N_0}, \tag{3.3}$$

where $H(z)$ is the z-transform of the discrete time signaling pulse h_k and $N_0/2$ is the spectral density of the noise process $w(t)$. If N_0 is sufficiently small, then equation (3.3) indicates that $C(z)$ approaches the inverse of the signaling pulse $H(z)$.

For the case of a finite length transversal filter, the optimum tap weights, $\mathbf{C}_{opt}$, are expressed by the matrix form of the Wiener-Hopf equation [28, ch. 5]

$$\mathbf{C}_{opt} = \mathbf{R}^{-1}\mathbf{p}, \tag{3.4}$$

where $\mathbf{R}$ is the (2K+1) X (2K+1) correlation matrix $E[\mathbf{y}_n\mathbf{y}_n^*]$, $\mathbf{p}$ is the (2K+1) element cross-correlation vector $E[\mathbf{y}z_n^*]$, and $\mathbf{y}_n$ is the tapped delay line input vector at discrete-time n.

3.1.1 The LMS Algorithm

As indicated by (3.3) and (3.4), the optimum MSE tap settings for the transversal equalizer are a function of the channel characteristics. Often these are not known a priori. An algorithm capable of adapting the equalizer coefficients such that the mean-square error is minimized without knowledge of the channel statistics is the LMS algorithm. This algorithm is simple yet very effective.

The instantaneous error at the output of the equalizer is given by

$$e_n = d_n - z_n, \tag{3.5}$$

where d_n is the desired response at time n. In the LMS algorithm, the update to weight c_k at time $n+1$ is obtained from the negative gradient of e_n with respect to c_k:

$$c_k(n+1) = c_k(n) - \frac{1}{2}\mu\nabla_{c_k}\left(e_n e_n^*\right), \tag{3.6}$$

where μ is the step size and * denotes complex conjugation. Since the tap weight updates are obtained from the instantaneous square error (as opposed to MSE used in a true gradient search algorithm), the tap updates are noisy and result in an excess mean-square error beyond that obtained by the optimum tap weights given by (3.4). However, a small step size has the effect of averaging the updates. Conversely, the larger the step size the faster the equalizer convergence, however this results in a larger excess mean-square error. Evaluating (3.6) with e_n given by (3.5) and z_n given by (3.2) yields the well known form of the LMS update equation [29, 31]

$$c_k(n+1) = c_k(n) + \mu e_n y_{n-k}^*. \tag{3.7}$$

Equation (3.7) may be expressed in vector form

$$\mathbf{c}_{n+1} = \mathbf{c}_n + \mu e_n \mathbf{y}_n^*, \tag{3.8}$$

where $\mathbf{c}_n$ is the tap weight vector at time n.

The desired response d_n in the expression for e_n, equation (3.5), may be replaced by the receiver output $\hat{d}_n$ if decisions are correct with moderate to high probability. Usually, this substitution can be made if the probability of error is less than 0.1. This mode of operation is known as decision-directed training and was invented by Lucky [32]. In this mode, the equalizer can track slow variations in the channel characteristics.

3.1.2 MSE Performance of the LMS algorithm

The MSE performance of the LMS algorithm is described as

$$J = E[e_n e_n^*] = J_{\min} + J_\Delta, \tag{3.9}$$

where $J_{\min}$ is the MSE obtained from the optimum tap settings, (3.4), and J_Δ is the excess MSE described above. The excess mean-square error J_Δ is difficult to evaluate mathematically, however the analysis can by simplified with certain assumptions [17, ch. 6]. First, it is assumed that the mean values of the equalizer coefficients $\mathbf{c}_n$ have converged to their optimum values $\mathbf{C}_{opt}$. Second, the instantaneous square error $|e_n|^2$ is assumed to be uncorrelated with the tap input vector.[2] Under these assumptions, it is found that the excess mean-square error may be

[2]This assumption is not strictly true. However, it is noted by Proakis [17] that it simplifies the derivation and yields useful results.

expressed as

$$J_\Delta = \mu^2 \sum_{k=-K}^{K} \frac{\lambda_k}{1-(1-\mu\lambda)^2}, \tag{3.10}$$

where λ_k is the kth eigenvalue of the tap input correlation matrix. This expression explicitly shows the dependence of the mean-square error on the step size.

3.1.3 Convergence of the LMS algorithm

Two types of convergence will be discussed: convergence in the mean and convergence of the excess mean-square error. Convergence in the mean refers to convergence of the mean tap vector $E[\mathbf{c}]$ to the optimum tap vector $\mathbf{C}_{opt}$ as the number of iterations n approaches infinity. By an appropriate change of basis [28, ch. 9], it is shown that the condition for convergence in the mean is for the step size to satisfy

$$0 < \mu < \frac{2}{\lambda_{\max}}, \tag{3.11}$$

where $\lambda_{\max}$ is the largest eigenvalue of the tap input correlation matrix. Also, in examining convergence in the mean it is found that the convergence rate (i.e., of $E[\mathbf{c}]$ to $\mathbf{C}_{opt}$) is limited by the eigenvalue spread $\lambda_{\max}/\lambda_{\min}$, where $\lambda_{\min}$ is the smallest eigenvalue of the tap input correlation matrix.

Unfortunately, expression (3.11) does not assure good mean-square error performance. Thus, it is useful to establish the conditions for which the excess mean-square error converges. Under the assumption that the sequence of tap input vectors $\mathbf{y}_n$ are independent (independence assumption) and that the error $e_{n,opt}$ (obtained by evaluating (3.5) with $\mathbf{C}_{opt}$) is independent of all tap inputs, the fol-

lowing condition for convergence of the excess mean square error is obtained [33, ch. 8].[3]

$$0 < \mu < \frac{2}{(2K+1)\lambda_{\max}}. \tag{3.12}$$

This expression indicates that for convergence of the excess MSE, the maximum step size is inversely proportional to the number of taps as well as $\lambda_{\max}$.

3.2 Adaptive Volterra Equalizer

In this section, the structure of a Volterra equalizer is described. Next, the method of adaption is discussed including convergence and a multiple-step size adaption method. With methods similar to [35], the mean-square error and probability of error performance of the adaptive equalizer is then evaluated via Monte-Carlo computer simulations.

As discussed in chapter 2, the Volterra series characterization of a nonlinear communication channel provides a relationship between the discrete-time input symbols and the discrete-time channel output symbols. Recall, that equation (2.15) expresses the time-domain transponder output as a nonlinear combination of past, present, and future information symbols d_{n_i}.

The Volterra series equalizer is motivated by the theory of the pth-order inverse for nonlinear systems [20 - 37]. The pth-order inverse K_p^{-1}, of a system H, consisting of a pth order Volterra series is defined as a system for which the

[3] In practice, results predicted by the independence assumption hold over a wide range of step values [33]. Other researchers have analyzed the LMS convergence without the independence assumption [34].

Volterra series of the system Q, formed by the tandem connection of K_p^{-1} and H, is such that the 2nd through the pth-order Volterra operators of Q are zero. Also, the pth-order pre-inverse is identical to the pth-order post-inverse. Thus, a pth order Volterra series with properly selected coefficients can remove the nonlinearity of a nonlinear system up to pth order. However, the theorem also states that the tandem connection of H and Q may produce nonlinear terms of higher-order (i.e., p+1, p+2, ...). These higher order terms, however, are negligible in weakly nonlinear systems.

In light of the discussion on pth-order inverse, the discrete-time low-pass equivalent Volterra series characterization of a nonlinear communications channel (2.16) suggests the form of the nonlinear Volterra equalizer. The output of the equalizer, z_n, consists of a linear combination of all linear terms and all possible combinations of nonlinear terms of y_n of odd degree and is given by [20]

$$\begin{aligned} z_n &= \sum_k c_k y_{n-k} + \sum_{k_1}\sum_{k_2}\sum_{k_3} c_{k_1,k_2,k_3} y_{n-k_1} y_{n-k_2} y^*_{n-k_3} \\ &\quad + \sum_{k_1}\sum_{k_2}\sum_{k_3}\sum_{k_4}\sum_{k_5} c_{k_1,k_2,k_3,k_4,k_5} y_{n-k_1} y_{n-k_2} y_{n-k_3} y^*_{n-k_4} y^*_{n-k_5} + \cdots \end{aligned} \quad (3.13)$$

This expression consists of infinite summations of linear ISI terms and nonlinear ISI terms of odd degrees. In practice, any channel has a finite memory and nonlinearity of finite degree, so that the summations in (3.13) are finite.

Fig. 3.3 illustrates a block diagram of a 3-tap 3rd-order Volterra equalizer, where y_0 in the figure corresponds to the y_n in (3.13). The samples from a tapped delay line are the inputs to a nonlinear combiner. The nonlinear combiner then outputs all single taps and all combinations of three taps. Each output from the

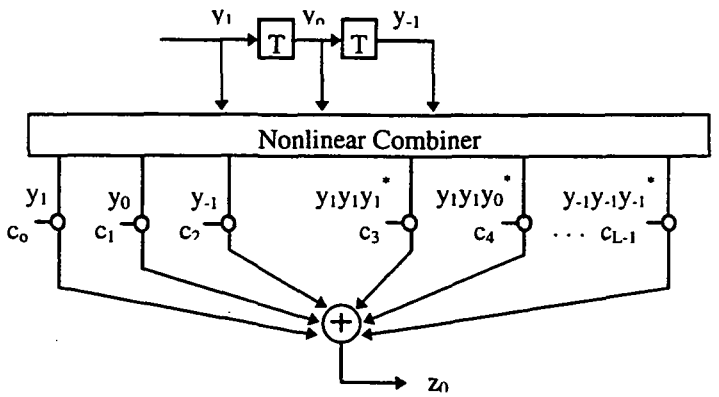

Figure 3.3. 3-tap 3rd-order Volterra equalizer.

nonlinear combiner is then scaled by a weight c_k to form an input to the summing device.

In [20], a significant reduction in complexity is proposed for M-PSK systems by eliminating terms from the nonlinear combiner of the form $y_i y_j y_k^* = y_j$ for $i = k$ (and similarly for $j = k$), assuming y_n is of modulus one. However, because y_n contains a noise term, this reduced complexity equalizer suffers a performance degradation. This will be discussed further in the mean-square error performance section, Section 3.2.2.

As indicated by equation (3.13), for each order m greater than one, the nonlinear combinations consist of $(m+1)/2$ terms times $(m-1)/2$ terms which are conjugated. For example, for $m = 3$ the nonlinear combinations take the form $y_{n-k_1} y_{n-k_2} y_{n-k_3}^*$. This suggests that for $m = 3$ there are 18 unique 3rd order terms, N_3. Therefore, the total number of terms in a nonlinear Volterra equalizer

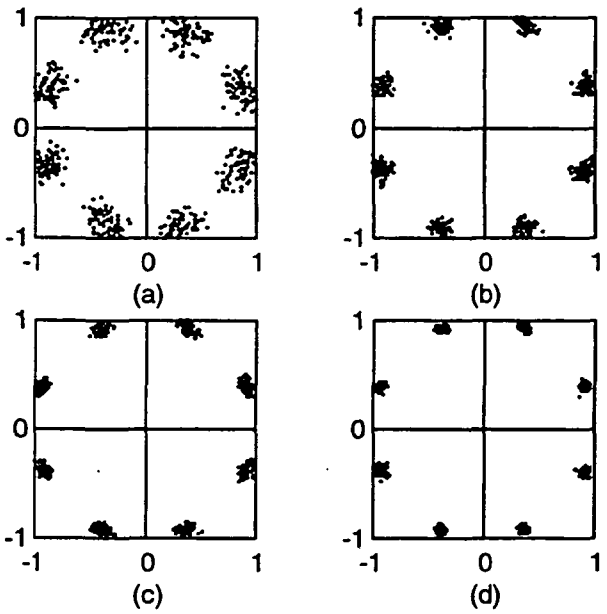

Figure 3.4. Scatter plots: (a) no equalization, (b) 7-tap linear, (c) 7-tap linear, 3-tap 3rd-order, (d) 7-tap 3rd-order.

is given by

$$L = N + \sum_{\substack{m=3, \\ m \text{ odd}}}^{\varphi} N_m, \tag{3.14}$$

where φ is the highest nonlinear term in the equalizer, and N_m is the number of terms of order m. For the Volterra equalizer of Fig. 3.3, the total number of terms is 21. It is also possible to have a nonlinear equalizer with N-tap linear compensation, but with a subset, N_S, of the N taps forming nonlinear combinations.

The ability of the Volterra equalizer to mitigate the effects of nonlinear ISI are illustrated in the 8-PSK scatter plots of Fig. 3.4. For this example, the transmit filter has a rectangular pulse shape and the receive filter is matched to the transmit pulse shape. The satellite pre- and post-filters are 6th order low-pass

butterworth with a 3 dB cutoff frequency of $0.75R_s$. The noise signal at the input to the receiver is set to zero (i.e., $w(t) = 0$) and the TWT is driven at 0 dB backoff.

For Fig. 3.4, the Volterra equalizers contained redundant terms and the total number of terms is given by

$$L' = \sum_{\substack{m=1 \\ m\ odd}}^{\varphi} N^m. \tag{3.15}$$

In this case, the number of terms for each nonlinear order is N^m where all possible terms of a particular order are included including redundant terms. This is the case for all of the Volterra equalizers evaluated via computer simulation in Sections 3.2.3 and 3.2.4. This will affect the rate of convergence since in effect the step size is increased for the redundant terms. However, the converged mean-square error and probability of error results are not affected. Including the redundant terms simplified the software implementation of the Volterra equalizer.

Fig. 3.4 (a) shows a scatter plot of the receive filter output without equalization. In Fig. 3.4 (b), the receiver output is equalized with a 7-tap linear equalizer. Figs. 3.4 (c) and (d) illustrate the equalizer output of a 7-tap linear, 3-tap 3rd-order equalizer (7 linear and 27 3rd-order terms) and a 7-tap 3rd-order equalizer (7 linear and 343 3rd-order terms), respectively. As is evident in Fig. 3.4, the Volterra equalizers reduce the signal distortion beyond that of the linear equalizer. Also, the 7-tap 3rd-order equalizer reduces the nonlinear distortion beyond that of the 7-tap linear, 3-tap 3rd-order device. In the following sections, 3.2.3 and 3.2.4, the performance of these devices will be evaluated in the presence of noise.

3.2.1 Volterra Equalizer Adaption.

In the following discussion, the method of adaption and associated convergence for the nonlinear Volterra equalizer are discussed. First, the form of the MSE at the output of the equalizer is described. Then, the LMS weight update algorithm is described in terms of the nonlinear Volterra equalizer. The convergence of the equalizer is then studied and compared to the linear equalizer. The nonlinear Volterra equalizer convergence is found to be slower than that of a linear equalizer with the same number of taps. Consequently, a multiple-step size LMS algorithm which improves the convergence characteristics is described.

3.2.1.1 MSE Output of the Nonlinear Volterra Equalizer

Denote the output of the nonlinear combiner at time n by the complex vector $\mathbf{u}_n = [u_0(n)\ u_1(n)\ \cdots\ u_{L-1}(n)]^T$, where T denotes transpose, $u_i(n)$ is the ith nonlinear combiner output at time n, L is given by (3.14), and each u_i $(i = 0, 1, ..., L-1)$ is an output of the nonlinear combiner. Denote the complex weight vector at time n by $\mathbf{c}_n = [c_o(n)\ c_1(n)\ \cdots\ c_{L-1}(n)]^T$, where $c_i(n)$, is the weight multiplying $u_i(n)$. Then, the output z_n of the Volterra equalizer at time n is given by

$$z_n = \mathbf{c}_n^T \mathbf{u}_n, \tag{3.16}$$

Define the nonlinear combiner output correlation matrix as

$$\tilde{\mathbf{R}} = E[\mathbf{u}_n \mathbf{u}_n^H], \tag{3.17}$$

where $E[x]$ denotes the expectation of x, and H denotes Hermitian transpose. As for the linear case, we define the error e_n at the output of the equalizer as

$$e_n = d_n - z_n, \tag{3.18}$$

where d_n is the desired response at time n. The cost function or MSE at time n is defined as

$$J_n = E[e_n e_n^*] \tag{3.19}$$

By substituting (3.18) and into (3.19) and with straightforward manipulations the following expression for the MSE is obtained

$$J_n = \sigma_d^2 - \mathbf{c}_n^T \mathbf{p} - \mathbf{p}^H \mathbf{c}_n^* + \mathbf{c}_n^T \bar{\mathbf{R}} \mathbf{c}_n^*, \tag{3.20}$$

where σ_d^2 is the variance of the desired response, $\mathbf{p} = \mathbf{E}[\mathbf{u}_n d_n^*]$ is the correlation of the tap input vector and desired response, and * denotes complex conjugate. Thus, the MSE varies with time, and is a quadratic function of the tap weights as it was for the linear equalizer. Thus, the LMS algorithm may be used for adaption as we now discuss.

3.2.1.2 The LMS Algorithm in Relation to the Nonlinear Volterra Equalizer

Since the nonlinear combining occurs before the tap weights, the outputs of the nonlinear combiner may be considered inputs to a linear adaptive filter. Also, since the MSE is a quadratic function of the tap weights, the LMS algorithm may be employed for adaption. The weight update equation is given by

$$\mathbf{c}_{n+1} = \mathbf{c}_n + \mu e_n \mathbf{u}_n^*. \tag{3.21}$$

Thus, the weight update equation is the same as for the linear equalizer. However, in the case of the nonlinear Volterra equalizer weights for nonlinear terms are included.

3.2.1.3 Convergence of the Nonlinear Volterra Equalizer

Define the correlation matrix of the inputs to the N-tap delay line whose outputs are inputs to the nonlinear combiner as (e.g., as in Fig. 3.3)

$$\mathbf{R} = E[\mathbf{y}_n \mathbf{y}_n^H], \tag{3.22}$$

where $\mathbf{y}_n$ is the tap input vector at time n. Denote the eigenvalues of $\mathbf{R}$ as λ_i and the eigenvalues of $\tilde{\mathbf{R}}$ as α_j. From [28, ch. 4] we have the following two expressions for the maximum and minimum eigenvalues of $\tilde{\mathbf{R}}$.

$$\alpha_{max} = max \frac{x^H \tilde{\mathbf{R}} x}{x^H x} : \; x \in \mathbb{C}^L, \; x \neq 0 \tag{3.23}$$

and

$$\alpha_{min} = min \frac{x^H \tilde{\mathbf{R}} x}{x^H x} : \; x \in \mathbb{C}^L, \; x \neq 0 \tag{3.24}$$

where $\mathbb{C}^L$ is complex vector space of dimension L.

Equation (3.23) states that the largest eigenvalue of $\tilde{\mathbf{R}}$ is obtained by the largest amount by which any vector is amplified by vector multiplication. Since from (3.14) $L > N$, $\mathbf{R}$ is a submatrix of $\tilde{\mathbf{R}}$, and $\mathbb{C}^N \subset \mathbb{C}^L$, we may conclude that $\alpha_{max} \geq \lambda_{\max}$ and $\alpha_{min} \leq \lambda_{\min}$. Furthermore, considering (3.12) with $2K+1$ replaced by L or N, we have that

$$\tilde{\mu}_{\max} < \mu_{\max}, \tag{3.25}$$

where $\tilde{\mu}_{\max}$ is the maximum allowable step size for the Volterra equalizer and $\mu_{\max}$ is the maximum step size for the corresponding linear equalizer. Note that $\tilde{\mu}_{\max}$ is smaller than $\mu_{\max}$ by at least the proportion of weights in the nonlinear equalizer (L) to the number of linear weights (N). In addition to the maximum step size being smaller for the nonlinear Volterra equalizer, the eigenvalue spread is potentially greater for the nonlinear Volterra equalizer as indicated above. Thus, the convergence rate is likely to be much slower for the Volterra equalizer.

3.2.1.4 Multiple-Step Size Algorithm

In the next section, the sensitivity to the step size is illustrated with an example. However, before considering such an example, it is useful to investigate a method for improving the convergence characteristics of the nonlinear equalizer. This can be done with the multiple-step size LMS algorithm [38-41]. With this algorithm each weight may be updated with a unique step size. This leads to the update equation

$$\mathbf{c}_{n+1} = \mathbf{c}_n + \mathbf{M}_D e_n \mathbf{u}_n^*. \tag{3.26}$$

where $\mathbf{M}_D$ is a diagonal matrix with the step sizes μ_i (corresponding to weight $c_i(n)$) along the main diagonal and zeros elsewhere. With the nonlinear equalizer, it has been found useful to choose a larger step size for linear terms and smaller step sizes for the nonlinear terms.

As shown in [38], for any positive choice of $\mathbf{M}_D$, the mean-square error will converge if the values of $\mathbf{M}_D$ are sufficiently small. However, if all the diagonal

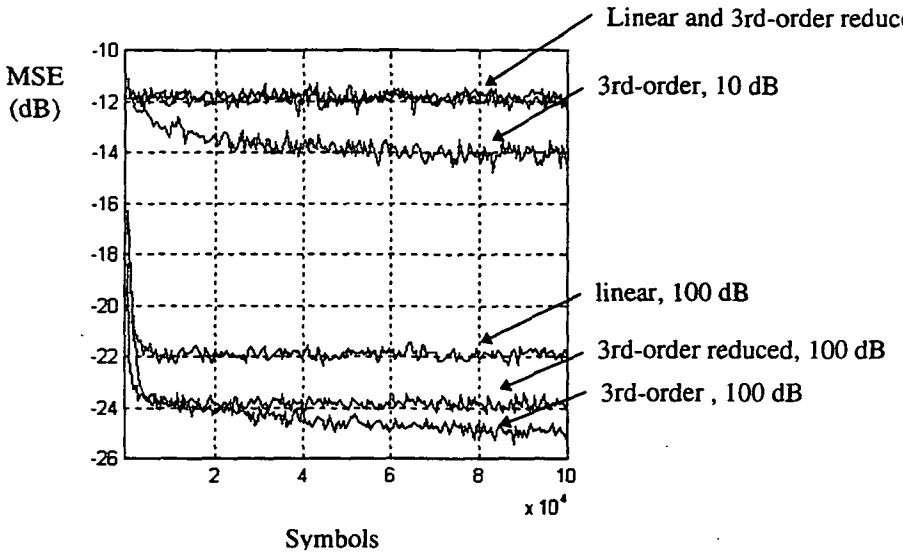

Figure 3.5. 3-tap reduced and non-reduced MSE performance for a QPSK system.

elements of $\mathbf{M}_D$ are equal then the conditions for convergence are given by (3.11) and (3.12) for convergence of the mean tap vector and excess mean-square error, respectively. Alternatively, as in the linear filtering case [41], the update constant may be replaced with $\mu\,\tilde{\mathbf{R}}$ so that all the eigenvalues become unity and all modes decay at the same rate.

3.2.2 MSE Performance

In this section, the MSE performance results obtained from computer simulation are presented. The parameters for the systems simulated are the same as that for Fig. 3.4.

Fig. 3.5 plots the mean-square error versus adaption time in symbols for various 3-tap equalizers for a QPSK system. The equalizers are a 3-tap linear, 3-

tap 3rd-order, and 3-tap 3rd-order reduced complexity. The reduced complexity equalizer has the tap weights of the terms $y_i y_j y_k^*$ set to zero for $i = k$ or $j = k$, as discussed previously. In all cases μ =10^{-3}. The MSE estimate is obtained by appropriately scaling a 512-symbol running sum of the squared error at the output of the equalizer. This method gives a quick but biased estimate of the MSE especially before the equalizer has converged. After the equalizer has converged, the method gives a good estimate of the MSE. Although ensemble averaging of learning curves is the proper method for estimating MSE, the computer time necessary for such an approach was prohibitive. The figure illustrates that for $E_b/N_0 = 10$ dB the reduced equalizer has performance (in MSE) comparable to that of the linear equalizer. As the E_b/N_0 increases, the reduced equalizer performance approaches that of the non-reduced equalizer but still has a performance loss even at very high E_b/N_0. The figure also illustrates that the convergence time for the 3rd-order equalizer is on the order of 100,000 symbols, although most of the MSE reduction is achieved in about 10,000 symbols.[4]

Fig. 3.6 illustrates the advantage of multiple-step size adaption for a QPSK system with an E_b/N_0 of 9 dB. The curve illustrating large jumps and higher overall mean-square error is for a 3-tap 5th-order equalizer with a single step size of 10^{-3}. The lower curve is for a 3-tap 5th-order equalizer with multiple-step size adaption and step sizes of 10^{-3}, 10^{-4}, and 10^{-5}, for the linear, 3rd-order, and 5th-order terms, respectively. The single step size equalizer is evidently unstable at

[4]As noted previously, these equalizers contain redundant terms, where the total number of terms is given by (3.15). The redundant terms are expected to affect the convergence rate.

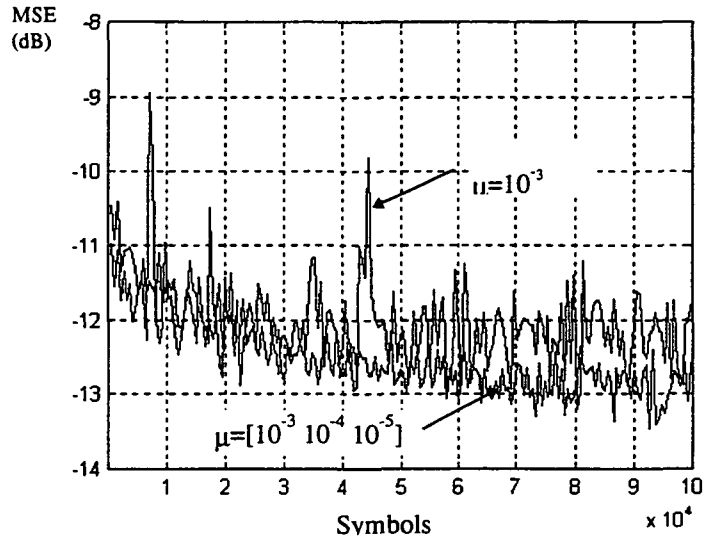

Figure 3.6. Multiple-step size adaption for a QPSK system.

this adaption rate. In order to stabilize the single step size equalizer, the step size may be reduced but at the expense of a much slower convergence time. The figure illustrates that the multiple-step size algorithm lessens the penalty of a slower convergence rate while improving performance in MSE.

Fig. 3.7 illustrates the noiseless MSE performance of the 7-tap linear, 7-tap 3rd-order, and 7-tap linear with 3-tap 3rd-order equalizers. As is clear from Fig. 3.4, the 7-tap 3rd-order equalizer gives the best MSE performance with greater than 5 dB improvement compared to the linear equalizer. The 7-tap linear with 3-tap 3rd-order equalizer obtains approximately 1.5 dB improvement compared to the linear equalizer.

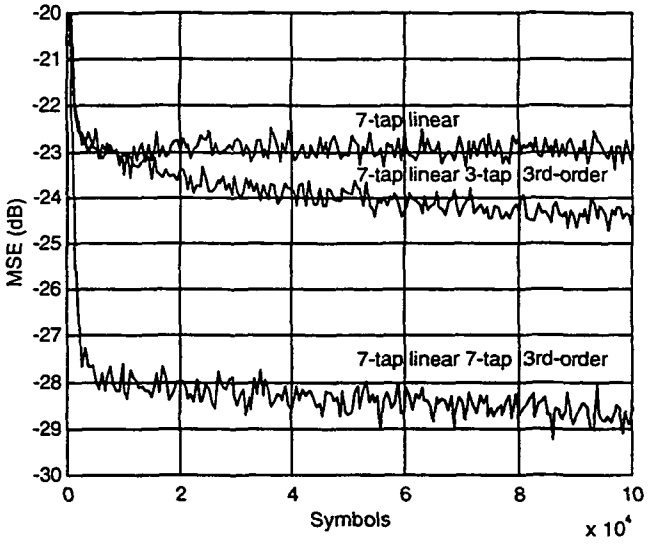

Figure 3.7. Noiseless MSE performance for a QPSK system.

3.2.3 Probability of Error Performance

The probability of bit error (P_b) performance for QPSK and 8-PSK systems is shown in Fig. 3.8. The system parameters are the same as for Fig. 3.4. For the QPSK system, the equalizers are a 5-tap linear, and 5-tap 3rd order. For the 8-PSK system the equalizers are a 7-tap linear, 7-tap 3rd-order, and 7-tap linear with 3-tap 3rd-order. Although according to Figs. 3.5 and 3.7, the MSE performance for a 3rd-order equalizer is significantly better than for a linear device, there is no improvement in P_b for QPSK.

Some insight may be obtained by looking at a scatter plot of the equalizer output, Fig. 3.9. For this case it has been determined that the signal to distortion ratio is approximately 15 dB whereas the symbol energy to noise spectral density ratio is 13 dB (i.e., E_b/N_0 =10 dB). With the distortion below or at a comparable

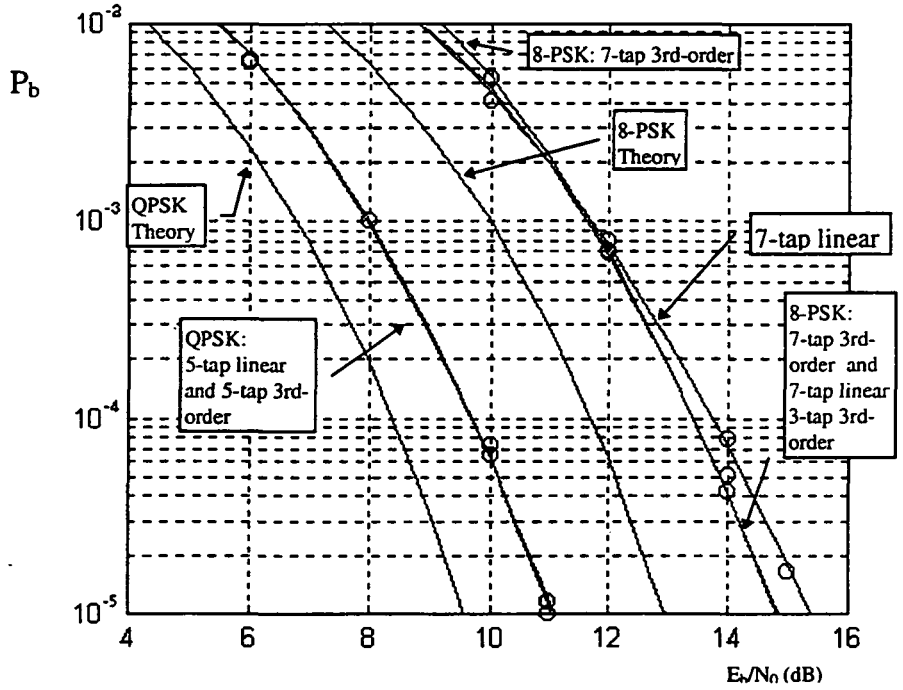

Figure 3.8. QPSK and 8-PSK probability of error performance.

level to the noise, the equalizer can significantly reduce the mean-square error by reducing the component of the noise in the radial direction, however, this noise reduction is orthogonal to the direction which leads to a decision error. Thus, although the mean-square error is significantly reduced, the probability of error is not improved. This result is unique to M-PSK when the distortion is comparable to or below the noise level.

In the 8-PSK case, the noise level is below the distortion level at $P_b = 10^{-5}$ (see Fig. 3.8) and the 3rd-order nonlinear equalizers give a 0.5 dB improvement in P_b compared to the linear equalizer. The 7-tap 3rd-order equalizer (343 3rd-order terms) shows a slightly degraded performance compared to the linear equalizer at

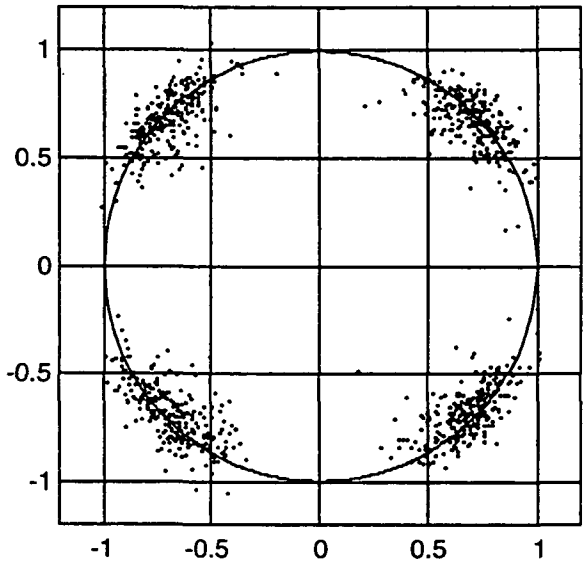

Figure 3.9. Scatter plot of 7-tap 3rd-order equalizer output.

low E_b/N_0 due to noise enhancement at the nonlinear equalizer output. The 7-tap linear, 3-tap 3rd-order equalizer (27 3rd-order terms) gives the same performance improvement at high E_b/N_0 at a much lower complexity compared to the former.

Thus far only constant modulus systems have been considered. Fig. 3.10 illustrates the performance in P_b for a 16-QAM system. For this case the satellite parameters are the same as for Fig. 3.4, however the transmitter and receiver differ. The transmitter has a square root raised cosine pulse shape with 100% excess bandwidth and the receiver is matched to the transmitter. In the case of PSK, in order to minimize the nonlinear distortion, it is desirable to transmit constant modulus signals, so it is typical to transmit with a rectangular pulse shape. However, since QAM employs multi-level signals, they are not constant modulus. Since the modulation is not constant modulus, a square root raised

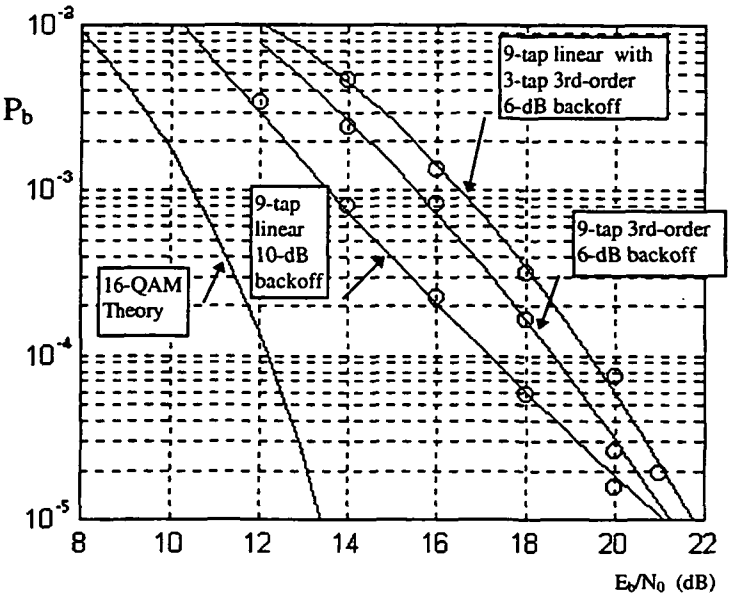

Figure 3.10. 16-QAM probability of bit error performance.

cosine pulse shape is employed at the transmitter and receiver as is usual for this scheme.

Several equalizer configurations are considered: 9-tap linear, 9-tap 3rd-order (9 linear and 729 3rd-order terms) , and 9-tap linear with 3-tap 3rd-order (9 linear, and 27 3rd-order terms). For the linear equalizer, the 16-QAM system must be operated at a large backoff for low P_b. However, the 3rd-order nonlinear equalizers can operate at 6 dB backoff and still maintain a low P_b.

The 9-tap 3rd-order equalizer has approximately a 0.5 dB better performance than the 9-tap linear with 3-tap 3rd-order equalizer, however the former is an impractical device. The significant improvement for the nonlinear equalizers is in link margin.[5] The E_b/N_0 for the curves in Fig. 3.10 is measured at the receiver.

[5]Link margin is defined as the energy in the received signal above a pre-defined level.

Consequently, a system with X dB backoff will have X dB less E_b/N_0 at the receiver than a system with 0 dB backoff. Therefore, the 9-tap linear, 3-tap 3rd-order equalizer at 6 dB backoff gives approximately a 3.5 dB improvement in link margin compared to the 9-tap linear device at 10 dB backoff.

The nonlinear equalizer improvement in P_b for 16-QAM versus PSK systems is attributed to two dominant factors: 16-QAM operates at a higher E_b/N_0 than the PSK systems and QAM is subject to constellation warping [10] so that the nonlinear effects of the HPA are more significant. In contrast to constellation warping, PSK suffers only a constellation rotation which is reversed by the phase synchronizer. Note also that the link penalty for the PSK systems considered is not nearly as severe compared to the QAM system.

3.3 FSE-Volterra Equalizer

Fig. 3.11 illustrates the noiseless received signal spectrum and receive filter response for a QPSK system as in Fig. 3.1. The receive filter is matched to the transmitter which has a rectangular impulse response, and the remaining system parameters are identical to those for the systems of the previous section. As is evident from the figure, the receive filters of the previous section, which were matched to the transmitter, are severely mismatched to the channel due to presence of the pre-filter, TWT, and post-filter. Thus, the signal to noise ratio at the output of the rectangular receive filter is suboptimal. Biglieri *et al.* [42] have derived the optimal linear receive filter for nonlinear channels. Unfortunately, the

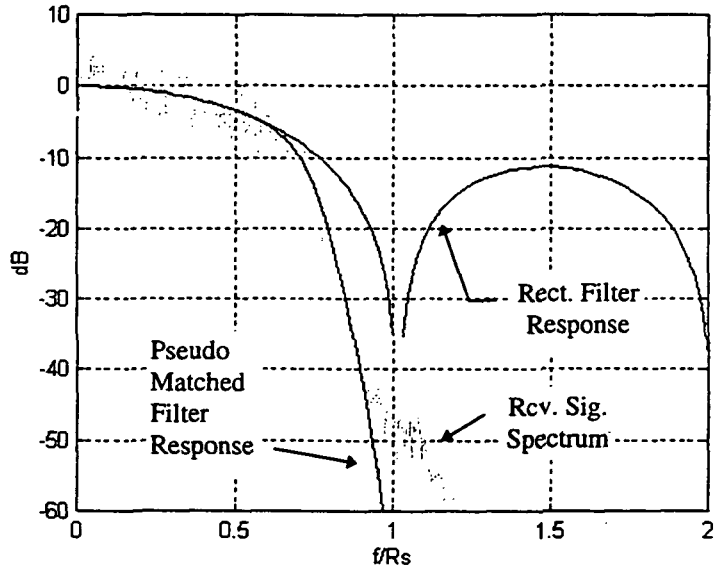

Figure 3.11. Received signal spectrum, rectangular filter response, and pseudo-matched filter response for QPSK.

channel statistics including state transition probabilities must be known and are not easily obtained.

Also shown in Fig. 3.11 is the response of a filter equivalent to the combined response of the pre-filter, post-filter, and transmit filter (in this case the filters are Butterworth, Butterworth, and rectangular, respectively). This filter will be called the pseudo-matched filter. As is evident, the pseudo-matched filter matches the received signal spectrum much better than the rectangular receive filter. In Section 3.3.3, the performance of the pseudo-matched filter followed by a T-spaced equalizer and Volterra equalizer is compared to that of the FSE-Volterra equalizer for 16-PSK.

Alternatively, an FSE has the capability of an analog filter so that it may be configured as the best linear receiver [25]. Also, the FSE can compensate for sample timing offset. Konstantinides and Yao [24], using a state machine description, have developed a method for obtaining the correlation matrix $\mathbf{R}'$ and cross correlation vector $\mathbf{p}'$ of a nonlinear satellite communications channel. The optimum FSE tap weights may then be obtained from (3.4). For the case of unknown channel statistics an adaptive FSE can adaptively realize the optimum linear filter [25]. The FSE is equivalent to a matched filter followed by a T-spaced tapped delay line.

The following section discusses in greater detail the salient features of FSE equalizers. Next, the receive filter is replaced with an FSE so that the satellite receiver consists of an FSE followed by a Volterra equalizer. The operation of this structure is then discussed. Finally, the performance of the FSE-Volterra receiver is evaluated for various M-PSK scenarios.

3.3.1 FSE Background

The following FSE discussion follows that of [25]. First, consider the symbol-spaced (synchronous) equalizer with noiseless input samples $x(nT+\tau)$ and taps c_k, where τ represents sample timing error. The output spectrum of the equalizer is given by

$$\begin{aligned} Z_T(f) &= \sum_k c_k e^{-j2\pi fT} \sum_l X\left(f+\frac{l}{T}\right) \exp\left[-j\left(f+\frac{l}{T}\right)\tau\right] \\ &= C_T(f)X_T(f), \end{aligned} \tag{3.27}$$

where $X_T(f)$ is the folded (aliased) spectrum of the input signal, and $C_T(f)$ is the frequency response of the equalizer tap weights. Since the spectrum of the equalizer filter is periodic, $C_T(f) = C_T(f + l/T)$, the synchronous equalizer can only act to modify the folded spectrum $X_T(f)$, as opposed to directly modifying $X(f)e^{-j2\pi f\tau}$. Thus, if, because of severe phase distortion and a poor choice of τ, a null is created in the aliased part of the spectrum then the equalizer can only synthesize a large gain in the region. Unfortunately, this leads to a severe performance degradation because of noise enhancement.

The fractionally-spaced equalizer is similar to the transversal filter; however the tap spacing is given by $T' < T$ such that the input signal $x(t)$ into the FSE is sampled at greater than twice the highest frequency component. For practical purposes, T' is chosen to be a rational fraction of T (it is typical to choose $T' = T/2$). Since data decisions are made at the symbol intervals, the output of the FSE is sampled at the rate $1/T$. The output spectrum of the FSE prior to symbol-rate sampling is given by

$$Z_{T'}(f) = \sum_k c_k e^{-j2\pi fT'} \sum_l X\left(f + \frac{l}{T'}\right) \exp\left[j\left(f + \frac{l}{T'}\right)\tau\right] \quad (3.28)$$

$$= C_{T'}(f) X_{T'}(f). \quad (3.29)$$

As in the previous case, the equalizer spectrum is periodic, $C_{T'}(f) = C_{T'}(f + l/T')$. However, the sample rate $1/T'$ is greater than twice the highest frequency component in $X(f)$. Consequently, only the $l = 0$ term in (3.28) is nonzero in the interval $|f| \leq 1/(2T')$. Thus, the FSE output spectrum becomes

$$Z_{T'}(f) = C_{T'}(f) X(f) e^{-j2\pi f\tau}, \quad |f| \leq \frac{1}{2T'}. \quad (3.30)$$

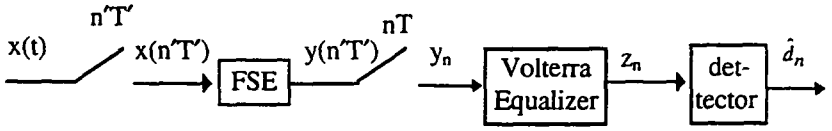

Figure 3.12. FSE-Volterra receiver.

From (3.30) it is clear that $C_{T'}(f)$ modifies $X(f)e^{-j2\pi f\tau}$ before aliasing and $C_{T'}(f)$ can compensate for sample timing offset.

After sampling the equalizer output at the rate $1/T$ the periodic spectrum is given by

$$Z_T(f) = \sum_l C_{T'}\left(f + \frac{l}{T}\right) X\left(f + \frac{l}{T}\right) \exp\left[-j\left(f + \frac{l}{T}\right)\tau\right]. \tag{3.31}$$

It is evident that (3.31) is the sum of equalized components while (3.27) is the sum of equalized aliased components. Thus, the FSE has the capabilities of an analog filter and can be configured as the optimum linear receiver.

3.3.2 FSE-Volterra Receiver

The FSE-Volterra receiver is shown in Fig. 3.12. The input signal $x(t)$ to the receiver is the output of the channel plus noise and is sampled at the rate $1/T'$. The receive filter (e.g., as in Fig. 3.1) is replaced with an FSE. The output of the FSE $y(n'T')$ is sampled at the symbol rate and is input to the Volterra equalizer. The detector then outputs an estimate $\hat{d}_n$ of the transmitted symbol d_n.

Adaptive setting of the FSE weights is obtained via the LMS algorithm where tap updates occur at the rate $1/T$. Since the FSE is a linear adaptive filter, and the nonlinear channel output characteristics can be expressed in the form of a

correlation matrix $\mathbf{R}'$ and cross-correlation vector $\mathbf{p}'$, then the previous discussions on the LMS convergence apply.

The adaption strategy is as follows. First the Volterra tap weights are initialized to an impulse function (i.e., the center linear tap is set to unity and all others set to zero). The initial FSE tap weights are set such that the FSE is matched to the transmit filter. The FSE is then adapted via the LMS algorithm until the MSE, at its output, is no longer decreasing. The FSE adaption is then turned off. Finally, the Volterra equalizer is adapted until the MSE, at its output, is no longer decreasing.

3.3.3 FSE-Volterra Receiver Probability of Error Performance

Probability of bit error performance curves for the FSE-Volterra receiver of Fig. 3.12 is shown in Figs. 3.13-3.15. For each of the curves, the number of equalizer taps was chosen such that increasing the number of taps did not provide performance improvement. Also, the performance of the FSE-Volterra equalizer is compared to the synchronous equalizer and FSE. The system parameters are the same as for Fig. 3.4.

The computer simulations which evaluated the following systems used discrete-time signals, where the sample rate was four samples per symbol. In the case of rectangular transmit and receive filters, this resulted in a sampling-phase error at the detector causing a loss of approximately 0.15 dB. That is, the optimum sampling point at the receive filter output was in error such that the signal to

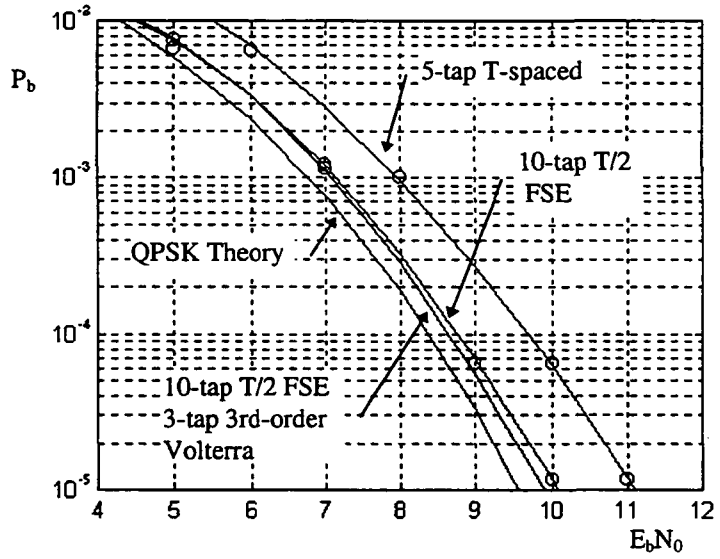

Figure 3.13. FSE-Volterra QPSK probability of bit error performance.

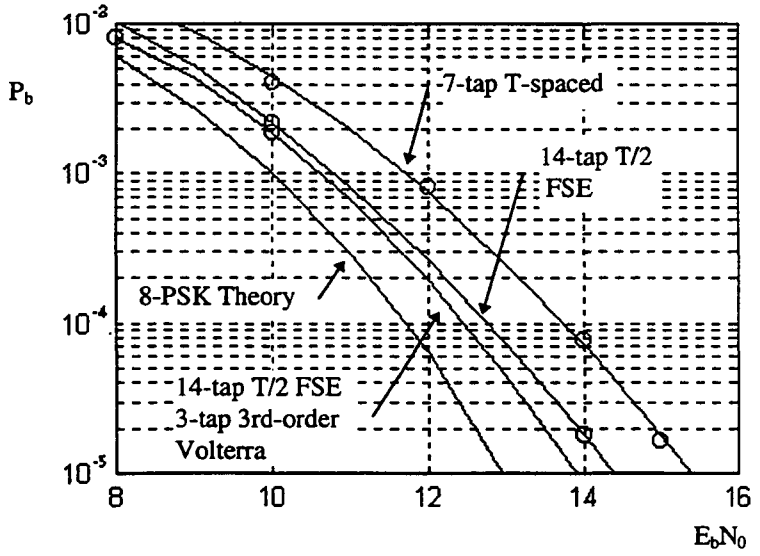

Figure 3.14. FSE-Volterra 8-PSK probability of bit error performance.

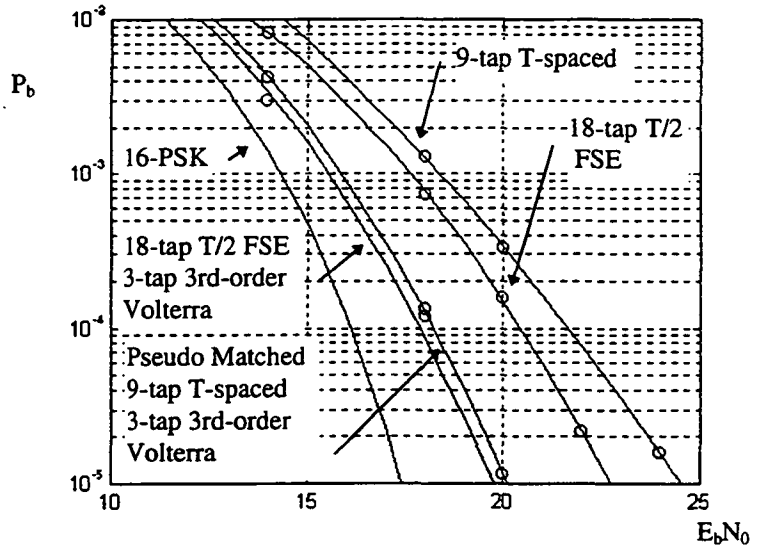

Figure 3.15. 16-PSK FSE-Volterra and pseudo-matched filter probability of error performance.

noise ratio was decreased by 0.15 dB. For the case of the pseudo-matched filter the sampling error loss was approximately 0.18 dB. The curves in Figs. 3.13-3.15 show the error rates obtained directly from computer simulation and have not been adjusted for sampling phase error.

Fig. 3.13 summarizes the QPSK probability of bit error performance. The FSE-Volterra equalizer shows a small gain in performance relative to the FSE, approximately 0.2 dB at $P_b = 10^{-5}$. Also it is evident that the FSE and FSE-Volterra obtain a significant improvement in performance ($\sim$1 dB at $P_b = 10^{-5}$) compared to the 5-tap linear equalizer following the receive filter. In this case, the FSE-Volterra P_b performance is approximately 0.3 dB away from theory.

For 8-PSK, Fig. 3.14 shows that the FSE-Volterra equalizer gain relative to the FSE is better than for the QPSK case. The FSE-Volterra P_b performance is

within 1 dB from theory and improves upon the FSE by 0.5 dB. The FSE gains 1 dB relative to the synchronous equalizer.

Fig. 3.15 illustrates that the FSE-Volterra P_b performance for 16-PSK is improved approximately 3-dB relative to the FSE equalizer and is 2-dB away from theory. The FSE gains over 1 dB relative to the synchronous equalizer. For this case, it was found that the communications link was incapable of providing error-less communication, even without noise. Thus, linear equalization provided a significant performance improvement. In addition, the noise level was low enough such that Volterra equalizer could effectively remove some of the nonlinear distortion without severely enhancing the noise.

Also, the pseudo-matched filter followed by a 9-tap T-spaced equalizer and 3-tap 3rd-order Volterra equalizer gives performance within 0.4 dB of the FSE-Volterra equalizer. When the pseudo-matched filter curve is adjusted in order to compensate for sampling-phase error, the difference in performance is approximately 0.2dB. This illustrates that when the response of the channel filters is known in advance, the FSE equalizer is not necessary (i.e., a pseudo-matched filter with a T-spaced equalizer gives similar performance).

3.4 Chapter Summary

This chapter has reviewed adaptive linear filter theory relevant to the understanding of the adaptive Volterra equalizer. The convergence of the adaptive (LMS algorithm) Volterra equalizer was analyzed. It was found that the conver-

gence rate is slower for the Volterra equalizer than for a linear device with an equivalent length tapped delay line. Consequently, a multiple-step size adaption method was employed which improved the convergence rate while maintaining good MSE performance.

The MSE and probability of bit error performance was characterized for the adaptive Volterra equalizer. It was found that a significant improvement in MSE did not necessarily translate to a commensurate improvement in probability of error. Also, an FSE-Volterra equalizer gives improved performance relative to the Volterra equalizer when the receive filter is not matched to the channel. The Volterra and FSE-Volterra equalizers give significant performance improvement for multi-level (i.e., QAM) and modulation schemes which require high signal to noise ratios. Also, the pseudo-matched filter followed by a T-spaced equalizer and Volterra equalizer gives similar performance to the FSE-Volterra equalizer.

Chapter 4

MLSD RECEIVERS FOR NONLINEAR SATELLITE CHANNELS

Forney [23] has shown that the optimum receiver for a linear channel with ISI is a whitened matched filter followed by a nonlinear processor known as the Viterbi algorithm [26]. A maximum-likelihood receiver for binary sequences over bandlimited nonlinear channels was derived by Campbell [21]. Benedetto *et al.* [22] derived the optimal receiver for a nonlinear bandlimited satellite channel and arbitrary modulation formats. Although the approach for deriving the receivers in [22] and [21] differed, both have a similar structure; a bank of matched filters followed by a Viterbi detector. In [21] a performance analysis was performed for a typical satellite system. It was found, for binary sequences, the MLSD receiver required an excess of approximately 1 dB of downlink SNR to equal the performance of binary transmission over an AWGN channel. The performance of the receiver derived in [22], however, was not evaluated for a specific satellite channel. In this chapter, the performance of a matched filter bank Viterbi detector (MFB-MLSD) receiver, as in [22], is evaluated for QPSK and 8-PSK satellite channels.

Because of the matched filter bank, the MFB-MLSD receiver may be high in complexity. Consequently, the performance of a suboptimal receiver, a single receive filter followed by Viterbi detector (SF-MLSD), is also studied. The SF-MLSD receiver represents the typical receiver where there is a single receive filter. Since the channels of the previous chapter employed a single receive filter, the

SF-MLSD probability of error performance serves as a lower bound to that of the Volterra and linear equalizers, where the linear equalizers follow a fixed receive filter. Furthermore, the MFB-MLSD lower bounds the probability of error performance of the SF-MLSD receiver and the FSE-Volterra equalizers of the previous chapter.

In order to minimize the effects of nonlinear distortion, it is typical to transmit rectangular pulse shapes through satellite channels. However, because of the satellite filters, this may result in a net signal loss. It may be advantageous to transmit bandlimited signals, such as signals with a square root raised-cosine pulse shape. It is found that a nonlinear bandlimited satellite channel which employs square root raised-cosine pulse shapes with a SF-MLSD receiver gives little performance loss relative to the same system with an MFB-MLSD receiver. Also, the performance of the MFB-MLSD receiver for the case of square root raised-cosine signals indicates that non-constant envelope signals through nonlinear satellite channels are capable of giving performance close to that of the AWGN channel.

The complexity of the MFB-MLSD receiver, specifically the filter bank, grows exponentially with the memory of the communications channel. Consequently, both rectangular and square root raised-cosine pulse shapes are considered, with the goal of minimizing the channel memory. It is found that for high excess bandwidth (i.e., 100%) the square-root raised cosine pulse shape results in minimal channel memory. Lower amounts of excess bandwidth result in larger memory.

The organization of this chapter is as follows. First, two representative nonlinear satellite channels and associated parameters are described. These systems will be referenced throughout this chapter. Next, an SF-MLSD receiver is described followed by a brief description of the Viterbi algorithm. The Viterbi algorithm discussion will be useful when describing the relationship between the Viterbi algorithm path metric calculations and the matched filter bank of the MFB-MLSD receiver. Pulse shaping and its effect on channel memory is then discussed. Next, a log-likelihood development of the MFB-MLSD receiver, as in [22], is reviewed. Next, the probability of error performance of these MLSD receivers is evaluated via computer simulation and calculation of d_{min}. The method of calculating d_{min}, as in [22], is also described. The calculation of $d_{\min}$ and the computer simulation employ state machine models of the communications channels as described in Chapter 2.

4.1 Satellite Channel Configurations

The discussion and analysis in this chapter will reference two representative nonlinear satellite communication systems. The first system is similar to those of the previous chapter which employed rectangular transmit and receive filters. The second system employs square root raised-cosine filters. The first system has the disadvantage of signal loss due to the satellite filters. For the second system, it is assumed that the transmitted signal spectrum fits sufficiently inside the satellite's filters. Thus, for this case the signal is unaffected by the pre- and post-filters and

there is no signal loss. For both systems, when an SF-MLSD receiver is employed, the receive filter is matched to the transmitter. The systems are described in more detail in the following paragraphs.

System I is identical to the channel studied in the previous chapter. That is, the satellite pre- and post-filters are 6th order butterworth with a 3 dB cutoff frequency of 0.75 R_s, where the symbol r $R_s = 1/T$, and T is the symbol interval. The TWT is driven at 0 dB backoff. The transmit filter has a rectangular pulse shape (i.e., $h_T(t) = 1, 0 \leq t < T$). The study of this system is interesting since the performance results of the SF-MLSD and MFB-MLSD receivers may be compared to the performance results of the previous chapter.

System II employs a square root raised-cosine pulse shape at the transmitter with 100% excess bandwidth. For this channel, the satellite pre- and post-filters are replaced with wires. The TWT is driven at 0 dB backoff. System II is unrelated to System I. For example, it is not assumed that a signal of 100% excess bandwidth is unaffected by filters with a 3 dB cutoff frequency of 0.75 R_s. It is interesting to study this system since the performance of the MFB-MLSD receiver will indicate the performance of an optimum receiver for non-constant modulus signals through a nonlinear satellite channel. The performance of the SF-MLSD receiver for this channel will indicate the performance of a practical receiver for this system configuration.

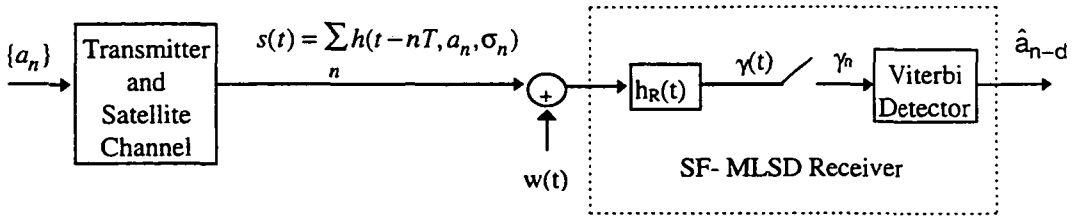

Figure 4.1. Satellite communication system with SF-MLSD receiver.

4.2 Single Filter MLSD Receiver

The SF-MLSD receiver is shown in Fig. 4.1. The input to the transmitter is the information sequence $\{a_n\}$, where $a_n \in \{0, 1, \ldots, M-1\}$, and M is the symbol set size. The waveforms $h(t, a_n, \sigma_n)$ are defined as in Section 2.3. The inputs to the detector $\{\gamma_n\}$ are the sampled outputs of the receive filter $h_R(t)$. The noise $w(t)$ is assumed to be an AWGN process. Also, the receive filter $h_R(t)$ is matched to the transmitter, as is typical in satellite communication systems. If the information sequence $\alpha_N = \{\alpha_n, \alpha_{n-1}, \ldots, \alpha_{n-N+1}\}$ is determined to be the maximum-likelihood transmitted sequence given that the sequence $\gamma_N = \{\gamma_n, \gamma_{n-1}, \ldots, \gamma_{n-N+1}\}$ was received, then the output of the Viterbi detector is $\hat{a}_{n-d} = \alpha_{n-d}$, where d is the decoding delay.

The following subsections discuss two important elements associated with the SF-MLSD receiver of Fig. 4.1. First, the Viterbi algorithm is discussed with respect to the nonlinear satellite channel. The complexity of the Viterbi algorithm is determined by the number of states assumed by the communications channel and the path memory. However, the path memory is directly proportional to the memory of the communications channel. Consequently, the effect of the pulse

shape on the communications system memory is then discussed. This includes evaluating the communication system memory for rectangular and raised-cosine pulse shapes.

4.2.1 The Viterbi Algorithm

The Viterbi algorithm [26] was developed for decoding of convolutional codes and later employed for detecting of ISI sequences [23, 43, 44]. Here, the Viterbi algorithm is discussed in the context of the satellite communications model of Fig. 4.1.

Suppose that the sequence γ_N of N (N large) detector inputs is received and the communication system has a memory of L symbols. The receive filter output sample is given by

$$\gamma_n = s_n + \eta_n \tag{4.1}$$

where s_n is the output of the satellite channel filtered by the receive filter and sampled at time nT. The noise sample η_n is obtained by filtering the noise process $w(t)$ and sampling. Since in this work $h_R(t)$ is either a rectangular filter or a square root raised-cosine filter, the noise samples η_n at the output of the receive filter are independent.

The objective of the MLSD receiver is to choose the sequence

$$\mathbf{a} = \{a_n,\ a_{n-1},\ \ldots,\ a_{n-N+1}\} \tag{4.2}$$

which maximizes the conditional probability density (or likelihood) function

$$p[\gamma|\mathbf{a}] = p[\gamma_n,\ \gamma_{n-1},\ \ldots,\ \gamma_{n-N+1}|a_n,\ a_{n-1},\ \ldots,\ a_{n-N+1}] \tag{4.3}$$

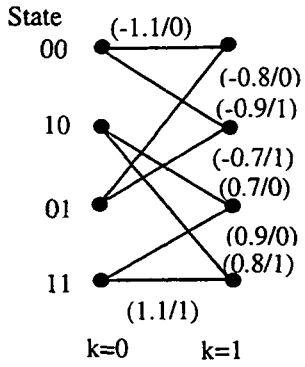

Figure 4.2. Trellis diagram for communications system with $M = 2$ and $L = 2$.

which is the probability of the receive filter output sequence conditioned on the input sequence. This is known as the maximum-likelihood criterion. Because the noise samples at the output of the receive filter are independent, the likelihood function (4.3) can be expressed as a product of marginal densities

$$p[\gamma|\mathbf{a}] = \prod_{k=0}^{N-1} p(\gamma_{n-k}|a_{n-k},\ a_{n-k-1},\ \ldots,\ a_{n-k-L}) \tag{4.4}$$

Since the noise process $w(t)$ is assumed to be AWGN, the objective of the detector reduces to choosing the sequence $\mathbf{a}$ which minimizes the Euclidean distance to the received sequence γ. That is, choose $\mathbf{a}$ such that the squared Euclidean distance

$$d^2(\gamma, \mathbf{s}) = \sum_{k=0}^{N-1} (\gamma_{n-k} - s(a_{n-k}, \varphi_{n-k}))^2 \tag{4.5}$$

is minimized, where $s(a_n, \varphi_n)$ is the noiseless input to the detector given that the data sequence $\mathbf{a}$ was transmitted and the state of the channel is σ_n with input a_n, and $\sigma_n = \{a_{n-1},\ a_{n-2},\ \ldots,\ a_{n-L}\}$.

As discussed in Section 2.3 (FSM model), a communications channel with a memory of L symbols has M^L states. Such a system may be described as a trellis

with M paths entering and leaving each state. For example, Fig. 4.2 illustrates a 4-state trellis with $M = 2$ and $L = 2$. The labels on each branch of the trellis indicate the output of the satellite channel followed by the input. For example, the label on the branch from state 00 to state 10 indicates that when the system is in state 00 and an input symbol 1 is transmitted, the system goes to state 10 with an output -0.9 volts.

The Viterbi decoder utilizes the trellis structure to efficiently provide a maximum-likelihood estimate $\hat{\mathbf{a}}$ of the transmitted sequence from which an estimate $\hat{a}_{n-d}$ of the transmitted symbol is derived. At each stage k of the trellis, a branch metric

$$b(a_{n-k}, \varphi_{n-k}) = (\gamma_{n-k} - s(a_{n-k}, \varphi_{n-k}))^2 \tag{4.6}$$

is calculated for each branch exiting a particular state at stage k of the trellis. The branch metric is then added to the path metric $p_{m,k}$ corresponding to state m and stage k. Next, the path entering each state with the smallest path metric is retained and the remaining paths are discarded. Since for two or more paths entering a particular state the paths will be identical from that point forward, a maximum-likelihood decoder may discard all paths entering a state, except the path with the smallest metric, without loss in performance. The number of decoding stages, d, required for near optimum performance is approximately $5L$ [17]. After d decoding stages the path with the minimum path metric determines the maximum likelihood estimate $\hat{a}_{n-d}$.

Table 4.4. State table for System I with rectangular receive filter.

a_n	σ_n						output real	output imag.
0	0	0	0	0	0	0	-0.735	-0.756
0	0	0	1	0	0	0	0.487	-0.809
0	0	0	2	0	0	0	-0.804	0.534
0	0	0	3	0	0	0	0.517	0.515
0	0	1	0	0	0	0	-0.849	-0.743
0	0	2	0	0	0	0	-0.689	-0.857
0	0	3	0	0	0	0	-0.787	-0.810
0	0	0	0	1	0	0	-0.434	-0.841
0	0	0	0	2	0	0	-0.840	-0.371
0	0	0	0	3	0	0	-0.461	-0.421
0	1	0	0	0	0	0	-0.701	-0.775
1	0	0	0	0	0	0	-0.737	-0.748
0	0	0	0	0	1	0	-0.730	-0.761
0	0	0	0	0	0	1	-0.744	-0.747

4.2.2 Pulse Shaping and Channel Memory

Because the satellite channel is bandlimited, this leads to a communication system with memory. The channel memory may be assessed by looking at the state table description of the communications channel. Table 4.4 contains a partial listing of the state table for a satellite channel with rectangular receive and transmit filters as in System I. Since this table is for QPSK modulation, the symbols, a_n, and state, $\sigma_n=\{a_{n-1}, a_{n-2}, \ldots, a_{n-6}\}$, entries are elements from the set $\{0,1,2,3\}$. The last two columns indicate the real and imaginary discrete-time outputs for the given state and input.

This table illustrates the effects of the three past and three future symbols on the center symbol, a_{n-3}. The symbol a_{n-3} determines the quadrant for the output of the particular input-state combination, and as in section 2.3.2 it will be called

the "punctual symbol." It is evident from the table that the input-state entries in the table with $a_{n-3} = 0, 1, 2, 3$, map to quadrants III, IV, II, I, respectively. The first row indicates that the output for an information symbol 0 preceded by three information symbols of 0 and followed by three information symbols of 0 is $-0.735 - j0.756$. Row number five illustrates that the output for an information symbol 0 immediately preceded by an information symbol 1 is $-0.849 - j0.743$. The distance between these points is 0.115. Using the first row as a reference, then the preceding information symbol of 1 affected the 0 input-state (row no. 1) by $10\log_{10}\left(0.115/\sqrt{E_s}\right)^2 = -18.8$ dB, where the symbol energy $E_s = 1$.

Studying the complete state table, the worst case effect of past and present symbols may be determined. Table 4.5 lists the worst case effect for past and future symbols, where L_p indicates a symbol L_p symbols in the past and L_f is defined similarly for future symbols. To measure the effect of a given symbol a_i on the punctual symbol a_p, we fix the values of the symbols surrounding a_i (including a_p), then vary a_i over all possible symbols, $\{0, 1, \ldots, M-1\}$. As we vary a_i, we measure the maximum difference, m_d, between all possible pairs of the M output levels corresponding to the a_i's. Then, the effect $\mathcal{E}_{ff}$ on the punctual symbol is defined as

$$\mathcal{E}_{ff} = 10\log_{10}\left(\frac{m_d}{\sqrt{E_s}}\right)^2 \tag{4.7}$$

The largest effect (i.e., worst case effect) is then obtained by considering all possible combinations of surrounding symbols, and choosing the maximum m_d. This

C-2

Table 4.5. Channel memory for System I.

L_p	$\mathcal{E}_{ff}$
1	< -10.6 dB
2	< -16.2 dB
3	< -26.6 dB
L_f	$\mathcal{E}_{ff}$
1	< -3.72 dB
2	< -29.1 dB

Table 4.6. State table for raised-cosine filtering

in	state				output real	output imag.
0	0	0	0	0	-0.735	-0.749
0	0	1	0	0	0.621	-0.710
0	0	2	0	0	-0.774	0.640
0	0	3	0	0	0.538	0.624
0	1	0	0	0	-0.715	-0.768
0	2	0	0	0	-0.745	-0.709
0	3	0	0	0	-0.686	-0.697
0	0	0	1	0	-0.721	-0.769
0	0	0	2	0	-0.747	-0.709
0	0	0	3	0	-0.681	-0.696
1	0	0	0	0	-0.732	-0.759
0	0	0	0	1	-0.738	-0.754

table illustrates that the significant channel memory is approximately two past symbols and one future symbol.

Table 4.6 contains a partial state table listing for a satellite communication system where the receive and transmit filters are square root raised-cosine as in System II. In this case, the punctual symbol is a_{n-1}. A lesser amount of excess bandwidth results in increasing the channel memory. For example, the square root raised cosine pulse shape is illustrated in Fig. 4.3. Because of the higher side lobes, a pulse with a smaller amount of excess bandwidth is affected more strongly by the nonlinearity and consequently influences neighboring symbols by a greater amount. Table 4.7 lists the worst case effect for past and future symbols for this channel. The significant channel memory is one past and one future symbol. In a similar fashion, it has been determined that the memory for a system using pulses with excess bandwidth of 50% is two past and two future symbols.

State tables similar to Tables 4.4 and 4.6 with $L_f = 1$, $L_p = 2$, and $L_f = 1$, $L_p = 1$, respectively, are used to simulate the performance of System I and System II in Section 4.4. This will be discussed further in Section 4.4.

4.3 Matched Filter Bank MLSD Receiver

The MFB-MLSD receiver is shown in Fig. 4.4. As in the SF-MLSD case, the input to the transmitter is the information sequence $\{a_n\}$, where $a_n \in \{0, 1, \ldots, M-1\}$ and M is the symbol set size. The receiver consists of a filter bank with a filter matched to each chip $h(t, a_n, \sigma_n)$ out of the satellite channel.

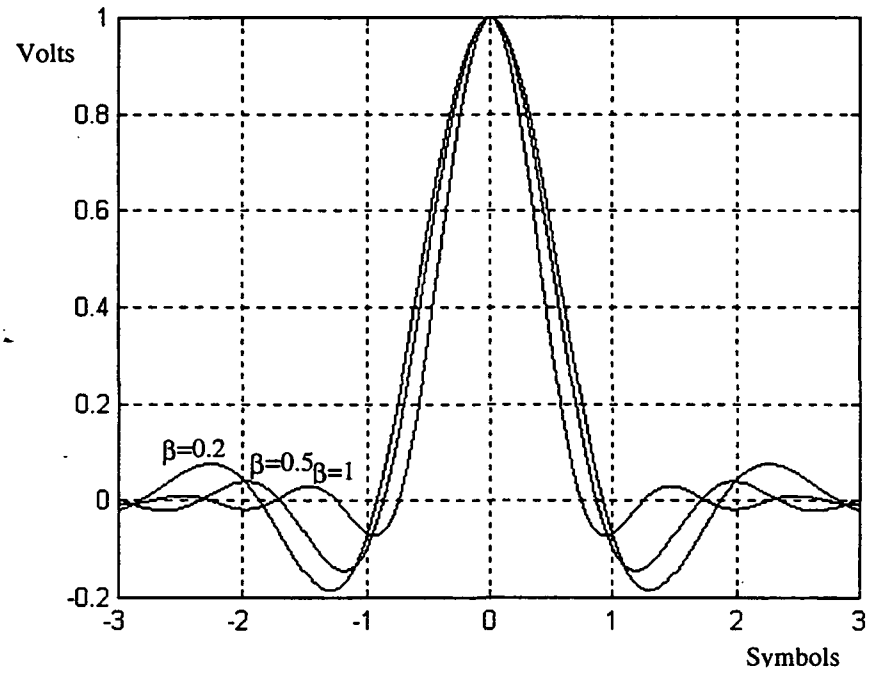

Figure 4.3. Square root raised-cosine pulse shapes.

Table 4.7. Channel memory for raised cosine filtering

L_p	$\mathcal{E}_{ff}$
1	< -16.6 dB
2	< -24.8 dB
L_f	$\mathcal{E}_{ff}$
1	< -16.6 dB
2	< -24.8 dB

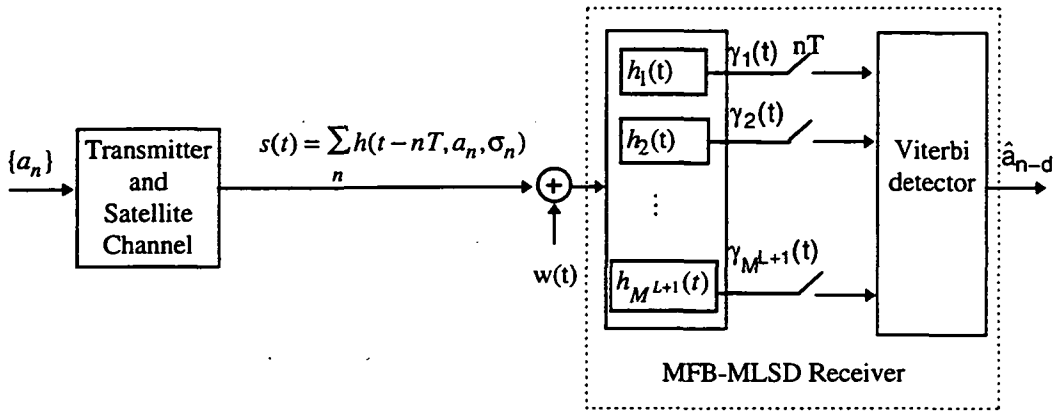

Figure 4.4. Satellite communication system with MFB-MLSD receiver.

Below, the MFB-MLSD receiver is developed using log-likelihood functions. Next, the relationship between the Viterbi algorithm path metrics and matched filter bank is described.

4.3.1 The Matched Filter Bank MLSD Receiver Derivation

The log-likelihood development of the MFB-MLSD receiver follows that of [22, ch. 10]. The objective is the maximum-likelihood detection of the finite sequence of symbols

$$\mathbf{a} = \{a_{N-1},\ a_{N-2}, \ldots,\ a_0\}, \tag{4.8}$$

where in order to simplify the notation in the following development, n as in (4.2) corresponds to $N-1$. The detection process is based on the observation of the waveform

$$r(t) = \sum_{n=0}^{N-1} h(t - nT, a_n, \sigma_n) + w(t) \tag{4.9}$$

where $N >> L$, L is the symbol memory, the state $\sigma_n = \{a_{n-1}, \ldots, a_{n-L}\}$, and $w(t)$ is an AWGN process. The chips $h(t, a_n, \sigma_n)$ are defined as in Section 2.3.

The log-likelihood function, may be derived with methods similar to those in [45, ch. 4] for the "M-hypothesis" case and is given by [22][6]

$$l_i = \frac{2}{N_0}\mathcal{R}\left\{\int_0^{NT} s_i^*(t)r(t)dt\right\} - \frac{1}{N_0}\int_0^{NT} |s_i(t)|^2\, dt, \tag{4.10}$$

where $s_i(t)$ is the noiseless transmitted waveform given that the sequence $\mathbf{a}_i = \left\{a^i_{N-1},\, a^i_{N-2}\,, \ldots,\, a^i_0\right\}$ was sent, and is given by

$$s_i(t) = \sum_{n=0}^{N-1} h(t - nT, a_n^i, \sigma_n^i). \tag{4.11}$$

The decision rule involves setting $\hat{\mathbf{a}} = \mathbf{a}_k$ whenever the log-likelihood function l_i is maximum for $i = k$. Recalling that $h(t, a_n, \sigma_n)$ is nonzero only over the interval $[0, T)$, substituting (4.11) in (4.10), and after straightforward manipulations, the likelihood function becomes

$$\begin{aligned} l_i \;=\; & \frac{2}{N_0}\mathcal{R}\left\{\sum_{n=0}^{N-1}\int_{nT}^{(n+1)T} h^*\left(t - nT, a_n^i, \sigma_n^i\right) r(t)dt\right\} \\ & -\frac{1}{N_0}\sum_{n=0}^{N-1}\int_{nT}^{(n+1)T}\left|h\left(t - nT, a_n^i, \sigma_n^i\right)\right|^2 dt \end{aligned} \tag{4.12}$$

Defining the functions

$$Z(a_n^i, \sigma_n^i) = \int_{nT}^{(n+1)T} h^*\left(t - nT, a_n^i, \sigma_n^i\right) r(t)dt, \text{ and} \tag{4.13}$$

$$\mathcal{E}(a_n^i, \sigma_n^i) = \int_0^T \left|h(t, a_n^i, \sigma_n^i)\right|^2 dt, \tag{4.14}$$

and substituting into (4.12) yields

$$l_i = \frac{2}{N_0}\mathcal{R}\left\{\sum_{n=0}^{N-1} Z(a_n^i, \sigma_n^i)\right\} - \frac{1}{N_0}\sum_{n=0}^{N-1}\mathcal{E}(a_n^i, \sigma_n^i). \tag{4.15}$$

$\mathcal{E}(a_n^i, \sigma_n^i)$ is the energy in the chips $h\,(t, a_n^i, \sigma_n^i)$. $Z(a_n^i, \sigma_n^i)$ can be obtained as the response, sampled at time $(n+1)T$, of a filter matched to $h\,(t, a_n^i, \sigma_n^i)$. This

[6]The M in "M-hypotheses" is not related to the signal set size.

implies that the number of matched filters required is M^{L+1}, which is the number chips $h(t, a_n, \sigma_n)$.

Multiplying l_i by $-N_0$, we obtain a new likelihood function

$$l_i = -2\mathcal{R}\left\{\sum_{n=0}^{N-1} Z(a_n^i, \sigma_n^i)\right\} + \sum_{n=0}^{N-1} \mathcal{E}(a_n^i, \sigma_n^i). \tag{4.16}$$

The decision rule involves setting $\hat{\mathbf{a}} = \mathbf{a}_k$ whenever the log-likelihood function l_i, given by equation (4.16), is minimum for $i = k$. A brute force method for minimization of (4.16) would require evaluating all M^N choices for the sequence $\mathbf{a}_i$. Alternatively, since the relationship between the elements of the sequence $\mathbf{a}_i$ may be described by a finite-state machine, the Viterbi algorithm can efficiently provide the maximum-likelihood estimate $\hat{\mathbf{a}}$. A program which calculates the path metrics according to equation (4.16) based on a state table description of the channel is listed in Appendix A.

4.3.2 The Matched Filter Bank and Viterbi Algorithm

As indicated above, the Viterbi algorithm can efficiently provide a maximum-likelihood estimate of the transmitted sequence $\mathbf{a}$. Also, the Viterbi algorithm was previously described in the context of the SF-MLSD receiver. In the case of the MFB-MLSD receiver, in order for the Viterbi algorithm to properly detect the sequence $\hat{\mathbf{a}}$, it is necessary for the appropriate matched filter output sample to be used in the calculation of the path metrics. From the previous discussion of the Viterbi algorithm the branch metrics were given by equation (4.6). In this case, the branch metric for the branch leaving state σ_{n-k}^i with input a_{n-k}^i is given by a

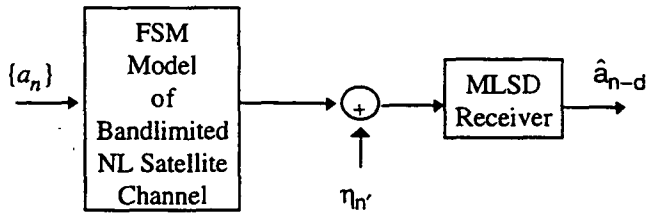

Figure 4.5. Communication system simulation model.

similar expression

$$b(a^i_{n-k}, \varphi^i_{n-k}) = -2\mathcal{R}\left\{Z(a^i_{n-k}, \sigma^i_{n-k})\right\} + \mathcal{E}(a^i_{n-k}, \sigma^i_{n-k}), \tag{4.17}$$

where $Z(a^i_{n-k}, \sigma^i_{n-k})$ is the output of the filter, sampled at time $(n+1)T$, matched to $h(t, a^i_{n-k}, \sigma^i_{n-k})$. That is, the metric for the branch leaving state σ^i_{n-k} with input a^i_{n-k} uses the output of the filter matched to the corresponding chip.

4.4 Probability of Error Performance

The probability of error (P_e) performance for the SF-MLSD and MFB-MLSD receivers is evaluated by calculating the minimum distance between channel sequences ($d_{\min}$) and by computer simulation. The minimum distance between channel sequences is derived from the state table description of the channel. For the MFB-MLSD receiver a program which reads the state table and calculates $d_{\min}$ is listed in Appendix A.

Fig. 4.5 illustrates the simulation model. For the systems with an SF-MLSD receiver, the FSM model, as described in section 2.3.2, is used to generate the channel outputs. Discrete noise samples are then added to the channel outputs. Since in the SF-MLSD case the output of the channel consists of one sample per

symbol, $n' = n$ in the figure. The MLSD receiver uses the same state table, as in the transmitter, for calculating the path metrics. The receiver then provides the maximum-likelihood estimates $\hat{a}_{n-d}$. For the systems with an MFB-MLSD receiver, the FSM model, as described in section 2.3.3, generates the oversampled channel outputs. Discrete time noise samples $\eta_{n'}$ are then added to the channel outputs. The MFB-MLSD receiver then updates the matched filter bank output at the end of each symbol period, calculates the path metrics, and provides the estimate $\hat{a}_{n-d}$. As in the previous case, the MFB-MLSD receiver uses the same state description of the channel as the transmitter. A program for simulating the nonlinear bandlimited satellite channel, as in Fig. 4.5, with an MFB-MLSD receiver is listed in Appendix A.

4.4.1 Calculation of d_{min}

An upper bound for the probability of symbol of error may be obtained using $d_{\min}$, defined as the minimum Euclidean distance between channel sequences. The nonlinear satellite channel is equivalent to a nonlinear transmitter with memory. This nonlinear transmitter nonlinearily encodes the information sequence, and the distance between the resulting "codewords" determines the probability of erroneously detecting a received sequence. As in the case of error control codes, the probability of error for the nonlinear ISI channel is bounded above by the probability of choosing an incorrect sequence that lies at the minimum distance away

from the correct sequence:[7]

$$P_e \lesssim Q\left(\frac{d_{\min}}{2\sigma}\right), \tag{4.19}$$

where σ^2 is the variance of the noise at the input to the detector. Also, (4.19) assumes independent Gaussian noise samples at the input to the detector.

For the nonlinear satellite channel with memory, finding $d_{\min}$ requires exhaustive search between all pairs of received symbol sequences. The distance structure from an all zeros path is not sufficient because, unlike the linear situation, the distance spectrum is not identical for every sequence. Since for the typical satellite channel the memory is short (e.g., Tables 4.6 and 4.7), a brute force approach to calculating $d_{\min}$ may be possible. With higher order modulations (i.e., large M), however, the brute force method may still be prohibitive.

An efficient algorithm for calculating $d_{\min}$ is given by Saxena [46] and Mulligan and Wilson [47], and a step by step description is given in [22, ch. 10]. A brief description of the algorithm is given here. For a communication system with modulation order M and memory L, the algorithm requires updating of an M^L by M^L matrix $D^{(n)}$. The elements $\delta_{ij}^{(n)}$ of the matrix $D^{(n)}$ represent the squared minimum distance between all pairs of paths diverging from some initial

[7]The bound is precisely given by

$$P_e \leq \sum_d N_d Q\left(\frac{d}{2\sigma}\right), \tag{4.18}$$

where N_d is the average number of sequences at a distance d. For large values of signal to noise ratio, the dominant error term is the one containing $d_{\min}$. We conjecture that $N_{d_{\min}}$ is on the order of 1 and possibly 2 as for M-PSK modulation, where $N_{d_{\min}}$ is the average number of sequences at a distance $d_{\min}$.

Table 4.8. Dmin values.

System	$d_{\min}$ (rct)	$d_{\min}$ (sqrc)
SF-MLSD QPSK	1.125	1.327
MFB-MLSD QPSK	1.222	1.39
SF-MLSD 8-PSK	0.566	0.684
MFB-MLSD 8-PSK	0.639	0.747

state and passing at the nth time instant through states S_i and S_j. The main diagonal represents paths which have merged, and the off diagonal terms represent minimum distances between diverged paths which have not yet merged.

The elements $\delta_{ij}^{(1)}$ of $D^{(1)}$ are set to the minimum distance between all pairs of paths (S_i, S_j) which have diverged from the same initial state and reaching the states S_i and S_j in one step. The algorithm then proceeds iteratively where at each step (i.e., $n = 2, 3, \ldots$) the elements of $D^{(n)}$ are updated according to the minimum distances of paths which have diverged and have reached states S_i and S_j at time n. The entry $\delta_{ij}^{(n)}$ is obtained by finding the predecessor path from the M^2 paths at the $(n-1)$th time instant which pass through the states S_i and S_j at the nth time instant. $\delta_{ij}^{(n)}$ is then the minimum squared distance obtained by accumulating the incremental distance from the predecessor states to states S_i and S_j. When the off diagonal terms are all greater than the main diagonal terms then there are no paths which can merge which have a smaller minimum distance than the already merged paths. This is where the algorithm stops and the minimum distance is set to the smallest diagonal entry $d_{\min} = min\left(\delta_{ii}^{(n)}\right)$.

The minimum distances were calculated for systems with SF-MLSD and MFB-MLSD receivers and are listed in Table 4.8 with rectangular (rct) and square

root raised-cosine (sqrc) filters. As expected the MFB-MLSD receivers have $d_{\min}$ greater than the corresponding SF-MLSD receivers. Also, the systems with square root raised-cosine filters have larger $d_{\min}$ than the corresponding systems with rectangular filters.

4.4.2 Computer Simulation Results

Figs. 4.6 and 4.7 illustrate the performance of SF-MLSD and MFB-MLSD receivers for System I, respectively. The symbol error performance for Q-PSK and 8-PSK are plotted and compared to the upper bound obtained by evaluating (4.19) using the appropriate entry from Table 4.8. For System I with QPSK and 8-PSK modulation, the SF-MLSD receiver is approximately 0.75 dB and 1 dB below the upper bound and 1 dB and 1.5 dB above theory at $P_e \doteq 10^{-5}$, respectively. For System I with the MFB-MLSD receiver, the performance is approximately 0.5 dB above theory for both QPSK and 8-PSK modulation.

Fig. 4.8 compares the performance of the SF-MLSD receiver to the performance obtained from an adaptive equalizer followed by a symbol by symbol detector for System I. Other than the receiver (the receiver includes the receive filter and detector), the systems are identical, however they both have rectangular receive filters. Recall that the 5-tap linear equalizer was the best performing equalizer (i.e., in the probability of error sense) for the systems of Fig. 4.8 and, of the 8-PSK equalizers, the 7-tap linear with 3-tap 3rd-order equalizer gave the best performance. In this case, the SF-MLSD receiver provides the maximum-likelihood

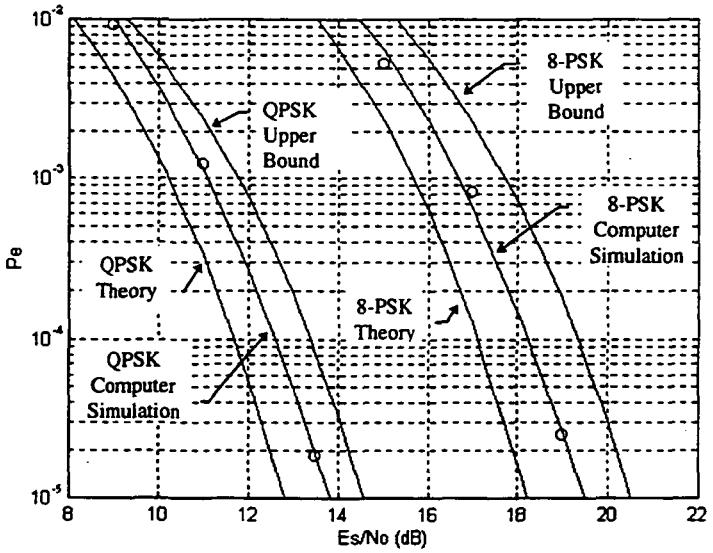

Figure 4.6. Performance of SF-MLSD receiver for System I.

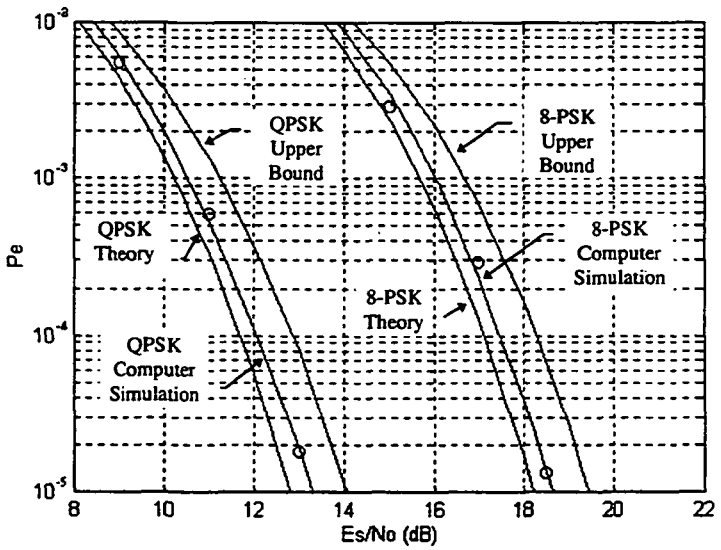

Figure 4.7. Performance of MFB-MLSD receivers for System I.

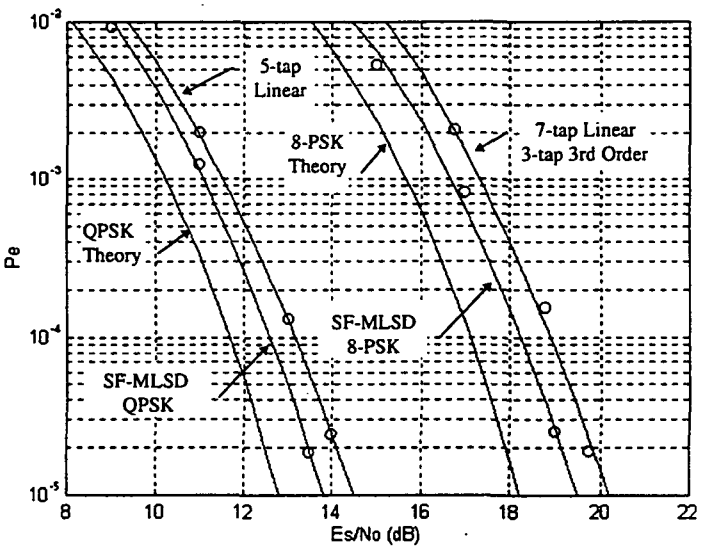

Figure 4.8. Performance of SF-MLSD receiver compared to adaptive equalizers for System I.

estimate and therefore a lower bound. For QPSK and 8-PSK, $P_e = 10^{-5}$, the performance of a T-spaced adaptive equalizer is above the lower bound by approximately 0.75 dB.

Fig. 4.9 compares the performance of the MFB-MLSD receiver to the FSE-Volterra equalizers for System I. For QPSK modulation the FSE Volterra equalizer, 10-tap T/2 FSE followed by 3-tap 3rd-order Volterra, gives performance approximately 0.1 dB above the lower bound provided by the MFB-MLSD receiver. The lower bound is approximately 0.5 dB above theory. For 8-PSK, the FSE-Volterra equalizer, 14-tap T/2 FSE followed by 3-tap 3rd-order Volterra, gives performance approximately 0.75 dB above the lower bound which is 0.5 dB above theory. Although the MFB-MLSD receiver may be impractical (i.e., the 8-PSK

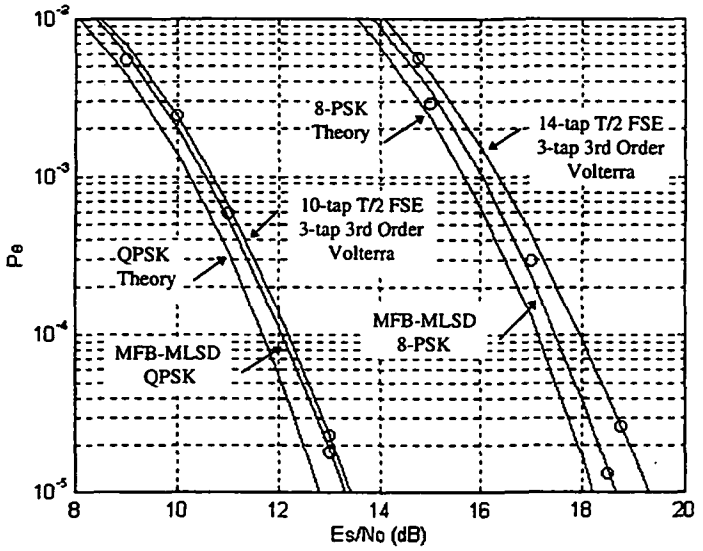

Figure 4.9. Performance of MFB-MLSD receiver compared to FSE-Volterra performance for System I.

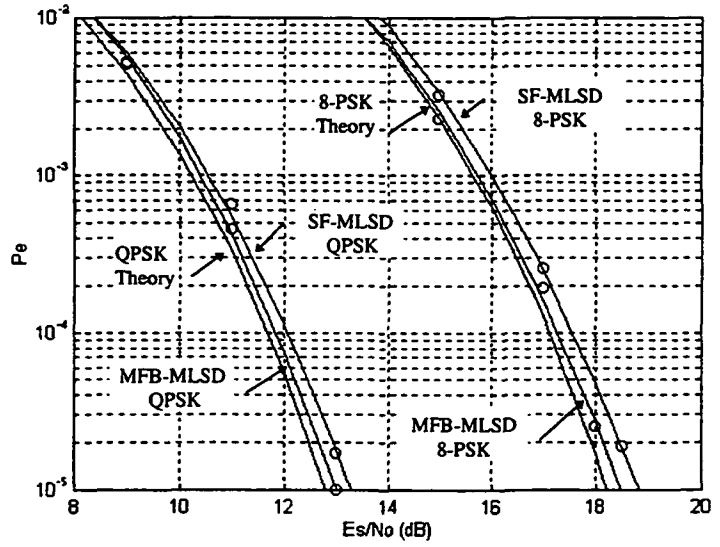

Figure 4.10. Performance of SF-MLSD and MFB-MLSD receivers for System II.

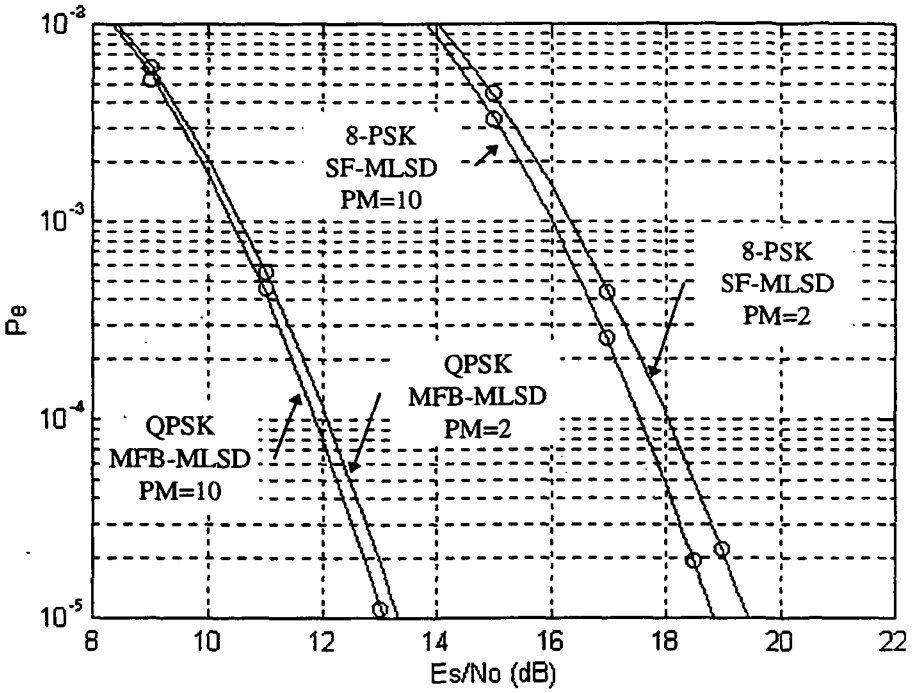

Figure 4.11. The affect of Viterbi detector path memory on probability of symbol error performance.

MFB-MLSD receiver has 4,096 matched filters), these results illustrate the usefulness of the MFB-MLSD receiver for bounding the performance of other receivers.

Fig. 4.10 compares the performance of SF-MLSD and MFB-MLSD receivers for System II. In the case of QPSK, the MFB-MLSD receiver gives performance approximately 0.25 dB away from theory. For this case, the SF-MLSD receiver has a performance loss of approximately 0.3 dB relative to the MFB-MLSD receiver. In the case of 8-PSK, the MFB-MLSD receiver gives performance approximately 0.3 dB above theory, and the SF-MLSD receiver has a performance loss approximately 0.4 dB relative to the MFB-MLSD receiver. The MFB-MLSD receiver results for System II illustrate that it is possible to obtain probability of symbol

error performance very close to theory for non-constant modulus signals through a nonlinear satellite channel.

Fig. 4.11 compares the performance of Viterbi detector path memory length for SF-MLSD and MFB-MLSD receivers. The curves in the figure are for System II. For QPSK modulation and the MFB-MLSD receiver there is a small performance loss for a path memory of 2 compared to a path memory of 10. The results are the same for 8-PSK and the SF-MLSD receiver. Similar results have been noticed for the other system-receiver combinations. This may be useful for simplifying the receiver implementation.

4.5 Chapter Summary

This chapter has described the SF-MLSD and MFB-MLSD receivers and their performance. The Viterbi algorithm, which efficiently provides a MLSD estimate, was discussed in the context of the nonlinear satellite channel with memory. Next, the symbol memory was characterized for a bandlimited satellite channel using rectangular and raised-cosine filtering. It was found that the symbol memory increased for systems using raised-cosine filtering with small amounts of excess bandwidth. Next, a log-likelihood development of the MFB-MLSD receiver, as in [22], was presented. Also, for the MFB-MLSD receiver, the relationship between the matched-filter bank and calculation of the path metrics for the Viterbi algorithm was described.

The probability of error performance of the MFB-MLSD and SF-MLSD receivers was then analyzed. First, a method of deriving the minimum distance between ISI code sequences was described. Next, the minimum distance calculations for QPSK and 8-PSK systems which employed SF-MLSD and MFB-MLSD receivers were reported. As expected, the systems with MFB-MLSD receivers had a smaller minimum distance. Also, the systems with square root raised-cosine filters had larger $d_{\min}$ than the corresponding systems with rectangular filters.

The probability of error for the receivers was then obtained via Monte-Carlo computer simulation. First, it was verified that the performance obtained from computer simulations was below the upper bound calculated from $d_{\min}$. Next, the performance of the MLSD receivers was compared to systems with adaptive equalizers. The MLSD receivers provided a lower bound for the probability of error performance of the adaptive equalizers, and helped gauge the performance of the adaptive equalizers. Finally, the performance of the MFB-MLSD receiver for System II indicate that it is possible to obtain performance near theory for non-constant modulus signals through nonlinear satellite channels.

Chapter 5

CONCLUSIONS AND FUTURE WORK

This dissertation has evaluated the performance of several receiver-based methods for mitigating the effects due to nonlinear bandlimited signal distortion present in high data rate satellite channels. The results of this dissertation may serve as a baseline for the evaluation of more complex structures based on the receiver structures evaluated in the dissertation. In addition, the performance results can help gauge the performance of hardware implementations of the receivers evaluated in the dissertation . Below, the results of each chapter are summarized and suggestions for future work are given.

5.1 Summary and Conclusions

Chapter 1 gave an overview of the dissertation and the necessary background regarding communications by satellite. Included was a discussion of a satellite's communication subsystem for a typical communications satellite. The communication subsystem was then simplified and a baseband equivalent model for evaluation of a satellite communications channel was presented. Next, the effects of the nonlinearity and bandlimiting on the digitally modulated data was demonstrated. Finally, a literature review of compensation methods for nonlinear distortion in various types of communication systems was presented.

In Chapter 2, various discrete-time equivalent models for efficiently evaluating the performance of the nonlinear bandlimited satellite channel were discussed.

First, the low-pass discrete-time equivalent model for a passband communication system was reviewed. Next, beginning with the low-pass equivalent Volterra series, a lucid development of the low-pass Volterra discrete-time model for a nonlinear satellite communications channel was presented. The resulting model is a polynomial expression relating an output symbol at discrete time n to the input symbols. Deriving this model for a particular channel is computationally intensive.

A FSM has previously been suggested for modeling a nonlinear satellite channel. The last Section of Chapter 2 specifically developed a FSM model for the nonlinear bandlimited satellite channel. Two special cases were considered: SF-MLSD and MFB-MLSD receivers. In contrast to the discrete-time Volterra model, the FSM model is easily derived. However, it may require a large amount of memory for storing the state machine. Despite the possibly large memory requirement, the FSM models are more efficient (i.e., computationally) than the Volterra model.

Nonlinear Volterra equalizers have previously been studied for equalization of nonlinear bandlimited signal distortion due to a noiseless satellite channel. However, the performance of adaptive Volterra equalizers has not been evaluated for a specific satellite channel. In Chapter 3, performance of nonlinear adaptive Volterra equalizers for mitigating the effects of nonlinear bandlimited signal distortion were studied for a nonlinear bandlimited satellite communications channel. First, the chapter reviewed adaptive linear filter theory which is important to the understanding of the adaptive Volterra equalizer. The convergence of the adaptive Volterra equalizer, which used the LMS algorithm for weight adaption, was then

analyzed. It was found that the convergence rate is slower for the Volterra equalizer than for a linear device with an equivalent length tapped delay line. Consequently, a multiple-step size adaption method was employed which improved the convergence rate while maintaining good MSE performance. Also, the MSE and probability of error performance was characterized for the adaptive Volterra equalizer. It was found that a significant improvement in MSE did not necessarily translate to a commensurate improvement in probability of error.

Also in Chapter 3, the salient characteristics of an FSE were reviewed and contrasted to those of the synchronous equalizer. Because of aliasing in the frequency spectrum, a synchronous equalizer is limited in its ability to mitigate the effects of nonlinear ISI. However, the FSE acts as an analog filter and can perform the combined functions of the receive filter and equalizer. It was found that an FSE-Volterra equalizer gave improved performance relative to the Volterra equalizer when the receive filter is not matched to the channel. The Volterra and FSE-Volterra equalizers give significant performance improvement for multi-level (i.e., QAM) and modulation schemes which require high signal to noise ratios. Also, it was found that the pseudo-matched filter followed by a T-spaced linear equalizer and Volterra equalizer gave similar probability of error performance as compared to the FSE-Volterra equalizer. This latter result indicates that when the channel filters are known a priori the FSE equalizer is not necessary.

In Chapter 4, the SF-MLSD and MFB-MLSD receivers were described and their probability of error performance was characterized for two representative

satellite channels. First, the Viterbi algorithm, which efficiently provides a MLSD estimate, was discussed in the context of the nonlinear satellite channel with memory. Next, the symbol memory was characterized for a bandlimited satellite channel using rectangular and raised-cosine filtering. It was found that the symbol memory increased for systems using raised-cosine filtering with small amounts of excess bandwidth. A log-likelihood development of the MFB-MLSD receiver, as in [22, ch. 10], was presented. Also, for the MFB-MLSD receiver, the relationship between the filter bank and the Viterbi algorithm path metric calculations was described.

Next, in Chapter 4, a method of deriving the minimum distance between ISI channel sequences was described. The minimum distance for SF-MLSD and MFB-MLSD receivers was then calculated. As expected, the systems with MFB-MLSD receivers had a larger minimum distance relative to the corresponding SF-MLSD system. Also, the systems with square root raised-cosine filters had larger $d_{\min}$ than the corresponding systems with rectangular filters. The probability of error for the receivers was then obtained via Monte-Carlo computer simulation. First, it was verified that the performance obtained from computer simulations was below the upper bound calculated from $d_{\min}$. Next, the performance of the MLSD receivers was compared to systems with adaptive equalizers. The MLSD receivers provide a lower bound for the probability of error performance of the adaptive equalizers, and helped gauge the performance of the adaptive equalizers. Finally, the performance of the MFB-MLSD receivers for systems with square

root raised-cosine filters indicate that it is possible to obtain performance close to that of the AWGN channel for non-constant modulus signals through nonlinear satellite channels.

5.2 Suggestions for Further Research

The results presented in this dissertation were derived from computer simulation. In some cases, the performance of a particular receiver may justify a hardware implementation. In the case of the Volterra equalizers, issues such as precision and dynamic range would need to be studied before undertaking a hardware development. In the case of the MLSD receivers, a hardware implementation would require obtaining an accurate FSM model for an existing physical system. In such a case, further research involving modeling an actual nonlinear bandlimited satellite channel may be necessary.

Several areas of research are possible as an extension to the equalizers of Chapter 3. Straightforward extensions to the adaptive Volterra equalizers are possible. For example, nonlinear decision feedback and ISI cancellers have not been studied for a satellite channel. However, these latter structures have been considered for voiceband and magnetic recording channels [48 - 50]. The performance results of the Volterra equalizer and FSE-Volterra equalizer can serve as a baseline for comparison of the performance of decision feedback and ISI canceller structures. Also, the MFB-MLSD receivers provide a lower bound on probability of error performance.

The MLSD receivers of Chapter 4 showed very good performance when used with rectangular and raised-cosine filtering. However, for the nonlinear satellite channel with memory, in order to implement an MLSD receiver a FSM model of the channel (or an equivalent channel description) is necessary. For an actual channel, obtaining such a description may not be possible. In such a case, an adaptive MLSD receiver as in [51, 52] for linear channels may be useful. The structure and performance of these receivers for the nonlinear channel would need to be developed.

The methods discussed in this dissertation have been focused on the receiver. However, methods which combine equalization and coding as in [53] have recently been studied for magnetic recording channels. These methods may also give significant probability of error improvement for the nonlinear satellite channel. Also, adaptive inverse modeling where the adaptive inverse filter is placed forward of plant (i.e., pre-equalization) as in the filtered-x LMS algorithm [54, ch. 11] may be useful for the nonlinear bandlimited satellite channel.

REFERENCES

[1] D. H. Martin, *Communication Satellites 1958-1992,* The Aerospace Corporation, El Segundo, CA 90245, 1991.

[2] G. Zorpette, "Sensing Climate Change," *IEEE Spectrum,* pp. 20-27, July 1993.

[3] A. C. Clarke, "Extraterrestrial Relays," *Wireless World,* pp. 305-308, Oct. 1945.

[4] M. J. Miller, B. Vucetic, and L. Berry, *Satellite Communications Mobile and Fixed Services,* Kluwer Academic Publishers, Norwell, Mass. 1993.

[5] T. Pratt and C. W. Bostian, *Satellite Communications,* Wiley, New York, NY, 1986.

[6] R. J. Schwarz and B. Friedland, *Linear Systems,* McGraw-Hill, New York, NY, 1965.

[7] H. Stewart, "16-QAM Modems in Satellites," *Communication Systems Design,* pp. 36-40, July 1995.

[8] M. Thomas, M. Y. Weidner, and S. H. Durrani, "Digital Amplitude-Phase Keying with M-ary Alphabets," *IEEE Trans. Commun.*, vol. COM-2, no. 2, Feb. 1974.

[9] A. A. M. Saleh, "Frequency-Independent and Frequency-Dependent Nonlinear Models of TWT Amplifiers," *IEEE Trans. Commun.,* vol. COM-29, pp. 1715-1720, Nov. 1981.

[10] S. Pupolin and L. J. Greenstein, "Performance Analysis of Digital Radio Links with Nonlinear Transmit Amplifiers," *IEEE Journal on Selected Areas in Commun.*, vol. SAC-5, pp. 534-546, Apr. 1987.

[11] G. Karam and H. Sari, "Analysis of Predistortion, Equalization, and ISI Cancellation Techniques in Digital Radio Systems with Nonlinear Transmit Amplifiers," *IEEE Trans. Commun.*, vol. 37, no. 12, pp. 1245-1252, Dec. 1989.

[12] A. A. M. Saleh and J. Salz, "Adaptive Linearization of Power Amplifiers in Digital Radio Systems," *Bell Syst. Tech. Journal,* vol. 62, no. 4, pp. 1019-1033, April 1983.

[13] G. Karam and K. Sari, "A Data Predistortion Technique with Memory," *IEEE Trans. on Commun,* vol. 39, no. 2, pp. 336-344, Feb. 1991.

[14] S. K. Nair and J. Moon, "Nonlinear Equalization for Data Storage Channels," in *Proc. of Int. Comm. Conf.,* pp. 250-254, 1994.

[15] M. Ibnkahla, F. Castanie, "Vector Neural Networks for Digital Satellite Communications," in *Proc. of Int. Comm. Conf.,* Seattle, WA, pp. 1865-1869, June 1995.

[16] S. U. H. Quereshi, "Adaptive Equalization," *Proceedings of the IEEE,* vol. 73, no. 9, Sept. 1985.

[17] J.G. Proakis, *Digital Communications,* 2nd Ed., New York, NY, McGraw-Hill 1989.

[18] K. Wesolowski, "On the Performance and Convergence of the Adaptive Canceller of Intersymbol Interference in Data Transmission," *IEEE Trans. on Commun.,* vol. COM-33, no. 5, pp. 425-432, May 1985.

[19] S. Benedetto, E. Biglieri, and R. Daffara, "Modeling and Performance Evaluation of Nonlinear Satellite Links - A Volterra Series Approach," *IEEE Trans. on Aerospace and Electronic Systems,* vol. AES-15, no. 4, pp. 494-507, July 1979.

[20] S. Benedetto and E. Biglieri, "Nonlinear Equalization of Digital Satellite Channels," *IEEE Journal on Selected Areas in Commun.,* vol. SAC-1, no. 1, pp. 57-62, Jan. 1983.

[21] M. F. Mesiya, P. J. McLane, and L. L. Campbell, "Maximum Likelihood Sequence Estimation of Binary Sequences Transmitted Over Bandlimited Nonlinear Channels," *IEEE Trans. on Commun.,* vol. COM-25, no. 7, pp. 633-643, July 1977.

[22] S. Benedetto, E. Biglieri, and V. Castellani, *Digital Transmission Theory.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[23] G. D. Forney, Jr., "Maximum-Likelihood Sequence Estimation of Digital Sequences in the Presence of Intersymbol Interference," *IEEE Trans. on Information Theory,* vol. IT-18, no. 3, May 1972.

[24] K. Konstantinides and K. Yao, "Modelling and Computationally Efficient Time Domain Linear Equalisation of Bandlimited QPSK Satellite Channels," *IEE Proceedings,* vol. 137, pt. I. no. 6, pp. 438-442, Dec. 1990.

[25] R. D. Gitlin and S. B. Weinstein, "Fractionally-Spaced Equalization: An Improved Digital Transversal Equalizer," *Bell Syst. Tech. J.,* vol. 18, pp. 275-296, Feb. 1981.

[26] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Trans. on Information Theory,* vol. IT-13, pp. 260-269, Apr. 1967.

[27] T. Ericson, "Structure of Optimum Receiving Filters in Data Transmission

Systems," *IEEE Trans. on Information Theory*, vol. IT-17, pp. 352-353, May 1971.

[28] S. Haykin, *Adaptive Filter Theory,* 2nd Ed., Englewood Cliffs, NJ, Prentice-Hall, 1991.

[29] B. Widrow, "Adaptive Filters," in *Aspects of Network and System Theory,* ed. R.E. Kalman and N. DeClaris, Holt, Rinehart and Winston, 1970.

[30] B. Widrow and M. E. Hoff, Jr., "Adaptive Switching Circuits," *IRE WESCON Conv. Rec.*, pt. 4, pp. 96-104, Aug. 1960.

[31] B. Widrow, J. McCool, and M. Ball, "The complex LMS algorithm," *Proc. IEEE*, vol. 63, pp. 719-720, April 1975.

[32] R. W. Lucky, "Automatic Equalization for Digital Communications," *Bell Syst. Tech. J.,* vol. 44, pp. 547-588, April, 1965.

[33] R. D. Gitlin, J. F. Hayes, and S. B. Weinstein, *Data Communication Principles,* Plenum Press, New York, NY, 1992.

[34] J. E. Mazo, "On the Independence Theory of Equalizer Convergence," *Bell Syst. Tech. J.,* vol. 58, pp. 963-993, May-June 1978.

[35] M. C. Jeruchim, P. Balaban, and K. S. Shanmugan, *Simulation of Communication Systems,* New York, NY, Plenum Press, 1992..

[36] M. Schetzen, "Theory of pth-Order Inverses of Nonlinear Systems," *IEEE Trans. on Circuits and Systems,* vol CAS-23, no. 5, pp. 503-507, May 1975.

[37] W. G. Jeon, J. S. Son, Y. S. Cho, Y H. Lim, and D. H. Youn, "Nonlinear Equalization for Reduction of Nonlinear Distortion in High-Density Recording Channels.", in *Proc. of Int. Comm. Conf.,* Seattle, WA, pp. 1865-1869, June 1995.

[38] C. S. Modlin and J. M. Cioffi, "A Fast Decision Feedback LMS Algorithm Using Multiple Step Sizes," in *Proc. IEEE Int. Conf. on Comm.*, pp. 1201-1205, May 1993.

[39] R. W. Harris, D. M. Chabries, and F. A Bishop, "A Variable Step (VS) Adaptive Algorithm," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* pp. 309-316, April 1986.

[40] W. B. Mikhael, F. H. Wu, L. G. Kazavsky, G. S. Kang, and L. J. Fransen, "Adaptive Filters with Individual Adaption of Parameters," *IEEE Trans. on Circuits and Systems,* vol. CAS-33, pp. 677-686, July 1986.

[41] K. H. Mueller, "A New, Fast Converging MS algorithm for Adaptive Equalizers with Partial Response Signaling," *Bell Syst. Tech. J.,* Jan. 1975.

[42] E. Biglieri, M. Elia, and L. Lopresti, "The Optimal Linear Receiving Filter for Digital Transmission Over Nonlinear Channels," *IEEE Trans. on Infor-*

mation Theory, vol. 35, no. 3, May 1989

[43] H. Kobayashi, "Application of Probabilistic Decoding to Digital Magnetic Recording Systems," *IBM J. Res. Develop.*, vol. 15, pp. 64-74, Jan. 1971.

[44] J. K. Omura, "Optimal Receiver Design For Convolutional Codes and Channels with Memory via Control Theoretical Concepts," *Inform. Sci.*, vol. 3, pp. 243-266, July 1971.

[45] H. L. Van Trees, *Detection, Estimation, and Modulation Theory,* part I, Wiley, New York, NY, 1968.

[46] R. C. P. Saxena, *Optimum Encoding in Finite State Coded Modulation,* Report TR83-2, Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institue, Troy, N.Y., 1983.

[47] M. G. Mulligan and S. G. Wilson, "An improved algorithm for evaluating trellis phase codes," *IEEE Trans. on Information Theory*, vol. IT-30, pp. 846-851, Nov. 1984.

[48] E. Biglieri, A. Gersho, R. D. Gitlin, T. L. Lim, "Adaptive Cancellation of Nonlinear Intersymbol Interference for Voiceband Data Transmission," *IEEE Journal on Selected Areas in Commun.*, vol. SAC-2, no. 5, pp. 765-777, Sept. 1984.

[49] J.-H. Lin and C.-H. Wei, "Adaptive Nonlinear Decision Feedback Equalization with Channel Estimation and Timing Recovery in Digital Magnetic Recording Systems," *IEEE Trans. on Circuits and Systems,* vol. 42, no. 3, pp. 196-205, March 1985.

[50] E. Biglieri, E. Chiaberto, G. P. Maccone, and E. Viterbo, "Compensation of Nonlinearities in High-Density Magnetic Recording Channels, *IEEE Trans. on Magnetics,* vol. 30, no. 6, pp. 5079-5086, Nov. 1994.

[51] G. Ungerboeck, "Adaptive Maximum-Likelihood Receiver for Carrier-Modulated Data-Transmission Systems," *IEEE Trans. on Commun.*, vol. COM-22, no. 5, pp. 624-636, May 1974.

[52] F. R. Magee, Jr. and J. G. Proakis, "Adaptive Maximum-Likelihood Sequence Estimation for Digital Signaling in the Presence of Intersymbol Interference," *IEEE Trans. on Information Theory,* vol. IT-19, pp. 120-124, Jan. 1973.

[53] W. L. Abbott and J. M. Cioffi, "Combined Equalization and Coding for High-Density Saturation Recording Channels," *IEEE Journal on Selected Areas in Commun.*, vol. 10, no. 1, pp. 168-181. Jan. 1992.

[54] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Englewood Cliffs, NJ, Prentice-Hall, 1985.

A. Computer Program Listings

A.1. Volterra Equalizer Program

```
/*************************************************************/
/* pskvltre.c - A C-program which interfaces with MATLAB     */
/*              for simulating a passband Volterra equalizer */
/*              in baseband. MATLAB interfaces with this     */
/*              program via the GATEWAY routine. The         */
/*              GATEWAY routine then calls the computational */
/*              routine.                                     */
/*************************************************************/


#include <math.h>
#include <stdio.h>
#include "c:\matlab\extern\include\mex.h"
#define pi 3.141593

/**** COMPUTATIONAL ROUTINE ****/
void pskvltre(
      Matrix *x, Matrix *wi, int K,
      Matrix *mssize, int M, int nltaps,char *aflag,
      Matrix **y, Matrix **wo,Matrix **d,Matrix **s)
{

        /*********************/
        /* declare variables */
        /*********************/
double X,Xr,Xi,Xtemp;          /* misc variables */
int carry;                     /* variable for calc. weight idx */
int wrow,wcol;                 /* weight row and column var. */
int flg;                       /* flag all weights updated   */
int Mw;                        /* number of rows in weight matrix */
int i,j,k,l,dummy,Nx,Neq,Ny;   /* misc counters and max counts */
int nsymbol;                   /* samples per symbol*/
double *wPr,*wPi;              /* output weight pointers */
double *dPr,*dPi;              /* decision pointers */
double *xPr,*xPi;              /* input pointer */
double *yPr,*yPi;              /* output pointer */
double *wiPr,*wiPi;            /* input weight pointer */
double *sPr;                   /* symbol pointer */
```

```
double *mssizePr;              /* step sizes pointer */
double err_r,err_i,err;        /* error variables */
int *nidx;                     /* nl term index array */
double ssize;                  /* current step size */
int log2M;                     /* log2(M) */
int xidx;                      /* nl comb out matrix idx */
int nlwflag;                   /* no. of non lin weights */
int hinltap;                   /* hi index of non lin weights */
int lonltap;                   /* lo index of non lin weights */
double *Xr_mat;                /* real NL comb output matrix */
double *Xi_mat;                /* imag NL comb output matrix */
int order;                     /* current order variable */
double alphar[16],alphai[16]; /* symbol real and imag */
int beta1[16],beta2[16];       /* bit assignment variables */
int beta3[16],beta4[16];       /* bit assignment variables */

void pskdecn(int M,double yr,double yi,
          double *alphar, double *alphai,
          int *beta1, int *beta2, int *beta3, int *beta4,
          double *dr, double *di,
          double *s);

              /*************************************/
              /* Prepare for Decision Making       */
              /* define symbol and bit assignments */
              /*************************************/

       if (M==4) {
      alphar[0]=1/sqrt(2.0);alphai[0]=1/sqrt(2.0);  /* data=11 */
      alphar[1]=-1/sqrt(2.0);alphai[1]=1/sqrt(2.0); /*      01 */
      alphar[2]=-1/sqrt(2.0);alphai[2]=-1/sqrt(2.0);/*      00 */
      alphar[3]=1/sqrt(2.0);alphai[3]=-1/sqrt(2.0); /*      10 */
      beta1[0]=1;beta2[0]=1;
      beta1[1]=0;beta2[1]=1;
      beta1[2]=0;beta2[3]=0;
      beta1[3]=1;beta2[2]=0;
      }
       if (M==8) {
      alphar[0]=0.9239;alphai[0]=0.3827;
      alphar[1]=0.3827;alphai[1]=0.9239;
      alphar[2]=-0.3827;alphai[2]=0.9239;
      alphar[3]=-0.9239;alphai[3]=0.3827;
      alphar[4]=-0.9239;alphai[4]=-0.3827;
      alphar[5]=-0.3827;alphai[5]=-0.9239;
```

```
alphar[6]=0.3827;alphai[6]=-0.9239;
alphar[7]=0.9239;alphai[7]=-0.3827;
beta1[0]=0;beta2[0]=0;beta3[0]=0;
beta1[1]=0;beta2[1]=0;beta3[1]=1;
beta1[2]=0;beta2[2]=1;beta3[2]=1;
beta1[3]=0;beta2[3]=1;beta3[3]=0;
beta1[4]=1;beta2[4]=1;beta3[4]=0;
beta1[5]=1;beta2[5]=1;beta3[5]=1;
beta1[6]=1;beta2[6]=0;beta3[6]=1;
beta1[7]=1;beta2[7]=0;beta3[7]=0;
}
 if (M==16) {
alphar[0]=0.9808;alphai[0]=0.1951;
alphar[1]=0.8315;alphai[1]=0.5556;
alphar[2]=0.5556;alphai[2]=0.8315;
alphar[3]=0.1951;alphai[3]=0.9808;
alphar[4]=-0.1951;alphai[4]=0.9808;
alphar[5]=-0.5556;alphai[5]=0.8315;
alphar[6]=-0.8315;alphai[6]=0.5556;
alphar[7]=-0.9808;alphai[7]=0.1951;
alphar[8]=-0.9808;alphai[8]=-0.1951;
alphar[9]=-0.8315;alphai[9]=-0.5556;
alphar[10]=-0.5556;alphai[10]=-0.8315;
alphar[11]=-0.1951;alphai[11]=-0.9808;
alphar[12]=0.1951;alphai[12]=-0.9808;
alphar[13]=0.5556;alphai[13]=-0.8315;
alphar[14]=0.8315;alphai[14]=-0.5556;
alphar[15]=0.9808;alphai[15]=-0.1951;

beta1[0]=0;beta2[0]=0;beta3[0]=0;beta4[0]=0;
beta1[1]=1;beta2[1]=0;beta3[1]=0;beta4[1]=0;
beta1[2]=1;beta2[2]=1;beta3[2]=0;beta4[2]=0;
beta1[3]=0;beta2[3]=1;beta3[3]=0;beta4[3]=0;
beta1[4]=0;beta2[4]=1;beta3[4]=0;beta4[4]=1;
beta1[5]=1;beta2[5]=1;beta3[5]=0;beta4[5]=1;
beta1[6]=1;beta2[6]=0;beta3[6]=0;beta4[6]=1;
beta1[7]=0;beta2[7]=0;beta3[7]=0;beta4[7]=1;
beta1[8]=0;beta2[8]=0;beta3[8]=1;beta4[8]=1;
beta1[9]=1;beta2[9]=0;beta3[9]=1;beta4[9]=1;
beta1[10]=1;beta2[10]=1;beta3[10]=1;beta4[10]=1;
beta1[11]=0;beta2[11]=1;beta3[11]=1;beta4[11]=1;
beta1[12]=0;beta2[12]=1;beta3[12]=1;beta4[12]=0;
beta1[13]=1;beta2[13]=1;beta3[13]=1;beta4[13]=0;
beta1[14]=1;beta2[14]=0;beta3[14]=1;beta4[14]=0;
```

```
   beta1[15]=0;beta2[15]=0;beta3[15]=1;beta4[15]=0;

    }
           /********/
           /* MISC */
           /********/
mssizePr=mxGetPr(mssize);            /* step size vec pointer */
                                     /* allocate nl comb */
                                     /* real/imag matrics */
Xr_mat=(double *)mxCalloc(Mw*Neq*sizeof(double));
Xi_mat=(double *)mxCalloc(Mw*Neq*sizeof(double));
nidx=mxCalloc(2*K,sizeof(int));   /* allocate weight idx array*/
Neq=mxGetN(wi);                      /* weight matrix col's */
Mw=mxGetM(wi);                       /* weight matrix rows */
Nx=mxGetN(x);                        /* input length */
Ny=Nx+Neq-1;                         /* output length */
log2M=log((double)M)/log((double)2); /* log2(M) */
hinltap=(Neq-1)/2 + (nltaps-1)/2;    /* hi nonl tap */
lonltap=(Neq-1)/2 - (nltaps-1)/2;    /* lo nonl tap */


           /*****************************/
           /* Initialize Output Weights */
           /*****************************/

  *wo=mxCreateFull(Mw,Neq,COMPLEX);/* allocate output weights */
   wiPr=mxGetPr(wi);                /* real in weight pointer */
   wiPi=mxGetPi(wi);                /* imag in weight pointer */
   wPr=mxGetPr(*wo);                /* real out weight pointer */
   wPi=mxGetPi(*wo);                /* imag out weight pointer */
   for (i=0;i<Neq;i++)
   for (j=0;j<Mw;j++) {             /* init out weights  */
    wPr[i*Mw+j]=wiPr[i*Mw+j];       /*       = in weights
    wPi[i*Mw+j]=wiPi[i*Mw+j];
    }}

         /***************************************/
         /* Equalizer Output and Weight Update  */
         /***************************************/

    *y=mxCreateFull(1,Ny,COMPLEX);    /* create out vec      */
    *d=mxCreateFull(1,Ny,COMPLEX);    /* create decision vec */
    *s=mxCreateFull(log2M,Ny,REAL);   /* create symbol vec   */
    yPr=mxGetPr(*y);                  /* real out vec pointer*/
```

```
yPi=mxGetPi(*y);                    /* imag out vec pointer*/
dPr=mxGetPr(*d);                    /* real decision pointer*/
dPi=mxGetPi(*d);                    /* imag decision pointer*/
sPr=mxGetPr(*s);                    /* symbol pointer */
xPr=mxGetPr(x);                     /* real in vec pointer*/
xPi=mxGetPi(x);                     /* imag in vec pointer*/


for (i=Neq;i<=Nx;i++) {             /* EQ Fully Loaded */
yPr[i-1]=0; yPi[i-1]=0;             /* init i-th out to zero */
wrow=0;wcol=0;                      /* init weight matrix idx's*/
for(j=1;j<=2*K-1;j++)               /* init weight idx to zero */
   nidx[j]=0;                       /* init nl term idx array */
for(k=1;k<=K;k++){
  flg=0;
  xidx=0;
  while(!flg) {                   /* for all lin and nl weights*/
    Xr=1.0;Xi=0;nlwflag=1;
    if (k>1){
     for(l=1;l<=2*k-1;l++)
     if (nidx[2*k-1-l]<lonltap || nidx[2*k-1-l]>hinltap)
         nlwflag=0;}
     if (nlwflag==1) {          /* Is this NL term? */

     for(j=1;j<=2*k-1;j++)  {  /* Combine NL Terms */
     if(j>=k+1) {
       Xtemp=Xr*xPr[i-nidx[2*k-1-j]-1]
            +Xi*xPi[i-nidx[2*k-1-j]-1];
      Xi=-Xr*xPi[i-nidx[2*k-1-j]-1]
          +Xi*xPr[i-nidx[2*k-1-j]-1];
       Xr=Xtemp;}
      else {
       Xtemp=Xr*xPr[i-nidx[2*k-1-j]-1]
            -Xi*xPi[i-nidx[2*k-1-j]-1];
       Xi=Xr*xPi[i-nidx[2*k-1-j]-1]
         +Xi*xPr[i-nidx[2*k-1-j]-1];
       Xr=Xtemp;}}

     Xr_mat[xidx]=Xr;    /* Save NL combiner output */
     Xi_mat[xidx]=Xi;
     xidx++;
                 /* Accumulate Output */
     yPr[i-1]+=Xr*wPr[wrow+wcol*Mw]-Xi*wPi[wrow+wcol*Mw];
     yPi[i-1]+=Xi*wPr[wrow+wcol*Mw]+Xr*wPi[wrow+wcol*Mw]; }
```

```
                    /* update nl term idx */
                    /* update weight matrix idx's */
            if (wcol < Neq-1) wcol++;
            else {wcol=0; wrow++;}
            carry=1;
            for (l=0;l<2*k-1;l++) { /*update nidx[] modulo Neq*/
            nidx[l]+=carry;
              if(nidx[l]<Neq) carry=0;
              else { if (nidx[2*k-2]==Neq)
                      flg=1; nidx[l]=0; carry=1;}
           } /* end of for l */
          } /* end of while !flg */
         }  /* end of for k */
                              /* make decision */
       pskdecn(M,yPr[i-1],yPi[i-1],alphar,alphai,beta1,beta2,
            beta3,beta4,&dPr[i-1],&dPi[i-1],&sPr[log2M*(i-1)]);

                             /*******************/
                             /* WEIGHT ADAPTION */
                             /*******************/
       if (!strcmp(aflag,"on")) {
        err_r=dPr[i-1]-yPr[i-1];
        err_i=dPi[i-1]-yPi[i-1];
        wcol=0;wrow=0;xidx=0;
        for(k=1;k<=K;k++){
          order=2*k-1;
          ssize=mssizePr[k-1];     /* Step Size For This Order */
          for(j=0;j<(int)pow((double)Neq,(double)order);j++){
          Xr=Xr_mat[xidx];
          Xi=Xi_mat[xidx];
          wPr[wrow+wcol*Mw]+=ssize*(Xr*err_r+Xi*err_i);
          wPi[wrow+wcol*Mw]+=ssize*(Xr*err_i-Xi*err_r);
          wcol++;
          if(wcol==Neq) {wrow++; wcol=0;}
          xidx++;}}

      }/* end if !strcmp(aflag,"on") */
      }/* end for i */

} /** END COMPUTATION ROUTINE **/
```

```
            /*******************************/
            /**** GATEWAY ROUTINE **********/
            /*******************************/
void mexFunction(
   int nlhs, Matrix *plhs[],
   int nrhs, Matrix *prhs[])
{
   int Neq,flg;
   int M;                      /* modulation order */
   int i,j,xcomplx,wcomplx;    /* input signal and weight flags */
   double etrgt;               /* error target */
   int mcycle;                 /* maximum number of training cycles*/
   char adpflagi[4];           /* adaption flag   */
   int dummy;                  /* dummy variable */
   int nx,nw;                  /* columns for input and weights */
   int m;                      /* number of rows for weights */
   int k;                      /* order of non-linearity is 2k-1*/
   int nltaps;
   Matrix *x,*win,*nsymbol,*aflg,*mssize;
   double *apPr;
   char aflag[10];
   Matrix *ssizes;

                        /*** PROCESS I/O ARGUMENTS ***/
   if (nrhs !=6) mexErrMsgTxt("6 input arguments are required\n");
   else
    if (nlhs>4) mexErrMsgTxt("maximum of 4 output arguements\n");
   if ((mxIsNumeric(prhs[0])==0) || (mxGetM(prhs[0])!=1))
    mexErrMsgTxt("input signal must be an 1 x n numeric vector\n");
   if ( mxIsNumeric(prhs[1])==0 )
    mexErrMsgTxt("weights must be an m x n numeric vector\n");
   m=mxGetM(prhs[1]);
   Neq=mxGetN(prhs[1]);
   i=3;j=1; k=0;   flg=0;
   while(!flg) {
     k++;
     if (m==j) break;
     j+=(int)pow((double)Neq,(double)(i-1));
     i+=2;
     if (m<j)
       { mexErrMsgTxt("cannot derive,k from input weights\n");}
     }
```

```
    if ((mxIsNumeric(prhs[2])==0) || (mxGetM(prhs[2]) != 1)
        || (mxGetN(prhs[2]) !=1)){
      mexErrMsgTxt("modulation order, M, must be a real scaler\n");}
    if ( (*mxGetPr(prhs[2])!=2) && (*mxGetPr(prhs[2])!=4)
        && (*mxGetPr(prhs[2])!=8)&& (*mxGetPr(prhs[2])!=16) )
      mexErrMsgTxt("modulation order, M, must be 2, 4, 8, or 16\n");
    if ( mxGetN(prhs[3])!=k)
      mexErrMsgTxt("number of step sizes must agree with order\n");
    if (mxIsNumeric(prhs[4])==0 || mxGetN(prhs[4])!=1
        || mxGetM(prhs[4])!=1)
      mexErrMsgTxt("No. of nltaps must be a numeric scaler\n");
    nltaps=*mxGetPr(prhs[4]);
    if (nltaps%2!=1)
      mexErrMsgTxt("No. of Nonlinear taps must be odd\n");
    if (mxIsString(prhs[5])==0)
      mexErrMsgTxt("adaption flag must be ascii\n");
          /*  CREATE OUTPUT VARIABLES and CALL COMP ROUTINE*/
     x=prhs[0];
     win=prhs[1];
     M=*mxGetPr(prhs[2]);
     ssizes=prhs[3];
     mxGetString(prhs[5],aflag,10);
     pskvltre(x,win,k,ssizes,M,nltaps,aflag,&plhs[0],
                      &plhs[1],&plhs[2],&plhs[3]);
}

 void pskdecn(int M,double yr,double yi,
          double *alphar, double *alphai,
          int *beta1, int *beta2, int *beta3,int *beta4,
          double *dr, double *di,
          double *s) {
 double yangle;
 int k;
 if (M==2) {
      if (yr > 0 ) {*dr=1;*s=1;}
      else {*dr=-1;*s=0;}
      }
 else {
     if (yr==0 && yi==0) yangle=0;
       else yangle=atan2(yi,yr);
       if(yangle<0) yangle+=2*pi;
       k=floor(yangle/(2*pi/M));
       if (M==4) {
```

```
        *dr=alphar[k];
        *di=alphai[k];
        s[0]=beta1[k];
        s[1]=beta2[k];
        }
      if (M==8) {
        *dr=alphar[k];
        *di=alphai[k];
        s[0]=beta1[k];
        s[1]=beta2[k];
        s[2]=beta3[k];
        }
      if (M==16) {
        *dr=alphar[k];
        *di=alphai[k];
        s[0]=beta1[k];
        s[1]=beta2[k];
        s[2]=beta3[k];
        s[3]=beta4[k];
        }
   }
} /* end pskdecn */
```

A.2. State Table Generation Program

```
%      srcfb:     a MATLAB program for Generating
%                 State Table
%                 Matched Filter Bank Rx
%                 Nonlinear Satellite Channel
%    CHANNEL PARAMETERS
%      - Tx = square root raised cosine
%      - Rx = Matched Filter Bank
%      - TWT 0 dB backoff
%    INSTRUCTIONS
%      - Change the System Parameters
%      - Indicate the file in which to store the state machine
%      - Execute from within MATLAB
%    COMM. SYS. COMPONENTS
%      Transmitter - square root raised cosine
%      Channel     - TWT
%      Receiver    - matched filter bank
%    STATE MACHINE FILE FORMAT
%      The file will have M^(L+1) entries (i.e., lines). The first
%      field of a line contains the input. The input is an integer
%      in the range 0, 1, 2, ... M-1. The next set of fields
%      contain the L previous inputs which correspond to the state.
%      The next field contains the channel output (note the output
%      may be complex). The number of outputs per line is 2*nsym,
%      where the nsym parameter, set below, indicates the number of
%      samples per symbol. The factor of 2 is because the outputs
%      are complex.
%      The following is a listing of the first few lines for
%      a system with modulation order M=4 memory L=2 and 1
%      sample per symbol.
%
%        0 0 0 -0.705858 -0.705858i
%        1 0 0 -0.757653 -0.705858i
%        2 0 0 -0.705858 -0.757653i
%        3 0 0 -0.757653 -0.757653i
%        0 1 0  0.51781  -0.705858i
%        1 1 0  0.466016 -0.705858i
%        2 1 0  0.51781  -0.757653i
```

```
%        3 1 0  0.466016 -0.757653i
%        0 2 0 -0.705858  0.51781i
%        1 2 0 -0.757653  0.51781i
%
%    OTHER MATLAB PROGRAMS CALLED BY THIS PROGRAM
%       These are straightforward and are only summarized here
%
%       randbits - generate random bits
%       pskmd    - bit to symbol mapping (i.e., baseband modulate)
%       rcff     - raised cosine filter, square root raised cosine
%                  with appropriate flag settings. Used here as
%                  square root raised cosine filter.
%       xltwt    - low-pass equivalent TWT model, implementation
%                  of Saleh's analytical model (IEEE Trans. Comm.
%                  vol. COM-29. No. 11, pp. 1715-1720, Nov. 1981)
%       symsync  - find sample number which gives largest
%                  correlation between two complex vectors. This
%                  is used to synchronize the received signal with
%                  the input signal. This way, a comparison of bit
%                  errors may be made. This also generates the
%                  phase offset between the two signals.
%       eavg     - calculate the average energy in a signal given
%                  the number of samples per symbol.
%       dec2bin  - convert a decimal number to a binary number

      %%%%%%%%%%%%%%%%%%%%%%%%%
      %%% SYSTEM Parameters %%%
      %%%%%%%%%%%%%%%%%%%%%%%%%
Rsym=1;                  % Sample Rate
nsym=4;                  % Samples Per Symbol
fs=nsym*Rsym;                  % Sample Rate
msm_flg='on';                  % make state machine flag
wsm_flg='on';                  % write state machine flag
Lf=1; % Comm System Memory future
Lp=1; % Comm System Memory past
L=Lf+Lp; % Total Memory
M=8; % Modulation Order
K=log(M)/log(2); % log2(M)
beta=1; % Rolloff parameter
twtp=[1.9638 0.9945 2.5293 2.8168]; % TWT Parameters
backoff=0; % TWT backoff
Nout=M^(L+1)-1; % no. of outputs <=M^(L+1)-1
%  M^(L+1)-1 for entire state
%  machine
```

```
%%%%%%%%%%%%%%
% FILENAME!!!%
%%%%%%%%%%%%%%
if msm_flg=='on'
  if wsm_flg=='on'
    fid=fopen('frc1_811','w');  % indicate state table file here
  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% tx and rx long sequence      %%%
%%% good for eye and scatter     %%%
%%% also determine sample offset %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear bl xldlta xltx xlprf % clr variables avoid confusion
clear xltwt xlrx y xlpof
bl=randbits(K,512);        % random bit sequence
bl=[zeros(K,50) bl zeros(K,50)]; % new bit sequence pad with zeros
xldlta=pskmd(bl,nsym);             % tx impulse sequence
xltx=rcff(xldlta,nsym,nsym*17,beta,1,0,0);%raised cosine tx filter
xltwt=ltwt(xltx,twtp,backoff,0);   % TWT
e_avg=eavg(xltwt(10:length(xltwt)-10),nsym); % avg sym. energy
xltwt=xltwt/sqrt(e_avg);           % normalize
                                   % correlate with tx seq
[xx smpofst phsofst]=symsync(xldlta,xltwt,nsym,100);
xltwt=xltwt*exp(-sqrt(-1)*phsofst); % phase sync
ofst=smpofst+(50+Lp)*nsym;          % sample offset


       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       %%%%   Create State Machine Model     %%%
       %%%%     write oversampled ch. output %%%
       %%%%     to file                      %%%
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear in_state b bits xdlta xtx      % clr variables
clear xprf xtwt xpof xrx rxo rxs
if msm_flg=='on'
eav=0;
erx=0;
   in_state=zeros(1,1+L);            % initialize variables
   b=zeros(K,1+L);
   for(i=0:Nout)
```

```
      for(k=1:L+1)
       b(:,k)=dec2bin(in_state(k),K)'; % input and state sequence
      end
      bits=[randbits(K,50) b randbits(K,50)]; % tx sequence
      xdlta=pskmd(bits,nsym);              % modulate
      xtx=rcff(xdlta,nsym,nsym*17,beta,1,0,0);%sqr.root rais. cos.
      xtwt=ltwt(xtx,twtp,backoff,0);       % TWT
      xtwt=xtwt/sqrt(e_avg);               % normalize energy
      xtwt=xtwt*exp(-sqrt(-1)*phsofst); % phase sync
      h=xtwt(ofst:ofst+nsym-1);            % channel output vec
                                           % WRITE TO FILE %
      for(j=1:L+1)
        fprintf(fid,'%g ',in_state(j)); % write input_state
      end
      if (M>2)                             % complex
       for j=1:nsym
                                           % write ch. output
         fprintf(fid,' %g %gi',real(h(j)),imag(h(j)));
        end
      else                                 % real
        for j=1:nsym
          fprintf(fid,'%g',real(h(j))); % write ch. out
       end
      end
      fprintf(fid,'\n');                    % done writing this
                                           % line
      carry=1;                             % determine next
      for(k=1:L+1)                         % input state sequence
        if (carry==1) in_state(k)=in_state(k)+carry;,end
        if (in_state(k)==M) in_state(k)=0; carry=1;
        else carry=0;, end
      end

      fprintf('%d ',i);                        % some output to let
      if (rem(i+1,10) == 0) fprintf('\n');,end% user know the prog.
                                           % is doing something
    end
    if wsm_flg=='on'
      fclose(fid);
    end
end
```

A.3. Matched Filter Bank Receiver Program

```
/*************************************************************/
/*   cv2.c:       A program for simulating the performance   */
/*                of a matched filter bank receiver followed */
/*                by a Viterbi Detector. The program first   */
/*                reads in the state table description and   */
/*                uses this to generate channel outputs and  */
/*                to derive Viterbi detector path metrics.   */
/*                The program assumes the state table        */
/*                contains the oversampled outputs from      */
/*                the channel.                               */
/*************************************************************/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>


                /***************************/
                /*** TYPE DECLERATIONS ****/
                /***************************/
typedef struct{
  double re;
  double im;
}cmplx_dbl;
typedef struct{
  float re;
  float im;
}cmplx_flt;
typedef struct{
  int        *input;
  int        *state;
  cmplx_flt  *output;
} stable;
typedef struct{
  int  *input;
  int  *state;
}trnsmt;
typedef struct{
```

```
  int path_lgth;
  int *prv_stbl_idx;
  double *met_new;
  double *met_old;
  int *pmem;
  int *pmem_old;
  double *fb;
  double *fbsig;
}trellis;

                /**************/
                /***  MAIN  ***/
                /**************/
int main() {
                /* declare variables */

 FILE *st_tbl_file;     /* state table file for receiver */
 FILE *st_tbl_file_tx;  /* state table file for transmitter */
 int num_states;        /* number of states in receiver */
 int num_states_tx;     /* number of states for transmitter */
 int M;                 /* modulation order */
 int L,L_tx;            /* memory length */
 int nsym;              /* samples per symbol */
 float n;               /* symbol number */
 float MaxSymbols;      /* maximum number of tx'd symbols */
 float MaxErrs;         /* maximum number of errs */
 float EsNodb;          /* symbol to noise spec dens ratio */
 float EsAvg;           /* Average Signal Energy */
 stable stbl;           /* state table for receiver */
 stable stbl_tx;        /* state table for transmitter */
 trnsmt tx;             /* transmitter structure */
 trellis trls;          /* trellis structure */
 int PathLgth;          /* path memory length*/
 cmplx_dbl noise;       /* noise variable */
 cmplx_dbl *ch_output;  /* channel output */
 int tx_out_idx;        /* channel output index */
 int i,j,k;             /* misc ints */
 int sym_err_cntr;      /* error counter   */
 int out;               /* detector output */
 int ref;               /* noiseless tx output */
 int *ref_reg;          /* ref delay register-for decoding delay*/
 int cntr1=0;           /* Number of symbols counter */
```

```
                /* declare functions */

void load_st_tbl_cmplx(FILE *st_tbl_file,
                       stable stbl,int M,int L,int nsym);
void print_st_tbl_cmplx(stable stbl,int M,int L,int nsym);
void updt_tx(trnsmt tx,int M,int L);
void updt_fb(stable stbl, cmplx_dbl *ch_output,
             trellis trls, int M, int L, int nsym);
void updt_trls(stable stbl, trellis trls, int M, int L);
int get_stbl_idx(trnsmt tx, int M, int L);
int get_trls_out(trellis trls,int M,int L);
void add_awgn(double esavg, double esnodb,
             cmplx_dbl *ch_output,int nsym);
char st_tbl_filename[100];
char st_tbl_filename_tx[100];
void get_ch_out(stable stbl_tx,int tx_out_idx,
               cmplx_dbl *ch_output, int nsym);
void init_fbsig(stable stbl, trellis trls, int M,
                int L, int nsym);

                /* get user input */

printf("Enter MaxSymbols, MaxErrs, Es/No(dB), EsAvg: ");
scanf("%f %f %f %f",&MaxSymbols,&MaxErrs,&EsNodb,&EsAvg);
printf("Enter Symbol Set Size, Samples/Symbol, Tx Mem,Rx Mem: ");
scanf("%d %d %d %d",&M,&nsym,&L_tx,&L);
printf("Enter Viterbi Detector Path Memory Length: ");
scanf("%d",&PathLgth);
printf("Enter Tx State Table Filename: ");
scanf("%s",st_tbl_filename_tx);
printf("Enter Rx State Table Filename: ");
scanf("%s",st_tbl_filename);


if ((st_tbl_file=fopen(st_tbl_filename,"r"))==NULL) {
    printf("ERROR: could not open \"%s\" for reading\n",
          st_tbl_filename);
    exit(1);
    }
if ((st_tbl_file_tx=fopen(st_tbl_filename_tx,"r"))==NULL) {
    printf("ERROR: could not open \"%s\" for reading\n",
           st_tbl_filename_tx);
    exit(1);
    }
```

```
                /* memory allocation */
num_states=pow((double)M,(double)L);
num_states_tx=pow((double)M,(double)L_tx);
stbl.input=(int *)malloc(M*num_states*sizeof(int));
stbl.state=(int *)malloc(M*L*num_states*sizeof(int));
stbl.output=(cmplx_flt *)malloc(nsym*M*num_states*
                                sizeof(cmplx_flt));
stbl_tx.input=(int *)malloc(M*num_states_tx*sizeof(int));
stbl_tx.state=(int *)malloc(M*L_tx*num_states_tx*sizeof(int));
stbl_tx.output=(cmplx_flt *)malloc(nsym*M*num_states_tx*
                                   sizeof(cmplx_flt));
tx.state=(int *)malloc(L_tx*sizeof(int));
tx.input=(int *)malloc(sizeof(int));
trls.met_new=(double *)malloc(num_states*sizeof(double));
trls.met_old=(double *)malloc(num_states*sizeof(double));
trls.pmem=(int *)malloc(num_states*PathLgth*sizeof(int));
trls.pmem_old=(int *)malloc(num_states*PathLgth*sizeof(int));
trls.prv_stbl_idx=(int *)malloc(num_states*sizeof(int));
trls.fb=(double *)malloc(M*num_states*sizeof(double));
trls.fbsig=(double *)malloc(M*num_states*sizeof(double));
ch_output=(cmplx_dbl *)malloc(nsym*sizeof(cmplx_dbl));
ref_reg=(int *)malloc(PathLgth*sizeof(int));


                /* load rx and tx state table */

load_st_tbl_cmplx(st_tbl_file,stbl,M,L,nsym);
load_st_tbl_cmplx(st_tbl_file_tx,stbl_tx,M,L_tx,nsym);


                /* initialize variables */

for(i=0;i<L_tx;i++) tx.state[i]=0;
*tx.input=0;
for(i=0;i< num_states*PathLgth;i++)
    trls.pmem[i]=0;
for(i=0;i< num_states*PathLgth;i++)
    trls.pmem_old[i]=0;
for(i=0;i< PathLgth;i++) ref_reg[i]=0;
*trls.met_new=0;
*trls.met_old=0;
trls.path_lgth=PathLgth;
```

```
init_fbsig(stbl,trls,M,L,nsym);

                /* simulate the tx, channel, MLSE rx */

sym_err_cntr=0;
n=0;
while(n<MaxSymbols && sym_err_cntr<MaxErrs) {

                                        /* use state table to */
   updt_tx(tx,M,L);                     /* generate state table */
   tx_out_idx=get_stbl_idx(tx,M,L);     /* address for the next */
                                        /* output */



   get_ch_out(stbl_tx,tx_out_idx,       /* get output from */
              ch_output,nsym);          /* state table and */
   add_awgn(EsAvg,EsNodb,ch_output,nsym);/* add noise */



   updt_fb(stbl,ch_output,trls,M,L,nsym);/* update filter bank */

                                         /* update trellis     */
   updt_trls(stbl,trls,M,L);             /* including path met. */

                                         /* get output with    */
   out=get_trls_out(trls,M,L);           /* best metrics       */

  for(i=trls.path_lgth-1;i>0;i--)        /* delay inputs       */
     ref_reg[i]=ref_reg[i-1];            /* so can compare     */
  ref_reg[0]=*tx.input;                  /* with outputs       */
  ref=ref_reg[trls.path_lgth-1];

  if (out!=ref) {                        /* print and update */
    sym_err_cntr++;                      /*  stats on errors */
    printf("Symbol Error, n=%g, Detect=%d,
               Ref=%d SymErrs=%d SER=%f\n",
               n+1,out,ref,sym_err_cntr,
               sym_err_cntr/(n+1));
    }

  if (++cntr1==10000){                   /* print stats every */
     printf("n=%g, SymErrs=%d, SER=%f\n",n+1,/*  10K symbols */
```

```
            sym_err_cntr,sym_err_cntr/(n+1));
      cntr1=0;
      }
  n++;
  }
                                          /* print final stats */
 printf("n=%g, SymErrs=%d, SER=%f\n",
        n,sym_err_cntr,sym_err_cntr/n);

 return(0);
}
 /*** END of MAIN ***/

void load_st_tbl_cmplx(FILE *st_tbl_file,stable stbl,
                       int M,int L,int nsym) {
 int num_states;
 int i,j;
 num_states=pow((double)M,(double)L);
 for(i=0;i< M*num_states;i++)  {
    fscanf(st_tbl_file,"%d",&stbl.input[i]);
    for(j=0;j< L;j++)  fscanf(st_tbl_file,"%d",&stbl.state[j+i*L]);
    for(j=0;j<nsym;j++)
      fscanf(st_tbl_file,"%f %fi",
             &stbl.output[i*nsym+j].re,&stbl.output[i*nsym+j].im);
    }
}

void print_st_tbl_cmplx(stable stbl,int M,int L, int nsym) {
 int num_states;
 int i,j;
 num_states=pow((double)M,(double)L);
 for(i=0;i<M*num_states;i++) {
   printf("%d ",stbl.input[i]);
   for(j=0;j<L;j++)  printf("%d ",stbl.state[j+i*L]);
   for(j=0;j<nsym;j++)
    printf(" %g %gi",
           stbl.output[i*nsym+j].re,stbl.output[i*nsym+j].im);
   printf("\n");
   }
}

void updt_tx(trnsmt tx,int M,int L) {
  int i;
  for(i=L-1;i>0;i--) tx.state[i]=tx.state[i-1];
```

```
  tx.state[0]=*tx.input;
  *tx.input=rand()%M;
}


int get_stbl_idx(trnsmt tx, int M, int L) {
  int i;
  int stidx;
  stidx=*tx.input;
  for (i=0;i<L;i++) stidx+=
                      tx.state[i]*pow((double)M,(double)(i+1));
  return(stidx);
}
int get_snum(stable stbl,int stbl_idx,int M, int L) {
  int i;
  int snum=0;
  for (i=0;i<L;i++)
    snum+=stbl.state[i+stbl_idx*L]*pow((double)M,(double)(i));
  return(snum);
}
int get_trls_out(trellis trls,int M,int L){
   int num_states;
   int i,best_state;
   double tmp;
   num_states=pow((double)M,(double)L);
   best_state=0;
   for(i=0;i<num_states;i++)
     if(trls.met_new[i]<trls.met_new[best_state]) best_state=i;
   if (trls.met_new[best_state]>1e9) {
     tmp=trls.met_new[best_state];
     for(i=0;i<num_states;i++){
       trls.met_new[i]-=tmp;
       trls.met_old[i]-=tmp;
       }
    }
   return(trls.pmem[(best_state+1)*trls.path_lgth-1]);
}


void updt_trls(stable stbl,trellis trls, int M, int L) {
  int num_states;
  int state;
  int acs(int state,stable stbl, trellis trls, int M, int L);
  void reg_xchg(stable stbl,trellis trls,int M, int L);
```

```
  num_states=pow((double)M,(double)L);
  for (state=0;state<num_states;state++)
     trls.prv_stbl_idx[state]=acs(state,stbl,trls,M,L);
  for (state=0;state<num_states;state++)
     trls.met_old[state]=trls.met_new[state];
  reg_xchg(stbl,trls,M,L);
}
int acs(int state,stable stbl, trellis trls, int M, int L) {
  int num_states;
  int i;
  int pvst_idx,best_pvst_idx;
  int prv_snum;
  double br_metric;
  double tmp_metric;
  int get_snum(stable stbl,int stbl_idx,int M, int L);
    for(i=0;i<M;i++) {
    pvst_idx=state+i*pow((double)M,(double)(L));
    br_metric=-2*trls.fb[pvst_idx]+trls.fbsig[pvst_idx];
    prv_snum=get_snum(stbl,pvst_idx,M,L);
    tmp_metric=br_metric+trls.met_old[prv_snum];
    if (i==0 ) {
       trls.met_new[state]=tmp_metric;
       best_pvst_idx=pvst_idx;
       }
    else if(tmp_metric<trls.met_new[state]) {
          trls.met_new[state]=tmp_metric;
          best_pvst_idx=pvst_idx;
          }
    }
    return(best_pvst_idx);
}

void reg_xchg(stable stbl,trellis trls,int M, int L){
  int i,state,num_states;
  int prv_state;
  int get_snum(stable stbl,int stbl_idx,int M, int L);

  num_states=pow((double)M,(double)L);
  for(state=0;state<num_states;state++) { /*** REGISTER XCHG ****/
    prv_state=get_snum(stbl,trls.prv_stbl_idx[state],M,L);
    for(i=0;i<trls.path_lgth;i++)
       trls.pmem[i+state*trls.path_lgth]=
                          trls.pmem_old[i+prv_state*trls.path_lgth];
     }
```

```
  for(state=0;state<num_states;state++) {  /*** REGISTER SHFT ****/
    prv_state=get_snum(stbl,trls.prv_stbl_idx[state],M,L);
    for(i=trls.path_lgth-1;i>0;i--)
       trls.pmem[i+state*trls.path_lgth]=
                                trls.pmem[i-1+state*trls.path_lgth];
    trls.pmem[state*trls.path_lgth]=
                               stbl.input[trls.prv_stbl_idx[state]];

  }
  for(state=0;state<num_states;state++) {  /*** copy to old ****/
    for(i=0;i<trls.path_lgth;i++)
       trls.pmem_old[i+state*trls.path_lgth]=
                                  trls.pmem[i+state*trls.path_lgth];
  }}

void init_fbsig(stable stbl, trellis trls, int M, int L, int nsym){
  int i,j;
  int num_states;
  num_states=pow((double)M,(double)L);
  for(i=0;i<M*num_states;i++) {
      trls.fbsig[i]=0;
      for(j=0;j<nsym;j++) {
         trls.fbsig[i]+=
                stbl.output[i*nsym+j].re*stbl.output[i*nsym+j].re
               +stbl.output[i*nsym+j].im*stbl.output[i*nsym+j].im;
      }
    }}

void updt_fb(stable stbl, cmplx_dbl *ch_output,
           trellis trls, int M, int L, int nsym) {
  int i,j;
  int num_states;
  num_states=pow((double)M,(double)L);
  for(i=0;i<M*num_states;i++) {
      trls.fb[i]=0;
      for(j=0;j<nsym;j++) {
         trls.fb[i]+=ch_output[j].re*stbl.output[i*nsym+j].re
                     +ch_output[j].im*stbl.output[i*nsym+j].im;
      }
    }}

void get_ch_out(stable stbl_tx,int tx_out_idx,
                cmplx_dbl *ch_output,int nsym) {
 int i;
```

```
 for(i=0;i<nsym;i++) {
  ch_output[i].re=stbl_tx.output[tx_out_idx*nsym+i].re;
  ch_output[i].im=stbl_tx.output[tx_out_idx*nsym+i].im;
 }}

void add_awgn(double esavg, double esnodb,
               cmplx_dbl *ch_output, int nsym ) {
  double u1,u2,n1,n2,No,esno;
  int j;
  esno=pow(10.0,esnodb/10.0);
  No=esavg/esno;
  for(j=0;j<nsym;j++) {
   u1=((double)rand())/(RAND_MAX);
   u2=((double)rand())/(RAND_MAX);
   if (u1 < 1e-8) u1=1e-8;
   ch_output[j].re+=sqrt(No/2)*sqrt(-2*log(u1))*cos(2*3.141593*u2);
   ch_output[j].im+=sqrt(No/2)*sqrt(-2*log(u1))*sin(2*3.141593*u2);
   }}
```

A.4. Program for Calculating d_{min}

```
/*********************************************************/
/*   dmin2.c:    A program for calculating the minimum  */
/*               Euclidean distance between channel     */
/*               sequences based on a state table       */
/*               description of the channel. The state  */
/*               table is assumed to contain the        */
/*               oversampled outputs of the channel.    */
/*               The program is based on the algorithm  */
/*               described in "Digital Transmission     */
/*               Theory," by Benedetto, Biglieri, and   */
/*               Castellani, Prentice Hall, 1987.       */
/*               chapter 10.                            */
/*********************************************************/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
typedef struct{
  double re;
  double im;
}cmplx_dbl;
typedef struct{
  float re;
  float im;
}cmplx_flt;
typedef struct{
  int        *input;
  int        *state;
  cmplx_flt  *output;
} stable;


int main () {
 int M;                          /*symbol set size*/
 int L;                          /*channel memory*/
 int flag;                       /*misc flag */
 int n;                          /*path length variable*/
 int i,j;                        /*misc variables*/
```

```
FILE *st_tbl_file;              /*state table filename */
int num_states;                 /*number of channel states */
int nsym;                       /*samples per symbol */
stable stbl;                    /*state table */
double *Dn,*Dn_old;             /*matrix of min sqrd distances*/
int *PM;                        /*predecessor matrix*/
double dmin;                    /*minimum distance squared */
char st_tbl_filename[100];

void load_st_tbl_cmplx(FILE *st_tbl_file,
                       stable stbl,int M,int L,int nsym);

void print_st_tbl_cmplx(stable stbl,int M,int L,int nsym);

void build_PM(stable stbl, int *PM, int M, int L);
void init_Dn(double *Dn,int *PM,stable stbl,int M,int L,int nsym);
void updt_Dn(double *Dn,double *Dn_old,int *PM,
                       stable stbl,int M,int L,int nsym);
void stat_Dn(double *Dn,int M,int L,int *flag,double *dmin);

                /****************/
                /** User Input **/
                /****************/
printf("Enter Symbol Set Size M: ");
scanf("%d",&M);
printf("Enter Samples per Symbol: ");
scanf("%d",&nsym);
printf("Enter Channel Memory L: ");
scanf("%d",&L);
printf("Enter State Table Filename: ");
scanf("%s",st_tbl_filename);

if ((st_tbl_file=fopen(st_tbl_filename,"r"))==NULL) {
  printf("ERROR: could not open \"%s\" for reading\n",
         st_tbl_filename);
  exit(1);
  }
                /* memory allocation */
num_states=pow((double)M,(double)L);
stbl.input=(int *)malloc(M*num_states*sizeof(int));
stbl.state=(int *)malloc(M*L*num_states*sizeof(int));
stbl.output=(cmplx_flt *)malloc(nsym*M*num_states*
            sizeof(cmplx_flt));
PM=(int *)malloc(num_states*M*sizeof(int));
```

```
  Dn=(double *)malloc(num_states*num_states*sizeof(double));
  Dn_old=(double *)malloc(num_states*num_states*sizeof(double));

                  /* load state table */
  load_st_tbl_cmplx(st_tbl_file,stbl,M,L,nsym);


  build_PM(stbl,PM,M,L);              /* build Predecessor Matrix */
                                      /* step 1a */

  n=0;                                /* init seq. length variable */
                                      /* Initialize Squared Minimum */
  init_Dn(Dn,PM,stbl,M,L,nsym);       /* Distance Matrix "Dn" */
                                      /* step 1b and step 2   */

  flag=1;
  while(flag>0) {
                                      /* Update Distance Matrix */
    updt_Dn(Dn,Dn_old,PM,stbl,M,L,nsym);  /* step 3 */

                                      /* Check if Done */
    stat_Dn(Dn,M,L,&flag,&dmin);      /* step 4 */

    n++;                              /* update seq. length */

    printf("n=%d d=%g\n",n,dmin);     /* print stats */
     }

    printf("\nn = %d, dmin^2 = %g\n",
                              n,dmin);/* print final stats */
    return(0);
}

void load_st_tbl_cmplx(FILE *st_tbl_file,stable stbl,
                       int M,int L,int nsym) {
 int num_states;
 int i,j;
 num_states=pow((double)M,(double)L);
 for(i=0;i< M*num_states;i++)  {
    fscanf(st_tbl_file,"%d",&stbl.input[i]);
    for(j=0;j< L;j++)  fscanf(st_tbl_file,"%d",&stbl.state[j+i*L]);
    for(j=0;j<nsym;j++)
      fscanf(st_tbl_file,"%f %fi",
            &stbl.output[i*nsym+j].re,&stbl.output[i*nsym+j].im);
```

```
    }
}


void print_st_tbl_cmplx(stable stbl,int M,int L, int nsym) {
 int num_states;
 int i,j;
 num_states=pow((double)M,(double)L);
 for(i=0;i<M*num_states;i++) {
   printf("%d ",stbl.input[i]);
   for(j=0;j<L;j++)  printf("%d ",stbl.state[j+i*L]);
   for(j=0;j<nsym;j++)
    printf(" %g %gi",stbl.output[i*nsym+j].re,
                     stbl.output[i*nsym+j].im);
   printf("\n");
   }
}

void build_PM(stable stbl, int *PM, int M, int L) {
 int pstate_idx,pstate,state,j,num_states;
 int get_snum(stable stbl,int stbl_idx,int M, int L);
 num_states=pow((double)M,(double)L);

 for(state=0;state<num_states;state++) {
   for (j=0;j<M;j++) {
     pstate_idx=state+j*pow((double)M,(double)L);
                                  /*j-th predecessor of state i*/
     PM[j+M*state]=get_snum(stbl,pstate_idx, M, L);
   }}
}

  /* get state number from state table index */
int get_snum(stable stbl,int stbl_idx,int M, int L) {
  int i;
  int snum=0;
  for (i=0;i<L;i++)
    snum+=stbl.state[i+stbl_idx*L]*pow((double)M,(double)(i));
  return(snum);
}

void init_Dn(double *Dn,int *PM,stable stbl,int M,int L,int nsym){
   int num_states,i,j,k;
   int pj,pi;                    /* previous i and j */
   int Spi,Spj;                   /* states pi and pj */
```

```
    int Spi_stbl_idx;
    int Spj_stbl_idx;
    double d2,dif_re,dif_im;
    double dij_min=-1;
    double dSpii_Spjj;
    int flag;
    int N;
    num_states=pow((double)M,(double)L);
    N=num_states;
    for(i=0;i<num_states;i++){          /*step 1a*/
     for(j=0;j<num_states;j++){
       Dn[j+i*num_states]=-1;
       }}
    for(i=0;i<num_states;i++) {         /*step 2 */
      for(j=i+1;j<num_states;j++) {
        flag=1;
        for(pi=0;pi<M;pi++) {
           Spi=PM[pi+i*M];
           for(pj=0;pj<M;pj++) {
             Spj=PM[pj+j*M];
             if (Spi==Spj) {
               Spi_stbl_idx=i+pi*N;
               Spj_stbl_idx=j+pj*N;
               d2=0;
               for(k=0;k<nsym;k++) {
                 dif_re=stbl.output[Spi_stbl_idx*nsym+k].re
                        -stbl.output[Spj_stbl_idx*nsym+k].re;
                 dif_im=stbl.output[Spi_stbl_idx*nsym+k].im
                        -stbl.output[Spj_stbl_idx*nsym+k].im;
                 d2+=dif_re*dif_re+dif_im*dif_im;
                 }
               dSpii_Spjj=d2;
               if (flag==1) {dij_min=dSpii_Spjj;flag=0;};
               if (dSpii_Spjj<dij_min) dij_min=dSpii_Spjj;
               }
        } }
      if (flag==0) {
          Dn[j+i*num_states]=dij_min;
          Dn[i+j*num_states]=dij_min;
          }
    } }
}

          /* Update Squared Minimum Distance Matrix, step 3 */
```

```
void updt_Dn(double *Dn,double *Dn_old,int *PM,stable stbl,
             int M,int L,int nsym) {
   int num_states,i,j,k;
   double pi,pj;                    /*predecessor i and j*/
   int Spi,Spj;                     /* State pi and pj */
   double dSpii_Spjj,dij_min;       /*sqrd distance*/
   double d2,dif_re,dif_im;
   int Spi_stbl_idx;
   int Spj_stbl_idx;
   int N, flag;
   N=pow((double)M,(double)L);      /* number of states */
   for (i=0;i<N;i++) {
     for(j=0;j<N;j++) {
      Dn_old[j+i*N]=Dn[j+i*N];
      }}
   for(i=0;i<N;i++){
     for(j=i;j<N;j++){
       flag=1;
       for(pi=0;pi<M;pi++){
          Spi=PM[pi+i*M];
          for(pj=0;pj<M;pj++) {
            Spj=PM[pj+j*M];
            if (Dn_old[Spj+Spi*N]>0) {
              Spi_stbl_idx=i+pi*N;
              Spj_stbl_idx=j+pj*N;
              d2=0;
              for(k=0;k<nsym;k++) {
                dif_re=stbl.output[Spi_stbl_idx*nsym+k].re
                       -stbl.output[Spj_stbl_idx*nsym+k].re;
                dif_im=stbl.output[Spi_stbl_idx*nsym+k].im
                       -stbl.output[Spj_stbl_idx*nsym+k].im;
                d2+=dif_re*dif_re+dif_im*dif_im;
                }
              dSpii_Spjj=d2;
              if (flag==1 ) {
                  if (Dn_old[Spj+Spi*N]==-1) dij_min=dSpii_Spjj;
                  else dij_min=Dn_old[Spj+Spi*N]+dSpii_Spjj;flag=0;
                  }
              if (Dn_old[Spj+Spi*N]==-1) {
                   if( dSpii_Spjj <= dij_min ) dij_min=dSpii_Spjj;
                else
                   if ((Dn_old[Spj+Spi*N]+dSpii_Spjj) <= dij_min)
                      dij_min=Dn_old[Spj+Spi*N]+dSpii_Spjj;
                }
```

```
            }
         }}
      if (i==j && flag==0) {
        if(Dn[j+i*N]==-1)  Dn[j+i*N]=dij_min;
        else if(dij_min<Dn[j+i*N]) Dn[j+i*N]=dij_min;
        }
      if(flag==0 && j>i ) {
           Dn[j+i*N]=dij_min;
           Dn[i+j*N]=dij_min;
           }
      }}
}
                              /* Status of Dn, step 4   */
void stat_Dn(double *Dn,int M,int L,int *flag,double *dmin) {
 int i,j,num_states;
 double dmin_tmp;
 double flag_tmp=0;
 num_states=pow((double)M,(double)L);
 dmin_tmp=Dn[0];
 for (i=1;i<num_states;i++) {
   if (Dn[i+i*num_states]< dmin_tmp) dmin_tmp=Dn[i+i*num_states];
   }
 *dmin=dmin_tmp;
 if (dmin_tmp>0) {
   for (i=0;i<num_states;i++) {
    for (j=0;j<num_states;j++) {
     if (Dn[i,j]<dmin_tmp) flag_tmp=1;
    }}}
  else flag_tmp=1;
 *flag=flag_tmp;
}
```