EXPERIMENTAL RESULTS AND ISSUES ON EQUALIZATION FOR NONLINEAR MEMORYCHANNEL --PRE-CURSOR ENHANCED RAM-DFE CANCELER

Lu Yuan
Dr. James LeBlanc

NMSU-ECE-98-004 August, 1998

EXPERIMENTAL RESULTS AND ISSUES ON EQUALIZATION FOR NONLINEAR MEMORY CHANNEL --PRE-CURSOR ENHANCED RAM-DFE CANCELER

BY

LU YUAN

A Thesis Submitted to the Graduate School

in Partial Fulfillment of the Requirements

for the Degree

Master of Science in Electrical Engineering

New Mexico State University

Las Cruces, New Mexico

August, 1998

Copyright 1998 by Lu Yuan

"Experimental results and issues on equalization for nonlinear memory channel -- Precursor Enhanced RAM-DFE Canceler," a thesis prepared by Lu Yuan in partial fulfillment of the requirements for the degree, Master of Science in Electrical Engineering, has been approved and accepted by the following:

Timothy J. Pettibone
Dean of the Graduate School

James P. LeBlanc
Chair of the Examining Committee

Date

Committee in charge:

Dr. James P. LeBlanc, Chair

Dr. Phillip L. DeLeon

Dr. Tonghui Wang

DEDICATION

To my parents and Haibo for their love.

ACKNOWLEDGMENTS

I am very grateful to my advisor Dr. James P. LeBlanc, for his help and guidance in the development and preparation of this thesis. I am truly indebted to Dr. Phillip L. DeLeon, who gave me previous advice on adaptive digital signal processing issues. I would like to thank Dr. Tonghui Wang for his guidance as member of my committee. I am especially grateful to Lawrence Alvarez for his help in the setting up of hardware. Also a special thank to the people in the Center for Space Telemetry and Telecommunications Systems at New Mexico State University for their assistance.

I would like to extend my appreciation to the Sandia National Laboratory, their financial support is gratefully acknowledged.

VITA

June 29, 1974 -- Born in Nantong, P.R. China

1996 -- B. Eng. Degree in Radio Engineering from Southeast University, Nanjing, P. R. China

1996-1997 -- Master of Science program in Radio Engineering of Southeast University, Nanjing, P.R. China

1997-1998 -- Research Assistant, Center for Space Telemetry and Telecommunications Systems, Department of Electrical and Computer Engineering, New Mexico State University

PUBLICATIONS

[1]"Forward CDMA Channel Convolutional Encoder", Proceeding of the 11th Nanjing District Graduate Conference on Communication & Techniques, Nanjing, 1996.

[2]"Basic Techniques of Wireless Radio Communication", International Academic Developments, April, 1997.

FIELD OF STUDY

Major Field: Electrical Engineering (Telecommunications and Telemetry)

ABSTRACT

EXPERIMENTAL RESULTS AND ISSUES ON

EQUALIZATION FOR NONLINEAR MEMORY CHANNEL

-- PRE-CUSOR ENHANCED RAM-DFE CANCELER

BY

LU YUAN

Master of Science in Electrical Engineering

New Mexico State University

Las Cruces, New Mexico, 1998

Dr. James P. LeBlanc, Chair

This thesis investigates the effects of the High Power Amplifier (HPA) and the filters over a satellite or telemetry channel. The Volterra series expression is presented for the nonlinear channel with memory, and the algorithm is based on the finite-state machine model. A RAM-based algorithm operating on the receiver side -- Pre-cursor Enhanced RAM-FSE Canceler (PERC) is developed. A high order modulation scheme -- 16-QAM is used for simulation, the results show that PERC provides an efficient and reliable method to transmit data on the bandlimited nonlinear channel.

The contribution of PERC algorithm is that it includes both pre-cursors and post-cursors as the RAM address lines, and suggests a new way to make decision on the pre-addresses. Compared with the RAM-DFE structure that only includes post-addresses, the BER versus *Eb/N0* performance of PERC is substantially enhanced. Experiments are performed for PERC algorithms with different parameters on AWGN channels, and the results are compared and analyzed.

The investigation of this thesis includes software simulation and hardware verification. Hardware is setup to collect actual TWT data. Simulation on both the software-generated data and the real-world data are performed. Practical limitations are considered for the hardware collected data. Simulation results verified the reliability of the PERC algorithm.

This work was conducted at NMSU in the Center for Space Telemetering and Telecommunications Systems in the Klipsch School of Electrical and Computer Engineering Department and is supported by grant AX-6425 from Sandia National Labs.

Table of Contents

LIST OF FIGURES ..... **X**

LIST OF ABBREVIATIONS ..... **XII**

CHAPTER 1 INTRODUCTION AND BACKGROUND **1**

1.1 PROBLEM DESCRIPTION 1

1.2 REVIEW OF PREVIOUS WORK 2

1.3 SCHEME DESCRIPTION AND OVERVIEW OF CHAPTERS 4

CHAPTER 2 THE CHANNEL MODEL **6**

2.1 16-QAM 7

2.2 FILTER MODEL 8

2.3 TWTA MODEL 10

2.4 VOLTERRA MODEL 13

CHAPTER 3 PERC ALGORITHM **17**

3.1 SCHEME OUTLINE 17

3.2 ADAPTIVE FILTERING 19

3.3 LMS ALGORITHM AND NLMS ALGORITHM 20

3.3.1 Wiener Filter *20*

3.3.2 Steepest Descent Algorithm *22*

3.3.3 LMS algorithm *23*

3.3.4 NLMS *24*

3.4 FRACTIONALLY SPACED EQUALIZER 25

3.4.1 FSE theory *25*

3.4.2 FSE training *28*

3.5 RAM STRUCTURE FOR PERC 30

CHAPTER 4 SIMULATION RESULTS 35

CHAPTER 5 HARDWARE DATA COLLECTION AND SIMULATION 44

5.1 HARDWARE SETUP DESCRIPTION AND DATA COLLECTION 44

5.2 HARDWARE GENERATED DATA RESULTS 48

CHAPTER 6 CONCLUSIONS AND RECOMMENDATIONS 57

REFERENCES 60

APPENDIX A. HARDWARE SETUP AND CONFIGURATION 62

APPENDIX B. CHARACTERISTICS OF TWTA AND SSPA 66

APPENDIX C. CHARACTERISTICS OF THE FILTER 67

APPENDIX D. SIMULATION SOURCE CODE 69

LIST OF FIGURES

FIGURE 2-1 LOW-PASS EQUIVALENT MODEL OF THE CHANNEL 6

FIGURE 2-2 16-QAM MODULATION BLOCK DIAGRAM 7

FIGURE 2-3 16-QAM CONSTELLATION AND TIMING 8

FIGURE 2-4 BUTTERWORTH FILTER FREQUENCY RESPONSE 9

FIGURE 2-5 BUTTERWORTH FILTER IMPULSE RESPONSE 10

FIGURE 2-6 TWTA INPUT-OUTPUT CURVE 12

FIGURE 2-7 THE EFFECT OF WARPING AND CLUSTERING OF TWTA 13

FIGURE 2-8 FSM MODEL OF THE CHANNEL 16

FIGURE 3-1 STRUCTURE OF THE RECEIVER 18

FIGURE 3-2 ADAPTIVE INVERSE MODELING 19

FIGURE 3-3 ERROR SURFACE 21

FIGURE 3-4 COMMUNICATION SYSTEM MODEL 25

FIGURE 3-5 EQUIVALENT MODELS OF FSE 28

FIGURE 3-6 FSE TRAINING DIAGRAM 29

FIGURE 3-7 RAM TRAINING DIAGRAM 31

FIGURE 3-8 PERC BLOCK DIAGRAM 33

FIGURE 4-1 THE RECEIVED DATA 36

FIGURE 4-2 LEARNING CURVE OF FSE 37

FIGURE 4-3 DATA AFTER FSE 38

FIGURE 4-4 DATA BEFORE THE DECISION DEVICE OF PERC 39

FIGURE 4-5 BER VS. EB/N0 41

FIGURE 4-6 RAM ACCESS TIMES COMPARISON 43

FIGURE 5-1 HARDWARE SETUP DIAGRAM 45

FIGURE 5-2 HARDWARE CONNECTION DIAGRAM 46

FIGURE 5-3 MODULATOR AND DEMODULATOR CONNECTION 47

FIGURE 5-4 THE TRANSMITTED SIGNAL (HARDWARE) 49

FIGURE 5-5 THE RECEIVED SIGNAL (HARDWARE) 50

FIGURE 5-6 TAPS OF FSE 52

FIGURE 5-7 FSE OUTPUT OF REAL DATA SIMULATION 53

FIGURE 5-8 BER VERSUS EB/N0 FOR REAL-DATA 54

FIGURE 5-9 PERC(1,2) FOR SOFTWARE GENERATED-DATA AND REAL-DATA SIMULATION 55

LIST OF ABBREVIATIONS

AM	Amplitude Modulation
AWGN	Additive White Gaussian Noise
ASCII	American Information Code for Information Interchange
bps	Bit Per Second
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
dB	Decibels
DC	Direct Current
DFE	Decision Feedback Equalizer
Eb/No	Bit Energy to Noise-Spectrual-Density Ratio
Es/No	Symbol Energy to Noise-Spectral-Density ratio
FIR	Finite Impulse Response
f_s	Sampling Frequency
FSE	Fractionally-Spaced Equalizer
FSM	Finite-state machine
Hz	Hertz
HP	Hewlett Packard
HPA	High Power Amplifier
IF	Intermediate Frequency

ISI	InterSymbol Interference
LMS	Lease Mean Square
MLSE	Maximum Likelihood Sequence Estimation
MSE	Mean square error
NLMS	Normalized least mean square
NMSU	New Mexico State University
PERC	Pre-cursor Enhanced RAM-DFE Canceler
PM	Phase Modulation
PRBS	Pseudo Random Bit Sequence
PSD	Power Spectral Density
PSK	Phase Shift Keying
QAM	Quadrature amplitude modulation
RF	Radio Frequency
SER	Symbol Error Rate
SNR	Signal-to-Noise Ratio
SSPA	Solid State Power Amplifier
TDRSS	Tracking and Data Relay Satellite System
TWTA	Traveling Wave Tube amplifier
V	Volts

CHAPTER 1 INTRODUCTION AND BACKGROUND

1.1 Problem description

With the rapid development of satellite communications, the frequency band is becoming more and more congested, the need to transmit more data through the bandwidth limited channel is becoming more and more compelling. To meet these needs, it is needed to either transmit data at a faster symbol rate or design higher order modulation schemes, which are more bandwidth efficient, such as Quadrature Amplitude Modulation (QAM). However, increasing symbol rate will also increase the Inter Symbol Interference (ISI), and the QAM scheme faces decoding difficulty because of the High Power Amplifier (HPA) involved in the satellite channel. To achieve power efficiency, the HPA works in its saturation region, which brings nonlinearity to the system. The QAM constellation will be distorted both in amplitude and in phase. Through bandlimited channels, the received data will be a warped constellation of clusters. This research is focused on a nonlinear equalizer -- Pre-cursor Enhanced RAM-DFE Canceler (PERC), which provides a method to successfully use the higher-order modulation scheme through nonlinear memory channel.

1.2 Review of previous work

The methods proposed for transmission through nonlinear memory channel can be divided into two categories: operation on the transmitter and operation on the receiver.

The basic idea of the transmitter-based techniques is pre-distortion, it performs an "inverse" nonlinear transform before the HPA such that the nonlinearity caused by HPA can be canceled. Both continuous-waveform predistortion which consists of an analog device [1] and data predistortion before modulator [2][3][4] have been proposed. The continuous-waveform predistorters are more complex and expensive, less flexible, and their performance is not as good as their digital counterparts [2]. Memoryless predistortion and predistortion with memory techniques are developed for digital data predistortion. Saleh and Salz [5] proposed a pioneering paper about memoryless predistortion, the compensator pre-warps the source constellation so that after the nonlinear transformation of HPA, the desired constellation can be achieved. However, in bandlimited cases, the clustering due to the nonlinear ISI of the channel can not be removed by such a memoryless device. A predistortion with memory scheme is initially presented by E. Biglieri, et al, in [2], based on the theory of pth-order inverse systems. A. Bernardini and S. De Fina [3] developed the idea by setting up the structure of two linear blocks with memory separated by a memoryless nonlinearity device. G. Lazzarin [4] further simplified the structure, the predistortion part is composed only of digital filters and memoryless nonlinear devices that works at the symbol rate. The predistortion with memory

schemes above are all based on the Volterra Model of the system. The transmitter-based techniques require a local receiver at the transmitter, and a feedback path to adaptively adjust the predistortion parameters, which increases the complexity and power consumption of the transmitter.

Schemes focused on the receiver side are proposed herein attempting to minimize the effects of the nonlinearity, ISI and noise. M. F. Meysiya, et al. [6] proposed the maximum likelihood sequence estimation (MLSE) scheme, which is based on Viterbi Algorithm, however, because of its complexity, it is not practical for implementation of very high data rates. Nonlinear equalizer schemes based on Volterra Model are proposed in [7] by D. D. Falconer and in [8] by S. Benedetto. In [9], an FSE-Volterra combined equalizer is suggested, which has a better performance. The Volterra equalizers act as an "inverse" of the nonlinear memory channel, but the noise in the channel is also amplified and distorted [10]. To prevent the excessive noise enhancement, equalizers that involve a random access memory (RAM) as a look-up table are developed. In these schemes, the local decisions on the received symbols are used as the address lines to access the RAM, and the content of the RAM compensates for the nonlinear memory channel. By adding the compensation part to the incoming distorted symbols, the desired symbols can be recovered. Different approaches are made to decide the address lines of the RAM. The RAM-DFE was proposed by K. Fisher, et al. In [11], which employs the Decision feedback structure, the past decisions are used as the address lines (post-addresses only) to access the RAM. However, often in the channel with ISI, both

previous and future symbols are interfering the current symbol, so a better approach is to also involve current and future symbols as address lines, in other words, the RAM address lines involve both post-addresses and pre-addresses. RAM-canceler schemes are proposed by E. Bigieri, et al. in [12] for voiceband data transmission, and O. Agazzi and N. Seshadri in [13] proposed schemes for magnetic recording channels, tentative symbol decisions are made on the current and future symbols that serve as pre-addresses. However, the inaccuracy of the tentative decision limits its BER performance. To make more reliable decision on current and future symbols, PERC algorithm is proposed.

1.3 Scheme description and overview of chapters

The PERC algorithm is first suggested by J. LeBlanc et al, in [10], it is a RAM-based algorithm operating on the receiver side. It provides a more reliable way to make decisions on the pre-addresses by testing over all the possible pre-address combinations and picking the best combination to make local decision on the current symbol. A Fractionally Spaced Equalizer (FSE) is involved before the RAM structure for synchronizing considerations.

It will be convenient to study on the lowpass equivalent discrete time channel model of the bandpass satellite system channel, Chapter 1 presents the lowpass equivalent model of the system, and also describes the models used in simulation for each part of the channel. The research work includes two parts, software simulation and hardware verification. The algorithm is tested by the software simulation first, followed by software implementation of PERC on the real data collected by

hardware. Chapter 3 presents a description about the algorithm, software simulation is presented in Chapter 4. Chapter 5 gives the outline of hardware setup and real-data implementation results, practical factors are considered for real-data. Chapter 6 gives conclusions and suggestions for future work on this topic. A detailed description about the setup and configuration of the each equipment in the hardware is provided in Appendix A. The power input-output curve for the TWTA and SSPA used in the hardware is measured and presented in Appendix B. The characteristics of the BLP-1.9 filter used in the hardware setup are provided in Appendix C. Finally the source codes in MATLAB for the simulation is given in Appendix C.

CHAPTER 2 THE CHANNEL MODEL

The channel of study is chosen to be representative of a satellite or telemetry channel which uses a nonlinear amplifier. The communication system is a bandpass system, and Traveling Wave Tube Amplifier working at S-band (1550-5200MHz). It will be convenient to study on its low-pass equivalent model [14]. The low-pass equivalent model of the channel can be considered as:

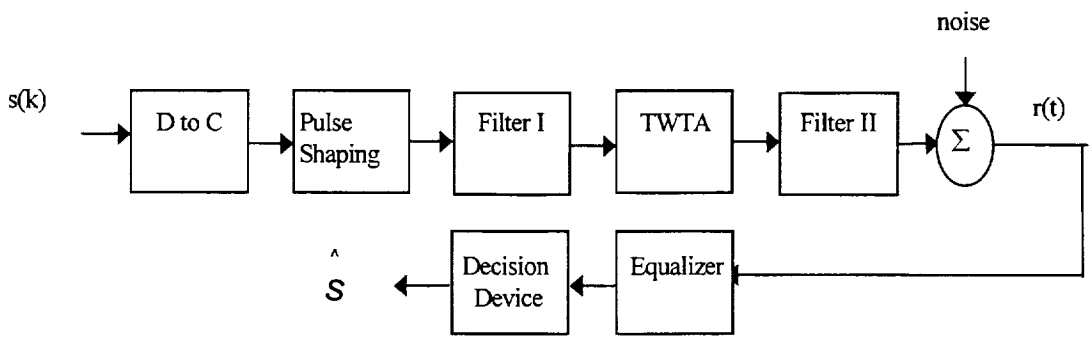


Figure 2-1 Low-pass equivalent model of the channel

Other parts of the system that are not included in this model, such as the modulator, the demodulator and so on, are considered to be ideal. *s(k)* is the digital symbol sequence after modulation, the D/C converter is assumed to be ideal, the pulse shaping is a rectangular function of unit amplitude over a time period of length *T* (symbol interval of *s(k)*). A Traveling Wave Tube Amplifier (TWTA) is contained in the bandlimited Additive White Gaussian Noise (AWGN) channel. Filter I is a lowpass pre-filter to limit the TWTA input noise band, filter II is a lowpass post-filter to limit the spectral emissions of the output of TWTA into other frequency bands.

2.1 16-QAM

Because of the nonlinear effect of the TWT amplifier, a typical modulation format for communications is *M* phase shift keying (*M*-PSK). However, Sixteen State Quadrature Amplitude Modulation (16-QAM) has recently been considered for communications. The modulation scheme studied in this thesis is 16-QAM.

The functional block diagram of QAM modulator is:

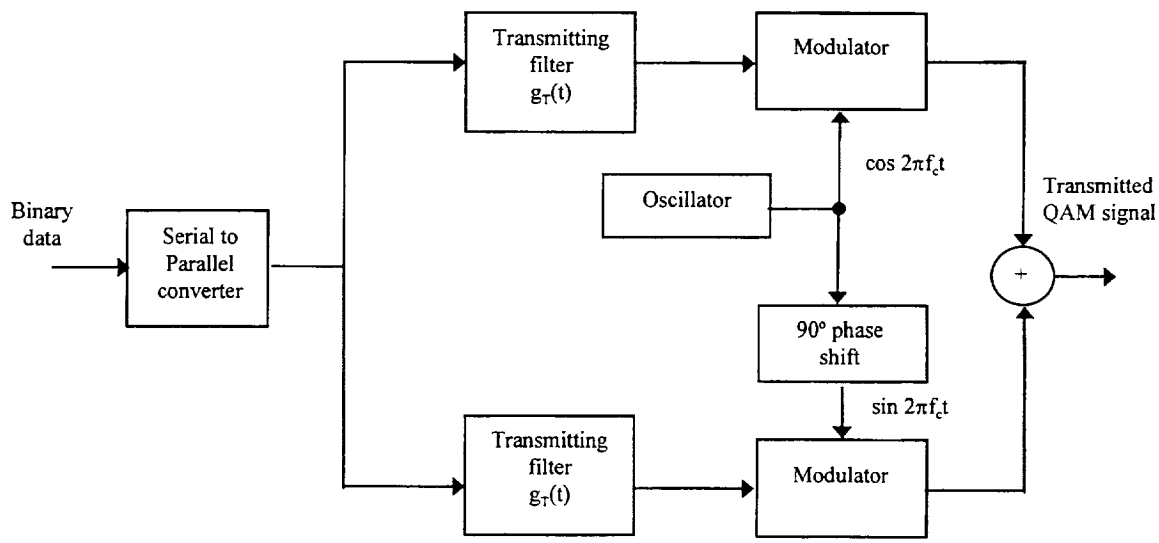


Figure 2-2 16-QAM modulation block diagram

The transmitted signal waveform is [15]:

$$u_m(t) = A_{mc}g_T(t)cos(2\pi f_c t) + A_{ms}g_T(t)sin(2\pi f_c t),\ m=1,2,\ ...,\ M \qquad (2.1)$$

For 16-QAM, $M=4$, The signal can be represented in the form of a two-dimensional vector $\boldsymbol{s_m}=[A_{mc}\ A_{ms}]$. The following figure illustrates the state mapping and serial input timing.

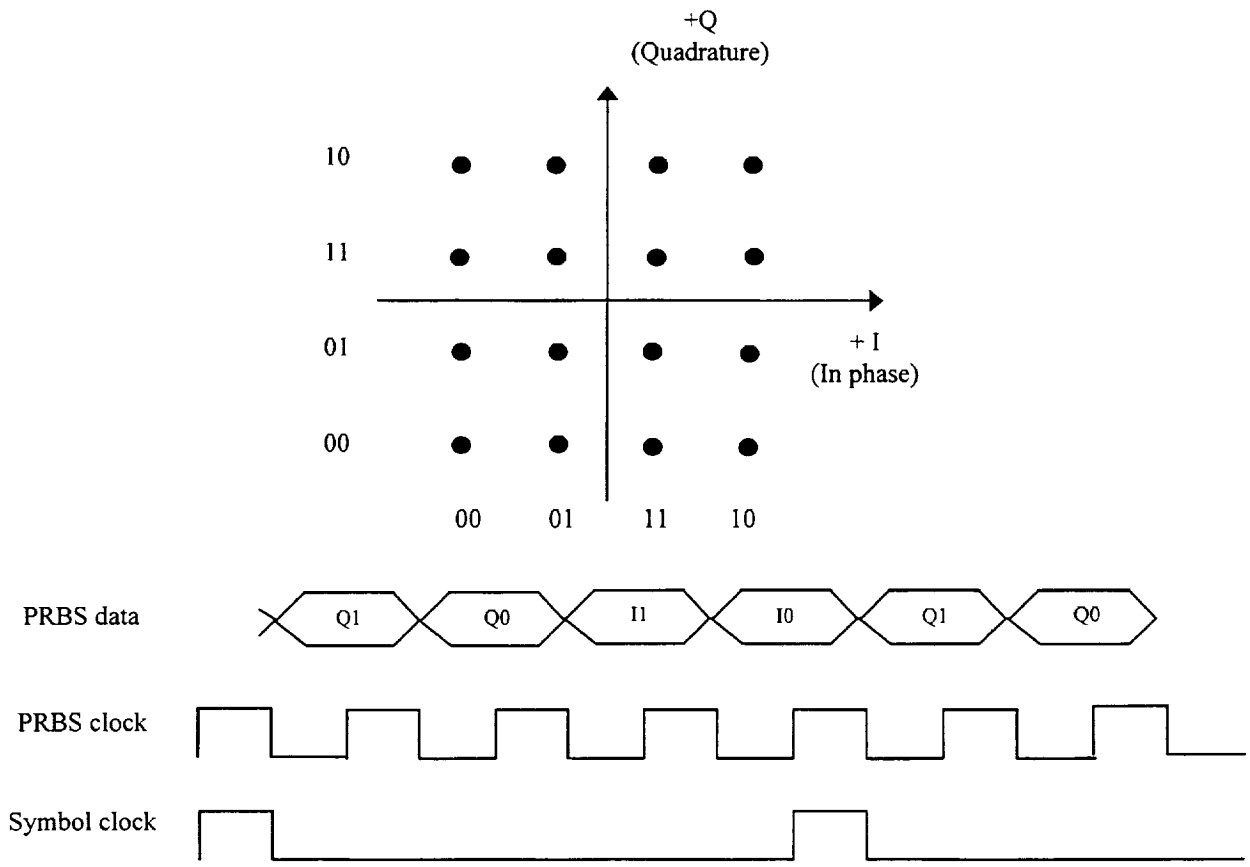


Figure 2-3 16-QAM constellation and timing

Four bits are mapped into one of sixteen I-Q vector states, and the four bits are Gray coded. I is the in phase component, and Q is the quadrature component.

2.2 Filter Model

In the simulation, both the pre-filter and the post-filter are modeled as 6th-order low-pass Butterworth filters with cutoff frequency equals to the symbol rate of *s(k)*. The magnitude response of Butterworth lowpass filters is maximally flat in the passband, it is an all-pole filter. For an *Nth*-order lowpass filter, the first *(2N-1)*

derivatives of the magnitude-squared function are zeroes at $\Omega=0$. The magnitude-squared function decreases monotonically with Ω in the passband and the stopband. The magnitude-squared function for a continuous-time Butterworth lowpass filter is of the form:

$$|H_c(j\Omega)|^2 = \frac{1}{1+(j\Omega / j\Omega_c)^{2N}} \tag{2.2}$$

The squared function is sketched as Figure 2-4:

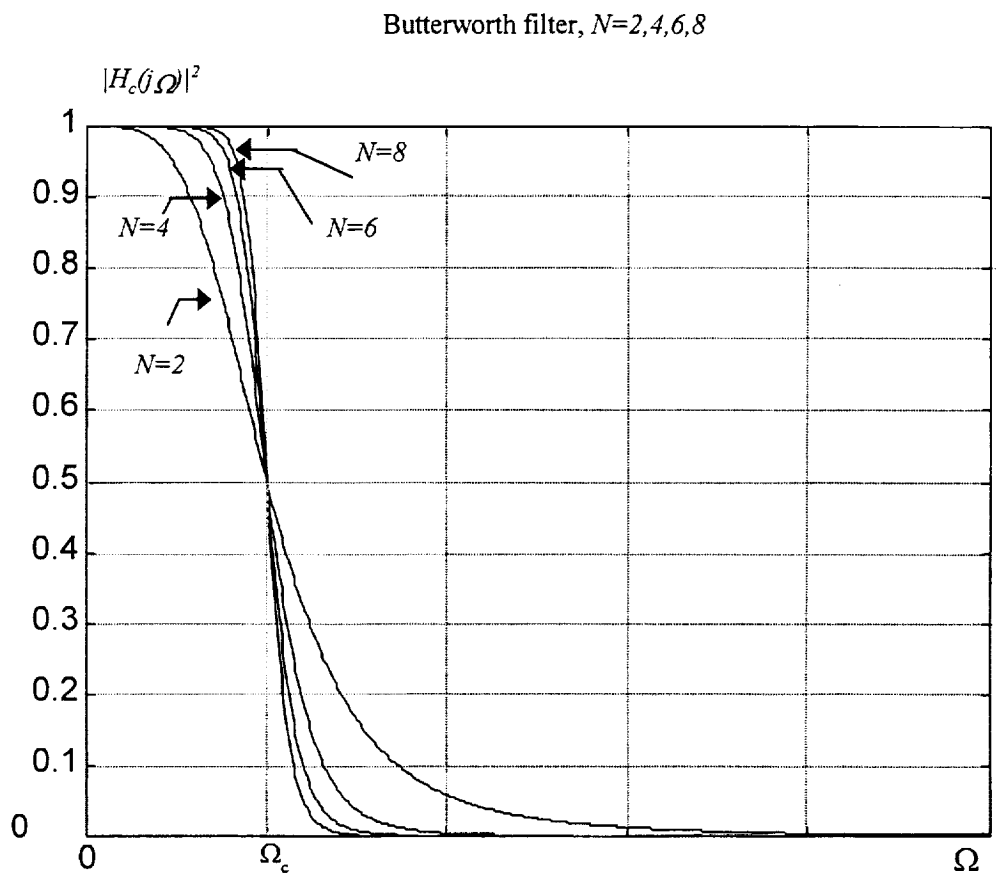


Figure 2-4 Butterworth filter frequency response

As the filter order N increases, the filter characteristic curve becomes sharper, the roll off rate is about *-20NdB/decade*. For all N, the magnitude-squared function equals to unity at $\Omega=0$, at the cutoff frequency Ω_c, the magnitude squared function equals to *1/2*. The impulse response of 6th-order Butterworth filter is:

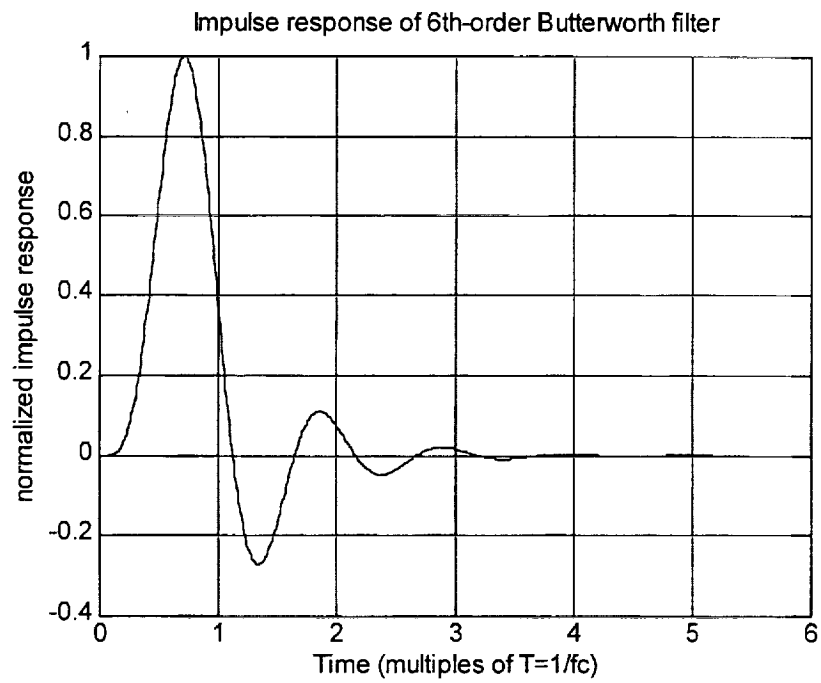


Figure 2-5 Butterworth filter impulse response

2.3 TWTA model

The High Power Amplifier (HPA) used in the satellite communication is usually a Traveling Wave Tube Amplifier (TWTA). The TWTA is modeled as a nonlinear memoryless frequency-independent amplifier, it has the effect of nonlinear distortion in both the amplitude and the phase [16].

Assume the normalized baseband input to the TWTA is $x = \rho \times e^{j\theta}$, where ρ is the amplitude and θ is the phase of the baseband input signal. The TWTA will exhibit AM/AM and AM/PM conversion effects. The normalized baseband output of the TWTA will be $y = A \times e^{j\theta + j\Phi}$, according to the model due to Saleh [16], the amplitude A and phase shift Φ is given by:

$$A(\rho) = \frac{\alpha_a \rho}{1 + \beta_a \rho^2} \quad \text{(AM/AM conversion)} \tag{2.3}$$

$$\Phi(\rho) = \frac{\alpha_\phi \rho^2}{1 + \beta_\phi \rho^2} \quad \text{(AM/PM conversion)} \tag{2.4}$$

For very large ρ, $A(\rho)$ approaches $\alpha_a/\beta_a\rho$, and $\Phi(\rho)$ approaches constant α_ϕ/β_ϕ.

In the simulation, the parameters are set as the following due to Saleh [16]:

$\alpha_a = 1.9638$ $\qquad$ $\alpha_\phi = 2.5293$

$\beta_a = 0.9945$ $\qquad$ $\beta_\phi = 2.8168$

The plot of the amplitude and phase conversion with the parameters above is shown in Figure 2-6. The input amplitude is normalized so that the saturation point is at unity. In applications, the TWTA is driven around saturation to achieve high power efficiency.

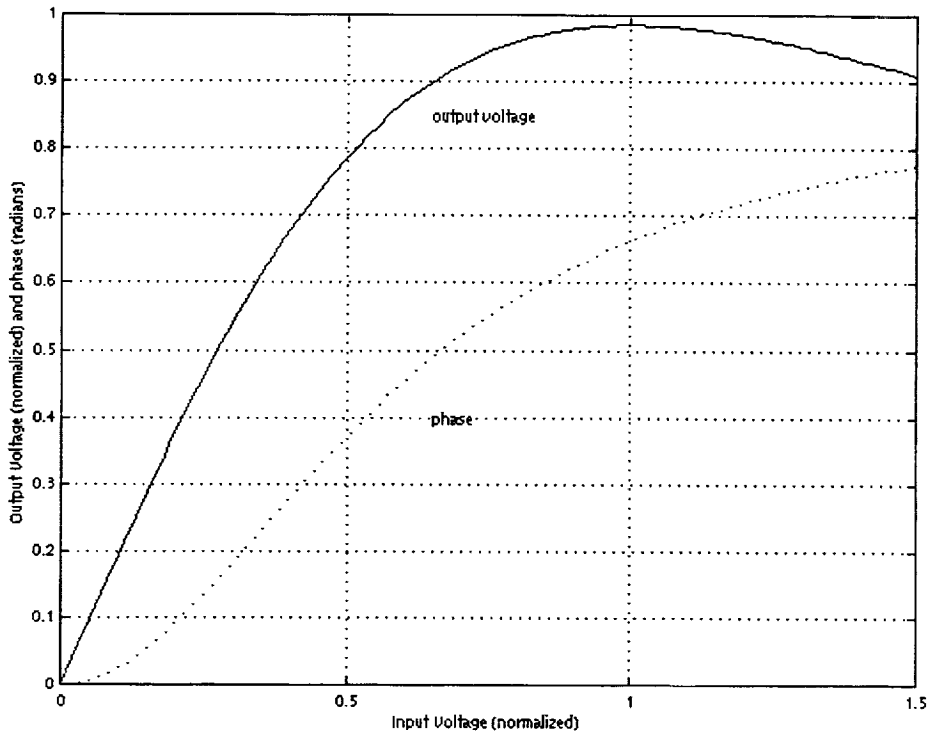


Figure 2-6 TWTA input-output curve

For power efficiency considerations, the 16-QAM constellation is chosen such that the average amplitude is 1, which drives the TWTA at the saturation point. The channel modeled as Figure 2-1 will have two effects on the QAM constellations: warping, which is due to the nonlinearity, and clustering, which is due to the ISI introduced by the filtering [17]. At the receiver side, the signal constellation is not on rectangular grid as 16-QAM, the points at outer corner are compressed, so the outer constellation forms a circle. Due to the Intersymbol interference (ISI) caused by filtering, each constellation point becomes a cluster, the plot in Figure 2-7 is an

illustration of the effect of warping and clustering of this channel in noise free case for 16-QAM.

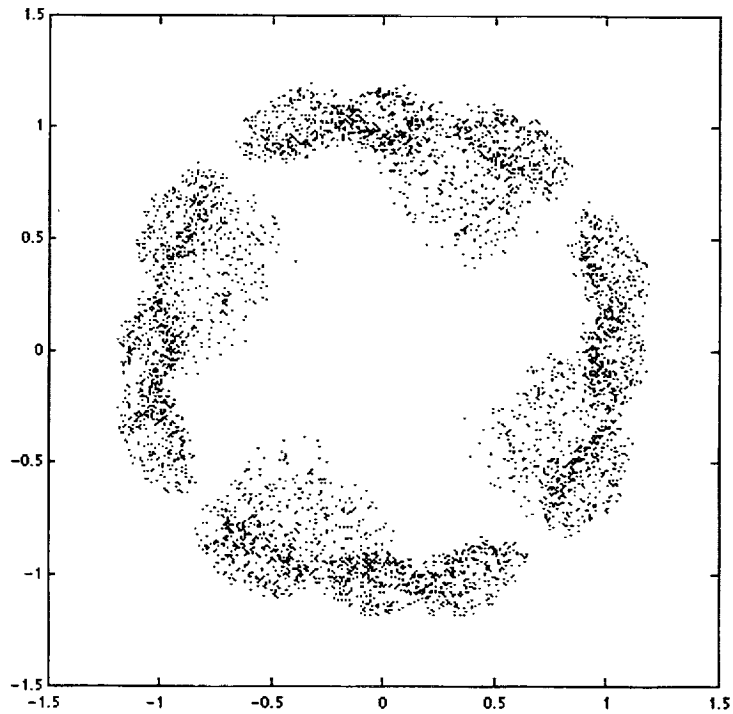


Figure 2-7 the effect of warping and clustering of the channel

2.4 Volterra Model

Note that although the source of nonlinearity (TWTA) is modeled as memoryless, the effect of the filters in the channel model will cause ISI, and will make the overall channel a nonlinear system with memory. The Volterra model approach provides a useful description on the nonlinear channel with memory.

Assume the transmitted signal is s_k, the continuous baseband waveform after pulse shaping is

$$s(t)=\sum_k s_k p(t-kT_s) \tag{2.5}$$

Where p is the pulse shaping function. After passing the pre-filter which has an impulse response of $h_1(t)$, it becomes

$$(s*h_1)(t) = \sum_k s_k (p*h_1)(t-kT_s) = \sum_k s_k \tilde{h}_{1,n-k} \tag{2.6}$$

where

$$\tilde{h}_{1,k} = (p*h_1)(t) \tag{2.7}$$

and

$$\tilde{h}_{1,n-k} = (p*h_1)(t-kT_s) \tag{2.8}$$

The signal in equation (2.6) is the input waveform to the TWTA. Consider the gain function of the TWTA in equation (2.3), its Taylor series expansion can be expressed as:

$$A(\rho) = \alpha_a\rho - \alpha_a \beta_a\rho^3 + \alpha_a \beta_a^2 \rho^5 - \alpha_a \beta_a^3 \rho^7 + \ldots \tag{2.9}$$

so the amplitude of TWTA output is

$$A(\rho) = A[(s*h_1)(t)] = \alpha_a \sum_k s_k \tilde{h}_{1,n-k} - \alpha_a \beta_a \sum_{i,j,k} s_i s_j s_k \tilde{h}_{1,n-i} \tilde{h}_{1,n-j} \tilde{h}_{1,n-k}$$

$$+ \alpha_a \beta_a^2 \sum_{i,j,k,l,m} s_i s_j s_k s_{li} s_m \tilde{h}_{1,n-i} \tilde{h}_{1,n-j} \tilde{h}_{1,n-k} \tilde{h}_{1,n-l} \tilde{h}_{1,n-m} \tag{2.10}$$

TWTA output A(ρ) is then passed through the post-filter which has an impulse response of $h_2(t)$, and the output is:

$$\tilde{s}(t) = A(\rho) * h_2(t) = A[(s * h_1)(t)] * h_2(t) = \sum_k s_k h^{(1)}_{n-k} + \sum_{i,j,k} s_i s_j s_k h^{(3)}_{n-i,n-j,n-k} + \ldots \tag{2.11}$$

where $$h^{(1)}_{n-k} = \alpha_a \tilde{h}_{1,n-k} * h_2, h^{(3)}_{n-i,n-j,n-k} = -\alpha_a \beta_a \tilde{h}_{1,n-i} \tilde{h}_{1,n-j} \tilde{h}_{1,n-k} * h_2 \tag{2.12}$$

and so on [10].

Sampling $\tilde{s}(t)$ at nT_s to get the discrete time output $\tilde{s}_n$ yields:

$$\tilde{s}_n = \sum_k s_k h^{(1)}_{n-k} + \sum_{i,j,k} s_i s_j s_k h^{(3)}_{n-i,n-j,n-k} + .. = \sum_k s_{n-k} h^{(1)}_n + \sum_{i,j,k} s_{n-i} s_{n-j} s_{n-k} h^{(3)}_{ijk} + \ldots \tag{2.13}$$

The terms $h_{n-k}^{(1)}$, $h_{n-i,n-j,n-k}^{(3)}$, ... are called the Volterra kernels, the characteristics of the system is decided by the Volterra kernels [2]. The stronger the nonlinearity is, the more Volterra kernels are needed to adequately characterize the system, and the length of the channel memory will decide how many terms are needed in the summation over index *i, j, k*, and so on [8].

There exist both linear ISI (first order term) and nonlinear ISI (third and higher order terms) in this model. Only odd-order kernels appear, this is due to the nature of the source of the nonlinearity -- TWTA.

Similarly, the phase function of the TWTA also has a Volterra model, although we only described the amplitude here.

Equation (2.13) suggests that the output $\tilde{s}_k$ is a nonlinear function of the input discrete time signal s_k and its neighboring symbols. Compared with voiceband

channels, satellite channels may have stronger nonlinearity but shorter memory [8], so the function will only involve a few terms centered around k:

$$\tilde{s}_k = f\ (s_{k+n-1}, \ldots, s_k, \ldots s_{k-m}) \tag{2.14}$$

where n and m are decided by the memory length of the channel.

This can be considered as a mapping from some neighboring input signals to an output signal, the mapping is characterized by certain Volterra kernels. Recognizing that the source symbols are drawn from a finite alphabet, we see that the mapping can be implemented by a look-up table -- random access memory (RAM), with the input signals as addresses, and the corresponding output as the data content accessed by that address. This leads to the Finite-State Machine (FSM) model of the channel as described below:

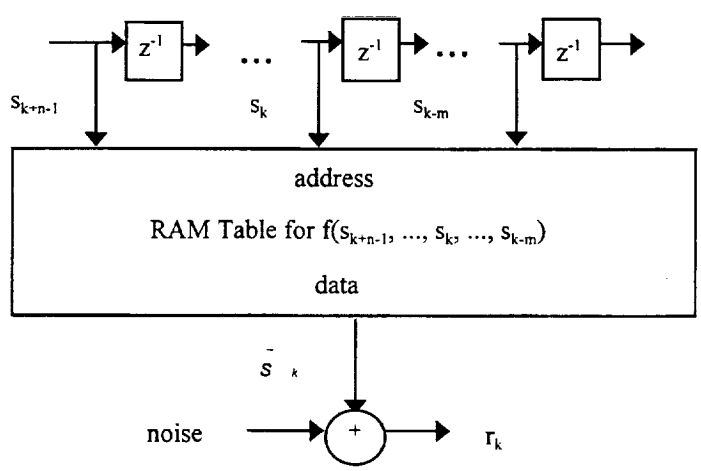


Figure 2-8 FSM model of the channel

The channel is characterized by a deterministic RAM and a noise adder, the current outgoing signal s_k and its neighboring signals decide the state of the model, r_k is the received signal. Additive white Gaussian noise is included.

CHAPTER 3 PERC ALGORITHM

3.1 Scheme outline

The model described in Figure 2-1 is applied in the simulation. A Pre-cursor Enhanced RAM-DFE Canceler (PERC) structure is used at the receiver side to compensate for the nonlinearity and ISI of the channel. As stated at the end of Chapter 1, the received signal is modeled:

$$r_k = \tilde{s}_k + v_k = f\ (s_{k+n-1}, \ldots, s_k, \ldots s_{k-m}) + v_k \tag{3.1}$$

where v_k is the additive white Gaussian noise of the channel.

Define the difference between the desired signal (the signal sent) and the received signal as g:

$$g(s_{k+n-1}, \ldots, s_k \ldots, s_{k-m}) = s_k - (f(s_{k+n-1}, \ldots, s_k \ldots, s_{k-m}) + v_k) \tag{3.2}$$

The receiver needs to compensate for $g(.)$ to recover the desired signal. Notice that, as $f(.)$, $g(.)$ is also a function of $s_{k+n-1}, \ldots, s_k \ldots, s_{k-m}$, it can be implemented by a RAM look-up table at the receiver side. For synchronization considerations, the RAM structure is used in conjunction with a Fractionally Spaced Equalizer (FSE). The FSE helps to fix the delay of the channel and removes linear ISI. In the real-data simulation in Chapter 5, we will see that the FSE will also cancel the rotation effect on the demodulated signal caused by the delay in hardware. The idea of adaptive filtering and FSE will be introduced later in this chapter. The overall structure of receiver is suggested in Figure 3-1:

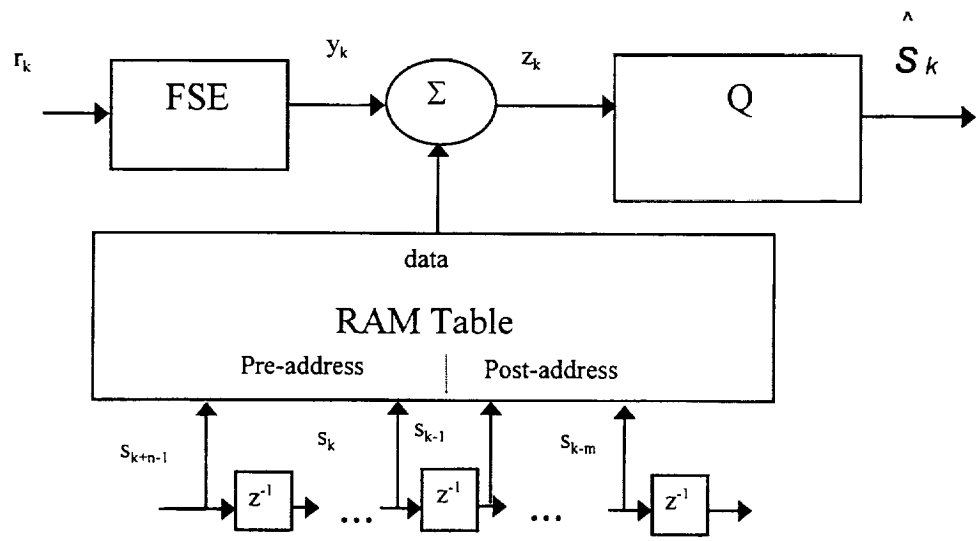


Figure 3-1 structure of the receiver

FSE is the fractionally spaced, linear feed-forward equalizer, and Q is the decision device. The RAM table implements the mapping defined by the function *g(.)*, with address lines s_{k+n-1}, ..., s_{k-m}. s_{k+n-1}, ..., s_k are called pre-addresses, *and* s_{k-1}, ..., s_{k-m} are called post-addresses.

The function *f(.)* and *g(.)*, are decided by the characteristics of the channel, to implement the mapping *g(.)*, thus the characteristics of the channel need to be known. This is achieved by sending training sequence. The idea of training is to send a sequence that is known by the receiver, then by comparing what the receiver receives and what is actually sent, the receiver can get knowledge about the characteristics of the channel. When the channel is relatively stable, and we can assume it will not change before the next time we "train" it. After training, the parameters of the FSE and the RAM are fixed, and the receiver works at a fixed mode for data transmission.

However, when the receiver is working in the fixed mode for data transmission, the pre-addresses $s_k, \ldots, s_{k+n-1}$ are not known, we will focus on how to get the correct local decision of these pre-addresses later in section 3.4. The taps of the FSE are set adaptively, the algorithm used is Normalized Least Mean Square (NLMS) algorithm. We will discuss adaptive filtering issues next.

3.2 Adaptive filtering

The adaptive filter is a self-designing device, which operates in a recursive fashion, and "makes it possible for the filter to perform satisfactorily in an environment where complete knowledge of the relevant signal characteristics is not available." [18] It starts with some initial conditions, following a certain algorithm, after some iterations, converges to the optimum Wiener solution.

One of the applications of adaptive filtering is "inverse modeling". The adaptive filter works as an equalizer, it acts as the best fit (in some sense) of the "inverse" of the channel. Ideally, the overall effect of the channel and the equalizer is a pure delay. The diagram of the inverse modeling is shown in Figure 3-2:

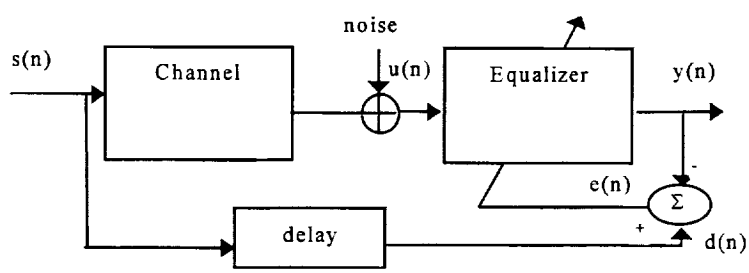


Figure 3-2 adaptive inverse modeling

$u(n)$ is the received signal, $y(n)$ is the output of the equalizer, , *let* $\boldsymbol{w}=[w_0 \ldots w_{M-1}]^T$ be the filter coefficient vector, $\boldsymbol{u}(n)= [u(n) \ldots u(n-M+1)]^T$ be the input vector, so the output is:

$$y(n)=\boldsymbol{w}^H\boldsymbol{u}(n) \tag{3.3}$$

The difference between the desired signal $d(n)$ and $y(n)$ is used to adjust the coefficients of the filter. The adjusting algorithm is Normalized Lease Mean Square (NLMS) algorithm, which is a normalized version of Lease Mean Square (LMS) algorithm.

3.3 LMS algorithm and NLMS algorithm

3.3.1 Wiener Filter

The output of the equalizer $y(n)$ provides an estimate of the desired signal $d(n)$. The difference between them is:

$$e(n)=d(n)-y(n) \tag{3.4}$$

The goal of adjusting the coefficients of the filter is to minimize the mean-square error (MSE). The MSE J is defined as:

$$J=E[e(n)e(n)^*]=E[|e(n)|^2] \tag{3.5}$$

where "E" denotes for the expectation.

$$\begin{aligned} J&=E[e(n)e(n)^*] \\ &=E[(d(n)-y(n))(d(n)-y(n))^*] \\ &=\sigma_d^2-\boldsymbol{w}^H\boldsymbol{p}-\boldsymbol{p}^H\boldsymbol{w}+\boldsymbol{w}^H\boldsymbol{R}\boldsymbol{w} \end{aligned} \tag{3.6}$$

where σ_d^2 is the variance of the desired signal, $\boldsymbol{w}=[w_0 \ldots w_{M-1}]^T$ is the filter coefficient vector, and $\boldsymbol{u}(n)= [u(n) \ldots u(n-M+1)]^T$ is the input vector, $\boldsymbol{p}=E[\boldsymbol{u}(n)d^*(n)]$ is the $M\times 1$ cross correlation vector between the input vector $\boldsymbol{u}(n)$ and the desired signal $d(n)$, and $\boldsymbol{R}=E[\boldsymbol{u}(n)\boldsymbol{u}^H(n)]$ is the $M\times M$ correlation matrix of the input vector.

It can be proved [18] that J will be minimized when

$$\boldsymbol{w}=\boldsymbol{R}^{-1}\boldsymbol{p}= \boldsymbol{w}_o \text{ (optimal Wiener Solution)} \tag{3.7}$$

and

$$J_{min}=\sigma_d^2 - \boldsymbol{p}^H\boldsymbol{R}^{-1}\boldsymbol{p}. \tag{3.8}$$

As an example, Figure 3-3 shows the error surface of a two-tap ($\boldsymbol{w}=[w_0\ w_1]$) filter, mean-squared error J versus tap w_0 and w_1. The bottom of the surface is J_{min}, the corresponding tap vector $\boldsymbol{w}_o$ is the Wiener solution.

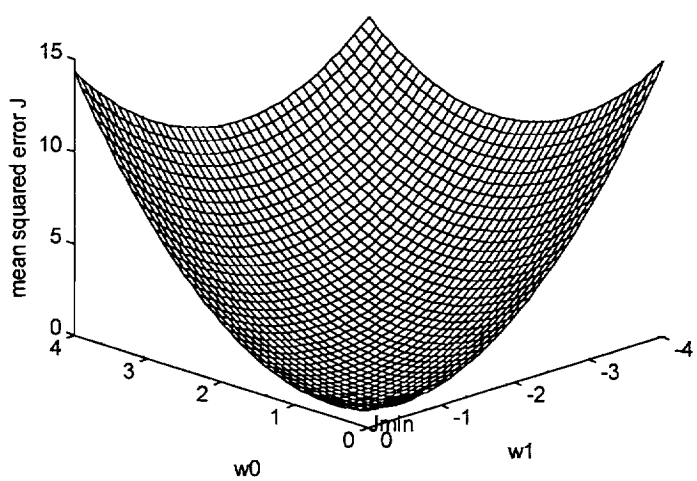


Figure 3-3 error surface

3.3.2 Steepest Descent Algorithm

When M (the order of the filter) is large, it is not convenient to compute the inverse of $\boldsymbol{R}$. The steepest descent algorithm is an algorithm that will search the error performance surface J for its minimum point J_{min}, and the corresponding filter coefficient vector will be the Wiener solution $\boldsymbol{w}_o$.

The steps of the Steepest Descent Algorithm are:

Initialize w(0)

For n=0 to L-1
  compute ∇J (gradient vector)
  w(n+1)= w(n)+μ'[- ∇J]
end;

---

where $$\nabla J=-2E[\boldsymbol{u}(n)e^*(n)] \tag{3.9}$$

The update equation for steepest descent algorithm is

$$\begin{aligned} \boldsymbol{w}(n+1) &= \boldsymbol{w}(n)+\mu' 2E[\boldsymbol{u}(n)e^*(n)] \\ &= \boldsymbol{w}(n)+\mu E[\boldsymbol{u}(n)e^*(n)] \\ &= \boldsymbol{w}(n)+\mu[\boldsymbol{p}-\boldsymbol{R}\boldsymbol{w}(n)] \end{aligned} \tag{3.10}$$

To be stable, the stepsize should satisfy $0<\mu<2/\lambda_{max}$, where λ_{max} is the largest eigenvalue of the correlation matrix $\boldsymbol{R}$.

3.3.3 LMS algorithm

The shortcoming of computing the Wiener solution and the steepest descent algorithm is that they both require knowledge of $\boldsymbol{R}$ and $\boldsymbol{p}$, which are usually unknown. LMS algorithm uses the instantaneous values as estimates of $\boldsymbol{R}$ and $\boldsymbol{p}$.

$$\hat{R} = \boldsymbol{u}(n)\boldsymbol{u}^H(n) \tag{3.11}$$

$$\hat{p} = \boldsymbol{u}(n)d^*(n) \tag{3.12}$$

The update equation for LMS is:

$$\hat{w}(n+1) = \hat{w}(n)+\mu\boldsymbol{u}(n)e^*(n) \tag{3.13}$$

The steps for LMS algorithm are:

Initialize $\hat{w}(0)$

for $n=0$ *to* $L-1$
  $e(n)=d(n)-\hat{w}^H(n)\boldsymbol{u}(n)$
  $\hat{w}(n+1)=\hat{w}(n)+\mu\boldsymbol{u}(n)e^*(n)$
end;

If we choose $0<\mu<2/\lambda_{max}$, LMS algorithm converges in mean, $\lim_{n\to\infty} E[\hat{w}(n)]=\boldsymbol{w}_o$, if we choose $0<\mu<2/tr(\boldsymbol{R})$, "$tr$" denotes the trace of a matrix, LMS converges in mean square, we have

$$J(n)=J_{min}+J_{ex}(n) \tag{3.14}$$

and $\lim_{n\to\infty} J_{ex}=constant$.

3.3.4 NLMS

In the LMS algorithm above, we have $\hat{w}(n+1) = \hat{w}(n) + \mu \boldsymbol{u}(n) e^*(n)$, the correction term $\mu \boldsymbol{u}(n) e^*(n)$ is directly proportional to $\boldsymbol{u}(n)$, which means the correction is sensitive to the power of the input signals. This will affect the convergence rate especially when the power range of input signal is large. To make the convergence rate independent of the input signal power, we normalize the correction by the input signal power, so the update equation becomes

$$\hat{w}(n+1) = \hat{w}(n) + \bar{\mu}\, \boldsymbol{u}(n) e^*(n) / ||\boldsymbol{u}(n)||^2 \tag{3.15}$$

where $||\boldsymbol{u}(n)||^2 = \mathbf{u}^T(n)\mathbf{u}(n)$. We can think of NLMS algorithm as a special case of LMS algorithm which has a time-varying stepsize

$$\mu(n) = \bar{\mu} / ||\boldsymbol{u}(n)||^2 \tag{3.16}$$

To avoid numerical error problems *when* $||\boldsymbol{u}(n)||^2$ is small, a small constant a is added to the denominator, the equation becomes:

$$\hat{w}(n+1) = \hat{w}(n) + \bar{\mu}\, \boldsymbol{u}(n) e^*(n) / (a + ||\boldsymbol{u}(n)||^2) \tag{3.17}$$

Typically, a is chosen to be 0.01.

To achieve convergence in the mean square, it is required that $0 < \bar{\mu} < 2$, NLMS algorithm converges faster than LMS algorithm especially when the data is correlated (large eigen spread) for same level of steady-state misadjustment.

3.4 *Fractionally Spaced Equalizer*

3.4.1 FSE theory

A typical communication system can be modeled as:

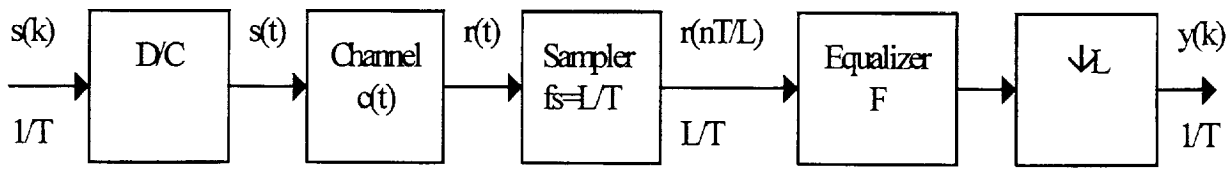


Figure 3-4 communication system model

Where *s(k)* is the source sequence with symbol interval *T*, assume the D/C converter is ideal:

$$s(t) = \sum_{i=-\infty}^{\infty} \delta(t - kT)s(k) \tag{3.18}$$

and the received waveform is

$$r(t) = \sum_{i=-\infty}^{\infty} s(i)c(t - iT) \tag{3.19}$$

r(t) is sampled at rate $f_s = L/T$,

$$r(kT / L) = \sum_{i=-\infty}^{\infty} s(i)c(kT / L - iT) \tag{3.20}$$

A sequence is called baud spaced when the interval between symbols is *T* units of time. In the above model, *s(k)* is baud spaced, when *L*=1, the received waveform *r(t)* is sampled at the same rate as the symbol rate of the baud spaced source sequence *s(k)*, the equalization is called baud spaced equalization. When *L* is

an integer larger than 1, the equalization above is called fractionally spaced equalization. The channel and equalizer taps are T/L spaced.

The system can be viewed as a multi-channel model, i.e., there are L parallel sub-channels, the delay between adjacent sub-channels is T/L. The lth subchannel is:

$$c_l(k) = c(kT + \frac{lT}{L}) \tag{3.21}$$

The signal after sampling the output of the lth subchannel is:

$$r(T(n + \frac{l}{L})) = \sum s(i)c((n-i)T) + l\frac{T}{L}) = \sum s(i)c_l(n-i) \tag{3.22}$$

A common choice for L is 2, when L=2:

$$r(k\frac{T}{2}) = \sum_{i=-\infty}^{\infty} s(i)c(k\frac{T}{2} - iT) \tag{3.23}$$

Denote the even and odd subchannels by c^{even} and c^{odd}, the even and odd samples of the received signal are:

$$r_n^{even} = r(nT) = \sum_{i=-\infty}^{\infty} s(i)c((n-i)T) = \sum_{i=-\infty}^{\infty} s(i)c^{even} \tag{3.24}$$

$$r_n^{odd} = r(nT - \frac{T}{2}) = \sum_{i=-\infty}^{\infty} s(i)c((n-i)T - \frac{T}{2}) = \sum_{i=-\infty}^{\infty} s(i)c^{odd} \tag{3.25}$$

Suppose the length of the equalizer is L_f, and denote the taps equalizer by f, the output of the equalizer is:

$$y(k\frac{T}{2}) = \sum_{i=0}^{L_f-1} f(i\frac{T}{2})r((k-i)\frac{T}{2}) \tag{3.26}$$

It is decimated by 2 to get the baud spaced output, without loss of generality, retain even samples:

$$y(nT) = \sum_{i=0}^{L_f-1} f(i\frac{T}{2})r(nT - i\frac{T}{2}) = \sum_{i=0}^{(L_f-1)/2} [f(iT)r((n-i)T) + f(iT + \frac{T}{2})r((n-i)T - \frac{T}{2})] \tag{3.27}$$

Equation (3.27) suggests that the equalizer can also be viewed as being composed of even and odd subequalizers, denote them by f^{even} and f^{odd}, we have:

$$y(nT) = \sum_{i=0}^{(L_f-1)/2} (f^{even} r^{even} + f^{odd} r^{odd}) \tag{3.28}$$

The output *y(nT)* will be the sum of the output of even and odd sub-equalizers. The above equations suggest that the following structures are equivalent:

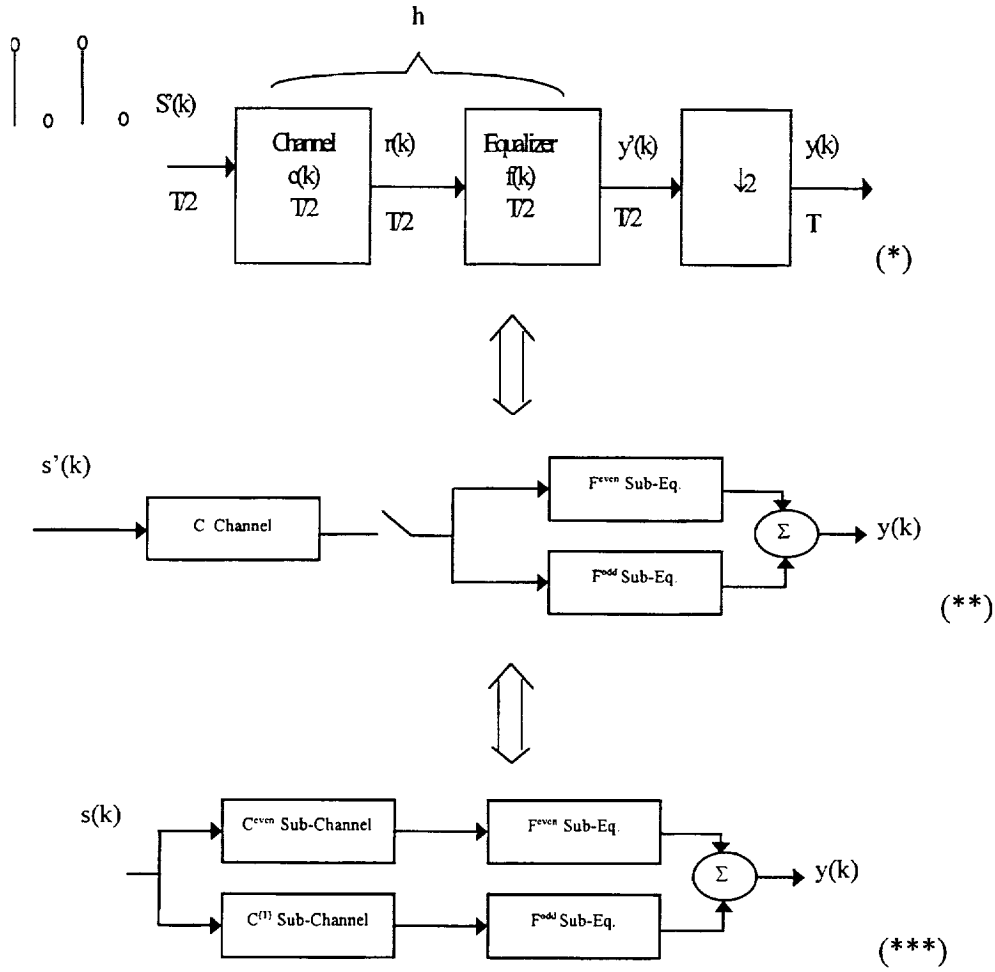


Figure 3-5 Equivalent models of FSE

In model (*), $$s'(k)=\begin{cases} s(k/2), keven \\ 0, kodd \end{cases} \tag{3.29}$$

3.4.2 FSE training

For BSE equalizers, when the channel is Finite Impulse Response (FIR) filter with length L_c, the length of the equalizer needs to be several (usually 3-5) times of L_c. This is not necessary for FSE, the length of the equalizer L_f only need to satisfy

$L_f \geq L_c\text{-}1$. L_f is the sum of the length of the even sub-equalizer Me and the odd sub-equalizer Mo, $Me = \lceil (Lc\text{-}1)/2 \rceil$, $Mo = \lfloor (Lc\text{-}1)/2 \rfloor$, $\lceil x \rceil$ rounds the elements of x to the nearest integers towards infinity, $\lfloor x \rfloor$ rounds the elements of x to the nearest integers towards minus infinity [19].

The FSE is trained to set the coefficients of the filter, the training process of FSE is described in the diagram below.

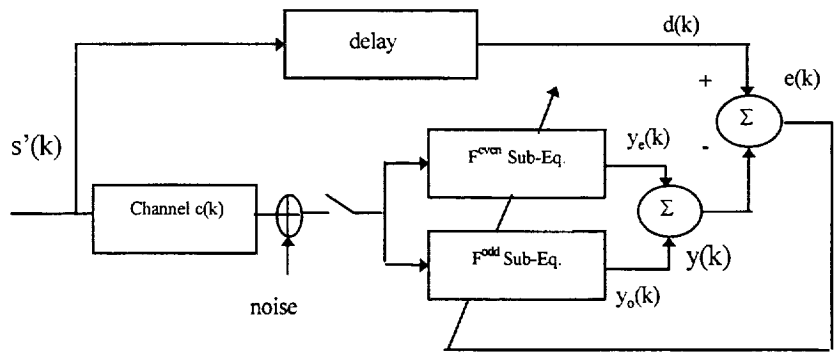


Figure 3-6 FSE training diagram

The FSE is adapted via NLMS algorithm, and the feedback error signal used to adjust the coefficients is

$$e(k)=d(k)-y(k)=d(k)-(y_e(k)+y_o(k)) \tag{3.30}$$

The taps updating process stops when the MSE does not decrease any more.

Experiments are needed to be done to decide the proper delay, the scheme is to test over possible delay values and check the MSE at the end of the adapting process. The delay corresponding to the minimum MSE is finally set to be the correct delay. When the number of possible delays is big, to avoid testing all delays, typically the delay may be set to somewhere in the middle of the FSE, even though

this may be non-optimal. The length of the filter is chosen by a trail and error way, i.e., starts from a short equalizer length, then increase the length and check the MSE, the equalizer length is chosen such that if continue to increase the length, the MSE will not decrease.

The advantages of FSE are: compared with BSE, which has one tap per symbol, FSE ($L=2$) has two taps per symbol, and the channel is also sampled twice the sampling rate of BSE, this helps to avoid aliasing. Also, by doubling the sampling rate, FSE is less sensitive to the sampling clock phase. Another advantage is that the length of FSE taps can be shorter than BSE, and it works much better than BSE when the channel has zeros close to the unit circle. After FSE adapting process, the taps are fixed, it will be working in a fixed mode, and the training process for the RAM begins.

3.5 RAM structure for PERC

The RAM will compensate for the difference between the output of FSE y_k and the desired signal s_k, which is a delayed version of the transmitted 16-QAM symbols. The content of the RAM is setup by training, i.e., comparing the output of FSE and the desired sequence, the delay of the desired sequence has been decided by the FSE experiment mentioned earlier (the delay with minimum MSE). The diagram for RAM training is:

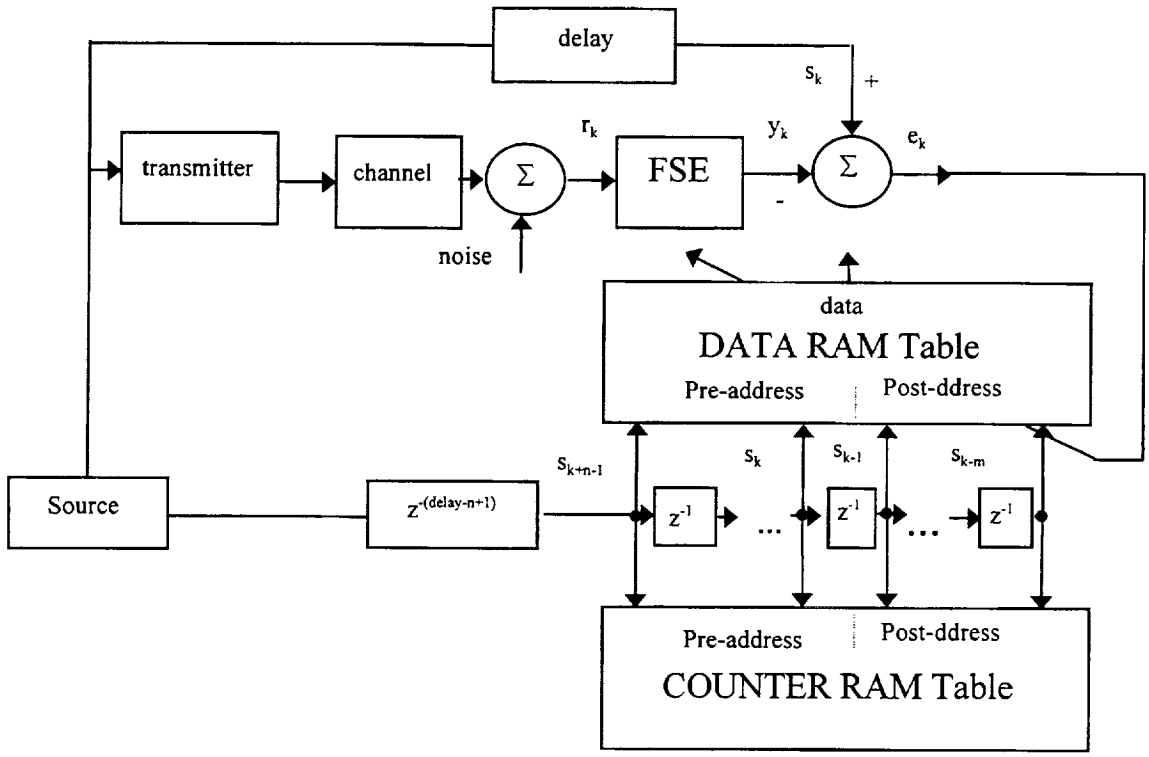


Figure 3-7 RAM training diagram

The symbols $s_{k+n+1}, \ldots, s_{k+1}, s_k, \ldots, s_{k-m}$ form the address lines. Denote the address vector by $A=[s_{k+n+1}, \ldots, s_{k+1}, s_k, \ldots, s_{k-m}]$. Denote the data content accessed by the address "A" by $RAM_data(A)$, which compensates for the nonlinearity and nonlinear ISI. Assume a symbol s_k is only interfered by its neighboring symbols $s_{k+n+1}, \ldots, s_{k+1}, s_{k-1}, \ldots, s_{k-m}$ (with n, m be finite integers), when the channel is time-invariant and noise free, after the FSE taps are fixed, the FSE output y_k is uniquely decided *by* $s_{k+n+1}, \ldots, s_{k+1}, s_k, \ldots, s_{k-m}$, $y_k=y(s_{k+n+1}, \ldots, s_{k+1}, s_k, \ldots, s_{k-m})$, let

$$RAM_data(A)=s_k-y_k= s_k-y(s_{k+n+1}, \ldots, s_{k+1}, s_k, \ldots, s_{k-m}) \quad (3.31)$$

refer to the structure in Figure 3-1, the signal before the decision device is:

$$z_k=y_k+RAM_data(A)=y_k+s_k-y_k=s_k \quad (3.32)$$

The symbol s_k is perfectly recovered. There are 16^{n+m} possible values for the address "A", which also means a RAM with size 16^{n+m} is needed to store these data. As the compensation is uniquely decided by the $n+m$ address lines, to fill up the RAM, we only need to generate all the 16^{n+m} possible address combinations, and calculate the data content.

However, two factors need to be taken into consideration. First, in a real channel, y_k is not only affected by $s_{k+n+1}, \ldots, s_{k+1}, s_k, \ldots, s_{k-m}$, symbols out of this range will also interfere s_k (with a less scale). Second, there exists additive noise in the channel. Due to them, with same $s_{k+n+1}, \ldots, s_{k+1}, s_k, \ldots, s_{k-m}$, the corresponding y_k and $RAM_data(A)=s_k-y_k$ may not be unique, they form clusters. A reasonable way to get the best data content for address "A" is to average many "s_k-y_k"s and take the centroid of the cluster.

Two RAMs with the same size 16^{m+n} and same address lines are involved, they are both set to 0 initially. One works as a data_RAM, the other works as a counter_RAM which record how many times each address has been accessed. Each time a certain address "A" is accessed, the corresponding "s_k-y_k" is accumulated to the data_RAM:

$$RAM_data(A)=RAM_data(A)+s_k-y_k \tag{3.33}$$

At the same time, the content in address A in the counter_RAM is increased by 1. When the training process finishes, divide data_RAM by counter_RAM in an address-by-address manner to get the average, and put the result back to the data_RAM.

After the look-up table is setup, the FSE coefficients, and RAM contents are both fixed, and the system works in a fixed-mode for data transmission. The diagram for fixed-mode is:

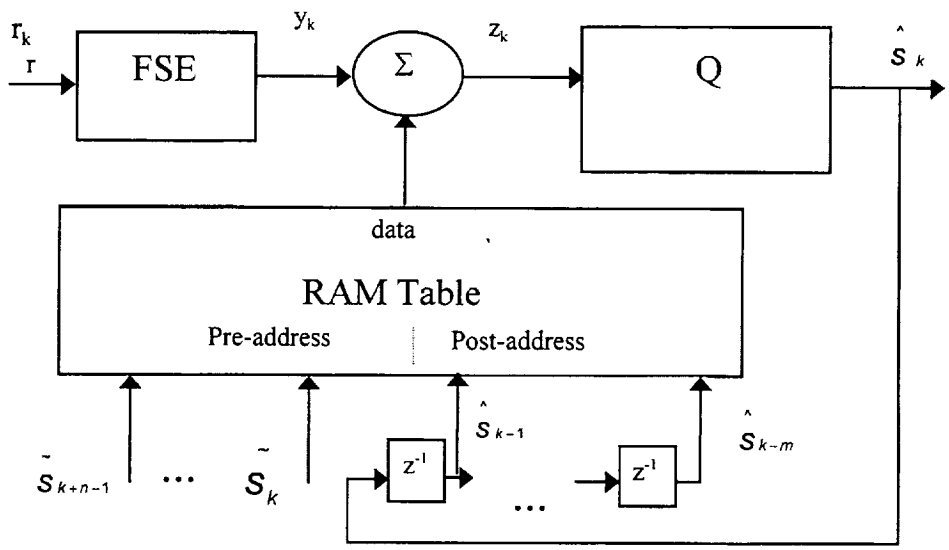


Figure 3-8 PERC block diagram

In data transmission, the receiver will decide the address lines by making local decisions on the received symbol sequence. At time k, previous local decisions $\hat{s}_{k-1}$, ..., $\hat{s}_{k-m}$ are used as post-addresses, the pre-address $\tilde{s}_k$, ..., $\tilde{s}_{k+n-1}$ are decided by testing over all possible pre-address combinations, and pick the best combination, then the corresponding testing pre-address for current symbol s_k is decided as the local decision $\hat{s}_k$ for the current symbol.

The process to select the best pre-address combination is following: After fixing the post-addresses by using previous local decision, there will be 16^n possible addresses "A" because of the 16^n possible pre-addresses combinations. The corresponding RAM contents causes $z_k=y_k+RAM_data(A)$ having up to 16^n different values, choose the z_k that is closest to the 16-QAM constellation grid point that is assumed for pre-address $\tilde{s}_k$ as the best one, i.e., calculate the distance function

$$dist_i=|z_k^{(i)}-\tilde{s}_k^{(i)}| \tag{3.34}$$

Where $i=1$ to 16^n, $\tilde{s}_k^{(i)}$ is the current symbol cursor assumed by the *i-th* pre-address test, and $z_k^{(i)}$ is the sum of y_k and the RAM content accessed by the *i-th* pre-address test. The pre-address combination with the smallest $dist_i$ is chosen to be the best pre-address combination, and the corresponding address line for $\tilde{s}_k$ is decided as the current symbol. Mathematically:

$$\hat{s}_k = \arg_i(\min|z_k^{(i)} - \tilde{s}_k^{(i)}|) \tag{3.35}$$

The performance of the algorithm will largely depend on the choice of n and m, how many neighboring symbols we are using will decide how completely we are modeling the channel. The disadvantage of this scheme is that the size RAM may be very large, and to increase n or m by 1, the RAM size will be 16 times larger. Experiments are done for the proper values of n and m. Denote the algorithm with n pre-addresses and m post-addresses by PERC(n,m).

CHAPTER 4 SIMULATION RESULTS

The simulation results of PERC algorithm are shown in this chapter. The simulation programs are written in MATLAB. The simulation process includes the training mode and the fixed-mode. The training mode sets the taps of FSE and the content in RAM, and the fixed-mode works for data transmission.

The 16-QAM constellation is normalized so that the average amplitude of the constellation points is 1. A random 16-QAM symbol sequence is generated as the transmitted symbols. The *6-th* order Butterworth analog filter is implemented by its digital approximation via bilinear transformation. And the digital filter is normalized so that the passband gain is 1, this guarantees the input to the TWTA has an average amplitude of 1, which will drive TWTA in saturation region. The received signal is the sum of the post-filter output symbol and the discrete time Additive White Guassian Noise (AWGN) component. For 16-QAM, the energy of one symbol is 4 times the energy of one bit, so the symbol-energy-noise-spectral-density-ratio (*Es/N0*) is four times the bit-energy-noise-spectral-density-ratio (*Eb/N0*), i.e., *Es/N0=4Eb/N0*.

The data at different stages of the receiver will be shown in this chapter, for *Eb/N0 14dB* case. The received data is shown in Figure 4-1 (10000 data points included):

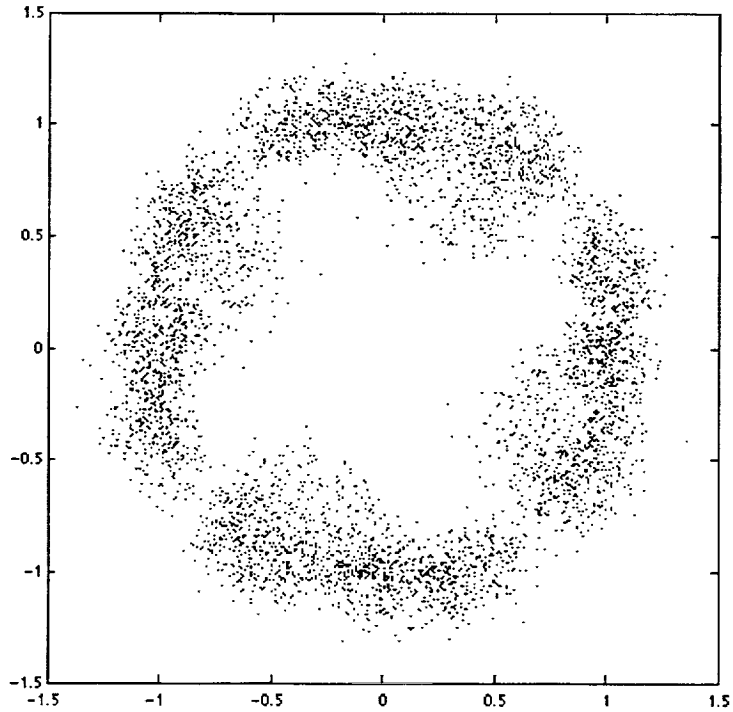


Figure 4-1 the received data

The training process for FSE is performed first, the input to FSE is at twice the symbol rate and the output of FSE is at the symbol rate. The training finishes when the MSE comes to the floor, Figure 4-2 shows the learning curve of the training process:

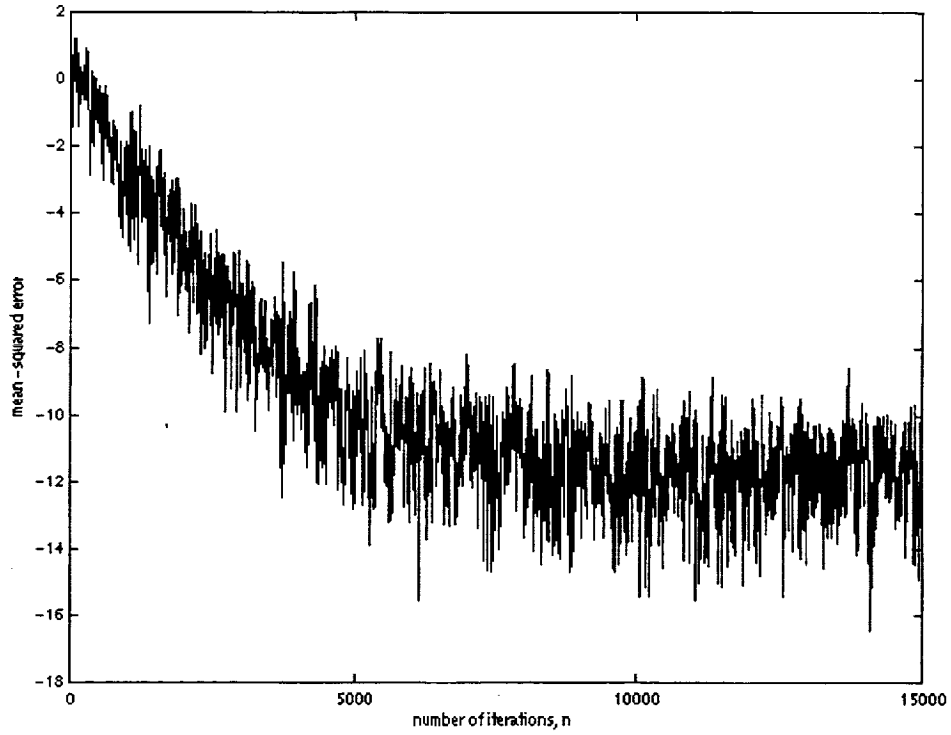


Figure 4-2 learning curve of FSE

After FSE training, the FSE taps are fixed, the training process for RAM described in Figure 3-8 begins. To do the averaging, we need to access each address of the RAM many times. For comparison, each address will be accessed 15 times in average for all PERC algorithms, i.e., for PERC(n,m), the training sequence for RAM has the length of $15 \times 16^{m+n}$.

When the training period ends, the system runs at a fixed mode, Figure 4-3 shows the output data of the FSE.

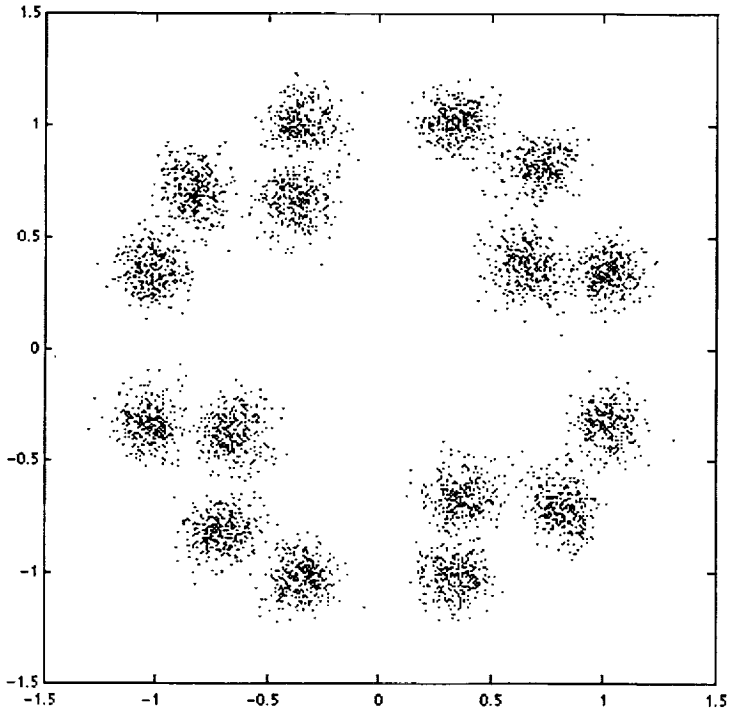
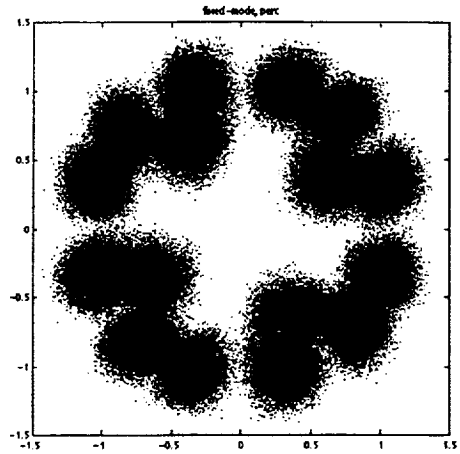
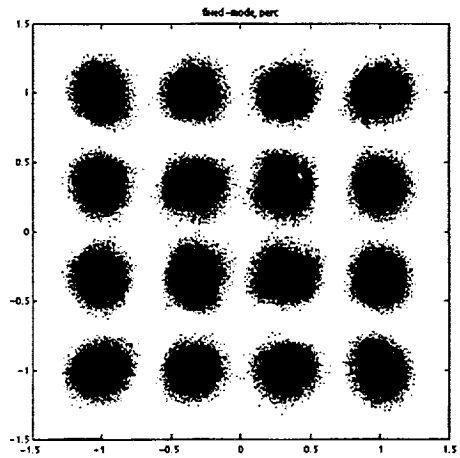


Figure 4-3 data after FSE

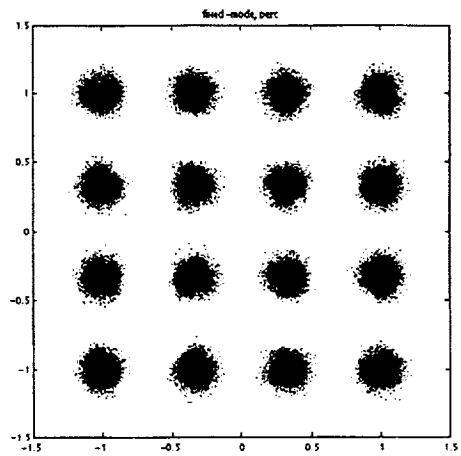
Since the FSE is a linear equalizer, it will remove (or partially remove) the linear ISI in the channel, the variance of the clusters becomes smaller. However, the center of the clusters are still not at the 16-QAM rectangular constellation grids, the nonlinear transformation and nonlinear ISI caused by the power amplifier and the filters are not removed. They will need to be further compensated by the look-up table -- RAM. The data after RAM compensation (also the input to the decision device) for PERC schemes with different *n, m* are compared in Figure 4-4 (1M data points included):



PERC(0,2)
MSE=-13.14



PERC(1,2)
MSE=-20.43



PERC(2,2)
MSE=-26.30

Figure 4-4 data before the decision device of PERC

PERC(*0,2*), which does not involve pre-address, fails to provide acceptable improvement in canceling the nonlinearity. By involving one pre-address, PERC(*1,2*) cancels the nonlinearity and recovers the 16-QAM constellation, the variance of the clustering also becomes smaller (refer to the MSE of the data points). PERC(*2,2*) includes two pre-address lines, it removes the clustering effect more. That the variance of the cluster (MSE of the data) is small does not necessarily mean that the error rate is small, because the RAM compensator may have moved an incoming symbol closer to an wrong constellation grid, the BER versus *Eb/N0* performance needs to be studied on, and the results are shown in Figure 4-5 for PERC algorithm with different n and m. The theoretical AWGN channel BER is also presented for reference.

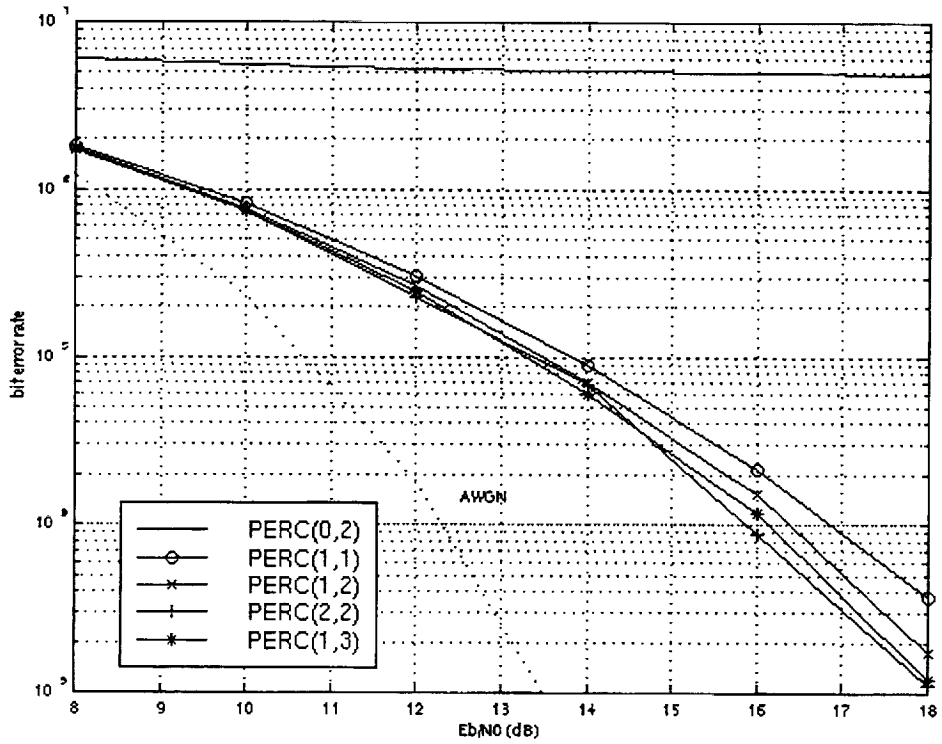


Figure 4-5 BER vs. *Eb/N0*

If a decision error happens, we assume that the wrong decided point is among the closest neighboring point of the correct one. Assume the 16-QAM is Gray coded, which means there is only 1 bit difference between the closest neighboring points on the constellation. Based on the two assumptions above, if a symbol is wrong decided, we can assume that only 1 bit is wrong, so the symbol error-rate is four times the bit symbol rate for 16-QAM. The BER in the above figure is obtained by dividing the symbol error rate by 4.

PERC(*0,2*) fails to remove the nonlinearity, and its BER are at the level around *5%* for all *Eb/N0* cases, this is because the nonlinearity is the dominating

factor for BER compared with the noise. For PERC(*1,1*), PERC(*1,2*) and PERC(*1,3*), as the post-address lines increase, the BER performance is improved. The calculation in fixed mode for them are the same, and they test over 16 possibilities to make decision on one symbol, the cost is that the RAM size is increased. PERC(*2,2*) has two pre-address lines, the RAM size is the same as the PERC(*1,3*), and PERC(2,2) needs to test over $16*16=256$ possible cases on one symbol decision. From the simulation for this channel, its performance is similar to that of PERC(*1,3*). In general, increasing pre-address by one will cause the calculation in fixed mode to increase 16 times, but the performance does not improve much, so it is not efficient to include more than 1 pre-addresses in this channel (but may be helpful for other channels).

The average times the RAM is accessed in the training mode will also affect the BER performance, the conclusion about how many access times will be proper is not included in this thesis, however, comparison for PERC(*1,2*) with different RAM access times is shown in Figure 4-6 for reference:

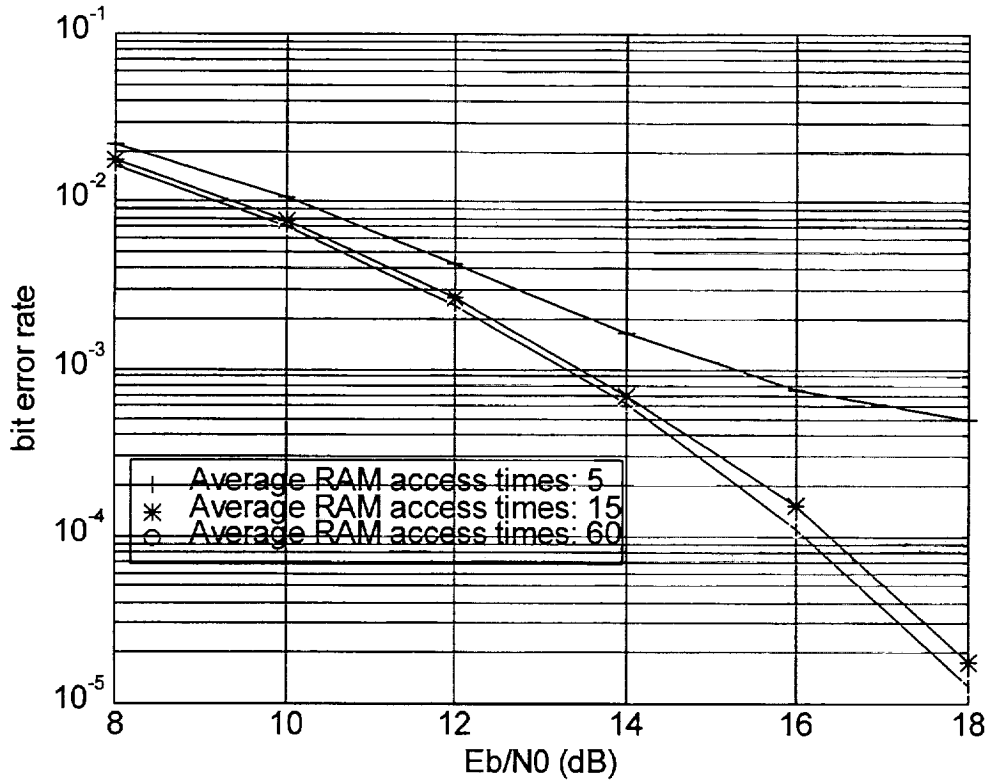


Figure 4-6 RAM access times comparison

CHAPTER 5 HARDWARE DATA COLLECTION AND SIMULATION

5.1 Hardware setup description and data collection

Hardware is setup to collect the data from both ends of the TWTA hardware channel. The parameters for the software simulation in Chapter 4 and the parameters for hardware setup described in this Chapter are set as close as possible to each other for comparison.

The hardware system is mostly composed of the Hewlett-Packard (HP) equipment and Mini-circuit components. A traveling wave tube amplifier (TWTA 8010H) is included which contributes for the nonlinearity. A pre-filter and a post-filter are connected to TWTA, which causes ISI. The TWTA works in *S*-band (1550-5200MHz), the pre-filter and post-filter should be passband filters centered at the frequency of the RF carrier, however, it is hard to find bandpass filters at this frequency range, so instead of using bandpass filters for pre- and post- filtering, lowpass filters are used at baseband, pre-filtering is done before up-converting the signal to *S*-band, and post-filtering is done after down-converting the signal to baseband. The general idea of the hardware setup is shown in the Figure 5-1.

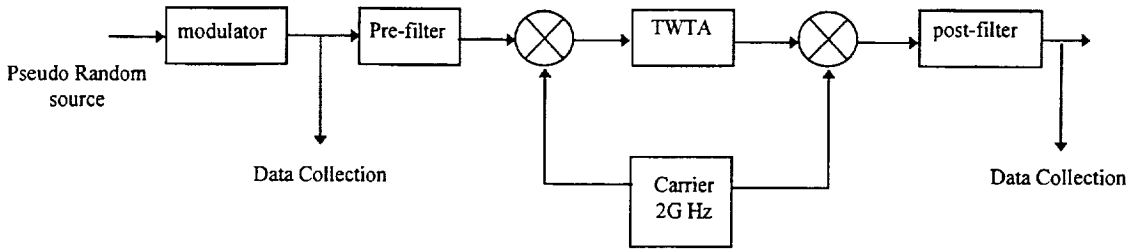


Figure 5-1 hardware setup diagram

The modulator is implemented by HP 8782B Vector Signal Generator, the Pseudo Random Bit Sequence (PRBS) can be generated by the equipment internally, the bit rate is 10Mbps. HP 8782B will implement the 16-QAM modulation, and provide the I, Q baseband sequence output. The I, Q symbol rates are 2.5M symbols per second. The HP 8981A Vector Modulation Analyzer works as a demodulator, it will extract the I, Q baseband sequence from the received waveform. The Mini-Circuit components BLP-1.9 are used as filters, with a *3dB* cutoff frequency of 2.5MHz, the symbol rate. The RF carrier is 2GHz sinusoidal waveform. The data collection is done by the LeCROY digital oscilloscope 9384. Figure 5-2 shows the hardware connection diagram:

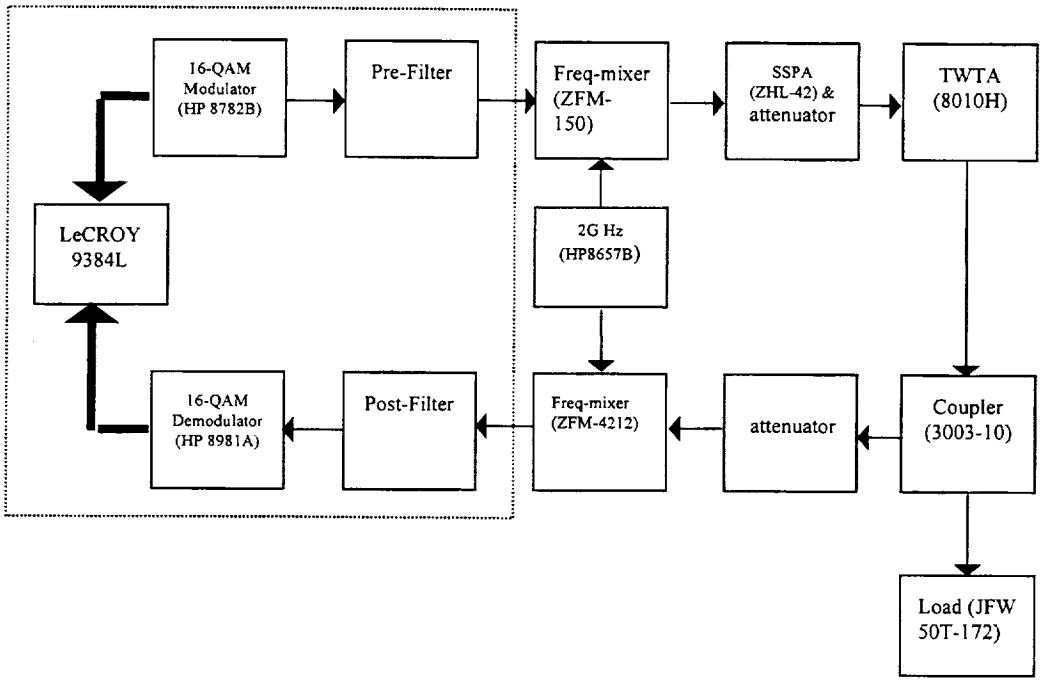
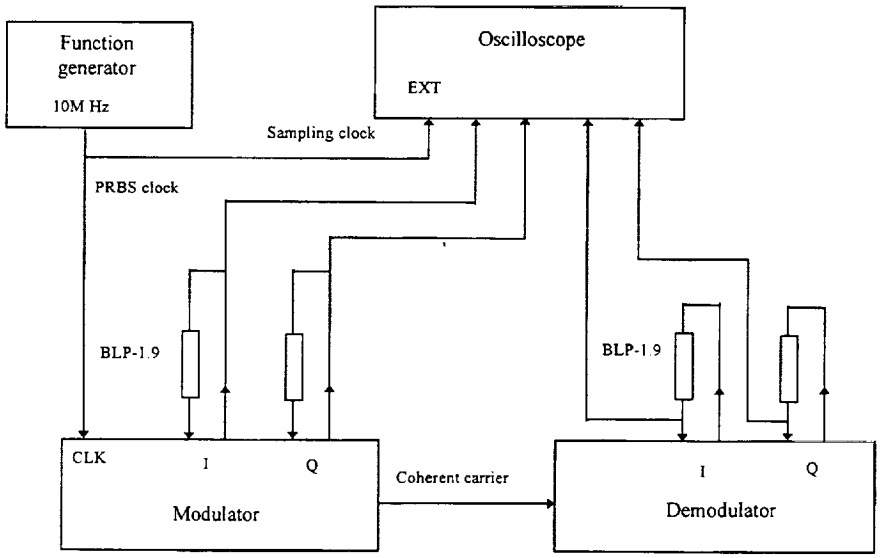


Figure 5-2 hardware connection diagram

The output of TWTA is connected to a coaxial directional coupler (Narda Corp. , Model: 3003-10 with a serial number 21540) and then to the load (N 9525 ARRA load with serial number of 858) to make the TWTA work properly. Due to the linear working range of the frequency mixer, the signal after up-converting can not drive the TWTA into the saturation region, a combination of Solid State Power Amplifier (SSPA ZHL-42) and *20dB* attenuator is used to pre-amplify the signal so that the signal will drive the TWTA in saturation region. To make the down-converter work in linear area, a *50dB* attenuator is applied to attenuate the output of the TWTA. The working range of all the components in this circuit are carefully studied and measured, to guarantee that only the TWTA works in saturation area, other devices work in linear area.

Notice that the baseband pre- and post- filtering is applied to baseband I and Q signals respectively for convenience. The details in the dotted square in Figure 5-2 is described in Figure 5-3:



BLP-1.9: Low pass filter, 1dB cutoff 1.9MHz, 3dB cutoff 2.5MHz

Figure 5-3 modulator and demodulator connection

The BLP-1.9 lowpass filters are connected as the external filters of HP 8782B and HP 8981A, and they serve as the pre- and post- filters of the TWTA. Data collection is made before the pre-filter and after the post-filter.

The coherent carrier of the modulator is 100MHz, it is fed to the demodulator as reference carrier for demodulation. The function generator provides a 10MHz clock signal, it is connected to the external clock of the HP 8782B, which triggers the pseudo-random sequence generator inside the HP 8782B Vector signal Generator. The 10MHz clock is also connected to the external sampling clock of the oscilloscope, which is the clock to record the data, so the sampling clock is four-times

the rate of the symbol, and it is synchronized with the symbol. For PERC algorithm, the sampling rate only needs to be twice the symbol rate to perform the FSE, the need for oversampling is due to the external clock frequency requirement for the oscilloscope.

The data collected at the oscilloscope is the *I, Q* baseband signals at the transmitter and the receiver side. The sampled data (1M points for each of the four data sequence) can be saved as a binary file in a floppy disk mounted on the oscilloscope, and can be converted to an ASCII file by a software "wavetran.exe" provided by LeCROY company. These are the raw data for the simulation in this Chapter.

The details about the setup of each equipment and the connections are listed in Appendix A.

5.2 Hardware generated data results

From the data collected by the oscilloscope, the transmitted data generated by HP 8782B is shown in Figure 5-4:

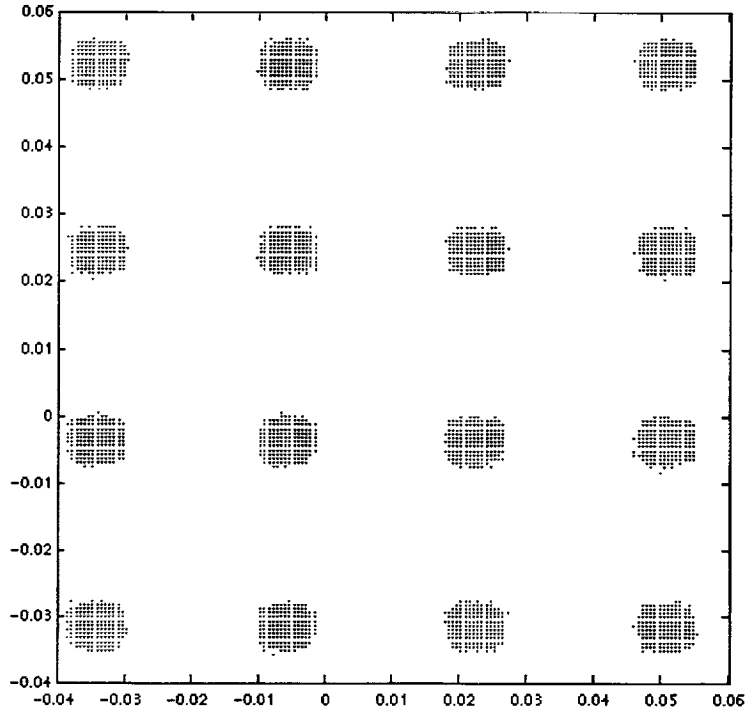


Figure 5-4 the transmitted signal (hardware)

The "noisefree" (without injecting noise) received baseband data extracted by HP 8981A is shown in Figure 5-5:

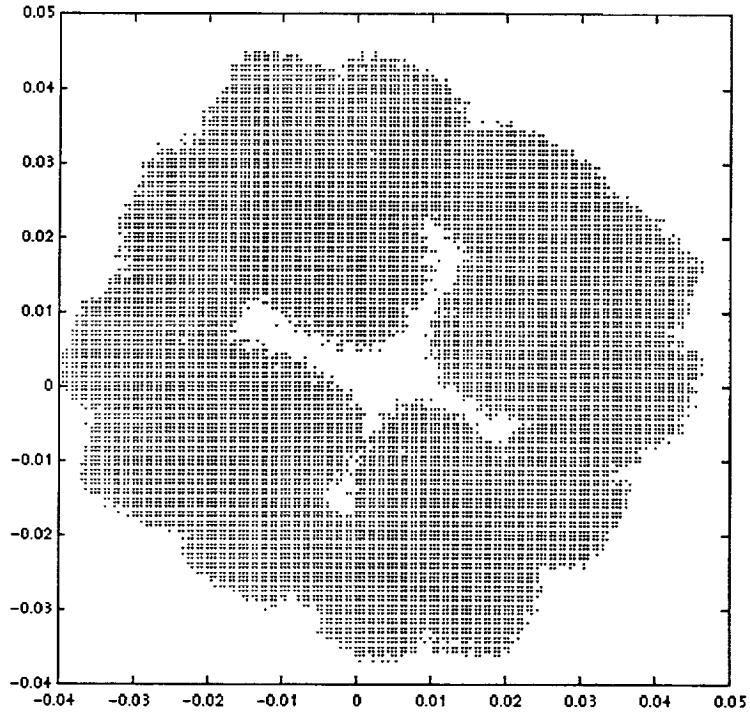


Figure 5-5 the received signal (hardware)

Practical factors need to be considered to perform the schemes described in Chapter 3 on the hardware collected data. Regarding Figure 5-4, due to the noise in the modulator and the cable, the transmitted data collected by the oscilloscope forms clusters around the constellation grids of 16-QAM. The constellation grids are gotten by taking the average of the points in each cluster. To remove the clustering, the raw data were quantized to their closest constellation grids. Then the quantized data are considered as the transmitted data and are fed to the program.

Notice that there are offsets in both the transmitted data and the received data plots, the constellations were moved to the upper-right corner, this is caused by the

oscilloscope, which is the data-collecting device. The overall vertical accuracy of the oscilloscope mentioned by the manual is 10mv, the inaccuracy causes a DC offset when recording data. Before performing other receiver algorithms, the offset is subtracted from all data points for both sent and received data. The offset is obtained by averaging all the data points. After removing the offset, the center of the data points is on the origin point, and the constellation is symmetric about origin.

Another practical factor is the phase offset in the demodulated signal, it is caused by the delay in the hardware, and its effect is a rotation on the demodulated baseband received data. From hardware measurement on the demodulator, the rotation angle is $3\pi/8$. After subtracting the offset from the received data, the rotation is centered at origin, so it carries on a linear transformation on the received data, the linear equalizer FSE can cancel the rotation effect, experiment also verified this. The length of FSE and the delay is decided by the method in Chapter 3. The length is chosen such that if continue to increase the length, the MSE will not decrease, and the delay is the one with minimum MSE. The final choice of FSE length is 20 taps for both even and odd subequalizers, and the delay is 12. Figure 5-6 shows the value of the taps.

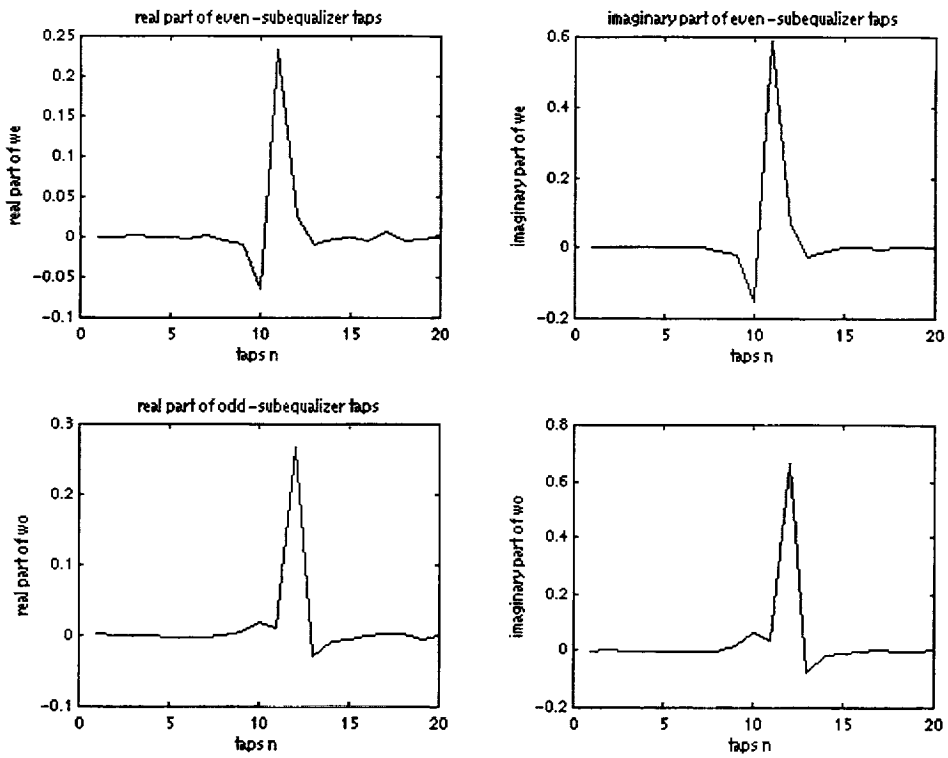


Figure 5-6 Taps of FSE

In the noisefree case, the output of FSE is given in Figure 5-7:

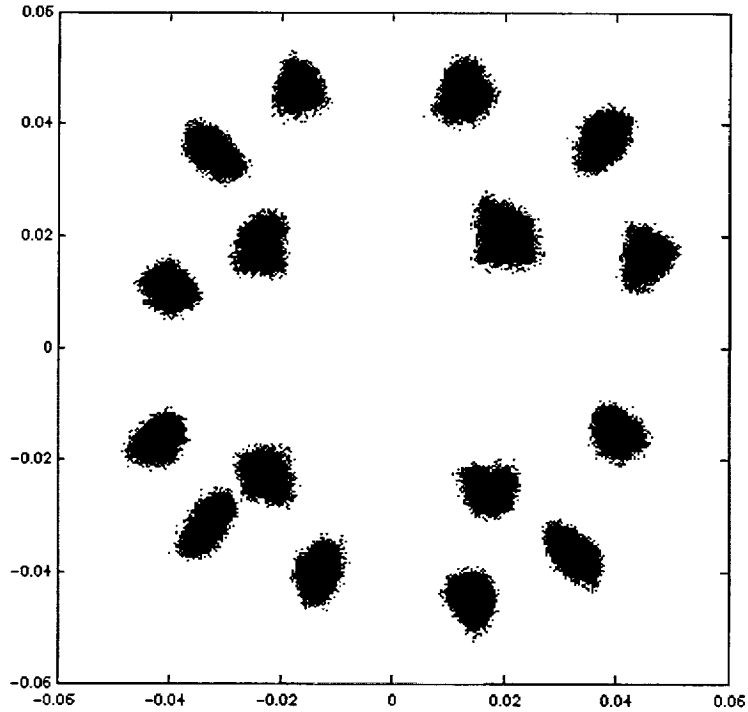


Figure 5-7 FSE output of real data simulation

Comparing with Figure 4-3, it will be interesting to notice that the hardware channel has caused some asymmetry, the reason and the compensation method for the asymmetry is not discussed in this thesis.

To study the performance in a noised environment, discrete time noise is injected in a software way, i.e., adding the discrete noise component in program instead of injecting noise in hardware. The same processes of training and fixed-mode are performed for the hardware collected data. The BER versus *Eb/N0* performance for different PERC schemes is shown in Figure 5-8:

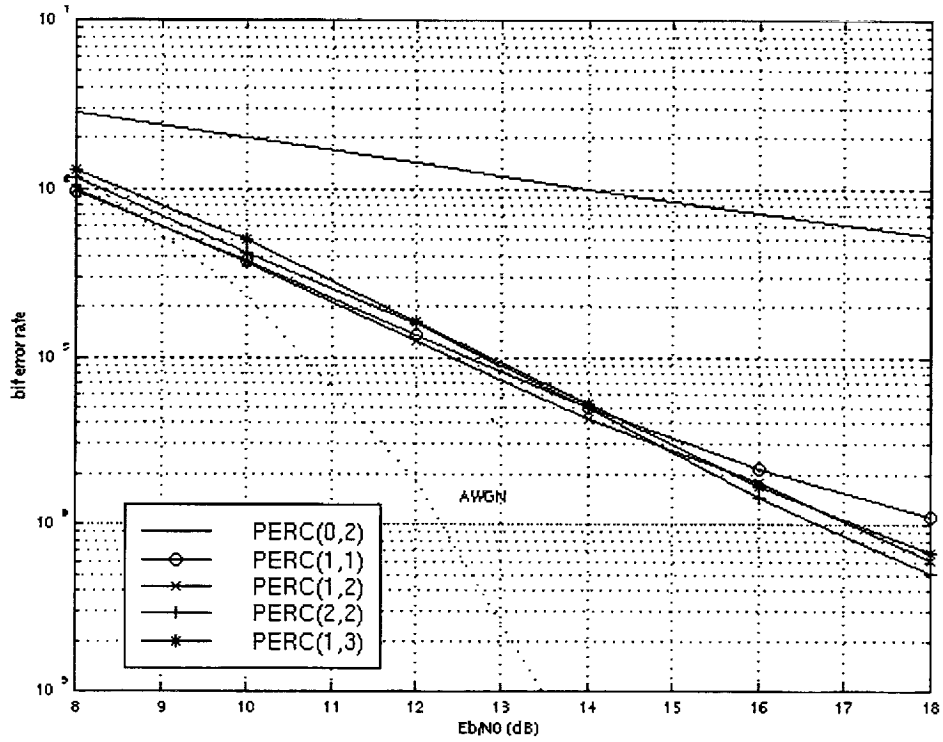


Figure 5-8 BER versus *Eb/N0* for real-data

For this real channel, similar results can be achieved as in Chapter 4, by including one pre-address, the BER versus *Eb/N0* performance is substantially increased, increasing post-address lines is helpful, but more than 1 pre-address lines is not efficient.

Refer to the results in Figure 5-8, PERC(*1,3*) and PERC(*2,2*) which have larger RAM sizes, do not show better BER performance as expected. This is because of the insufficient RAM access times. Due to the floppy disk size, a maximum of 1M data points, which is 250,000 symbols can be involved. With these data, the average

RAM access times is $250,000/16^4 = 3.8$ for PERC(*1,3*) and PERC(*2,2*). Especially in low *Eb/N0* case, the noise can not be fully averaged by access RAM so few times.

To compare the software-generated-data simulation and the real-data simulation, PERC(*1,2*) of them is shown in Figure 5-9:

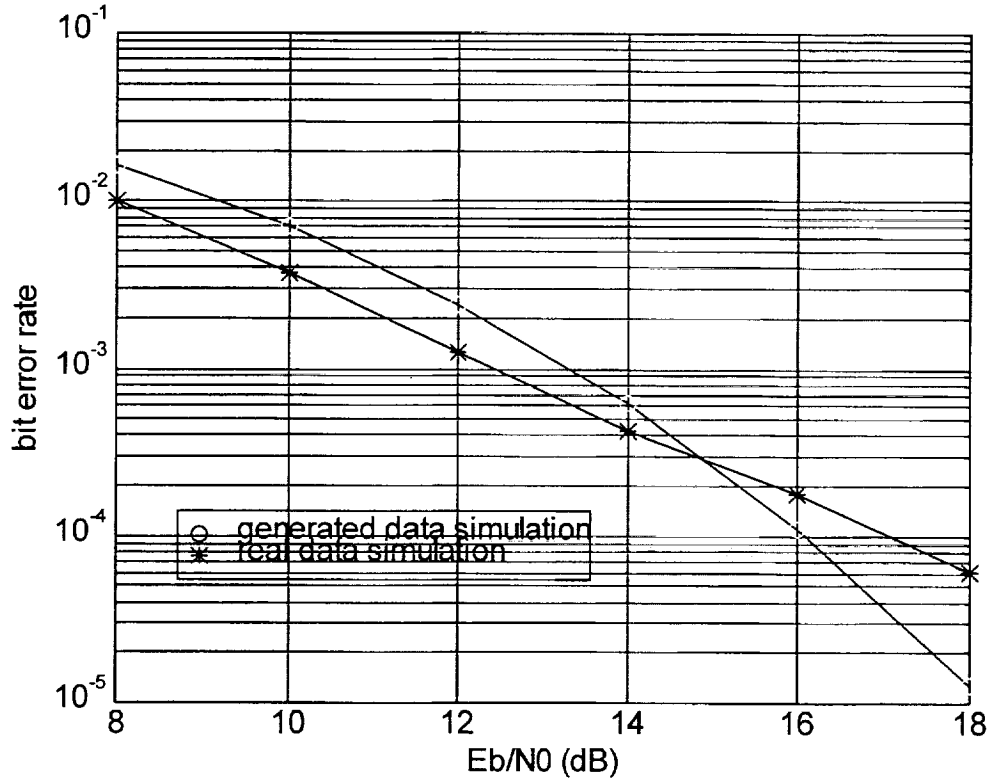


Figure 5-9 PERC(*1,2*) for software generated-data and real-data simulation

To make a fair comparison, the RAM should have the same average access times, the real-data collection includes 250,000 symbols, for PERC(*1,2*), each RAM address is accessed about 60 times, so the generated-data simulation with 60 average access times are used here for comparison. Generally, the real-data simulation shows better BER performance than the software generated data simulation, which suggests

that the nonlinearity and the ISI in the hardware is over-modeled by the model we assumed in Chapter 2. In high *Eb/N0* case, the simulation on the collected data shows a higher error-rate than the generated-data simulation, this is because of the inherent noise in the hardware equipment, cable and connectors. In high *Eb/N0* case, the noise in the hardware is comparable to the injected noise, they are both contributing to the overall noise, which is the reason why the BER versus *Eb/N0* performance is worse than the software generated data simulation with the same injected noise. When injecting higher power noise, the injected noise is dominating, and the hardware noise can be neglected.

CHAPTER 6 CONCLUSIONS AND RECOMMENDATIONS

This thesis presents a technique to transmit high order modulated (16-QAM) symbols through nonlinear memory channel. A RAM-based equalization algorithm that operates on the receiver -- the Pre-cursor Enhanced RAM-DFE Canceler (PERC) algorithm, is described and tested. The algorithm is based on the theory of Volterra model of the nonlinear system and the Finite State Machine (FSM) model of the system.

Compared with the algorithm based on the transmitter, the receiver-based methods prevent enhancing the transmitter complexity. Among the receiver-based schemes, the RAM-based algorithms are robust to the additive noise in the channel. The contribution of the PERC algorithm is that it includes pre-addresses to access the RAM and suggests a new way to decide the pre-address lines, and the state of the FSM model is more accurately defined. Compared with the results provided in [9] PERC provides better performance than the Volterra equalizers.

The PERC algorithm will compensate for both the warping and the clustering effects due to the High Power Amplifier (HPA) and the filters of the channel. PERC has lower error rate than the linear equalization schemes and algorithms that only involve post-cursors as address lines. For the channel model assumed in this thesis, involving one pre-address line (the current symbol) increases the performance greatly, while further increasing the pre-address lines costs more calculation, but does not leads to much improvement in performance. This is because the model we

assumed for the satellite channel has short memory. It may be necessary to include more pre-addresses for channels with longer memory. Only addition calculation (except the averaging operation in the training period) is needed to implement the PERC algorithm, it saves hardware, and will be suitable for real-time system.

The disadvantage of the algorithm is that for channels with long memory or for higher modulation schemes, the RAM size will be huge, and to increase the address lines by 1, it is needed to increase the size of RAM 16 times for 16-QAM, and even more if higher order modulation is considered. There will be error propagation for the PERC algorithm because a current wrong decision will be used as future post-address lines.

The work of this thesis is composed of software and hardware part. In the software simulations, the channel is modeled by a lowpass equivalent system. Comparisons for different PERC schemes are provided. Hardware is setup to collect real-world data, and practical factors are considered for the real-data simulation, satisfying results are achieved.

Some recommendations for future study includes:

- Verify the algorithm using data from actual TDRSS.
- Consider adjusting the FSE or RAM parameters in data transmission period for nonstationary channels.
- Study on a method for choosing n and m for different channel models.
- Study the proper length the training sequence.

- Study on a method to replace the FSE, and this method can decide the delay of the channel without amplifying the noise.

REFERENCES

[1] J. Namiki, "An automatically controlled predistorter for multilevel quadrature amplitude modulation", IEEE Trans. On Comm., Vol. Com-33 pp707-712, No.5 May 1983.

[2] E. Biglieri, et al, "Analysis and compensation of nonlinearities in digital transmission systems", IEEE J. On selected areas in Comm., vol. 6, pp.42-51, Jan. 1988.

[3] A. Bernardini and S. De Fina, "Performance analysis of new techniques of predistortion with memory in digital links", European trans. On Telecomm., Vol. 4, pp. 395-402, July-Aug. 1993.

[4] G. Lazzarin, S. Pupoiln and A. Sarti, "Nonlinearity compensation in digital radio systems", IEEE Trans. On Comm. Vol. 42, pp. 988-999, Feb. 1994.

[5] A. Saleh and J. Salz, "Adaptive linearization of power amplifiers in digital radio systems", The Bell Syst. Tech. J., Vol. 62, No. 4, pp.1019-1033, April 1983.

[6] M. F. Mesiya, et al. "Maximum likelihood receiver for carrier-modulated data transmission systems", IEEE Trans. Comm. Vol. Com-32 pp624-636 May 1973.

[7] D. D. Falconer, "Adaptive equalization of channel nonlinearities in QAM data transmission systems," Bell Syst. Tech. J. Vol. 57 pp. 2589-2611, Sept. 1978.

[8] S. Benedetto and E. Biglieri, "Nonlinear equalization of digital satellite channels", IEEE journal on selected areas in Comm., vol. SAC-1, No. 1, pp. 57-62, Jan. 1983.

[9] A. Gutierrez, Jr., "Equalization and detection for digital communication over nonlinear bandlimited satellite communication channels", Ph.D. Thesis, New Mexico State University, Dec. 1995.

[10] J. LeBlanc, W. Ryan and R. Kennedy, "Performance of RAM-Based Equalizers on Nonlinear Satellite Channels", 1998 DSP Workshop, Bryce Canyon UTAH (to appear), Aug. 1998.

[11] K. Fisher, et al, "An adaptive RAM-DFE for storage channels", IEEE Trans. On Comm. Pp.1559-1568, Nov. 1991.

[12] E. Biglieri et al. "Adaptive Cancellation of nonlinear Intersymbol Interference for voiceband data transmission", IEEE J. On selected areas in Comm., Vol. SAC-2, No.5. Sept. 1984.

[13] O. E. Agazzi and N. Seshadri, "On the use of tentative decisions to cancel Intersymbol interference and nonlinear distortion (with application to magnetic recording channels)," IEEE Trans. Info. Theory, Vol. 43, No. 2, March 1997.

[14] J. G. Proakis, Digital Communications, McGraw-Hill, 3rd ed., 1995.

[15] J. G. Proakis and M. Salehi, Communication Systems Engineering, Prentice Hall, 1994.

[16] A. Saleh, "Frequency-independent and frequency-dependent nonlinear models of TWT amplifiers". IEEE Trans. Comm. Pp. 1715-1720, Nov. 1981.

[17] G. Karam and H. Sari, "Improved data predistortion using intersymbol interpolation", IEEE 1989, pp. 9.5.1-9.5.6.

[18] S. Haykin, Adaptive Filter Theory, Prentice Hall, 3rd ed., 1996.

[19] J. Treichler, I. Fijalkow and C. Johnson, JR., "Fractionally spaced equalizers -- how long should they really be?", IEEE signal processing magazine, pp. 65-81, May 1996.

APPENDIX A. HARDWARE SETUP AND CONFIGURATION

Setup and connections for the Modulator (HP 8782B Vector signal Generator):

***Front panel*:**

1. The Modulation on
2. The RF output on
3. The PRBS on
4. Set Modulation to 16 QAM
5. Set the frequency to 100 MHz
6. Set the level as desired (typically, -3dBm to observe the nonlinear distortion)
7. EXT FILT on
8. PRBS CLK on.
9. The SERIAL/PRBS is connected to the external clock for the PRBS clock.
10. The RF OUTPUT is connected to "I" of Mixer ZFM 4212.

Rear panel:

1. Use the jumper cable to connect the 10MHz time base IN and OUT connectors together
2. INT/EXT Time Base be set to INT.
3. COHERENT CARRIER is connected to COHERENT CARRIER of HP 8981A at the rear panel.

4. BLP-1.9 filter is connected between EXT FILT IN and OUT of I and Q respectively.
5. Signal at OUT of I is connected to the channel 1 of LeCroy Oscilloscope 9384.
6. Signal at OUT of Q is connected to the channel 2 of LeCroy Oscilloscope 9384.

Setup and connections for the Demodulator (HP 8981A Vector Modulation Analyzer):

Front Panel:

1. Set Demodulation to 16 QAM
2. Set REF FREQ to 100MHz
3. Press DEMOD key, press MORE softkey, set DEMOD INT/EXT to INT.
4. Press DEMOD key, press MORE softkey, set EXT FILTERS ON/OFF to ON.
5. Press DEMOD key, press MORE softkey, set RF ON/OFF to ON.
6. Choose display mode by pressing CONSTL.

Rear Panel:

1. RF IN is connected to "I" of Mixer ZFM-150.
2. COHERENT CARRIER is connected to COHERENT CARRIER of HP 8782B at the rear panel.

3. BLP-1.9 filter is connected between EXT FILT IN and OUT of I and Q respectively.
4. Signal at IN of I is connected to the channel 3 of LeCroy Oscilloscope 9384.
5. Signal at IN of Q is connected to the channel 4 of LeCroy Oscilloscope 9384.

Setup and connections for LeCroy Digital Oscilloscope 9384:

1. CH1 is connected to I, OUT of the HP 8782 B.
2. CH2 is connected to Q, OUT of the HP 8782 B.
3. CH3 is connected to I, OUT of the HP 8981A.
4. CH4 is connected to Q, OUT of the HP 8981A.
5. EXT is connected to the external sampling clock.
6. Set the COUPLING of each channel to DC 50ohm.
7. Press TIMEBASE SETUP, set the sample clock to "0v", external coupling DC1Mohm, set sequence OFF, turn the knob, set the record points to "1M"
8. Insert floppy to the top of Lecroy 9384.
9. Press STOP to capture 1M points of each channel. press STORE, select "TO FLOPPY", set format to "BINARY", press "DO STORE"

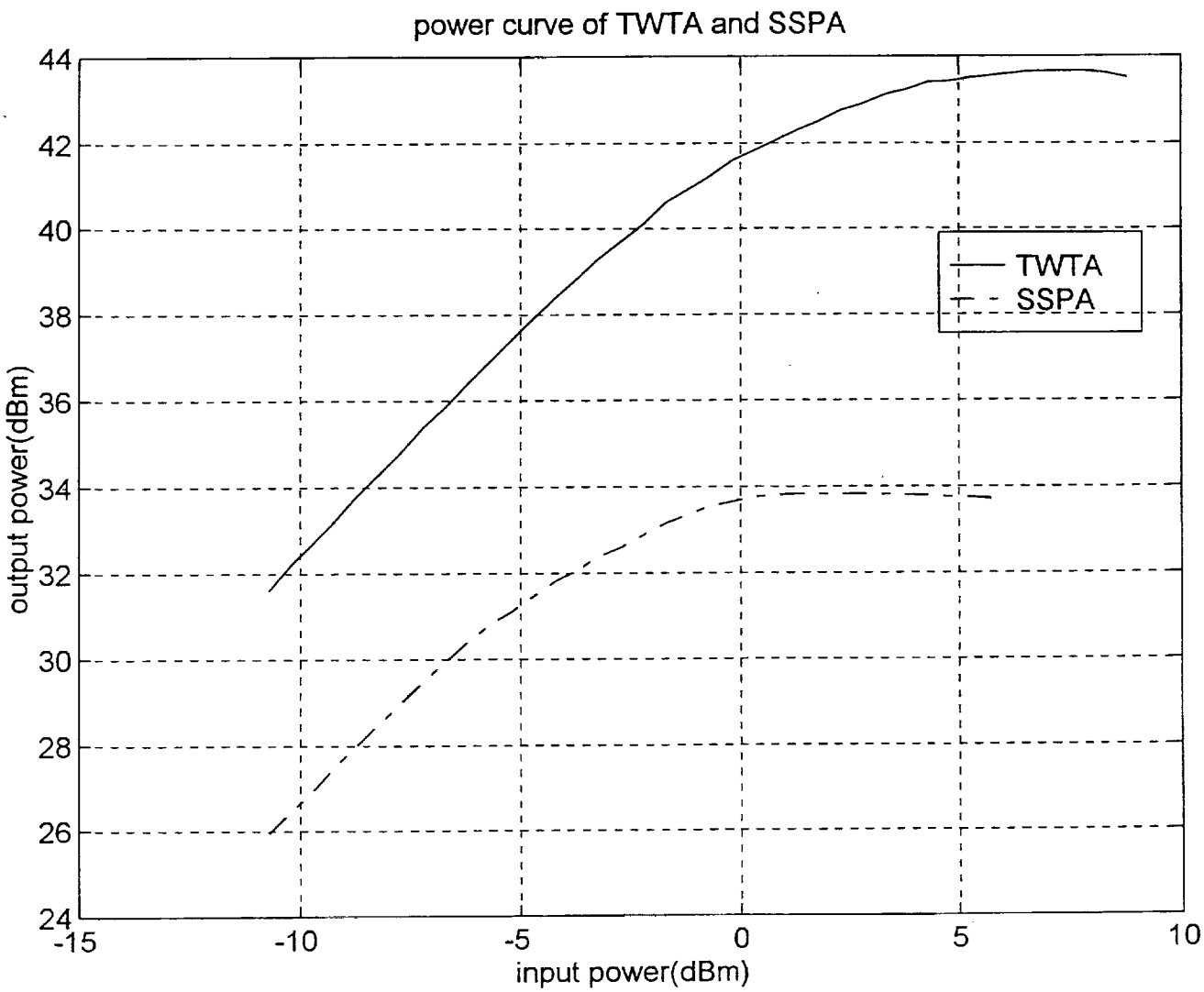
Setup for the signal generator (HP 8657) :

1. The RF output on
2. Set the frequency to 2000MHz
3. Set the amplitude to 12dBm.

Setup for the Traveling Wave Tube Amplifier (TWTA) 8010H:

1. Output of TWTA is connected to a coaxial directional coupler (Narda Corp. , Model: 3003-10 with a serial number 21540) and then to the load (N 9525 ARRA load with serial number of 858). The other output of the coupler, which has -10dB attenuation is connected to later circuit.
2. Turn on the power, (do not turn on the operatoer switch at this time.)warm up for about 10 minutes.
3. Turn on the operator switch.
4. Do not let TWTA work for more than 15 minutes continuously. Turn off the operation switch for 10 minutes, (do not need to turn off the power) then turn on and start operation again.

APPENDIX B. CHARACTERISTICS OF TWTA AND SSPA



APPENDIX C. CHARACTERISTICS OF THE FILTER

MODEL NO.	PASSBAND, MHz (loss < 1dB)	fco, MHz Nom. (loss 3dB)	STOP BAND, MHz (loss > 20dB)	STOP BAND, MHz (loss > 40dB)	VSWR Passband Typ.	VSWR Stopband Typ.	CAPD DATA (see RF-lit Design or Handbook) Page	CASE STYLE Note 8	[illegible]	PRICE $ Dollars (U.S.) Qty. (1-9)
BLP-1.9**	DC-1.9	2.5	3.4-4.7	4.7-200	1.7:1	18:1	8-34	FF55	—	34.95
BLP-2.5**	DC-2.5	2.75	3.8-5.0	5.0-200	1.7:1	18:1	8-34	FF55	—	35.95
BLP-5	DC-5	6	8-10	10-200	1.7:1	18:1	8-12	FF55	—	32.95
■ BLP-7-75	DC-7	8	11-15	15-200	1.7:1	18:1	8-35	FF55	—	33.95
BLP-10.7	DC-11	14	19-24	24-200	1.7:1	18:1	8-12	FF55	—	32.95
■ BLP-10.7-75	DC-11	14	19-24	24-200	1.7:1	18:1	8-35	FF55	—	33.95
BLP-15	DC-15	17	23-32	32-200	1.7:1	18:1	8-34	FF55	—	32.95
■ BLP-15-75	DC-15	17	23-32	32-200	1.7:1	18:1	8-35	FF55	—	33.95
BLP-21.4	DC-22	24.5	32-41	41-200	1.7:1	18:1	8-12	FF55	—	32.95
■ BLP-21.4-75	DC-22	24.5	32-41	41-200	1.7:1	18:1	8-36	FF55	—	33.95
BLP-30	DC-32	35	47-61	61-200	1.7:1	18:1	8-13	FF55	—	32.95
■ BLP-30-75	DC-32	35	47-61	61-200	1.7:1	18:1	8-36	FF55	—	33.95
BLP-50	DC-48	55	70-90	90-200	1.7:1	18:1	8-13	FF55	—	32.95
■ BLP-50-75	DC-48	55	70-90	90-200	1.7:1	18:1	8-36	FF55	—	33.95
BLP-70	DC-60	67	90-117	117-300	1.7:1	18:1	8-13	FF55	—	32.95
BLP-90	DC-81	90	121-157	157-400	1.7:1	18:1	8-14	FF55	—	32.95
BLP-100	DC-98	108	146-189	189-400	1.7:1	18:1	8-14	FF55	—	32.95
■ BLP-100-75	DC-98	108	146-189	189-400	1.7:1	18:1	8-37	FF55	—	33.95
BLP-150	DC-140	155	210-300	300-600	1.7:1	18:1	8-14	FF55	—	32.95
BLP-200	DC-190	210	290-390	390-800	1.7:1	18:1	8-15	FF55	—	32.95
BLP-250	DC-225	250	320-400	400-1200	1.7:1	18:1	8-15	FF55	—	32.95
BLP-300	DC-270	297	410-550	550-1200	1.7:1	18:1	8-15	FF55	—	32.95
BLP-450	DC-400	440	580-750	750-1800	1.7:1	18:1	8-16	FF55	—	32.95
BLP-550	DC-520	570	750-920	920-2000	1.7:1	18:1	8-16	FF55	—	32.95
BLP-600	DC-580	640	840-1120	1120-2000	1.7:1	18:1	8-16	FF55	—	32.95
■ BLP-600-75	DC-580	640	840-1120	1120-2000	1.7:1	18:1	8-37	FF55	—	33.95
BLP-750	DC-700	770	1000-1300	1300-2000	1.7:1	18:1	8-17	FF55	—	32.95
BLP-800	DC-720	800	1080-1400	1400-2000	1.7:1	18:1	8-17	FF55	—	32.95
BLP-850	DC-780	850	1100-1400	1400-2000	1.7:1	18:1	8-17	FF55	—	32.95
■ BLP-850-75	DC-750	850	1150-1490	1490-2000	1.7:1	18:1	8-37	FF55	—	33.95
BLP-1000	DC-900	990	1340-1750	1750-2000	1.7:1	18:1	8-18	FF55	—	32.95
BLP-1200	DC-1000	1200	1620-2100	2100-2500	1.7:1	18:1	8-18	FF55	—	32.95

NOTES:

** 1dB compression at +13 dBm input power

■ Denotes 75 ohm model, for coax connector models 75 ohm BNC connectors are standard.

A. General Quality Control Procedures, Environmental Specifications, Hi-Rel, MIL and TX description are given in General Information (section 0).

B. Connector types and case mounted options, case finishes are given in section 0, see "Case styles & outline drawings".

C. Prices and specifications subject to change without notice

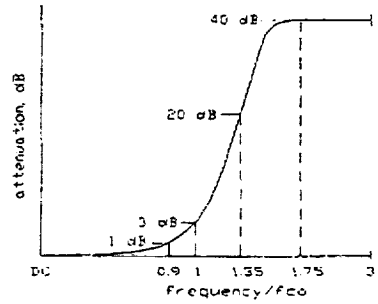
1. Absolute maximum power, voltage and current rating: 1a. RF power, 0.5 Watt

2. Models are available with male/female coax connectors, for other configurations and inter-series versions consult factory. See section 0, case styles and outline drawings.

NSN GUIDE

MCL NO.	NSN
SLP-30	5915-01-327-4692
SLP-21.4	5915-01-414-9165

LOW PASS
TYPICAL FREQUENCY RESPONSE



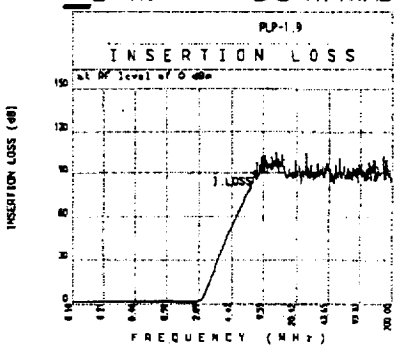
Mini-Circuits® *INTERNET* http://www.minicircuits.com
P.O. Box 350166, Brooklyn, New York 11235-0003 (718) 934-4500 Fax (718) 332-4661
Distribution Centers/ NORTH AMERICA 800-654-7949 417-335-5935 Fax 417-335-5945 EUROPE 44-1252-835094 Fax 44-1252-837010

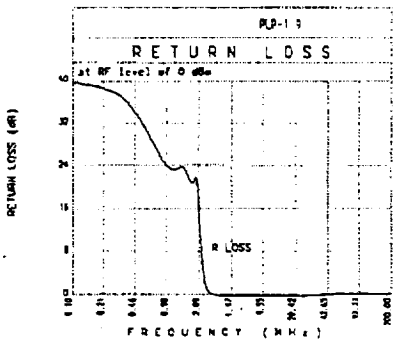
rugged, high selectivity

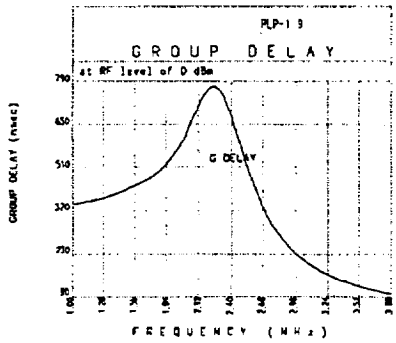
Low Pass Filters

MODEL SELECTION
(choose prefix)
P = plug in N = type N
B = BNC S = SMA

LP-1.9 DC-1.9 MHz







Freq.	I. Loss $\bar{x}$	I. Loss σ	R.L. $\bar{x}$	Freq.	G. Delay $\bar{x}$
(MHz)	(dB)	(dB)	(dB)	(MHz)	(nsec)
0.1	0.09	0.01	40.00	1.0	390.937
1.2	0.24	0.02	22.11	1.0	393.923
1.4	0.26	0.01	23.88	1.1	397.209
1.6	0.31	0.01	26.19	1.1	400.265
1.8	0.40	0.02	21.19	1.2	406.342
1.9	0.45	0.02	21.37	1.3	410.760
2.1	0.75	0.12	15.39	1.3	417.765
2.3	2.93	0.75	5.05	1.4	425.387
2.5	8.32	1.21	1.43	1.4	434.334
2.7	14.69	1.17	0.59	1.5	444.369
2.8	17.71	1.12	0.45	1.6	455.268
3.0	23.30	1.01	0.32	1.6	468.790
3.1	25.89	0.96	0.29	1.7	484.497
3.3	30.68	0.90	0.24	1.8	506.645
3.4	32.92	0.87	0.22	1.9	554.145
3.6	37.12	0.84	0.20	1.9	579.783
3.8	41.00	0.81	0.18	2.0	642.409
4.0	44.61	0.79	0.17	2.1	720.762
4.3	49.67	0.81	0.15	2.2	779.406
4.5	52.78	0.80	0.14	2.3	768.865
4.7	55.74	0.88	0.13	2.4	675.186
5.0	60.00	0.86	0.11	2.5	579.639
29.4	92.50	4.17	0.17	2.7	390.841
53.8	91.19	3.63	0.42	2.8	286.216
78.1	92.39	2.10	0.45	3.0	230.561
102.5	87.80	2.75	0.46	3.1	182.807
126.9	89.65	3.87	0.47	3.3	153.959
151.3	87.17	2.60	0.46	3.4	132.224
175.6	92.87	7.14	0.44	3.6	109.190
200.0	86.82	4.04	0.44	3.8	92.795

APPENDIX D. SIMULATION SOURCE CODE

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%sperc12.m
%PERC(1,2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


qam16;
cluster=[3.96e-3 6.28e-3 0.00995 0.01577 0.025 0.03962] ;

for index=1:6
v=randn(length(x3),1)+j*randn(length(x3),1);
v=v/sqrt(v'*v/length(v))*sqrt(cluster(index)*x3'*x3/length(x3));
y1=x3+v;
clear v;

sfsetrain;
sramtrain12;

v=randn(length(x3),1)+j*randn(length(x3),1);
v=v/sqrt(v'*v/length(v))*sqrt(cluster(index)*x3'*x3/length(x3));
y1r=x3+v;
clear v;

Lyr=length(y1r);
yer=conv(conj(we),y1r(1:2:Lyr));
yor=conv(conj(wo),y1r(2:2:Lyr));
yr=yer(1+delay:length(s))+yor(1+delay:length(s));

clear yer; clear yor;
figure
plot(yr,'.');
title('fixed-mode, after fse');

zr=zeros(length(yr),1);
z_test=zeros(16,1);
error=zeros(16,1);
sr=zeros(length(yr),1);

sr(1:2)=s(1:2);
zr(1:2)=s(1:2);

for n=3:length(yr)
  s2r=find((sr(n-2)*ones(16,1)-Q)==0)-1;
  s1r=find((sr(n-1)*ones(16,1)-Q)==0)-1;

for i=1:16
      address=s2r+s1r*16^1+(i-1)*16^2+1;
      z_test(i)=yr(n)+ram(address);
      if (i==1) best=1;end;
```

```
      error(i)=abs(z_test(i)-Q(i));
      if (error(i)<error(best)) best=i;end;
end; % test 16 defferent addresses
zr(n)=yr(n)+ram(s2r+s1r*16^1+(best-1)*16^2+1);
sr(n)=Q(best);
end; % all sequence

figure
plot(zr,'.');
title('fixed-mode, perc');

cc=0;
for i=1:length(zr)
if (s(i)==sr(i)) cc=cc+1; end;
end;
error_rate=(length(zr)-cc)/length(zr)

end;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%qam16.m
%generating 16-QAM random sequence
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
rand('seed',0);
j=sqrt(-1);
index=1;
Q=zeros(16,1);
for i=3:-2:-3
for k=-3:2:3
Q(index)=k+j*i;
index=index+1;
end;
end;
Q=Q/(sum(abs(Q))/16);

number=1e6;
sample=4;
x1=zeros(number*sample,1);
for i=1:sample:number*sample
x1(i)=Q(rem(floor(rand(1,1)*16),16)+1);
x1(i+1:i+sample-1)=ones(sample-1,1)*x1(i);
end;

[b,a]=butter(6,0.5);
x2=filter(b,a,x1);
s=x1(1:sample:length(x1));
clear x1;

ro=abs(x2);
theta=angle(x2);
clear x2;
```

```
A=1.9638*ro./(1+0.9945*ro.^2);
phi=theta+2.5293*ro.^2./(1+2.8168*ro.^2);

clear ro;
clear theta;
x2twt=A.*exp(j*phi);

clear A;
clear phi;
clear x2;

x33=filter(b,a,x2twt);
x3=x33(1:sample/2:length(x33));

clear x2twt;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%sfsetrain.m
%FSE training
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


L=15000;
Ly=length(y1);
ue=y1(1:2:Ly);
uo=y1(2:2:Ly);
clear y1;

M=5;
delay=ceil(M/2)+1;
w0e=zeros(M,1);w0o=zeros(M,1);
[e,we,wo]=nlms_fse(w0e,w0o,ue(1:L),uo(1:L),[zeros(delay,1);s(1:L-
delay)],0.001,0);

MSE=abs(e.^2);
clear e;
figure
plot(10*log10(MSE));
J=sum(MSE(length(MSE)-49:length(MSE)))/50

ye=conv(conj(we),ue);
yo=conv(conj(wo),uo);
y=ye(1+delay:length(ue))+yo(1+delay:length(ue));

figure
plot(y,'.');
title('training, the signal after FSE, before distortion');

clear ue;
clear uo;
clear ye;
clear yo;
clear y1;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%sramtrain12.m
%RAM training for PERC(1,2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


ram=zeros(16^3,1);
ram_count=zeros(16^3,1);
z=zeros(length(y),1);

for n=3:15*16^3
 s2=find((s(n-2)*ones(16,1)-Q)==0)-1;
 s1=find((s(n-1)*ones(16,1)-Q)==0)-1;
 s0=find((s(n)*ones(16,1)-Q)==0)-1;
 address=s2+s1*16^1+s0*16^2+1;
 ram(address)=ram(address)+(s(n)-y(n));
 ram_count(address)=ram_count(address)+1;
end;

for address=1:16^3
 if ram_count(address)~=0
 ram(address)=ram(address)/ram_count(address);end;
end;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%sperc22.m
%PERC(2,2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

qam16;
cluster=[3.96e-3 6.28e-3 0.00995 0.01577 0.025 0.03962] ;

for index=1:6

v=randn(length(x3),1)+j*randn(length(x3),1);
v=v/sqrt(v'*v/length(v))*sqrt(cluster(index)*x3'*x3/length(x3));
y1=x3+v;
clear v;

sfsetrain;
clear y1;
sramtrain22;

v=randn(length(x3),1)+j*randn(length(x3),1);
v=v/sqrt(v'*v/length(v))*sqrt(cluster(index)*x3'*x3/length(x3));
y1r=x3+v;
clear v;

Lyr=length(y1r);
yer=conv(conj(we),y1r(1:2:Lyr));
yor=conv(conj(wo),y1r(2:2:Lyr));
clear y1r;
```

```
yr=yer(1+delay:length(s))+yor(1+delay:length(s));



clear yer; clear yor;
figure
plot(yr,'.');
title('fixed-mode, after fse');

zr=zeros(length(yr),1);
z_test=zeros(16,16);
error=zeros(16,16);
sr=zeros(length(yr),1);

sr(1:2)=s(1:2);
zr(1:2)=s(1:2);

for n=3:length(yr)
  s2r=find((sr(n-2)*ones(16,1)-Q)==0)-1;
  s1r=find((sr(n-1)*ones(16,1)-Q)==0)-1;


for i=1:16
  for k=1:16
      address=s2r+s1r*16^1+(k-1)*16^2+(i-1)*16^3+1;
      z_test(i,k)=yr(n)+ram(address);

      if (i==1 & k==1) besti=1; bestk=1; end;
        error(i,k)=abs(z_test(i,k)-Q(k));
      if (error(i,k)<error(besti,bestk))
          besti=i; bestk=k;
          end;

end; % test 16 defferent addresses
end;

zr(n)=yr(n)+ram(s2r+s1r*16^1+(bestk-1)*16^2+(besti-1)*16^3+1);
sr(n)=Q(bestk);
end; % all sequence
figure
plot(zr,'.');
title('fixed-mode, perc');

cc=0;
for i=1:length(zr)
if (s(i)==sr(i)) cc=cc+1; end;
end;
error_rate=(length(zr)-cc)/length(zr)


end;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%sramtrain22.m
%RAM training for PERC(2,2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


ram=zeros(16^4,1);
ram_count=zeros(16^4,1);
z=zeros(length(y),1);

for n=3:15*16^4
s2=find((s(n-2)*ones(16,1)-Q)==0)-1;
s1=find((s(n-1)*ones(16,1)-Q)==0)-1;
s0=find((s(n)*ones(16,1)-Q)==0)-1;
s01=find((s(n+1)*ones(16,1)-Q)==0)-1;
address=s2+s1*16^1+s0*16^2+s01*16^3+1;

ram(address)=ram(address)+(s(n)-y(n));
ram_count(address)=ram_count(address)+1;

end;

for address=1:16^4
if ram_count(address)~=0
ram(address)=ram(address)/ram_count(address);end;
end;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%perct12.m
%real-data simulation PERC(1,2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fsetrain;
cluster=[3.96e-3 6.28e-3 0.00995 0.01577 0.025 0.03962] ;

for index=1:6

rand('seed',0);
v=randn(length(x2),1)+j*randn(length(x2),1);
v=v/sqrt(v'*v/length(v))*sqrt((cluster(index))*x2'*x2/length(x2));
y1=x2+v;
clear v;

ue=y1(1:2:length(y1));
uo=y1(2:2:length(y1));
L=15000;
M=20;
delay=12;
w0e=zeros(M,1);w0o=zeros(M,1);
[e,we,wo]=nlms_fse(w0e,w0o,ue(1:L),uo(1:L),[zeros(1,delay) s(1:L-
delay)],0.01,0);
```

```
MSE=abs(e.^2);
figure
plot(MSE);
J=sum(MSE(length(MSE)-49:length(MSE)))/50

Lr=length(ue);
ye=zeros(length(we)+length(ue)-1,1);
yo=zeros(length(wo)+length(uo)-1,1);
ye=conv(conj(we),ue);
yo=conv(conj(wo),uo);
y=ye(1+delay:length(s))+yo(1+delay:length(s));


ramtrain12t;

% fixed mode

v=randn(length(x2),1)+j*randn(length(x2),1);
v=v/sqrt(v'*v/length(v))*sqrt((cluster(index))*x2'*x2/length(x2));
y1r=x2+v;
clear v;

Lyr=length(y1r);
yer=conv(conj(we),y1r(1:2:Lyr));
yor=conv(conj(wo),y1r(2:2:Lyr));
yr=yer(1+delay:length(s))+yor(1+delay:length(s));

zr=zeros(length(yr),1);
z_test=zeros(16,1);
error=zeros(16,1);
sr=zeros(length(yr),1);

sr(1:2)=s(1:2);
zr(1:2)=s(1:2);

for n=3:length(yr)
  s2r=find((sr(n-2)*ones(16,1)-Q)==0)-1;
  s1r=find((sr(n-1)*ones(16,1)-Q)==0)-1;

for i=1:16
      address=s2r+s1r*16^1+(i-1)*16^2+1;
      z_test(i)=yr(n)+ram(address);
        if (i==1) best=1;end;
            error(i)=abs(z_test(i)-Q(i));
            if (error(i)<error(best)) best=i; end; % if
end; % test 16 defferent addresses
zr(n)=yr(n)+ram(s2r+s1r*16^1+(best-1)*16^2+1);
sr(n)=Q(best);
end; % all sequence

figure
plot(zr,'.');
title('fixed-mode, perc');
```

```
cc=0;
for i=1:length(zr)
if (s(i)==sr(i)) cc=cc+1; end;
end;
error_rate=(length(zr)-cc)/length(zr)


%%%%%%%%%%%%%%%%%%%%%%%%%
%fsetrain.m
%Real-data simulation
%load data and FSE training
%%%%%%%%%%%%%%%%%%%%%%%%%

clear
% load training sequence, sent and received.
load ./data/sc1.dat
load ./data/sc2.dat
load ./data/sc3.dat
load ./data/sc4.dat

L=length(sc1);
x1=sc1(1:2:L)+j*sc2(1:2:L);
clear sc1;
clear sc2;
Lx=length(x1);
x=x1(1:2:Lx);

x2=sc4(1:2:L)+j*sc3(1:2:L);
clear sc3;
clear sc4;

constellation;

for i=1:length(x)
s(i)=quantize(x(i),Q);
end;

center_Q=sum(Q)/length(Q);
Q=Q-center_Q;
s=s-center_Q;

center_y=sum(x2)/length(x2);
x2=x2-center_y;

rand('seed',0);
v=randn(length(x2),1)+j*randn(length(x2),1);
v=v/sqrt(v'*v/length(v))*sqrt((cluster)*x2'*x2/length(x2));
y1=x2+v;
clear v;

ue=y1(1:2:length(y1));
uo=y1(2:2:length(y1));
L=15000;
M=20;
```

```
delay=12;
w0e=zeros(M,1);w0o=zeros(M,1);
[e,we,wo]=nlms_fse(w0e,w0o,ue(1:L),uo(1:L),[zeros(1,delay) s(1:L-
delay)],0.01,0);
MSE=abs(e.^2);
figure
plot(MSE);
J=sum(MSE(length(MSE)-49:length(MSE)))/50

Lr=length(ue);
ye=zeros(length(we)+length(ue)-1,1);
yo=zeros(length(wo)+length(uo)-1,1);
ye=conv(conj(we),ue);
yo=conv(conj(wo),uo);
y=ye(1+delay:length(s))+yo(1+delay:length(s));

figure
plot(y,'.');
title('training, the signal after FSE, before distortion');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ramtrain12t.m
%real-data simulation RAM training for PERC(1,2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ram=zeros(16^3,1);
ram_count=zeros(16^3,1);
z=zeros(length(y),1);

for n=3:length(y)

s2=find((s(n-2)*ones(16,1)-Q)==0)-1;
s1=find((s(n-1)*ones(16,1)-Q)==0)-1;
s0=find((s(n)*ones(16,1)-Q)==0)-1;
address=s2+s1*16^1+s0*16^2+1;
ram(address)=ram(address)+(s(n)-y(n));
ram_count(address)=ram_count(address)+1;

end;

for address=1:16^3
if ram_count(address)~=0
ram(address)=ram(address)./ram_count(address);end;
end;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%constellation.m
%Constellation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
QAM=zeros(16,1);
count=zeros(16,1);
```

```
for i=1:length(x)

if ( (real(x(i))<=-0.02) & (imag(x(i))>0.035) ) QAM(1)=QAM(1)+x(i);
                                    count(1)=count(1)+1;end;
if ((-0.02<real(x(i)))&(real(x(i))<=0.005)&(imag(x(i))>0.035))
QAM(2)=QAM(2)+x(i);
                                    count(2)=count(2)+1;end;
if ((0.005<real(x(i)))&(real(x(i))<=0.035)&(imag(x(i))>0.035))
QAM(3)=QAM(3)+x(i);
                                    count(3)=count(3)+1;end;
if ((real(x(i))>0.035)&(imag(x(i))>0.035)) QAM(4)=QAM(4)+x(i);
                                    count(4)=count(4)+1;end;

if ((real(x(i))<=-0.02)&(0<imag(x(i)))&(imag(x(i))<=0.035))
QAM(5)=QAM(5)+x(i);
                                    count(5)=count(5)+1;end;
if ((-0.02<real(x(i)))&(real(x(i))<=0.005)&(0.005<imag(x(i)))&
(imag(x(i))<=0.035))            QAM(6)=QAM(6)+x(i);
                                count(6)=count(6)+1;end;
if ((0.005<real(x(i)))&(real(x(i))<=0.035)&(0.005<imag(x(i)))&
(imag(x(i))<=0.035))            QAM(7)=QAM(7)+x(i);
                                count(7)=count(7)+1;end;
if ((real(x(i))>0.035)&(0<imag(x(i)))&(imag(x(i))<=0.035))
QAM(8)=QAM(8)+x(i);
                                    count(8)=count(8)+1;end;

if ((real(x(i))<=-0.02)&(-0.02<imag(x(i)))&(imag(x(i))<=0.005))
QAM(9)=QAM(9)+x(i);
                                    count(9)=count(9)+1;end;
if ((-0.02<real(x(i)))&(real(x(i))<=0.005)&(-
0.02<imag(x(i)))&(imag(x(i))<=0.005)) QAM(10)=QAM(10)+x(i);
                                    count(10)=count(10)+1;end;
if ((0.005<real(x(i)))&(real(x(i))<=0.035)&(-
0.02<imag(x(i)))&(imag(x(i))<=0.005)) QAM(11)=QAM(11)+x(i);
                                    count(11)=count(11)+1;end;
if ((real(x(i))>0.035)&(-0.02<imag(x(i)))&(imag(x(i))<=0.005))
QAM(12)=QAM(12)+x(i);
                                    count(12)=count(12)+1;end;

if ((real(x(i))<=-0.02)&(imag(x(i))<=-0.02)) QAM(13)=QAM(13)+x(i);
                                    count(13)=count(13)+1;end;
if ((-0.02<real(x(i)))&(real(x(i))<=0.005)&(imag(x(i))<=-0.02))
QAM(14)=QAM(14)+x(i);
                                    count(14)=count(14)+1;end;
if ((0.005<real(x(i)))&(real(x(i))<=0.035)&(imag(x(i))<=-0.02))
QAM(15)=QAM(15)+x(i);
                                    count(15)=count(15)+1;end;
if ((real(x(i))>0.035)&(imag(x(i))<=-0.02)) QAM(16)=QAM(16)+x(i);
                                    count(16)=count(16)+1;end;
end;
Q=QAM./count;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%quantize.m
```

```
%Quantization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function d=quantize(c,Q)
for tt=1:16
dist(tt)=abs(c-Q(tt));
end;

d=Q(find((abs(dist)-min(abs(dist)))==0));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%perc22t.m
%real-data simulation PERC(2,2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fsetrain;

cluster=[3.96e-3 6.28e-3 0.00995 0.01577 0.025 0.03962] ;

for index=1:6

rand('seed',0);
v=randn(length(x2),1)+j*randn(length(x2),1);
v=v/sqrt(v'*v/length(v))*sqrt((cluster(index))*x2'*x2/length(x2));
y1=x2+v;
clear v;

ue=y1(1:2:length(y1));
uo=y1(2:2:length(y1));
L=15000;
M=20;
delay=12;
w0e=zeros(M,1);w0o=zeros(M,1);
[e,we,wo]=nlms_fse(w0e,w0o,ue(1:L),uo(1:L),[zeros(1,delay) s(1:L-
delay)],0.01,0);
MSE=abs(e.^2);
figure
plot(MSE);
J=sum(MSE(length(MSE)-49:length(MSE)))/50

Lr=length(ue);
ye=zeros(length(we)+length(ue)-1,1);
yo=zeros(length(wo)+length(uo)-1,1);
ye=conv(conj(we),ue);
yo=conv(conj(wo),uo);
y=ye(1+delay:length(s))+yo(1+delay:length(s));

ramtrain22t;

% fixed mode

v=randn(length(x2),1)+j*randn(length(x2),1);
v=v/sqrt(v'*v/length(v))*sqrt((cluster(index))*x2'*x2/length(x2));
y1r=x2+v;
```

```
Lyr=length(y1r);
yer=conv(conj(we),y1r(1:2:Lyr));
yor=conv(conj(wo),y1r(2:2:Lyr));
yr=yer(1+delay:length(s))+yor(1+delay:length(s));

zr=zeros(length(yr),1);
z_test=zeros(16,16);
error=zeros(16,16);
sr=zeros(length(yr),1);

sr(1:2)=s(1:2);
zr(1:2)=s(1:2);

for n=3:length(yr)
  s2r=find((sr(n-2)*ones(16,1)-Q)==0)-1;
  s1r=find((sr(n-1)*ones(16,1)-Q)==0)-1;

for i=1:16
  for k=1:16
      address=s2r+s1r*16^1+(k-1)*16^2+(i-1)*16^3+1;
      z_test(i,k)=yr(n)+ram(address);

      if (i==1 & k==1) besti=1; bestk=1; end;
        error(i,k)=abs(z_test(i,k)-Q(k));
      if (error(i,k)<error(besti,bestk))
          besti=i; bestk=k;
          end;

end; % test 16 defferent addresses
end;

zr(n)=yr(n)+ram(s2r+s1r*16^1+(bestk-1)*16^2+(besti-1)*16^3+1);
sr(n)=Q(bestk);
end; % all sequence

figure
plot(zr,'.');
title('fixed-mode, perc');

cc=0;
for i=1:length(zr)
if (s(i)==sr(i)) cc=cc+1; end;
end;
error_rate=(length(zr)-cc)/length(zr)

end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ramtrain22t.m
%real-data simulation RAM training for PERC(2,2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ram=zeros(16^4,1);
```

```
ram_count=zeros(16^4,1);
z=zeros(length(y),1);

for n=3:length(y)-1
s2=find((s(n-2)*ones(16,1)-Q)==0)-1;
s1=find((s(n-1)*ones(16,1)-Q)==0)-1;
s0=find((s(n)*ones(16,1)-Q)==0)-1;
s01=find((s(n+1)*ones(16,1)-Q)==0)-1;
address=s2+s1*16^1+s0*16^2+s01*16^3+1;

ram(address)=ram(address)+(s(n)-y(n));
ram_count(address)=ram_count(address)+1;
end;

for address=1:16^4
if ram_count(address)~=0
ram(address)=ram(address)./ram_count(address);end;
end;
```